UNIVERSIDAD DE SEVILLA

Departamento de Lenguajes y Sistemas Informáticos

AVANCES EN LA TOMA DE DECISIONES EN PROYECTOS DE DESARROLLO DE SOFTWARE

TESIS DOCTORAL

Javier Aroba Páez

Sevilla, Noviembre de 2003

UNIVERSIDAD de SEVILLA

TESIS DOCTORAL

AVANCES EN LA TOMA DE DECISIONES EN PROYECTOS DE DESARROLLO DE SOFTWARE

Por

D. Javier Aroba Páez

Licenciado en Informática por la Facultad de Informática de la Universidad de Sevilla

Presentada en el

Departamento de Lenguajes y Sistemas Informáticos

de la

Universidad de Sevilla

Para la obtención del

Grado de Doctor en Informática

Directores

Dra. Dª. Isabel Ramos Román

Dr. D. José Cristóbal Riquelme Santos

Doña Isabel Ramos Román y Don José Cristóbal Riquelme Santos, Profesores Titulares de Universidad, adscritos al área de Lenguajes y Sistemas Informáticos,

CERTIFICAN QUE:

Don Javier Aroba Páez, Licenciado en Informática por la Universidad de Sevilla, ha realizado bajo nuestra supervisión el trabajo de investigación titulado

Avances en la toma de decisiones en Proyectos de Desarrollo de Software

Una vez revisado, autorizamos la presentación del mismo como Tesis Doctoral en la Universidad de Sevilla y estimamos oportuna su presentación al tribunal que habrá de valorarlo.

Fdo.: Isabel Ramos Román
Profesora Titular de Universidad
Lenguajes y Sistemas Informáticos

Fdo.: José C. Riquelme Santos
Profesor Titular de Universidad
Lenguajes y Sistemas Informáticos

Sevilla, Noviembre 2003

Tesis Doctoral parcialmente subvencionada por la Comisión Interministerial de Ciencia y Tecnología con el Proyecto TIC2001-1143-C03-02.

MINISTERIO
DE CIENCIA
Y TECNOLOGÍA

CICYT
TIC2001-1143-C03-02

Agradecimientos

Quiero agradecer a todos los que han participado directa o indirectamente en el desarrollo de esta tesis doctoral. En este sentido quiero hacer mención especial:

- a la profesora Isabel Ramos Román por su gran apoyo en el desarrollo del trabajo que se presenta, su constante interés, su generosa disposición y sus innumerables sugerencias.

- al profesor José C. Riquelme Santos por la confianza depositada, su entera disposición en todo momento y sus enriquecedoras aportaciones en los momentos más difíciles.

- al profesor Miguel Toro Bonilla por la confianza mostrada en la línea de investigación que se presenta y sus útiles comentarios.

- al profesor Javier Dolado Cosín por el apoyo en el trabajo que se presenta y sus experimentados consejos.

- a mis compañeros de investigación, por las enriquecedoras discusiones en sobremesas y viajes, durante los años que ha durado este trabajo.

- a mis compañeros de Departamento Manuel J. Redondo González y José Manuel Andújar Márquez, por su gran apoyo tanto en el terreno profesional como personal.

- a mi familia, por su constante aliento en todo momento, la confianza depositada en mi, así como su enorme apoyo en todos los terrenos.

ÍNDICE DE CONTENIDOS

Capítulo 5: Aplicación de PreFuRGe a PDS

Capítulo 6: Aplicación de técnicas tradicionales de predicción

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Resumen

La aparición en los últimos años de los modelos dinámicos para *Proyectos de Desarrollo de Software (PDS)* y de potentes entornos de simulación, ha facilitado la generación de los llamados *Simuladores de Proyectos de Software (SPS)* que permiten estudiar el comportamiento de estos proyectos. Los SPS utilizados permiten, entre otras cosas, la creación automática de bases de datos formadas por un gran número de atributos, en las que se almacena toda la información que se genera. Estas bases de datos por sí solas no son de mucha utilidad a la hora de tomar decisiones sobre el proyecto, ya que es muy difícil extraer de éstas información cualitativa. Por ello, es necesaria la investigación en nuevas técnicas de análisis de datos capaces de proporcionar información útil, a priori, desconocida.

El proceso para extracción de conocimiento en bases de datos se conoce de manera general como *Knowledge Data Discovery (KDD)*, y en concreto las técnicas fundamentales que lo soportan se denominan *Minería de Datos.* Dada la variedad de métodos que existen, es importante analizar cuáles serán los más apropiados para poder obtener información a partir de las bases de datos generadas por los SPS; en este sentido, se realizará un estudio y análisis de las diferentes técnicas de Minería de Datos que permitan obtener información *cualitativa* de las bases de datos *cuantitativas*, para poder elegir la más adecuada y poder contrastar posteriormente los resultados obtenidos.

El primer objetivo de esta tesis es aportar nuevas contribuciones en el campo de la toma de decisiones en PDS, que permitan obtener información de tipo *cualitativo* y de manera automática, sobre el funcionamiento de tales proyectos, mejorando la eficiencia y la precisión obtenidas con otras técnicas; de las múltiples existentes, y dada la especial naturaleza de las bases de datos a tratar, son de especial interés en esta investigación las técnicas de *Fuzzy Clustering*, por considerarlas las más adecuadas para obtener el conocimiento cualitativo deseado. En este sentido, se desarrollará una herramienta de Minería de Datos, que se pueda aplicar de manera práctica a datos de proyectos reales, y que genere la información *cualitativa* deseada, de forma que el director del proyecto pueda tomar decisiones lo más acertadas posible.

El segundo objetivo, es el análisis comparativo e implementación de diferentes técnicas de *estimación*, que permitan predecir el comportamiento de las distintas variables del proyecto a partir de nuevos valores en los atributos sin necesidad de utilizar los SPS; finalmente se realizará una comparación de los resultados obtenidos.

Capítulo 1

Introducción

1.1 Planteamiento

En la actualidad, la facilidad para almacenar información está produciendo un fenómeno que puede resultar contradictorio. Lo ideal y deseable sería que el crecimiento del volumen de datos almacenados conllevara un aumento proporcional del conocimiento que se pueda obtener a partir de ellos; pero esto no siempre es así por varias razones. Una de ellas, en la que se centra este trabajo, es la existencia de información oculta entre los datos que no se puede obtener con los métodos clásicos. Para resolver este problema, en los últimos años han surgido una serie de técnicas que facilitan el proceso avanzado de los datos y permiten realizar un análisis en profundidad de los mismos de forma automática.

El proceso completo de extracción de conocimiento a partir de bases de datos se conoce como *KDD* (*Knowledge Data Discovery*). Este proceso consta de varias etapas que transcurren desde la preparación de los datos hasta la presentación de los resultados obtenidos. Dentro de estas etapas se encuentra la Minería de Datos (*Data Mining*), que puede definirse como el proceso no trivial de extracción de información implícita, previamente desconocida y potencialmente útil, a partir de los datos.

Hasta hace poco tiempo, el conocimiento que se obtenía de los datos consistía básicamente en información fácilmente recuperable mediante algún lenguaje de consultas (*SQL*, *QBE*). Más adelante, surgieron las herramientas de análisis multidimensional (*OLAP, On-line Analytical Processing*), que permiten analizar los datos considerando unas variables en relación con otras y permitiendo realizar el análisis desde distintos puntos de vista. Ambas técnicas requieren que un usuario realice la consulta, es decir, el análisis de los datos está condicionado a la hipótesis planteada.

En cambio, mediante la Minería de Datos, es el propio proceso el que descubre nuevas relaciones o patrones obteniendo modelos con los que se podrán tomar posteriores decisiones que enriquezcan la actividad en la que se esté aplicando.

La mayor parte del conocimiento que se necesita obtener de los datos se puede recuperar mediante consultas sencillas o técnicas estadísticas, mientras que otra parte de ese conocimiento se encuentra oculto y requiere, por tanto, el uso de técnicas más avanzadas de análisis para su recuperación. En porcentaje, se puede decir que el conocimiento oculto es menor que el conocimiento superficial, pero resulta de vital importancia puesto que generalmente descubre comportamientos inesperados. De ahí, que se estén investigando y perfeccionando continuamente estas técnicas.

1.2 Objetivos de la Tesis

El primer objetivo es aportar nuevas contribuciones en el campo de la *toma de decisiones* en proyectos de desarrollo de software, que permitan obtener información de tipo *cualitativo* sobre el funcionamiento de Proyectos de Desarrollo de Software (PDS), que mejoren la eficiencia y la precisión de los resultados obtenidos con otras técnicas. Para ello, se realizará un estudio y análisis de las diferentes técnicas de Minería de Datos que permitan obtener información de tipo *cualitativo* a partir de las bases de datos *cuantitativas* que se obtienen como resultado de las simulaciones de modelos para PDS. Como resultado final de este objetivo, se desarrollará una *herramienta de Minería de Datos* basada en las técnicas propuestas, que se pueda aplicar de manera práctica a datos de proyectos reales, y que genere la información *cualitativa* deseada sobre el comportamiento de los PDS, de forma que el director del proyecto pueda tomar decisiones lo más acertadas posible.

El segundo objetivo es el análisis comparativo e implementación de técnicas de predicción que puedan ser aplicadas en el ámbito de los PDS, de forma que, sin necesidad de utilizar los SPS, se pueda predecir el comportamiento de las distintas variables del proyecto (coste, tiempo, calidad, etc.), a partir de nuevos valores de los parámetros (dedicación media, retraso en la contratación, etc.) del PDS. Para la consecución de este objetivo se recurrirá principalmente a:

- Técnicas estadísticas: Basadas en los modelos de Regresión Multivariante, lineales y no lineales. Estos modelos de regresión se obtendrán con métodos numéricos, utilizando herramientas de análisis estadístico.
- Técnica algorítmica de Aprendizaje Automático: Basado en la técnica de los k-Vecinos más cercanos, se desarrollará un algoritmo que proporcione unas predicciones lo más fiables posibles.
- Programación lineal: Aplicación del método Simplex, a partir de los modelos lineales obtenidos con técnicas estadísticas.

Se han seleccionado estas técnicas para abordar el segundo objetivo, por dos razones fundamentalmente:

- Son técnicas usualmente utilizadas para abordar problemas como los que se intentan analizar en este objetivo.
- Se consideran las más adecuadas para trabajar con el tipo de bases de datos numéricas que se obtendrán tras la simulación de los modelos para PDS.

1.3 Aportaciones de la tesis

En una primera etapa se ha realizado un análisis de las diferentes técnicas de Minería de Datos que se pudieran aplicadas a bases de datos cuantitativas de proyectos de desarrollo de software. Los primeros pasos en este sentido han consistido en el estudio y aplicación de las siguientes herramientas de Minería de Datos:

- La Herramienta C4.5 [Quinl93]: Se trata de un clasificador ortogonal que construye árboles de decisión, en el ámbito del aprendizaje supervisado, a partir de los que se pueden crear reglas lógicas recorriéndolos desde la raíz hasta las hojas. La información que se obtiene con esta herramienta es de más utilidad, para el gestor del proyecto que la analiza, que la que proporcionan las propias bases de datos, aunque su interpretación es algo compleja dado el elevado número de reglas que se pueden obtener.

- La Herramienta COGITO [Rique97], [Aguil97], [Aguil98]: Esta formada por una familia de algoritmos que permite obtener un modelo de reglas capaz de clasificar a un conjunto de datos con el mínimo error posible en el contexto del aprendizaje supervisado. A diferencia de C4.5, no divide el espacio por un atributo, sino que extrae secuencialmente una región del espacio. La información obtenida en este caso es mucho más útil para el gestor que la

obtenida con C4.5, no obstante su interpretación sigue siendo algo compleja dado el carácter semicualitativo de las reglas obtenidas.

A raíz de los resultados obtenidos y los problemas detectados, se han decidido analizar técnicas de aprendizaje no supervisado (capítulo 3, apartado 2.4), y en concreto de *Clustering* (capítulo 3, apartado 3.4). Con el fin de poder obtener una información sobre el funcionamiento del proyecto de desarrollo de software lo más cercana posible al modo de razonar humano, se ha decidido utilizar *lógica borrosa* en la aplicación del clustering. En este caso estamos hablando, por tanto, de una técnica de clustering borroso. En este sentido se han estudiado los trabajos de Sugeno y Yasukawa [Sugen93] en el campo del clustering borroso, corrigiendo los problemas que surgen al aplicarlos a la toma de decisiones en PDS (capítulo 4, apartado 2.3).

Por ello, se ha propuesto un conjunto de algoritmos de *clustering borroso*, que basados en la metodología de Sugeno y Yasukawa, y utilizando técnicas estadísticas y de visualización, permite obtener información cualitativa en forma de reglas lingüísticas y gráfica sobre el funcionamiento de PDS, trabajando con bases de datos formadas por múltiples atributos de entrada y variables de salida (capítulo 5, apartado 1.1) dependientes de los atributos de entrada, simultáneamente. Para poner en práctica estos algoritmos, se ha desarrollado la herramienta PreFuRGe, que proporciona una información de tipo cualitativo muy fácil de evaluar por el director del proyecto, sin necesidad de tener conocimientos previos de Minería de Datos.

Con el fin de poder realizar predicciones sobre el valor de las variables de los Proyectos, se han analizado y desarrollado diversas técnicas de predicción que pudieran ser aplicadas en el ámbito de los proyectos de desarrollo de software, de forma que a partir de nuevos valores de los parámetros del proyecto, y sin necesidad de simular el proyecto, se pudiera predecir el comportamiento de las distintas variables del mismo, con el menor error posible. Las técnicas de predicción que se han utilizado y desarrollado, son:

1. Modelos de regresión multivariante (Lineales y no Lineales)
2. Algoritmo wk-NN (weighted k-Nearest Neighbor)
3. Programación lineal (Método Simplex)

éstas son técnicas de predicción tradicionales, que se desean aplicar en el ámbito de los proyectos de desarrollo de software, para ver las posibles aportaciones que pueden proporcionar en el campo de la toma de decisiones. Con los resultados que se obtengan con cada una de ellas, se realizará un análisis comparativo para determinar cuál es la técnica que proporciona los mejores resultados, y que por tanto puede ser utilizada con mayor fiabilidad.

Fruto del trabajo realizado durante la realización de esta tesis es la participación en los siguientes congresos nacionales e internacionales:

- J. Aroba, I. Ramos, J.C. Riquelme. *Decision making in software projects using fuzzy clustering algorithms*. PROSIM (Workshop on Software Process Simulation and Modeling). Actas del Congreso, section: ”Combining learning models with simulation”, Imperial College, Londres, 2000.

- I. Ramos, J. Aroba, J.C. Riquelme. *Mejoras en la toma de decisiones en proyectos de desarrollo de software: Aplicación de técnicas de lógica borrosa*. V JISBD (Jornadas de Ingeniería de Software y Bases de Datos). Pag. 153. Valladolid, 2000.

- I. Ramos, J.C. Riquelme, J. Aguilar, M. Ruiz, J. Aroba. *Aplication of machine learning techniques to software project management.* V CIIP (Congreso Internacional de Ingeniería de Proyectos). Pag. 150. Lérida, 2000.

- I. Ramos, J.C. Riquelme, J. Aguilar, M. Ruiz, J. Aroba. *Toma de decisiones en proyectos de desarrollo de software*. IV CINTE 2000. (IV Jornadas Científicas Andaluzas en Tecnologías de la Información). Pag. 196. Cádiz, 2000.

- I. Ramos, J. Aroba, J.C. Riquelme. *Aplication of machine learning techniques to software project management*. ICEIS 2001 (III International Conference on Enterprise Information Systems). Pag. 433. Setúbal (Portugal), 2001

- J. Aroba, I. Ramos, J.C. Riquelme. *Mejoras en la toma de decisiones en proyectos de desarrollo de software*. IC-AI 2001 (International Conference on Artificial Intelligence). Pag. 354. Las Vegas (Estados Unidos), 2001.

- J. Aroba, I. Ramos, J.C. Riquelme. *Forecasting in Software Development Projects*. SERP 2003 (Software Engineering Research and Practice). Pendiente de celebración (Junio). Las Vegas (Estados Unidos), 2003.

- I. Ramos, M. Ruiz, J. Aroba, P. Pérez. Simulación *de Proyectos de desarrollo de software durante la ejecución*. C.I.T. Información tecnológica. Revista Internacional *Volumen*: 12 Nº2. pp. 151-160. ISSN 0716-8756. Marzo 2001.

1.4 Estructura de la tesis

El presente documento está dividido en los siguientes capítulos:

- **Capítulo 2:** *Métodos tradicionales de estimación:* En este capítulo se hace una revisión del estado del arte en cuanto al proceso de estimación y toma de decisiones en Ingeniería del Software. Además se aborda de forma más concreta la toma de decisiones en los proyectos de desarrollo de software, que será el problema concreto que se trata en esta investigación.

- **Capítulo 3:** *La Minería de Datos en la de toma de decisiones*: Aquí se presenta una descripción del proceso general de Minería de Datos, describiendo de forma detallada aquellas técnicas que se van a utilizar en la investigación, tanto en las tareas de Knowledge Discovery in Databases (KDD), como en las de predicción de resultados.

- **Capítulo 4:** *PreFuRGe: Predictive Fuzzy Rules Generator:* En este capítulo se hace un análisis de la herramienta de Minería de Datos propuesta en la tesis (PreFuRGe), describiendo las principales características de ésta y los algoritmos que la forman.

- **Capítulo 5:** *Aplicación de PreFuRGe a PDS*: Se presentan los resultados obtenidos tras la aplicación de la herramienta propuesta a diversas bases de datos de proyectos de desarrollo de software; estos resultados son comparados con los obtenidos con otras herramientas de Minería de Datos que intentan obtener el mismo tipo de información.

- **Capítulo 6:** *Aplicación de técnicas tradicionales de predicción*: En este capítulo se pretende comprobar que aportan las técnicas estadísticas de predicción más tradicionales, como son la regresión lineal y no lineal o la programación lineal, en el ámbito de la toma de decisiones en proyectos de desarrollo de software. Además de éstas, se ha desarrollado un algoritmo basado en los k-vecinos más cercanos ponderados, cuyos resultados se compararán con los obtenidos con las demás técnicas.

- **Capítulo 7:** *Conclusiones y futuras líneas de trabajo*: Se analizan las conclusiones obtenidas en los distintos capítulos y se presentan las futuras líneas de trabajo que pueden derivarse de esta investigación.

- **Apéndices:**

  - A) *Lógica Borrosa:* En este apéndice se describe de forma general qué es la lógica borrosa, detallando sus principales aplicaciones y los principales conceptos que rodean a este tema; el sentido de este apartado estriba en que la herramienta propuesta, PreFuRGe, está basada en algoritmos de clustering borroso.

  - B) *Modelo de estimación de coste: COCOMO II*: Se realiza una descripción general del conocido modelo de estimación COCOMO II, y se compara con su predecesor (COCOMO81); ambas versiones del modelo COCOMO pretenden obtener modelos de estimación utilizando técnicas de regresión (utilizadas en esta investigación en el capítulo 6), basado en datos históricos.

  - C) *Glosario de términos*: Se muestra una relación de términos relacionados con los distintos temas abordados en la investigación.

Capítulo 2

Métodos Tradicionales de Estimación

2.1 Introducción

En líneas generales, *estimar* consiste en determinar el valor de una variable desconocida a partir de otras conocidas, o de una pequeña cantidad de valores conocidos de esa misma variable. La estimación forma parte de la inferencia estadística. El razonamiento por inferencia va de lo particular a lo general, de lo conocido a lo desconocido, y puede decirse que es lo inverso a la deducción, que va de lo general a lo particular. Se da el nombre de *población* a cualquier conjunto o conglomerado numeroso de objetos a estudiar. Pero esta definición es tan general que prácticamente no adquiere verdadero significado si no se asocia a la definición de *muestra*: alguna parte de una población seleccionada deliberada u ordinariamente, para que las propiedades de la población se pongan de relieve. Los valores de las diversas medidas descriptivas de poblaciones se conocen como *parámetros*, pero cuando se refieren a muestras se denominan estadísticos. Así como el parámetro define una población, el estadístico describe una muestra.

Puede considerarse, el estadístico de una muestra, como una estimación del parámetro de una población. Una medida de estimación o estimador, es una función de las puntuaciones de una muestra, que da lugar a un valor denominado estimación, el cuál suministra información sobre el parámetro. Por ejemplo, la media de la muestra es un estimador de las puntuaciones promedio de la población.

Si enfocamos todos estos conceptos estadísticos hacia nuestro caso particular de la estimación del software, tendremos que las variables a estimar principalmente son: el tamaño del proyecto, el esfuerzo, el coste, la calidad y el tiempo que se tardará en desarrollarlo. La población serían los proyectos similares. La muestra podrían ser los proyectos similares realizados en nuestra compañía, y los estimadores los valores de los estadísticos correspondientes a los parámetros de la muestra. Un posible estimador para el tamaño del proyecto podría ser la media del tamaño de los proyectos de la muestra.

Es importante pensar en una predicción como en un rango más que como en un simple número. Una estimación o predicción no es un objetivo, sino una valoración probabilística. El valor que se obtiene de una estimación es el centro del rango. DeMarco [DeMar82] fue uno de los primeros en explicar que una estimación es una predicción que es igualmente probable que esté por encima o por debajo del resultado real. Para comprender lo que esto significa, consideremos una situación en la cual queremos estimar (desde un conjunto de requerimientos) el tiempo que nos llevará finalizar un proyecto. Imaginemos que pudiéramos registrar los valores reales de finalización de un gran número de proyectos, que hayan implementado el mismo conjunto de requerimientos. Entonces podríamos ser capaces de dibujar la función de densidad del tiempo t, necesario para finalizar el proyecto con los requerimientos establecidos; esto puede verse en el ejemplo de la figura 2.1.

La función de densidad en la figura 2.1 es una distribución normal, pero no necesariamente ha de ser siempre así. Desde este gráfico podemos calcular la probabilidad de que cualquier proyecto basado en los requerimientos dados, será realizado en un intervalo de tiempo [t1, t2]; la probabilidad es simplemente el área bajo la curva, entre t1 y t2. Por ejemplo, la probabilidad de que el proyecto se complete entre los meses 8 y 16 es aproximadamente 0,9, y la probabilidad de que el proyecto sea realizado en menos de 12 meses es de 0,5. Normalmente, estaremos interesados en la probabilidad de que un proyecto pueda ser finalizado en un tiempo t, es decir, que el intervalo sea [0, t].

Formalmente una estimación se puede definir como la mediana de una distribución. De este modo, la estimación debe dividir el área bajo la curva en dos partes iguales: parte A y parte B (figura 2.1).

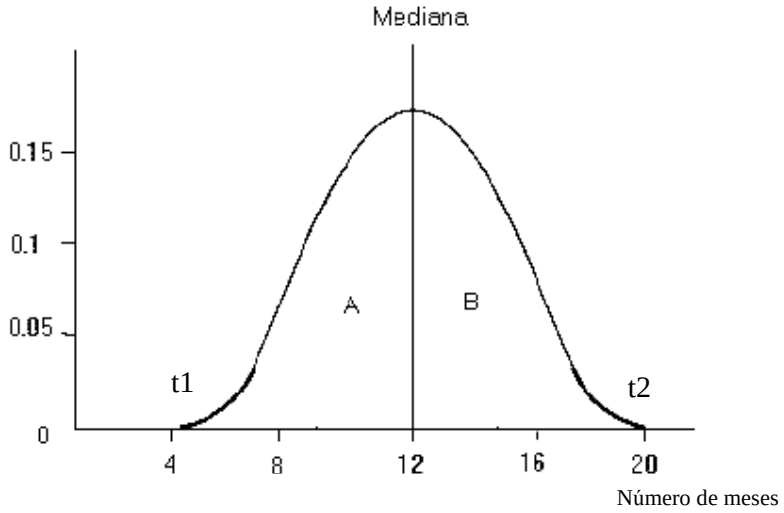


Figura 2.1: Curva de evolución de un proyecto

Los ingenieros de software frecuentemente entienden mal el concepto de estimación, considerándole en el ejemplo, como el número más pequeño de meses en el cual el proyecto puede ser finalizado. Esta interpretación apunta al mínimo valor t para el cual un intervalo [t , t+e] tiene una probabilidad distinta de cero (esto es, el valor más a la izquierda de la curva, distinto de cero). Este valor mínimo es el tiempo más corto en el cual se puede completar el proyecto. En la figura este valor es aproximadamente cuatro meses.

Incluso cuando el significado de la estimación se entiende correctamente, muchos ingenieros de software crean estimaciones y las establecen como objetivos. Mediante la designación de la mediana de la distribución de probabilidad como el objetivo, sabemos que hay un 50% de probabilidad de que el proyecto dure más tiempo. De este modo hay el 50% de probabilidad de que no se alcance el objetivo. Un estudio realizado por Jeffery y Lawrence [Jeffe85] encontró que la productividad era peor en proyectos donde fueron establecidos objetivos. Los gestores de proyectos entienden que la estimación es el centro del rango en el cual el proyecto puede ser completado, y estudian la probabilidad de los casos en los que el proyecto puede tomar menos o más tiempo que el estimado. De este modo, para indicar el rango, la estimación se ha de presentar como una tripleta: el valor más probable (que es, la mediana de la distribución), más los límites superior e inferior del valor. En términos estadísticos, al conjunto de valores entre estos límites inferior y superior se les denomina *intervalo de confianza*.

2.1.1 Precisión de las Estimaciones

Una vez que nuestro proyecto está terminado, tenemos la oportunidad de comparar los valores reales con las estimaciones que hicimos anteriormente. Supongamos que V_f es nuestro valor estimado y V_a nuestro valor real. El error relativo en nuestra estimación es:

$$RE = \frac{(V_a - V_f)}{V_a} \tag{1}$$

Frecuentemente, necesitamos obtener el error relativo de un conjunto de estimadores. Por ejemplo, nosotros usualmente queremos saber si nuestras predicciones de esfuerzo son precisas para un grupo de proyectos desarrollado. Aquí tenemos el error relativo medio para n proyectos:

$$\overline{RE} = \frac{1}{n}\sum_{i=1}^{n} RE_i \tag{2}$$

También es posible calcular la magnitud de estos mismos valores, considerando el valor absoluto, es decir, $MRE_i = |RE_i|$, por lo que para n proyectos tendríamos:

$$\overline{MRE} = \frac{1}{n}\sum_{i=1}^{n} |RE_i| \tag{3}$$

Si la media de la magnitud del error relativo es pequeña, entonces nuestras predicciones son buenas. Esta noción se usa para definir la métrica *calidad de predicción*. Para un conjunto de n proyectos, *i* es el número de ellos cuya magnitud media del error relativo es menor o igual que *q*:

$$pred(q) = \frac{i}{n} \tag{4}$$

La predicción de nivel q '*pred (q)'* proporciona una indicación del grado de ajuste para un conjunto de datos, basado en el valor de RE obtenido para cada dato. A modo de ilustración, si pred(0.25)=0.4, entonces se puede decir que el 40% de los proyectos tienen un error medio relativo menor al 25%. En términos de evaluación del rendimiento de un modelo dado, se podría considerar un buen modelo aquel que consiguiera que MRE<=0.25 y pred(0.25)>=0.75 [Conte86].

DeMarco sugiere el uso del método denominado factor de calidad de la estimación (*Estimating Quality Factor,* EQF) para evaluar la precisión del proceso [DeMar82]. Con este método, las estimaciones se realizan repetidamente a lo largo de todo el proceso. La estimación va aproximándose al valor real según avanza el proyecto. EQF varía desde

cero hasta infinito. Para una serie de estimaciones, DeMarco sugiere un valor de 4 (el cual corresponde a una estimación dentro del 25% del valor real del esfuerzo de desarrollo), que debe ser relativamente fácil de retener; por su parte, los estimadores de coste deben guardar valores de 8 ó superior.

2.1.1.1 Principios de Estimación

La estimación del software es un tema abordado de forma general por todos los autores dedicados a la Ingeniería del Software [Press98], [Humph95], [Jones96]. En este sentido, suelen coincidir en que a la hora de realizar una estimación, hay que tener en consideración los siguientes principios básicos:

A. Aplicar la cantidad de recursos correcta para crear y refinar las estimaciones. Para determinar el nivel de detalle hay que considerar:
   – Magnitud del proyecto
   – Riesgo de las estimaciones inexactas
   – Incertidumbres del proyecto

B. La estimación de recursos requerida para un escenario dado no puede cambiarse arbitrariamente. Las características que afectan a la precisión requerida de la estimación pueden ser:
   –Riesgo inherente al proyecto
   –Fiabilidad de la información usada
   –Efectividad del proceso de estimación

C. Reestimar con frecuencia. A medida que evoluciona el proyecto se dispone de más información que confirmará o refutará las estimaciones originales. Esa información servirá de base para realizar estimaciones más exactas del resto del proyecto.

2.2 El proceso de Predicción

Un proceso para realizar predicciones y mejorar su precisión debe incluir tres actividades: seleccionar y modelar las magnitudes a predecir, realizar las predicciones y evaluar la precisión de éstas. La inclusión de las dos primeras actividades es evidente. La evaluación es necesaria a fin de mejorar el ajuste de los resultados obtenidos. La evaluación proporciona una realimentación dentro del proceso. Sin realimentación no sería posible aprender de la experiencia, y no se establecerían las bases para mejorar futuras predicciones.

Se puede establecer un paralelismo entre el proceso de predicción y el paradigma de mejora de calidad QIP (Quality Improvement Paradigm) [Basil94]. Ambos seleccionan métricas, realizan medidas, analizan los datos obtenidos y aprenden de la experiencia. El proceso de predicción se puede ver como una especialización de QIP, el cual se centra en la realización de predicciones acerca de procesos y productos, para posteriormente ajustarlas todo lo posible. La Tabla 2.1 muestra los seis pasos del QIP, junto a sus equivalentes en el proceso de predicción.

QIP
1. Caracterizar el proyecto en curso y su entorno, con respecto a modelos y métricas.
2. Establecer metas cuantificables para conseguir un rendimiento óptimo y perfeccionamiento.
3. Elegir los modelos de procesos adecuados y soportar los métodos y herramientas para este proyecto.
4. Ejecutar el proceso, construir el producto, recolectar y validar datos prescritos, y analizarlos para proporcionar realimentación en tiempo real para corregir la acción.
5. Analizar los datos para evaluar el proyecto, determinar problemas, registrarlos y realizar recomendaciones para mejoras en futuros proyectos.
6. Almacenar la experiencia en el formulario de modelos modificados y mejorados, y otros formularios de conocimientos estructurados, obtenidos de éste y de proyectos previos, y guardarlos en una base de datos.

Proceso de Predicción
1. Caracterizar el proyecto en curso y su entorno con respecto a las restricciones y metas establecidas para él.
2. Seleccionar la métricas predictivas y métodos por los que evaluar si el proyecto puede conseguir satisfacer restricciones y metas.
3. Planificar la cantidad de datos necesarios para soportar el proceso de predicción.
4. Realizar las predicciones y documentar la suposición en la cual está basado. Valorar la precisión de las predicciones, para proporcionar una realimentación en tiempo real para acciones correctivas, y para realizar nuevas predicciones basadas en ello.
5. Evaluar las métricas predictivas, analizando los datos recolectados. Determinar las causas de las imprecisiones y realizar recomendaciones para el futuro.
6. Almacenar la experiencia como en QIP.

Tabla 2.1: Pasos del QIP y equivalente en el proceso de predicción

A continuación se describen los pasos a seguir en el proceso de predicción:

a) Caracterizar el proyecto.

El primer paso en el proceso de predicción consiste en obtener información acerca del proyecto y de su entorno, la cual es relevante para determinar las restricciones y objetivos del proyecto. El tipo de información obtenida es la misma que se obtendría en el primer paso de QIP.

b) Seleccionar las métricas y métodos.

El segundo paso en el proceso es seleccionar las métricas a predecir para el proyecto y los métodos mediante los que realizaremos esa predicción. Este paso es similar al segundo paso de QIP, el cual consiste en seleccionar las métricas mediante las que evaluar si se ha encontrado el modo de perfeccionar el proyecto. Las métricas a predecir deben reflejar las restricciones y objetivos del proyecto, a fin de evaluar si pueden ser satisfechas. Por esto, los puntos de función han sido utilizados como métricas de funcionalidad, y se han desarrollado modelos que los relacionan con el esfuerzo de desarrollo [Albre83]. Es posible predecir el número de estos puntos que se pueden desarrollar hasta la fecha de entrega; para el sistema completo, éstos pueden ser contados a partir de las especificaciones del sistema. El método mediante el que predecir la métrica también ha de ser seleccionado. Un modelo de regresión que relacione el esfuerzo con los puntos de función se puede usar en modo inverso para predecir tales puntos, siempre que se tengan suficientes datos disponibles acerca de proyectos similares previamente realizados.

c) Planificar la magnitud y recolección de datos.

El tercer paso en el proceso de predicción planifica cómo medir y recolectar los datos necesarios para analizar la precisión de las predicciones y los datos necesarios para construir nuevas predicciones o revisar las ya realizadas. El plan desarrollado en este paso no difiere con el que se realizaría en un proceso QIP. Este plan se ejecuta durante el desarrollo del sistema. Si una predicción puede venir condicionada por una decisión no procedente del desarrollo del sistema, tal como sería un estudio de factibilidad, este paso ha de ser aplazado hasta después de que se realice la predicción y se tomen las decisiones oportunas.

d) Realizar las predicciones y evaluar su precisión.

En el cuarto paso del proceso, se realizan las predicciones. Según los datos van estando disponibles, las predicciones son analizadas para evaluar si están siendo cumplidas o no. Los resultados de este análisis son realimentados tanto para el desarrollo del sistema como a la realización de las predicciones. Esta realimentación proporciona a los desarrolladores la oportunidad de tomar acciones correctivas, si ello es procedente, y permite la revisión de las predicciones basándose en los datos nuevos. El cuarto paso del proceso de predicción es análogo al cuarto paso de QIP, donde los

datos son recolectados y analizados para evaluar y proporcionar una realimentación sobre si el perfeccionamiento de metas está siendo cumplido o no durante el desarrollo del sistema. Algunas predicciones pueden requerir datos que aún no están disponibles antes del desarrollo del sistema, por esto se harán nuevas predicciones como si fueran estimaciones revisadas durante el desarrollo del proyecto.

e) Analizar las medidas y los métodos.

El quinto paso del proceso de predicción consiste en la evaluación de métricas y métodos usados para hacer las predicciones. Este paso es necesario para aprender de la experiencia. Las razones por las que las predicciones son imprecisas, han de ser comprendidas y buscadas las soluciones. Estas soluciones deberían acompañarse de sugerencias para mejorar la precisión de la seguridad de predicciones futuras.

f) Almacenar la experiencia.

El sexto y último paso implica guardar la experiencia adquirida a fin de que sea usada por la organización en el futuro. Este paso es el mismo que el paso final de QIP. Esto es especialmente pertinente en un proceso de predicción, ya que las predicciones precisas sólo pueden ser esperadas cuando hay experiencia previa. Las predicciones hechas sin referencia a experiencia previa serán poco menos que conjeturas.

2.2.1 Métodos Predictivos

Cualquier predicción se debe basar en un modelo empírico, que relacione el atributo de interés con otros atributos mensurables. Este modelo es el punto de partida para cada método de predicción.

Los métodos empíricos analizan datos para establecer un modelo numérico de las relaciones entre las medidas de los atributos dentro del modelo. El análisis de regresión, utilizado en esta investigación, es un ejemplo de este tipo de método.

Los métodos de predicción por analogía, utilizan medidas de los atributos del modelo empírico a fin de caracterizar el caso actual, para el que se realiza la predicción. Las medidas conocidas para el caso actual son usadas para buscar un conjunto de datos que identifiquen casos análogos. La predicción se hace interpolando desde uno o varios casos análogos al caso actual.

Los métodos de predicción teóricos proponen un modelo numérico basado en el modelo empírico. Estos modelos deben ser validados por comparación con los datos actuales de las medidas.

Los métodos heurísticos suelen usarse como extensiones de otros métodos. Las heurísticas son reglas, desarrolladas mediante experiencia, que obtienen conocimiento

acerca de las relaciones entre los atributos del modelo empírico. Las heurísticas se pueden utilizar para ajustar predicciones realizadas con otros métodos.

Las opiniones de los expertos se reconocen también como métodos de predicción [Boehm81]. Las opiniones de los expertos no se incluyen dentro del marco de trabajo para seleccionar métodos de predicción, ya que estos métodos no se pueden caracterizar fácilmente.

2.2.2 Propósito de la Predicción

Los principales propósitos, aunque existen otros muchos, por los que se requiere la predicción de determinados tipos de medidas son principalmente los siguientes:

- Investigación de la viabilidad del desarrollo o compra de un sistema nuevo.
- Investigación del impacto del cambio en la funcionalidad de un sistema existente.
- Determinación del personal de soporte necesario, para el desarrollo del proyecto.
- Cuantificar el coste económico del sistema nuevo.

El propósito de una predicción tiene la mayor influencia sobre las decisiones tomadas en base a los atributos del sistema, que determinan los tipos de medidas apropiadas. Las medidas a predecir se pueden evidenciar a partir del propósito (preparación de presupuestos, ampliar la funcionalidad de las aplicaciones, etc.).

La métrica no tiene en cuenta el coste del sistema; este coste puede ser dividido entre software y hardware. El coste hardware incluye el coste del servidor sobre el que reside la base de datos necesaria para el servicio de información, y el coste del hardware cliente más periféricos. Antes de solicitar un presupuesto para el servidor, el equipo contable de la compañía, ha de predecir la capacidad del almacenamiento requerida, el ancho de banda de los interfaces de comunicaciones, y la capacidad de procesamiento requerida para manejar la demanda de los usuarios. Partiendo de una métrica inicial del coste total, el número de medidas que han de ser predichas crece según se descompone el problema.

El propósito puede también indicar lo rápidamente que es necesario tener una predicción, y el grado de certeza necesario en ésta. Este factor de tiempo de obtención de la predicción influye en la elección del método predictivo. Por ejemplo, puede ser necesario preparar rápidamente un presupuesto. Si el equipo contable tiene acceso a un modelo numérico desarrollado para sistemas similares, podría ser usado rápidamente para elaborar la oferta. Si no existen estos modelos, la predicción se realizará por

analogía con sistemas similares, de este modo se consume menos tiempo que desarrollando modelos numéricos.

La necesidad de cuantificar la incertidumbre en una predicción también influye en la elección del método de predicción. La incertidumbre que puede tolerarse en el coste puede ser determinado por la política de la compañía sobre márgenes de contingencia y si el coste ha de ser competitivo o no. La estimación de la incertidumbre en una predicción puede ser dificultosa, si la predicción se hace por analogía.

2.3 Procesos de estimación

La mayoría de los procesos de estimación describen cómo aplicar un método simple de predicción, el cual está basado en uno o más modelos algorítmicos. Boehm proporciona una amplia visión del proceso de estimación de costes de software e incluye el establecimiento de los objetivos de la estimación, la planificación de la estimación y la recopilación de estimaciones realizadas mediante varios métodos.

En contraste con la mayoría de procesos de estimación, hay dos ejemplos de procesos para grupos de estimación: “Wide-band Delphi” y “Estimeeting”. Esos grupos de estimación, son adjuntados a los otros, más que reemplazados y sugieren dos aproximaciones distintas para implicar un número de personas en el proceso de estimación.

La estimación del software es un problema que hoy en día sigue creando controversia entre los distintos investigadores que lo estudian, dado que entre muchos de los métodos de estimación suele haber grandes diferencias en sus resultados. Un reciente estudio de Dolado [Dolad01] propone un método de estimación utilizando programación genética, así como técnicas de regresión clásicas. En [Aguil01b], Aguilar et. al., hacen una propuesta en la que utilizan Simuladores de Proyectos Software y computación evolutiva para realizar las estimaciones en proyectos de desarrollo de software.

A continuación se describen los principales procesos de estimación que han venido siendo utilizados hasta el momento, que son:

- Proceso de Boehm
- Proceso de Bailey-Basili
- Proceso de DeMarco
- Proceso de Heemstra
- Proceso de Arifoglu
- Proceso de Humphrey (PROBE)
- Wideband Delphi
- Estimeeting

2.3.1 Proceso de Boehm

Boehm [Boehm81] propuso un proceso de siete pasos, para estimar el coste del software:

- *Establecer objetivos*: El primer paso es determinar nuestros objetivos en la estimación. Para ello hemos de evitar gastar tiempo en recopilar información y realizar estimaciones que no sean relevantes para las decisiones que tenemos que tomar.
- *Planificar recursos y datos requeridos*: En este paso, debemos planificar nuestra actividad de estimación, a fin de evitar la tentación de preparar una estimación con demasiada antelación.
- *Fijar los requerimientos software*: En este tercer paso determinaremos si las especificaciones en las que están basadas nuestra estimación son económicamente viables. Necesitamos saber qué estamos intentando construir a fin de estimarlo todo con precisión.
- *Detallar tanto como sea posible*: El cuarto paso nos indica que hacer una estimación con mucho detalle, mejora su precisión.
- *Utilizar varias técnicas y fuentes independientes*: Boehm clasifica las técnicas de estimación como modelos algorítmicos, opiniones de los expertos, analogía, Parkinson, price-to-win, top-down y botton-up. Este paso nos indica que hemos de usar más de una técnica para realizar una estimación del coste. El uso de una combinación de técnicas nos permite evitar los puntos débiles de cada técnica, así como tomar lo mejor de cada una.
- *Comparar e iterar estimaciones*: Una vez realizada nuestra estimación utilizando dos o más técnicas, debemos comparar los resultados. Si son inconsistentes, nuestro siguiente paso es investigar las diferencias. El objetivo es converger en una estimación lo más realista posible, mas que establecer un compromiso arbitrario entre las estimaciones iniciales.
- *Continuación*: El paso final en el proceso de estimación del coste continúa la estimación realizada en pasos anteriores. Debemos usar los resultados de la comparación entre la estimación y el valor real para mejorar nuestra técnica de estimación y nuestros modelos. Deberíamos re-estimar el coste cuando el alcance del proyecto cambia.

2.3.2 Proceso de Bailey-Basili

Bailey y Basili [Bayle81] propusieron el "Meta Modelo de Bailey-Basili" como un proceso para estimar el esfuerzo. El proceso tiene tres pasos principales:

- Calibrar el modelo de esfuerzo: El modelo de esfuerzo se calibra usando una regresión por mínimos cuadrados, sobre un conjunto de datos locales para calcular los coeficientes del modelo.
- Estimar los límites superior e inferior del modelo de predicción: Se trata de calcular el error estándar de estimación (*Standard Error of the Estimate,* SEE). SEE se puede utilizar para calcular los límites superior e inferior del intervalo de confianza construido para el modelo, asumiendo una distribución normal.
- Ajustar las predicciones del modelo para tener en cuenta la significancia de los atributos conductores del coste: El efecto de los conductores del coste sobre el esfuerzo se estima calculando un factor de ajuste del esfuerzo y aplicándolo a la estimación de esfuerzo inicial. El factor de ajuste para un punto de la regresión original es el número mediante el que el valor efectivo ha de ser multiplicado para obtener el valor predicho. Un modelo de ajuste de esfuerzo se calibra utilizando regresión lineal múltiple para determinar los coeficientes conductores del coste para cada uno de los atributos conductores.

La estimación final se realiza utilizando ambos modelos: el primero para realizar una estimación inicial del esfuerzo, y el segundo para ajustar la estimación, teniendo en cuenta los conductores del coste.

2.3.3 Proceso de DeMarco

DeMarco [DeMar82] propone un proceso para desarrollar y aplicar modelos de coste basados en datos adquiridos localmente. Él defendía el uso de modelos de coste de factores simples derivados de la regresión lineal. Un modelo de coste de factores simples predice el esfuerzo para todo o parte del proyecto basándose en una variable independiente. El esfuerzo estimado obtenido desde un modelo de coste de factores simples se ha de ajustar aplicando un factor de corrección. Los factores de corrección, se derivan del mismo modo al propuesto por Bailey y Basili.

El modelo de DeMarco soporta estimación botton-up y top-down. Sugiere dividir el proyecto en componentes de coste, tales como: esfuerzo de diseño, esfuerzo de implementación y esfuerzo de integración. Cada componente de coste es estimado

mediante uno o más modelos basados en la medida, los cuales están disponibles en el momento de realizar la estimación.

DeMarco también defendía el uso de modelos sensibles al tiempo, los cuales relacionan el esfuerzo total del proyecto con la duración. Sugiere que se use el modelo COCOMO para este propósito. A continuación se indican las actividades llevadas a cabo para utilizar este modelo:

- *Seleccionar el coste del componente a modelar*: El primer paso del proceso implica la selección de los componentes sobre los que se va a realizar una estimación de coste. El coste de los componentes está basado en una descomposición del sistema de desarrollo en actividades tales como especificación, diseño, implementación y test.
- *Seleccionar las medidas desde las cuales predecir los componentes y el coste total*: Una vez elegidos los componentes, el siguiente paso es determinar las medidas desde las cuales, el esfuerzo para completar esos componentes pueda ser predicho.
- *Desarrollar los modelos de coste basados en un simple valor, basándose en los datos existentes*: Una vez que se han determinado las métricas, deben desarrollarse los modelos de regresión mediante los que se predice el esfuerzo.
- *Estimar el coste y el error estándar de la estimación para cada componente del modelo*: Cuando el valor de las métricas utilizadas en el modelo de coste basado en un factor simple puede ser estimado o medido, el modelo se utiliza para predecir el esfuerzo de los componentes. Si se utiliza una estimación del valor de las métricas de entrada, el esfuerzo debería ser reestimado cuando haya sido calculado el valor real de tales métricas.
- *Ajustar las estimaciones basadas en diferencias entre el proyecto actual y proyectos pasados*: DeMarco indica que la estimación del esfuerzo debe ser ajustada mediante una evaluación subjetiva de las diferencias entre el proyecto en curso y los proyectos pasados.
- *Calcular el coste total a partir de los costes de los componentes*: Una vez estimado el coste de todos los componentes, se calcula el coste total como la suma de éstos.
- *Calcular el error de estimación global*: El error en la estimación total se calcula sumando el error estándar de la estimación de cada componente aislado.
- *Utilizar un modelo de coste sensible al tiempo para restringir la estimación total del esfuerzo del proyecto*: El paso final en el proceso de DeMarco utiliza la estimación del esfuerzo como entrada a un modelo sensible al tiempo, tal como COCOMO, para estimar la duración del proyecto.

2.3.4 Proceso de Heemstra

Heemstra [Heems92] describe un proceso general de estimación de costes. Su proceso asume el uso de modelos de esfuerzo dependientes del tamaño, y el uso de atributos conductores del coste, unidos a estos modelos para realizar ajustes de productividad. Este proceso se divide en siete pasos:

- *Crear una base de datos de proyectos finalizados*: El paso inicial en el proceso de estimación es la recolección de datos locales para validar y calibrar un modelo.
- *Validar el modelo de estimación*: El entorno en el que ha sido desarrollado el modelo de coste y los proyectos finalizados en los que está basado, pueden diferir del entorno en el que el modelo es usado. Heemstra advierte, por consiguiente, que antes de utilizar un modelo precalibrado por primera vez en una organización, éste debería ser validado, evaluando su precisión sobre datos de proyectos ya finalizados en la organización.
- *Calibrar el modelo de estimación*: Si en el paso previo se indica que el modelo no es válido para el entorno en el cual va a ser utilizado, el modelo necesita ser recalibrado usando datos de proyectos finalizados en el entorno actual.
- *Estimar el tamaño*: En este paso, se calcula el tamaño del software partiendo de las características propias del software a desarrollar.
- *Estimar el esfuerzo y la productividad*: Una vez estimado el tamaño, el siguiente paso convierte el resultado obtenido en estimación de esfuerzo. Los atributos conductores del coste con influencia en el esfuerzo del desarrollo del software son evaluados y utilizados para ajustar la estimación de dicho esfuerzo.
- *Distribuir el esfuerzo a lo largo de las fases de realización del proyecto*: En este paso, el esfuerzo total y la duración del proyecto son distribuidas a lo largo de las fases de desarrollo del software.
- *Analizar la sensibilidad de la estimación y los riesgos asociados*: En este paso se evalúa la sensibilidad del coste estimado para valores de los atributos conductores del software, y se determinan los riesgos asociados con la estimación de costes del proyecto.

2.3.5 Proceso de Arifoglu

El proceso de estimación de Arifoglu [Arifo93] consta de cuatro pasos:

- *Estimación del tamaño*: Este paso es equivalente a la estimación de tamaño del proceso de Heemstra.
- *Estimación del esfuerzo y duración*: Este paso convierte la estimación del tamaño en estimación del esfuerzo y estimación de la duración. Arifoglu sugiere el uso del modelo COCOMO en este paso.
- *Distribución del esfuerzo y la duración durante el ciclo de vida*: Una vez que el esfuerzo total y la duración del proyecto ha sido estimados, han de ser distribuidos a lo largo del ciclo de vida.
- *Reflejar el esfuerzo en el calendario*: El paso final del proceso de Arifoglu es convertir el número de días de trabajo requeridos para completar el proyecto en número de días transcurridos en el calendario. Arifoglu es partidario del uso del modelo de Esterling, para realizar esta conversión. Esterling [Ester80] propone un modelo para estimar el porcentaje de jornada laboral utilizado en la realización directa una determinada tarea. Este paso realiza la predicción de la duración del proyecto usando el modelo COCOMO u otro similar. La duración predicha por COCOMO es ya un tiempo estimado de calendario (incluyendo fines de semana, y tiempo medio de vacaciones y bajas por enfermedad).

2.3.6 Proceso de Humphrey (PROBE)

En [Humph95] Humphrey describe un proceso de estimación basada en proxys (*Proxy-Based Estimation* PROBE) como parte de su proceso de Software Personal. El método PROBE defiende el uso de medidas seleccionadas personalmente y modelos de regresión para estimar el tamaño de un producto software. Un proxy es una medida de tamaño que puede ser utilizada para estimar la longitud de un proyecto medida en líneas de código. Por ejemplo PROBE utiliza una cuenta de objetos como un proxy de medida de tamaño. La estimación de líneas de código se utiliza para predecir el esfuerzo individual para desarrollar el producto. Los límites superior e inferior del intervalo de predicción deben ser también estimados.

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. Se realiza una vuelta atrás desde cada estimación, comparando el valor estimado con el real. La vuelta atrás se introduce para mejorar el rendimiento de la estimación individual. La filosofía del Proceso Personal del Software es mejorar las habilidades de

los individuos lo cual dificulta en cierto modo la aplicabilidad de PROBE a equipos de trabajo.

2.3.7 Wideband Delphi

Es posible que la precisión de un método de estimación pueda ser mejorado mediante la opinión y la estimación de un grupo de personas. En 1981 Boehm [Boehm81] describe la técnica Wideband Delphi para combinar las opiniones de los expertos y realizar una estimación de tamaño.

A cada experto se le proporciona una especificación y un formulario de estimación. En una reunión los expertos discuten los asuntos de la estimación. Cada experto rellena su formulario de forma anónima. En esta ronda, se proporciona a los expertos un sumario de las estimaciones realizadas, posteriormente se mantiene otra reunión para discutir las estimaciones de la ronda anterior. Se celebrarán más rondas hasta que las estimaciones converjan satisfactoriamente.

Wideband Delphi es similar al Delphi original, desarrollado por RAND Corporation [Helme67]. No obstante, antes de hacer cada ronda de estimaciones, los expertos discuten sobre las estimaciones anteriores. En la técnica original no había interacción entre los expertos.

2.3.8 Estimeeting

Taff [Taff91] describe otro proceso para la estimación en grupo, *Estimeeting*, el cual ha sido usado por AT&T Bell Laboratories. Se mantiene una reunión conocida como *Estimeeting* para producir una estimación del esfuerzo para desarrollar una caracterización del sistema.

Antes de la reunión, un grupo de estimadores revisan los requerimientos y una caracterización de alto nivel del sistema. Estos estimadores no son parte del equipo de desarrollo, pero son expertos en subsistemas y serán cambiados cuando la característica en cuestión del sistema sea establecida. El equipo de trabajo que la establece, presenta los requerimientos y el diseño de los atributos de estimación. A la presentación le sigue un periodo de preguntas y respuestas. Entonces los estimadores realizan estimaciones para los subsistemas en los que son expertos. Los estimadores consultan entre ellos y con el equipo que establece la característica. El equipo que determina la característica es responsable de sumarizar los resultados de la reunión y calcular la estimación total.

A diferencia de Wideband Delphi que determinaba la estimación final por consenso, los resultados de *Estimeeting* son la conjunción de todas las estimaciones individuales.

Además permite estimaciones en mayor detalle, pero requiere un diseño de alto nivel, en el que se conozcan los requerimientos de los subsistemas. Wideband Delphi tiene la ventaja del consenso en cada estimación, pero el detalle de la estimación es menor, y cada estimador duplica el trabajo de los otros.

2.3.9 Conclusiones

En todos los procesos de estimación descritos en los anteriores apartados, se proponen una serie de pasos a seguir con el objetivo de realizar una estimación del software.

No cabe duda que Boehm es uno de los autores de referencia en cuanto a estimación de software se refiere. Este hecho queda reflejado en alguno de los procesos de estimación más relevantes como pueden ser el de Demarco o de Arifoglu, donde se aprecia la clara influencia de Boehm; en ambos procesos se toma como base para definir sus pasos el modelo COCOMO definido por este autor. Además, otro de los procesos analizados, el Wideband Delphi, fue también descrito por él.

Esta es una de las principales razones que nos ha hecho seleccionar el *Proceso de Boehm* para realizar el proceso de estimación en PDS, que se realizará en el capítulo 6, utilizando técnicas de estimación tradicionales.

2.4 Modelos de Estimación del Coste

Esta sección revisa una variedad de modelos de estimación del coste del software. El marco de trabajo de selección clasifica los modelos en: empíricos, analógicos, teóricos y heurísticos. En esta sección se clasificará cada modelo de acuerdo con el método de predicción en el que está basado fundamentalmente, ya que un modelo puede estar basado en más de uno de estos métodos.

Se realiza aquí una subclasificación de los modelos empíricos en paramétricos y no paramétricos. Un modelo paramétrico tiene una fórmula funcional explícita, relacionando una variable dependiente con una o más variables independientes, por ejemplo el modelo COCOMO. Un modelo no paramétrico no tiene una fórmula funcional explícita; por ejemplo, un modelo desarrollado en base a redes neuronales.

