REVISTA COLOMBIANA DE COMPUTACIÓN
Volumen 3, número 1
Págs. 53-71

Obtención y Validación de Modelos de Estimación de Software Mediante Técnicas de Minería de Datos

María N. Moreno García[*] , Luis A. Miguel Quintales[*], Francisco J. García Peñalvo[*] y M. José Polo Martín[*]

Resumen

La medición del software está adquiriendo una gran importancia debido a que cada vez se hace más patente la necesidad de obtener datos objetivos que permitan evaluar, predecir y mejorar la calidad del software así como el tiempo y coste de desarrollo del mismo. El valor de las mediciones aumenta cuando se realiza sobre modelos construidos en las primeras fases del proyecto ya que los resultados obtenidos permiten tenerlo bajo control en todo momento y corregir a tiempo posibles desviaciones. La proliferación actual de métricas y el gran volumen de datos que se maneja ha puesto de manifiesto que las técnicas clásicas de análisis de datos son insuficientes para lograr los objetivos perseguidos. En este trabajo se presenta la forma en que pueden aplicarse las nuevas técnicas de minería de datos en la construcción y validación de modelos de ingeniería del software, cambiando el análisis tradicional de datos *dirigido a la verificación* por un enfoque de análisis de datos *dirigido al descubrimiento del conocimiento*.

Palabras claves: *Minería de datos, métricas, estimación de software.*

Abstract

Based upon the necessity of achieving objective data that allow evaluation, forecast and improvement of software quality as well as the time and cost of the project, software measurement is strongly increasing in importance. Metrics are particularly useful when they are used upon early-stages software models, since the obtained results allow an accurate control of the project across of the development and, additionally, to solve possible deviations immediately. At present, metrics proliferation and the huge volume of data needed have displayed that the classic data analysis techniques are not sufficient in order to obtain the desired objectives. In the present study the way in which the new techniques of data mining may be used on the construction and validation of software engineering models is shown, by means of a drastic change of the traditional verification oriented data analysis in an approach focused on the knowledge discovery.

Keywords*: Data mining, metrics, software estimation.*

[*] Universidad de Salamanca, Departamento de Informática y Automática, Plaza de la Merced S/N, 37008, Salamanca, e-mail: {mmg, lamq, fgarcia, mjpolo}@usal.es

1 Introducción

Las primeras etapas del desarrollo de software son cruciales en la consecución de productos de calidad dentro de los límites de tiempo y coste establecidos para un proyecto. Los errores introducidos en dichas etapas o durante su evolución son causa frecuente de dificultades en el mantenimiento, baja reutilización y comportamiento defectuoso de los programas. Igualmente, las malas estimaciones realizadas al comienzo del proyecto tienen consecuencias desastrosas en cuanto a costes y plazos de entrega. Estas son las principales causas por las que la medición del software en el ámbito de la especificación de requisitos (ERS) está adquiriendo cada vez mayor importancia, debido a la necesidad de obtener datos objetivos que contribuyan a mejorar la calidad desde las primeras etapas del proyecto.

Los estudios relacionados con la medición en el nivel de la especificación de requisitos se han centrado fundamentalmente en el desarrollo de métricas para determinar el tamaño y la funcionalidad del software. Entre las de mayor difusión se encuentran las métricas de puntos de función [1], métricas Bang [11] o los puntos objeto [4]. La medición de atributos de calidad de las especificaciones de requisitos del software (ERS) ha sido también objeto de algunos trabajos que van desde la medición de especificaciones formales [31] hasta la aplicación de métricas para evaluar la calidad de especificaciones expresadas informalmente en lenguaje natural (métricas de facilidad de comprensión del texto contenido en los documentos [21] o métricas de estructura y organización en documentos convencionales [2] y con hipertexto [14] [30]), pasando por técnicas encaminadas a determinar el cumplimiento de los estándares, directrices, especificaciones y procedimientos, que requieren información procedente de revisiones técnicas, inspecciones**,** *Walkthrough*, o auditorías [10] [6] [13]. La creciente adopción de la tecnología de orientación a objetos en el desarrollo de software ha dado lugar a la aparición de nuevas métricas específicas para este tipo de sistemas [8], [Lorenz y Kidd 1994], [9] [3]. Recientemente se han propuesto métricas para la evaluación de la calidad a partir de modelos producidos en etapas iniciales del ciclo de vida, como son las métricas de calidad y complejidad en modelos OMT [15], métricas de calidad de los diagramas de clases en UML [16] o las técnicas de medición de modelos conceptuales basados en eventos [28].

La proliferación actual de métricas y la necesidad de medir diferentes aspectos del software está contribuyendo a crear confusión sobre las relaciones entre tales medidas, así como sobre su forma y ámbito de aplicación. Este hecho ha abierto una nueva vía en la investigación orientada hacia la propuesta de modelos, arquitecturas y marcos de referencia ("frameworks") que permitan la organización de las medidas y la clasificación de las entidades