Los modelos de estimación que se van a analizar, basados en esta clasificación son:

- Modelos empíricos paramétricos.
- Modelos empíricos no paramétricos.
- Modelos analógicos.

- Modelos Heurísticos.

No hay modelos que estén basados solamente en heurísticas, de este modo discutiremos brevemente cómo se usan las heurísticas conjuntamente con otros modelos de estimación. Para el resto de los modelos, se discuten las técnicas de calibración de modelos.

Los modelos de estimación más comunes son modelos paramétricos empíricos, donde el esfuerzo es predicho basándose en una o más medidas simples. Estos modelos han sido extendidos en algunos casos mediante el uso de atributos conductores del coste. Las ventajas y desventajas de los atributos conductores del coste se discuten en este contexto.

2.4.1 Modelos empíricos paramétricos

La forma más simple de un modelo empírico paramétrico es una función que relaciona el esfuerzo para desarrollar un sistema o programa con una medida del tamaño. En este contexto, en la medida del tamaño se tiene en cuenta alguna característica del producto que se va a desarrollar, por ejemplo el conteo de líneas de código del programa. El esfuerzo se mide normalmente en personas-mes o personas-año.

Los modelos de datos se desarrollan creando una función y ajustando el conjunto de pares (tamaño, esfuerzo) mediante técnicas de regresión. Los modelos más comunes son los que tienen funciones lineales y exponenciales. El modelo de Albrecht et al. [Albre83] se basa en una relación lineal entre el esfuerzo y el tamaño medido en puntos de función. El modelo COCOMO descrito en [Boehm81], se basa en una relación exponencial entre el esfuerzo y el tamaño en líneas de código.

a) Modelos basados en líneas de código

Uno de los modelos de esfuerzo más conocido tiene la forma:

$$\text{EFFORT} = a\ \text{LOC}^{\,b} \qquad (5)$$

Donde EFFORT es el esfuerzo para realizar el sistema medido en personas-mes y LOC es el tamaño en número de líneas de código a desarrollar, medido en miles de líneas de código. Este modelo ha sido investigado por Walston y Felix [Walst77], Bailey y Basili [Bayle81], y Boehm [Boehm81], este último en la versión básica del modelo COCOMO. El valor del exponente *b*, indica la escala en la que varía el coste según aumenta el número de líneas de código.

Kitchenham [Kitch92] examina diez conjuntos de datos, incluidos los conjuntos de datos de Bailey y Basili, y Boehm, y determina que no hay soporte estadístico para suponer que el valor del exponente es distinto de uno. En este caso la posibilidad de que la relación sea lineal no puede ser descartada. No obstante el modelo no tiene en cuenta el grado en que varía el coste (por ejemplo la escala de variación del coste según las líneas de código), dentro de un conjunto de datos. Para investigar la posibilidad de error de estas conclusiones, Banker et al. [Banke94] usan un modelo de la forma:

$$EFFORT = a + b\ LOC + c\ LOC^2 \qquad (6)$$

Examinando once conjuntos de datos encontraron que el valor del coeficiente del término cuadrático c era considerablemente distinto de cero en seis de ellos. Estos seis incluían los conjuntos de datos de Bailey y Basili, y Boehm. Esto implicaba que algunos conjuntos de datos no eran lineales.

La forma cuadrática del modelo debería considerarse válida sobre un rango de valores de LOC que aparecen en el conjunto de datos utilizado y en el que el modelo ha sido ajustado. Banker et al. [Banke94] indican un valor negativo para el coeficiente cuadrático en cuatro de los seis casos significativos. Esto es desalentador, porque en los casos en los que el coeficiente cuadrático es negativo, el esfuerzo también es negativo; es más razonable esperar que el esfuerzo se pueda realizar de manera monótona, incrementándose en función de las líneas de código.

Los modelos basados en líneas de código tienen éxito explicando las variaciones del esfuerzo, donde la función asociada puede ser lineal o no. Conte et al. [Conte88] dan un ejemplo de modelo lineal con un coeficiente de correlación del 82% y un error medio absoluto del 37%. Miyazaki y Mori [Miyaz85] presentan un ejemplo de un modelo COCOMO calibrado con un error medio absoluto del 20%. La dificultad de usar modelos predictivos basados en líneas de código es que las líneas de código no pueden medirse hasta que el sistema esté completo.

b) Modelos basados en líneas de código y otras métricas.

La métrica de los puntos de función fué desarrollada por Albretch [Albre79], como una alternativa a las líneas de código para medir el tamaño del software. Algunos investigadores (Albretch y Gaffney en [Albre83], Kemerer en [Kemer87], Matson et al. en [Matso94]) han investigado sobre modelos de la forma:

$$EFFORT = a + b\ FP \qquad (7)$$

Donde EFFORT es el esfuerzo para desarrollar el sistema en personas-mes, y FP es el valor del tamaño medido en puntos de función. El cálculo de los puntos de función es una métrica aceptada como un estándar en el mercado [Ifpug94].

Van der Poel y Schach [Vande83] consideran a la vez número de ficheros, flujo de datos y procesos para derivar el tamaño del código. En COCOMO 2.0 [Boehm95] se introduce una medida del tamaño en puntos de objeto (Object-Point). Realiza la suma ponderada de pantallas, informes y módulos 3GL de un sistema. Se utilizan diferentes ponderaciones para graduar la cuenta de objetos, de acuerdo con su tipo, además se establece un grado de complejidad para los objetos: complejidad simple, media o alta.

En estudios más recientes, Dolado [Dolad97] analiza las relaciones entre los puntos de función de Albrecht y MarkII [Charl91], las líneas de código 4GL y el esfuerzo.

c) Modelos basados en otras medidas del tamaño.

DeMarco [DeMar82] propone varias métricas, que pueden ser utilizadas para predecir el esfuerzo para unas actividades particulares o en su totalidad. Las métricas basadas en la especificación del sistema se llaman "*Function Bang*" y "*Data Bang*". Estas métricas son designadas para ser calculadas mediante diagramas de flujo de datos y diagramas entidad-relación respectivamente. Una desventaja de estas métricas es que no hay conteo simple. Dependen de la ponderación de la complejidad para modificar el conteo, antes de que la suma total sea calculada. A este respecto se comparte algunas de las desventajas de las métricas.

Basili y Panlilio-Yap [Basil85] presentaron un modelo del esfuerzo basado en el número de páginas de documentación técnica.

$$\text{EFFORT} = a + b\ \text{DocPags} \tag{8}$$

Donde se cuentan las páginas escritas en documentos que describen el sistema, excluyendo el código fuente. Este modelo tiene éxito en explicar el 85 % de la variación total del esfuerzo, para un conjunto de datos de 23 proyectos.

Mukhopadyay y Kekre [Mukho91] investigan modelos para estimar el esfuerzo, basados en las medidas de las características del sistema, que contrastan con las métricas típicas basadas en la característica del dominio del sistema, como las líneas de código. En este caso el dominio es el proceso de control de manufacturación, y se encarga de realizar el conteo de las características del sistema, así el proceso de manufacturación necesita comunicarse con flujos de entrada y salida, y conocer su posición y capacidad de mecanismo de control. Basándose en los datos de 34 proyectos, ellos son capaces de demostrar que el esfuerzo estimado basado en su medida es similar a la estimación realizada mediante la actual medida de líneas de código.

Brownlow [Brown94] presenta un modelo de esfuerzo el cual puede ser aplicado al desarrollo de un sistema usando análisis y diseño orientado a objetos. Está basado en el número de objetos y en el número de servicios en el sistema. La forma del modelo investigado es:

$$EFFORT = a + b\ OBJECTS + c\ SERVICES \quad (9)$$

Lo et al. [Lo96] investigan modelos para predecir el esfuerzo de desarrollo de las interfaces gráficas de usuario (GUIs). El modelo está basado en contar las características del GUI:

$$EFFORT = a + b\ ACTIONWIDGETS + c\ DATAWIDGETS \quad (10)$$

Un control de acciones (ACTIONWIDGETS) se utiliza para iniciar comandos o funciones, por ejemplo un botón de aceptación. Un control de datos (DATAWIDGETS) visualiza o acepta datos, por ejemplo una caja de texto. EFFORT es el número de personas-hora necesarias para desarrollar el código asociado a una ventana que contiene un número conocido de controles de acción y de datos. Este modelo es capaz de explicar aproximadamente el 75% de la variación del esfuerzo con un conjunto de datos asociado a 35 ventanas.

d) Modelos dependientes del equipo de desarrollo.

Uno de los argumentos aportados por los que admiten un aumento del gasto durante el desarrollo del sistema, es que si el sistema crece, lo hace también el equipo de desarrollo. Un incremento del equipo conlleva un incremento de las líneas de comunicación entre los miembros del mismo, implicando un mayor gasto de tiempo en comunicación y toma de decisiones. Jeffery en [Jeffe87] explora la relación entre productividad, líneas de código (LOC) y máximo tamaño del equipo (MAX STAFF).
El modelo tiene la forma:

$$PRODUCTIVITY = a\ LOC^{b} / MAX\ STAFF^{-c} \quad (11)$$

La productividad es equivalente al número de líneas de código dividido por el esfuerzo de desarrollo medido en personas-mes. Este modelo es capaz de explicar entre un 60% y un 80% de la variabilidad en la productividad para el conjunto de datos investigados, y denota que el aumento de productividad disminuye el aumento del equipo.

Conte et al. [Conte86] describen el modelo COPMO, el cual relaciona el esfuerzo total con el tamaño y el equipo del proyecto. El modelo COPMO está basado en la suposición de que el esfuerzo para desarrollar un sistema de un tamaño dado, puede ser modelado mediante el esfuerzo de implementación del sistema, más el esfuerzo requerido para coordinar el proceso de implementación con un equipo interactivo de trabajo. El modelo derivado de esta forma es:

$$EFFORT = a + b\ LOC + c\ STAFF^{d} \quad (12)$$

Donde EFFORT se mide en personas-mes, LOC en líneas de código, y STAFF es el número medio de técnicos. El número medio de técnicos es equivalente al esfuerzo total dividido por la duración del proyecto medida en meses.

e) Modelos de tiempo transcurrido.

Los gestores de proyecto, los clientes y los desarrolladores de sistemas, están normalmente muy interesados en estimar cuánto tiempo llevará desarrollar un sistema desde el inicio al fin, o si éste puede ser terminado en un plazo determinado.

Normalmente lo que se quiere es estimar el mínimo tiempo en el cual el sistema puede ser terminado. La duración del desarrollo del sistema está claramente relacionada con el número de personas que componen el equipo del proyecto. Lo que estos modelos plantean, es como se puede optimizar la productividad si se aumenta o disminuye él número de personas que están trabajando en el sistema.

Algunos investigadores han establecido relaciones entre el esfuerzo de desarrollo y la duración. En [Putna78] Putnam propone un modelo relacionando el tiempo para desarrollar el sistema con el esfuerzo total y el tamaño del sistema:

$$LOC = C_k K^{1/3} T_d^{4/3} \qquad (13)$$

Esta ecuación es la denominada *ecuación del software* y en ella LOC es el tamaño en líneas de código, K es el esfuerzo total durante el ciclo de vida del proyecto y T_d es el tiempo en el que el sistema ha de ser entregado. C_k es el factor tecnológico y su valor depende de las características del entorno de desarrollo.

Parr [Parr80] propone una variación del modelo de Putnam (que no contempla las fases de análisis del sistema y especificación), reemplazando la distribución de Rayleigh utilizada por Putnam, con una similar pero que no es cero en el origen. El cambio fue propuesto para proyectos donde parte del personal estaba ya trabajando desde el inicio del proyecto. En este sentido, COCOMO incluye un modelo que relaciona el tiempo con el esfuerzo de desarrollo:

$$T_{nom} = a\ EFFORT^{B} \qquad (14)$$

Donde T_{nom}, es el valor nominal del tiempo de desarrollo en meses y el esfuerzo en personal-mes. Basándose en una observación empírica, Boehm teoriza con que hay una "zona imposible" para el tiempo de desarrollo. Esta zona está definida aproximadamente por:

$$T_{dev} < 0.75\ T_{nom} \qquad (15)$$

Donde T_{dev} es el tiempo comprimido de desarrollo. Si el tiempo de desarrollo deseado es el 75% del tiempo nominal, el modelo COCOMO intermedio recomienda un valor multiplicador del 1.23, o un incremento de un 23% en el esfuerzo actual, sobre el valor nominal debido a la compresión de la planificación.

Los modelos de Putnam y COCOMO implican que reduciendo la duración del proyecto por debajo de lo que podría ser esperado para un proyecto de un tamaño dado, se incrementará el esfuerzo total. Un estudio de Jeffery [Jeffe94] explora la relación existente entre el valor de esfuerzo actual y el esperado y el tiempo transcurrido para un total de 47 proyectos. Este estudio revela una relación compleja entre esos valores. Algunos proyectos con duración más corta a la esperada, muestran un considerable incremento en el esfuerzo. Sin embargo, son posibles otras combinaciones como menor duración implica menor esfuerzo, mayor duración implica mayor esfuerzo y mayor duración implica menor esfuerzo.

Kitchenham en [Kitch92b] confirma la existencia de esta relación dentro de conjuntos separados de datos. Claramente hay factores a los cuales se puede recurrir en orden a explicar las variaciones en la productividad, la cual puede ser observada regularmente. Jeffery [Jeffe87], reexamina sus datos, establece y encuentra que la relación entre el tamaño del proyecto y el personal se estima en un 70% de la variación de la productividad. Muchos modelos empíricos paramétricos usan parámetros conductores del coste para tener en cuenta los factores que influyen en la productividad.

2.4.1.1 Controladores del coste

Los modelos discutidos en los puntos anteriores no tienen en cuenta una variedad de factores como la experiencia de los desarrolladores del sistema, con los que esperamos mejorar la productividad. Estos factores de productividad son denominados habitualmente *conductores o controladores del coste*. Estos parámetros son valorados en una escala ordinal, por ejemplo, baja, media o alta experiencia, o en una escala nominal, por ejemplo una clasificación del dominio de aplicación del sistema.

En el modelo COCOMO Intermedio se tienen en cuenta 15 controladores del coste. Un controlador del coste en el esfuerzo es un multiplicador, que incrementa o decrementa el valor nominal. Una vez que un controlador del coste ha sido evaluado, el valor del multiplicador se determina mediante una tabla que asocia el controlador del coste al valor del multiplicador. Por ejemplo, un alto nivel de experiencia en la aplicación reduce el esfuerzo en un 9%.

Los puntos de función incorporan también la influencia de los controladores del coste. Esto permite que el valor de los puntos de función para un sistema puede ser usado como base para realizar comparaciones de productividad entre proyectos. Los puntos de función no ajustados se multiplican por factores de complejidad, que son calculados evaluando el grado de influencia de 14 factores de productividad independientes.

No siempre es necesario modelar la influencia de un factor de productividad, como un multiplicador de esfuerzo. COCOMO 2.0 [Bohem95] modela la influencia con cinco factores multiplicadores exponenciales del valor de las líneas de código, dentro de su ecuación del esfuerzo.

Una dificultad implícita al uso de un número elevado de controladores del coste en la estimación del software, es que estos controladores pueden no ser independientes. Kitchenham [Kitch90], demostró que hay una relación entre dos controladores del coste en el modelo COCOMO, y apunta que las relaciones entre parámetros de entrada pueden hacer el modelo inestable. Una manera de cubrir la necesidad de considerar un gran número de controladores del coste es considerar sólo aquellos que son significativos en una organización o entorno de desarrollo particular.

Subramanian y Breslawski [Subra93] exponen técnicas que pueden ser utilizadas para elegir un subconjunto de controladores relevantes del coste para una organización o entorno específico, dentro de un conjunto más grande como puede ser el usado por el modelo COCOMO Intermedio. Estas técnicas han sido demostradas sobre la base de datos de COCOMO y reducen el número de controladores del coste de 15 a 4, sin sacrificar la precisión. Kitchenham llega a una conclusión similar, reduciendo los controladores del coste de 21 a 7 con una variación de la productividad del 75%.

Hay más evidencias de que un modelo puede ser más satisfactorio cuanto más simple sea. Jeffery y Stathis [Jeffe96] estudiaron que en el caso de los puntos de función, los factores de complejidad técnica no mejoran significativamente la precisión del modelo para su conjunto de datos. Simplificar los modelos es interesante porque hace que los modelos sean más sencillos de aplicar, y más fáciles de calibrar.

2.4.1.2 Calibración.

Cuando se usan modelos empíricos fuera de la organización o entorno en el que están basados sus datos, las predicciones realizadas por el modelo probablemente serán imprecisas, aunque el modelo sea recalibrado utilizando datos locales. Incluso modelos genéricos como COCOMO falla a la hora de realizar ajustes de la predicción sin calibración. La necesidad de la calibración ha sido confirmada en estudios de Kemerer [Kemer87], Kitchenham y Taylor [Kitch84], Low y Jeffery [Low90].

Para calibrar modelos empíricos paramétricos se ha de retornar a la forma funcional básica y ajustar la función al conjunto de datos locales. El número de datos necesarios depende del número de parámetros independientes en el modelo y de sus rangos posibles. Esto restringe la complejidad del modelo y permite que sea calibrado con precisión.

2.4.1.3 Estimación de la incertidumbre.

Un simple valor no puede ofrecer ninguna pista acerca del grado de certeza de una estimación. Cuando estimamos por medio de un modelo, se introduce incertidumbre adicional porque ningún modelo será capaz de explicar todo acerca de la variación de una variable dependiente, y porque los valores que aportamos como parámetros de entrada son inciertos en sí mismos.

Conte, Dunsmore y Shen [Conte86] sugieren que un modelo es aceptable si su error absoluto medio es menor o igual al 25%, y si al menos el 75% de los valores predichos están dentro del 25% de sus correspondiente valores actuales. Estas métricas de aceptabilidad son ampliamente usadas cuando se evalúa la precisión de un modelo. Incluso para modelos que satisfacen estos criterios, el 25% de los valores predichos pueden diferir de los reales en más del 25%. La incertidumbre de los modelos de estimación no puede ser pasada por alto. Los parámetros de entrada son inciertos cuando son incapaces de medir sus valores directamente. Por ejemplo, cuando usamos un modelo como COCOMO Intermedio, para estimar el esfuerzo al inicio del proyecto, el número de líneas de código debe ser estimado y se deben tomar decisiones sobre los valores de los controladores del coste.

Low y Jeffery [Low90] demuestran que los puntos de función pueden ser estimados de manera más consistente que las líneas de código, esto implica que las estimaciones basadas en puntos de función introducen menos incertidumbre que las basadas en líneas de código. La incertidumbre de una estimación depende tanto de la del modelo como de la de las entradas al modelo.

La desviación estándar del error en la estimación se suele utilizar para evaluar el grado de certeza en los modelos empíricos y estimar el rango de posibles valores. Por ejemplo, COCOMO 2.0 sugiere usar un rango de aproximadamente el valor de la desviación estándar entorno al valor más probable.

La incertidumbre asociada con un parámetro de entrada se puede incorporar dentro de un modelo, estimando el valor más bajo (L), el más alto (H) y el más probable (M) de dicho parámetro. De este modo el valor esperado para la variable de estimación y su desviación estándar vendrían determinados por las siguientes expresiones:

$$\text{Valor Esperado} = (L + 4M + H) / 6 \tag{16}$$

$$\text{Desviación Estándar} = (H - L) / 6 \tag{17}$$

Esta técnica fue empleada por Putnam [Putna78] para obtener una estimación del tamaño del sistema, añadiendo los valores esperados de tamaño a cada componente del sistema, y estimando la desviación estándar total como la raíz cuadrada de la suma de los cuadrados de las desviaciones estándar de cada componente. La técnica puede ser utilizada para estimar el rango de un valor de entrada al modelo.

Los conjuntos borrosos son una alternativa para evaluar la incertidumbre de los valores de los parámetros de entrada. Fei y Liu [Fei92] propusieron f-COCOMO, la versión borrosa del modelo COCOMO. En este modelo se representan los multiplicadores asociados a los controladores del coste como intervalos borrosos en lugar de como valores simples, a fin de modelar la incertidumbre asociada con la influencia de los controladores del coste sobre el esfuerzo. De esta manera se estiman los límites superior e inferior del esfuerzo.

Determinar la incertidumbre asociada a una estimación es una tarea difícil. No obstante obtener un rango para los valores de entrada es útil, al menos para investigar la sensibilidad del modelo con sus entradas.

2.4.2 Modelos empíricos no paramétricos

Briand, Basili y Thomas [Brian92] describieron la técnica de reducción del conjunto optimizado (*Optimized Set Reduction* OSR) la cual emplea técnicas de reconocimiento de patrones para analizar conjuntos de datos. OSR está basada, en parte, en árboles de decisión y ha sido aplicada a una combinación de los conjuntos de datos de los modelos de Boehm (COCOMO) y Kemerer, para predecir la productividad. Cada proyecto dentro del conjunto de datos se representa mediante sus controladores de coste COCOMO y su valores de esfuerzo. La técnica OSR selecciona un subconjunto de proyectos en los que está basada la predicción de la productividad del nuevo; esta productividad está medida en personas-mes / líneas de código.

Los proyectos seleccionados para el subconjunto óptimo, comparten algunos controladores del coste comunes con el nuevo proyecto. Por ejemplo, todos los proyectos con complejidad nominal, baja fiabilidad de requerimientos y gran tamaño de la base de datos.

OSR selecciona los valores de estos controladores del coste, de tal modo que, los valores de distribución de productividad que estén dentro del subconjunto seleccionado, son medidos como buenos, de acuerdo con un criterio estadístico establecido, derivando una distribución de probabilidad de la frecuencia de distribución de los proyectos seleccionados, para un rango de intervalos de productividad. La productividad para el nuevo proyecto puede ser predicha calculando el valor esperado, basado en la derivada de la densidad de distribución.

Briand et al. [Brian92] comparan la precisión de la técnica OSR para un modelo COCOMO calibrado para el conjunto de datos combinado de COCOMO y Kemerer y un modelo de regresión. La técnica OSR tiene una media de error absoluto menor que los dos modelos paramétricos. Una ventaja de OSR es que puede ser aplicado con entradas de datos incompletas. Es posible realizar una estimación para un proyecto

donde sólo un subconjunto de controladores de coste es conocido. Otra ventaja es que los valores nominales u ordinales de los controladores del coste se pueden usar como entradas sin ser convertidos en valores numéricos multiplicadores.

Srinivasan y Fisher [Srini95] describen dos métodos no paramétricos más para generar modelos de esfuerzo. El primero de ellos utiliza un algoritmo de aprendizaje para generar un árbol de decisión. El segundo método utiliza un algoritmo de retropropagación para entrenar una red neuronal. Estos métodos son también testados utilizando un conjunto de datos combinado de COCOMO y Kemerer. Los valores de los controladores de coste COCOMO y las líneas de código estimadas son entradas a los modelos de esfuerzo.

El esfuerzo estimado mediante la red neuronal tiene una menor media de error absoluto que el árbol de decisión. La precisión de ambos modelos es comparable con la de OSR.

2.4.2.1 Calibración.

Cada una de las técnicas descritas anteriormente, requiere un gran número de datos, debido al gran número de variables independientes y rangos de valores cubiertos por los modelos. Los autores que trabajan en este tema, destacan la necesidad de un pequeño conjunto de datos COCOMO (63 proyectos) para aplicar sus técnicas, así como que el entorno de los proyectos sea el mismo.

Aunque sea pequeño el conjunto de datos COCOMO, es significativamente mayor que el que muchas organizaciones puede recolectar. Donde un conjunto de datos lo suficientemente grande está disponible, puede ser difícil conseguir que todos los proyectos sean del mismo entorno.

Los árboles de decisión, las redes neuronales y las técnicas OSR pueden ser aplicados en aquellos casos en los que el número de variables independiente sea reducido para completar el tamaño del conjunto de datos disponible. No obstante, no está claro si estas técnicas son mejores que la regresión simple bajo esas circunstancias.

2.4.2.2 Estimación de la incertidumbre.

La incertidumbre de un modelo puede ser evaluada por medio del error relativo. Briand et al. [Brian92], sugieren que la incertidumbre de una estimación OSR puede ser evaluada por medio de la entropía de la distribución de probabilidad. El valor de esta medida de entropía es la que es minimizada cuando se selecciona el subconjunto de proyectos óptimo. Observaron que este valor correlaciona bien con la media del error relativo.

Los modelos no paramétricos basados en un árbol de decisión u OSR, tienen en común la división del conjunto de datos para generar una estimación. Diferentes valores de entrada pueden, en general, seleccionar diferentes subconjuntos. Existe el riesgo de

seleccionar un subconjunto no representativo, en el sentido de que aunque contenga proyectos similares, el esfuerzo real puede variar significativamente debido a factores no considerados por el modelo. Esto puede provocar que una estimación basada en un rango de los valores de entrada varíe de manera discontinua.

2.4.3 Modelos Analógicos

El modelo analógico de estimación, se basa en comparar uno o más proyectos finalizados en un dominio similar al nuestro como medio para producir una nueva estimación. La estimación se lleva a cabo analizando los datos recopilados de estos proyectos finalizados y contrastándolos con el nuevo proyecto, evaluando así posibles similitudes. Como el esfuerzo de desarrollo se conoce de antemano, se utiliza como base de la estimación del nuevo proyecto. La estimación por analogía parece sencilla y directa, sin embargo hay una serie de problemas que han de ser determinados. La manera de establecer las características del proyecto es uno de estos problemas, puesto que al principio del proceso de estimación hay restricciones en cuanto a la información disponible. Ejemplos de esto pueden ser: el dominio de la aplicación, el proceso de desarrollo del software, etc. Otro problema común es la manera de contrastar varios proyectos a la vez. Siempre puede haber proyectos discrepantes que provoquen una desviación en la estimación.

El proceso para predecir la estimación del software, se puede dividir en los siguientes pasos:

- a) Selección de proyectos análogos.
- b) Evaluación de similitudes y diferencias.
- c) Evaluación de la calidad de la analogía.
- d) Consideración de casos especiales.
- e) Realización de la estimación.

*a) Selección de analogía*s: La primera etapa en el proceso es recopilar los proyectos adecuados que puedan ser comparados con el nuestro, en una base de datos. Este proceso implica la selección de proyectos análogos que reflejen tanto el entorno de desarrollo como las características del nuevo.

*b) Evaluación de similitudes y diferencia*s: Una vez que tenemos la base de datos de proyectos análogos, la siguiente etapa es evaluar las similitudes con el nuevo. Estas similitudes se determinan en base a las características del proyecto; el número de éstas depende de los datos disponibles para poder caracterizarlo .

*c) Evaluación de la calidad de la analogí*a: Las medidas que determinan la calidad son las siguientes:

- Error absoluto.
- Porcentaje de error y media del porcentaje de error.
- Magnitud del error relativo (MRE).
- Media de la magnitud del error relativo (MMRE).
- Métrica de calidad de predicción dentro de un 25% pred (0,25).

Los resultados de este punto, pueden ser utilizados por el proceso de analogía para examinar los resultados obtenidos con otras estimaciones, y de este modo realizar un proceso de realimentación.

*d) Consideración de casos especiale*s: En algunos casos puede ser necesario no considerar algunos proyectos. Incluso para proyectos seleccionados en la etapa segunda, puede ser que tengan características que deseemos ignorar. Por ejemplo, si el proyecto en consideración utiliza una metodología de diseño poco utilizada.

e) Realización de la estimación: Una vez completados todos los puntos anteriores se procede a realizar la estimación.

Mukhopadhyay, Vicinanza y Prietula [Mukho92] desarrollaron ESTOR, un modelo de razonamiento basado en casos (*Case-Base Reasoning* CBR), para estimar el esfuerzo de desarrollo del software. El razonamiento basado en casos es una forma de razonamiento analógico que emplea cinco procesos básicos:

- Construcción de una representación del problema objetivo.
- Recuperación de un caso adecuado para actuar como caso fuente de analogía.
- Transferir la solución desde el caso fuente al destino.
- Establecer las diferencias entre los casos fuente y destino.
- Ajustar la solución inicial para tener en cuenta esas diferencias.

En ESTOR los casos son proyectos software y cada uno se representa mediante los valores de este conjunto de medidas. Las métricas utilizadas por ESTOR son puntos de función y entradas al modelo COCOMO Intermedio. ESTOR recupera un caso para actuar como fuente de la analogía, basándose en el valor de los puntos de función del proyecto a estimar. Se calcula un vector distancia para encontrar el vecino (caso) más cercano. La solución inicial es el valor de esfuerzo para el proyecto análogo. Las diferencias entre éste último y el nuevo se determinan comparando los valores de sus

medidas. El valor del esfuerzo correspondiente al proyecto análogo se ajusta teniendo en cuenta esas diferencias, aplicando un conjunto de reglas.

Las reglas utilizadas por ESTOR se derivan de las opiniones del experto, cuya estimación es precisa para el conjunto de datos usado. Las reglas ajustan el esfuerzo mediante un multiplicador, si las precondiciones particulares de los proyectos fuente y destino se conocen previamente. El conjunto de datos utilizado para desarrollar ESTOR es un subconjunto de 10 proyectos tomados del conjunto de datos de Kemerer. Además fue testado con los 15 proyectos de este conjunto de datos, obteniéndose un error relativo del 53%.

ANGEL [Bourn97] es otro modelo de estimación por analogía en el que los proyectos se representan mediante componentes de puntos de función. Los proyectos análogos son vecinos de los nuevos, y se identifican calculando un vector distancia desde el nuevo a otros en el conjunto de datos. El esfuerzo para el proyecto nuevo se predice mediante una media ponderada del valor de esfuerzo de sus proyectos vecinos.

En ANGEL, el usuario puede especificar las métricas en las que se basa la búsqueda de proyectos análogos. ANGEL puede determinar automáticamente un subconjunto óptimo de medidas para un conjunto de datos particular.

La precisión de ANGEL es mejor que la de los modelos de regresión lineal. Los modelos de regresión están basados en las medidas del conjunto de datos, que presentan la más alta correlación con el esfuerzo. En el conjunto de datos Kermerer el error relativo para ANGEL es del 62%, el cual puede ser comparado con el 100% de los modelos de regresión y el 53% para ESTOR. Mientras que ESTOR parece ser mejor que ANGEL en este conjunto de datos, las reglas de ajuste para ESTOR estaban desarrolladas sólo con 10 de los 15 proyectos del conjunto de datos de Kemerer, y estas reglas pueden no tener éxito cuando se aplica a proyectos con distinto conjunto de datos.

2.4.3.1 Calibración

El conjunto de datos que se explora para encontrar casos análogos, deben ser representativos del caso para el que se quiere hacer la estimación. Las diferencias entre el caso fuente y el caso objetivo han de ser tan pocas como sea posible. Para soportar la estimación para una gran variedad de proyectos, son necesarios muchos puntos de datos y un rango de valores muy elevados. No obstante la estimación basada en analogía puede ser aplicada con éxito con pequeños conjuntos de datos que estén dentro del mismo entorno. En realidad la partición de los conjuntos de datos de manera acorde con los atributos del entorno, mejora la precisión de modelos de estimación como ANGEL.

2.4.3.2 Estimación de la incertidumbre

La incertidumbre de un modelo puede ser evaluada por medio del error relativo y un rango de estimación se genera calculando valores de estimación basados en el rango de entrada. No obstante, como sucedía en los modelos empíricos no paramétricos, la estimación basada en un rango de entrada muestra una entrada que varía de modo discontinuo, debido a las diferentes analogías seleccionadas.

Es también posible que los casos fuente seleccionados puedan compartir los mismos valores que el caso objetivo, pero no ser representativo del caso objetivo debido a factores que no se consideran en la representación de los casos. Esto puede cuantificar la precisión de estimaciones basadas en conjuntos de datos particionados.

2.4.4 Modelos Heurísticos

Los controladores del coste son un ejemplo de cómo las heurísticas son incorporadas dentro de los modelos paramétricos de estimación de esfuerzo. La influencia de los controladores del coste puede ser explorada estadísticamente, la evaluación inicial de los factores con más probabilidad de influir en el coste es un ejemplo de heurística. Los esfuerzos multiplicadores correspondientes a los controladores del coste pueden ser determinados de manera heurística si los datos disponibles para cuantificar el efecto de los controladores del coste estadísticamente son insuficientes.

Las reglas usadas en el modelo de razonamiento basado en casos de ESTOR son también ejemplos de heurísticas. Las reglas ajustan el esfuerzo para un proyecto análogo basándose en la opinión de un experto sobre cómo ciertas circunstancias influyen en el esfuerzo. De este modo, las heurísticas pueden utilizarse en dominios tales como la estimación del coste del software, donde existen unos cuantos modelos teóricos de relaciones causales y donde interactúan muchos factores de modo que resulta dificultosa la aplicación de modelos empíricos.

2.4.5 Conclusiones

En los apartados anteriores se ha realizado una revisión de los principales tipos de modelos de estimación del coste del software, y se ha clasificado cada modelo de acuerdo con el método de predicción en el que está basado principalmente (dado que un modelo puede estar basado en más de un método).

En resumen, las características de cada unos de los modelos presentados son:

- Los modelos empíricos se clasifican a su vez en paramétricos y no paramétricos. Un modelo paramétrico tiene una fórmula funcional explícita que relaciona una variable dependiente con una o más variables independientes. Un modelo no paramétrico no tiene una fórmula funcional explícita; por ejemplo, un modelo desarrollado en base a redes neuronales.
- Los modelos analógicos se basan en la comparación de uno o más modelos similares al que se esté estudiando, con el fin de obtener una nueva estimación.
- En cuanto a los modelos heurísticos, se puede observar que no hay modelos basados sólo en heurísticas; en este sentido, se ha podido comprobar cómo se utilizan las heurísticas conjuntamente con otros modelos de estimación.

Los gestores de proyectos suelen aplicar la estimación por analogía, sobre todo utilizando la información de proyectos realizados bajo su supervisión, además del modelo de estimación concreto que se esté aplicando. Normalmente la experiencia adquirida suele ser utilizada en la realización de otros proyectos.

La técnica wk-NN, descrita y aplicada en el capítulo 6 de esta tesis, utiliza una metodología basada en analogías, ya que para estimar el valor de una variable, utiliza los valores obtenidos en los k-proyectos más similares de la base de datos de PDS que se esté utilizando.

Los modelos de estimación más comunes son los empíricos paramétricos (modelo COCOMO/II), donde el esfuerzo es estimado basándose en una o más medidas simples. Las técnicas de estimación basadas en modelos de regresión (lineal y no lineal) aplicadas también en el capítulo 6, son modelos empíricos paramétricos, donde para cada variable del PDS se calcula una fórmula funcional explícita que la relaciona con uno o más atributos.

2.5 Modelos Dinámicos en la Toma de Decisiones en Proyectos de Desarrollo de Software

El proceso de desarrollo de software puede definirse como un conjunto de herramientas, métodos y prácticas que se emplean para producir software. Como cualquier otra organización, las dedicadas al desarrollo de software, mantienen entre sus principales fines la producción de software de acuerdo con la planificación inicial realizada, además de una constante mejora con el fin de lograr los tres objetivos últimos de cualquier proceso de producción: alta calidad y bajo coste en el mínimo tiempo.

La reducción del ciclo de desarrollo de software sin comprometer la calidad del producto final se ha convertido en un objetivo fundamental. No sólo por constituir una ventaja desde el punto de vista del cliente, sino que, según [Colli95], ofrece una serie de importantes beneficios entre los que destacan, una mayor vida comercial del producto y, la posibilidad de comenzar a producir lo más tarde posible acabando dentro del plazo, lo que permite emplear los últimos avances tecnológicos disponibles en el momento.

La mayoría de los avances producidos en el área de la Ingeniería del Software encaminados a la reducción del ciclo de desarrollo, pretenden lograr sus objetivos mediante una mejora sustancial de la tecnología de desarrollo empleada, tal y como se pretende en el modelo CMM (*Capability Maturity Model*) del SEI (*Software Engineering Institute*) y en la propuesta de la norma ISO 9000. El modelo CMM clasifica las organizaciones de desarrollo de software en una escala de uno a cinco, basándose en la madurez de los procesos que tienen lugar dentro de ellas. Según [Paulk93] podemos asegurar que conforme una organización de desarrollo de software progresa de un nivel al inmediatamente superior, la duración del ciclo de desarrollo disminuye garantizando siempre la calidad del producto final.

Tradicionalmente, se ha intentado afrontar el conocido problema de la crisis del software desde el punto de vista de la tecnología de desarrollo empleada. Así, se han producido significativos avances relativos al empleo de nuevas metodologías y enfoques de desarrollo, herramientas CASE, reutilización de código, etc.

A principios de los 90 se produce un salto significativo en la gestión de Proyectos de Desarrollo de Software con la aparición del primer modelo dinámico [Abdel91], que modela el proceso llevado a cabo en la gestión de dichos proyectos.

2.5.1 Simuladores de Proyectos de Software

La aparición, en los últimos años de los modelos dinámicos para Proyectos de Desarrollo de Software y de entornos de simulación potentes como Stella, Vensim, iThink o Powersim, ha facilitado la creación de los llamados *Simuladores de Proyectos de Software* (SPS) que ofrecen la posibilidad de simular el comportamiento de tales proyectos. Estos simuladores permiten a los directores de proyectos experimentar con diferentes políticas de gestión y dirección con un coste nulo, de forma que las decisiones que se tomen sean lo más acertadas posible [Chich93].

En definitiva, un SPS permite realizar los análisis siguientes:

1. Análisis a priori del proyecto: Simulación del proyecto, antes de empezar el proceso de desarrollo, bajo diferentes políticas de gestión y/o diferentes estimaciones iniciales de recursos y bajo el empleo de distintas tecnologías de desarrollo.

2. Monitorización del proyecto: Simulación del proyecto durante el proceso de desarrollo para ir adaptando las estimaciones del proyecto a la evolución real del mismo.

3. Análisis post-mortem del proyecto: Simulación de proyectos que ya han finalizado, en los que interesa saber cómo podrían haberse mejorado los resultados obtenidos.

En otras palabras, un SPS permite responder a cuestiones del tipo: *¿Qué ocurrirá si?* antes de comenzar, *¿Qué está ocurriendo?* durante la ejecución y *¿Qué hubiese ocurrido si?* una vez que ha finalizado.

La idea intuitiva y natural recogida en un modelo dinámico para PDS es que la evolución del proyecto y por tanto el cumplimiento de los objetivos depende de: la estimación inicial de los recursos necesarios para realizar el proyecto, de las políticas de gestión que se apliquen y de determinados aspectos relacionados con la propia organización de desarrollo (nivel de madurez o consolidación, experiencia, aspectos ambientales, disponibilidad de recursos, etc.).

La aparición, por tanto, de los SPS ha significado un avance significativo en la gestión y estimación de proyectos. Aún así, estos simuladores presentan un inconveniente importante que han frenado su utilización (a nivel empresarial): el elevado número de atributos (tanto del proyecto como de la organización de desarrollo) cuyos valores se deben conocer previamente.

Las variables básicas que permiten conocer el comportamiento de un sistema dinámico, y por tanto el proceso de desarrollo de software que modelamos mediante dicho sistema, están definidas mediante ecuaciones diferenciales. El sistema incluye también una serie de atributos que nos permiten estudiar el comportamiento del mismo. Estos comportamientos vienen dados por las políticas de gestión que pueden ser aplicadas en los PDS, tanto las relacionadas con el entorno del proyecto (número de tareas, tiempo, coste, número de técnicos, complejidad del producto, etc.) como las relacionadas con la organización de desarrollo.

2.5.1.1 Simulación de Proyectos Software

Los modelos dinámicos para PDS incluyen una serie de parámetros y tablas que nos permiten definir las políticas de gestión que pueden ser aplicadas en dichos proyectos, tanto las relacionadas con el entorno del proyecto como las relacionadas con la organización de desarrollo, y el nivel de madurez de la organización.

La Tabla 2.2 muestra una clasificación de los parámetros y las tablas de un Modelo Dinámico para PDS. Dentro del grupo “Entorno del proyecto” se encuadran los parámetros relacionados con las estimaciones iniciales del proyecto y con la complejidad del mismo. En el grupo denominado “Entorno de la organización” aparecen los relacionados con las diferentes políticas de gestión que se pueden aplicar

sobre el PDS y con el grado de madurez de la propia organización de desarrollo. Dentro del subgrupo "Políticas de gestión" se recogen los diferentes parámetros y tablas que definen las políticas de gestión relacionadas con la asignación de esfuerzo, las políticas de gestión relacionadas con el personal (contratación, despido, adecuación, etc.) y las políticas de gestión relacionadas con las restricciones en el tiempo de entrega del proyecto. Por otro lado, el subgrupo "Grado de madurez" incluye, fundamentalmente, los parámetros vinculados a los tiempos medios en los que se realizan determinadas actividades del proyecto y los valores nominales utilizados. En "Otros" contiene los relacionados con el grado de madurez de la organización pero difíciles de encuadrar en los apartados anteriores.

ENTORNO DEL PROYECTO	Estimaciones iniciales	
	Complejidad	
ENTORNO DE LA ORGANIZACIÓN	Políticas Gestión	Asignación esfuerzo
		Personal
		Tiempo final de entrega
	Grado de madurez	Tiempos medios
		Valores nominales
		Otros

Tabla 2.2: Parámetros y Tablas de un Modelo Dinámico para PDS.

Una vez definidos los parámetros del modelo, el gestor de proyecto debe decidir cuáles son las variables que se van a analizar. Las opciones habituales son las variables que definen el desarrollo del proyecto: tiempo de entrega, coste, productividad media de desarrollo, etc. Cada vez que se asigna un valor a cada parámetro del modelo, éste queda completamente definido y mediante su simulación se obtendrán unos valores para estas variables. Para generar un conjunto de casos de entrenamiento, el responsable del proyecto debe escoger un rango para cada uno de los parámetros del modelo. A continuación, la herramienta de simulación genera aleatoriamente un valor en cada uno de esos intervalos. A cada tupla de parámetros así definidos le corresponderá entonces una tupla de valores para las variables resultado de la simulación. De esta forma se genera un registro en una base de datos, con los valores de los parámetros y los valores obtenidos para las variables del proyecto que se deseen analizar. Repitiendo este proceso un número determinado de veces se puede obtener una base de datos, que se podrá utilizar para extraer información acerca del PDS, que permita al director tomar decisiones de la forma más acertada.

Para simular el comportamiento de un PDS mediante un SPS, el responsable del proyecto debe seguir por tanto, los siguientes pasos (Figura 2.2):

1. Definir los valores concretos que tomarán los atributos (parámetros y funciones) del proyecto.
2. Simular el proyecto mediante el SPS para obtener la evolución las variables que se desee analizar.
3. Analizar los resultados obtenidos y comprobar si estos resultados coinciden con los objetivos del proyecto (coste, calidad, tiempo, etc.).
4. Si los resultados obtenidos:

    a. Coinciden con los objetivos del proyecto, se conocerán los atributos, y por tanto las políticas de gestión, que nos permitirán obtener los resultados deseados en nuestro proyecto.
    b. No coinciden con los objetivos del proyecto, se deberá volver a redefinir el proyecto (paso 1) y se repetirán los pasos anteriores. Por tanto, se tendrá que simular el proyecto tantas veces como sea necesario hasta obtener los resultados deseados.

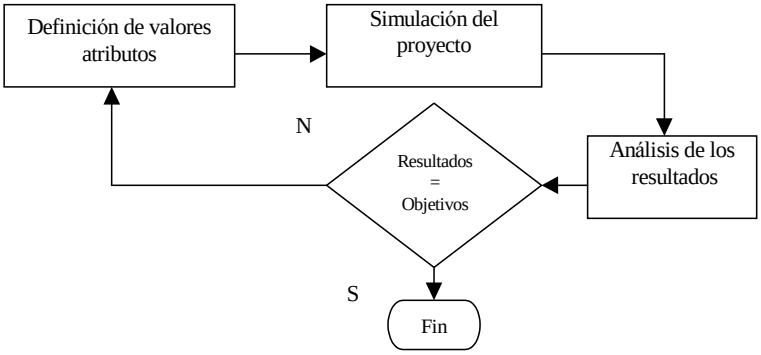


Figura 2.2: Pasos en la simulación de un PDS con un SPS

Por tanto, para disminuir el número de veces que se simulará el proyecto, el responsable del mismo deberá tener un nivel de conocimiento detallado de los valores concretos que tomarán los atributos del modelo, lo cual puede llegar a ser bastante dificultoso especialmente por dos motivos: primero, por el número de atributos que contiene un modelo dinámico y las múltiples combinaciones que se pueden establecer entre los mismos y segundo, porque en muchas ocasiones ni el propio responsable del proyecto conoce el valor concreto que tomarán determinado atributos aunque sí tendrá una idea clara del intervalo de valores dónde se moverá. Por ejemplo, posiblemente no sepa si la dedicación media de los técnicos en el proyecto será del 50% o del 24%, pero sí sabrá que dicha dedicación estará entre un 20 y un 60%.

Entre los pasos descritos anteriormente, sólo el paso 2 se realiza de manera automática. En el resto de los pasos, debe actuar el responsable del proyecto.

Capítulo 3

La Minería de Datos en la Toma de Decisiones

3.1 Introducción

La Minería de Datos (la extracción de información oculta y predecible de grandes bases de datos), es una poderosa tecnología con gran potencial para ayudar a las compañías a obtener la información más importante de sus Bases de Información (Data Warehouse). Las herramientas basadas en esta tecnología permiten predecir futuras tendencias y comportamientos, permitiendo tomar decisiones conducidas por un conocimiento acabado de la información (knowledge-driven). Los *análisis prospectivos* automatizados obtenidos de esta forma van más allá de los resultados suministrados por las herramientas típicas de apoyo a la toma de decisiones. Estas herramientas pueden responder a preguntas que de manera tradicional consumirían demasiado tiempo para poder ser resueltas; para ello, exploran las bases de datos en busca de patrones ocultos, encontrando información predecible que un experto no podría llegar a encontrar por mecanismos clásicos (experiencia adquirida o herramientas tradicionales de apoyo a la toma de decisión).

En la actualidad ya son muchas las compañías que almacenan y procesan cantidades masivas de datos. Las técnicas de Minería de Datos pueden ser implementadas en las distintas plataformas (software y hardware) existentes y pueden ser integradas con nuevos sistemas. Una vez que las herramientas fueron implementadas en potentes computadoras, pudieron analizar gigantescas bases de datos y proporcionar respuesta a preguntas tales como, *"¿Qué clientes tienen más probabilidad de responder al próximo mailing promocional, y por qué?*", que por su complejidad, las empresas ni siquiera se planteaban, aunque si deseaban conocer su respuesta.

3.2 Los Fundamentos de la Minería de Datos

Las técnicas de Minería de Datos son el resultado de un largo proceso de investigación y desarrollo, y se puede considerar que están en un estado de evolución idóneo para su aplicación, ya que está soportado por tres tecnologías con un alto nivel de madurez en la actualidad, que son:

- Recolección masiva de datos
- Potentes computadoras con multiprocesadores
- Algoritmos de Aprendizaje Automático

Las bases de datos comerciales están creciendo a un ritmo sin precedentes. Las cifras de almacenamiento estimadas (50 Gigabytes) a finales de los 90 en un estudio realizado por el META GROUP sobre los proyectos de Data Warehouse, ya se han superado con creces en la actualidad; por ejemplo, cualquier ordenador personal posee una capacidad de almacenamiento de más de 60 Gigabytes, y a nivel de empresas, ya hace años que muchas se mueven a nivel de TeraBytes en cuanto a capacidad de almacenamiento. Hoy en día ya se comienza a hablar del concepto de PetaByte (1 PB = 1.024 billones de bytes), aunque todavía no se han desarrollado memorias ni dispositivos de almacenamiento comerciales de esta capacidad. En Abril de 2003, IBM ha firmado un acuerdo con la Organización Europea para la Investigación Nuclear (CERN) con el objetivo de colaborar en la creación de un sistema masivo de administración de datos construido sobre tecnología Grid Computing. En este sentido, se espera que en 2005 se pueda estar manejando más de un PetaByte de datos.

Los algoritmos de Minería de Datos utilizan técnicas que han existido desde hace varias décadas, pero que debido principalmente a la evolución tan increíble que ha sufrido la capacidad de cálculo de los ordenadores en los últimos diez años, han sido implementadas recientemente como herramientas maduras y confiables, demostrando que son más útiles que los métodos estadísticos clásicos. Esta madurez, junto con el gran rendimiento de los motores de bases de datos relacionales, han hecho que su

aplicación sea esencial en los entornos de Data Warehouse actuales. Dada una base de datos de suficiente tamaño y calidad (sin ruidos, outliers, registros incompletos, etc.), la tecnología de Minería de Datos puede proporcionar los siguientes resultados:

a) Predicción automatizada de comportamientos

Automatización del proceso de encontrar información predecible en grandes bases de datos. Preguntas que tradicionalmente requerían un intenso análisis manual, ahora pueden ser contestadas directa y rápidamente desde los datos. Un típico ejemplo de problema predecible es el marketing orientado a objetivos (targeted marketing). Otros problemas predecibles incluyen pronósticos de problemas financieros futuros e identifican segmentos de población que probablemente respondan de manera similar a determinadas campañas.

b) Descubrimiento automatizado de modelos

Exploración de las bases de datos e identificación de modelos previamente desconocidos. Algunos problemas de descubrimiento de modelos son la detección de transacciones fraudulentas de tarjetas de créditos o la identificación de datos anormales que pueden representar errores de escritura en la carga de las bases de datos.

En general, las bases de datos pueden ser grandes tanto en profundidad (número de registros [filas]), como en ancho (número de atributos [columnas]):

- **Más columnas**. Los analistas muchas veces deben limitar el número de variables a examinar cuando realizan análisis manuales debido a limitaciones de tiempo. Sin embargo, variables que son descartadas porque parecen sin importancia pueden proveer información acerca de modelos desconocidos. En este sentido, la Minería de Datos permite a los usuarios explorar toda la base de datos, sin preseleccionar un subconjunto de variables.

- **Más filas.** Muestras mayores producen menos errores de estimación y desvíos, y permite a los usuarios hacer inferencias acerca de pequeños pero importantes segmentos de población.

Las técnicas más comúnmente usadas en Minería de Datos son:

- **Redes neuronales artificiales:** modelos predecibles no-lineales que aprenden a través del entrenamiento y que se asemejan a la estructura de una red neuronal biológica [Haiki99].

- **Árboles de decisión:** estructuras en forma de árbol que representan conjuntos de decisiones. Estas decisiones generan reglas para la clasificación de un conjunto de datos. Métodos específicos de árboles de decisión incluyen Árboles de Clasificación y Regresión (CART: Classification And Regression Tree) [Breim84] y Detección de Interacción Automática de Chi Cuadrado (CHAI: Chi Square Automatic Interaction Detection) [Kass80].
- **Algoritmos genéticos:** técnicas de optimización que usan procesos tales como combinaciones genéticas, mutaciones y selección natural en un diseño basado en los conceptos de evolución [Goldb89].
- **Reglas de inducción:** la extracción de este tipo de reglas [Cohen95] [Clark91], normalmente del tipo *if-then*, puede realizarse mediante diferentes técnicas de Minería de Datos, como pueden ser el *clustering* [Agraw98], [Guha98] o las *Reglas de asociación* [Agraw93].

Muchas de estas tecnologías han sido usadas durante más de una década, en herramientas de análisis especializadas que trabajan con volúmenes de datos relativamente pequeños. Estas capacidades están ahora evolucionando para integrarse directamente con herramientas OLAP y de Data Warehousing.

3.2.1 Sistemas de Aprendizaje

Un *sistema de aprendizaje* es un programa de ordenador que toma decisiones basadas en experiencias acumuladas de casos resueltos con éxito. A diferencia de un sistema experto que resuelve problemas usando un ordenador pero con un modelo de razonamiento humano, un sistema de aprendizaje puede usar diferentes técnicas para aprovechar el poder computacional de un ordenador, sin relación con el proceso de pensamiento humano.

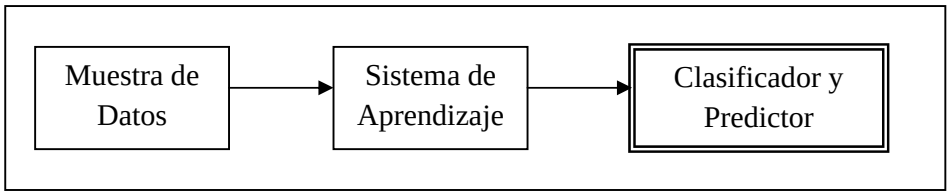


Figura 3.1: Sistema de Aprendizaje

Una de las tarea más destacadas y básicas del aprendizaje es la de clasificación o predicción. En este caso el sistema tiene disponible un conjunto finito o muestra de ejemplos de casos resueltos. Los datos de cada caso estriban en un conjunto de

observaciones o características de ese caso (patrón) y la correspondiente clasificación correcta.

El *aprendizaje* consiste (figura 3.1) en elegir una estructura modelo y adaptar correctamente unos parámetros de ésta para una óptima clasificación de los datos conocidos. El resultado es un clasificador, sistema que proporciona una decisión para cada patrón admisible que se le proporcione (figura 3.2).

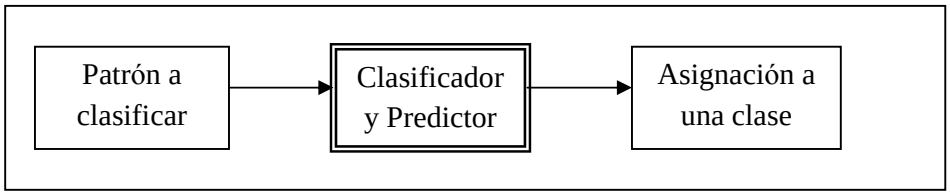


Figura 3.2: Utilización de un clasificador

Técnicamente, esta forma de aprendizaje se denomina *aprendizaje supervisado*, en el sentido de que el sistema aprende de un conjunto conocido de casos correctamente clasificados, que han sido producidos por un experto humano que supervisa la elección de estos casos. El *aprendizaje no supervisado* o *clustering* es un problema más difícil; en éste los casos no están clasificados y el objetivo es identificar clusters de patrones que son similares, identificando, de esa manera, posibles clases.

Los sistemas que aprenden en base a reglas, examinan una muestra de casos resueltos y proponen un conjunto de reglas de decisión en términos de un modelo subyacente. El objetivo fundamental de un sistema de este tipo es encontrar un conjunto de reglas que cubra todos los casos de una clase particular sin cubrir los casos de otras clases, o al menos, minimizando el número de estos casos o errores cometidos.

Dentro de las técnicas desarrolladas para encontrar un conjunto de reglas que represente el conocimiento de una base de datos, destacan dos metodologías: la construcción previa de árboles de decisión o la construcción directa de las reglas mediante otros métodos como algoritmos genéticos o clustering borroso.

3.2.1.1 Rendimiento de un sistema de aprendizaje

La medida más usada para evaluar el éxito de un clasificador es la tasa de error, que de forma empírica puede definirse como:

Tasa de error = número de errores / número de casos

utilizándose como estimador de la tasa de error verdadera o real la tasa de error de un clasificador sobre un número asintóticamente grande de nuevos casos seleccionados de

manera independiente de los usados en el diseño del clasificador. Si no es posible generar estos nuevos casos de manera fácil, se puede recurrir a la tasa de error aparente, definida como la tasa de error de la muestra usada en el diseño del clasificador. Lógicamente la tasa de error aparente casi siempre es un valor menor que la tasa de error real y a menudo un estimador demasiado optimista de ésta.

Para solucionar este problema se divide la muestra de aprendizaje en casos de entrenamiento y casos de test, y se calcula la tasa de error empírica sobre los casos de test. Existen diversas técnicas para obtener esa división de manera que la tasa de error calculada se aproxime a la real. Las más importantes son:

- *Muestreo aleatorio*: Se reparte de manera aleatoria la población de muestra entre casos de entrenamiento y casos de test, obteniéndose la tasa de error sobre estos últimos. Se repite un número de veces el procedimiento, estimándose la tasa de error real como la media de las tasas de error de cada prueba.

- *Validación cruzada*: Se fracciona la población en *k* conjuntos de aproximadamente igual tamaño. Se toma como casos de test un conjunto dado y se entrena el sistema con los *k-1* restantes, obteniéndose para cada conjunto una tasa de error. La tasa de error real se aproxima por la media de la *k* tasas resultantes. Una de las variantes más utilizadas de esta técnica es la conocida como "dejando uno fuera" ("Leaving one out" [Lache68]), que consiste en utilizar cada caso como conjunto test de un solo individuo, y los restantes como conjunto de entrenamiento. La tasa de error es ahora el número de errores cometidos en cada test dividido por el número de elementos en la muestra. Realmente la validación cruzada es una generalización del método de "dejando uno fuera".

- *Bootstrapping*: Este método desarrollado en [Efron82], consiste en realizar un muestreo con reemplazamiento en la muestra inicial siendo copiados en el conjunto de entrenamiento sucesivos casos de la muestra escogidos aleatoriamente y con reemplazamiento. Este proceso se repite hasta que el conjunto de entrenamiento tenga los mismos individuos que la muestra inicial. El conjunto de test son aquellos casos que no estén en el conjunto de entrenamiento. La tasa de error de este grupo es un estimador de la tasa real, aunque lo habitual es repetir el proceso un número determinado de veces y hallar la media.

En [Weiss91] se realiza una comparación entre estos métodos recomendándose validación cruzada o su variante "dejando uno fuera" para muestras mayores de 50 casos; para muestras de menos de 50 casos se recomienda Bootstrapping.

3.3 Descripción de las Técnicas Utilizadas

En este apartado, se realiza una descripción de las técnicas Estadísticas y de Minería de Datos utilizadas en esta investigación. Se han elegido estas técnicas por considerar que son las más adecuadas para procesar el tipo de información que se desea, fundamentalmente, estudiar en esta tesis (bases de datos cuantitativas de PDS).

3.3.1 Técnicas Estadísticas

Normalmente, las variables que se analizan en la realización de un muestreo son numerosas, existiendo entre ellas dependencias estadísticas. En tales situaciones puede plantearse el problema de hasta qué punto el conocimiento de unas variables, llamadas *explicativas*, aportan información suficiente para predecir los valores de otras denominadas de *respuesta*.

Dependiendo de los contextos, de las hipótesis que se consideren válidas, de la naturaleza de las variables y del número de éstas, se utilizarían los diferentes métodos de regresión:

- Regresión simple
- Regresión multivariante

El modelo de *regresión lineal simple* es un método sencillo para analizar la relación lineal entre dos variables cuantitativas. Sin embargo, en la mayoría de los casos lo que se pretende es predecir una respuesta en función de un conjunto más amplio de variables (por ejemplo bases de datos de PDS), siendo necesario considerar el modelo de *regresión lineal múltiple* como una extensión de la recta de regresión que permite la inclusión de un número mayor de variables.

Una de las principales dificultades a la hora de ajustar un modelo de regresión múltiple surge cuando es necesario identificar entre el conjunto de variables disponibles aquellas que están relacionadas con la respuesta y que la predicen de la mejor forma posible [Kittl86], [Siedl88]. Cuando el número de variables es reducido, la selección no resulta complicada.

En la mayoría de los casos se dispone de información de un conjunto muy amplio de variables, de las que se desconoce cuáles están relacionadas o pueden utilizarse para

predecir la respuesta de interés. La identificación del conjunto de variables que proporcionan el mejor modelo de regresión dependerá en gran medida del objetivo del estudio y de experiencias previas. Así, aunque la práctica habitual es eliminar del modelo aquellas variables que no resultan significativas, puede ser recomendable mantenerlas en caso de que en experiencias previas se haya constatado una relación con la variable dependiente.

La mayoría de paquetes estadísticos proporcionan una variedad de técnicas para identificar el mejor conjunto de variables regresoras que introducen o eliminan sucesivamente variables atendiendo a su significación en el modelo (hacia delante, hacia atrás, pasos sucesivos). Existen otras alternativas basadas en la comparación de todos los modelos posibles que se pueden formar con un conjunto inicial de variables. Todas estas técnicas deben considerarse meramente orientativas. Así, identificado el mejor conjunto de variables y ajustado el modelo, es conveniente realizar un análisis de residuos exhaustivo para valorar la posibilidad de elegir un modelo distinto a pesar de que tenga un valor menor de R^2.

En los últimos años se han propuesto diversos métodos para realizar selección de características (feature selection). En [Grazi00] se propone el algoritmo MFCMS (Modified Fuzzy C-Means algorithm with Supervision), basado en el algoritmo fuzzy c-means, y que utiliza el algoritmo de k-vecinos más cercanos como algoritmo de aprendizaje. En [Pabit02] se describe un algoritmo de selección de características no supervisada utilizando similitud entre características. Last, Kandel y Maimon en [Last00] y [Last01], analizan diferentes aspectos de los algoritmos de selección de características, y realizan interesantes propuestas en este sentido.

Interacción, Confusión y Colinealidad.

Cuando se introduce más de una variable en el modelo de regresión es necesario contrastar además la independencia de los efectos de todas ellas. Es decir, se supone que la asociación de cada variable con la respuesta no depende del valor que tome el resto en la ecuación de regresión. En otro caso se dirá que existe *interacción*. Antes de aprobar el modelo definitivo, por lo tanto, se debe explorar la necesidad de incluir términos de interacción calculados a partir del producto de pares de variables, comprobando si mejora la predicción, siendo aconsejable investigar solamente aquellas interacciones que puedan tener una explicación.

En ocasiones el fenómeno de la interacción se hace coincidir erróneamente con los de *confusión* y *correlación*. Existe confusión cuando el efecto de una variable difiere significativamente según se considere o no en el modelo alguna otra. Ésta se asociará tanto con la variable inicial como con la respuesta, de modo que en casos extremos puede invertir el primer efecto observado. En ese caso las estimaciones adecuadas son aquellas que proporciona el modelo completo, y se dirán que están controladas o ajustadas por variables de confusión.

Por otro lado, el fenómeno que se produce cuando dos variables explicativas muestran una correlación alta recibe el nombre *de cuasi-colinealidad* y puede producir estimaciones inestables de los coeficientes que se traducen en valores desorbitados de sus errores típicos y resultados poco creíbles. La mayoría de paquetes estadísticos muestran en sus salidas diagnósticos de colinealidad (tolerancia, factor de inflacción de la varianza, índice de condición) que pueden ayudarnos a solventar estos problemas. Por lo tanto, se ha de tener un cuidado especial en la etapa de construcción del modelo: un cambio significativo en las estimaciones tras la inclusión de una nueva variable puede evidenciar cualquiera de estos fenómenos. Es tarea del experto evaluar la conveniencia de incluirla o no en el modelo.

3.3.1.1 Modelos de Regresión Lineal Multivariante

En esta investigación, la regresión lineal se va a utilizar como técnica de estimación, por tanto se trata de predecir el valor de cada variable *y* como función lineal de una familia de *m* atributos (x_1, x_2,..., x_m), a partir de una muestra de tamaño *n* cuyas ocurrencias se ordenan matricialmente de la forma:

$$\begin{aligned} &((y_{1,}, x_{11}, x_{12}, \ldots\ldots\ldots\ldots\ldots\ldots, x_{1m}), \\ &\quad (y_{2,}, x_{21}, x_{22}, \ldots\ldots\ldots\ldots\ldots\ldots, x_{2m}), \\ &\qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &\quad (y_{n,}, x_{n1}, x_{n2}, \ldots\ldots\ldots\ldots\ldots\ldots, x_{nm})) \end{aligned}$$

siendo y_i la *i*-ésima variable de salida, y $x_{i,j}$ el *j*-ésimo parámetro asociada a la observación *i*. De esta forma, se trata de ajustar los datos a un *modelo lineal* de la forma:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots\ldots + \beta_m x_{im} + \alpha_i$$

Bajo las siguientes hipótesis:

1. Los residuos α_i son normales de media 0 y varianza común desconocida σ^2; además, estos residuos son independientes.
2. El número de variables explicativas (*m*) es menor que el de observaciones (*n*); esta hipótesis se conoce con el nombre de *rango completo*.
3. No existen relaciones lineales exactas entre las variables explicativas.

A) Estimación de los parámetros de regresión:

El estimador del vector paramétrico es:

$$\hat{\beta} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \cdot \\ \cdot \\ \hat{\beta}_m \end{bmatrix} = (X^T X)^{-1} X^T Y$$

siendo:

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ y_n \end{bmatrix}, X = \begin{bmatrix} 1 & x_{11} & x_{12} \ldots x_{1m} \\ 1 & x_{21} & x_{22} \ldots x_{2m} \\ \vdots & \vdots & \vdots \ \ldots \ \vdots \\ 1 & x_{n1} & x_{n2} \ldots x_{nm} \end{bmatrix}$$

habiéndose indicado la transposición matricial mediante el superíndice *T*.

3.3.1.2 Modelos de Regresión no Lineal Multivariante

Al igual que en el caso de regresión lineal, se trata de predecir el valor de cada variable *y* como *función no lineal* de una familia de *m* parámetros (x_1, x_2,..., x_m), a partir de una muestra de tamaño *n* cuyas ocurrencias se ordenan matricialmente:

$$\begin{aligned} &((y_{1,}, x_{11}, x_{12}, \ldots\ldots\ldots\ldots\ldots\ldots, x_{1m}), \\ &\quad (y_{2,}, x_{21}, x_{22}, \ldots\ldots\ldots\ldots\ldots\ldots, x_{2m}), \\ &\qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &\quad (y_{n,}, x_{n1}, x_{n2}, \ldots\ldots\ldots\ldots\ldots\ldots, x_{nm})) \end{aligned}$$

siendo y_i la *i*-ésima variable de salida, y $x_{i,j}$ el *j*-ésimo parámetro asociado a la observación *i*. Se trata por tanto de ajustar los datos a un modelo *no lineal*, que puede tener por ejemplo las siguientes formas:

$$y_i = \beta_1 x_{i1}^{\delta 1} + \beta_2 x_{i2}^{\delta 1} + \ldots\ldots + \beta_m x_{im}^{\delta 1} + \beta_0$$
$$y_i = Exp\ (\beta_1 x_{i1} + \beta_2 x_{i2} + \ldots\ldots + \beta_m x_{im} + \beta_{0)}$$

La idea básica sería igual que para el caso de regresión lineal, siendo la diferencia fundamental la forma de la función que queremos obtener. Obviamente, al tratarse de formas de función no lineales, los cálculos se hacen mucho más complejos que en el caso lineal, pudiéndose darse el caso de no poder aproximar la variable y mediante determinadas formas de función no lineales.

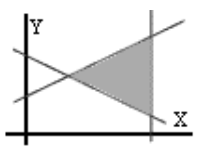
3.3.1.3 Programación lineal.

A groso modo, los *problemas de optimización restringida* consisten en encontrar el *mejor* valor de una cierta función dada. Así, en un problema de optimización restringida genérico intervienen un conjunto de variables entre cuyos valores se va a encontrar la solución del problema, unas *restricciones* que determinan los valores aceptables de dichas variables, y una *función objetivo* sobre tales variables, que se desea optimizar (minimizar o maximizar). Las restricciones pueden ser de dos clases: restricciones de igualdad, en las que se especifica que cierta función de las variables debe ser constante, mientras que en las restricciones de desigualdad se especifica que cierta función de las variables debe ser mayor o igual que (o menor o igual que) una constante. Por su parte, un problema de *programación lineal o programa lineal (PL)* es un problema de optimización en el que la función objetivo y todas las funciones que aparecen en las restricciones son lineales.

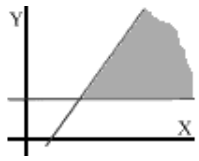
El abanico de situaciones en las que las técnicas de programación lineal (PL) pueden aplicarse es bastante amplio. Un buen ejemplo de ello es la programación de redes [Ahuja93], con múltiples aplicaciones en control de tráfico, tanto en las redes urbanas como en las de telecomunicación. En general, los programas lineales que es preciso resolver en este tipo de aplicaciones responden a una estructura especial, lo que fuerza a utilizar una algoritmia particular para cada tipo de problema.

El *aprendizaje de redes neuronales* [Rojas96] se ha replanteado recientemente para realizarlo mediante técnicas de optimización lineal restringida [Trafa99], debido a la extrema lentitud de los procesos de aprendizaje habituales basados en optimización irrestringida, motivada por el mal condicionamiento del planteamiento original del problema [Saari93].

La solución de un problema de programación lineal, en el supuesto de que exista, debe estar en la región determinada por las distintas desigualdades de sus restricciones. Esta recibe el nombre de *región factible*, y puede estar o no acotada (figura 3.3).



Región factible acotada



Región factible no acotada

Figura 3.3: Región factible acotada y no acotada

La región factible incluye o no los lados y los vértices según que las desigualdades sean en sentido amplio o en sentido estricto respectivamente. Si la región factible está acotada, su representación gráfica es un polígono convexo con un número de lados menor o igual que el número de restricciones.

El *teorema fundamental de la programación lineal*, nos describe el método analítico para solucionar un programa con dos variables:

- En un programa lineal con dos variables, si existe una solución única que optimice la función objetivo, ésta se encuentra en un punto extremo (vértice) de la región factible acotada, nunca en el interior de dicha región.

- Si la función objetivo toma el mismo valor óptimo en dos vértices, también toma idéntico valor en los puntos del segmento que determinan.

- En el caso de que la región factible no es acotada, la función lineal objetivo no alcanza necesariamente un valor óptimo concreto, pero, si lo hace, éste se encuentra en uno de los vértices de la región

- La evaluación de la función objetivo en los vértices de la región factible nos va a permitir encontrar el valor óptimo (máximo o mínimo) en alguno de ellos.

Uno de los métodos más conocidos para resolver problemas de programación lineal es el método *Simplex,* creado en 1947 por el matemático George Dantzig. Borgwardt en [Borgw87] hace un análisis del método desde el punto de vista probabilístico.

El método Simplex se utiliza, sobre todo, para resolver problemas de programación lineal en los que intervienen tres o más variables. El álgebra matricial y el proceso de eliminación de Gauss-Jordan para resolver un sistema de ecuaciones lineales constituyen la base de este método. Básicamente, se trata de un procedimiento iterativo que permite ir mejorando la solución a cada paso. El proceso concluye cuando no es posible seguir mejorando más dicha solución. Partiendo del valor de la función objetivo en un vértice cualquiera, el método consiste en buscar sucesivamente otro vértice que mejore al anterior. La búsqueda se hace siempre a través de los lados del polígono (o de

las aristas del trapecio, si el número de variables es mayor). Como el número de vértices (y de aristas) es finito, siempre se podrá encontrar la solución.

En resumen, el método del Simplex se basa en la siguiente propiedad: si la función objetivo, f, no toma su valor máximo en el vértice A, entonces hay una arista que parte de A, a lo largo de la cual f aumenta.

3.3.2 Árboles de Decisión: C4.5

Un árbol de decisión consiste en una estructura árbol con nodos y ramas. Cada nodo interno representa una condición simple y cada nodo terminal u hoja, una clase a asignar. Si el árbol es binario, las condiciones se evalúan con dos posibles resultados: verdadero o falso. Por el contrario, si el árbol no es binario, las condiciones pueden cubrir distintos valores, por ejemplo, si una característica es alta, media o baja.

La construcción de reglas lógicas a partir de estos árboles es inmediata, bastando recorrerlos en profundidad desde la raíz hasta las hojas, obteniendo reglas del tipo:

Si *Condición1 es falsa* o *Condición2 es verdadera* entonces *Clase A*

Los árboles de decisión son una estructura particularmente útil en el contexto del aprendizaje supervisado porque realizan la clasificación mediante una secuencia de test cuya semántica es intuitivamente clara y fácil de entender. Algunas técnicas, como C4.5 propuesta por Quinlan [Quinl93], construyen árboles de decisión seleccionando el mejor atributo mediante un test estadístico que determina cuán bueno clasificaría él solo los ejemplos de entrenamiento. Este tipo de árboles de decisión se denominan *paralelos a los ejes*, ya que los test realizados en cada nodo son equivalentes a hiperplanos paralelos a los ejes. Por el contrario, otras técnicas construyen árboles de decisión oblicuos, como OC1 [Murth94], que prueba una combinación lineal de atributos en cada nodo, por lo que se consideran equivalentes a hiperplanos con orientación oblicua respecto de los ejes coordenados. Encontrar el árbol de decisión más pequeño (paralelo a los ejes u oblicuo) es un problema NP [Blum88].

La metodología empleada por Quinlan para la construcción de árboles de decisión está basada en las ideas descritas en [Hunt66] sobre un algoritmo recursivo con técnica de divide y vencerás para la construcción del árbol, pero donde se encuentra la base del problema es en la búsqueda del árbol óptimo entre la infinidad de árboles posibles. Quinlan en el antecesor ID3 propuso un criterio que se basa en la siguiente norma de la Teoría de la Información: *la información transmitida por un mensaje depende de su probabilidad y puede ser medida en bits como el menos logaritmo en base 2 de esta probabilidad*.

En la herramienta C4.5, Quinlan corrigió numerosos de los problemas que aparecían en sus antecesores OC1 e ID3. En cuanto a la salida, el C4.5 proporciona una representación gráfica del árbol encontrado, además de una posible simplificación de éste, con un aumento mínimo del error. Además facilita los tamaños en número de nodos de estos árboles y unas tasas de errores aparentes (sobre el fichero de entrenamiento) para ambos, así como una tasa de error estimado para el árbol simplificado.

En C4.5 se introducen numerosas extensiones respecto a su antecesor ID3, entre las que destacan:

- Maneja ausentes mediante el cómputo de la ganancia de información para aquellos valores presentes
- Maneja atributos continuos

Una variante de C4.5 es la herramienta C4.5rules que genera reglas de decisión en lugar de árboles de decisión; en realidad, es un traductor de árboles de reglas. Con esta técnica puede que existan casos que las reglas no sean capaces de cubrir. Asimismo, también se necesita saber qué regla aplicar cuando más de una regla cubra a un nuevo ejemplo. La solución adoptada por C4.5rules consiste en localizar subconjuntos de reglas que cubran cada clase, utilizando el principio de Longitud de Descripción Mínima [Quinl94], [Rissa83]. El orden asociado con este criterio determina la clase por defecto que será asignada a un ejemplo no cubierto por las reglas.

Los sistemas de aprendizaje basados en árboles de decisión son muy efectivos en la práctica, además de ofrecer una representación simple y clara de los resultados. Sin embargo tienen dos limitaciones importantes:

1. Un árbol con un tamaño fijo no tiene porqué ser el mejor árbol de ese tamaño. La razón es que cada nodo es dividido una vez sin posibilidad de vuelta atrás
2. Las regiones de decisión tienen formas rectangulares (en 2 dimensiones, en general n-ortoedros), lo que dificulta encontrar fronteras de decisión tan simples como X > Y

La producción de reglas directamente es una herramienta potencialmente más poderosa que derivándolas a partir de un árbol de decisión. Las aproximaciones más extendidas son dos: los algoritmos genéticos y la lógica borrosa, que son descritas en los siguientes apartados.

3.3.3 Reglas de decisión mediante algoritmos genéticos: COGITO

El interés por aplicar la metodología de optimización de los algoritmos genéticos ha ido creciendo desde mediado de los ochenta. El principal problema de su aplicación es encontrar una representación adecuada para captar las características del problema y representar una posible solución. Además, la representación clásica mediante soluciones de longitud fija es un problema, ya que a priori no se conoce el número de reglas óptimo para cada caso. En este sentido existen dos enfoques diferentes:

- La metodología desarrollada en [Holla86], (*la escuela de Michigan*), consiste básicamente en representar en un cromosoma una regla que compite con otros individuos, y usa una población de individuos de longitud fija.

- La metodología propuesta en [Smith80], (*la escuela de Pittsburgh),* representa mediante un individuo de dimensión variable, un conjunto de reglas que compite con otro conjunto de reglas por cubrir todos los casos. Sin embargo, las reglas individuales mantienen la longitud constante.

La herramienta COGITO [Rique97], [Aguil97], [Aguil98] genera reglas de decisión mediante algoritmos genéticos; se trata de una familia de algoritmos que permite obtener un modelo de reglas capaz de clasificar a un conjunto de datos con el mínimo error posible en el contexto del aprendizaje supervisado. A diferencia del C4.5, COGITO no divide el espacio por un atributo, sino que extrae secuencialmente una región del espacio. Esto permite obtener regiones completas de la misma clase, tal y como se aprecia en la figura 3.4 b.

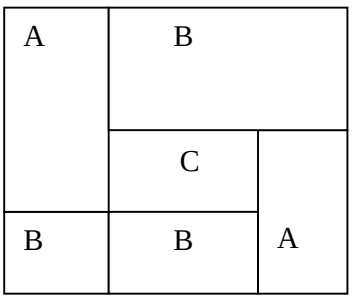


Figura 3.4 a: Partición con C4.5

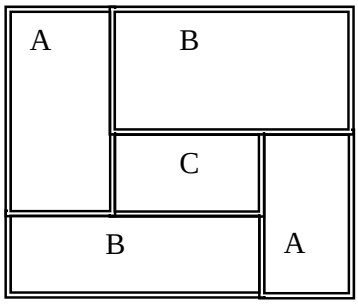


Figura 3.4 b: Partición con COGITO

Véase la región etiquetada como B en la esquina inferior izquierda de las figuras 3.4.a y 3.4.b. En la figura 3.4.a, C4.5 divide la región en dos partes; sin embargo, COGITO obtiene la región completa. Una de las principales ventajas de COGITO respecto a C4.5 es la reducción del número de reglas debido al solapamiento entre ellas. Esta es la razón por la que COGITO utiliza reglas jerárquicas de decisión en lugar de otras alternativas, como las reglas no jerárquicas o los árboles de decisión, que no consideran este aspecto.

Las formas de representar los valores de los atributos (en forma numérica o intervalar), utilizada en COGITO son las siguientes [Aguil99]:

- Hiperrectángulos paralelos a los ejes
- Hiperrectángulos oblicuos
- Hiperelipses

Las dos últimas representaciones se utilizan fundamentalmente para los atributos continuos. Dependiendo de la distribución de los ejemplos en la base de datos con la que se esté trabajando, conviene utilizar una u otra representación; aunque, dado que no se presupone conocimiento alguno acerca de la distribución de los datos, la herramienta se debería aplicar con cada representación, y elegir la más satisfactoria.

El algoritmo de COGITO se ajusta a un típico modelo evolutivo de cubrimiento secuencial [Mitch97]. Selecciona el mejor individuo del proceso evolutivo, transformándolo en una regla, la cual es utilizada para eliminar ejemplos del conjunto de datos [Ventu93]. De esta forma, el conjunto de entrenamiento se reduce para la siguiente iteración. El criterio de finalización se alcanza cuando no existen más ejemplos que cubrir en el conjunto de datos.

Algunas de aportaciones más recientes en la obtención de reglas de decisión utilizando algoritmos genéticos son [Rique00], [Aguil02], [Giral03] y [Aguil03].

3.3.4 Clustering

De forma general, el *Clustering* es la tarea de organizar un conjunto de objetos en grupos significativos. Estos grupos pueden ser disjuntos, solapados u organizados jerárquicamente.

Los algoritmos clásicos de *clustering* generan una partición en el conjunto de datos en estudio, de forma que cada dato del conjunto es asignado exactamente a un *cluster*. En otras palabras, dados n elementos en un espacio métrico d-dimensional, tales elementos se agrupan en k-clusters de forma que se optimice un cierto criterio de similitud, que normalmente es algún tipo de distancia. Según Jain y Dubes [Jain88] el

clustering es el proceso por el que se descubren relaciones y distribuciones de interés en un conjunto de datos.

Los algoritmos de *clustering* se pueden clasificar de varias formas, dependiendo del aspecto que estemos teniendo en cuenta a la hora de obtener los clusters. Los principales criterios de clasificación son:

- Según l*a metodología básica del algoritmo, t*enemos algoritmos:

  - *Particionales*: Los algoritmos particionales, como son: K-means [MacQu67], [Jain88], K-medoids [Kaufm90], [NgHan94], Probabilísticos [Chees96] o espectrales [Boley98], encuentran los clusters particionando el conjunto completo de datos en un predeterminado o calculado algorítmicamente número de clusters.
    Los algoritmos particionales se pueden considerar como un procedimiento de optimización que intenta crear clusters de calidad según una determinada función objetivo. En [Duda01] y [Zhao01] se proponen varias de estas funciones.

  - *Aglomerativos*: Inicialmente asignan cada objeto a su propio cluster, y a continuación fusiona repetidamente pares de clusters según algún criterio de afinidad o cercanía, hasta que se alcanza el criterio de parada que se establezca (normalmente un número de clusters determinado). Los criterios básicos para determinar qué par de clusters se fusionan son: enlace-simple [Sneat73], enlace-completo [King67] y grupo-promedio (UPGMA) [Jain88]. Los principales algoritmos aglomerativos en la actualidad son: CURE [Guha98], ROCK [Guha99] y CHAMELEON [Karyp99].
    Los algoritmos aglomerativos, se suelen clasificar a su vez en: Aglomerativos Jerárquicos y No Jerárquicos, dependiendo de la estructura de la solución proporcionada.

- Según l*as características del espacio en el que se está operando*, tenemos los enfoques:

  - *Basado en Característica*: Cada objeto del espacio de trabajo se representa como un vector multidimensional de características, y la solución del clustering se obtiene optimizando iterativamente la similitud (distancia) entre cada objeto y el centroide de su cluster.
    Recientes estudios en el contexto del clustering en conjuntos de datos muy grandes y altamente dimensionales [Aggar99] [Bjorn99] [Stein00], muestran las ventajas de estos algoritmos sobre los basados en analogía.

- *Basado en Analogía*: Calculan la solución del clustering, calculando en primer lugar la similitud entre cada uno de los pares de objetos del espacio, para luego calcular en función de estas similitudes la solución global. Pertenecen a esta categoría los algoritmos aglomerativos jerárquicos [Enrig00], y el K-medoid.

- Según e*l tipo de clusters que descubren*, tenemos métodos:

  - *Globulares:* Los clusters estarán formados por objetos en los que la analogía entre todos los pares de objetos del cluster es muy alta. El algoritmo K-means (y sus variantes) y los aglomerativos utilizando enlace-completo son apropiados para obtener este tipo de clusters.

  - *Transitivos*: En estos clusters habrá muchos objetos cuya analogía directa entre pares de ellos será muy baja, pero estos objetos estarán conectados por diversos caminos formados dentro del cluster a través de fronteras de alta analogía. Los aglomerativos utilizando enlace-simple son apropiados para obtener este tipo de clusters.

  Recientemente se han desarrollado algoritmos especializados, denominados *subspace*, para encontrar los dos tipos de clusters (globulares y transitivos), operando directamente en el espacio de características de los objetos [Agraw98] [Burd01] [Nages99].

En muchos algoritmos de clustering, cada cluster se define a partir de ciertos datos representativos, como puede ser el *centroide*, que permiten medir la similitud entre clusters. Estos algoritmos generan clusters de forma esférica. En otros, cada cluster se define por una serie de datos o por todos los datos; de esta forma se producen clusters de forma arbitraria.

Dos recientes estudios [Jain99], [Han01], ofrecen un completo sumario sobre las distintos algoritmos y aplicaciones de *clustering*. Por otra parte, una de las últimas aportaciones respecto al conocido algoritmo K-means se realiza en [Kanug02], donde se propone una eficiente implementación del algoritmo de clustering k-means de Lloyd.

3.3.4.1 Clustering Borroso: Algoritmo FCM

La teoría de los conjuntos borrosos (Apéndice A), es un área de activa investigación, con un origen fuertemente matemático. Los defensores de esta metodología creen que la lógica borrosa puede proporcionar fundamentos robustos y consistentes para el proceso

de información incluyendo el reconocimiento de patrones y la construcción de sistemas de aprendizaje.

Existen al menos tres circunstancias en las que los conceptos y técnicas de los conjuntos borrosos son útiles en el *Reconocimiento de Patrones*:

1. Si algunas de las características de la población a clasificar son simbólicas o no numéricas, un conjunto borroso puede servir de interface entre esta característica lingüística y medidas cuantitativas.
2. La utilización de la función de pertenencia como discriminador de si un elemento pertenece o no a una clase (definida como conjunto borroso), es algo más ambigua que los clasificadores clásicos, pero la clasificación borrosa, al asignar a cada caso un valor de pertenencia (posibilidad) para cada clase, proporcionan más información que una simple asignación a una determinada clase.
3. Si se tienen casos con información incompleta, la función de pertenencia puede proporcionar un estimador de ese conocimiento incompleto.

Desde este punto de vista, la lógica borrosa juega dos importantes papeles en el *reconocimiento de patrones*:

a. Papel de Interface

Interface entre construcciones realizadas en base a variables lingüísticas, cercanas al hombre, y caracterizaciones cuantitativas apropiadas para las máquinas. En este sentido una característica o elemento de clasificación será definida como un conjunto borroso, es decir, es determinada en términos de una función de pertenencia. Tal característica es definida en el dominio de alguna variable medible y el valor de la función de pertenencia indica hasta que punto un valor de la variable es compatible con el concepto de esa característica. Los valores de la función de pertenencia deben ser interpretados más en términos de distribución de posibilidad que de probabilidad.

Así el concepto de que un parámetro sea “razonablemente alto”, es algo impreciso y debería poder utilizarse como tal, no sólo porque es una forma de representar el conocimiento humano, sino también porque los humanos razonan con este tipo de conceptos. En la teoría de los conjuntos borrosos se utilizan conceptos como “razonablemente alto” como una cantidad imprecisa y el parámetro correspondiente es definido como una variable en un dominio de medida determinista. Para ello se proporciona una interface entre las dos vistas definiendo “razonablemente alto” como un conjunto borroso en el dominio de un parámetro con el uso de una función de pertenencia.

De esta forma si X es una colección de objetos de x, un conjunto borroso A en X se define como un conjunto de pares ordenados:

$$A = \{(x, \mu_A(x)) / x \in X\}$$

La entidad $\mu_A(x)$ llamada *función de pertenencia*, es el valor que representa el grado de pertenencia de *x* en *A*. Usualmente la función de pertenencia se representa normalizada, es decir,: $\mu_A : X \rightarrow [0,1]$

b. Papel de la Distribución de Posibilidad

Permite una correcta interpretación de los conceptos imprecisos, es decir, los conjuntos borrosos proporcionan una herramienta para interpretar algunas distribuciones que difícilmente se podrían justificar con la teoría de la probabilidad clásica.

En los sistemas tradicionales de clasificación o de reconocimiento de patrones, se debe decidir sobre si un elemento estaba en la clase *i* o en la *j*. Esto comúnmente consiste en usar una función de decisión $g_i(x)$ para cada clase *i*, decidiendo que x es de la clase *i* si y sólo si $g_i(x) > g_j(x)\ \forall\ j \neq i$.

La lógica borrosa introduce la noción de posibilidad para modificar el concepto de pertenencia a una clase. La idea no es preguntar si x pertenece a la clase *i*, sino preguntar cuál es el grado de pertenencia que x tiene a cada clase *i*, para todo i. La pertenencia a una clase se convierte en un conjunto borroso sobre el dominio de todos los posibles patrones. El valor de la función de pertenencia es numéricamente igual al valor de una función de distribución de posibilidad que expresa la posibilidad de que el caso x pueda ser incluido en la clase *i*.

Zadeh en [Zadeh78] recomienda el punto de vista de la distribución de posibilidad y discute las diferencias con el concepto de probabilidad. En algunos aspectos, la posibilidad constituye una cota superior de la probabilidad: lo que es imposible es también improbable, pero lo que es posible no siempre es probable.

La extensión de las operaciones y conceptos matemáticos a los conjuntos borroso se debe, principalmente, a Zadeh en [Zadeh75] y a Dubois y Prade en [Duboi80].

Los algoritmos clásicos de *clustering* generan una partición de la población de forma que cada caso es asignado exactamente a un *cluster*. El algoritmo *Isodata* [Ball66] fue modificado por [Dunn74] y generalizado en [Bezde81] hasta convertirlo en un algoritmo de C*lustering Borroso* de propósito general, denominado algoritmo de la C-medias borrosas, y conocido por sus siglas en ingles como FCM (Fuzzy C-means Method).

Básicamente, este algoritmo intenta clasificar *n* elementos $x_k \in X$ con $1 \leq k \leq n$, con p características cada uno, es decir, $X \subset \Re^p$, en *c* clusters borrosos, asignando una función de pertenencia μ_{ik}:

$$\mu_{ik}:\ X \rightarrow [0,1]\ \forall i\ \ 1 \leq\ i \leq C,\ \ \forall k\ 1 \leq\ k \leq n$$

para ello, trata de minimizar la función J_m definida:

$$J_m \text{ (U, P:X)} = \sum_{k=1}^{n}\sum_{i=1}^{c}(\mu_{ik})^m D_{ik,A}^2 \tag{7}$$

donde :

- m > 1 es un exponente de peso sobre cada pertenencia borrosa
- U es la matriz μ_{ik} (Matriz de pertenencia)
- P = (v_1, v_2, ..., v_c) son *c* vectores de $\Re^p$, que son los centroides de los clusters
- A es una matriz p x p definida positiva
- $D_{ik,A}$ es la distancia asociada a la norma A de las x_k a v_i (8):

$$D_{ik,A} = \| x_k - v_i \|_A = \sqrt{(x_k - v_i)^t A(x_k - v_i)} \tag{8}$$

Las *c* particiones del conjunto X obtenidas por la aplicación de un algoritmo de clustering se caracterizan como conjuntos de $c\times n$ valores $\{\mu_{ik}\}$ que satisfacen las siguientes restricciones:

a) $0 \le \mu_{ik} \le 1, \quad 1 \le i \le c, \quad 1 \le k \le n$

b) $0 \le \sum_{k=1}^{n} \mu_{ik} \le n, \quad 1 \le i \le C$

c) $\sum_{i=1}^{c} \mu_{ik} = 1, \quad 1 \le k \le n$

Las condiciones necesarias para minimizar aproximadamente J_m son conocidas, (9) y (10):

$$\mu_{ik} = \left[\sum \left[\frac{\| x_k - v_i \|_A}{\| x_k - v_j \|_A} \right]^{\frac{2}{m-1}} \right]^{-1} \forall i, k \tag{9}$$

$$v_i = \frac{\sum_{k=1}^{n} (\mu_{ik})^m x_k}{\sum_{k=1}^{n} (\mu_{ik})^m} \forall i \tag{10}$$

3.3.4.1.1 Algoritmo FCM

La descripción del *algoritmo FCM* de clustering borroso es la siguiente:
Dado un conjunto de elementos x_k , $1 \leq k \leq n$, se clasifica en un cierto número de clusters c. Un cluster borroso está caracterizado por μ_{ik}, que muestra el grado de pertenencia del dato k-ésimo, al cluster i-ésimo. Sabemos que se debe cumplir que:

$$\sum_{i=1}^{c} \mu_{ik} = 1, \forall k$$

Se define una matriz U con los μ_{ik}; el problema ahora es encontrar c y determinar U.

El algoritmo es como sigue:

1- Fijar c =2, un valor inicial $U^{(1)}$ de U y l =1.
2- Calcular el centro v_i del cluster borroso por:

$$v_i = \sum_{k=1}^{n} (\mu_{ik})^m x_k \Big/ \sum_{k=1}^{n} (\mu_{ik}), 1 \leq i \leq c$$

Se define la distancia del *k-ésimo* dato al centro del *i-ésimo* cluster como:

$$d_{ik} = \| x_k - v_i \|$$

3- Calcular el elemento de un nuevo $U^{(l)}$ como sigue, para $l = l+1$:

$$I_k \overset{\Delta}{=} \{ i \mid 1 \leq i \leq c; d_{ik} = \| x_k - v_i \| = 0 \}$$

$$\overline{I_k} \overset{\Delta}{=} \{1,2,.....,c\} - I_k$$

$$I_k \neq \square \Rightarrow \mu_{ik} = 0, \forall i \in \overline{I_k} \wedge \sum_{i \in Ik} \mu_{ik} = 1$$

$$I_k = \square \Rightarrow \mu_{ik} = 1 \Big/ \left[\sum_{j=1}^{c} (d_{ik} / d_{jk})^{2/(m-1)} \right]$$

4- Si $\|U^{(l-1)} - U^{(l)}\| \leq \varepsilon$, entonces parar; en otro caso ir al paso 2.

En este algoritmo, *m* es un parámetro ajustable, que normalmente se suele establecer a 2. Para fijar el número de clusters, se utiliza el criterio propuesto por [Fukuy89], consistente en minimizar S(c) según la siguiente expresión (11):

$$S(c)= \sum_{k=1}^{n}\sum_{i=1}^{c}(\mu_{ik})^m(\| x_k - v_i \|^2 - \| v_i - \bar{x} \|^2) \qquad (11)$$

Donde:

n: Número de datos
c: Número de clusters, $c \geq 2$
x_k: Registro k-ésimo de la Base de datos, $1 \leq k \leq n$
$\bar{x}$: Vector media de los datos $x_1, x_2, ..., x_n$
v_i: Vector que expresa el centro del cluster i-ésimo
μ_{ik}: Grado de pertenencia del dato k-ésimo al cluster i
m: Peso ajustable entre 1.5 y 3

El primer término de la diferencia de S(c) mide la varianza de los datos en un cluster y el segundo término la varianza entre clusters. De esta forma, el número de clusters óptimo es aquel que minimiza la varianza en cada cluster y maximiza la varianza entre ellos.

El modelo borroso que se obtiene, una vez optimizado, puede proporcionar un modelo cualitativo mediante una aproximación lingüística. Para ello se ajustan los conjuntos borrosos obtenidos con unos conjuntos borrosos predefinidos en base a términos, modificadores (adjetivos) y conectivos (conjunciones) lingüísticos.

3.3.4.1.2 Aproximación de Sugeno y Yasukawa

El algoritmo FCM descrito en el apartado anterior fue utilizado por Sugeno y Yasukawa [Sugen93] para construir un modelo borroso a base de reglas de la forma (12):

$$R^i\text{: Si } x_i \in A^i \text{ entonces } y \in B^i \qquad (12)$$

donde:

- $x=(x_1, x_2, x_3, ..., x_n) \in \Re^n$ son los parámetros de entrada
- $A=(A_1, A_2, A_3, ..., A_n)$ son n conjuntos borrosos
- $y \in \Re$ es la variable de salida
- B es un conjunto borroso para la variable de salida

La metodología propuesta por Sugeno y Yasukawa consiste básicamente en aplicar la partición borrosa FCM a la variable de salida. Como resultado de este proceso, se obtiene el grado de pertenencia de cada uno de los datos de salida a cada uno de los conjuntos borrosos B^i.

Una vez obtenida una partición del espacio de salida en clusters borrosos B^i, se realiza una proyección de estos clusters sobre el espacio de entrada obteniendo un conjunto borroso en $\Re^n$, que proyectado sobre cada eje asigna a cada parámetro x_i un conjunto borroso A^i según la ecuación 12 (figura 3.5).

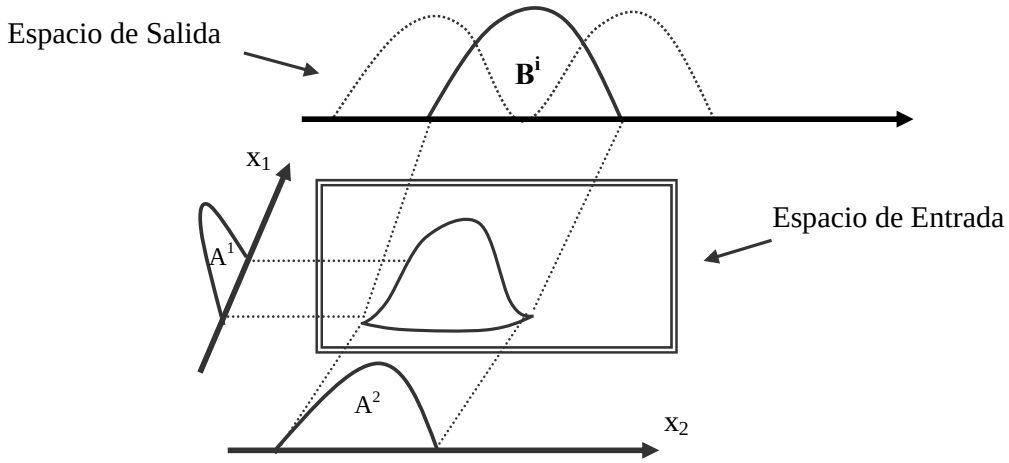


Figura 3.5: Proyección de clusters borrosos

A pesar de que los clusters B^i son convexos, puede que el que se obtenga por proyección en alguna característica de entrada x_i no lo sea; en este caso, el conjunto borroso A^i obtenido, se aproxima por un conjunto borroso convexo que represente al conjunto original.

Otro problema que puede aparecer, es que en el espacio de entrada existan más de dos conjuntos borrosos que correspondan al mismo conjunto borroso B^i del espacio de salida. En este caso, al realizar la proyección sobre cada parámetro de entrada x_i, se producirá una múltiple proyección, obteniendo más de un conjunto borroso A^i en cada parámetro. En este caso (figura 3.6) se formarán varias reglas borrosas con distinto antecedente y el mismo consecuente B^i, como por ejemplo:

$$R^1\text{: Si } x_1 \in A^{11} \ \underline{\text{y}}\ x_2 \in A^{21} \ \underline{\text{entonces}}\ \ y \in B^i$$
$$R^2\text{: Si } x_1 \in A^{12} \ \underline{\text{y}}\ x_2 \in A^{22} \ \underline{\text{entonces}}\ \ y \in B^i$$

A partir de este modelo inicial obtenido, se realiza un refinamiento intentando minimizar la suma del cuadrado de los errores que se cometen al intentar estimar el valor de la variable de salida con el valor propuesto por el modelo. Este refinamiento se

produce por un ajuste de los parámetros que definen los conjuntos borrosos para cada parámetro, según un procedimiento secuencial que actúa repetidamente sobre cada uno de los atributos de cada cluster borroso asociado a cada parámetro de entrada x_i, modificándolo en una cantidad constante y estudiando si el error se minimiza, en cuyo caso el parámetro queda modificado, obteniéndose así un modelo borroso a partir de la base de datos.

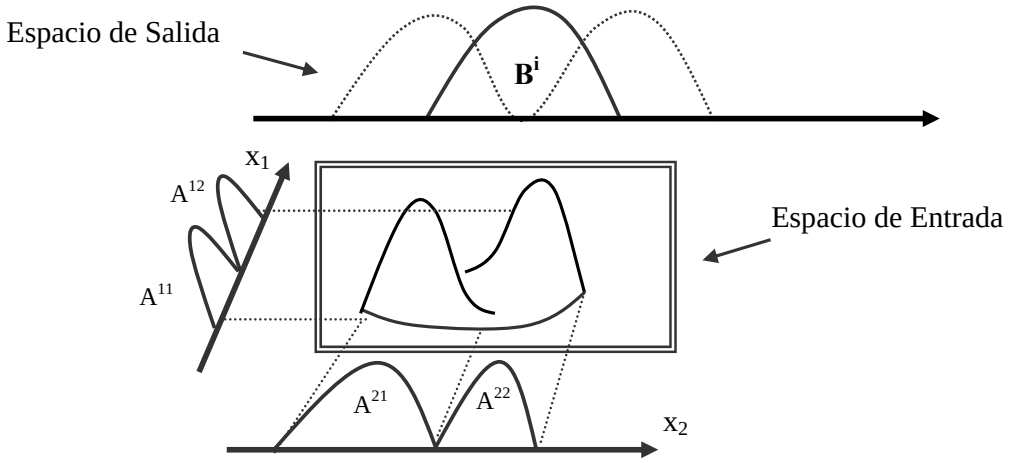


Figura 3.6: Múltiple proyección de clusters borrosos

3.3.5 Técnicas basadas en los k-vecinos más cercanos

El paradigma del vecino más cercano ha sido objeto de numerosos estudios en las últimas tres décadas. Dentro del Aprendizaje Automático, este algoritmo aparece como uno de los más utilizados [Cost93], [Cover67], [Cover68], [Dasar91], [Duda73]. El criterio de aprendizaje de éste se basa en que los miembros de una población suelen convivir (pudiendo llegar a coexistir) rodeados de individuos similares, con características y propiedades parecidas; bajo esta hipótesis, cualquier información descriptiva de un elemento, puede extraerse mediante la observación de sus vecinos más cercanos. El poder del aprendizaje por vecindad ha sido demostrado en infinidad de problemas reales, tales como la correcta pronunciación de palabras.

Una aplicación directa de este método es la utilización del mismo como técnica de clasificación. El conjunto completo de entrenamiento se almacena en memoria y para clasificar un *nuevo* ejemplo se calcula la distancia entre dicho ejemplo y cada uno de los almacenados del conjunto de entrenamiento, de forma que se asignará al nuevo ejemplo la clase del vecino más cercano. De forma más general se calculan los *k-vecinos* más

cercanos, y se asigna al nuevo ejemplo la clase más frecuente de entre los *k*-vecinos. A diferencia de otros modelos predictivos, el conjunto de entrenamiento no es procesado para crear un modelo, sino que dicho conjunto es el modelo.

Los sistemas de clasificación basados en los vecinos más cercanos utilizan como modelo el propio conjunto de datos y clasifican en función de las instancias que son más similares. Con este método, el espacio de búsqueda queda dividido en regiones en forma de teselación de Voronoi con fronteras polinomiales [Prepa85], que tienden a ser más irregulares a medida que crece el número de instancias.

En el algoritmo de los k-vecinos más cercanos o *k*-NN (*k*-Nearest Neighbor), hay dos parámetros básicos que hay que definir:

- El número de ejemplos cercanos que van a ser analizados (*k*)
- Una métrica que indique que significa *más cercano*.

Cada uso del algoritmo *k*-NN requiere la especificación de un valor entero positivo para *k*. Esto determina cuántos ejemplos serán examinados al predecir sobre un nuevo ejemplo. La selección de una métrica es a la vez arbitraria, ya que no hay criterio establecido de qué es una buena métrica, y muy importante, ya que la predicción depende totalmente de ella. De hecho, diferentes métricas utilizadas sobre el mismo conjunto de entrenamiento pueden producir diferentes predicciones.

Respecto al valor óptimo de *k*, hay diversas opiniones. El mejor valor exige experimentación y cotejar los resultados para distintos valores, ya que depende del conjunto de entrenamiento (algunos algoritmos comerciales la fijan a 10). En este sentido, un valor óptimo puede estimarse mediante la validación cruzada dejando uno fuera como puede comprobarse en [Weiss89]. Por su parte, Wettschereck [Wetts95] propone tres métodos para elegir un buen valor de *k*. En un reciente estudio, Aguilar y Ferrer [Aguil00] proponen la posibilidad de que el valor de *k* sea dinámico, es decir, que dependiendo de la región del espacio en donde se encuentre el nuevo ejemplo a clasificar, el valor de *k* cambie. Para ello, obtienen un árbol de decisión utilizando la herramienta C4.5, el cual precisa qué valor de *k* debe adoptarse en función de los valores de los atributos.

Otras aproximaciones de este algoritmo asignan pesos a los *k* vecinos; en ellas se utilizan medidas ponderadas inversamente proporcionales a las distancias normalizadas. Dudani [Dudan76] introduce la evaluación de los *k* vecinos con pesos asociados, de forma que el vecino más cercano tenga más peso o influencia que un vecino más lejano. La forma de calcular los pesos es la siguiente: para el ejemplo en cuestión se calculan los *k*-vecinos y se ordenan crecientemente por la distancia. Cada uno tendrá un peso asociado en función del orden de vecindad, que se calcula según la ecuación (13):

$$w_j = \begin{cases} \dfrac{d_k - d_j}{d_j - d_1}, & d_k \neq d_1 \\ 1, & d_k = d_1 \end{cases} \tag{13}$$

No obstante, el mismo Dudani indica que cuando *n*, es decir, el número de elementos del conjunto de entrenamiento, es muy grande los resultados son muy similares a los obtenidos con *k*-NN.

A pesar de la simplicidad de implementación, en la práctica, la probabilidad de error para la regla obtenida por el vecino más cercano es suficientemente cercana a la probabilidad de error (óptima) de Bayes. Cover y Hart [Cover67] han demostrado que cuando el tamaño del conjunto de entrenamiento tiende a infinito, el error asintótico del vecino más cercano no supera jamás al doble del error de Bayes (óptimo). Esto puede interpretarse como que la mitad de toda la información que facilita una muestra infinita está contenida en la regla del vecino más próximo.

En los últimos años, muchos investigadores se han dedicado a la búsqueda de soluciones eficientes al problema de los *vecinos más cercanos;* algunos de los trabajos más recientes en este campo son el de Lin y Yang [Lin01] que proponen una estructura basada en índices (el árbol ANN) para resolver el problema, y el trabajo de Maneewongvatana [Manee01], que propone un estudio empírico de una nueva aproximación para resolver el problema, en la que se devuelve el vecino más cercano con una cierta probabilidad fijada.

3.4 Conclusiones

En los apartados anteriores se han descrito las distintas técnicas estadísticas y de Minería de Datos que se van a utilizar en los siguientes capítulos de esta tesis.

Las técnicas y herramientas de Minería de Datos analizadas, se van a utilizar para abordar el problema de la toma de decisiones en PDS. Las herramientas que se van a utilizar son C4.5 y COGITO, que clasifican un conjunto de datos con el mínimo error posible, en el contexto del aprendizaje supervisado, lo que obliga a que la base de datos con la que se trabaje debe estar etiquetada.

Por otro lado, dentro de las técnicas de clustering descritas, nos vamos a centrar en el clustering borroso; en este sentido, en esta tesis se propone una herramienta basada en el algoritmo FCM, que trabaja en el contexto del aprendizaje no supervisado, lo que implica que las bases de datos no deben estar etiquetadas, lo cual, es de entrada una gran ventaja, ya que el proceso de etiquetado depende en demasiada medida de la experiencia del experto.

Las técnicas estadísticas descritas, serán utilizadas para abordar el problema de la estimación en PDS, de forma que a partir de la información proporcionada por una base de datos de PDS concreta, podamos estimar los valores de las variables del proyecto para nuevos valores en los atributos, con el menor error posible.

Capítulo 4

PreFuRGe
Predictive Fuzzy Rules Generator

4.1 Introducción

La motivación básica de esta investigación es la obtención de reglas de decisión a partir de bases de datos numéricas multiparamétricas no etiquetadas, que expresen el comportamiento de los distintos atributos, en el formato más cualitativo posible. Cabe destacar el detalle de que la base de datos esté no etiquetada, ya que como se describió en el capítulo anterior, este es un requisito necesario para poder aplicar herramientas encuadradas en el contexto del *Aprendizaje Supervisado*. El etiquetado de los registros de una base de datos es un proceso que suele realizarse de forma manual basado normalmente en la experiencia de algún experto, y que influye de forma determinante en las reglas que se obtengan.

En este sentido, uno de los principales objetivos de esta investigación se centra en desarrollar una familia de algoritmos, en lo sucesivo PreFuRGe (*Predictive Fuzzy Rules Generator*), que partiendo de una base de datos multiparamétrica, no etiquetada, compuesta por atributos numéricos y con más de una variable de salida (dependientes de los atributos), proporcione un modelo de reglas de decisión borrosas en formato

gráfico y en lenguaje natural [Aroba00], [Ramos01], que expresen, de forma cualitativa, el comportamiento de los distintos atributos para obtener todas las posibles combinaciones de valores (de las que se disponga información en la base de datos) en las variables, de manera simultánea.

Además, una vez obtenido de la base de datos, el modelo cualitativo de comportamiento de los distintos atributos, es posible predecir el valor de las distintas variables a partir de nuevos valores en los atributos, no almacenados en la base de datos.

4.2 La herramienta PreFuRGe

PreFuRGe es una herramienta de Minería de Datos basada en el algoritmo de fuzzy clustering FCM+, por lo que se encuadra en el contexto del *Aprendizaje no Supervisado*, lo que significa que no es necesario que cada registro de la base de datos que se vaya a utilizar para obtener las reglas borrosas tenga definido una etiqueta que determine su clase, como sí ocurre con las herramientas C4.5 o COGITO, descritas en el capítulo anterior. Este detalle es de gran importancia, ya que como se ha comentado anteriormente, este proceso suele hacerlo un experto de forma manual utilizando criterios basados en la experiencia que influyen de forma decisiva en las reglas que se obtengan.

4.2.1 Algoritmo FCM+

El algoritmo de fuzzy clustering propuesto, que se aplicará a bases de datos cuantitativos fin de obtener reglas de gestión borrosas, está basado en el algoritmo FCM (Fuzzy C-Means Method) [Bezde81] descrito en la sección 3.3.4.1. del capítulo anterior.

FCM+ es un algoritmo de fuzzy clustering fundamentado en la metodología propuesta por Sugeno y Yasukawa, descrita en detalle en la sección 3.3.4.1.2 del capítulo anterior, que se aplica a bases de datos cuantitativas con una solo parámetro de salida, y que consiste básicamente en aplicar la partición borrosa FCM a la variable de salida y; una vez obtenida una partición del espacio de salida en clusters borrosos, se realiza una proyección de estos clusters sobre el espacio de entrada obteniendo un conjunto borroso en $\Re^n$, que proyectado sobre cada eje asigna a cada atributo un conjunto borroso.

Esta metodología ha sido modificada y adaptada, con el fin de que pueda ser aplicada a bases de datos cuantitativas multiparamétricas con más de una variable de salida.

Las principales modificaciones realizadas sobre la propuesta de Sugeno y Yasukawa son:

- El algoritmo puede trabajar bases de datos cuantitativas multiparamétricas, con más de una variable de salida; estas variables son procesadas simultáneamente, lo que implica el espacio de salida no está restringido a una única variable como ocurre en la propuesta de Sugeno y Yasukawa.
- Las diferentes variables del proyecto pueden ponderarse, asignándoles pesos para el cálculo de las distancias entre los puntos del espacio que se está particionando.
- Las variables se homogeneizan en el rango de valores [0,1], para evitar que la diferencia de los rangos en que se pueda mover cada una de ellas, distorsione el cálculo de distancias.

4.2.2 Características de PreFuRGe

Las principales características de PreFuRGe son:

- **C1.** Trabaja con bases de datos cuantitativas con múltiples atributos y variables (dependientes de los atributos); además las distintas variables son procesadas de manera simultánea.
- **C2.** Las diferentes variables de la base de datos pueden ponderarse con pesos, lo que permite obtener unas reglas de decisión borrosas mucho más fiables, ya que se le puede asignar un factor de relevancia en la base de datos diferente a cada atributo, que puede ser lo que ocurra en la realidad.
- **C3.** Los conjuntos borrosos obtenidos son procesados por otro algoritmo, para obtener unas aproximaciones poliédricas, en las reglas de decisión gráficas. Es decir, en vez de trabajar con los conjuntos borrosos, tal y como se obtienen con el algoritmo, hacemos una aproximación mediante trapecios de forma que la información que se proporcione sea más clara y siga teniendo el mismo significado (figura 4.1).

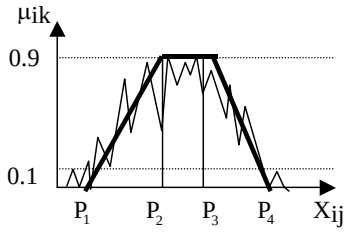


Figura 4.1: Aproximación de un conjunto borroso por un trapecio

El algoritmo para realizar esta aproximación se describe en el apartado 4.2.3.2 de este capítulo.

- **C4.** La salida de la herramienta está compuesta por una interface gráfica, que muestra en modo gráfico el conjunto de reglas de decisión obtenidas.

Un ejemplo de regla de decisión borrosa en modo gráfico, obtenida de una base de datos con dos atributos (A1, A2) y una variable (O), tiene el siguiente formato:

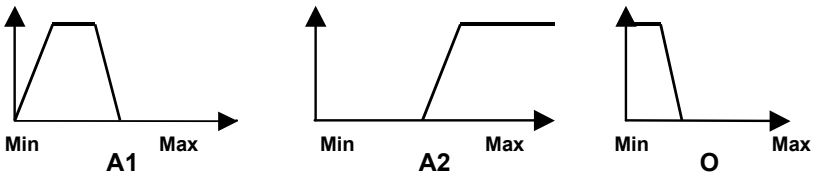


Figura 4.2: Ejemplo de regla de gestión borrosa

En esta regla el conjunto borroso asignado a cada parámetro de la base de datos, es representado por un trapecio. En el eje X de cada conjunto borroso se representan los valores del atributo en cuestión, y en el eje Y el valor de pertenencia a un Cluster.

- **C5.** El conjunto de reglas de decisión en modo gráfico obtenidas, son interpretadas en lenguaje natural.

La regla de decisión borrosa anterior sería interpretada por otro algoritmo (descrito en el apartado 4.2.3.3 de este capítulo), en lenguaje natural, de la siguiente forma:

Si *A1* es bajo **y** *A2* es medio o alto **Entonces** *O* es muy bajo

Al aplicar un PreFuRGe a bases de datos numéricas multiparamétricas, por la propia naturaleza de los algoritmos utilizados (fuzzy clustering), es posible obtener múltiples proyecciones en los parámetros de entrada (montículos).

En la regla borrosa mostrada a continuación se representa una múltiple proyección en el atributo *A1*. En este caso podemos ver (figura 4.3) como el atributo *A1* puede tomar diferentes tipos de valores para un determinado tipo de valor en la variable de salida (S).

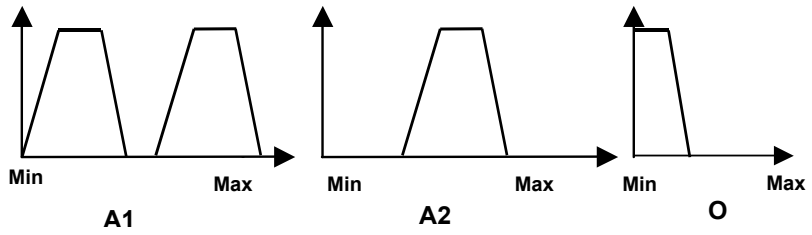


Figura 4.3: Ejemplo de regla de gestión borrosa con múltiples proyecciones en A1

Esta regla de decisión borrosa podría interpretarse de la siguiente forma:

Si *A1* es bajo **O** alto **y** *A2* es medio **Entonces** *S* es muy bajo

- **C6.** PreFuRGe está preparada para detectar múltiples proyecciones, que de no identificarse, podría hacer que la regla en cuestión proporcionase una información incorrecta.

Si no se hubiese detectado la múltiple proyección en el parámetro *A1* en el ejemplo anterior, la regla de decisión que se habría generado habría sido la siguiente (figura 4.4):

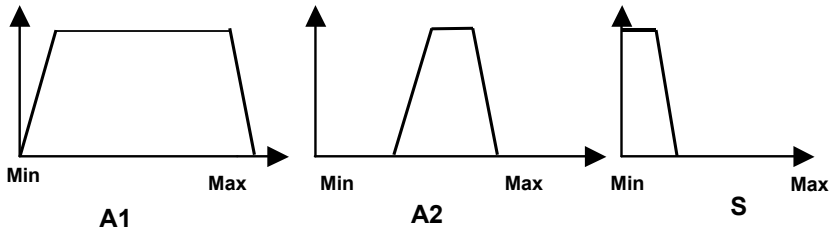


Figura 4.4: Ejemplo de regla de gestión borrosa sin múltiples proyecciones

La interpretación de esta regla borrosa es:

Si *A1* toma cualquier valor **y** *A2* es medio **Entonces** *S* es muy bajo

Como se puede ver en esta regla, *A1* deja de ser un atributo relevante, desde el punto de vista cualitativo, a la hora de ver el efecto sobre la variable *S* ya que, según se aprecia, *S* toma valores muy bajos sea cual sea el valor de *A1*. Por tanto la información cualitativa que se obtiene es totalmente diferente si se procesan o no las proyecciones múltiples detectadas.

4.2.3 Algoritmo PreFuRGe

La estructura general, en pseudocódigo, del algoritmo de la herramienta PreFuRGe se muestra en la figura 4.5. Se puede observar que básicamente esta dividido en cuatro subalgoritmos que realizan las principales tareas de la herramienta.

La descripción en secuencia de los distintos pasos del algoritmo es:

- *Paso 1*: Se aplica el algoritmo de clustering *FCM+* descrito anteriormente sobre cada uno de los datos x_{ij} de la base de datos que se está procesando. En este paso se particiona el espacio m-dimensional (m variables) de salida de la base de datos en *k* clusters, obteniendo el valor de pertenencia μ_{ik} de cada uno de los *i* vectores m-dimensionales de variables, a cada uno de los *k* clusters. A continuación, mediante proyección de estos clusters sobre el espacio de entrada n-dimensional (n-atributos), se obtienen los valores de pertenencia de cada dato x_{ij} del espacio de entrada, a cada uno de los *k* clusters obtenidos.
- *Paso 2*: Se aplica el algoritmo Mounds para procesar las múltiples proyecciones (montículos) que pueden aparecer en el espacio de entrada al realizar la proyección de clusters anteriormente descrita.
- *Paso 3*: Se procesan y generan las reglas de decisión en formato gráfico (figura 4.4), mediante el algoritmo *DrawRules*.
- *Paso 4*: Se realiza la interpretación, en lenguaje natural, de las reglas de decisión borrosas obtenidas en el paso 3, con el algoritmo *SayRules*.

La descripción de las distintas variables y constantes utilizadas en el algoritmo es la siguiente:

- $\mathbf{x_{ij}}$: Valor almacenado en el i-ésimo registro del atributo j-ésimo de la base de datos cuantitativa que se está procesando.
- **k**: Número de clusters en que se particiona el espacio de salida de la base de datos; el número de clusters determina el número de reglas de decisión borrosas que se obtengan.

- μ_{ik}: Grado de pertenencia del dato x_{ij} de la base de datos, al cluster *k*.
- **Pj_{ik}**: Punto j-ésimo de los cuatro que definen el conjunto borroso aproximado por un trapecio, asociado al parámetro *i* de la base de datos, en el cluster *k*. Si se detecta una múltiple proyección al proyectar el cluster *k* sobre el parámetro *i*, los puntos Pj_{ik} definen el conjunto borroso asociado a la 1ª de las proyecciones. En caso de no detectarse ninguna múltiple proyección en el parámetro *i*, el conjunto borroso definido por los puntos Pj_{ik} es el que se único que se asocia a este parámetro; en este caso no estarían definidos los puntos Mj_{ik}.
- **Mj_{ik}**: Punto j-ésimo de los cuatro que definen el conjunto borroso aproximado por un trapecio, correspondiente a la 2ª de las proyecciones detectadas al proyectar el cluster *k* en el parámetro *i-ésimo* del espacio de entrada. Estos puntos Mj_{ik} sólo se calculan en caso de detectar alguna múltiple proyección.

Hay que aclarar, que en el caso de detectar múltiples proyecciones, en esta investigación, se ha decidido identificar y procesar a lo sumo dos de ellas, ya que se comprobó que la identificación de tres o más proyecciones sobre un determinado parámetro no aportaba más información de tipo cualitativo sobre los valores que podía tomar éste, es más todo lo contrario, hacía aún más confusa la interpretación.

Algoritmo PreFuRGe

Entrada: Los valores x_{ij} de la base de datos (n-atributos y m-variables)
Salida: Conjunto de reglas de decisión borrosas en formato gráfico y en lenguaje natural

1- Algoritmo de fuzzy clustering → *FCM+* (x_{ij})
- División del espacio de salida en k-clusters y obtención de los valores de pertenencia μ_{ik} de cada uno de los *i* vectores m-dimensionales de variables, a cada uno de los *k* clusters
- Proyección de los k-clusters en el espacio de entrada, y obtención de los valores de pertenencia de cada punto de la base de datos a cada cluster k

2- Procesamiento de las múltiples proyecciones (montículos)→ *Mounds*(x_{ij}, μ_{ik})
3- Obtención de las Reglas de Gestión Borrosas en formato gráfico →
DrawRules(x_{ij}, μ_{ik}, $M1_{ik}$, $M2_{ik}$, $M3_{ik}$, $M4_{ik}$)
- Obtención de los puntos $P1_{ik}$, $P2_{ik}$, $P3_{ik}$, $P4_{ik}$
- Procesamiento de los puntos $P1_{ik}$, $P2_{ik}$, $P3_{ik}$, $P4_{ik}$, $M1_{ik}$, $M2_{ik}$, $M3_{ik}$, $M4_{ik}$

4- Interpretación de las Reglas de Gestión Borrosas en lenguaje natural →
SayRules($P1_{ik}$, $P2_{ik}$, $P3_{ik}$, $P4_{ik}$, $M1_{ik}$, $M2_{ik}$, $M3_{ik}$, $M4_{ik}$)

Figura 4.5: Pseudocódigo de PreFuRGe

4.2.3.1 Algoritmo Mounds

El algoritmo *Mounds* (figura 4.7) detecta si se producen proyecciones múltiples en el espacio de entrada e identifica los puntos $M1_{ik}$, $M2_{ik}$, $M3_{ik}$, $M4_{ik}$ que definen el conjunto borroso aproximado por un trapecio, de la *múltiple proyección* detectada al proyectar el cluster *k* en el parámetro i-ésimo del espacio de entrada.

El procedimiento seguido para detectar múltiples proyecciones en un determinado atributo del espacio de entrada para un cluster *k*, consiste en analizar la evolución de los valores de pertenencia μ_{ik} correspondiente (por proyección) a cada uno de los puntos del atributo *i*, en cada uno de los *k* clusters.

Si en un punto concreto x_{aj} del atributo que se está analizando, el valor de pertenencia μ_{ak} cambia de un valor anterior mayor que 0.9 a otro menor que 0.1, se marca dicho punto; si en un punto posterior x_{bj}, el valor de pertenencia μ_{bk} vuelve a ser mayor que 0.9, se determina la distancia de x_{aj} a x_{bj} y se observa si es mayor que un ε, conocido de antemano. Si se supera este ε, se considera que se ha detectado una múltiple proyección en el atributo (*es_Monticulo*=verdad), procediendo a identificar los puntos $M1_{ik}$, $M2_{ik}$, $M3_{ik}$, $M4_{ik}$ que lo definirán siguiendo el procedimiento descrito anteriormente para un conjunto borroso (figura 4.1). En esta investigación se considera un ε diferente para cada atributo, que se fija al 10% de la amplitud del rango de valores en que se mueva este; por ejemplo si los valores en la base de datos el atributo A1 se mueven entre 10 y 100, ε se fijará a 9.

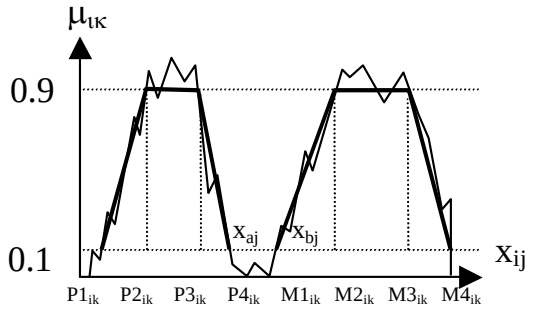


Figura 4.6: Proyección múltiple en el parámetro j-ésimo (cluster *k*)

En esta figura se observa una proyección múltiple en el parámetro j-ésimo en el cluster *k*, por lo que es necesaria la identificación de los puntos Pj_{ik} (conjunto borroso asociado a la 1ª proyección) y Mj_{ik} (conjunto borroso asociado a la 2ª proyección), que identifican a los conjuntos borrosos aproximados por trapecios, formados por ambas proyecciones.

Algoritmo Mounds

Entrada: Los valores x_{ij} de la Base de datos y los valores μ_{ik} resultado de aplicar FCM+

Salida: Los cuatro valores $M1_{ik}$, $M2_{ik}$, $M3_{ik}$, $M4_{ik}$ que definen cada conjunto borroso asociado a una 2ª proyección

Para cada parámetro de entrada j
  ε =10% (amplitud rango de valores del parámetro j)
  Para i=Min Hasta Max
    Para cada cluster *k*
      - Si en un punto x_{aj} la función de pertenencia μ_{ik} es < 0.1 y el punto anterior tenía un valor >0.9 Entonces Marcar_punto (x_{aj})
      - Si el valor de la función de pertenencia μ_{ik} vuelve a ser > 0.9 en un punto x_{bj} Entonces Si distancia (x_{aj} , x_{bj}) > ε Entonces *es_Montículo*=verdad
  - Si *es_Montículo* entonces calcular puntos $M1_{ik}$, $M2_{ik}$, $M3_{ik}$, $M4_{ik}$.

Figura 4.7: Pseudocódigo de Mounds

4.2.3.2 Algoritmo DrawRules

El algoritmo *DrawRules* (figura 4.10) calcula los puntos $P1_{ik}$, $P2_{ik}$, $P3_{ik}$, $P4_{ik}$ que definen el conjunto borroso aproximado por un trapecio, asociado a cada parámetro *i* de la base de datos, en cada cluster obtenido *k* (figura 4.8). A continuación, se realiza un procesamiento gráfico de los puntos obtenidos proporcionando una representación de cada una de las reglas obtenidas (*k* reglas = número de clusters), para lo que se tienen en cuenta también los puntos $M1_{ik}$, $M2_{ik}$, $M3_{ik}$, $M4_{ik}$ que definen el conjunto borroso aproximado por un trapecio, de la *múltiple proyección* detectada al proyectar el cluster *k* en el parámetro i-ésimo del espacio de entrada.

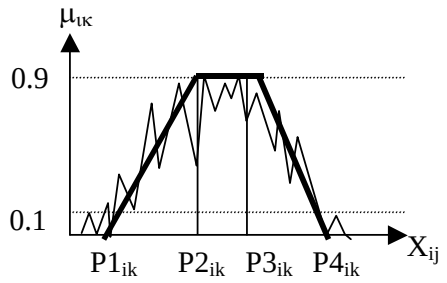


Figura 4.8: Puntos Pj_{ik} que definen un conjunto borroso

Un ejemplo de regla de gestión en modo gráfico obtenida con PreFuRGe se muestra en la figura 4.9. En esta regla intervienen 3 atributos (P1, P2, P3) y 2 variables (V1,V2); además se pueden observar las múltiples proyecciones detectadas en los atributos P1 y P2.

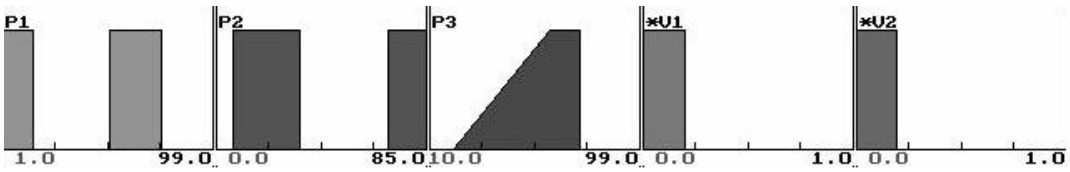


Figura 4.9: Regla de gestión en modo gráfico

El algoritmo *SayRules* que se describe posteriormente (apartado 3.2.3), realiza la interpretación de esta regla en lenguaje natural.

Algoritmo DrawRules

Entrada: - Los valores x_{ij} de la Base de datos
- Los valores μ_{ik} resultado de aplicar FCM+
- Los valores $M1_{ik}$, $M2_{ik}$, $M3_{ik}$, $M4_{ik}$ que definen cada conjunto borroso asociado a las 2ª proyecciones

Salida: - Los valores $P1_{ik}$, $P2_{ik}$, $P3_{ik}$, $P4_{ik}$ que definen cada conjunto borroso asociado a las 1ª proyecciones
- Conjunto de reglas de decisión borrosas en formato grafico

- Inicializar los puntos $P1_{ik}$, $P2_{ik}$, $P3_{ik}$, $P4_{ik}$
  <u>Para</u> cada parámetro j de entrada
    <u>Para</u> i=Min <u>Hasta</u> Max
      <u>Para</u> cada cluster k
        $P1_{ik}$: Min x_{ij} <u>con</u> μ_{ik} >0.1
        $P2_{ik}$: Min x_{ij} <u>con</u> μ_{ik} >0.9
        $P3_{ik}$: Max x_{ij} <u>con</u> μ_{ik} >0.9
        $P4_{ik}$: Max x_{ij} <u>con</u> μ_{ik} >0.1
- Procesamiento gráfico de los puntos:
  - $P1_{ik}$, $P2_{ik}$, $P3_{ik}$, $P4_{ik}$
    {Definen cada conjunto borroso asociado a la 1ª proyección}
  - $M1_{ik}$, $M2_{ik}$, $M3_{ik}$, $M4_{ik}$
    {Definen cada conjunto borroso asociado a la 2ª proyección}

Figura 4.10: Pseudocódigo de DrawRules

4.2.3.3 Algoritmo SayRules

Una vez obtenido un modelo borroso, el objetivo es obtener un modelo cualitativo mediante una aproximación lingüística a los clusters borrosos obtenidos. La técnica consiste en medir el grado de equiparación entre los conjuntos obtenidos (FCM+) y otros previamente definidos según las características de los parámetros del modelo.

El rango de valores de cada parámetro se ha dividido en tres clases que se han denominado pequeño, mediano y grande; para ello, se divide el intervalo de cada parámetro en varios intervalos borrosos, asignándole a cada uno un nombre (figura 4.11).

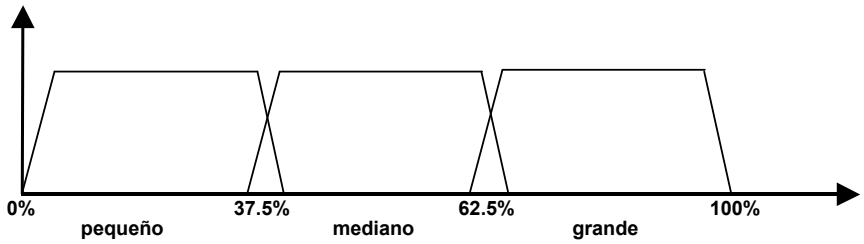


Figura 4.11: Posible equivalencia entre términos y conjuntos borrosos

Estos son los términos lingüísticos básicos, que pueden venir acompañados de modificadores para un mejor ajuste del modelo cualitativo (figura 4.12).

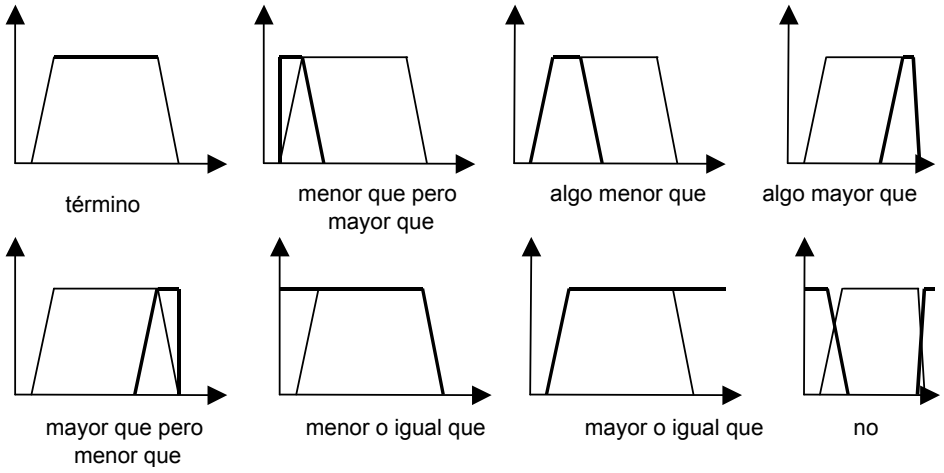


Figura 4.12: Modificadores de términos lingüísticos

Cada modificador actuando sobre un término da lugar a otro conjunto borroso definido por cuatro valores $Q1_{ik}$, $Q2_{ik}$, $Q3_{ik}$, $Q4_{ik}$, que identifica una determinada etiqueta

lingüística y que se puede calcular en base a ciertas expresiones en función de los cuatro valores $P1_{ik}$, $P2_{ik}$, $P3_{ik}$, $P4_{ik}$ que definen el término original. Por ejemplo el Conjunto borroso asociado a la etiqueta "*Algo menor que*", según la forma expresada en la figura 4.12, estaría definido de la siguiente forma:

$$Q1_{ik}= P1_{ik} \qquad Q2_{ik}= P2_{ik} \qquad Q3_{ik}= (3\times P2_{ik}+P3_{ik})/4 \qquad Q4_{ik}= (P2_{ik}+P3_{ik})/2 \qquad (1)$$

La medida de similitud entre dos conjuntos borrosos utilizada es la propuesta por Dubois [Duboi80], que establece que dados dos conjuntos borrosos *C1* y *C2*, su grado de similitud *S(C1, C2)* es medido por:

$$S(C1,C2) = \frac{\|C1 \cap C2\|}{\|C1 \cup C2\|} \qquad (2)$$

donde $\| \ \|$ indica el área de un conjunto borroso. Dado que los conjuntos borrosos están aproximados mediante trapecios, es fácil el cálculo de las áreas de la unión y la intersección. La metodología a seguir es comparar los conjuntos borrosos obtenidos en las reglas de decisión borrosas con FCM+, con las colección de conjuntos borrosos resultado de aplicar todos los modificadores posibles (figura 4.12) a los términos lingüísticos básicos (i.e.: (1)). Como resultado de esta comparación, se obtendrá un grado de similitud máxima entre cada conjunto borroso de las reglas obtenidas y un término lingüístico.

Algoritmo SayRules

Entrada: - Los puntos $P1_{ik}$, $P2_{ik}$, $P3_{ik}$, $P4_{ik}$ que definen cada conjunto borroso asociado a las 1ª proyecciones
- Los valores $M1_{ik}$, $M2_{ik}$, $M3_{ik}$, $M4_{ik}$ que definen los conjuntos borrosos asociados a las 2ª proyecciones

Salida: -Conjunto de reglas de gestión borrosas en lenguaje natural

- Definir las etiquetas borrosas y sus intervalos: calcular ($Q1_{ik}$, $Q2_{ik}$, $Q3_{ik}$, $Q4_{ik}$)
  Para cada parámetro j de entrada
    Para cada cluster k
    - Analizar grado de similitud *S* de cada conjunto borroso ($P1_{ik}$, $P2_{ik}$, $P3_{ik}$, $P4_{ik}$) con las etiquetas borrosas ($Q1_{ik}$, $Q2_{ik}$, $Q3_{ik}$, $Q4_{ik}$)
    - Asignar a cada conjunto borroso (1ª proyección) la mejor etiqueta {similitud}
    - Analizar grado de similitud *S* de cada 2ª proyección ($M1_{ik}$, $M2_{ik}$, $M3_{ik}$, $M4_{ik}$) con las etiquetas borrosas ($Q1_{ik}$, $Q2_{ik}$, $Q3_{ik}$, $Q4_{ik}$)
    - Asignar a cada conjunto borroso (2ª proyección) la mejor etiqueta {similitud}
- Procesar los datos obtenidos y escribir las reglas en lenguaje natural

Figura 4.13: Pseudocódigo de SayRules

La interpretación que el algoritmo *SayRules* (figura 4.13) realizaría de la regla mostrada en la figura 4.9 sería la siguiente:

> "**Si** *P1* es Bajo ***o*** Algo mayor que medio **y** *P2* es Algo menor que medio ***o*** Alto **y** *P3* es Medio **Entonces** *V1* es Bajo **y** *V2* es Bajo".

4.3 Validación de la Herramienta

El hecho de que el clustering borroso se ubique dentro del área del aprendizaje no supervisado, añade una dificultad adicional a la hora de evaluar los resultados, ya que no se dispone de criterios objetivos para evaluar la fiabilidad de las reglas de decisión que se obtengan.

Por tanto, para verificar de manera objetiva el correcto funcionamiento de la herramienta propuesta, es necesario conocer la distribución de los puntos dentro del conjunto de datos utilizado. Por ello, se han creado una serie de bases de datos artificiales utilizando distintas funciones para distribuir los puntos de la base de datos; en este sentido, los valores para cada parámetro se han seleccionado dentro de unos rangos determinados, e introduciendo anomalías como outliers o múltiples proyecciones, añadidas de forma controlada para poder detectar si la herramienta es capaz de procesar toda la información que la base de datos contiene.

4.3.1 Experimento 1

El primer experimento que se realizó fue con una base de datos con dos parámetros de salida y ninguno de entrada, es decir, en este caso no se quería hacer ninguna proyección de conjuntos borrosos sobre el espacio de entrada (como sería lo normal en el algoritmo FCM+), sino una clasificación de un espacio de salida multidimensional.

Las características de la base de datos en cuestión son:

- Número de registros: 400
- Número de parámetros: 2
  - Edad: Rango (20,80)
  - Sueldo: Rango (50000, 125000)

- Las reglas usadas para generar los distintos registros de la base de datos son:
  - Si Edad $\in$ (60,80) $\rightarrow$ Sueldo $\in$ (25000, 75000)
  - Si Edad $\in$ (40,60) $\rightarrow$ Sueldo $\in$ (75000, 125000)
  - Si Edad $\in$ (20,40) $\rightarrow$ Sueldo $\in$ (50000, 100000)

En esta base de datos, por tanto, lo que se relacionan en sus 400 registros son pares de datos (edad, sueldo). El objetivo es obtener información cualitativa de forma automática de ella, para que podamos tener conocimiento sobre que tipo (alto, bajo, medio, etc.) de sueldo ganan los trabajadores de la empresa en función de su edad.

Los resultados obtenidos con PreFuRGe al procesar esta base de datos son los mostrados en la figura 4.14. Se puede comprobar que estos resultados coinciden cualitativamente de forma exacta con las características de los datos que conforman la base de datos, y que eran conocidos a priori. Además la herramienta realiza una interpretación en lenguaje natural de las reglas borrosas en formato gráfico generadas, con lo que la información proporcionada es lo más cercana a la forma de razonar humana:

- **Regla 1:** Si la *Edad* es **Alta** Entonces el *Sueldo* es **Bajo**
- **Regla 2:** Si la *Edad* es **Media** Entonces el *Sueldo* es **Alto**
- **Regla 3:** Si la *Edad* es **Baja** Entonces el *Sueldo* es **Medio**

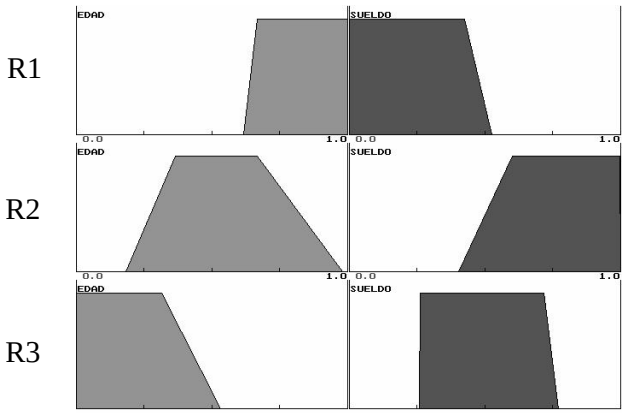


Figura 4.14: Resultados sobre BD artificial de sueldos / edades

Este conocimiento cualitativo generado por la herramienta sería muy difícil de obtener a partir de la observación de la base de datos.

4.3.2 Experimento 2

El siguiente conjunto de pruebas de PreFuRGe se realizó con bases de datos artificial también, pero con múltiples parámetros de entrada y salida. La distribución de los datos fue realizada a medida, para poder así detectar ciertas irregularidades (outliers

y/o múltiples proyecciones) que se introdujeron para chequear el correcto funcionamiento de la herramienta. Las características de esta base de datos son:

- Número de registros: 120
- Número de parámetros de entrada: 3
  - P1: Rango (1, 99)
  - P2: Rango (0, 85)
  - P3: Rango (10, 99)

- Número de parámetros de salida: 2
  - V1: Rango (10, 60)
  - V2: Rango (0, 125)

En esta base de datos se han generado los registros dándole valores aleatoriamente a cada parámetro, dentro de los rangos definidos anteriormente, de forma que estuviesen estructurados en base a 5 reglas de la siguiente forma:

- <u>Si</u> P1 ∈ (45, 60) <u>y</u> P2 ∈ (55, 85) <u>y</u> P3 ∈ (20, 35)
  <u>Entonces</u> V1∈ (30, 40) <u>y</u> V2 ∈ (50, 75)

- <u>Si</u> P1 ∈ (10, 30) <u>y</u> P2 ∈ (0, 30) <u>y</u> P3 ∈ (75, 100)
  <u>Entonces</u> V1∈ (50, 60) <u>y</u> V2 ∈ (100, 125)

- <u>Si</u> P1 ∈ (1, 15) <u>y</u> P2 ∈ (7, 35) <u>y</u> P3 ∈ (60, 75)
  <u>Entonces</u> V1∈ (10, 20) <u>y</u> V2 ∈ (0, 25)

- <u>Si</u> P1 ∈ (75, 100) <u>y</u> P2 ∈ (0, 20) <u>y</u> P3 ∈ (58, 60)
  <u>Entonces</u> V1∈ (40, 50) <u>y</u> V2 ∈ (75, 100)

- <u>Si</u> P1 ∈ (5, 30) <u>y</u> P2 ∈ (25, 40) <u>y</u> P3 ∈ (10, 30)
  <u>Entonces</u> V1∈ (20, 30) <u>y</u> V2 ∈ (25, 50)

La regla introducida para generar múltiples proyecciones en la tercera de las anteriores reglas es:

- <u>Si</u> P1 ∈ (50, 75) <u>y</u> P2 ∈ (70, 85) <u>y</u> P3 ∈ (60, 75)
  <u>Entonces</u> V1∈ (10, 20) <u>y</u> V2 ∈ (0, 25)

Los resultados obtenidos con PreFuRGe, al procesar esta base de datos son los mostrados en la figura 4.15. Se puede comprobar que los resultados obtenidos coinciden

cualitativamente con la distribución de los datos de la base de datos artificial utilizada, y que son conocidos a priori, tal y como se ha descrito en las reglas anteriores.

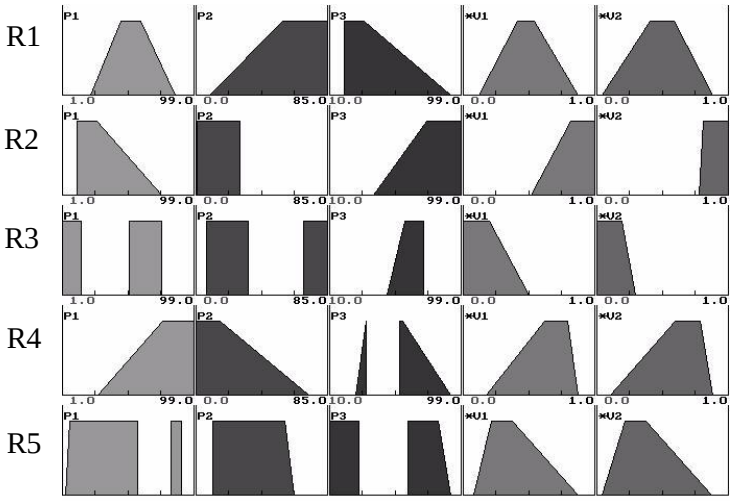


Figura 4.15: Resultados sobre BD artificial multiparamétrica

Además, como ya se ha comentado, la herramienta realiza una interpretación en lenguaje natural de las reglas borrosas en formato gráfico generadas, obteniendo una información cualitativa muy cercana a la forma de razonar humana:

- **Regla 1:** Si *P1* es **Medio** Y *P*2 es **Alto** Y *P3* es **Bajo**
  Entonces *V1* es **Medio** Y *V2* es **Medio**

- **Regla 2:** Si *P1* es **Bajo** Y *P*2 es **Bajo** Y *P3* es **Muy Alto**
  Entonces *V1* es **Muy Alto** Y *V2* es **Muy Alto**

- **Regla 3:** Si *P1* es **Bajo** O **Medio** Y *P*2 es **Bajo** O **Alto** Y *P3* es **Medio**
  Entonces *V1* es **Bajo** Y *V2* es **Bajo**

- **Regla 4:** Si *P1* es **Alto** Y *P*2 es **Bajo** Y *P3* es **Medio**
  Entonces *V1* es **Mayor que medio** Y *V2* es **Mayor que medio**

- **Regla 5:** Si *P1* es **Mayor que Bajo** O **Alto** Y *P*2 es **Medio** Y *P3* es **Bajo** O **Mayor que medio**
  Entonces *V1* es **Menor que medio** Y *V2* es **Menor que medio**

Se puede observar como la herramienta ha creado las reglas que debía, es decir, las cinco definidas, ajustándose de esta forma a la estructura que tenían los datos de la base de datos. Además ha detectado perfectamente cada una de las anomalías que se han introducido de forma intencionada:

- En la regla cuarta, en el parámetro P3 se observa que se ha detectado el pequeño conjunto borroso formado en el intervalo [50,60].

- En la regla tercera se observa cómo se han identificado correctamente las múltiples proyecciones en los parámetros P1 y P2, justamente en los intervalos en los que se crearon. Además de éstas, se han identificado otras múltiples proyecciones debido a que se han introducido una serie de registros aleatorios a modo de ruido.

- El resto de reglas obedece de forma fiel a la estructura que se ha fijado en la base de datos.

4.3.3 Experimento 3

El siguiente conjunto de pruebas de PreFuRGe se realizó con una base de datos, con múltiples parámetros de entrada y salida, donde, al igual que en el experimento anterior, los distintos registros han sido generados a medida, para poder comprobar el correcto funcionamiento de la herramienta. Las características de esta base de datos son:

- Número de registros: 150
- Número de parámetros de entrada: 4
    - P1: Rango (1, 50)
    - P2: Rango (5, 75)
    - P3: Rango (1, 60)
    - P4: Rango (5, 100)

- Número de parámetros de salida: 3
    - F1, F2, F3

En esta base de datos se han generado los registros dándole valores aleatoriamente a cada parámetro de entrada dentro de los rangos definidos anteriormente, y calculando el valor de los parámetros de salida (F1, F2 y F3) por grupos de registros y de acuerdo a unas fórmulas que permitan obtener el tipo de valor (alto, medio o bajo) deseado en cada variable. En cada regla, el valor de cada parámetro de salida F_i se calcula en función de los de entrada P_i, de forma que el tipo valores de salida que se obtenga sea

conocido, y se pueda chequear el correcto funcionamiento de la herramienta; por ejemplo, en la regla 1 se puede observar como para obtener valores medios en F1, se suman los valores de dos parámetros de entrada que tomarán valores medios (P1 y P4).

Los registros de la base de datos generada, están estructurados fundamentalmente en base a las tres reglas siguientes:

- **Regla 1:** 40 registros

  Si P1 es **medio**, P2 es **pequeño**, P3 es **grande** y P4 es **medio**
  Entonces F1 es **medio,** F2 es **grande** y F3 es **pequeño**

  donde, **F1**= p1+p4 , **F2**= p3+(p4/p2), **F3**= p2+(1/p3)

- **Regla 2:** 40 registros

  Si P1 es **grande**, P2 es **grande**, P3 es **medio** y P4 es **pequeño**
  Entonces F1 es **pequeño,** F2 es **medio** y F3 es **grande**

  donde, **F1**= p4+(1/p1)+(1/p2), **F2**= p3+(p2/p1), **F3**= p1+p2

- **Regla 3:** 40 registros

  Si P1 es **pequeño**, P2 es **medio**, P3 es **pequeño** y P4 es **grande**
  Entonces F1 es **grande,** F2 es **pequeño** y F3 es **medio**

  donde, **F1**= p4+(1/p1)+(1/p3), **F2**= p1+(1/p4), **F3**= p2+(p4/p1)

- **Valores de ruido:** 30 registros
  Se han introducido una serie de registros con valores aleatorios, escogidos de forma que interfieran en la distribución de valores seleccionada en cada uno de los tipos de reglas.

Los resultados obtenidos con PreFuRGe, al procesar esta base de datos son los mostrados en la figura 4.16. Se puede comprobar que los resultados obtenidos coinciden cualitativamente con la distribución de los datos de la base de datos artificial utilizada, y que son conocidos a priori, tal y como se ha descrito en las reglas anteriores.

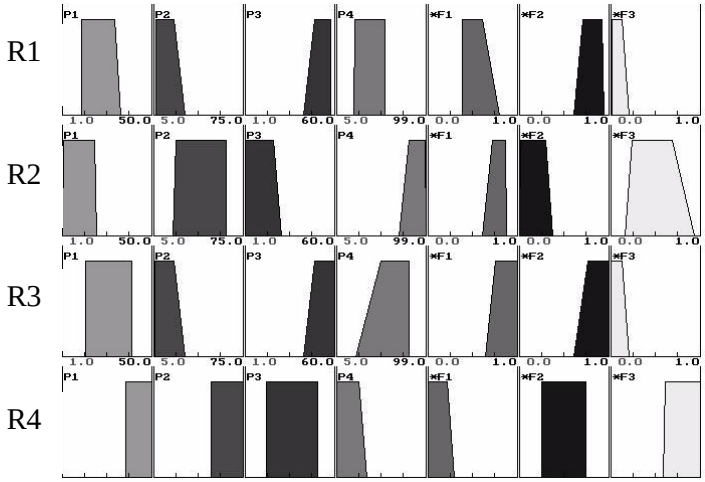


Figura 4.16: Resultados sobre BD artificial multiparamétrica (con funciones)

La interpretación en lenguaje natural que la herramienta realiza de las reglas borrosas en formato gráfico generadas (figura 4.16) es la siguiente:

- **R1:** Si *P1* es **Medio** Y *P*2 es **Bajo** Y *P3* es **Alto** Y *P4* es **Medio**
  Entonces *F1* es **Medio** Y *F2* es **Alto** Y *F3* es **Bajo**

  La información cualitativa proporcionada por esta regla, coincide totalmente con la impuesta en la *regla 1.* Esto quiere decir que los 40 registros generados siguiendo la distribución marcada por esta regla, han sido identificados y procesados de forma correcta.

- **R2:** Si *P1* es **Bajo** Y *P*2 es **Medio** Y *P3* es **Bajo** Y *P4* es **Alto**
  Entonces *F1* es **Alto** Y *F2* es **Bajo** Y *F3* es **Medio**

  La información cualitativa proporcionada por esta regla, coincide con la exigida en la *regla 3,* lo que quiere decir que los registros generados según esta regla, han sido identificados y procesados de correctamente.

- **R3:** Si *P1* es **Medio** Y *P*2 es **Bajo** Y *P3* es **Alto** Y *P4* es **Medio**
  Entonces *F1* es **Alto** Y *F2* es **Alto** Y *F3* es **Bajo**

  La información cualitativa proporcionada por esta regla, no coincide con la marcada por ninguna de las reglas utilizadas para generar la base de datos

artificial. Esto quiere decir que la herramienta ha encontrado un tipo de información adicional que inicialmente no se ha prefijado en los datos generados; en concreto se tiene información de cómo deben ser los parámetros de entrada para que F1 sea alta, F2 sea alta y F3 sea baja.

- **R4:** Si *P1* es **Alto** Y *P*2 es **Alto** Y *P3* es **Medio** Y *P4* es **Bajo**
  Entonces *F1* es **Bajo** Y *F2* es **Medio** Y *F3* es **Alto**

La información cualitativa proporcionada por esta regla, coincide con la establecida en la *regla 2,* lo que significa que los registros generados basándose en esta regla, han sido identificados correctamente.

4.3.4 Experimento 4

Una vez chequeado el correcto funcionamiento de PreFuRGe, se pasó a probar con una de las bases de datos más utilizadas para realizar pruebas de Minería de Datos, como es IRIS [Blake98]. Las características principales de esta base de datos son:

- Número de registros: 150
- Descripción: Base de datos con anchura y longitud de pétalos y sépalos de flores, clasificados en tres clases según estos valores. Existen 50 registros de cada clase.
- Número de parámetros de entrada: 4
  - Sepal Length: Rango (4.0, 8.0)
  - Sepal Width : Rango (2.0, 4.5)
  - Petal Length : Rango (1.0, 7.0)
  - Petal Width : Rango (0.1, 2.5)

- Número de parámetros de salida: 1
  - Clase: Discreta con 3 posibles valores:
    - A: Iris Setosa
    - B: Iris Versicolor
    - C: Iris Virginica

Los resultados obtenidos con PreFuRGe, al procesar esta base de datos son los mostrados en la figura 4.17. Las clases identificadas por la herramienta se corresponden exactamente con las de la base de datos, con la siguiente correspondencia: clase A (Bajo), clase B (Medio) y clase C (Alto).

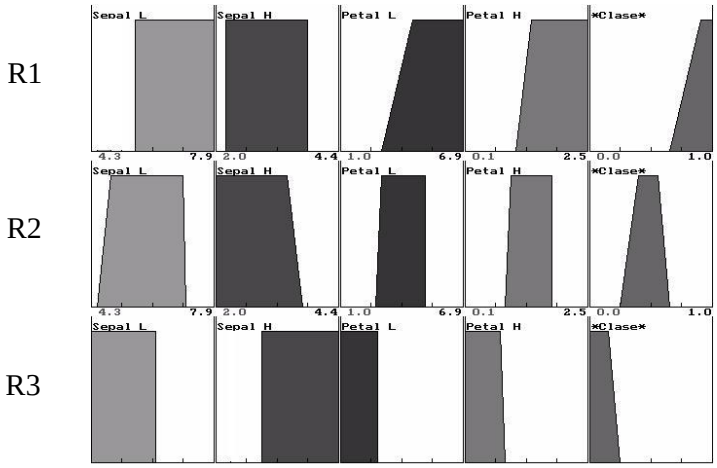


Figura 4.17: Resultados sobre BD IRIS

La interpretación en lenguaje natural de las reglas borrosas generadas, proporcionada por la herramienta es:

Regla 1:
Si *Sepal L* es **Mayor o igual que Media** Y *Sepal W* es **Menor o igual que Media** Y *Petal L* es **Alta** Y *Petal W* es **Alta** Entonces Clase es *Iris Virginica*

Regla 2:
Si *Sepal L* es **Mayor que baja y menor o igual que Media** Y *Sepal W* es **Menor o igual que Media** Y *Petal L* es **Media** Y *Petal W* es **Media**
Entonces Clase es *Iris Versicolor*

Regla 3:
Si *Sepal L* es **Baja** Y *Sepal W* es **Mayor o igual que Media** Y *Petal L* es **Muy baja** Y *Petal W* es **Muy Baja** Entonces Clase es *Iris Setosa*

Se observa la gran calidad de la información cualitativa proporcionada por PreFuRGe, tanto en modo gráfico como en lenguaje natural. Este tipo de resultados pueden ser interpretados de manera fácil y rápida, incluso por personas no expertas en la materia. Estos resultados obtenidos, contrastan en claridad con los obtenidos por COGITO [Aguil01] sobre la misma base de datos, que fueron:

Regla 1:
Si *Sepal L* $\in$ [4.305, -] Y *Petal L* $\in$ [1.2613, 3.3090] Entonces *Clase Iris Setosa*

Regla 2:
Si *Sepal L* ∈ [4.696, -] Y *Sepal W* ∈ [2.0104, -] Y *Petal L* ∈ [1.157, -] Y *Petal W* ∈ [1.731, -] Entonces *Clase Iris Virginica*

Regla 3:
Si *Petal L* ∈ [-, 4.894] Y *Petal W* ∈ [-, 1.602] Entonces *Clase Iris Versicolor*

Regla 4:
Si *Petal L* ∈ [1.276, -] Y *Petal W* ∈ [0.3444, -] Entonces *Clase Iris Virginica*

Uno de los primeros detalles que se observan es que no intervienen todos los atributos de la flor (Sepal L, Sepal W, Petal L y Petal W) en cada una de las reglas. Además, al obtener información semicualitativa de tipo intervalar, es difícil interpretar cualitativamente cómo deben de ser los pétalos o los sépalos para clasificar como *Iris Setosa, Versicolor* o *Virginica* una determinada flor.

4.4 Conclusiones

En este capítulo se ha presentado PreFuRGe, una herramienta de Minería de Datos, basada en un algoritmo de clustering borroso, que proporciona reglas de decisión borrosas, sin necesidad de que la base de datos esté etiquetada, es decir que cada registro de la misma tenga asociada una clase.

Como se ha podido comprobar en los distintos experimentos realizados, la información proporcionada por PreFuRGe no necesita ser interpretada, de forma que la persona que deba manejar los resultados, no necesita tener ningún conocimiento previo sobre Minería de Datos para poder utilizar la información obtenida.

A la vista de los resultados mostrados, podemos concluir que las principales características que posee PreFuRGe, son las siguientes:

- La herramienta puede trabajar con bases de datos cuantitativas formadas por múltiples atributos de entrada y múltiples variables de salida.

- Es posible ponderar la importancia de cada variable de la base de datos antes de obtener las reglas de decisión, de forma que éstas se aproximen todo lo posible a la realidad del entorno.

- Es posible trabajar con bases de datos formadas sólo por variables de salida (Figura 4.4) sin necesidad de que existan atributos de entrada donde proyectar los clusters obtenidos.

- La herramienta detecta la existencia de *outliers* en la base de datos, dejándolos fuera durante el proceso de generación de reglas, ya que podrían producir resultados erróneos desde el punto de vista cualitativo.

- La herramienta detecta y procesa las proyecciones múltiples (montículos) que puedan aparecer (figura 4.6), de forma que las reglas obtenidas sean lo más precisas posibles.

- Con idea de facilitar la interpretación de las reglas de decisión obtenidas, éstas son mostradas de forma cualitativa en dos formatos muy cercanos al modo de razonar humano:

    - Modo gráfico, con reglas del tipo *Si-Entonces*, que pueden tener varios elementos en el consecuente (detrás del entonces).

    - En lenguaje natural, con reglas del tipo *Si-Entonces*, realizando una interpretación lingüística de las reglas gráficas, utilizando operadores borrosos que le dan mucha significación a las reglas.

Capítulo 5

Aplicación de PreFuRGe a proyectos de desarrollo de software

5.1 Introducción

Como se describió en el capítulo 1, uno de los principales objetivos de este proyecto de tesis es aportar nuevas contribuciones en el campo de la toma de decisiones en proyectos de desarrollo de software; en este sentido se ha desarrollado la herramienta, PreFuRGe, descrita en el capítulo 4. Aunque la herramienta de Minería de Datos propuesta es aplicable a cualquier base de datos cuantitativa multiparamétrica, la aplicación a bases de datos de PDS es una de las principales motivaciones de esta investigación, ya que se cuenta con la posibilidad de que un ingeniero de software, experto en proyectos de desarrollo de software, pueda medir la calidad de la información generada por la herramienta, y determine si ésta es aplicable en la toma de decisiones en proyectos de desarrollo.

Con el desarrollo de PDS a gran escala, una de las principales preocupaciones de los responsables ha sido la de optimizar las variables que miden la viabilidad del proceso final; esta tarea, en un principio, era el propio responsable quien, basándose en su experiencia, estimaba los valores en los que se tenían que mover los atributos (política

de gestión) que afectan al proceso de desarrollo. El principal problema radica en la cantidad de atributos que deben ser estimados, ya que una incorrecta estimación de alguno de ellos puede provocar que el resultado final no cumpla con los objetivos marcados.

Uno de los primeros trabajos orientados a facilitar la toma de decisiones en Proyectos de Desarrollo de Software (PDS) estuvo centrado en el desarrollo de un SPS [Ramos97], [Ramos98], que permitiese simular el comportamiento de un proyecto utilizando la mitad de atributos que utiliza el modelo recogido en [Abdel91].

En este punto aparece un problema adicional, común al proceso que conlleva la toma de decisiones en cualquier proyecto: *la diversidad y simultaneidad de políticas de gestión y factores que influyen sobre el proceso de desarrollo de software*. En otras palabras, poder conocer cuál es la combinación adecuada de los atributos del proyecto para obtener unos objetivos concretos.

Es necesario resaltar que la aplicación de técnicas de Minería de Datos al proceso de desarrollo de software cuenta con un problema añadido, que no existe en otros campos, como es la escasez de bases de datos reales con información sobre los atributos que han condicionado el desarrollo de otros PDS. Actualmente, esta deficiencia se resuelve gracias a la existencia de potentes sistemas de simulación (SPS) y a la posibilidad de construir modelos dinámicos para la simulación de proyectos de software.

5.2 Bases de Datos de PDS

Con la implantación de los Modelos Dinámicos para PDS presentados en [Ramos98], en el entorno de simulación Vensim, se han podido generar con relativa facilidad las *bases de datos* cuantitativas sobre las que opera la herramienta presentada. Con la utilización de los SPS, el gestor del proyecto puede aplicar diferentes políticas de gestión, y analizar los resultados obtenidos.

Es importante destacar, que en esta investigación se va a trabajar con proyectos ya finalizados (análisis post-mortem), de los que se conocen por tanto los valores que se estimaron para cada atributo así como los valores finales de las distintas variables. Los atributos que se analizarán en este estudio, serán los que aparecen en las Tablas 5.1 y 5.2. En ellas se indica, para cada atributo, el valor que se estimó inicialmente en el proyecto, los intervalos de valores cuantitativos en los que puede moverse y una valoración cualitativa de los mismos.

5.2.1 Generación de las bases de datos de PDS

Los modelos dinámicos para PDS incluyen una serie de parámetros y tablas que permiten definir las políticas de gestión que pueden ser aplicadas. A partir de la información proporcionada (estimaciones iniciales y valores finales) el simulador de proyectos con el que se esté trabajando (SPS) permite crear diferentes escenarios (estrategias de gestión[1]) a partir de un proyecto ya finalizado [Ramos99]. Para cada escenario es necesario definir:

- Los intervalos de valores que pueden tomar los diferentes atributos del proyecto. Esto dependerá tanto del proyecto como de la empresa de desarrollo.
- Indicar para qué valores finales de tiempo, coste y calidad se consideran cumplidos los objetivos del proyecto.

Para la obtención de cada escenario se ha considerado que la mayoría de los atributos del proyecto tomarán los valores estimados para el mismo. En cambio los atributos que se desean analizar, para ver qué incidencia puede tener sobre los resultados finales del proyecto, variarán dentro de un rango de valores que se definirá previamente. Naturalmente, estos valores estarán siempre entre los permitidos para el tipo de empresa y proyecto que se considere.

Cada uno de los posibles escenarios del proyecto se guarda en un registro de una base de datos. Cada uno de los escenarios indica la evolución o trayectoria que habría tenido (o tendrá, para análisis a priori) el proyecto, si los atributos o características del mismo fuesen las que aparecen en ese registro. Repitiendo este proceso el número de veces que se desee, se obtiene la base de datos completa con un número de registros igual al número de escenarios que se hayan simulado. Para esta investigación, cada una de las bases de datos generadas tiene un total de 300 registros.

Hay que señalar que el modelo dinámico utilizado para simular los Proyectos de Desarrollo de Software posee más parámetros de los descritos anteriormente, pero sólo se han destacado aquellos en los que tenemos especial interés para ver el efecto que producen, en función de los valores que tomen, dentro de los intervalos de valores posibles. Además son parámetros que el gestor de proyecto puede modificar en función de la política de gestión de personal o de tiempo de entrega que decida aplicar en cada fase del proyecto.

El resto de parámetros del modelo, tomarán valores que vendrán determinados por el tipo de proyecto que se esté simulando, y que por tanto estarán fijados a priori.

[1] Conjunto de decisiones particulares adoptadas en un proyecto

En la figura 5.1 se muestra un ejemplo de base de datos cuantitativa, obtenida tras la simulación de un PDS, y que será utilizada como entrada por la herramienta propuesta. Las bases de datos generadas por los simuladores pueden ser tan grandes como se desee tanto en número de registros (número de escenarios simulados) como en número de atributos y variables de salida:

Asimdy	Hiredy	Mxscdx	Trnsdy	**	Jbszmd	Schcdt	Anerpt
8.1760	7.2630	37.116	8.9457		1861.85	349.5	0.4764
5.7184	9.9905	27.688	7.6122		1750.81	349.5	0.5089
9.0010	8.4211	18.094	8.596		1901.66	349.5	0.4948
12.642	5.6209	8.4597	9.710		2160.08	350.5	0.4037
5.8365	8.7244	12.320	8.9029		1756.44	349.5	0.5444

Figura 5.1: Ejemplo de base de datos cuantitativa generada

En este ejemplo se muestran una serie de valores x_{ij} que tomará la herramienta PreFuRGe para su posterior análisis. Se trata de una base de datos con cuatro atributos de entrada Asimdy, Hiredy, Mxscdx, Trnsdy (tabla 5.1) y tres variables de salida Jbszmd, Schcdt, Anerpt (tabla 5.2).

5.2.2 Descripción de las bases de datos

En este apartado se describen los atributos y variables utilizadas de las bases de datos de PDS obtenidas tras la simulación del modelo dinámico correspondiente.

Para comprobar los resultados que puede generar la herramienta PreFuRGe, se han utilizado tres bases de datos cuantitativas, obtenidas tras la simulación del proyecto descrito a continuación (tabla 5.2), usando diferentes estrategias para analizar cuál es la más adecuada para el desarrollo del proyecto. Estas estrategias consideran las siguientes políticas de *gestión de personal* y *tiempo de entrega* sobre el proyecto:

1- **Sin restricciones en la gestión de personal ni en el tiempo de entrega**: Los atributos se pueden mover dentro de todo el rango posible de valores

2- **Gestión de personal rápida con restricción en el tiempo de entrega**: Los atributos relacionados con la gestión de personal toman valores dentro del intervalo considerado como rápido, y el atributo relacionado con la gestión del tiempo de entrega toma valores dentro de los intervalos de plazo fijo y

plazo moderado, es decir, el aplazamiento en el tiempo de entrega no puede superar al tiempo estimado en más de un 20%

3- **Gestión de personal rápida sin restricción en el tiempo de entrega**: Los atributos relacionados con la gestión de personal toman valores dentro del intervalo considerado como rápido, y el atributo relacionado con la gestión del tiempo de entrega se puede mover dentro de todo el rango posible de valores.

La descripción de los atributos que vamos a analizar relacionados con políticas de gestión de personal y tiempo de entrega es la mostrada en la tabla 5.1.

La división en subintervalos de cada atributo, así como su interpretación cualitativa es realizada por el experto basándose principalmente en su experiencia; esta información es fundamental para poder etiquetar los registros de las bases de datos. Cabe recordar que las herramientas de Minería de Datos basadas en aprendizaje supervisado (i.e.: C4.5), necesitan que las bases de datos estén etiquetadas.

ATRIBUTO	***DESCRIPCIÓN (Unidad)***	***VALOR INICIAL***	***INTERVALO MÁXIMO***	***SUBINTERVALOS Y VALORACIÓN CUALITATIVA***
Asimdy	Retraso medio de adecuación de los técnicos nuevos que se ha incorporado al proyecto (días)	20	[5, 120]	[5, 15] Adecuación rápida [16,40] Adecuación moderada [41, 120] Adecuación lenta
Hiredy	Retraso medio de incorporación de los técnicos nuevos en el proyecto (días)	30	[5, 40]	[5, 10] Integración rápida [11,20] Integración moderada [21, 40] Integración lenta
Mxscdx	Aplazamiento máximo permitido en el tiempo de entrega respecto del estimado (%)	3	[1, 50]	[1, 1.10] Plazo fijo [1.11, 1.20] Plazo moderado [1.21, 50] Sin plazo estricto
Trnsdy	Retraso medio en la salida de los técnicos del proyecto (días)	10	[5, 15]	[5, 10] Salida rápida [11, 15] Salida lenta

Tabla 5.1: Atributos relacionados con gestión de personal y tiempo de entrega

En la siguiente tabla (tabla 5.2) se recogen, para el proyecto ya finalizado, los valores estimados y reales para cada una de las variables de interés:

VARIABLE	*DESCRIPCIÓN (UNIDADES) [Concepto que representa]*	*VALOR ESTIMADO*	*VALOR REAL*
Jbszmd	Esfuerzo necesario para realizar el proyecto (técnicos-días)[Coste]	1111	2092
Schcdt	Tiempo de desarrollo (días) [Tiempo de entrega]	320	387
Anerpt	Calidad del producto final (errores / tareas) [Calidad]	0	0.26

Tabla 5.2: Variables de un PDS

Para poder analizar correctamente las reglas de gestión que se obtengan al aplicar la herramienta PreFuRGe a las bases de datos cuantitativas que se obtengan, es importante conocer una serie de resultados obtenidos en otras investigaciones. En [Ramos 99], se recoge una discusión sobre el impacto que tienen las políticas de gestión de personal y de tiempo de entrega sobre un proyecto. De esta discusión es importante indicar para entender algunos de los resultados que obtendremos a continuación, lo siguiente:

- Las políticas de gestión de personal son dominantes frente a las de tiempo de entrega.

- Cuando se aplican políticas de gestión de personal extremas (rápidas o lentas) es difícil encontrar buenos resultados para el tiempo, esfuerzo y calidad, simultáneamente.

- Una política de gestión de personal rápida favorece la obtención de buenos resultados en el tiempo a costa de aumentar el esfuerzo del proyecto. Con las políticas de gestión de personal lentas, ocurre lo contrario.

Los intervalos de valores que se pueden considerar como buenos en cada una de las variables son definidos por el responsable del proyecto. A continuación se describen (para las variables de la tabla 5.2) estos intervalos, utilizando la siguiente notación:

VARIABLE→[intervalo bueno, intervalo malo].

- **Jbszmd →** [1111-1444, 1444-en adelante]
- **Schcdt →** [320-384, 384-en adelante]
- **Anerpt →** [0-0.25, 0.25-adelante]

Esta información es de vital importancia si se desean utilizar técnicas de aprendizaje supervisado, para poder etiquetar los registros, como se describió en el capítulo 3 al hablar de estas técnicas.

5.3 Aplicación de técnica de Minería de Datos

La aplicación de técnicas de Minería de Datos en el área de la Ingeniería del Software no está muy difundida y es una actividad muy reciente, que ofrece grandes posibilidades a los responsables de PDS; una de las principales contribuciones es la posibilidad de mejorar la gestión del proceso de desarrollo de proyectos de software.

La aplicación de estas técnicas en el proceso de obtención de reglas de gestión o de decisión[2], permiten solventar muchos de los problemas que aparecían en la simulación de proyectos, analizada en el capítulo 2, al hablar de los Simuladores de Proyectos de Software. Recordando lo detallado en este capítulo, los principales problemas que aparecen en la simulación de proyectos son:

- El Director del proyecto debe definir de forma concreta los valores para los atributos del proyecto. Esta tarea es compleja y suele estar basada en la experiencia ya que, seguramente el director no sepa de manera concreta si la dedicación media de los técnicos al proyecto será del 25% o del 45%; lo que puede llegar a conocer es que dicha dedicación se moverá entre un 20% y un 60%.
- Simulación del proyecto tantas veces como sea necesario, hasta alcanzar los objetivos deseados en el mismo. En cada nueva simulación se reajustan los valores asignados a los atributos, con el fin de corregir las desviaciones producidas en los objetivos. Este reajuste del valor de los atributos lo realiza el director del proyecto de forma manual, y basándose en su experiencia.

En este capítulo, analizaremos la forma de obtener automáticamente reglas de gestión para un PDS. La obtención de reglas de gestión permite al responsable del proyecto evaluar cuáles son las políticas de gestión más significativas para conseguir los objetivos del proyecto, así como saber si estas políticas son aplicables o no.

Para la obtención de las reglas de gestión se van a utilizar las técnicas de Aprendizaje Supervisado y no Supervisado, descritas en el capitulo anterior:

[2] Se entiende por regla de gestión o de decisión un conjunto de políticas de gestión (decisiones) que estima el director para cumplir con los objetivos del proyecto.

b) Aprendizaje Supervisado:
   - Árboles de decisión (C4.5)
   - Reglas jerárquicas (COGITO)

c) Aprendizaje No Supervisado
   - Reglas de gestión borrosas (PreFuRGe)

La aplicación de las tres herramientas de Minería de Datos sobre cada una de las bases de datos de PDS que se generen, permitirá comparar los resultados obtenidos, y medir la calidad de la información generada por cada una de ellas.

En los siguientes apartados se describe el proceso de generación de reglas de gestión mediante estas técnicas, y su aplicación a la toma de decisiones en PDS.

5.3.1 Aprendizaje Supervisado: C4.5 y COGITO

La aplicación de técnicas de Aprendizaje Automático, concretamente las herramientas C4.5 y COGITO [Aguil01], permiten resolver muchas de las dificultades comentadas anteriormente que tiene el uso de un SPS [Ramos98b] [Ramos99b]. Estas técnicas permiten obtener reglas de gestión o de decisión para facilitar la toma de decisiones y cumplir con los objetivos del Proyecto de Desarrollo de Software. Para la obtención de reglas de gestión, se conjugan las ventajas que presentan los sistemas que aprenden en base a reglas y la información que proporciona un modelo dinámico para PDS.

El conocimiento de estas reglas de gestión sirve de ayuda en la toma de decisiones para estimar automáticamente los resultados deseados por el responsable del proyecto (coste, tiempo de entrega, calidad, productividad, etc.). Además, la obtención de reglas de gestión permite al director del proyecto analizar cuáles son las políticas de gestión más significativas para conseguir los objetivos del Proyecto Software.

En la Figura 5.2 se recoge el proceso seguido para la simulación de un Proyecto Software y la obtención de reglas de gestión, que permiten estimar dichos resultados u objetivos.

Los pasos necesarios en este caso son los siguientes:

1. Definición de:
   a. Los intervalos de valores que pueden tomar los atributos del proyecto.

b. Los objetivos del proyecto que se desean obtener. Por ejemplo, "Que el coste final del proyecto no supere al inicial en más de un 20% y que el tiempo de entrega final no supere al inicial en más de un 10%".

2. Simular el proyecto mediante el SPS. Al simular el proyecto, los atributos toman valores elegidos aleatoriamente dentro del intervalo de valores definido anteriormente.
3. Generación automática de una base de datos. En cada simulación se obtiene un registro de la base de datos en el que se guardan los valores obtenidos en la simulación para los atributos, además de los valores finales de las variables (obtenidos también en la simulación) que nos interese analizar (coste, tiempo, calidad, etc.). Cada registro de la base de datos representa un escenario posible del proyecto.
   Por tanto, los valores de la base de datos son de tipo numérico, y están organizados en *n* atributos x_i, y *m* variables y_i, dependientes de los atributos.
4. Aplicación de técnicas de Aprendizaje Automático. A partir de la base generada, las herramientas referenciadas, aprenden por diferentes técnicas, examinando la muestra de casos resueltos y proponiendo un conjunto de reglas para la toma de decisiones.
5. Obtención de reglas de gestión para la ayuda en la toma de decisiones en el proceso de desarrollo del proyecto software.

En este caso, el responsable del proyecto sólo debe realizar el primer paso, donde además se facilita la definición de los atributos respecto del proceso descrito en el capítulo 2, y donde el director del proyecto debería definir valores concretos para los parámetros, lo cual ya se vió que normalmente debía hacerse improvisando; el resto de los pasos descritos se realizan automáticamente.

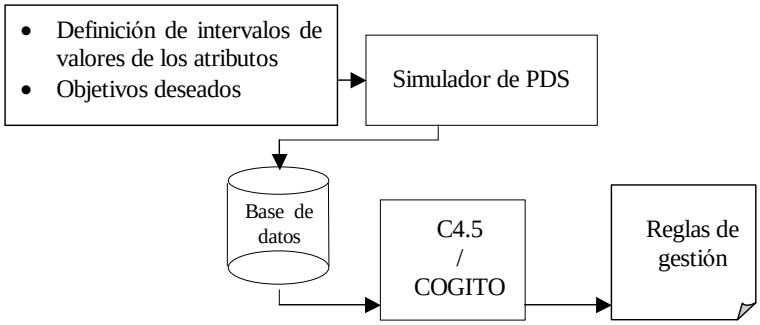


Figura 5.2: Pasos en la simulación de un PDS utilizando un SPS y técnicas de Aprendizaje Automático (C4.5/COGITO).

Al trabajar con estas dos herramientas, nos movemos en el contexto del *Aprendizaje Supervisado*, por tanto es necesario que la base de datos (BD) que vayamos a utilizar para obtener las reglas de gestión, esté etiquetada, es decir, que cada registro tenga definido una etiqueta que determine su clase dentro de la BD. Este proceso debe hacerlo el experto de forma manual utilizando criterios basados en la experiencia, y que hacen que la obtención de un determinado tipo de reglas dependa en gran medida de este proceso de etiquetado.

Las reglas de gestión obtenidas tras la aplicación de técnicas de Aprendizaje Automático, sobre la base de datos obtenida tras la simulación del PDS, pueden ser difíciles de interpretar incluso por parte del propio experto. Uno de los principales inconvenientes es que utilizan intervalos de valores para definir el rango de valores en los que se mueve cada parámetro. Esto hace que el experto deba interpretar a posteriori, si esos rangos corresponden a valores altos, medios, muy bajos, etc., de tales parámetros.

Un ejemplo de este tipo de reglas generadas por las herramientas anteriores es:

$$1 < \mathbf{S1} < 15 \ \underline{\text{si}} \ \ P1<5,\ P3>2,\ 1<P5<7 \qquad \text{(R1)}$$

esta regla (R1) sería interpretada cualitativamente por un experto de la siguiente forma:

> "La variable **S1** toma valores entre 1 y 15 (considerados como buenos) si el parámetro *P1* es menor que 5 (que podría ser alto para P1), el *P3* es mayor que 2 (que podría ser medio alto para P3) y el *P5* está entre 1 y 7 (que podría ser medio para P5)".

Como se puede apreciar este tipo de reglas proporcionan al gestor una información sobre el proyecto de un valor cualitativo mayor que la base de datos cuantitativa obtenida mediante simulación. Normalmente estas reglas involucran a un gran número de atributos y suelen tener varios niveles de anidamiento lo que las hacen aún más complejas de utilizar. Además el carácter cualitativo de términos como alto o bajo, lo debe proporcionar el propio experto tras la interpretación de la regla.

En definitiva, la utilización de herramientas como las analizadas, para la obtención de reglas de gestión, permite a los responsables de PDS afrontar diferentes situaciones cuando tienen que definir las políticas de gestión más adecuadas para optimizar los valores finales de determinadas variables por separado o bien conjuntamente. Una vez obtenidas las reglas de gestión será el responsable del proyecto el que decida qué regla o

reglas son las más fáciles de aplicar en función del proyecto concreto y de la organización software en la que esté trabajando.

5.3.2 Aprendizaje no Supervisado: PreFuRGe

La utilización de técnicas Minería de Datos junto con la lógica borrosa (algoritmo de *Fuzzy Clustering FCM+* usado en PreFuRGe) permiten añadir, de entrada, dos nuevas mejoras en el uso de los SPS:

- La obtención de reglas de tipo cualitativo, lo que facilita la interpretación de las mismas por el experto.
- El responsable no tiene que definir previamente los resultados u objetivos deseados en el proyecto.

En la figura 5.3, se muestra los pasos a seguir para obtener este nuevo tipo de reglas de gestión. En el primer paso se observa cómo el responsable del proyecto no tiene que *definir los objetivos* del proyecto ya que el propio algoritmo los calcula de forma automática. En este caso, la base de datos numérica generada por el SPS es procesada por un algoritmo de fuzzy clustering que proporciona una información de tipo cualitativo. Esta información cualitativa que se obtiene a partir de los datos cuantitativos de los atributos del proyecto, es un conjunto de reglas de gestión borrosas que permiten al responsable interpretar los datos del proyecto de una manera rápida y bastante clara.

Al trabajar con conjuntos borrosos, en vez de con valores concretos o intervalos en los parámetros, el responsable no tiene por que tener un conocimiento detallado sobre los valores o intervalos de valores de los parámetros del Proyecto Software, como ocurre con la aplicación directa de los SPS, o como ocurre, en menor medida, con la aplicación de las herramientas analizadas en la sección 5.2.1.

Un ejemplo de reglas que se obtienen con este tipo de técnicas puede ser:

Si *P1* es bajo **y** *P2* es mayor que medio **Entonces** *S1* es muy bajo (R2)

Este tipo de reglas cualitativas (R2) se obtienen directamente, por lo que el experto no tiene que realizar ningún tipo de interpretación ni de intervención a posteriori, como ocurría con las reglas proporcionadas por C4.5 o COGITO, que debían ser analizadas e interpretadas por el experto una vez generadas para poder llegar a tener la información cualitativa (cercana a la forma de razonar humana) deseada.

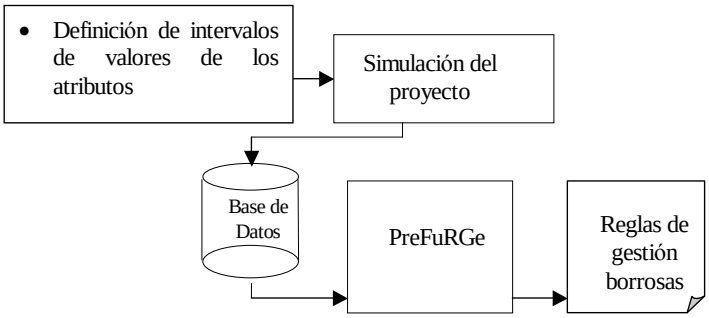


Figura 5.3: Pasos en la simulación de un PDS utilizando un SPS y técnicas de lógica borrosa (PreFuRGe).

5.4 Reglas de gestión obtenidas

El objetivo que se persigue es que el gestor disponga de unas reglas de gestión fáciles de aplicar y que le permitan obtener buenos resultados. En el proyecto, ya finalizado, con el que se va a trabajar, se conocen tanto los datos iniciales como los finales; el objetivo será obtener reglas de gestión, que habrían permitido estimar mejores resultados (simultáneamente para el tiempo, esfuerzo y calidad) a los que finalmente se obtuvieron; es decir, qué atributos se tendrían que haber modificado para obtener mejores resultados.

La estimación de los recursos necesarios para realizar un proyecto y la estimación de los atributos o características del proyecto y del proceso a seguir, dependerán en gran medida de la prioridad que tengan los diferentes objetivos.

Si consideramos que la calidad debe ser siempre prioritaria, es decir, el producto final debe cumplir siempre con los criterios de calidad, tenemos tres alternativas:

- Que el objetivo sea el tiempo de desarrollo, independientemente del esfuerzo necesario para realizar el proyecto.
- Que el objetivo sea el esfuerzo o coste del proyecto, independientemente del tiempo invertido.
- Que el objetivo sea, simultáneamente, el tiempo y el esfuerzo.

Los criterios que se seguirán para realizar un análisis comparativo de las reglas obtenidas con la herramienta PreFuRGe y las proporcionadas por las herramientas C4.5 y COGITO, son:

- Elegir las reglas que recojan el mayor número de escenarios y aciertos.
- Elegir las reglas que tengan el menor número de atributos.
- Si se realiza un análisis post-mortem, se considera que las mejores reglas son las que implican la modificación de un número menor de atributos. Es decir, de entre los atributos que aparecen en la regla cuáles son los que se mantienen en los valores iniciales y cuáles tendrían que haberse modificado.
- Si se realiza un análisis a priori, se deben elegir las reglas que contengan atributos que sean fáciles de controlar a lo largo del proceso de desarrollo.
- Y finalmente, elegir la o las reglas con las que se estimen los mejores resultados.

Para la obtención de las distintas reglas de gestión con PreFuRGe se han generado tres bases de datos (BD):

- BD *CrSrTi*: Base de datos en la que los atributos relacionados con la gestión de personal toman valores dentro del intervalo considerado como rápido, es decir, la gestión de personal se realiza con rapidez; y el atributo relacionado con el tiempo de entrega se puede mover dentro de todo el rango posible de valores.

- BD *CrCrTi*: Base de datos en la que los atributos relacionados con la gestión de personal toma valores dentro del intervalo considerado como rápido, es decir, la gestión de personal se realiza con rapidez; y el atributo relacionado con el tiempo toma valores dentro de los intervalos de plazo fijo y plazo de entrega moderado, es decir, el aplazamiento permitido en el tiempo de entrega no puede superar al tiempo estimado en más de un 20%.

- BD *SrCoTi*: Base de datos en la que tanto los atributos relacionados con la gestión de personal como el atributo relacionado con el tiempo de entrega, se pueden mover dentro de todo el rango posible de valores.

Al aplicar la herramienta PreFuRGe sobre las distintas Bases de Datos que se generan con las simulaciones, se obtiene un conjunto de reglas de gestión borrosas en formato gráfico. A partir de estas reglas gráficas, PreFuRGe genera la interpretación de las mismas en lenguaje natural. Para la discusión de los resultados obtenidos, vamos a considerar las reglas borrosas en formato gráfico, y a continuación se realizará un exhaustivo análisis de las mismas describiendo todo lo que de ellas se pueda deducir.

Cabe destacar que al considerar las tres variables: coste (Jbszmd), tiempo (Schcdt) y calidad (Anerpt) simultáneamente, sólo tendremos información sobre aquellas combinaciones de éstas de las que se tengan datos en la BD, es decir, por ejemplo puede no haber ninguna regla que informe de cómo deben ser los atributos del proyecto para

obtener un coste bajo, tiempo bajo y calidad buena a la vez, aunque este dato fuese de interés para el gestor del proyecto. Este hecho puede deberse a que en la simulación del proyecto, se ha aplicado una determinada política para la que, ese tipo de valores en las variables del proyecto (coste, tiempo y calidad) no se pueden dar.

En cada una de las tres bases de datos sobre las que se va a trabajar, para poder aplicar las herramientas C4.5 y COGITO, es necesario realizar un etiquetado de las mismas, ya que como se explicó en el capitulo 3, como utilizan *aprendizaje supervisado*, es necesario que cada registro tenga asignada una clase. Para realizar el etiquetado de las bases de datos es necesario que el experto determine los intervalos en los que se puede mover cada variable del proyecto (coste, tiempo o calidad) para ser considerados como valores "buenos" (por ejemplo coste bajo, tiempo alto, etc.). Este detalle del etiquetado, determinante en el proceso de generación de reglas en aprendizaje supervisado, depende en gran medida de la experiencia del experto, por lo que toda la automatización posterior del proceso se ve afectada por una decisión humana y no determinista.

La herramienta PreFuRGe sin embargo, como se detalló en el capítulo 4, no necesita el etiquetado previo de los registros de las bases de datos (*aprendizaje no supervisado*), por lo que el proceso de generación de reglas es totalmente automático desde que se le proporciona a la herramienta la base de datos a tratar.

En los siguientes apartados se describen los resultados obtenidos con la aplicación de la herramienta PreFuRGe sobre cada una de las bases de datos de PDS generadas, así como los obtenidos sobre las mismas bases de datos con COGITO y C4.5, para posteriormente realizar un análisis comparativo sobre las reglas obtenidas.

5.4.1 Proyecto con política de gestión de personal rápida sin restricción en el tiempo de entrega: CrSrTi

La estrategia seguida en el proyecto está relacionada con las políticas de gestión de personal (contratación rápida) y aplicando una política de plazo de entrega, sin restricción en el tiempo de entrega; esto queda reflejado en el comportamiento del parámetro *Mxscdx,* que representa el porcentaje de desviación máximo permitido en el tiempo de entrega respecto del inicial estimado que como se observa, se mueve prácticamente en todo el intervalo posible de valores [1,50], es decir, se permite cualquier desviación respecto del valor inicial estimado.

5.4.1.1 Resultados de PreFuRGe

Las reglas de gestión borrosas obtenidas al aplicar la herramienta PreFuRGe a esta base de datos y considerando de manera simultánea las tres variables del proyecto Coste (Jbszmd), Tiempo (Schcdt) y Calidad (Anerpt), son las mostradas en la figura 5.4.

En efecto se puede observar cómo en todas las reglas de gestión borrosas obtenidas con PreFuRGe, el tiempo de entrega (Schcdt) siempre toma valores bajos o muy bajos, tal y como era de esperar, dado el tipo de política aplicada respecto al plazo de entrega.

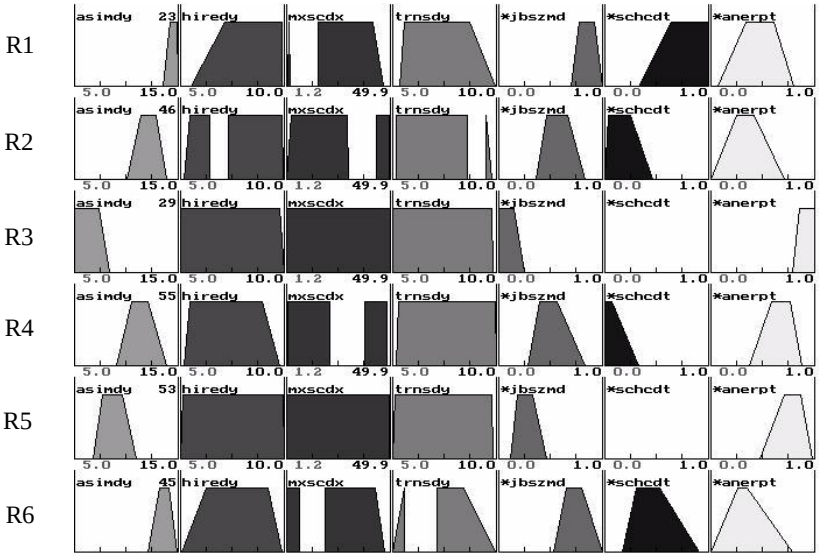


Figura 5.4: Reglas obtenidas con BD: CrSrTi

Por otro lado, se observa cómo el parámetro referente al tiempo medio de contratación (Hiredy) se mueve en el intervalo entre 1 y 10 días, es decir, se corrobora que se está aplicando una política de contratación de personal rápida.

Un análisis de las reglas de gestión borrosas generadas por la herramienta PreFuRGe, lleva a obtener las siguientes conclusiones:

1. Se puede observar que al tratarse de un proyecto con una política de plazo de entrega sin restricción inicial, el parámetro asociado al tiempo de entrega (*Schcdt*) se mueve en todo el rango del intervalo.

2. Las reglas que se descartarían en este caso son R3, R4 y R5, debido a dos razones fundamentalmente: la calidad (Anerpt) en estas reglas es mala, y la mayoría de los atributos se pueden mover en todo su rango de valores, lo que las hace unas reglas poco significativas para la toma de decisiones.
3. Los mejores resultados, simultáneamente, en tiempo, coste y calidad, se recogen en las reglas 2 y 6 (R2, R6). La regla 2 es la que expresa los mejores resultados de las variables; en ésta se aprecia claramente cómo se obtiene un coste (Jbszmd) medio, un tiempo de entrega (Schcdt) bajo y una calidad (Anerpt) buena, en 46 de los 300 registros de la base de datos. La interpretación cualitativa de esta regla, que genera PreFuRGe de manera automática, es la siguiente:

   Si "el tiempo medio de adaptación es *medio* **Y** el tiempo medio de contratación de técnicos nuevos es *bajo* **O** *mayor o igual que medio* **Y** el tiempo de entrega es *menor o igual que medio* **O** *muy alto* **Y** el tiempo medio de salida de los técnicos es *menor o igual que medio* **O** *muy alto*"
   Entonces
   "el *coste* es *medio*, el tiempo de entrega es *bajo* **Y** la calidad es *buena*"

   - En la regla 6 (R6) se puede ver cómo al reducir sensiblemente el número de errores (Anerpt), es decir, mejora algo la calidad, sube el coste a más de medio y el tiempo de entrega a medio, como era de esperar.
   - En este conjunto de reglas de gestión podemos ver que para obtener buenos valores simultáneos en las distintas variables, es importante que la adecuación de los técnicos (Asimdy) sea rápida, es decir, que tome valores altos, y que además se mueva en un rango de valores muy estrecho. Como veremos a continuación, esto coincide con los resultados obtenidos con COGITO y C4.5.

4. En este conjunto de reglas de gestión borrosas obtenido, se puede comprobar que se cumplen los comportamientos esperados, dados los tipos de políticas que se están aplicando. Además, se tiene información cualitativa sobre el comportamiento de los atributos del proyecto para llegar a obtener los diversos estados en las variables del PDS. Cabe recordar que al considerar las tres variables: coste, tiempo y calidad, simultáneamente, sólo tendremos información sobre aquellas combinaciones de éstas de las que se tengan datos en la BD.

5.4.1.2 Resultados de C4.5 y COGITO

En la Tabla 5.3 se indican los valores máximos y mínimos de cada variable, así como los valores considerados como buenos para las variables analizadas. Considerando los intervalos de corte mostrados en la tabla 5.3, en esta base de datos (CrSrTi) se pueden encontrar 11 registros en los que el esfuerzo es bueno y 106 registros en los que el tiempo es bueno. De los 11 registros en los que el esfuerzo es bueno, además, el tiempo es muy bueno ya que es menor de 352 días.

Nombre	Rango de Valores Obtenido	Rango de Valores Buenos (Intervalos de corte)
Jbszmd	[1693, 2415]	[1777, 1999]
Schcdt	[349.5, 361.5]	[352, 359]
Anerpt	[0.236, 0.567]	[0.35, 0.45]

Tabla 5.3: Valores de las variables en la BD CrSrTi

Las reglas obtenidas considerando las tres variables simultáneamente son:

- COGITO: Se han obtenido 2 reglas para 11 registros que lo cumplen, con 3 errores. Estas reglas son:

  El esfuerzo, el tiempo y la calidad serán buenos simultáneamente si:

  R1: 8.46 ≤ Adecuación ≤ 11.5 y Contratación ≥ 8.7 y
  Restricción en tiempo ≥ 9.31
  R2: 9.85 ≤ Adecuación ≤ 11.23 y 6.8 ≤ Contratación ≤ 8 y
  Restricción en tiempo ≤ 39.5

- C4.5: Se han obtenido 3 reglas para 9 registros de los 11 que lo cumplen. Estas reglas son:

  El esfuerzo, el tiempo y la calidad serán buenos simultáneamente si:

  R1: 10.22 ≤ Adecuación ≤ 10.81 y Contratación ≥ 6.89 y
  Restricción en tiempo ≥ 17.44
  R2: 10.22 ≤ Adecuación ≤ 10.81 y Contratación ≥ 6.89 y
  Restricción en tiempo ≤ 12.28
  R3: 9.17 ≤ Adecuación ≤ 9.27

Estas reglas, tanto para COGITO como para el C4.5, nos indican que para obtener buenos resultados simultáneamente en el tiempo y esfuerzo es importante que la adecuación de los técnicos nuevos que se incorporan al proyecto sea rápida y que además se mueva en un rango de valores muy estrecho. La regla (R2) de COGITO tiene un carácter diferenciador frente al resto de las reglas y es que, concreta más el valor que debe tomar el retraso medio en la contratación de los técnicos.

5.4.1.3 Conclusiones

Se comprueba que la información obtenida con las dos herramientas usadas para comparar los resultados, coincide cualitativamente con la obtenida con PreFuRGe en las reglas 2 y 6 (R2, R6) de la figura 5.4.

La regla R2, al tratarse de una regla con múltiples proyecciones en el espacio de entrada (Hiredy, Mxscdx y Trnsdy), tiene una interpretación cualitativa que incluye operadores de conjunción (Y) y disyunción (O); puede comprobarse que esta regla proporciona la misma información cualitativa que dan las dos reglas generadas por COGITO, simultáneamente.

5.4.2 Proyecto con política de gestión de personal rápida, con restricción en el tiempo de entrega: CrCrTi

La estrategia seguida para generar esta base de datos es la de aplicar restricciones en las políticas de gestión de personal (contratación rápida), imponiendo una política de plazo de entrega, con restricción iniciales en el tiempo de entrega, ya que como se observa, el parámetro *Mxscdx* que representa el porcentaje máximo permitido en el tiempo de entrega respecto del inicial estimado, varía entre 1 y 1.20, es decir, se permite sólo un 20% de desviación respecto del tiempo inicial estimado.

5.4.2.1 Resultados de PreFuRGe

Las reglas de gestión borrosas obtenidas al aplicar la herramienta PreFuRGe a esta base de datos, con las políticas de gestión de personal y de plazos de entrega descritas anteriormente, y considerando de manera simultánea las tres variables del proyecto Coste (Jbszmd), Tiempo (Schcdt) y Calidad (Anerpt), son las mostradas en la figura 5.5.

En efecto, se puede observar cómo en todas las reglas de gestión borrosas obtenidas con PreFuRGe, la variable *tiempo de entrega* (Schcdt) siempre toma valores bajos o muy bajos, tal y como era de esperar dado el tipo de política aplicada respecto al plazo de entrega, con fuertes restricciones iniciales.

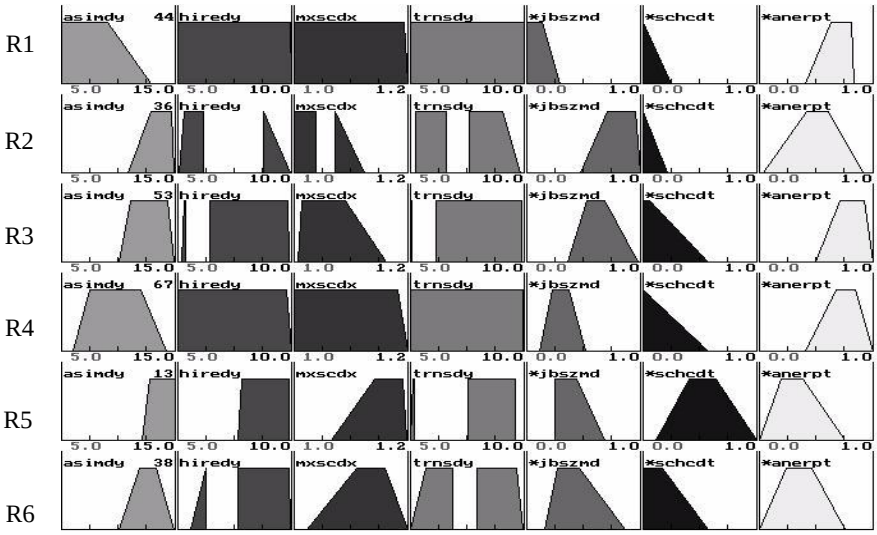


Figura 5.5: Reglas Borrosas BD: CrCrTi

Un análisis de las reglas de gestión borrosas generadas por la herramienta PreFuRGe al aplicarla a la BD de PDS, comentado anteriormente, nos lleva a obtener las siguientes conclusiones:

1. Se puede observar en todas las reglas que, al tratarse de un proyecto con una política de plazo de entrega con fuerte restricción inicial, el atributo asociado al tiempo de entrega (Schcdt) se mueve siempre entre valores bajos del intervalo.

2. Las reglas que se descartarían en este caso son la primera, tercera y cuarta, debido a dos razones fundamentalmente: la calidad (Anerpt) en estas reglas es mala, y la mayoría de los atributos se pueden mover en todo su rango de valores, lo que las hace unas reglas poco significativas para la toma de decisiones.

3. Este tiempo de entrega bajo se obtiene a costa de aumentar el coste y / o de bajar la calidad del proyecto. Este detalle puede apreciarse claramente en diversas de las reglas obtenidas:

   - En la regla 2, se aprecia claramente como se obtiene un tiempo de entrega muy bajo, pero con una coste (Jbszmd) alto y una calidad (Anerpt) muy baja. La interpretación cualitativa de esta regla, que también genera PreFuRGe de manera automática, es la siguiente:

Si "el tiempo medio de adaptación es *alto* **Y** el tiempo medio de contratación de nuevos técnicos es *bajo* **O** muy alto **Y** el tiempo de entrega es *bajo* **O** *medio* **Y** el tiempo medio de salida de los técnicos es *bajo o medio*"

Entonces

"el coste es *alto* **Y** el tiempo de entrega es *muy bajo* **Y** la calidad es media"

- En la regla 4, se puede ver como al relajar el tiempo de entrega, permitiendo valores de tipo medio, y aumentar el esfuerzo (coste), hasta valores medios, se mejora la calidad, tomando ésta valores aceptables.

4. Como era de esperar, al aplicar una política de contratación rápida, y tener restricción en el tiempo, no se obtienen reglas en las que las tres variables tomen valores "buenos" simultáneamente, ya que si baja el tiempo y el coste del proyecto, también baja la calidad.

5. En este conjunto de reglas borrosas obtenido, se puede comprobar que se cumplen los comportamientos esperados dados los tipos de políticas que se estaban aplicando. Además, se tiene información cualitativa sobre el comportamiento de los parámetros de entrada para llegar a obtener los diversos estados en las variables de salida del PDS.

6. Se comprueba que la política de gestión de personal es dominante frente a la de tiempo de entrega

Al aplicar la herramienta, pero considerando sólo las variables Coste y Calidad simultáneamente, tampoco se obtienen reglas en las que ambos tomen valores "buenos".

Sin embargo al aplicar la herramienta, pero considerando las variables Tiempo (Schcdt) *y* Calidad (Anerpt) simultáneamente, se obtienen las siguientes reglas de gestión borrosas (figura 5.6).

Se puede apreciar como en este caso sí hay reglas en las que son buenos el tiempo y la calidad simultáneamente. En la primera regla podemos ver que el tiempo de entrega es bajo y la calidad es buena si:

> "el tiempo medio de adaptación es *alto* **Y** el tiempo medio de contratación de nuevos técnicos es *mayor o igual que medio* **Y** el tiempo de entrega es *media o alta* **Y** el tiempo medio de salida de los técnicos es *medio*"

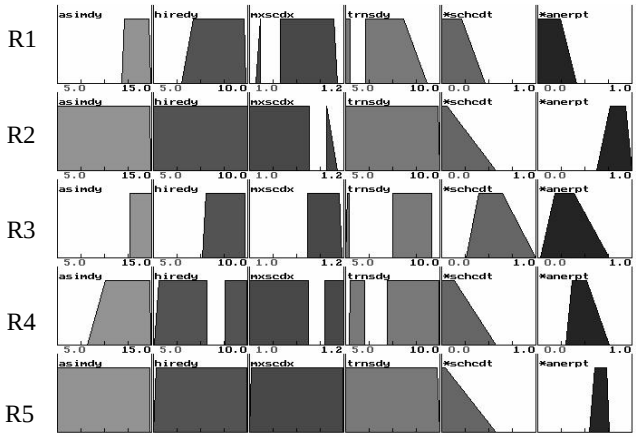


Figura 5.6: Reglas obtenidas con BD: CrCrTi (Tiempo / calidad)

5.4.2.2 Resultados de C4.5 y COGITO

En la Tabla 5.4 se indican los valores máximos y mínimos de cada variable, así como los valores considerados como buenos para las variables analizadas. Considerando los intervalos de corte mostrados en la tabla 5.4, en esta base de datos (CrCrTi) se puede encontrar un registro en el que el esfuerzo es bueno y 47 registros en los que el tiempo es bueno. No se han encontrado registros en los que el esfuerzo y el tiempo sean buenos simultáneamente. Lo anterior avala la teoría de que la imposición de fuertes restricciones en el tiempo de entrega dificulta la obtención de buenos resultados en el esfuerzo.

Nombre	**Rango de Valores Obtenido**	**Rango de Valores Buenos (Intervalos de corte)**
Jbszmd	[1709, 3395]	[1777, 1999]
Schcdt	[349.5, 354.3]	[-, 352]
Anerpt	[0.235, 0.661]	[0.35, 0.45]

Tabla 5.4: Valores de las variables en la BD CrCrTi

En esta base de datos, ninguna de las dos herramientas ha encontrado ninguna regla en la que se consigan buenos resultados para las tres variables del proyecto de forma simultánea. Este hecho también quedó comprobado tras la aplicación de PreFuRGe, con la que, de forma gráfica, se podían ver las distintas combinaciones de valores de las variables existentes en la base de datos.

Al aplicar ambas herramientas pero considerando sólo las salidas: Coste y Calidad simultáneamente, tampoco se obtienen reglas en las que ambos tomen valores “buenos”, tal y como ocurría con PreFuRGe. Sin embargo al aplicar las herramientas, pero

considerando las salidas Tiempo (Schcdt) *y* Calidad (Anerpt) simultáneamente, se obtienen las siguientes reglas de gestión:

- COGITO: Se han obtenido 3 reglas para 36 registros que lo cumplen, con 2 errores. Estas reglas son:
  El tiempo y la calidad serán buenos simultáneamente si:

  R1: Adecuación ≥ 16.6 y Contratación ≥ 6.3 y
  1.1≤Restricción en tiempo ≤ 1.15 (18 registros)
  R2: Adecuación ≥ 8.6 y 6.8 ≤ Contratación ≤ 9.5 y
  Restricción en tiempo ≥ 1.16 (18 registros)
  R3: 10.5 ≤ Adecuación ≤ 14.8 y Contratación ≥ 9.2 y
  Restricción en tiempo ≥ 1.16 y 6.9 ≤ Salida ≤ 9.7 (13 registros)

- C4.5: Se han obtenido 3 reglas para 22 registros de 45. Estas reglas son:
  El tiempo y la calidad serán buenos simultáneamente si:

  R1: Adecuación ≥ 12.1 y Contratación ≥ 6.2 y
  1.1≤Restricción en tiempo ≤ 1.12 (12 registros)
  R2: Adecuación ≥ 12.1 y Contratación ≥ 6.2 y
  Restricción en tiempo ≥ 1.12 y Salida ≤ 5.49 (6 registros)

  R3: Adecuación ≥ 12.8 y 6.2 ≤ Contratación ≤ 9.5 y
  Restricción en tiempo ≥ 1.16 y Salida ≥ 5.5 (4 registros)

5.4.2.3 Conclusiones

Se comprueba que la información proporcionada por las distintas reglas obtenidas con las dos herramientas anteriores, coincide cualitativamente con la interpretacón cualitativa de la primera regla de las obtenidas con PreFuRGe, mostrada en la figura 5.6, e interpretada en el punto anterior; se puede apreciar que, al tratarse de una regla de decisión borrosa, los intervalos en que se debe mover cada atributo no están definidos de una forma tan determinada como ocurre en las reglas semicualitativas generadas por COGITO o C4.5.

5.4.3 Proyecto sin restricción en el tiempo de entrega ni la política de gestión de personal: SrCoTi

El proyecto simulado se trata de un PDS sin restricciones iniciales en las políticas de gestión de personal y aplicando una política de plazo de entrega sin restricción en el tiempo de entrega, ya que, como se observa en la figura 5.7, tanto el atributo que representa el porcentaje máximo permitido en el tiempo de entrega respecto del inicial estimado (Mxscdx), como el asociado al retraso medio en la incorporación de nuevos técnicos (Hiredy), se mueven dentro de todo el intervalo posible de valores, es decir, se permite cualquier desviación.

5.4.3.1 Resultados de PreFuRGe

Las reglas de gestión borrosas obtenidas al aplicar la herramienta PreFuRGe a esta base de datos y considerando de manera simultánea las tres variables del proyecto Coste (Jbszmd), Tiempo (Schcdt) y Calidad (Anerpt), son las mostradas en la figura 5.7.

En estos resultados se puede observar cómo, a diferencia del caso anterior con restricción en el tiempo de entrega, el tiempo de entrega (Schcdt) toma valores dentro de todo su rango posible, tal y como era de esperar dado el tipo de política aplicada respecto al plazo de entrega, donde no hay ninguna restricción inicial en el tiempo de entrega del proyecto.

Por otro lado se observa que el parámetro referente al tiempo medio de contratación (Hiredy) se mueve entre 5 y 40 días, es decir, se comprueba que se está aplicando una política de contratación de personal sin ninguna restricción inicial.

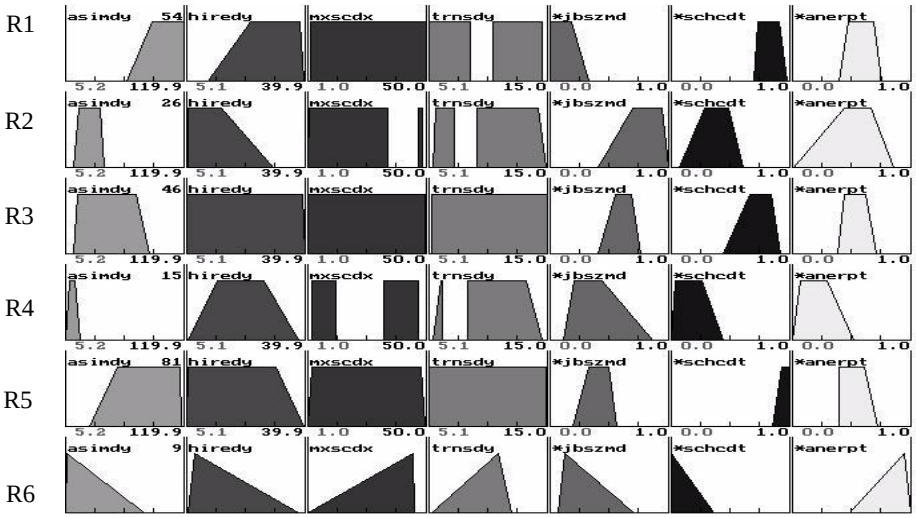


Figura 5.7: Reglas obtenidas con BD: SrCoTi

Un análisis de las reglas de gestión borrosas generadas por la herramienta PreFuRGe aplicándola a la BD de PDS simulado, comentada anteriormente, nos lleva a obtener las siguientes conclusiones:

1. Se puede observar que, al tratarse de un proyecto con una política de plazo de entrega sin restricción inicial, el parámetro asociado al tiempo de entrega (Schcdt) se mueve entre todo el rango del intervalo.

2. Las reglas que se descartarían en este caso son la quinta debido a que la calidad (Anerpt) en esta regla es mala, y las reglas tercera y quinta ya que la mayoría de los atributos se pueden mover en todo su rango de valores, lo que las hace unas reglas poco significativas para la toma de decisiones.

3. Los mejores resultados para las tres variables, simultáneamente, se obtienen en la regla 4, en la que se aprecia claramente como se obtiene un coste (Jbszmd) de medio a bajo, con un tiempo de entrega (Schcdt) bajo y una calidad (Anerpt) aceptable. La interpretación cualitativa de esta regla, que también genera PreFuRGe de manera automática es la siguiente:

   Si "el tiempo medio de adaptación es *muy bajo* **Y** el tiempo medio de contratación de nuevos técnicos es *medio* **Y** el tiempo de entrega es *muy bajo* **O** *alto* **Y** el tiempo medio de salida de los técnicos es *muy bajo* **O** *mayor que medio* "
   Entonces
   "el coste es *de medio a bajo* **Y** el tiempo de entrega es *bajo* **Y** la calidad es *buena*"

4. En la regla 1 se puede ver cómo al reducir el coste a valores bajos, la calidad baja bastante y el tiempo de entrega del proyecto aumenta considerablemente a valores muy altos.
5. En este conjunto de reglas borrosas obtenido, se puede comprobar que se cumplen los comportamientos esperados dados los tipos de políticas que se estaban aplicando. Además, se tiene información cualitativa sobre el comportamiento de los parámetros de entrada, para llegar a obtener los diversos estados en las variables de salida del PDS.

Dado que en un PDS con este tipo de políticas es difícil encontrar reglas que hagan buenas las tres variables (coste, tiempo y calidad) a la vez, vamos a considerarlas de dos en dos, manteniendo la calidad del proyecto en rangos considerados como buenos, es decir, con un número medio de errores por tarea bajo.

Al aplicar la herramienta, pero considerando sólo las salidas Coste (Jbszmd) y Calidad (Anerpt) simultáneamente, se obtienen las reglas mostradas en la Figura 5.8.

En la regla cuarta se puede apreciar como se obtienen muy buenos valores para el esfuerzo (Jbszmd), y valores medios en la calidad (Anerpt), si: "se tienen valores altos en la adecuación (Asimdy) [adecuación lenta] y en la contratación (Hiredy) [Contratación lenta], sin restricción en el tiempo (Mxscdx)".

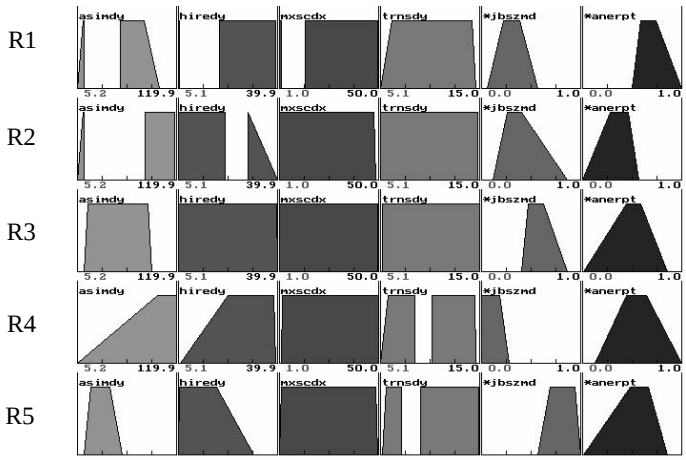


Figura 5.8: Reglas obtenidas con BD: SrCoTi (Coste / calidad)

Sin embargo, al aplicar la herramienta pero considerando las salidas: Tiempo (Schcdt) y Calidad (Anerpt), simultáneamente, se obtienen las reglas de gestión borrosas mostradas en la Figura 5.9.

En la regla quinta se puede apreciar que el tiempo de entrega (Schcdt) es bajo y la calidad (Anerpt) es buena si:

> "la *Adecuación* (Asimdy) es rápida [valores muy bajos], la *Contratación* (Hiredy) es rápida o moderada y la restricción en el *tiempo* (Mxscdx) es moderada o alta, no siendo el retraso medio en la contratación (Trnsdy) un parámetro influyente".

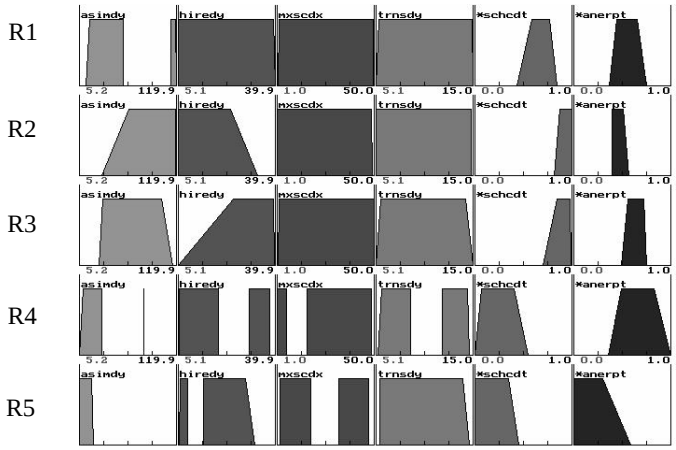


Figura 5.9: Reglas obtenidas con BD: SrCoTi (Tiempo / calidad)

5.4.3.2 Resultados de C4.5 y COGITO

En la Tabla 5.5 se indican los valores máximos y mínimos de cada variable, así como los valores considerados como buenos para las variables analizadas. Considerando los intervalos de corte mostrados en la tabla 5.5, en esta base de datos (SrCoTi) se pueden encontrar 26 registros en los que el esfuerzo es bueno y 27 registros en los que el tiempo es bueno. No se han encontrado registros en los que el esfuerzo y el tiempo sean buenos simultáneamente.

Nombre	Rango de Valores Obtenido	Rango de Valores Buenos (Intervalos de corte)
Jbszmd	[1589, 3241]	[1666, 1888]
Schcdt	[349.5, 437]	[352, 384]
Anerpt	[0.297, 0.556]	[0.35, 0.45]

Tabla 5.5: Valores de las variables en la BD SrCoTi

En esta base de datos, ninguna de las dos herramientas ha encontrado ninguna regla en la que se consigan buenos resultados para las tres variables del proyecto de forma simultánea. Este hecho también quedó comprobado tras la aplicación de PreFuRGe, con la que, de forma gráfica, se pueden ver las distintas combinaciones de valores de las variables existentes en la base de datos.

Al aplicar ambas herramientas considerando sólo las variables Coste (Jbszmd) y Calidad (Anerpt) simultáneamente, se obtienen las siguientes reglas de gestión:

- COGITO: Se han obtenido una regla para 22 registros que lo cumplen, con 0 errores. Estas regla es:
  El coste y la calidad serán buenos simultáneamente si:

  R1: Adecuación ≥ 101 (lenta) y Contratación ≥ 20.3 (lenta) y
  Restricción en tiempo ≥ 6.94 (Sin restricciones)

- C4.5: Se han obtenido dos reglas para 22 registros de 26. Estas reglas son:

  El coste y la calidad serán buenos simultáneamente si:

  R1: 102.9 ≤ Adecuación ≤ 115.5 (lenta) y Contratación ≥ 21.4 (lenta) y
  Restricción en tiempo ≥ 8.18 (sin restricciones)
  R2: Adecuación ≥ 115.5 (lenta) y Contratación ≥ 16.5 (de moderada a lenta)

Al aplicar ambas herramientas pero considerando las salidas: *Tiempo* (Schcdt) *y Calidad* (Anerpt) simultáneamente, considerando un valor de calidad buena *calidad 0.45,* se obtienen las siguientes reglas de gestión:

- COGITO: Se han obtenido dos reglas para 20 registros que lo cumplen, con 1 error. Estas reglas son:

  El tiempo y la calidad serán buenos simultáneamente si:

  R1: 7 ≤ Adecuación ≤ 17.95 (rápida a moderada baja) y
  5.64 ≤ Contratación ≤ 39.5 (rápida, moderada o lenta) y
  Restricción en tiempo ≥ 2.37 (sin restricciones) y
  Salida ≥ 5.44 (rápida o lenta) (11 registros)

  R2: Adecuación ≤ 22.4 (rápida a moderada baja) y
  5.64 ≤ Contratación ≤ 29.5 (de rápida a lenta baja) y
  2.37 ≤ Restricción en tiempo ≤ 48.2 (sin restricciones) y
  5.65 ≤ Salida ≤ 14.03 (rápida o lenta) (12 registros)

  Se observa que son reglas muy similares. La principal diferencia está en la adecuación que debe tender a ser rápida; los demás atributos pueden tomar valores (casi) dentro de todo el rango posible.

- C4.5: No proporcionó ninguna regla de gestión.

Al aplicar ambas herramientas pero considerando las salidas Tiempo (Schcdt) y Calidad (Anerpt), simultáneamente, considerando un valor de calidad buena *calidad 0.35,* se obtienen las siguientes reglas de gestión:

- COGITO: Se han obtenido una regla para 6 registros que lo cumplen, con un error. Estas reglas son:

  El tiempo y la calidad serán buenos simultáneamente si:

  R1: Adecuación ≤ 23.2 (rápida a moderada baja) y
  5.92 ≤ Contratación ≤ 32.8 (de rápida a lenta media) y
  13.1 ≤ Restricción en tiempo ≤ 47.1 (sin restricciones) y
  5.45 ≤ Salida ≤ 10.88 (rápida)

- C4.5: Se han obtenido dos reglas para 6 registros que lo cumplen, con 0 errores. Estas reglas son:

  R1: Adecuación ≤ 13.09 (rápida) y Contratación ≤ 31.04
  (de rápida a lenta media) y Restricción en tiempo ≥ 5.19
  (sin restricciones) y Salida ≤ 7.48 (rápida) (3 registros)

  R2: Adecuación ≤ 10.69 (muy rápida) y 15.36 ≤ Contratación ≤ 31.04
  (de moderada a lenta) y Restricción en tiempo ≥ 5.19
  (sin restricciones) y Salida ≥ 7.48 (de rápida a lenta) (3 registros)

5.4.3.3 Conclusiones

a) Variables Coste y Calidad

Se comprueba que las políticas de gestión de personal lenta favorecen la obtención de buenos resultados en el esfuerzo o coste. Además, se puede ver que la información proporcionada por las reglas obtenidas con las dos herramientas, coincide cualitativamente con la regla cuarta obtenida con PreFuRGe, mostrada en la figura 5.8, e interpretada en el punto anterior.

b) Variables Tiempo y Calidad

Se comprueba que la información cualitativa proporcionada por las reglas obtenidas con las dos herramientas, coincide con la de la regla quinta de las obtenidas con PreFuRGe (figura 5.9). En este caso, con PreFuRGe no hace falta hacer una consideración tan ajustada en la variable calidad para poder obtener reglas de gestión más adecuadas, ya que, como se observa en la regla quinta, quedan contempladas todas las posibilidades exploradas con COGITO y C4.5.

5.5 Conclusiones generales

Las principales conclusiones a las que se puede llegar, a partir de las reglas obtenidas tras la aplicación de las tres herramientas, son:

- En las bases de datos en la que los atributos se pueden mover dentro del intervalo máximo permitido (BD SrCoTi), son muy pocos los escenarios encontrados en los que se cumplan los objetivos del proyecto, es decir, en los que los valores del tiempo, del coste, de la calidad, o de los tres simultáneamente se tomen valores considerados como buenos. Esto puede dar una idea de la complicación que conlleva la gestión de proyectos.
- En las tres bases de datos, el número de registros en los que se cumplen los criterios de tiempo y esfuerzo son notablemente mayores cuando no se imponen restricciones en la calidad. Y ese número de registros, desciende al aumentar las restricciones de calidad.
- Se comprueba que, al imponer al proyecto restricciones en el tiempo de entrega hace que aumente el rango de posibles valores para el esfuerzo. En la base de datos CrCrTi (restricciones en tiempo de entrega) el rango posible para el esfuerzo es de [1709, 3395] y para la base de datos CrSrTi (sin restricciones en tiempo de entrega) es de [1693, 2415]. Esto confirma la idea de que al imponer fuertes restricciones en el tiempo de entrega, mayor será la probabilidad de encarecer el proyecto.
- Podemos comprobar también que el retraso medio en la salida de los técnicos (Trnsdy) no es significativo en la mayoría de los casos para la obtención de buenos resultados.
- COGITO proporciona reglas más concretas que las proporcionadas por C4.5. A su vez, PreFuRGe proporciona unas reglas más cualitativas e interpretables por el gestor del proyecto que ambas herramientas.
- En general el número de atributos en cada regla generada por C4.5 y COGITO es similar por lo que desde ese punto de vista, no existe diferencia

entre ellas. PreFuRGe incluye en cada regla normalmente todos los atributos del proyecto, salvo que éstos no sean significativos para decidir el valor de las variables en estudio.

- Las reglas obtenidas con las tres herramientas utilizadas, recogen la teoría conocida y validada con otros trabajos sobre el comportamiento de proyectos software.
- PreFuRGe, herramienta basada en técnicas de lógica borrosa, presenta una gran ventaja respecto a las otras herramientas con las que se ha comparado: No es necesario etiquetar las salidas. Lo anterior implica además que se puedan conocer reglas para diferentes combinaciones de los resultados finales (tiempo bueno y coste medio, tiempo medio y coste muy bueno, etc.).

En definitiva, podemos concluir que, las principales ventajas de PreFuRGe respecto a las herramientas con las que ha sido comparada son:

1. No hace falta etiquetar los registros de las bases de datos, evitándose así el preprocesamiento, a veces manual y por tanto expuesto a errores, necesario en herramientas basadas en aprendizaje supervisado.
2. Los resultados no tienen que ser interpretados por el experto ya que, tanto la salida en formato gráfico como su interpretación en lenguaje natural, poseen un nivel cualitativo muy cercano a la forma de razonar humana. Estas reglas pueden ser entendidas incluso por personal no experto.
3. Se obtienen reglas de gestión para todas las combinaciones de valores de las variables de salida que existan en la base de datos, con lo que el gestor tiene un conocimiento mucho más completo del comportamiento del proyecto.
4. A la hora de generar las reglas de gestión, se pueden ponderar las variables de salida con pesos, de forma que la herramienta trabajará con un conocimiento mucho más exacto del proyecto y los resultados serán más acertados.
5. El trabajar con conjuntos borrosos en vez de intervalos hace que, sin perder valor cualitativo, la interpretación y posterior aplicación de las reglas sea mucho más natural y eficaz.

Capítulo 6

Aplicación de técnicas tradicionales de predicción

6.1 Introducción

La obtención de información cualitativa a partir de bases de datos numéricas de proyectos de desarrollo de software (PDS), puede lograrse con la utilización de técnicas de Minería de Datos, como se ha podido comprobar en el capítulo anterior.

En esta investigación, la herramienta propuesta está basada en el Fuzzy Clustering, técnica *descriptiva* de KDD, que a diferencia de los clasificadores (i.e. C4.5) que utilizan técnicas *predictivas*, no son capaces de clasificar nuevos datos en las clases más apropiadas. Al utilizar un algoritmo de fuzzy clustering para obtener información cualitativa a partir de bases de datos cuantitativas, no se tiene, a priori, capacidad de predicción de resultados (variables de salida) a partir de nuevos datos (atributos) de proyectos software.

Para predecir estos resultados de salida a partir de nuevos datos, vamos a utilizar diversas técnicas tradicionales de predicción, con el fin de poder analizar y comparar los resultados obtenidos.

Para abordar la predicción de variables en PDS, de los diferentes procesos de estimación descritos en el capítulo 2, hemos optado por aplicar el proceso propuesto por Boehm, por considerarlo el más adecuado para el tipo de información que debe ser procesada (bases de datos cuantitativas de PDS). Además, los modelos de estimación de costes COCOMO (versiones I y II), propuestos por este autor, son de los más conocidos y utilizados por los ingenieros de software (apéndice B).

6.2 Proceso de estimación de Boehm en PDS

Boehm proporciona una amplia visión del proceso de estimación en proyectos de software, y propuso un proceso del que destacan los siguientes pasos: establecimiento de los objetivos, la planificación de la estimación y recopilación de estimaciones realizadas por diversos métodos.

En los siguientes apartados se aplican cada uno de los pasos propuestos por Boehm en su proceso de predicción, para abordar de manera formal, el problema de la estimación de determinadas variables en proyectos de desarrollo de software.

Las bases de datos de PDS que se utilizarán para realizar las estimaciones, son las obtenidas mediante simulación y utilizadas en el capítulo anterior: CrCrTi, CrSrTi y SrCoTi.

6.2.1 Establecer objetivos

En este paso se determinan los objetivos en la estimación, evitando gastar tiempo en recopilar información y realizar estimaciones que no sean relevantes para las decisiones que se deseen tomar.

Las bases de datos cuantitativas, de las que se desea extraer conocimiento están formadas por una serie de atributos de entrada P_i y unas variables de salida S_i cuyos valores dependen en mayor o menor medida de tales atributos. Una vez procesada una determinada base de datos de PDS, y obtenidas las correspondientes reglas de gestión borrosas, habremos extraído la información cualitativa que se deseaba. Pero, si llega una nueva serie de datos P_j, correspondiente a nuevos valores de los atributos del proyecto, para poder conocer el valor de las salidas que le corresponderían, tendríamos que utilizar de nuevo el *simulador de proyectos software* (SPS) para obtener una nueva base de datos con los resultados de la simulación, donde ya aparecerían los nuevos datos junto con los valores de salida correspondientes.

Nuestro objetivo es verificar si podemos estimar las salidas que le corresponderían a una nueva serie de datos sin tener que volver a simular todo el proyecto, y con unos

márgenes de error bajos. Para intentar conseguir este objetivo, se aplicarán las técnicas tradicionales de predicción descritas en el capítulo 3.

6.2.2 Planificar recursos y datos requeridos

Se debe planificar la actividad de estimación, a fin de evitar la preparación de estimaciones con demasiada antelación.

Las variables que se desean estimar a partir de las bases de datos de proyectos de desarrollo de software de que se disponen, son:

- El tiempo de desarrollo medido en días.
- El esfuerzo necesario para la realización medido en técnicos-día (coste)
- La calidad del producto final medido en número medio de errores por tarea

6.2.3 Fijar los requerimientos software

En este punto hay que determinar si las especificaciones en las que están basadas las estimaciones que se realicen son económicamente viables.

Las bases de datos de proyectos de desarrollo de software que se utilizarán para realizar las predicciones son obtenidas mediante simulación, por tanto el coste asociado a la obtención de los datos necesarios es nulo.

Es importante resaltar que lo ideal sería disponer de bases de datos de PDS reales, en cuyo caso si se tendría que considerar un coste asociado a la obtención de la información. En este punto nos encontramos con un problema de difícil solución, que es la dificultad de las empresas de desarrollo de software para obtener bases de datos históricas propias acerca de sus proyectos.

Actualmente, esta deficiencia de datos reales se resuelve gracias a la existencia de potentes entornos de simulación y a la posibilidad de construir modelos dinámicos para la simulación de PDS en una determinada organización.

6.2.4 Utilizar varias técnicas y fuentes independientes

En este punto Boehm indica que se ha de utilizar más de una técnica par realizar las estimaciones. El uso de una combinación de técnicas permite evitar los puntos débiles de cada técnica, así como poder tomar lo mejor de cada una de ellas.

Las técnicas de predicción tradicionales que se van a utilizar en este punto son:

- K-vecinos más cercanos ponderados
- Modelos de Regresión Multivariante:
  - Lineal
  - No Lineal
- Programación lineal: Simplex

Para comprobar el grado de bondad de las estimaciones realizadas, se han utilizado tres medidas de error medio diferentes. El error absoluto medio (1), se va a utilizar para comparar los resultados obtenidos en la estimación de una determinada variable, con los diferentes métodos propuestos. Para poder determinar cuál es el atributo mejor estimado por una determinada técnica, se utilizarán los errores relativo medio (2) y relativo disperso medio (3).

$$ErrorA = \frac{\sum_{i=1}^{n} | Y_i - \hat{Y}_i |}{n} \text{ : Error Absoluto Medio} \quad (1)$$

$$ErrorB = \frac{\sum_{i=1}^{n} \frac{| Y_i - \hat{Y}_i | * 100}{Y_i}}{n} \text{ : Error Relativo Medio} \quad (2)$$

$$ErrorC = \frac{\sum_{i=1}^{n} \frac{| Y_i - \hat{Y}_i |}{| Y_i - \bar{Y}_i |}}{n} \text{ : Error Relativo Disperso Medio} \quad (3)$$

donde Y_i es la salida real de la BD en cuestión e $\hat{Y}_i$ es la salida estimada por cada uno de las técnicas aplicadas. Antes de aplicar las diferentes técnicas, se hará una pequeña referencia al concepto de "*Coeficiente de correlación*" entre variables; esta información se utilizará en la descripción de la técnica wk-NN (weighted k-Nearest Neighbor) propuesta en esta investigación.

6.2.4.1 Análisis de Correlación

Antes de abordar el tema de la estimación de resultados para nuevos datos de entrada, es necesario conocer qué parámetros son más influyentes a la hora de determinar las salidas del proyecto y en qué grado lo son. Con esta información podremos realizar unas estimaciones de resultados mucho más exactas.

El *coeficiente de correlación* es una técnica estadística aplicable a una distribución bidimensional que nos indica la intensidad o grado de dependencia entre una variable *x* y otra *y*. El coeficiente de correlación *r* es un número que se obtiene mediante la fórmula siguiente:

$$r = \frac{\sum (x_i - \bar{x})(y_i - \overline{y})}{N\sigma_x \sigma_y}$$

El numerador es el producto de las desviaciones de los valores *x* e *y* respecto de sus valores medios. En el denominador tenemos las desviaciones cuadráticas medias de *x* y de *y*. El coeficiente de correlación puede valer cualquier número comprendido entre -1 y +1, de forma que la interpretación sería:

- Cuando $r = 1$, la correlación lineal es perfecta, de forma directa entre X e Y
- Cuando $r = -1$, la correlación lineal es perfecta, de forma inversa entre X e Y
- Cuando $r = 0$, no existe correlación alguna, es decir, independencia total de los valores X e Y

Esta información será muy útil para saber qué parámetros (dedicación, retraso medio en la incorporación de técnicos, etc.) del proyecto son los que determinan los valores de una determinada variable de salida (Coste, Tiempo, etc.).

Al realizar un análisis de correlación sobre las tres bases de datos con las que estamos trabajando se han obtenido los siguientes resultados:

A) BD: Gestión de personal rápida, Con restricción en tiempo de entrega (CrCrTi)

BD: CrCrTi	**COSTE**	**TIEMPO**	**CALIDAD**
Asimdy	0.868	0.416	-0.447
Hiredy	-0.008	0.293	-0.256
Mxscdx	-0.36	0.344	-0.331
Trnsdy	-0.016	-0.074	0.128

Tabla 6.1: Análisis de Correlación en BD CrCrTi

En la tabla 6.1 se observa cómo el atributo más influyente sobre las variables es Asimdy, o sea, que la adecuación de los técnicos al proyecto es un parámetro importante para obtener los objetivos del proyecto. A continuación le sigue en relevancia la variable Mxscdx, lo que quiere decir que en un proyecto con restricción en el tiempo de entrega, el porcentaje de aplazamiento es influyente. El retraso en la contratación (Hiredy) influye de alguna forma sobre el tiempo y la calidad, pero muy poco sobre el coste del proyecto.

B) BD: Gestión de personal rápida, Sin restricción en el tiempo de entrega (CrSrTi)

BD: CrSrTi	**COSTE**	**TIEMPO**	**CALIDAD**
Asimdy	0.986	0.154	-0.821
Hiredy	0.042	0.251	-0.224
Mxscdx	-0.078	-0.034	0.074
Trnsdy	0.049	0.032	-0.04

Tabla 6.2: Análisis de Correlación en BD CrSrTi

En este caso (tabla 6.2) podemos ver cómo el atributo más influyente sobre las variables es también Asimdy, principalmente en el coste y la calidad, aunque en este caso no influye tanto sobre el tiempo de entrega. En esta BD, Mxscdx no es un atributo muy influyente sobre las variables del proyecto, como era de esperar, ya que en esta BD no hay restricción en el tiempo de entrega. Por su parte Hiredy sigue teniendo prácticamente la misma influencia sobre las mismas variables que en el caso anterior.

C) BD: Sin restricción en la gestión de personal ni en el tiempo de entrega (SrCoTi)

BD: SrCoTi	**COSTE**	**TIEMPO**	**CALIDAD**
Asimdy	-0.665	0.708	0.075
Hiredy	-0.422	0.21	0.384
Mxscdx	-0.018	0.108	-0.005
Trnsdy	0.06	0.084	-0.036

Tabla 6.3: Análisis de Correlación en BD SrCoTi

En esta BD se puede observar (tabla 6.3) cómo el atributo más influyente sobre las variables vuelve a ser Asimdy, sobre todo en el coste y el tiempo, aunque en este caso no influye casi nada sobre la calidad del proyecto. En esta BD, Mxscdx tampoco es un atributo muy influyente sobre las variables, como era de esperar, ya que en esta BD no

hay restricción en el tiempo de entrega. Por su parte Hiredy pasa a ser el segundo atributo más relevante teniendo una influencia considerable sobre todas las variables.

Con esta información obtenida, a la hora de realizar estimaciones sobre las variables del proyecto, a partir de nuevos datos de entrada, tendremos una información muy útil sobre todo a la hora de aplicar algoritmos en los que haya que calcular distancias entre distintos puntos de las Bases de Datos.

6.2.4.2 Algoritmo wk-NN: weighted k-Nearest Neighbor

El algoritmo wk-NN desarrollado es una modificación del clásico algoritmo k-NN (k-nearest neighbor), donde se pondera el peso *w* de cada atributo, asociando dicho peso con el coeficiente de correlación de cada atributo con cada variable del PDS; los coeficientes de correlación obtenidos para cada una de las bases de datos son los mostrados en las tablas 6.1, 6.2 y 6.3. Se ha realizado una comparación de la calidad de la predicción sin incluir, e incluyendo estos pesos en el proceso del algoritmo, y se ha comprobado que los resultados mejoraron considerablemente al incluirlos. Los datos con los que trabaja el algoritmo wk-NN son de la forma: $X= x_1, x_2,....., x_n.$, donde la dimensión de *X* es igual al número de atributos del proyecto en cuestión.

El criterio utilizado en el algoritmo propuesto para calcular la distancia ponderada entre dos puntos *X* y *Z,* de una base de datos es:

$$d(X,Z)=\sqrt{\sum_{i=1}^{n}(X_i-Z_i)^2} \qquad (5)$$

Si ponderamos la importancia de los parámetros, usaremos un peso w_i para rectificar la distancia, con lo que obtendremos:

$$d(X,Z)=\sqrt{\sum_{i=1}^{n}w_i(X_i-Z_i)^2} \qquad (6)$$

Como ya se ha comentado anteriormente, el peso w_i se calculará a partir de los coeficientes de correlación (ρ) de cada atributo x_i respecto a cada variable del proyecto y_i. En concreto a cada atributo a_j le asociaremos un peso w_j igual a la media de los coeficientes de correlación de dicho atributo respecto a cada una de las salidas, es decir:

$$w_j=\sum_{i=1}^{n}\rho(a_j,y_i)\Big/_{n} \qquad (7)$$

El objetivo es obtener un modelo de predicción a partir de los resultados obtenidos (bases de datos de PDS) mediante la simulación del proyecto de software con el uso de SPS, y comparar los resultados que se obtengan con el modelo de predicción con los valores obtenidos en la simulación del proyecto.

La técnica de los *k*-vecinos más cercanos establece que cada variable estimada, para una nueva serie de datos, se calcula de la siguiente forma:

$$\hat{y}_i = \frac{y_{i1} + y_{i2} + ... + y_{ik}}{k} \tag{8}$$

Donde la estimación de una variable para una nueva serie de datos n-dimensional *A*, se calcula como la media de la suma de los valores en dicha variable de los *k* datos vecinos más cercanos (y_{ik}: valor de la variable y_i en el vecino más cercano *k-ésimo*) al nuevo dato A, en orden de cercanía, según la formula de distancia definida anteriormente. Este cálculo de la salida estimada se puede ponderar, de forma que la distancia de uno de los vecinos al dato A, influya en el cálculo de cada variable estimada $\hat{y}$:

$$\hat{y}_i = \frac{\alpha_1 y_{i1} + \alpha_2 y_{i2} + ... + \alpha_k y_{ik}}{k} \tag{9}$$

Donde cada α_i se calcula como el inverso de la distancia del vecino *i-ésimo* al punto A, cuya salida se desea estimar. Para verificar la bondad de esta técnica de predicción, hemos estimado las salidas de cada uno de los valores de las tres bases de datos descritas anteriormente, de forma que pudiésemos comprobar el error cometido en cada estimación ya que la salida correcta es conocida.

En todos los casos se ha considerado una *k=3* ya que al utilizar una k mayor, es decir, más vecinos, no se obtuvieron mejores resultados.

Los errores cometidos al estimar cada variable en cada una de las tuplas de las distintas bases de datos, usando la técnica de los k-vecinos considerando los pesos w_i (coeficientes de correlación) en el cálculo de las distancias, son los siguientes:

A) BD: Gestión de personal rápida, con restricción en el tiempo de entrega (CrCrTi)

BD: CrCrTi *wk-NN*	**Estimación de COSTE**	**Estimación de TIEMPO**	**Estimación de CALIDAD**
Error A	31.81	0.039	0.029
Error B	1.363	0.011	6.515
Error C	0.372	0.115	1.446

Tabla 6.4: Resultados con *wk-NN* en BD CrCrTi

En esta tabla (6.4) se observa que si analizamos el error relativo medio, el tiempo es el atributo mejor estimado, mientras que teniendo en cuenta el promedio de valores de cada variable (error C), es el coste la variable mejor estimada.

B) BD: Gestión de personal rápida, Sin restricción en el tiempo de entrega (CrSrTi)

BD: CrSrTi *wk-NN*	**Estimación de COSTE**	**Estimación de TIEMPO**	**Estimación de CALIDAD**
Error A	11.30	0.169	0.023
Error B	0.558	0.048	6.038
Error C	0.143	0.291	1.147

Tabla 6.5: Resultados con *wk-NN* en BD CrSrTi

En esta otra tabla (6.5) se observa que si analizamos el error B, el tiempo es el parámetro mejor estimado, mientras que teniendo en cuenta el error relativo disperso medio, es el coste la variable mejor estimada.

C) BD: Sin restricción en la gestión de personal ni el tiempo de entrega (SrCoTi)

BD: SrCoTi *wk-NN*	**Estimación de COSTE**	**Estimación de TIEMPO**	**Estimación de CALIDAD**
Error A	51.91	2.689	0.009
Error B	2.33	0.68	2.21
Error C	0.782	0.500	1.178

Tabla 6.6: Resultados con *wk-NN* en SrCoTi

En estos resultados (tabla 6.6), se vuelve a verificar que el tiempo es el parámetro que se logra estimar con mayor precisión sea cual sea la medida de error utilizada. En este caso la calidad no es el parámetro peor estimado, ya que el coste es estimado casi con el mismo grado de error.

Conclusiones: wk-NN

Los resultados obtenidos con las distintas bases de datos (tablas 6.4, 6.5 y 6.6), muestran que con esta técnica (wk-NN), el tiempo es el parámetro que se logra estimar con mayor precisión. Analizando el error C en las tres bases de datos procesadas, se observa que todas las variables son estimadas con bastante exactitud, obteniéndose los mejores resultados al estimar el coste y el tiempo.

En definitiva, podemos concluir que la técnica propuesta permite estimar el valor de las variables de un PDS con un grado de error realmente bajo. Esto permitiría al gestor del proyecto conocer el valor que tomarían las variables del proyecto (coste, tiempo y calidad) ante una nueva serie de valores en los atributos, sin necesidad de simular todo el proyecto de nuevo.

6.2.4.3 Modelos de Regresión Multivariante.

Cuando se construyen modelos de regresión se debe tener en cuenta el efecto de determinados casos anómalos (outliers). Estos casos se pueden identificar mediante la distancia de Mahalanobis, para valores inusuales de la variable independiente, y mediante la distancia de Cook, que mide el impacto de un determinado punto en las estimaciones de los parámetros [Duda73], [Cook99], [Chatt00]. Para determinar el modelo más aceptable también se han de tener en cuenta criterios como la linealidad de los datos. Si el número de casos es suficientemente grande, se suelen utilizar las dos terceras partes de la población para construir el modelo de regresión. No obstante, dadas las características de los conjuntos de datos que se manejan en Ingeniería de Software, casi todos los autores usan todos los puntos, tanto para la construcción como para la evaluación del modelo. En líneas generales, se asume que al utilizar todos los puntos se sobreestima la capacidad de predicción, aunque eso no tiene por qué ser así en todos los casos. En este sentido, se puede encontrar una evaluación de las capacidades de predicción con varias muestras aleatorias en [Dolad99].

6.2.4.3.1 Modelos de Regresión Lineal Multivariante: RLM

Para calcular los modelos de regresión lineal de cada una de las variables a partir de las distintas bases de datos, se ha utilizado *DataFit* [Dataf01], una potente herramienta de análisis de datos.

En este sentido, Dolado [Dolad98] describe las posibilidades de la Regresión lineal, la Programación genética y las Redes neuronales como técnicas de estimación en proyectos de desarrollo de Software. Krishnamoorthy por su parte, en [Krish02] presenta uno de los últimos trabajos relacionados con la predicción de información utilizando regresión lineal.

Los errores cometidos al estimar el valor de las variables de cada una de las tuplas de las distintas bases de datos, usando la técnica de RLM (Regresión Lineal Multivariante) y utilizando la forma de función lineal (10) se muestran a continuación.

A) Forma de Función lineal:

$$\textbf{Variable} = \textbf{a}\ \text{Asimdy} + \textbf{b}\ \text{Hiredy} + \textbf{c}\ \text{Mxscdx} + \textbf{d}\ \text{Trnsdy} + \textbf{e} \qquad (10)$$

donde a, b, c, d, e son los coeficientes que se desean calcular para cada una de las variables a estudiar.

A.1) BD: *Gestión de personal rápida con restricción en tiempo de entrega (CrCrTi)*

Las ecuaciones lineales obtenidas para cada una de las variables del proyecto, que permitirán realizar las predicciones para nuevos valores en los atributos son:

Coste = **123.7** Asimdy – **23.4** Hiredy – **2915.3** Mxscdx + **9.19** Trnsdy + **4321.9**

Tiempo = **0.0717** Asimdy + **0.102** Hiredy + **3.222** Mxscdx – **0.0148** Trnsdy + **344.77**

Calidad = **-0.010** Asimdy – **0.011** Hiredy – **0.39** Mxscdx + **0.0045** Trnsdy + **1.10**

El error cometido al predecir el valor de cada variable para cada una de las tuplas de atributos de la base de datos CrCrTi, utilizando las ecuaciones obtenidas y medido con tres fórmulas de error diferentes, es el mostrado en la tabla siguiente (6.7):

BD: CrCrTi RLM	**Estimación de COSTE**	**Estimación de TIEMPO**	**Estimación de CALIDAD**
Error A	88.58	0.30	0.04
Error B	4.00	0.08	9.63
Error C	1.08	1.30	1.49

Tabla 6.7: Resultados con RLM en BD CrCrTi

En esta tabla (6.7) se observa que analizando el error relativo medio, el tiempo es el atributo mejor estimado con bastante diferencia, mientras que teniendo en cuenta el

error C, es el coste la variable mejor estimada. Cabe destacar que, observando esta medida de error, las tres variables se estiman prácticamente con la misma exactitud.

A.2) BD: *Gestión de personal rápida sin restricción en tiempo de entrega (CrSrTi)*

Las ecuaciones lineales obtenidas para cada una de las variables del proyecto, que permitirán realizar las predicciones para nuevos valores en los atributos son:

Coste = **63.54** Asimdy – **5.74** Hiredy + **0.0066** Mxscdx + **0.0623** Trnsdy + **1389**

Tiempo = **0.698** Asimdy + **0.377** Hiredy + **0.00252** Mxscdx – **0.011** Trnsdy + **341**

Calidad = **- 0.0221** Asimdy – **0.0088** Hiredy + **0.00011** Mxscdx + **0.00019** Trnsdy + **0.739**

El error cometido al predecir el valor de cada variable para cada una de las tuplas de la base de datos CrSrTi, utilizando las ecuaciones obtenidas y medido con tres tipo de error diferentes, es el mostrado en la tabla siguiente (tabla 6.8):

BD: CrSrTi RLM	**Estimación de COSTE**	**Estimación de TIEMPO**	**Estimación de CALIDAD**
Error A	24.76	1.49	0.03
Error B	1.24	0.42	8.31
Error C	0.36	1.89	1.29

Tabla 6.8: Resultados con RLM en BD CrSrTi

En esta tabla (6.8) se observa que analizando el error relativo medio, el tiempo es el atributo mejor estimado, mientras que teniendo en cuenta el error relativo disperso medio, es el coste la variable mejor estimada, con bastante diferencia respecto a las otras dos variables; además el error de la estimación casi coincide con el obtenido al estimar el tiempo con el error B.

A.3) BD: *Sin restricción en la gestión de personal ni el tiempo de entrega* (SrCoTi)

Las ecuaciones lineales obtenidas para cada una de las variables del proyecto, que permitirán realizar las predicciones para nuevos valores en los atributos son:

Coste = **-6.7** Asimdy – **13.2** Hiredy + **0.117** Mxscdx + **5.39** Trnsdy + **2965.7**

Tiempo = **0.465** Asimdy + **0.417** Hiredy + **0.0902** Mxscdx + **0.662** Trnsdy + **370**

Calidad = **0.000054** Asimdy + **0.0011** Hiredy + **0.000043** Mxscdx – **0.00016** Trnsdy + **0.40**

El error cometido al predecir el valor de cada variable para cada una de las tuplas de la base de datos SrCoTi, utilizando las ecuaciones obtenidas y medido con tres errores diferentes, es el mostrado en la tabla siguiente (tabla 6.9):

BD: SrCoTi RLM	**Estimación de COSTE**	**Estimación de TIEMPO**	**Estimación de CALIDAD**
Error A	138.8	12.45	0.02
Error B	6.19	3.05	4.62
Error C	1.29	3.47	2.79

Tabla 6.9: Resultados con RLM en BD SrCoTi

En esta tabla (6.9) se observa que teniendo en cuenta el error B, el tiempo es el atributo mejor estimado, mientras que teniendo en cuenta el error relativo disperso medio, es el coste la variable mejor estimada. En este caso se puede apreciar que el tiempo es estimado casi con la misma exactitud considerando cualquiera de las dos medidas de error (B o C).

Conclusiones: RLM

Los resultados obtenidos con las distintas bases de datos (tablas 6.7, 6.8 y 6.9), muestran que con esta técnica (RLM), y analizando el error relativo medio, el tiempo es el parámetro que se logra estimar con una mayor precisión. No obstante, considerando el error relativo disperso medio (error C), se observa que todas las variables son estimadas con una exactitud muy similar, en las tres bases de datos procesadas, obteniéndose los mejores resultados al estimar el coste, sobre todo en las bases de datos CrSrTi y SrCoTi.

Podemos concluir que con esta técnica se puede estimar el valor de las variables de un PDS con un grado de error aceptable, aunque mayor que el obtenido con la técnica wk-NN. Por otro lado, cabe destacar que una vez obtenidas las ecuaciones lineales para cada variable, las estimaciones se realizan de manera muy rápida y con un coste computacional realmente bajo.

6.2.4.3.2 Modelos de regresión no lineal multivariante: RNLM

Para calcular los modelos de regresión no lineal de cada una de las variables, a partir de las distintas bases de datos, se ha utilizado *DataFit* [Dataf01], una herramienta de análisis de datos.

Los errores cometidos al estimar el valor de las variables de cada una de las tuplas de las distintas bases de datos, usando la técnica de RNLM (Regresión No Lineal Multivariante) y utilizando las formas de función no lineal polinómica y exponencial, son los siguientes:

A) Forma de Función Polinómica

La forma de función polinómica que deseamos calcular es:

$$\textbf{Variable} = \mathbf{a}\ \text{Asimdy}^{\mathbf{a1}} + \mathbf{b}\ \text{Hiredy}^{\mathbf{b1}} + \mathbf{c}\ \text{Mxscdx}^{\mathbf{c1}} + \mathbf{d}\ \text{Trnsdy}^{\mathbf{d1}} + \mathbf{e}$$

donde a, a1, b, b1, c, c1, d, d1, e son los coeficientes que se desean calcular para cada una de las variables.

A.1) BD: *Gestión de personal rápida con restricción en tiempo de entrega* (CrCrTi)

Las ecuaciones no lineales polinómicas obtenidas para cada una de las variables del proyecto, que permitirán realizar las predicciones para nuevos valores en los atributos son:

$$\textbf{Coste} = \mathbf{34.27}\ \text{Asimdy}^{1.41} - \mathbf{0.04}\ \text{Hiredy}^{3.49} - \mathbf{1013.11}\ \text{Mxscdx}^{2.48} + \mathbf{515}\ \text{Trnsdy}^{0.09} + \mathbf{2055.1}$$

$$\textbf{Tiempo} = \mathbf{17.61}\ \text{Asimdy}^{0.03} + \mathbf{105.18}\ \text{Hiredy}^{0.006} + \mathbf{112.8}\ \text{Mxscdx}^{0.03} + \mathbf{111.29}\ \text{Trnsdy}^{-0.001}$$

$$\textbf{Calidad} = \mathbf{6.05}\ \text{Asimdy}^{-0.01} + \mathbf{2.54}\ \text{Hiredy}^{-0.03} + \mathbf{3.86}\ \text{Mxscdx}^{-0.11} + \mathbf{0.42}\ \text{Trnsdy}^{0.072} - \mathbf{12.02}$$

El error cometido al predecir el valor de cada variable para cada una de las tuplas de atributos de la base de datos CrCrTi, utilizando las ecuaciones obtenidas y medido con tres fórmulas diferentes, es el mostrado en la tabla siguiente (tabla 6.10):

BD: CrCrTi RNLM Pol.	**Estimación de COSTE**	**Estimación de TIEMPO**	**Estimación de CALIDAD**
Error A	87.23	0.30	0.05
Error B	3.94	0.09	9.93
Error C	0.93	1.31	1.50

Tabla 6.10: Resultados con RNLM en BD CrCrTi (Polinómica)

En esta tabla (6.10) se observa que si analizamos el error relativo medio, el tiempo es el atributo mejor estimado con mucha diferencia respecto a las otras variables,

mientras que teniendo en cuenta el error C, es el coste la variable mejor estimada; con este error, las tres variables son estimadas con la misma exactitud aproximadamente.

A.2) BD: *Gestión de personal rápida sin restricción en tiempo de entrega* (CrSrTi)

Las ecuaciones no lineales polinómicas obtenidas para cada una de las variables del proyecto, que permitirán realizar las predicciones para nuevos valores en los atributos son:

$$\textbf{Coste} = \mathbf{2.32}\ \text{Asimdy}^{2.11} - \mathbf{2.30}\ \text{Hiredy}^{1.28} - \mathbf{59.5}\ \text{Mxscdx}^{0.020} + \mathbf{66.28}\ \text{Trnsdy}^{0.10} + \mathbf{1661.22}$$

$$\textbf{Tiempo} = \mathbf{0.17}\ \text{Asimdy}^{1.44} + \mathbf{49.09}\ \text{Hiredy}^{0.049} + \mathbf{229.71}\ \text{Mxscdx}^{0.00013} + \mathbf{61.99}\ \text{Trnsdy}^{-0.0007}$$

$$\textbf{Calidad} = \mathbf{3.76}\text{Asimdy}^{-0.06} + \mathbf{0.35}\text{Hiredy}^{-0.14} + \mathbf{0.024}\ \text{Mxscdx}^{0.084} + \mathbf{0.038}\ \text{Trnsdy}^{0.125} - \mathbf{3.320}$$

El error cometido al predecir el valor de cada variable para cada una de las tuplas de atributos de la base de datos CrSrTi, utilizando las ecuaciones obtenidas y medido con tres errores diferentes, es el mostrado en la tabla siguiente (tabla 6.11):

BD: CrSrTi RNLM Pol.	**Estimación de COSTE**	**Estimación de TIEMPO**	**Estimación de CALIDAD**
Error A	13.02	1.41	0.04
Error B	0.66	0.40	9.08
Error C	0.12	1.98	2.19

Tabla 6.11: Resultados con RNLM en BD CrSrTi (Polinómica)

En esta tabla (6.11) se observa que analizando el error B, el tiempo es el atributo mejor estimado, aunque en este caso el coste también se logra estimar con bastante exactitud. Por otro lado, teniendo en cuenta el error relativo disperso medio, es el coste la variable mejor estimada con mucha diferencia respecto a las otras dos.

A.3) BD: *Sin restricción en la gestión de personal ni el tiempo de entrega* (SrCoTi)

Las ecuaciones no lineales polinómicas obtenidas para cada una de las variables del proyecto, que permitirán realizar las predicciones para nuevos valores en los atributos son:

Coste = **2537.48** $\text{Asimdy}^{-0.11}$ + **1636.24** $\text{Hiredy}^{-0.32}$ + **1.045** $\text{Mxscdx}^{0.39}$ + **0.016** $\text{Trnsdy}^{3.12}$

Tiempo = **496.37**$\text{Asimdy}^{0.044}$ + **60.6**$\text{Hiredy}^{0.095}$ + **30.6**$\text{Mxscdx}^{0.044}$ + **12.05**$\text{Trnsdy}^{0.24}$ – **310.8**

Calidad = **0.58** $\text{Asimdy}^{0.01}$ – **0.87** $\text{Hiredy}^{-0.02}$ – **0.051** $\text{Mxscdx}^{0.014}$ – **0.038** $\text{Trnsdy}^{0.063}$ + **0.74**

El error cometido al predecir el valor de cada variable para cada una de las tuplas de atributos de la base de datos SrCoTi, utilizando las ecuaciones obtenidas y medido con tres fórmulas diferentes, es el mostrado en la tabla siguiente (tabla 6.12):

BD: SrCoTi RNLM Pol.	**Estimación de COSTE**	**Estimación de TIEMPO**	**Estimación de CALIDAD**
Error A	223.95	9.22	0.02
Error B	10.09	2.24	4.84
Error C	1.82	3.02	15.67

Tabla 6.12: Resultados con RNLM en BD SrCoTi (Polinómica)

En esta tabla (6.12) se observa que teniendo en cuenta el error relativo medio, el tiempo es el atributo mejor estimado, mientras que considerando el error relativo disperso medio, es el coste la variable mejor estimada.

B) Forma de Función Exponencial

La forma de función exponencial que deseamos calcular es :

Variable = **EXP (a** Asimdy + **b** Hiredy + **c** Mxscdx + **d** Trnsdy + **e)**

donde a, b, c, d, e son los coeficientes que se desean calcular para cada una de las variables.

B.1) BD: *Gestión de personal rápida con restricción en tiempo de entrega* (CrCrTi)

Las ecuaciones no lineales exponenciales obtenidas para cada una de las variables del proyecto, que permitirán realizar las predicciones para nuevos valores en los atributos son:

Coste = **EXP (0.055** Asimdy – **0.01** Hiredy – **1.37** Mxscdx + **0.004** Trnsdy + **8.69)**

Tiempo = **EXP (0.0002** Asimdy + **0.0003** Hiredy + **0.009** Mxscdx – **0.00004**Trnsdy + **5.84)**

Calidad = **EXP (-0.01** Asimdy – **0.02** Hiredy – **0.072** Mxscdx + **0.008** Trnsdy + **0.40)**

El error cometido al predecir el valor de cada variable para cada una de las tuplas de atributos de la base de datos CrCrTi, utilizando las ecuaciones obtenidas y medido con tres fórmulas diferentes, es el mostrado en la tabla siguiente (tabla 6.13):

BD: CrCrTi RNLM Exp.	**Estimación de COSTE**	**Estimación de TIEMPO**	**Estimación de CALIDAD**
Error A	69.60	0.30	0.10
Error B	3.18	0.09	9.72
Error C	0.70	1.38	1.69

Tabla 6.13: Resultados con RNLM en BD CrCrTi (Exponencial)

En esta tabla (6.13) se observa que analizando el error B, el tiempo es el atributo mejor estimado llegándose a obtener una alta exactitud en la estimación, mientras que teniendo en cuenta el error C, es el coste la variable mejor estimada.

B.2) BD: *Gestión de personal rápida sin restricción en tiempo de entrega* (CrSrTi)

Las ecuaciones no lineales exponenciales obtenidas para cada una de las variables del proyecto, que permitirán realizar las predicciones para nuevos valores en los atributos, son:

Coste = **EXP (0.03** Asimdy – **0.003** Hiredy – **0.000002** Mxscdx + **0.000048** Trnsdy + **7.28)**

Tiempo = **EXP (0.002** Asimdy + **0.001**Hiredy + **0.000007**Mxscdx – **0.00003**Trnsdy + **5.83)**

Calidad = **EXP (-0.047** Asimdy – **0.017** Hiredy + **0.00012**Mxscdx + **0.001**Trnsdy – **0.197)**

El error cometido al predecir el valor de cada variable para cada una de las tuplas de atributos de la base de datos CrSrTi, utilizando las ecuaciones obtenidas y medido con tres tipos de error diferentes, es el mostrado en la tabla siguiente (tabla 6.14):

BD: CrSrTi RNLM Exp.	**Estimación de COSTE**	**Estimación de TIEMPO**	**Estimación de CALIDAD**
Error A	19.36	1.51	0.03
Error B	0.97	0.43	8.58
Error C	0.26	2.20	2.25

Tabla 6.14: Resultados con RNLM en BD CrSrTi (Exponencial)

En esta tabla (6.14) se observa que considerando el error relativo medio, el tiempo es el atributo mejor estimado seguido del coste, mientras que teniendo en cuenta el error C, el coste es la variable mejor estimada con bastante diferencia.

B.3) BD: *Sin restricción en la gestión de personal ni el tiempo de entrega* (SrCoTi)

Las ecuaciones no lineales exponenciales obtenidas para cada una de las variables del proyecto, que permitirán realizar las predicciones para nuevos valores en los atributos son:

Coste = **EXP (-0.003**Asimdy – **0.0055**Hiredy + **0.00003** Mxscdx + **0.0026** Trnsdy + **8.004)**

Tiempo = **EXP (0.001** Asimdy + **0.0009**Hiredy + **0.0002**Mxscdx + **0.0015** Trnsdy + **5.92)**

Calidad = **EXP (0.0001**Asimdy + **0.003**Hiredy + **0.0009**Mxscdx – **0.0003**Trnsdy – **0.906)**

El error cometido al predecir el valor de cada variable para cada una de las tuplas de atributos de la base de datos SrCoTi, utilizando las ecuaciones obtenidas y medido con tres fórmulas diferentes, es el mostrado en la tabla siguiente (tabla 6.15):

BD: SrCoTi RNLM Exp.	**Estimación de COSTE**	**Estimación de TIEMPO**	**Estimación de CALIDAD**
Error A	153.19	12.63	0.02
Error B	6.88	3.10	4.61
Error C	1.44	3.46	25.57

Tabla 6.15: Resultados con RNLM en BD SrCoTi (Exponencial)

En esta tabla (6.15) se observa que si teniendo en cuenta el error B, el tiempo es el atributo mejor estimado seguido de la calidad, mientras que analizando el error C, es el coste la variable mejor estimada, y la calidad la peor con mucha diferencia.

Conclusiones: RNLM

Los resultados obtenidos con las distintas bases de datos (tablas 6.10, 6.11, 6.12, 6.13, 6.14 y 6.15), muestran que con esta técnica (RNLM), y analizando el error relativo medio, el tiempo es el parámetro que se logra estimar con una mayor precisión, con las dos formas de función usadas (polinómica y exponencial). No obstante, considerando el error relativo disperso medio, se observa que, a diferencia del caso anterior (RLM), no todas las variables son estimadas con similar exactitud; en este caso los mejores resultados se obtienen al estimar el coste, en todas las bases de datos.

Podemos concluir que con esta técnica se puede estimar el valor de las variables de un PDS con un grado de error aceptable, aunque en general mayor que el obtenido con la técnica RLM. Además, la obtención de las ecuaciones no lineales para cada variable, es bastante más costosa que en el caso anterior.

6.2.4.4 Programación Lineal: Simplex

Como ya se describió en el capítulo 3, un problema de *programación lineal o programa lineal (PL)* es un problema de optimización en el que la función objetivo y todas las funciones que aparecen en las restricciones son lineales.

A partir de esta definición, y con los resultados obtenidos en la sección anterior, se ha propuesto en esta investigación, plantear el problema de la toma de decisiones en Proyectos de Desarrollo de Software desde el prisma de la programación lineal. En este sentido, dado que hemos podido representar cada una de las variables (coste, tiempo y calidad) como una función lineal, dependiente de los atributos del proyecto, vamos a buscar los valores de los atributos que maximicen o minimicen dichas variables.

En resumen, lo que intentamos resolver son Problemas de Programación Lineal con la forma expresada en el siguiente ejemplo:

Minimizar **FO: 123.7** Asimdy – **23.4** Hiredy – **2915.3** Mxscdx + **9.19** Trnsdy + **4321.9**

Sujeto a

C1: 0.0717 Asimdy + **0.102** Hiredy + **3.222** Mxscdx – **0.0148** Trnsdy + **344.77 < K1**
C2: -0.010 Asimdy – **0.011** Hiredy – **0.39** Mxscdx + **0.0045** Trnsdy + **1.10 < K2**
C3: 0 < Asimdy < **K3**
C4: 0 < Hiredy < **K4**
C5: 0 < Mxscdx < **K5**
C6: 0 < Trnsdy < **K6**

En este ejemplo se han seleccionado las funciones lineales obtenidas para la base de datos CrCrTi, donde FO es la Función Objetivo (Coste) a minimizar; C1, C2 y C3 son las restricciones del problema C1: Tiempo < K1, C2: Calidad < K2; las restricciones C3 a C6 determinan las restricciones para los atributos del problema, donde K1, K2, K3, K4, K5 y K6 son constantes que fija el experto en función de lo restrictivo que se quiera plantear el problema. Una vez resuelto este problema, la solución será el conjunto de valores para los atributos (Asimdy, Hiredy, Mxscdx y Trnsdy) que minimizan el coste del proyecto.

La información que podemos obtener resolviendo un problema de programación lineal puede ser de mucha utilidad en el proceso de toma de decisiones, y puede utilizarse de forma complementaria a los resultados obtenidos con la herramienta PreFuRGe, descritos en el capítulo cuatro.

Para la resolución de los distintos problemas de programación lineal que se han planteado para cada una de las bases de datos de proyectos con las que hemos realizado todas las mediciones hasta ahora, se ha utilizado el conocido método Simplex.

Los resultados obtenidos para cada una de las bases de datos, buscando minimizar el coste del proyecto sujeto a restricciones en tiempo y calidad, son los siguientes:

A) Optimización del Coste (Jbszmd) del proyecto

Las constantes utilizadas en cada restricción, en las ecuaciones correspondientes al tiempo y la calidad, han sido seleccionadas en función del rango de valores en el que puede moverse cada variable del proyecto según el tipo de política aplicado, y en base a criterios basados en la experiencia, dados por el experto.

A.1) BD: *Gestión de personal rápida, con restricción en tiempo de entrega (CrCrTi)*

Minimizar **123.7** Asimdy – **23.4** Hiredy – **2915.3** Mxscdx + **9.19** Trnsdy + **4321.9**

Sujeto a

0.0717 Asimdy + **0.102** Hiredy + **3.222** Mxscdx – **0.0148** Trnsdy + **344.77** $\leq$ **352**
-0.010 Asimdy – **0.011** Hiredy – **0.39** Mxscdx + **0.0045** Trnsdy + **1.10** $\leq$ **0.45**
Asimdy, Hiredy, Mxscdx, Trnsdy $\geq$ **0**
Asimdy ≤ 15, Hiredy ≤ 10, Mxscdx ≤ 1.2, Trnsdy ≤ 10

Figura 6.1: Problema de PL definido en BD CrCrTi

Los rangos de valores en los que se mueven los atributos en esta base de datos son los siguientes:

Asimdy $\in$[5,15], Hiredy $\in$[5,10], Mxscdx $\in$[1.0,1.2], Trnsdy $\in$[5,10]

Los valores que deben tomar los atributos para alcanzar el mínimo en la función objetivo marcada (coste) son:

Min. Coste	**Asimdy**	**Hiredy**	**Mxscdx**	**Trnsdy**
	0	10 (Muy Alto)	1.2 (Muy Alto)	0

Tabla 6.16: Resultados con Simplex en BD CrCrTi (Min Coste)

En esta tabla (6.16) se muestran los resultados obtenidos al aplicar el método Simplex al problema de descrito en la figura 6.1. Se puede apreciar como el valor mínimo para el coste (función objetivo) se obtiene para los valores máximos permitidos para Hiredy y Mxscdx.

Por otro lado, los valores que deben tomar los atributos para obtener el máximo coste, son los mostrados en la tabla 6.17:

Max. Coste	**Asimdy**	**Hiredy**	**Mxscdx**	**Trnsdy**
	15 (Muy Alto)	0	0	10 (Muy Alto)

Tabla 6.17: Resultados con Simplex en BD CrCrTi (Max Coste)

En esta tabla (6.17) se muestran los resultados obtenidos al aplicar el método Simplex al problema de descrito en la figura 6.1, pero considerando la función objetivo la maximización del coste. Se puede apreciar como se alcanza el valor máximo para el coste (función objetivo), para los valores máximos permitidos para Asimdy y Trnsdy.

A.2) BD: *Gestión de personal rápida, Sin restricción en tiempo de entrega (CrSrTi)*

Minimizar **63.54** Asimdy – **5.74** Hiredy + **0.0066** Mxscdx + **0.0623** Trnsdy + **1389**

Sujeto a

0.698 Asimdy + **0.377** Hiredy + **0.00252** Mxscdx – **0.011** Trnsdy + **341** $\leq$ **359**

- 0.0221 Asimdy – **0.0088** Hiredy + **0.00011** Mxscdx + **0.00019** Trnsdy + **0.739** $\leq$ **0.45**

Asimdy, Hiredy, Mxscdx, Trnsdy $\geq$ **0**

Asimdy $\leq$ 15, Hiredy $\leq$ 10, Mxscdx $\leq$ 49, Trnsdy $\leq$ 10

Figura 6.2: Problema de PL definido en BD CrSrTi

Los rangos de valores en los que se mueven los atributos en esta base de datos son los siguientes:

Asimdy $\in$[5,15], Hiredy $\in$[5,10], Mxscdx $\in$[1.2,49], Trnsdy $\in$[5,10]

Los valores que deben tomar los atributos para alcanzar el mínimo en la función objetivo marcada (coste) son:

Min. Coste	**Asimdy**	**Hiredy**	**Mxscdx**	**Trnsdy**
	0	10 (Muy Alto)	0	0

Tabla 6.18: Resultados con Simplex en BD CrSrTi (Min Coste)

En esta tabla (6.18) se muestran los resultados obtenidos al aplicar el método Simplex al problema de descrito en la figura 6.2. Se puede apreciar como el valor mínimo para el coste (función objetivo) se obtiene únicamente considerando los valores máximos permitidos para Hiredy.

A.3) BD: *Sin restricción en la gestión de personal ni el tiempo de entrega (SrCoTi)*.

Minimizar **-6.7** Asimdy – **13.2** Hiredy + **0.117** Mxscdx + **5.39** Trnsdy + **2965.7**

Sujeto a

0.465 Asimdy + **0.417** Hiredy + **0.0902** Mxscdx + **0.662** Trnsdy + **370** $\leq$ **384**

0.000054 Asimdy + **0.0011** Hiredy + **0.000043** Mxscdx – **0.00016** Trnsdy + **0.40** $\leq$ **0.45**

Asimdy, Hiredy, Trnsdy $\geq$ **5**

Mxscdx $\geq$ **1**

Asimdy $\leq$ 120, Hiredy $\leq$ 40, Mxscdx $\leq$ 50, Trnsdy $\leq$ 15

Figura 6.2: Problema de PL definido en BD SrCoTi

Los rangos de valores en los que se mueven los atributos en esta base de datos son los siguientes:

Asimdy $\in$[5,120], Hiredy $\in$[5,40], Mxscdx $\in$[1,50], Trnsdy $\in$[5,15]

Los valores que deben tomar los atributos para alcanzar el mínimo en la función objetivo marcada (coste) son:

Min. Coste	**Asimdy**	**Hiredy**	**Mxscdx**	**Trnsdy**
	120 (Muy Alto)	40 (Muy Alto)	0	0

Tabla 6.19: Resultados con Simplex en BD SrCoTi (Min Coste)

En esta tabla (6.19) se muestran los resultados obtenidos al aplicar el método Simplex al problema de descrito en la figura 6.3. Se puede apreciar como el valor mínimo para el coste (función objetivo) se obtiene para los valores máximos permitidos para Asimdy e Hiredy.

6.2.4.4.1 Conclusiones: Programación Lineal

Podemos observar que los resultados obtenidos no coinciden a nivel cualitativo con los obtenidos con las herramientas C4.5 o COGITO, ni tampoco con los obtenidos con PreFuRGe. Este hecho se debe fundamentalmente a que las funciones lineales utilizadas para definir los problemas de Programación Lineal (PL), se han obtenido aplicando técnicas de regresión lineal multivariante sobre cada una de las bases de datos de PDS. Es conocido, que estas funciones lineales pueden utilizarse para realizar estimaciones, eso sí, con un error asociado, como se ha demostrado anteriormente. En esta investigación se ha calculado una función lineal para cada una de las variables, y se ha estimado cada base de datos con estas funciones, obteniendo unos buenos resultados.

Ahora bien, al integrar las funciones lineales asociadas a cada variable, en un problema de programación lineal, e intentar calcular el valor de los atributos que minimizan la función asociada al coste teniendo como restricciones funciones asociadas al tiempo y calidad, se pierde mucha de la información asociada a los valores almacenados en la base de datos, con lo que sólo se trabaja con funciones, que ya por separado estiman cada variable con un cierto error asociado.

Al aplicar el método Simplex de resolución de problemas lineales, nos interesará minimizar entre otras cosas la función del coste, dando valores a los atributos de una forma algorítmica que nada tiene que ver con lo que en la realidad, en gestión de proyectos, se haría, por lo que se obtienen resultados que en efecto minimizan el coste y tienen en cuenta las restricciones de tiempo y calidad fijadas, pero que desde el punto de vista del experto no tienen demasiada lógica en el ámbito de la gestión de PDS, no siendo aplicables en la toma de decisiones.

La modificación de la restricción que marca el valor límite para cada atributo, no altera la forma del resultado; en este sentido, se ha probado bajar los límites superiores de los parámetros que afectan a la solución final, a valores inferiores de los iniciales, comprobando que el óptimo se volvía a alcanzar en los valores máximos que pudieran tomar.

Además se puede comprobar como, por ejemplo, en la base de datos CrCrTi (tabla 6.17), al calcular qué valor deben tomar los atributos para obtener un coste mínimo, con restricciones de tiempo y calidad bajos también, Simplex obtiene como solución que Hiredy y Mxscdx deben ser muy altos, manteniendo los atributos Asimdy y Trnsdy a cero. Este resultado contradice la política de gestión de personal y de tiempo de entrega aplicada en el proyecto; además, el método Simplex devuelve un resultado sobre una base de datos donde no hay ninguna combinación de atributos que proporcionen coste, tiempo y calidad buenos de manera simultánea. Es conocido en Ingeniería del Software, que para reducir coste y tiempo de entrega, la dedicación de los técnicos (Asimdy) debe ser muy alta, dato que además de no ser corroborado con el Simplex, se contradice.

Por otro lado se probó resolver un problema lineal del que, a priori, se tenía información almacenada en la base de datos (CrCrTi), con la que se obtuvieron los modelos lineales utilizados en problema de la figura 6.1, pero considerando como función objetivo la maximización del coste; los resultados de este PL se muestran en la tabla 6.17; se puede observar que la solución obtenida ya no es tan discrepante como los obtenidos en la tabla 6.16, pero sigue estando sesgada y centrada sólo en algunos atributos del proyecto, por lo que sigue comprobándose la poca validez del método para la toma de decisiones.

Un análisis en profundidad de los problemas lineales planteados en cada caso y de las soluciones obtenidas, desvelan las razones de la poca validez de la información proporcionada por Simplex en los problemas que se han planteado:

- En todos los problemas lineales planteados, las restricciones impuestas (tiempo y calidad) no añaden información nueva a la Función Objetivo (FO), por tanto la minimización del coste es trivial y sólo atiende a la forma de la propia FO.

- La explicación analítica de este hecho es que las restricciones no interseccionan con la región factible definida por la función objetivo, por lo que no intervienen en su optimización. Esto justifica, que para minimizar el coste sólo se atienda a los parámetros con menores coeficientes, dándole a estos los valores máximos permitidos, como se puede observar en las tablas de resultados.

6.2.5 Comparar e iterar estimaciones

A la vista de los resultados obtenidos, se puede decir que la mejor técnica de estimación de las presentadas es la de los k-vecinos, ponderados con pesos (wk-NN), con la que se obtienen los mejores resultados en la estimación de variables, con las tres bases de datos de PDS utilizadas. En este sentido, si al realizar una simulación de un PDS, y obtener por tanto la base de datos asociada, llega un nuevo conjunto de datos (atributos) y se deseara conocer cual sería el valor de las variables (coste, tiempo y calidad) que le correspondería, sin necesidad de volver a simular todo el proyecto, el gestor podría aplicar cualquiera de las tres técnicas propuestas, en especial la de los wk-vecinos, obteniendo así una información bastante aproximada.

A continuación se muestra una comparativa de los errores de estimación obtenidos para cada variable y con cada una de las técnicas propuestas. Analizando los resultados mostrados en la tabla 6.20 podemos concluir que, considerando el orden de técnicas mostrado, el error cometido en las distintas estimaciones va decreciendo de forma que con wk-NN el error relativo medio obtenido se reduce a la décima parte del obtenido con RLM al estimar el tiempo, a una cuarta parte al estimar el coste, mientras que con al estimar la calidad sólo se obtiene una reducción del error del 30%.

CrCrTi		Error A	Error B	Error C
RLM	Coste	88.58	4.00	1.08
	Tiempo	0.30	0.08	1.30
	Calidad	0.04	9.63	1.49
RNLM Polinómica	Coste	87.23	3.94	0.93
	Tiempo	0.30	0.09	1.31
	Calidad	0.05	9.93	1.50
RNLM Exponencial	Coste	69.60	3.18	0.70
	Tiempo	0.30	0.09	1.38
	Calidad	0.10	9.72	1.69
wk-NN	Coste	31.81	1.36	0.37
	Tiempo	0.039	0.011	0.115
	Calidad	0.029	6.515	1.446

Tabla 6.20: Resultados globales obtenidos en BD CrCrTi

Por otro lado, con los resultados mostrados en la tabla 6.21, podemos concluir que considerando el orden de técnicas mostrado, el error cometido en las distintas estimaciones, no es del todo decreciente como en el caso CrCrTi, existiendo pequeñas diferencias entre las técnicas RNLM Exponencial y RNLM Polinómica.

CrSrTi		Error A	Error B	Error C
RLM	Coste	24.76	1.24	0.36
	Tiempo	1.49	0.42	1.89
	Calidad	0.03	8.31	1.29
RNLM Polinómica	Coste	13.02	0.66	0.12
	Tiempo	1.41	0.40	1.98
	Calidad	0.04	9.08	2.19
RNLM Exponencial	Coste	19.36	0.97	0.26
	Tiempo	1.51	0.43	2.20
	Calidad	0.03	8.58	2.25
wk-NN	Coste	11.30	0.558	0.143
	Tiempo	0.169	0.048	0.291
	Calidad	0.023	6.038	1.147

Tabla 6.21: Resultados globales obtenidos en BD CrSrTi

En este caso, al igual que en el anterior, con la técnica wk-NN se obtienen los mejores resultados, de forma que el error relativo medio obtenido al estimar las distintas variables, se reduce a la décima parte del obtenido con RLM al estimar el tiempo, a algo menos de la mitad al estimar el coste, mientras que con al estimar la calidad sólo se obtiene una reducción del error del 25%.

Con los resultados mostrados en la tabla 6.22, podemos concluir que, considerando el orden de técnicas mostrado, el error cometido en las distintas estimaciones, no es exactamente decreciente como en el caso CrCrTi, existiendo pequeñas diferencias en los errores de estimación entre las técnicas RNLM y RLM, y entre las técnicas RNLM Exponencial y RNLM Polinómica. Al igual que en los casos anteriores, la técnica que proporciona los mejores resultados vuelve a ser wk-NN; el error relativo medio obtenido con esta técnica al estimar las distintas variables, se reduce al 20% del obtenido con RLM al estimar el tiempo, al estimar el coste se obtiene una reducción del 30%, y una reducción del 50% al estimar la calidad.

SrCoTi		Error A	Error B	Error C
RLM	Coste	138.8	6.19	1.29
	Tiempo	12.45	3.05	3.47
	Calidad	0.02	4.62	2.79
RNLM Polinómica	Coste	223.95	10.09	1.82
	Tiempo	9.22	2.24	3.02
	Calidad	0.02	4.84	15.67
RNLM Exponencial	Coste	153.19	6.88	1.44
	Tiempo	12.63	3.10	3.46
	Calidad	0.02	4.61	25.57
wk-NN	Coste	51.91	2.33	0.782
	Tiempo	2.686	0.68	0.50
	Calidad	0.009	2.21	1.178

Tabla 6.22: Resultados globales obtenidos en BD SrCoTi

Se puede observar que el tiempo es una variable que se logra estimar con gran precisión con todas las técnicas aplicadas, sea cual sea la medida de error utilizada, mientras que la calidad es la peor estimada. Llama la atención los valores de error obtenidos en la estimación del *Coste* al medirlo con el error A. Esto se debe fundamentalmente a que el error absoluto, es una medida muy sensible al rango de los valores que manipula.

Dado que estamos trabajando con un proyecto ya finalizado, del que se conocen los valores iniciales de los atributos así como los resultados obtenidos en las distintas variables (tablas 5.1 y 5.2), se realizó una estimación de las variables del proyecto partiendo de los valores iniciales de los atributos (tabla 5.1) con cada una de las técnicas de estimación propuestas, y probando con las ecuaciones obtenidas para cada una de las políticas aplicadas (CrCrTi, CrSrTi, SrCoTi) para verificar también cuál fue la que se aplicó, obteniendo los mejores resultados con las ecuaciones del modelo CrSrTi, lo que implica que el atributo relacionado con el tiempo de entrega (Mxscdx) debe tomar valores dentro del intervalo [1.21,50], es decir, sin plazo estricto de entrega.

Post-Mortem		Error A	Error B	Error C
RLM	Coste	396.24	18.94	3.52
	Tiempo	20.83	5.38	0.58
	Calidad	0.22	86.45	1.14
RNLM Polinómica	Coste	703.22	33.61	6.25
	Tiempo	24.67	6.37	0.68
	Calidad	0.144	55.45	0.73
RNLM Exponencial	Coste	325.22	15.55	2.89
	Tiempo	22.06	5.70	0.61
	Calidad	0.06	25.14	0.33
wk-NN	Coste	239.3	11.43	2.12
	Tiempo	26.53	6.85	0.73
	Calidad	0.12	47.69	0.63

Tabla 6.23: Resultados globales obtenidos con datos Post-Mortem

En esta tabla (6.23) se puede observar que analizando el error B (error relativo medio), las técnicas que proporcionan mejores resultados de estimación de las variables son las de Regresión No Lineal Multivariante (RNLM), utilizando formas de función exponencial y la técnica de los k-vecinos ponderados con pesos (wk-NN). En este caso, con la técnica RNLM exponencial se obtienen mejores resultados en la estimación de la calidad que con la técnica wk-NN.

Capítulo 7

Conclusiones y Futuras líneas de investigación

7.1 Conclusiones finales

Las conclusiones a las que podemos llegar, a la vista de los resultados obtenidos en los diferentes capítulos, las podemos dividir en dos grupos, las referidas a PreFuRGe y las referentes a las técnicas de predicción propuestas:

A) *Herramienta PreFuRGe:*

- PreFuRGe, herramienta basada en técnicas de lógica borrosa, presenta de entrada una gran ventaja respecto a las herramientas con que ha sido comparada, C4.5 y COGITO: *No es necesario etiquetar las salidas*, evitándose así el preprocesamiento, a veces manual y por tanto expuesto a errores, necesario en herramientas basadas en aprendizaje supervisado. Lo anterior implica además que se pueden conocer reglas para diferentes combinaciones de los resultados finales (tiempo bueno y coste medio, tiempo medio y coste muy bueno, etc.).

- *Los resultados no tienen que ser interpretados por el experto*, ya que tanto la salida en formato gráfico como su interpretación en lenguaje natural, poseen un nivel cualitativo muy cercano a la forma de razonar humana. Estas reglas pueden ser entendidas incluso por personal no experto.

- *Se obtienen reglas de gestión para todas las combinaciones de valores* de las variables de salida que existan en la base de datos, con lo que el gestor tiene un conocimiento mucho más completo del comportamiento del proyecto.

- A la hora de generar las reglas de gestión, *se pueden ponderar las variables de salida con pesos*, de forma que la herramienta trabajará con un conocimiento mucho más exacto del proyecto y los resultados serán más acertados.

- *Trabajar con conjuntos borrosos en vez de intervalos* como, hace que, sin perder valor cualitativo, la interpretación y posterior aplicación de las reglas sea mucho más natural y eficaz.

- En el capítulo 5 se pudo comprobar que COGITO proporciona reglas más concretas que las proporcionadas por C4.5. A su vez, PreFuRGe proporciona unas reglas más cualitativas e interpretables por el gestor del proyecto, que ambas herramientas.

- En general el número de atributos en cada regla generada por C4.5 y COGITO es similar por lo que desde ese punto de vista no existe diferencia entre ellas. Por el contrario, PreFuRGe incluye normalmente todos los atributos del proyecto, salvo que estos no sean significativos para decidir el valor de las variables en estudio.

- Las reglas obtenidas con las tres herramientas utilizadas recogen la teoría conocida y validada con otros trabajos sobre el comportamiento de proyectos de desarrollo de software.

B) *Técnicas de predicción:*

- Los resultados de predicción de las tablas 6.20, 6.21 y 6.22, muestran que en las tres bases de datos (CrCrTi, CrSrTi, SrCoTi) en las que se aplicaron las distintas técnicas de predicción (RLM, RNLM Pol, RNLM Exp, wk-NN), la técnica que aproximó de manera más exacta las variables del PDS fue la

técnica de los vecinos más cercanos ponderados, wk-NN, con bastante diferencia respecto a las otras.

- En la Tabla 6.20 (base de datos CrCrTi), podemos observar que considerando el orden de técnicas mostrado, el error cometido en las distintas estimaciones va decreciendo de forma que con wk-NN el error relativo medio obtenido se reduce a la décima parte del obtenido con RLM al estimar el tiempo, a una cuarta parte al estimar el coste, mientras que con al estimar la calidad sólo se obtiene una reducción del error del 30%.

- En la Tabla 6.21 (base de datos CrSrTi), podemos observar que al igual que en el anterior, con la técnica wk-NN se obtienen los mejores resultados, de forma que el error relativo medio obtenido al estimar las distintas variables, se reduce a la décima parte del obtenido con RLM al estimar el tiempo, a algo menos de la mitad al estimar el coste, mientras que con al estimar la calidad sólo se obtiene una reducción del error del 25%.

- En la Tabla 6.22 (base de datos SrCoTi), podemos observar que al igual que en los casos anteriores, la técnica que proporciona los mejores resultados vuelve a ser wk-NN. En este caso, el error relativo medio obtenido con esta técnica al estimar las distintas variables, se reduce al 20% del obtenido con RLM al estimar el tiempo, al estimar el coste se obtiene una reducción del 30%, y una reducción del 50% al estimar la calidad.

- Se puede observar que el tiempo es una variable que se logra estimar con una gran precisión con todas las técnicas aplicadas, sea cual sea la medida de error utilizada, mientras que la calidad es la variable que se estima con peor precisión.

7.2 Futuras Líneas de Investigación

Una vez analizados los resultados más relevantes obtenidos en el desarrollo de esta investigación, se describen una serie de propuestas que pueden servir como punto de partida para futuras líneas de investigación.

- *Monitorización y estudio del transitorio durante el desarrollo del proyecto*: Con esto se pretende controlar la evolución del proyecto, en distintos momentos de su desarrollo, de forma que se vayan corrigiendo, en base a

reglas de gestión, las posibles desviaciones de las variables que se desean estudiar.

- *Optimización multiobjetivo de las variables de un PDS*: Utilizando diversas técnicas de optimización multiobjetivo y multicriterio se desean obtener los valores adecuados en los atributos, para llegar a tener buenos resultados en las diversas variables de salida, a la vez.

- *Dotar a la herramienta de una mayor interacción con el usuario*:

    - Selección automática de las mejores reglas de gestión obtenidas, en base a criterios marcados por el gestor del proyecto

    - De las reglas gráficas, eliminar las reglas que no aporten nada positivo al gestor, y aquellos parámetros que no intervengan de forma relevante en las reglas

    - Adaptar el código de la herramienta a un entorno visual, con posibilidad de interactuar sobre los distintos parámetros de los algoritmos utilizados, de forma que el gestor del proyecto pueda obtener un conjunto de reglas con una serie de características.

- *Selección de atributos (Feature Selection)* según criterios fijados por el gestor, de forma que se pueda reducir el número de parámetros a tratar, sin perder información cualitativa en los resultados.

- *Aplicación de PreFuRGe a conjuntos de bases de datos reales*, con el fin de comprobar su funcionamiento con conjuntos de datos reales, con un elevado número de atributos y de registros.

Apéndice A

Lógica Borrosa (Fuzzy Logic)

1. Introducción

La mayor parte del razonamiento humano está más cerca de lo aproximado que de lo preciso. De un modo bastante eficiente, los humanos somos capaces de tomar decisiones racionales con información imprecisa o incompleta, reconocer voces e imágenes distorsionadas, resumir y completar datos parcialmente desconocidos, etc. Desde esta perspectiva, la Lógica Borrosa puede verse como un intento de construir un modelo del razonamiento humano que refleje su carácter aproximado, cualitativo. En este modelo, el razonamiento preciso debe verse como un caso límite que estará incluido en el anterior. La gran utilidad de la lógica borrosa, está en la posibilidad de tratar problemas demasiado complejos o muy mal definidos como para admitir tratamiento por métodos tradicionales. Por este motivo, los avances y aportaciones en representación del conocimiento impreciso, semántica del lenguaje natural e inferencia a partir de datos imprecisos aparecen mezclados entre sí.

El desarrollo de la lógica borrosa ha sido inspirado y guiado por Zadeh [Zadeh65], profesor de Ingeniería Electrónica en el Departamento del mismo nombre de la Universidad de California en Berkeley, quien ahora es considerado como '*alma mater'* indiscutido e indiscutible de la misma.

La lógica borrosa (Fuzzy Logic) ha surgido como una herramienta para el control de subsistemas y procesos industriales complejos, así como para la electrónica de entretenimiento y hogar, sistemas de diagnóstico y otros sistemas expertos. Aunque la lógica borrosa se inventó en Estados Unidos, el crecimiento rápido de esta tecnología ha comenzado desde Japón y ahora nuevamente ha alcanzado USA y también Europa. En Japón y china todavía es un fenómeno a nivel científico, donde el número de patentes de aplicaciones aumenta exponencialmente.

El término *borroso* ha llegado a ser clave para vender. Los artículos electrónicos sin componentes borrosos se están quedando gradualmente desfasados. Muestra de ello, es el hecho de que cada vez es más frecuente un sello con "fuzzy logic" impreso sobre tales productos.

En Japón, la investigación sobre lógica borrosa es apoyada ampliamente con un presupuesto enorme. En Europa y USA se están realizando esfuerzos para alcanzar al tremendo éxito japonés. Por ejemplo, la NASA emplea lógica borrosa para el complejo proceso de maniobras de acoplamiento.

En resumen, la lógica borrosa es básicamente una lógica multievaluada que permite valores intermedios para poder definir evaluaciones convencionales como sí/no, verdadero/falso, negro/blanco, etc. Las nociones como "más bien caliente" o "poco frío" pueden formularse matemáticamente y ser procesadas por computadoras. De esta forma se ha realizado un intento de aplicar una forma más humana de pensar en la programación de computadoras.

2. Los conjuntos borrosos

La noción de Conjunto Borroso (Fuzzy Set) aparece por primera vez en 1964 en un memorando de la Universidad de California en Berkeley, y es debida como ya se ha comentado a Zadeh. Este memorando sería publicado un año más tarde en “Information and Control” [Zadeh65], marcando el nacimiento de una nueva teoría (o tecnología). Algunas de las principales publicaciones, donde aparecen diferentes enfoques sobre la Teoría de Conjuntos Borrosos son [Kaufm75], [Duboi80], [Trill80], [Zimme85], [Klir88].

La idea más básica de sistema borroso es un (sub)conjunto borroso. Veamos un ejemplo para explicar este concepto. En primer lugar consideremos un conjunto X con todos los números reales entre 0 y 10 que llamaremos *universo de discurso*. Se define un subconjunto A de X ($A \subseteq X$) con todos números reales en el rango entre 5 y 8.

$$A = [5,8]$$

La *función característica* o *función de pertenencia* del Conjunto A (μ_A), asigna un número 1 ó 0 a los elementos de X, dependiendo de si el elemento está o no en el subconjunto A. Representando de forma gráfica esta función en el eje de ordenado, y los valores del Conjunto X en abcisa, se obtiene:

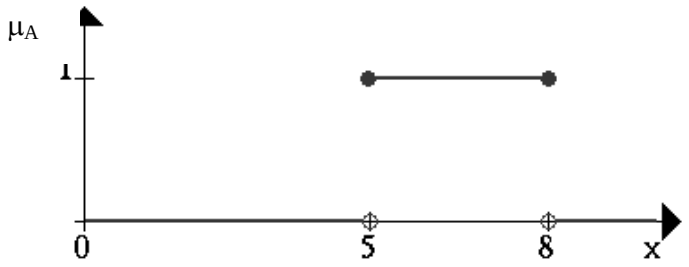


Figura A.1: Representación gráfica de función de pertenencia

De esta representación se puede interpretar que los elementos que tienen pertenencia 1 ($\mu_A = 1$) son los elementos que están en el conjunto A y los elementos que tienen pertenencia 0 ($\mu_A = 0$) son los elementos que no están en el conjunto A

Este concepto es suficiente para muchas áreas de aplicación, pero se pueden encontrar fácilmente situaciones donde carece de flexibilidad. Para comprender este concepto veamos un ejemplo: “Se quiere describir el conjunto de gente joven”. Más formalmente se puede denotar por B = {conjunto de gente joven}.

Como la edad comienza en 0, el rango inferior de este conjunto está claro. El rango superior, por otra parte, es más bien complicado de definir. Como un primer intento se puede colocar el rango superior en 20 años. Por lo tanto se define B como un intervalo denominado *B = [0,20]*

Ahora la pregunta es: “¿por qué alguien es en su 20 cumpleaños joven y al día siguiente no?”. Obviamente, este es un problema estructural, porque si se mueve el límite superior del rango desde 20 a un punto arbitrario podemos plantear la misma pregunta. Una manera más natural de construir el conjunto B estaría en suavizar la separación estricta entre joven y no joven. Esto se hace para permitir no solamente la decisión de "SI está en el conjunto de gente joven" o "NO está en el conjunto de gente joven", sino también las frases más flexibles como "SI pertenece un poquito más al conjunto de gente joven" o "NO pertenece aproximadamente al conjunto de gente joven".

A continuación se muestra como un conjunto borroso permite definir una noción como "es un poco joven". Tal y como se ha constatado en la introducción se pueden usar conjuntos borrosos para hacer computadoras más sabias; es necesario codificar esta idea de manera más formal. En el ejemplo primero se codificaron todos los elementos

del Universo de Discurso con 0 ó 1. Una manera de generalizar este concepto está en permitir más valores entre 0 y 1. De hecho, se permiten infinitas alternativas entre 0 y 1, denominando el intervalo de unidad Yo = [0, 1].

La interpretación de los números ahora asignados a todos los elementos del Universo de Discurso es algo más difícil. Por supuesto, un elemento con un valor de pertenencia 1 significa que el elemento está en el conjunto B y con valor 0 significa que el elemento no está definitivamente en el conjunto el B. El resto de valores (entre 0 y 1) significan una pertenencia gradual al conjunto B.

Para ser más concretos, se muestra gráficamente el conjunto de gente joven de forma similar al primer ejemplo por su función característica o de pertenencia (μ_B).

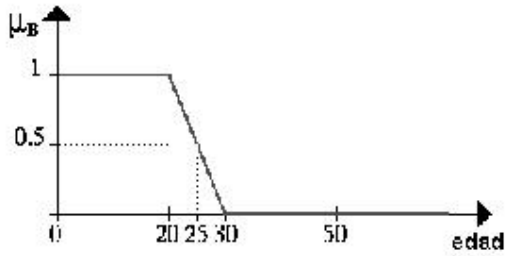


Figura A.2: Ejemplo de función de pertenencia

De esta forma una persona de 25 años de edad todavía sería joven al grado de 50 por ciento (μ_B= 0.5). Hemos definido qué es un conjunto borroso. A continuación vamos a describir qué operaciones se pueden hacer con él.

Una vez definido qué es un conjunto borroso, vamos a describir qué operaciones se pueden hacer con él.

3. Operaciones con conjuntos borrosos

Con una idea de lo que son los conjuntos borrosos, se pueden introducir las operaciones básicas que se pueden realizar sobre éstos. Parecido a las operaciones sobre conjuntos booleanos, también se puede Interseccionar, Unir y Negar conjuntos borrosos. En su primer artículo sobre conjuntos borrosos, Zadeh sugirió el operador *mínimo* para la intersección y el operador *máximo* para la unión de dos conjuntos borrosos.

Es fácil ver que estos operadores coinciden con la unión booleana, e intersección si sólo se consideran los grados miembros 0 y 1.

A fin de aclarar esto, se muestran varios ejemplos. Sea A un intervalo borroso entre 5 y 8, y B un número borroso entorno a 4.

Las representaciones gráficas correspondientes se muestran a continuación:

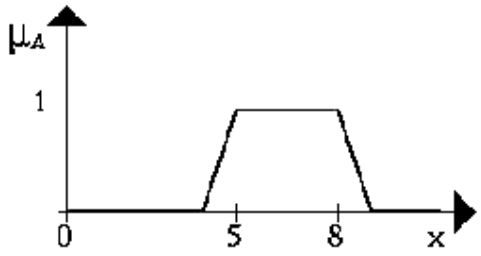


Figura A.3: Ejemplo de conjunto borroso A

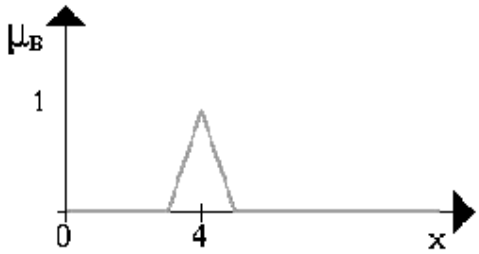


Figura A.4: Ejemplo de conjunto borroso B

La figura siguiente muestra la operación **AND** (Y) del conjunto borroso A y el número borroso B (el resultado es la línea oscura más gruesa).

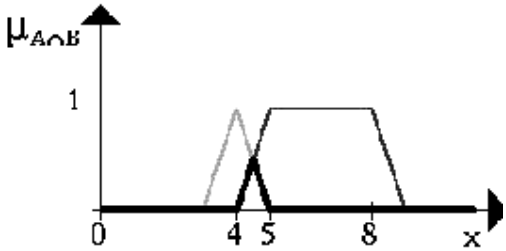


Figura A.5: Operación AND entre los conjuntos A y B

La operación **OR** (O) del conjunto borroso A con el número borroso B se muestra en la próxima figura (nuevamente, es la línea oscura más gruesa).

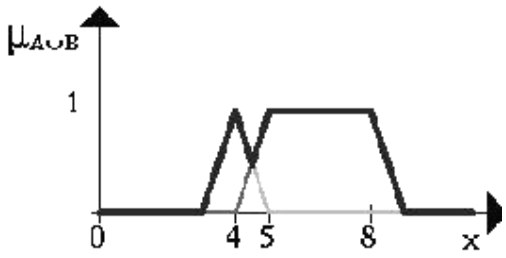


Figura A.6: Operación OR entre los conjuntos A y B

Esta figura da un ejemplo para una negación. La línea más oscura es la **NEGACION** del conjunto borroso A.

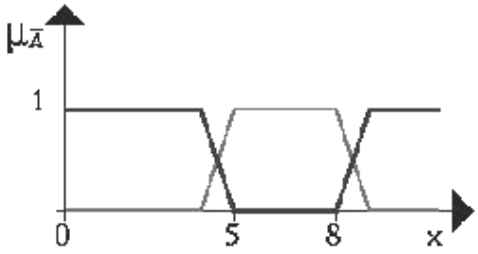


Figura A.7: Operación NEGACIÓN del conjunto borroso A

4. Aplicaciones de la Lógica Borrosa

En los últimos años la Lógica Borrosa, como herramienta del razonamiento aproximado, ha encontrado numerosas aplicaciones que van desde el campo de las finanzas hasta la ingeniería de terremotos. Todas ellas corresponden a problemas muy complejos o mal conocidos que por ello no admiten tratamiento por métodos basados en la Lógica Clásica. De hecho la introducción de la Lógica Borrosa ha permitido modelizar y resolver situaciones tradicionalmente consideradas como intratables, y en este sentido ha constituido una revolución en muchos campos.

Como ya se ha comentado con anterioridad, puede considerarse que la lógica borrosa surgió como una respuesta a la necesidad de disponer de modelos de inferencia para tratar el conocimiento impreciso, básicamente en el contexto de la construcción de sistemas expertos. Sin embargo, actualmente su mayor, más conocida y notable

aplicación es en el control de procesos, un campo muy alejado de aquél que la impulsó en sus primeras etapas.

La aplicación de la Lógica Borrosa y la teoría de la posibilidad al campo de los sistemas basados en reglas se ha desarrollado principalmente en dos direcciones:

a) De una parte se ha tratado de generalizar el modelo de los factores de certeza (empleado por primera vez en MYCIN) para dar cabida a la imprecisión no probabilística. En esta línea se han estudiado modelos de representación y propagación de la incertidumbre; ejemplos típicos de sistemas construidos con estas herramientas son el sistema RUM, desarrollado por Bonissone [Bonis87], o el sistema de inferencia MILORD, desarrollado por el grupo de investigación en inteligencia artificial del Centro de Estudios Avanzados de Blanes [Godo88].

b) Por otro lado se ha estudiado el manejo de predicados imprecisos en la descripción de las reglas o en la información disponible acerca de los hechos. En esta línea, los trabajos realizados han ido básicamente encaminados a encontrar métodos de inferencia más potentes, interés que ha sido reforzado por las necesidades de la aplicación al Control de sistemas que se comentará en el siguiente punto. Concretamente se ha investigado la regla Composicional de inferencia y el empleo de distintas funciones de implicación [Trill84], [Trill85], [Mizum82], [Mizum85], [Kiska85], [Cao90].

4.1. El control Borroso

Los fundamentos del control por medio de la Lógica Borrosa pueden encontrarse en [Zadeh72], [Zadeh73] y [Tong85]. La idea de base es que las relaciones de entrada-salida (reglas de control) de un sistema complejo suelen ser descritas por los expertos en forma de sentencias condicionales *Si-Entonces* con antecedente(s) y consecuente(s) dado en forma de proposiciones borrosas. A partir de esta descripción y de la observación del comportamiento del sistema, se puede deducir la *acción de control* empleando las leyes de inferencia de la lógica borrosa.

Los pioneros en la implementación práctica de estas ideas fueron Mandani y Assilian, junto con su equipo en el Queen Mary College (London). En sus primeros trabajos [Manda74] y [Manda75], desarrollan la regulación de un motor de vapor, aplicación a la que seguirán otras muchas, que en su mayoría fueron simples ejercicios de laboratorio, a excepción de un controlador para plantas de fabricación de cemento que llegó a estar comercialmente disponible [Holmb82]. Aún así, la aparición de estos primeros desarrollos despertó un interés inesperado en la comunidad científica, ya que

mostraban la posibilidad de controlar sistemas clásicamente considerados como intratables por su complejidad o mal conocimiento. Después de esta primera época, comienza un periodo en el que la mayor parte de las investigaciones estuvieron dedicadas a estudiar y formalizar el concepto de "*controlador borroso*". R.M. Tong [Tong85] analiza detalladamente la evolución e historia del control borroso con un marcado carácter teórico.

Desde 1985 se observa un cambio sustancial en la orientación de las investigaciones que se dirigen básicamente a la consecución de productos exclusivamente comerciales. Los responsables de este cambio son los investigadores japoneses, quienes desde mediados de los ochenta están a la vanguardia de estos desarrollos. Aplicaciones dignas de mención en este sentido son las de control automático del funcionamiento de un tren (realizado por Hitachi), control de un coche y un helicóptero mediante órdenes verbales dadas en lenguaje natural (realizado en el "Laboratory forinternational fuzzy engineering research" bajo la dirección de M. Sugeno), el control de estabilización de un sistema (desarrollado por T. Yamakawa), etc. Gran número de referencias sobre aplicaciones de control desarrolladas en Japón pueden encontrarse en [Prepr87], [Proce90], [Proce91].

En la mayor parte de las aplicaciones actuales de la lógica borrosa, se emplea el software para la implementación de los algoritmos borrosos y las reglas de control. No obstante y a partir de los trabajos pioneros de Yamakawa [Yamak86], y Togai [Togai86], se inicia el camino de las implementaciones hardware.

Los controladores borrosos son las aplicaciones más importantes de la teoría borrosa. Ellos trabajan de una forma bastante diferente a los controladores convencionales; el conocimiento experto se usa en vez de ecuaciones diferenciales para describir un sistema. Este conocimiento puede expresarse de manera muy natural, empleando las Variables Lingüísticas que son descritas mediante conjuntos borrosos.

Uno de los problemas más característicos donde se utiliza el control borroso para su resolución es el *péndulo invertido*, que consiste en "equilibrar una pértiga sobre una plataforma móvil que puede moverse en dos únicas direcciones, a la izquierda o a la derecha".

El empleo del control borroso es recomendable:

- Para procesos muy complejos, cuando no hay un modelo matemático simple.
- Para procesos altamente no lineales.
- Si el procesamiento del (lingüísticamente formulado) conocimiento experto puede ser desempeñado.

Por el contrario, el empleo del control borroso <u>no</u> será una buena elección si:

- El control convencional teóricamente ofrece un resultado satisfactorio.

- Existe un modelo matemático fácilmente soluble y adecuado.
- El problema no es resoluble.

Una interesante colección de artículos que cubren diferentes aplicaciones de la lógica borrosa pueden encontrarse en [Yager92].

5. Definiciones

- **Intersección de Conjuntos**: Es un nuevo conjunto generado desde dos conjuntos determinados A y B que contiene exactamente los elementos que están contenidos en A y en B.

- **Unión de Conjuntos**: Es un nuevo conjunto generado desde dos conjuntos determinados A y B, que contiene todos los elementos que están en A o en B o en ambos.

- **Negación de Conjuntos**: Es un nuevo conjunto que contiene todos los elementos que están en el universo de discurso pero no en el conjunto negado.

- **Variables lingüísticas**: Un variable lingüística es un quíntuplo (X, T(X), U, G, M), donde X es el nombre de la variable, T(X) es el término conjunto (es decir, el conjunto de nombres de valores lingüísticos de X), U es el universo de discurso, G es la gramática para generar los nombres y M es un conjunto de reglas semánticas para asociar cada X con su significado.

Apéndice B

Modelo de estimación de costes COCOMO II

1. Introducción

El modelo original COCOMO se público por primera vez en 1981 por Barry Boehm y reflejaba las prácticas en desarrollo de software de aquel momento. En la década y media siguiente las técnicas de desarrollo software cambiaron drásticamente. Estos cambios incluyen el gasto de tanto esfuerzo en diseñar y gestionar el proceso de desarrollo software como en la creación del producto software, un giro total desde los mainframe que trabajan con procesos batch nocturnos hacia los sistemas en tiempo real y un énfasis creciente en la reutilización de software ya existente y en la construcción de nuevos sistemas que utilizan componentes software a medida.

Estos y otros cambios hicieron que la aplicación del modelo COCOMO original empezara a resultar problemática. La solución al problema era reinventar el modelo para aplicarlo en los noventa. Después de muchos años de esfuerzo combinado entre USC-CSE1, IRUS y UC Irvine22 y las Organizaciones Afiliadas al Proyecto COCOMO II, el resultado es COCOMO II, un modelo de estimación de coste que refleja los cambios en la práctica de desarrollo de software profesional que ha surgido a partir de los años 70.

Este nuevo y mejorado COCOMO resultará de gran ayuda para los estimadores profesionales de coste software.

Por tanto, COCOMO II es un modelo que permite estimar el coste, esfuerzo y tiempo cuando se planifica una nueva actividad de desarrollo software. Está asociado a los ciclos de vida modernos. El modelo original COCOMO ha tenido mucho éxito pero no puede emplearse con las prácticas de desarrollo software más recientes tan bien como con las prácticas tradicionales. COCOMO II apunta hacia los proyectos software de los 90 y de la primera década del 2000, y continuará evolucionando durante los próximos años. En resumen, los objetivos a la hora de la creación del modelo COCOMO II fueron:

- Desarrollar un modelo de estimación de tiempo y de coste del software de acuerdo con los ciclos de vida utilizados en los 90 y en la primera década del 2000.
- Desarrollar bases de datos con costes de software y herramientas de soporte para la mejora contínua del modelo.
- Proporcionar un marco analítico cuantitativo y un conjunto de herramientas y técnicas para la evaluación de los efectos de la mejora tecnológica del software en costes y tiempo del ciclo de vida software.

Estos objetivos apoyan las necesidades primarias expresadas por los usuarios de la estimación de costes del software. En orden de prioridades, estas necesidades eran: el apoyo de la planificación de proyectos, la previsión de personal del proyecto, la preparación del proyecto, la replanificación, el seguimiento del proyecto, la negociación del contrato, la evaluación de la propuesta, la nivelación de recursos, exploración de conceptos, la evaluación del diseño y decisiones referentes a la oferta / demanda. Para cada una de estas necesidades COCOMO II proporcionará un apoyo más moderno que sus predecesores, el COCOMO original y Ada COCOMO.

Los cuatro elementos principales de la estrategia que ha seguido COCOMO II son:

- Preservar la apertura del COCOMO original.
- Desarrollar COCOMO II de forma que sea compatible con el futuro mercado del software
- Ajustar las entradas y salidas de los submodelos de COCOMO II al nivel de información disponible.
- Permitir que los submodelos de COCOMO II se ajusten a las estrategias de proceso particulares de cada proyecto.

COCOMO II sigue los principios de apertura usados en el COCOMO original. De esta manera todos sus algoritmos y relaciones están disponibles públicamente. También todos sus interfaces están diseñadas para ser públicas, bien definidas y parametrizadas para que los preprocesos complementarios (modelos de analogía basados en casos de otras medidas de estimación), post-procesos (planificación de proyectos y herramientas de control, modelos dinámicos de proyectos, analizadores de riesgo) y paquetes de alto nivel (paquetes de gestión de proyectos, ayudas de negociación de producto) puedan combinarse estrechamente con COCOMO II.

Para apoyar a los distintos sectores del mercado software, COCOMO II proporciona una familia de modelos de estimación de coste software cada vez más detallado y tiene en cuenta las necesidades de cada sector y el tipo de información disponible para sostener la estimación del coste software. Esta familia de modelos está compuesta por tres submodelos cada uno de los cuales ofrece mayor fidelidad a medida que uno avanza en la planificación del proyecto y en el proceso de diseño. Estos tres submodelos se denominan:

- *El modelo de Composición de Aplicaciones*: Indicado para proyectos construidos con herramientas modernas de construcción de interfaces gráficos para usuario.

- *El modelo de Diseño anticipado*: Este modelo puede utilizarse para obtener estimaciones aproximadas del coste de un proyecto antes de que esté determinada por completo su arquitectura. Utiliza un pequeño conjunto de drivers de coste nuevo y nuevas ecuaciones de estimación. Está basado en Punto de Función sin ajustar o KSLOC (Miles de Líneas de Código Fuente).

- *El modelo Post-Arquitectura*: Este es el modelo COCOMO II más detallado. Se utiliza una vez que se ha desarrollado por completo la arquitectura del proyecto. Tiene nuevos drivers de coste, nuevas reglas para el recuento de líneas y nuevas ecuaciones.

La primera implementación de COCOMO II se presentó al público general a mediados de 1997. USC COCOMOII.1997 se calibró con 83 puntos de datos (proyectos de desarrollo software históricos), utilizando un enfoque de media ponderada del 10% para combinar datos empíricos con la opinión del experto y calibrar los parámetros del modelo. USC COCOMOII.1998.0 beta se creó en Octubre de 1998. Esta versión se calibró con 161 puntos de datos y utilizando por primera vez un enfoque bayesiano para realizar la calibración del modelo. USC COCOMO II.1999.0 se ha ubicado a mediados de 1999 y USC-COCOMO II.2000.0 será presentado en el año 2000. Mientras cada versión nueva de la herramienta USC COCOMO II fue mejorando en cuanto a amigabilidad con el usuario, las calibraciones del modelo de los años 1998, 1999 y 2000

son la misma. Es decir, no se han añadido nuevos puntos de datos a la base de datos utilizada para calibrar las versiones de la herramienta del año 1999 y del año 2000 aparte de aquellos que aparecieron en la calibración de la base de datos del año 1998. En las tres implementaciones aparecen los mismos 161 puntos de datos a los que hacíamos referencia anteriormente En este sentido, cabe decir que la versión USC-COCOMO II.2001.0 ya incluye una nueva calibración. Por último, la experiencia ha demostrado que si una organización calibra la constante multiplicativa en COCOMO II para sus propios datos empíricos, la precisión del modelo puede aumentar significativamente por encima de los resultados de calibración genéricos obtenidos con las versiones mencionadas anteriormente.

2. Evolución de COCOMO 81 a COCOMO II

Es importante recalcar los cambios experimentados por COCOMO II con respecto a COCOMO 81 ya que reflejan cómo ha madurado la tecnología de la Ingeniería del software durante las dos décadas pasadas.

Por ejemplo, cuando se publicó el primer modelo COCOMO 81 los programadores estaban sometidos a tareas batch. El giro total que se experimentó afectó a su productividad. Por lo tanto un parámetro como TURN que reflejaba la espera media de un programador para recibir de vuelta su trabajo ahora no tiene sentido porque la mayoría de los programadores tienen acceso instantáneo a los recursos computacionales a través de su estación de trabajo. Este parámetro ha desaparecido en el nuevo modelo.

Las tablas B.1 y B.2 recalcan los principales cambios hechos a la versión original COCOMO 81 cuando se desarrolló COCOMO II y un resumen de las principales diferencias entre ambos. La tabla B.2 presenta una comparativa por modelos más detallada de COCOMO II y además incluye el modelo AdaCOCOMO. De ambas tablas podemos deducir:

- COCOMO II se dirige a las siguientes tres fases del ciclo de vida en espiral: desarrollo de aplicaciones, diseño anticipado y Post-Arquitectura.
- Los tres modos del exponente se han reemplazado por cinco factores de escala.
- Se han añadido los siguientes drivers de coste a COCOMO II: DOCU, RUSE, PVOL, PEXP, LTEX, PCON y SITE.
- Se han eliminado los siguientes drivers de coste del modelo original: VIRT, TURN, VEXP, LEXP, Y MODP.
- Los valores de aquellos drivers de coste que se han conservado del modelo original fueron modificados considerablemente para reflejar las calibraciones actualizadas. Las otras diferencias principales se refieren a efectos relacionados

con el tamaño, que incluyen reutilización, reingeniería y cambios en efectos de escala.

	COCOMO 81	**COCOMO II**
ESTRUCTURA DEL MODELO	Un modelo único que asume que se ha comenzado con unos requisitos asignados para el software	Tres modelos que asumen que se progresa a lo largo de un desarrollo de tipo espiral para consolidar los requisitos y la arquitectura, y reducir el riesgo.
FORMA MATEMÁTICA DE LA ECUACIÓN DEL ESFUERZO	Esfuerzo = A (cI) (SIZE)Exponenente	Esfuerzo = A (cI) (SIZE)Exponenente
EXPONENTE	Constante fija seleccionada como una función de modo: ☑ Orgánico = 1.05 ☑ Semi-libre = 1.12 ☑ Rígido = 1.20	Variable establecida en función de una medida de cinco factores de escala: ☑ **PREC** Precedencia ☑ **FLEX** Flexibilidad de desarrollo ☑ **RESL** Resolución de Arquitectura / Riesgos ☑ **TEAM** Cohesión del equipo ☑ **PMAT** Madurez del proceso
MEDIDA	Líneas de código fuente (con extensiones para puntos de función)	Puntos objeto, Puntos de función ó líneas de código fuente.
DRIVERS DE COSTE(ci)	15 drivers, cada uno de los cuales debe ser estimado: ☑ **RELY** Fiabilidad ☑ **DATA** Tamaño Base de datos ☑ **CPLX** Complejidad ☑ **TIME** Restricción tiempo de ejecución ☑ **STOR** Restricción de almacenamiento principal ☑ **VIRT** Volatilidad máquina virtual ☑ **TURN** Tiempo de respuesta ☑ **ACAP** Capacidad del analista ☑ **PCAP** Capacidad programador ☑ **AEXP** Experiencia aplicaciones ☑ **VEXP** Experiencia máquina virtual ☑ **LEXP** Experiencia lenguaje ☑ **TOOL** Uso de herramientas software ☑ **MODP** Uso de Técnicas modernas de programación ☑ **SCED** Planificación requerida	17 drivers, cada uno de los cuales debe ser estimado: ☑ **RELY** Fiabilidad ☑ **DATA** Tamaño Base de datos ☑ **CPLX** Complejidad ☑ **RUSE** Reutilización requerida ☑ **DOCU** Documentación ☑ **TIME** Restricción tiempo de ejecución ☑ **STOR** Restricción de almacenamiento principal ☑ **PVOL** Volatilidad plataforma ☑ **ACAP** Capacidad del analista ☑ **PCAP** Capacidad programador ☑ **AEXP** Experiencia aplicaciones ☑ **PEXP** Experiencia plataforma ☑ **LTEX** Experiencia lenguaje y herramienta ☑ **PCON** Continuidad del personal ☑ **TOOL** Uso de herramientas software ☑ **SITE** Desarrollo Multi-lugar ☑ **SCED** Planificación requerida
OTRAS DIFERENCIAS DEL MODELO	Modelo basado en: ☑ Fórmula de reutilización lineal ☑ Asunción de requisitos razonablemente estables	Tiene muchas otras mejoras que incluyen: ☑ Fórmula de reutilización No-lineal ☑ Modelo de reutilización que considera esfuerzo necesario para entender y asimilar ☑ Medidas de rotura que se usan para abordar la volatilidad de requisitos ☑ Características de autocalibración

Tabla B.1. Comparativa entre COCOMO 81 Y COCOMO II

	COCOMO 81	**AdaCOCOMO**	**COCOMO II Modelo Composición de aplicaciones**	**COCOMO II Modelo Diseño Anticipado**	**COCOMO II Modelo PostArquitec-tura**
MEDIDA	Instrucciones fuente entregadas (DSI) ó Líneas de Código fuente (SLOC)	DSI ó SLOC	Puntos Objeto	Puntos de función (FP) y lenguaje ó SLOC	Puntos de función (FP) y lenguaje ó SLOC
REUTILIZACIÓN	SLOC Equivalente= lineal f(DM,CM,IM)	SLOC Equivalente = lineal f(DM,CM,IM)	Implícito en el modelo	%reutilización sin modificar: SR %reutilización modificada: no lineal f(AA,SU,DM,CM,IM)	SLOC equivalente= no lineal f(AA,SU,DM,CM,IM)
ROTURA	Medida de Volatilidad de los requisitos (RVOL)	Medida RVOL	Implícito en el modelo	Rotura%(BRAK)	Rotura%(BRAK)
MANTENIMIENTO	Tráfico anual de cambio (ACT) =%añadido + %modificado	ACT	Modelo de reutilización de Puntos Objeto	Modelo de reutilización	Modelo de reutilización
Factor B en M_{Nnom}== A (Size)$_B$	☑ Orgánico=1.05 ☑ Semi-libre = 1.12 ☑ Rígido = 1.20	Rígido: 1.04-1.24 dependiendo del grado de : ☑ Eliminación anticipada del riesgo ☑ Arquitectura solida ☑ Requisitos estables ☑ Madurez del proceso Ada	1.0	1.01-1.21 dependiendo del grado de: ☑ Precedencia ☑ Conformidad ☑ Arquitectura anticipada, resolución de riesgos. ☑ Cohesión del equipo ☑ Madurez del proceso	1.01-1.21 dependiendo del grado de: ☑ Precedencia ☑ Conformidad ☑ Arquitectura anticipada, resolución de riesgos. ☑ Cohesión del equipo ☑ Madurez del proceso
DRIVERS DE COSTE DE PRODUCTO	RELY, DATA, CPLX	RELY*,DATA*,CPLX* RUSE	Ninguno	RCPX*$_T$, RUSE*$_T$	RELY, DATA, DOCU*$_T$, , CPLX*$_T$, RUSE*$_T$
DRIVERS DE COSTE DE LA PLATAFORMA	TIME,STOR,VIRT,TURN	TIME, STOR, VMVH, VMVT, TURN	Ninguno	Dificultad de la plataforma: PDIF*$_T$	TIME, STOR, PVOL(=VIRT)
DRIVERS DE COSTE DEL PERSONAL	ACAP, AEXP, PCAP, VEXP, LEXP	ACAP*, AEXP, PCAP*, VEXP, LEXP*	Ninguno	Capacidad yexperienciadelpersonal: PERS*$_T$, PREX*$_T$	ACAP*, AEXP $_T$, PCAP*, PEXP*$_T$, LTEX*$_T$, PCON*$_T$
DRIVERS DE COSTE DEL PROYECTO	MODP, TOOL, SCED	MODP*, TOOL*, SCED, SECU	Ninguno	SCED*$_T$, FCIL*$_T$	TOOL*$_T$, SCED, SITE*$_T$

Tabla B.2. Comparativa entre COCOMO II y sus predecesores.

3. Utilización de cada Modelo de Coste

3.1. Utilización del Modelo de Composición de Aplicaciones

Las primeras fases o ciclos en espiral utilizados en proyectos software de Generador de Aplicaciones, Integración de Sistema e Infraestructura implicarán generalmente prototipado y utilizarán las utilidades del Módulo de Composición de Aplicaciones. El Modulo de Composición de Aplicaciones COCOMO II, soporta estas fases y cualquier otra actividad de prototipado que se realice más adelante en el ciclo de vida.

Este modelo se dirige a aplicaciones que están demasiado diversificadas para crearse rápidamente en una herramienta de dominio específico, (como una hoja de cálculo) y que todavía no se conocen suficientemente como para ser compuestas a partir de componentes interoperables. Ejemplos de estos sistemas basados en componentes son los creadores de interfaces gráficas para usuario, bases de datos ó gestores de objetos, middleware para proceso distribuido ó transaccional, manejadores hipermedia, buscadores de datos pequeños y componentes de dominio específico tales como paquetes de control de procesos financieros, médicos ó industriales.

Dado que el Modelo de Composición de Aplicaciones incluye esfuerzos de prototipado para resolver asuntos potenciales de alto riesgo tales como interfaces de usuario, interacción software / sistema, ejecución ó grado de madurez tecnológica, los costes de este tipo de esfuerzo se estiman mejor mediante dicho modelo.

3.2. Utilización del Modelo de Diseño Anticipado

Hemos visto que las primeras fases o ciclos en espiral utilizados en proyectos software de Generador de Aplicaciones, Integración de Sistema e Infraestructura se ajustaban mejor al modelo de Composición de Aplicaciones.

Las siguientes fases ó ciclos espirales normalmente incluirán la exploración de arquitecturas alternativas ó estratégicas de desarrollo incremental. Para sostener estas actividades COCOMO II proporciona un modelo de estimación anticipado: el Modelo de Diseño Anticipado. El nivel de detalle de este modelo puede ser consistente con el nivel general de informa ión disponible y con el nivel general de aproximación de la estimación requerida en esta etapa.

El Diseño Anticipado incluye la exploración de arquitecturas de software/sistema alternativas y conceptos de operación. En esta fase no se sabe lo suficiente como para dar soporte a la estimación de grano fino. La correspondiente capacidad de COCOMO II incluye el uso de Puntos de Función y un conjunto de siete drivers de coste de grano grueso (por ejemplo, dos drivers de coste para capacidad del personal y experiencia del personal en lugar de los seis drivers de coste del Modelo Post-Arquitectura que cubren varios aspectos de capacidad del personal, continuidad y experiencia).

El modelo de Diseño Anticipado usa Puntos de Función No Ajustados como métrica de medida. Este modelo se utiliza en las primeras etapas de un proyecto software, cuando se conoce muy poco sobre el tamaño del producto que se va a desarrollar, la naturaleza de la plataforma objetivo, la naturaleza del personal involucrado en el proyecto ó especificaciones detalladas del proceso que se va a usar. Este modelo puede aplicarse a cada uno de los sectores de desarrollo de Generador de Aplicaciones, Integración de sistemas ó Infraestructura.

3.3. Utilización del Modelo PostArquitectura

Hemos visto que las primeras fases o ciclos en espiral usados en proyectos software de Generador de Aplicaciones, Integración de Sistema e Infraestructura se ajustaban mejor al modelo de Composición de Aplicaciones y que las siguientes fases ó ciclos espirales normalmente serán sostenidas por el modelo de Diseño Anticipado. Una vez que el proyecto está listo para desarrollar y sostener un sistema especializado, debe haber una arquitectura de ciclo de vida que proporcione información más precisa de los drivers de coste de entradas y permita cálculos de coste más exactos. Para apoyar esta etapa COCOMO II proporciona el Modelo Post-Arquitectura.

El Modelo PostArquitectura incluye el actual desarrollo y mantenimiento de un producto software. Esta fase avanza rentablemente si se desarrolla una arquitectura de ciclo de vida software válida con respecto a la misión del sistema, al concepto de operación y al riesgo, y establecido como marca de trabajo del producto. El modelo correspondiente de COCOMO II tiene aproximadamente la misma granularidad que los anteriores modelos COCOMO y AdaCOCOMO. Utiliza instrucciones fuente y/o Puntos de Función para medir, con modificadores para reutilización y objetos software; un conjunto de 17 drivers de coste multiplicativos; y un conjunto de 5 factores que determinan el exponente de escala del proyecto. Estos factores sustituyen los modos de desarrollo (Orgánico, Semilibre y Rígido) del modelo original COCOMO y refina los 4 factores de exponente-escala en Ada COCOMO.

4. Modelo de Coste de Composición de Aplicaciones

4.1. Introducción a los Puntos Objeto

Los Puntos Objeto son el recuento de pantallas, informes y módulos de lenguajes de 3ª generación desarrollados en la aplicación, cada uno ponderado mediante un factor de complejidad de tres niveles (simple, medio y complejo) . La estimación de Puntos Objeto es un enfoque relativamente nuevo de medida del software, pero encaja bien en las prácticas del sector de la Composición de Aplicaciones. También encaja muy bien en los esfuerzos de prototipado asociados, basados en el uso de herramientas ICASE que proporcionan constructores de interfaces gráficos de usuario, herramientas de desarrollo software; y en general, infraestructura que puede componerse y componentes de aplicación. En estas áreas se ha comparado bien con la estimación de Puntos de Función en un conjunto de aplicaciones no triviales (pero todavía limitadas).

Uno de los estudios comparativos entre la estimación de Puntos de Función y Puntos Objeto analizó una muestra de 19 proyectos software sobre inversión bancaria de una misma organización, desarrollado utilizando las utilidades de Composición de Aplicaciones ICASE y con un rango de esfuerzo de 4.7 a 71.9 MM (Meses-Persona). El estudio descubrió que el enfoque de Puntos Objeto justificó un 73% de la varianza (R2) en meses-persona ajustados por la reutilización, comparada con el 76% para los Puntos de Función.

Un experimento posterior diseñado estadísticamente incluyó cuatro gestores de proyecto experimentados utilizando Puntos de Función y Puntos Objeto para estimar el esfuerzo requerido en dos proyectos completos (3.5 y 6 Meses-persona actuales), basado en las descripciones del proyecto disponibles al principio para dichos proyectos. El experimento descubrió que los Puntos de Función y los Puntos Objeto dan resultados comparablemente precisos (un poco más precisos con Puntos Objeto pero estadísticamente insignificantes). Desde el punto de vista de la utilización, el tiempo medio para producir un valor de Punto Objeto era de alrededor del 47% del correspondiente tiempo medio para producir un valor de Puntos de Función. También los gestores consideraron el método de Puntos Objeto más fácil de usar (ambos resultados fueron estadísticamente significantes).

De esta manera aunque estos resultados no están todavía extendidos, su ajuste al desarrollo software de Composición de Aplicaciones parece justificar la selección de Puntos Objeto como el punto de partida para el modelo de estimación de Composición de Aplicaciones.

4.2. Procedimiento de Obtención de Puntos Objeto

La definición de los términos utilizados en los Puntos de Objetos es la siguiente:

- NOP: Nuevos Puntos Objeto. (Cantidad de Puntos Objeto ajustados por la reutilización).
- Srvr: Número de tablas de datos del servidor (mainframe ó equivalente) usadas junto con la pantalla o el informe.
- Clnt: Número de tablas de datos del cliente (estación de trabajo personal) usadas junto con la pantalla o el informe.
- %reuse: Porcentaje de pantallas, informes y módulos 3GL reutilizados a partir de aplicaciones anteriores prorrateadas por grado de reutilización.

Los ratios de productividad se basan en los datos del proyecto del año 1 y año 2. En el año 1 la herramienta CASE estaba construyéndose y los desarrolladores no la

conocían. La productividad media de NOP/Mes-persona en los 12 proyectos del año 1 se asocia con los niveles bajos de desarrollador y madurez y capacidad ICASE. En los siete proyectos del año 2, ambos, herramientas CASE y capacidades de los desarrolladores se consideran más maduras. La productividad media era 25 NOP/ Meses-persona, que se corresponde con los niveles altos de madurez ICASE y del desarrollador. Hemos de destacar que el uso del término "objeto" en Puntos Objeto define pantallas, informes y módulos 3GL como objetos. Esto puede tener relación ó no con otras definiciones de objetos como aquellas características de posesión tales como, por ejemplo, pertenencia a una clase, herencia, encapsulación, paso de mensajes y así sucesivamente. Las reglas de recuento para "objetos" de esa naturaleza, cuando se usan en lenguajes como C++, se discutirán en el Capítulo del Modelo Post-Arquitectura.

a. Hacer el recuento de objetos: Estimar el número de pantallas, informes y componentes de las que consta esta aplicación. Suponer las definiciones estándar de estos objetos en el entorno ICASE correspondiente.
b. Clasificar cada instancia de objeto dentro de niveles de complejidad simple, media y difícil dependiendo de los valores de las dimensiones de la característica. Usar la tabla B.3:

Para Pantallas				**Para Informes**			
Nº de vistas que contiene	# y fuente de tablas de datos			Nº de secciones que contiene	# y fuente de tablas de datos		
	Total <1 (<2 srvr <3 clnt)	Total <8 (2/3 srvr 3-5 clnt)	Total 8+ (>3 srvr >5 clnt)		Total <1 (<2 srvr <3 clnt)	Total <8 (2/3 srvr 3-5 clnt)	Total 8+ (>3 srvr >5 clnt)
<3	Simple	Simple	Medio	0 ó 1	Simple	Simple	Medio
3-7	Simple	Medio	Difícil	2 ó 3	Simple	Medio	Difícil
>8	Simple	Difícil	Difícil	4+	Medio	Difícil	Difícil

Tabla B.3. Complejidad asociada a las instancias de objetos

c. Pesar el número de cada celda usando la tabla B.4. El peso refleja el esfuerzo relativo que se requiere para implementar una instancia de ese nivel de complejidad:

Tipo de Objeto	**Complejidad-Peso**		
	simple	medio	difícil
pantalla	1	2	3
informe	2	5	8
componente 3GL			10

Tabla B.4. Pesos asociados a los niveles de complejidad

d. Determinar Puntos Objeto: Suma todas las instancias de objeto pesadas para conseguir un número. El recuento de Puntos Objeto.
e. Estimar el porcentaje de reutilización que se espera lograr en este proyecto. Calcular los Nuevos Puntos Objeto a desarrollar.

NOP = [(Object Points) x (100 - %Reuse)] / 100

f. Determinar un ratio de productividad PROD = NOP/ Meses-persona a partir de la tabla B.5:

Experiencia y capacidad de los desarrolladores	Muy Bajo	Bajo	Nominal	Alto	Muy Alto
ICASE madurez y capacidad	Muy Bajo	Bajo	Nominal	Alto	Muy Alto
PROD	4	7	13	25	50

Tabla B.5 Ratio de productividad PROD

g. Calcular el valor Meses-Persona (Month-Man) estimado según la ecuación:

MM =NOP/ PROD

5. Modelo de Diseño Anticipado y PostArquitectura

Los modelos de Diseño Anticipado y PostArquitectura se basan en la misma filosofía a la hora de proporcionar una estimación. Como hemos indicado ya 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. La fórmula básica para obtener una estimación de esfuerzo con ambos modelos es:

$$\mathbf{MM}_{\boldsymbol{Nominal}} = A * (Size)^{n}$$

Esta ecuación calcula el esfuerzo nominal para un proyecto de un tamaño dado expresado en Meses-persona (MM).

Las entradas son la medida del desarrollo del software, una constante, A, y un factor de escala, B. La medida está en unidades de líneas de código fuente (KSLOC). Esto se deriva de la medida de módulos software que constituirán el programa de aplicación, puede estimarse también a partir de Puntos de Función sin ajustar convirtiendo a SLOC y luego dividiendo por 1000.

El factor de escala (exponencial B), explica el ahorro ó gasto relativo de escala encontrado en proyectos software de distintos tamaños. La constante A, se usa para cortar los efectos multiplicativos de esfuerzo en proyectos de tamaño incremental.

A continuación se de describen los componentes del modelo de Diseño Anticipado:

1) **Constante A:** Como ya hemos visto anteriormente la constante A, se usa para capturar los efectos multiplicativos de esfuerzo en proyectos de tamaño incremental. Provisionalmente se le ha estimado un valor de 2.45.
2) **Variable Size:** Donde:

$$\mathbf{Size} = \mathbf{Size} * (1 + (\mathbf{BRAK}/100))$$

COCOMO II utiliza un porcentaje de Rotura BRAK para ajustar el tamaño eficaz del producto. La rotura refleja la volatilidad de los requisitos en un proyecto. Es el porcentaje de código desperdiciado debido a la volatilidad de los requisitos. El factor BRAK no se usa en el Modelo de Composición de Aplicaciones donde se espera un cierto grado de iteración en el producto y se incluye en la calibración de datos.

El tamaño de una aplicación se mide en unidades de líneas de código fuente (KSLOC). Al igual que en la versión inicial del COCOMO, este valor se deriva de la medida de módulos software que constituirán el programa de aplicación, sin embargo, en la nueva versión COCOMO II puede estimarse también a partir de Puntos de Función sin ajustar convirtiendo a SLOC y luego dividiendo por 1000.

Si se opta por utilizar directamente el valor del número de líneas de código, la meta es medir la cantidad de trabajo intelectual que se emplea en el desarrollo del programa, pero las dificultades aparecen al intentar definir medidas consistentes en diferentes lenguajes.

Si se opta por utilizar los Puntos de Función sin Ajustar para determinar el tamaño del proyecto, éstos deben convertirse en líneas de código fuente en el lenguaje de implementación (ensamblador, lenguajes de alto nivel, lenguajes de cuarta generación, etc.) para evaluar la relativamente concisa implementación por Puntos de Función (utilizando unas tablas de conversión).

3) Variable B (Ahorro y gasto de escala):

$$B = 0.91 + 0.01\sum_{j=1}^{5} SF_j \qquad (1)$$

Los modelos de estimación de coste del software a menudo tienen un factor exponencial para considerarlos gastos y ahorros relativos de escala encontrados en proyectos software de distinto tamaño. El exponente B se usa para capturar estos efectos. El valor de B es calculado en la ecuación anterior.

- Si B < 1.0. El proyecto presenta ahorros de escala. Si el tamaño del producto se dobla, el esfuerzo del proyecto es menor que el doble. La productividad del proyecto aumenta a medida que aumenta el tamaño del producto.

- Si B = 1.0. Los ahorros y gastos de escala están equilibrados. Este modelo lineal se usa a menudo para la estimación de coste de proyectos pequeños.

- Si B > 1.0. El proyecto presenta gastos de escala. Esto se debe normalmente a dos factores principales: El crecimiento del gasto en comunicaciones y el gasto en crecimiento de la integración de un gran sistema.

El exponente B se obtiene mediante los denominados *drivers de escala*. La selección de drivers de escala se basa en la razón de que ellos son un recurso significante de variación exponencial en un esfuerzo ó variación de la productividad del proyecto. Cada driver de escala tiene un rango de niveles de valores desde Muy Bajo hasta Extra Alto (tabla B.6). Cada nivel de valores tiene un peso, SF, y el valor específico del peso se llama *factor de escala*.

Factores de Escala (SR)	Muy Bajo	Bajo	Nominal	Alto	Muy Alto	Extra Alto
PREC	Completamente sin precedentes	Prácticamente sin precedentes	Casi sin precedentes	Algo familiar	Muy familiar	Completamente familiar
FLEX	Riguroso	Relajación ocasional	Algo de relajación	Conformidad general	Algo de conformidad	Metas generales
RESL*	Poco (20%)	Algo (40%)	A menudo (60%)	Generalmente (75%)	En su mayor parte (90%)	Por completo (100%)
TEAM	Interacciones muy difíciles	Algo de dificultad en las interacciones	Interacciones básicamente cooperativas	Bastante cooperativo	Altamente cooperativo	Completas interacciones
PMAT	Peso medio de respuesta “Si” para el cuestionario de Madurez CMM					

Tabla B.6. Factores de escala para el Modelo de COCOMO II de Diseño Anticipado

La descripción de cada uno de los factores de escala mostrados en la tabla B.6, es:

- **(PREC) (FLEX)**. Precedencia y Flexibilidad de desarrollo: Estos dos factores de escala capturan en gran parte las diferencias entre los modos Orgánico, Semilibre y Rígido del modelo original COCOMO.

- **(RESL)**. Arquitectura / Resolución de Riesgos: Este factor combina dos factores de medida de AdaCOCOMO "Minuciosidad del diseño por revisión del diseño del producto (PDR)" y "Eliminación de riesgos por PDR".

- **(TEAM)**. Cohesión del Equipo: El factor de escala de Cohesión del Equipo explica los recursos de turbulencia y entropía del proyecto debido a dificultades en la sincronización de los implicados en el proyecto, usuarios, clientes, desarrolladores, los que lo mantienen, etc. Estas dificultades pueden aparecer por las diferentes culturas y objetivos de los implicados; dificultades en conciliar objetivos y la falta de experiencia y de familiaridad de los implicados en trabajar como un equipo.

- **(PMAT)**. Madurez del proceso: El procedimiento para determinar PMAT se obtiene a través del Modelo de Madurez de Capacidad del Instituto de Ingeniería del Software (CMM). El periodo de tiempo para medir la madurez del proceso es el momento en el que el proyecto comienza. Hay dos formas de medir la madurez del proceso.

    - *Nivel de Madurez Global:* La primera toma el resultado de una evaluación organizada basada en el CMM.

        - ☑ Nivel 1 CMM (Mitad inferior)
        - ☑ Nivel 1 CMM (Mitad superior)
        - ☑ Nivel 2 CMM
        - ☑ Nivel 3 CMM
        - ☑ Nivel 4 CMM
        - ☑ Nivel 5 CMM

    - *Áreas de Proceso Principales:* La segunda está organizada en base a 18 áreas de proceso principales (KPA´s) en el modelo de Madurez de Capacidad SEI. El procedimiento para determinar PMAT es decidir el porcentaje de conformidad para cada uno de los KPA´s. Si el proyecto ha sufrido una valoración CMM reciente entonces se usa el porcentaje de conformidad para la KPA global (basada en datos de valoración de la conformidad práctica principal). Si no se ha hecho una valoración

entonces se usan los niveles de conformidad para las metas de los KPA´s para poner el nivel de conformidad.

4) **Ajuste Mediante Drivers De Coste:** Los drivers de coste se usan para capturar características del desarrollo del software que afectan al esfuerzo para completar el proyecto. Los drivers de coste tienen un nivel de medida que expresa el impacto del driver en el esfuerzo de desarrollo. Estos valores pueden ir desde Extra Bajo hasta Extra Alto. Para el propósito del análisis cuantitativo, cada nivel de medida de cada driver de coste tiene un peso asociado. El peso se llama multiplicador de esfuerzo (EM). La medida asignada a un driver de coste es 1.0 y el nivel de medida asociado con ese peso se llama nominal. Si un nivel de medida produce más esfuerzo de desarrollo de software, entonces su correspondiente EM está por encima de 1.0. Recíprocamente si el nivel de medida reduce el esfuerzo entonces el correspondiente EM es menor que 1.0. La selección de multiplicadores de esfuerzo se basa en una fuerte razón que explicaría una fuente significativa de esfuerzo de proyecto ó variación de la productividad, independientemente. Los EM se usan para ajustar el esfuerzo Meses-persona nominal. Hay 7 multiplicadores de esfuerzo para el Modelo de Diseño Anticipado y 17 para el modelo de PostArquitectura .

$$MM = A \times (Size)^{B} \times \prod \mathbf{EM}_i$$

Los drivers de coste del Diseño Anticipado se obtienen combinando los drivers de coste del modelo Post-Arquitectura de la tabla B.7. Siempre que una evaluación de un driver de coste está entre niveles de ratio, hay que redondear al valor más próximo al nominal. Por ejemplo, si un valor de un driver de coste está entre Muy Bajo y Bajo, entonces seleccionar Bajo.

Drivers de coste del Diseño anticipado	Drivers de coste combinados, homólogos del PostArquitectura
RCPX	RELY, DATA, CPLX, DOCU
RUSE	RUSE
PDIF	TIME, STOR, PVOL
PERS	ACAP, PCAP, PCON
PREX	AEXP, PEXP, LTEX
FCIL	TOOL, SITE
SCED	SCED

Tabla B.7 Multiplicadores de esfuerzo del Diseño Anticipado y PostArquitectura

Apéndice C

Glosario de Términos

- **Algoritmos genéticos:** Técnicas de optimización que usan procesos tales como combinación genética, mutación y selección natural en un diseño basado en los conceptos de evolución natural.

- **Análisis de series de tiempo (time-series):** Análisis de una secuencia de medidas hechas a intervalos específicos. El tiempo es usualmente la dimensión dominante de los datos.

- **Análisis prospectivo de datos:** Análisis de datos que predice futuras tendencias, comportamientos o eventos basado en datos históricos.

- **Análisis exploratorio de datos:** Uso de técnicas estadísticas tanto gráficas como descriptivas para aprender acerca de la estructura de un conjunto de datos.

- **Análisis retrospectivo de datos:** Análisis de datos que provee una visión de las tendencias, comportamientos o eventos basado en datos históricos.

- **Aprendizaje automático:** Modelo de entrenamiento para determinar el conocimiento a partir de un conjunto de datos.

- **Aprendizaje no supervisado:** Modelo de entrenamiento para determinar el conocimiento a partir de un conjunto de datos, en el cual no se dispone de otra información durante la fase de entrenamiento

- **Aprendizaje supervisado:** Modelo de entrenamiento para determinar el conocimiento a partir de un conjunto de datos, donde se dispone de cierto conocimiento de los resultados durante la fase de entrenamiento

- **Árbol de decisión:** Estructura en forma de árbol que representa un conjunto de decisiones. Estas decisiones generan reglas para la *clasificación* de un conjunto de datos. Ver *CART* y *CHAID.*

- **Base de datos multidimensional:** Base de datos diseñada para procesamiento analítico on-line (*OLAP*). Estructurada como un hipercubo con un eje por dimensión.

- **CART Árboles de clasificación y regresión**: Una técnica de *árbol de decisión* usada para la *clasificación* de un conjunto da datos. Provee un conjunto de reglas que se pueden aplicar a un nuevo (sin clasificar) conjunto de datos para predecir cuáles registros darán un cierto resultado. Segmenta un conjunto de datos creando 2 divisiones. Requiere menos preparación de datos que *CHAID*.

- **CHAID Detección de interacción automática de Chi cuadrado**: Una técnica de *árbol de decisión* usada para la *clasificación* de un conjunto da datos. Provee un conjunto de reglas que se pueden aplicar a un nuevo (sin clasificar) conjunto de datos para predecir cuáles registros darán un cierto resultado. Segmenta un conjunto de datos utilizando test de chi cuadrado para crear múltiples divisiones. Antecede, y requiere más preparación de datos, que *CART.*

- **Clasificación**: Proceso de dividir un conjunto de datos en grupos mutuamente excluyentes de tal manera que cada miembro de un grupo esté lo "más cercano" posible a otro, y grupos diferentes estén lo "más lejos" posible uno del otro, donde la distancia está medida con respecto a variable(s) específica(s) las cuales se están tratando de predecir. Por ejemplo, un problema típico de clasificación es el de dividir una base de datos de compañías en grupos que son lo más homogéneos posibles con respecto a variables como "posibilidades de crédito" con valores tales como "Bueno" y "Malo".

- **Clustering (agrupamiento)**: Proceso de dividir un conjunto de datos en grupos mutuamente excluyentes de tal manera que cada miembro de un grupo esté lo "más cercano" posible a otro, y grupos diferentes estén lo "más lejos" posible uno del otro, donde la distancia está medida con respecto a todas las variables disponibles.

- **Cocomo**: (Constructive Cost Model). Modelo constructivo de costes. Su primera versión fue creada por Boehm en 1981. Cocomo ha evolucionado hasta su reciente versión CocomoII.

- **Computadoras con multiprocesadores**: Una computadora que incluye múltiples procesadores conectados por una red. Ver *procesamiento paralelo.*

- **Conjunto de prueba:** Conjunto de instancias utilizadas para evaluar un modelo de KDD

- **Data cleaning:** Proceso de asegurar que todos los valores en un conjunto de datos sean consistentes y correctamente registrados.

- **Data Mining (Minería de Datos)**: Etapa del proceso de KDD cuyo objetivo es inducir patrones a partir de un conjunto de datos preprocesado y transformado.

- **Data Warehouse**: Sistema para el almacenamiento y distribución de cantidades masivas de datos

- **Datos anormales**: Datos que resultan de errores (por ejemplo: errores en el tipeado durante la carga) o que representan eventos inusuales.

- **Dimensión**: En una base de datos relacional o plana, cada campo en un registro representa una dimensión. En una *base de datos multidimensional*, una dimensión es un conjunto de entidades similares; por ejemplo: una base de datos multidimensional de ventas podría incluir las dimensiones Producto, Tiempo y Ciudad.

- **KDD (Knowledge Discovery in Data):** Proceso compuesto por varias etapas cuyo objetivo es inducir un conjunto de patrones que se transformará en información útil.

- **Lógica difusa (Fuzzy Logic):** Estrategia basada en conjuntos difusos, donde cada elemento del conjunto tiene una probabilidad de pertenencia dentro del intervalo [0,1].

- **Modelo analítico**: Una estructura y proceso para analizar un conjunto de datos. Por ejemplo, un *árbol de decisión* es un modelo para la *clasificación* de un conjunto de datos

- **Modelo lineal:** Un *modelo analítico* que asume relaciones lineales entre una variable seleccionada (dependiente) y sus predictores (variables independientes).

- **Modelo no lineal**: Un *modelo analítico* que no asume una relación lineal en los coeficientes de las variables que son estudiadas.

- **Modelo predictivo:** Estructura y proceso para predecir valores de variables especificadas en un conjunto de datos.

- **Navegación de datos:** Proceso de visualizar diferentes dimensiones, "fetas" y niveles de una *base de datos multidimensional*. Ver *OLAP*.

- **OLAP Procesamiento analítico on-line (On Line Analitic prossesing):** Se refiere a aplicaciones de bases de datos orientadas a array que permite a los usuarios ver, navegar, manipular y analizar *bases de datos multidimensionales*.

- **Outlier:** Un item de datos cuyo valor cae fuera de los límites que encierran a la mayoría del resto de los valores correspondientes de la muestra. Puede indicar *datos anormales*. Deberían ser examinados detenidamente; pueden dar importante información.

- **Patrones (Patterns):** Relaciones no lineales entre atributos de un conjunto de datos que posteriormente se transforman en conocimiento.

- **Preprocesado:** Etapa del KDD cuyo objetivo es la preparación y adecuación de los datos

- **Procesamiento paralelo**: Uso coordinado de múltiples procesadores para realizar tareas computacionales. El procesamiento paralelo puede ocurrir en una *computadora con múltiples procesadores* o en una red de estaciones de trabajo o PCs.

- **RAID**: Formación redundante de discos baratos (Redundant Array of inexpensive disks). Tecnología para el almacenamiento paralelo eficiente de datos en sistemas de computadoras de alto rendimiento.

- **Regla de Asociación:** Información sobre la relación entre los atributos de un conjunto de datos. Está formada por un antecedente y un consecuente.

- **Regresión lineal:** Técnica estadística utilizada para encontrar la mejor relación lineal que encaja entre una variable seleccionada (dependiente) y sus predicados (variables independientes).

- **Regresión logística:** Una regresión lineal que predice las proporciones de una variable seleccionada categórica, tal como Tipo de Consumidor, en una población.

- **Ruido:** Conjunto de instancias que no se adaptan a un modelo preestablecido.

- **Selección de atributos (Feature Selection):** Técnica de reducción de atributos para un conjunto de datos, con el objetivo de obtener un conjunto de datos de menor dimensión, pero con la misma información cualitativa.

- **SMP Multiprocesador simétrico (Symmetric multiprocessor):** Tipo de *computadora con multiprocesadores* en la cual la memoria es compartida entre los procesadores

- **Vecino más cercano:** Técnica que clasifica cada registro en un conjunto de datos basado en una combinación de las clases del / de los k registro (s) más similar / es a él en un conjunto de datos históricos (donde $k > 1$). Algunas veces se llama la técnica del vecino k-más cercano.

Bibliografía

[Abdel91] T. Abdel-Hamid, S. Madnick. *Software Project Dynamics: an integrated approach*. Prentice-Hall, 1991.

[Aggar99] C.C. Aggarwal, S.C. Gates, P.S. Yu. *On the merits of building categorization systems by supervised clustering*. In Proc. fifth ACM SIGKDD conference on KDD and Data Mining, pp. 352-356, 1999

[Agraw93] R. Agrawal, T. Imielinski, A. Swami. *Mining association rules between sets of items in large databases*. In Proceedings of the 1993 ACM-SIGMOD International Conference on Management of Data, pages 207-216, Washington, D.C., 1993.

[Agraw98] R. Agrawal, J. Gehrke, D. Gunopoulos, P. Raghavan. *Automatic subspace clustering of high dimensional data for data mining applications*. In Proceedings of the 1998 ACM-SIGMOD International Conference on Management Data, pages 94-105, 1998.

[Aguil97] J.S. Aguilar, J.C. Riquelme, M. Toro. *Cogito: Un sistema de autoaprendizaje basado en algoritmos genéticos*. In III Jornadas de Informática, 1997

[Aguil98] J.S. Aguilar, J.C. Riquelme, M. Toro. *A tool to obtain a hierachical qualitative set of rules from quantitative data*. In Lecture Notes in Artificial Inteligence 1415. Springer-Verlag, pp. 336-346, 1998

[Aguil99] J.S. Aguilar, J.C. Riquelme, M. Toro. *Three geometric approaches for representing decicion rules in a supervised learning system*. In Genetic and Evolutionary Computattion Conference (GECCO ’99), pag. 771, Orlando, Florida, 1999

[Aguil00] J.S. Aguilar, F. Ferrer. *Tree-nn: Clasificación por vecindad usando u k variable*. In IV Jornadas Científicas Andaluzas Tecnologías de la Información, pp. 169-174, Cádiz, Noviembre, 2000.

[Aguil01] J.S. Aguilar. *Generación de reglas jerárquicas de decisión con algoritmos evolutivos en aprendizaje supervisado.* Tesis doctoral. 2001.

[Aguil01b] J.S. Aguilar, I. Ramos, J.C. Riquelme, M. Toro. *An evolutionary approach to estimating software development projects*. Information and Software Technology. Ed. Elsevier. 43(2001), pp. 875-882, 2001.

[Aguil02] J.S. Aguilar, J.C. Riquelme, D. Rodríguez, I. Ramos. *Generation of management rules through system dynamics and evolutionary computation*. Lecture Notes in Computer Science. Vol. 2559, pp. 615-628. Springer-Verlag. ISBN 3-540-00234-0. 4th International Conference on Product Focused Software Process Improvement (PROFES 2002), Rovaniemi (Finland), December 2002

[Aguil03] J.S. Aguilar, J.C. Riquelme, M. Toro. *Evolutionary Learning of Hierarchical Decision Rules*. IEEE Systems, Man and Cibernetics Part B, Vol. 33(2), pp. 324 - 331 2003. ISSN 1083-4419

[Ahuja93] R.K. Ahuja, T.L. Magnanti, J.B. Orlin. *Network flows.* Prentice-Hall, Englewod Cliffs, NJ, USA. 1993.

[Albre79] A.J. Albrecht. *Measuring Application Development Productivity.* In IBM Applications Development Symposium, Monterey, CA, 1979, pp. 83-92.

[Albre83] A.J. Albrecht, Allan and J. Gaffney. “Software Function, Source Lines of Code, and Effort Prediction: A Software Science Validation” *IEEE Transactions on Software Engineering* vol. SE-9 no. 6 pp. 639-648. 1983

[Arifo93] A. Arifoglu. *Methodology for Software Cost Estimation*. ACM Sigsoft Software Engineering Notes 18(2):96-105, 1993.

[Aroba00] J. Aroba, I. Ramos, J.C. Riquelme. *Decision making in software projects using fuzzy clustering algorithms*. Process Simulation and Modeling PROSIM 2000. Section: ”Combining learning models with simulation”, Londres, 2000

[Banke94] R. Banker, H. Chang, C. Kemerer. *Evidence on Economies of Scale in Software Development.* Information and Software Technology, 1994.

[Basil85] V.R. Basili, N.M. Panlilio-Yap. *Finding Relationships Between Effort and Other Variables in the SEL.*" IEEE COMPSAC. October 1985.

[Basil94] V.R. Basili, S. Green. *Software Process Evolution at the SEL.* IEEE Software, pp. 58-66, July 1994.

[Bayle81] J.W. Bayley, V.R. Basili. *A meta-model for software development resource expenditures*. Proceedings of the 5th international conference on Software engineering , San Diego, California, United States. 1981

[Bezde81] J.C. Bezdek. *Pattern Recognition with fuzzy objective function algorithm*. New York : Plenum press., 1981.

[Bjorn99] B. Larsen, C. Aone. *Fast and effective text mining using linear-time document clustering*. In Proc. fifth ACM SIGKDD conference on KDD and Data Mining, pp. 16-22, 1999

[Blake98] C. Blake, C.J. Merz. *UCI repository of machine learning databases*. Irvine, California. University of California. Department of Information and Computer Sciences. 1998

[Blum88] A. Blum, R.L. Rivest. *Training a 3-node neutral network is np-complete.* In *proceedings of the first ADM workshop on the Computational Learning Theory, pages 9-18, Cambridge,* MA, 1988

[Boehm81] B.W. Boehm. *Software Engineering Economics.*Prentice-Hall, Inc. Englewood Cliffs, New Jersey. 1981

[Boehm95] B.W. Boehm, B. Clark, E. Horowitz, R. Madachy, R. Shelby, C. Westland. *The COCOMO 2.0 Software Cost Estimation Model.* International Society of Parametric Analysts, 1995.

[Boley98] D. Boley. *Principal direction divisive partitioning*. Data Mining and Knowledge discovery, 2(4), 1998.

[Bonis87] P.P. Bonissone, S.S. Gans, K.S. Decker. *RUM: A layered architecture for reasoning with uncertainly*. Proceedings of the 10th International Join Conference on Artificial Intelligence, IJCAI87, Milan, pp.891-898. 1987

[Borgw87] K. Borgwardt. *The Simplex method: A probabilistic analysis.* Springer-Verlag. 1987.

[Bourn97] Bournemouth University. *ANGEL - Analogy Effort Estimation* Tool. 17th January 1997

[Breim84] L. Breiman, J. Friedman, C.J. Stone, R.A. Olshen. *Classification and regresion trees*. Chapman & Hall /CRC eds. 1984

[Brian92] L. Briand , V. Basili, W. Thomas. *A Pattern Recognition Approach for Software Engineering Data Analysis.* IEEE Transactions on Software Engineering, 18(11):931-942, November 1992.

[Brook96] P. Brooks. *Targeting Customers. MCI Leverages Data Warehouse Techno-logy to Strengthen its Marketing Campaigns*. DBMS, December 1996.

[Brown94] L. Brownlow. *Early Estimating in the Object-Oriented Analysis and Design Environment*. 1994 European Software Cost Modelling Meeting.

[Burdi01] D. Burdick, M. Calimlim, J. Gehrke. *Mafia: a maximal frequent itemset algorithm for transactional databases.* In 17th Bibliography international conference on Data Engineering, 2001.

[Cao90] Z. Cao, A. Kandel, L. Li. *A new model of fuzzy logic reasoning.* Fuzzy sets and systems, nº36, pp.311-325.1990

[Chees96] P. Cheeseman, J. Stutz. *Baysian classification (autoclass): Theory and re-sults*. In Advances in KDD, pp.153-180, AAAI/MIT press., 1996.

[Charl91] R.C. Symons. *Software sizing and estimating: Mk II FPA (function point analysis).* John Wiley & Sons, 1991

[Chatt00] S. Chatterjee, A. Hadi, B. Price. *Regression Analysis by Example*. 3ªEd., John Wiley. 2000

[Chich93] K.J. Chichacly. *The bifocal vantage point: managing software projects from a Systems Thinking Perspective. American Programmer,* pp. 18-25, May 1993.

[Clark91] P. Clark, R. Boswell. *Rule induction with CN2: some recent mprovements*. Proceedings of the 5th European working session on learning. 1991

[Cohen95] W.W. Cohen. *Fast effective rule induction*. Proceedings of the 12th Intrnational conference on Machine Learning. 1995

[Conte86] S.D. Conte, H.E. Dunsmore, V. Shen. *Software Engineering, Metrics and Models.* Benjamin/Cummings publishing comp. Inc, 1986

[Cook99] R.D. Cook, S. Weisberg. *Applied Regression including Computing and Graphics.* Ed. John Wiley. 1999

[Cost93] S. Cost, S. Salzberg. *A Weighted nearest neighbor algorithm for learning with symbolic features*. Machine learning,10,pp. 57-78, 1993.

[Cover67] T.M. Cover, J.A. Hart. *Nearest neighbor pattern classification*. IEEE Transaction on Information Theory, IT-13(1).pp.21-27, 1967

[Cover68] T.M. Cover. *Estimation by nearest neighbor rule*. IEEE Transaction on Information Theory, IT-14.pp.59-55, 1968

[Cover77] T.M. Cover, J. Van Campenhout. *On the possible orderings in the measurement selection problem.* IEEE Trans. Systems, man and cybernetics, Vol. SMC-7, pp. 657-661, 1977.

[Dasar91] B.V. Dasarathy. *Nearest neighbor (NN) norms: nn pattern clasification thechiques*. IEEE Computer society press, 1991

[Dataf01] DataFit V 7.1.44, © 1995-2001, Oakdale Engineering, RC136

[Dejon75] K.A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, 1975.

[Dejon93] K.A. De Jong, W.M. Spears, D.F. Gordon. *Using Genetic Algorithms for concept learning*. Machine Learning, 13. pp. 161-188, 1993.

[DeMar82] T. DeMarco. *Controlling Software Project*s. Yourdon Press, Prentice-Hall. Englewood Cliffs, New Jersey. 1982

[Dolad97] J.J. Dolado. *A study of the relationships among Albrecht and Mark II function points, lines of code 4GL and effort.* Journal of System and Software 37(2), pp.161-173, 1997.

[Dolad98] J.J. Dolado, L. Fernandez *Genetic programming, neural networks and linear regression in software project estimation.*In INSPIRE III, Process Improvement through training and education. C. Hawkins, M.Ross, G. Staples, J.B. Thompson (Eds.), The British Computer Society, pp. 157-171, 1998.

[Dolad99] J.J. Dolado, L. Fernandez, M.C. Otero, L. Orkola. *Software effort estimation: The elusive goal in project management*. Proceedings of ICEIS'99, pp. 412-418, 1999.

[Dolad01] J.J. Dolado. *On The problem of the software cost function*. Information and Software Technology. Ed. Elsevier, 43(2001), pp.61-72, 2001.

[Duboi80] D. Dubois, H. Prade. *Fuzzy sets and systems: Theory and applications.* Academic press. New York, 1980.

[Duda73] R. Duda, P. Hart. *Pattern classification and scene analysis*. John Wiley and sons, 1973.

[Duda01] R. Duda, P.E. Hart, D.G. Stork. *Pattern classification*. John Wiley and sons, 2001.

[Dudan76] S.A. Dudani. *The distance-weighted k-nearest meighbor rule*. IEEE Transaction on Systems, Man and Cybernetics, SMC-6, 4:325-327, 1976.

[Efron83] B. Efron. *Estimating the error rate of a prediction rule*. Journal of the American statistical association, nº78.pp. 316-333, 1983.

[Enrig00] A.J. Enright, C. Ouzounis. *GeneRAGE: a robust algotihm for sequence clustering and domain detection.* Bioinformatics, 16(5):451-457, 2000

[Ester80] R. Esterling. *Software Manpower Costs : A Model.* Datamation, March 1980.

[Estev97] P. Estevez. *Clasificación de patrones mediante redes neuronales artificiales.* Anales del Instituto de Ingenieros de Chile,.Vol. 111, nº1, pp.24-31, 1999.

[Fang01] S.C. Fang, H.S. Tsao. *Maximun entropy principle: Image recons-truction*. In Floudas and Pardalos, pp. 245-249, 2001.

[Fei92] Z. Fei, X. Liu. *f-COCOMO: Fuzzy constructive cost model in software engineering*. In Proc. IEEE Int. Conf. on Fuzzy Systems, 1992, pp331-337

[Fukuy89] Y. Fukuyama, M. Sugeno. *A new method of choosing the number of clusters for fuzzy c-means method.* Proceedings of the 5th Fuzzy Systems Symposium, 1989.

[Giral03] R. Giráldez , J.S. Aguilar, J.C. Riquelme. *An efficient data structure for decision rules discovery*. Proceedings of 18th ACM Symposium on Applied Computing (SAC 2003), pp: 475-479 Melbourne (Florida, USA) March 2003. ACM Inc. ISBN 1-58113-624-2

[Godo88] L. Godo, R. Lopez, C. Sierra, A. Verdaguer. *Managing linguistically expressed uncertainly in Milord: application to medical diagnosis*. Artificial Intelligence Communications, pp.14-31. 1988

[Grazi00] G. Frosini, B. Lazzerini, F. Marcelloni. *A modified Fuzzy c-Means Algorithm for Feature Selection*. In Thomas Whalen, editor, NAFIPS00, Atlanta, Georgia, USA, pages 148-152, 2000.

[Grefe91] J. Grefenstette. *Lamarckian learning in multi-agent enviromets*. Proceedings of the fourth International Conference on Genetic Algorithms. pp 303-310, Morgan Kaufmann. 1991

[Goldb89] D.E. Goldberg. *Genetic algoritms in search, optimization and machine learning*. Addison Wesley, Reading, M. 1989

[Guha98] S. Guha, R. Rastogi, K. Shim.*CURE: An efficient clustering algorithm for large databases*. Proceedings of ACM SIGMOD, 1998

[Guha99] S. Guha, R. Rastogi, K. Shim.*ROCK: A Robust clustering algorithm for categorical attributes.* Proceedings of the 15th International Conference on Data Engineering, 1999

[Han01] J. Han, M. Kamber, A.K. Tung. *Spatial clustering methods in data mining: A survey*. Geographic Data Mining and Knowledge Discovery. Taylor and Francis, 2001

[Hayki99] S. Haykin. *Neural Networks*. 2º Edition, Prentice Hall. 1999

[Heems92] F.L. Heemstra. *Software Cost Estimation*. Information and Software Technology, Vol. 34, No. 10, Oct. 1992.

[Helme67] O. Helmer. *Analysis of the Future: The Delphi Method* Santa Monica, CA. RandCorporation, 1967

[Holla86] J. Holland. *Escaping brittleness: The possibilities of general purpose learning algorithms applied to parallel rule-based systems*. Machine learning: An artificial intelligence approach, vol. 2, Ed. Michalski, 1986.

[Holmb82] L.P. Holmblad, J.J. Ostegaard. *Control of a cement kiln by fuzzy logic*. Fuzzy information and decision processes. M. Gupta, E. Sanchez Eds. North Holland, Amsterdam. 1982

[Humph95] A. Humphrey, S. Watts. *A Discipline for Software Engineering* Addison-Wesley. Reading, Massachusetts. 1995

[Hunt66] E.B. Hunt, J. Marin, P.J. Stone, *Experiments in induction.* Academic Press, 1966

[Ifpug94] IFPUG (International Function Point Users Group). CPM 4.0 de 1994 (Counting Practice Manual)

[Jain88] A.K. Jain, C. Dubes. *Algorithms for clustering data*. Prentice Hall, 1988

[Jain97] A.K. Jain, D. Zongker. *Feature selection: Evaluation, application and sample performance.* IEEE Trans. on pattern analysis and machine intelligence, Vol.19, nº2, pp. 153-158, 1997.

[Jain99] A.K. Jain, M.N. Murthy, P.J. Flynn. *Data clustering: A review*. ACM Computing Surveys, 31(3):264-323, 1999.

[Janik93] C.Z. Janikow. *A knowledge intensive genetic algorithm for supervised learning.* Machine learning, 13.pp. 189-228, 1993.

[Jeffe85] D.R. Jeffery, M.J. Lawrence. *Managing programming productivity.* Journal of Systems and Software, v.5 n.1, p.49-58, Feb. 1985

[Jeffe87] D.R. Jeffery. *A software development productivity model for MIS environments.* Journal of Systems and Software, v.7 n.2, p.115-125, June 1, 1987.

[Jeffe94] R. Jeffery. *Implementing Successful Function Point and Other Metrics Initiatives*. In European Software Cost Modelling Conference, Ivrea, pp. 1-60. 1994.

[Jeffe96] R. Jeffery, J. Stathis. *Function Point Sizing: Structure, Validity and Applicability*. In Empirical Software Engineering, vol. 1, pp. 11-30. 1996.

[Jones96] C. Jones. *Applied Software Measurement* 2º Edition. McGraw-Hill. NewYork., 1996.

[Kanug02] T. Kanungo, D.M. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A.Y. Wu. *An Efficient k-Means Clustering Algorithm: Analysis and Implementation.* IEEE Trans. PAMI, 24(7):881-892, 2002.

[Karyp99] G. Karypis, E. Han, V. Kumar. *CHAMELEON: A Hierarchical clustering algorithm using dynamic modeling*. IEEE computer: Special issue on Data Analysis and Mining, 32(8), 1999.

[Kass80] G.V. Kass. *An exploratory technique for investigating large quantities of categorical data*. Applied Statistics, nº29, pp. 119-127. 1980

[Kaufm75] A. Kaufman. *An introduction to the theory of fuzzy sets*. Academic press, New York, 1975.

[Kaufm90] A. Kaufman, P.J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley and sons, 1990

[Kemer87] C.F. Kemerer. *An empirical validation of software cost estimation models.* In Communications of the ACM, vol. 30, pp. 406-429. 1987.

[Kiska85] J.B. Kiska, M.E. Kochanska, D.S. Sliwinska. *The influence of some fuzzy implication operators on the accuracy of a fuzzy model*. Fuzzy sets and systems, nº15, pp.111-128. 1985

[Kitch84] B. Kitchenham, N.R. Taylor. *Software cost models*. ICL Tech. J. 4.1, 73-102. 1984.

[Kitch90] B.A. Kitchenham. *Measuring to manage*. in Mitchell, R.J.: Managing Complexity in Software Engineering. IEE Computing Series 17, London, pp. 153-166. 1990.

[Kitch92] B.A. Kitchenham. *Empirical studies of assumptions that underlie software cost estimation.* Information and Softw. Tech., 34(4), 211-18, 1992.

[Kitch92b] B. Kitchenham. *A Methodology for Evaluating Software Engineering Methods and Tools*. Experimental Software Engineering Issues, 1992. pp. 121-124

[Kittl86] J. Kittler. *Feature selection and extraction*. In handbook of pattern recognition and image processing, T. Young and K. Fu Eds., Academic press, Chapter 3, pp.59-83, 1986

[Kilr88] G.J. Klir, T.A. Folger. *Fuzzy sets, Uncertainly and information*. Englewood cliffs, New Jersey, Prentice Hall, 1988.

[Krish02] K. Krishnamoorthy , B. Moore. *Combining information for prediction in linear regression*, Metrika, 56, 73-81, 2002.

[Lache68] P. Lacehnbruch, M. Mickey. *Estimation of error rates in discriminant analysis*. Technometrics, nº10.pp1-11,1968

[Last00] M. Last, A. Kandel, O. Maimon, E. Eberbach. *Anytime Algorithm for Feature Selection.* Proceedings of the Second International Conference on Rough Sets and Current Trends in Computing (RSCTC'2000), Banff, Canada, 2000.

[Last01] M. Last, A. Kandel, O. Maimon. *Information-Theoretic Algorithm for Feature Selection.* Pattern Recognition Letters, Vol. 22, No. 6-7, pp. 799-811, 2001.

[Lin01] K. Lin, C. Yang. *The ANN-tree: An index for efficient approximate nearest neighbor search.* Proceedings of the Seventh International Conference on Database Systems for Advanced Applications, April, 2001

[Lo96] R. Lo, R. Webby, R. Jeffery. *Sizing and Estimating the Coding and Unit Testing Effort for GUI Systems*. In Proceedings of Third international software metrics symposium,(metrics'96), Technical Univ. Berlin 1996.

[Low90] G.C. Low, D.R. Jeffery. *Function Points in the Estimation and Evaluation of the Software Process.* IEEE Trans. Software Engineering, 16 (1), pp. 64-71, 1990

[MacQu67] J. MacQueen. *Some methods for classification and analysis of multivariate observations.* In Proceedings 5th Symposium Maths Statistic, Prob., pp. 281-297, 1967

[Manda74] E.H. Mandani. *Applications of fuzzy algorithms for control of simple dynamic plant*. Proceedings IEEE, 121(12), pp- 1585-1588. 1974

[Manda75] E:H. Mandani, S. Assilian. *An experiment in linguistic systhesis with a fuzzy logic controler*. International Journal Man-Machine Stud. 7(1), pp.1-13. 1975

[Manee01] K.S. Maneewongvatana, D.M. Mount. *The Analysis of a Probabilistic Approach to Nearest Neighbor Searching.*' in Proc. 7th Workshop on Algorithms and Data Structures (WADS 2001), 2001, 276-286.

[Manga90] O.L. Mangasarian, R. Setiono, W.H. Wolberg. *Pattern recognition via linear programming: theory and application to medical diagnosis*. In T.F. Coleman, Y. Li editors, *Large-scale numerical optimization,* pp.22.30. 1990. SIAM publications, USA.

[Matso94] J.E. Matson, B.E. Barrett, J.M. Mellichamp. *Software development cost estimation using function points.* In IEEE Transactions on Software Engineering, vol. 20, 1994, pp. 275-287.

[Mitch97] T. Mitchell. *Machine learning*. McGraw Hill, 1997

[Miyaz85] Y. Miyazaki, K. Mori. *COCOMO evaluation and tailoring*. Proceedings of the 8th international conference on Software engineering, London, England. 1985

[Mizum82] M. Mizumoto, H.J. Zimmermann. *Comparison of fuzzy reasoning methods*. Fuzzy sets and system, nº8, pp. 253-283. 1982

[Mizum85] M. Mizumoto. *Extended fuzzy reasoning*. Aproximate reasoning in expert systems. M. Gupta, A. Kandel, W. Bandler Eds, pp. 71-85. 1985

[Mukho91] T. Mukhopadyay, S. Kekre, S. Datar, E. Svaan. *Overloaded Overheads: Activity-based Cost Analysis of Material Handling in Cell Manufacturing*. Journal of Operations Management. 1991. 119-137

[Mukho92] T. Mukhopadhyay, S. Vicinanza, M. Prietula. *Estimating the feasibility of a case-based*. Reasoning model for software effort estimation", MIS Quarterly, 16(2), 1992, pp. 155-171.

[Murth94] S.K. Murthy, S. Kasif, S. Salzberg. *A system for induction of oblique decision trees. Journal of artificial inteligence Research, 1994*

[Nages99] H. Nagesh, S. Goil, A. Choudhary. *Mafia: efficient and scalable sub-space clustering for very large data sets.* Technical report TR 9906-010, Northwest Univ., 1999

[NgHan94] R. Ng, J. Han. *Efficient and effective clustering method for spatial data mining*. In Proc. of 20th VLDB conference, pp. 144-155, Chile, 1994.

[Pabit02] M. Pabitra, C.A. Murthy, K. Sankar. *Unsupervised Feature Selection Using Feature Similarity*. TPAMI, 24(3):301-312, 2002

[Parr80] F.N. Parr. *An Alternative to Rayleigh Norden Curve Model for Software Development Effort.* IEEE Trans. on Software Eng., SE-6(3), May 1980

[Paulk93] M.C. Paulk, et al. *Capability Mature Model, version 1.1*, *IEEE Software*, July, 1993, pp. 18-27.

[Phili01] A.T. Philips. *Molecular structure determination: Convex global underestimation*. In floudas and Pardalos, pp. 421-425, 2001.

[Prepa85] F.P. Preparata, M.I. Shamos. *Computational Geometry: an Introduction*. Springer-Verlag, New York, 1985

[Prepr87] Preprints of the second congress of the international fuzzy systems association, Tokyo, Japan. 1987

[Press98] R. Pressman. *Ingeniería de Software, Un enfoque práctico*. Cuarta edición. Mc Graw Hill., 1998.

[Proce90] Proceedings of the international conference on fuzzy logic and neural networks, IIZUKA'90, Kyusu Institute of technology Iizuka, Japan.1990

[Proce91] Proceedings of the international fuzzy engineering symposium, IFES'91, Japan. 1991

[Putna78] L.H. Putnam. *A General Empirical Solution to the Macro Software Sizing and Estimating Problem. IEEE Transactions on Software Engineering* vol. SE-4 no. 4 pp. 345-361. 1978.

[Quinl93] J.R. Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kaufmann Publ. San Francisco, 1993.

[Quinl94] J.R. Quinlan. *The minimun description length principle and categorical theories.*In *Proceedings of the eleventh International Conference on Machine Learning, pages 233-241,* 1994

[Radcl90] N.J. Radcliffe. *Genetic Neural Networks on MIMD Computers*. PhD, University of Edinburg, 1990

[Ramos97] I. Ramos, M. Ruiz. *Modelo Dinámico simplificado para la gestión de proyectos software.* II Jornadas de Ingeniería del Software. Pág.: 146-157. San Sebastián, 1997.

[Ramos98] I. Ramos, M. Ruiz. *A Reduced Dynamic Model to Make Estimations in the Initial Stages of a Software Development Project*. In C. Hawkings et al (eds), INSPIRE III. Process Improvement through Training and Education, Pp.:172-185. London, 1998.

[Ramos98b] I. Ramos, J.C. Riquelme. *Obtención de reglas de gestión para proyectos de desarrollo de software*. III Jornadas de Ingeniería del Software. Pág.: 387-398. Murcia, 1998.

[Ramos98c] I. Ramos. *Un nuevo enfoque en la gestión de proyectos de desarrollo de software*. Tesis Doctoral. Universidad de Sevilla, 1998

[Ramos99] I. Ramos, M. Ruiz. *Monitorización y seguimiento estratégico de Proyectos de Desarrollo de Software*. XV Congreso Nacional de Ingeniería de Proyectos. EA04. León, 1999.

[Ramos99b] I. Ramos, J.C. Riquelme. *The Dynamic Models for Software Development Projects and the Machine Learning Techniques*. International Conference on product focussed software process improvement. Pp.: 560-574. Oulu (Finland), 1999.

[Ramos01] I. Ramos, J. Aroba, J.C. Riquelme. *Aplication of machine learning techniques to software project management*. ICEIS 2001 (III International Conference on Enterprise Information Systems), Setubal (Portugal), pag. 433, 2001.

[Rique97] J.C. Riquelme, J.S. Aguilar, M. Toro. *Cogito 2.0: Una herramienta para obtener un clasificador jerárquico de aprendizaje supervisado.* In VII Conferencia de la Asociación Española para la Inteligencia Artificial, pp. 489-498, 1997.

[Rique00] J.C. Riquelme, J.S. Aguilar, M. Toro. *Discovering hierarchical decision rules with evolutive algorithms in supervised learning.* International Journal of Computer, Systems and Signals, Vol. 1 nº 1, pp. 73 - 84, 2000. IAAMSAD ISSN 1608-5655

[Rissa83] J. Rissane. *A universal prior for integers and estimation by minimun description length. Annals of Statistics, 11:416-431,* 1983

[Rojas96] R. Rojas. *Neural networks: A systematic introduction.* Springer-Verlag, Berlin, Germany, 1996.

[Saari93] S. Saarinen, R. Bramley, G. Cybenko. *Ill-conditoning in neural network training problems*. SIAM J. Sci. Comput. 14(3):693-714, 1993.

[Siedl88] W. Siedlecki, J. Sklansky. *On automatic feature selection*. Int. Journal of pattern recognition and artificial intelligence, ol 2, nº2, pp. 197-220, 1988

[Smith80] S.F. Smith. *A learning system based on genetic algoritmhs"*. Tesis doctoral, Department of Computer Science. Universidad de Pittsburg, 1980.

[Srini95] K. Srinivasan, D. Fisher. *Machine Learning Approaches to Estimating Software Development Effort*. IEEE Transaction on Software Engineering 21(2): 126-137 (1995)

[Stein00] M. Steinbach, G. Karypis, V. Kumar. *A comparision of document clustering techniques*. In KDD Workshop on text mining, 2000

[Subra93] G.H. Subramanian, S.T. Breslawski. *Dimensionality Reduction in Software Development Effort Estimation.* The Journal of Systems and Software, V 21, May 1993, 187-196.

[Sugen93] M. Sugeno, T. Yasukawa. *A Fuzzy-Logic Based approach to qualitative Modeling*. IEEE Transactions on Fuzzy Systems, Vol.1.Pp.: 7-31, 1993.

[Taff91] L.M. Taff, J.W. Borchering, R.W. Hudgins. *Estimeetings: Development Estimates and a Front-End Process for a Large Project*. IEEE transactions on Software Engineering, Vol 17, nº8, pp. 839-849. 1991.

[Trafa99] T.B. Trafalis. *Primal-dual optimization methods in neural networks and support vector machines training*. Thechnical report, School of industrial engineering, Univ. of Oklahoma, Norman, USA 1999.

[Togai86] M. Togai, H. Watanabe. *A fuzzy inference engine on a VLSI chip: design and implementation*. nº3, pp.106-114. 1986

[Tong85] R.M. Tong. *An annotated Bibliogaphy of fuzzy control*. Industrial Applications of fuzzy control. M. Sugeno ed. North Holland, Elsevier Science, Amsterdam, pp.249-269. 1985

[Trill80] E. Trillas. *Subconjuntos borrosos*. Vicens Universidad, 1980.

[Trill84] E. Trillas, L. Valverde. *On implicattion and indistinguishability in the setting of fuzzy logic.* Management decision support systems using fuzzy sets and possibility theory. Verlag TUV Rheinland, pp.198-212. 1984

[Trill85] E. Trillas, L. Valverde. *On mode and mplication.* Approximate reasoning in expert systems. M.Gupta, A.Kandel, W. Bandler Eds., pp.157-166. 1985

[Vande83] K.J. van der Poel, S.R. Schach. *A Software Metric for Cost Estimation and Efficiency Measurement in Data Processing System Development*. J. Syst. and Software 3 (1983), 187-191.

[Ventu93] G. Venturini. *SIA: A supervised inductive algorithm with genetic search for learning attributes based concepts*. In Proceedings of European Conference on machine learning, pp. 281-296. 1993

[Walts77] C.E. Walston, C.P. Felix. *A Method for Programming Measurement and Estimation.* IBM Systems Journal, No.16, 1, 1977, pp. 54-73

[Weiss89] S.M. Weiss, I. Kapouleas. *An empirical comparison of pattern recognition, neural nets and machine learning classification methods*. In internantional Joint Conference on Artificial Intelligence, 1989.

[Weiss91] S.M. Weiss, C.A. Kulikowski. *Computer systems that learn*. Morgan Kaufmann Publishers, Inc., San Francisco, 1991.

[Wetts95] C. Wettschereck. *A study of distance-based machine learning algorithms.* PhD thesis, Oregon State University, 1995

[Yager92] R. Yager, L.A. Zadeh. *An introduction to fuzzy logic applications in intelligent systems*. Kluwer academic publs., Boston. 1992

[Yamak86] T. Yamakawa. *High speed fuzzy controller hardware system.* pp. 16-18. 1986

[Zadeh65] L.A. Zadeh. *Fuzzy Sets*. Information and control, nº83, pp. 228-353, 1965.

[Zadeh72] L.A. Zadeh. *A Rationale of fuzzy control*. Journal of Dynamic Systems management and control. Nº94-G..1972.

[Zadeh73] L.A. Zadeh. *Outline of a new approach to the analysis of complex systems and decision processes*. IEEE transaction on systems man and cybernetics, SSMC-3. pp 28-44, 1973.

[Zadeh78] L.A. Zadeh. *Fuzzy sets as a basis for a theory of possibility*. Fuzzy sets and Systems. Vol1.pp 3-28, 1978.

[Zhao01] Y. Zhao, G. Karypis. *Criterion functions for document clustering: experiments and analysis*. Technical Report TR 01-40, Department. of Computer Sciences, University of Minnesota. 2001. Available at *http://cs.umn.edu/~karypis/publications*

[Zimme85] H.J. Zimmermann. *Fuzzy sets theory and applications*. Kluwer academic publish, Dordrecht, 1985.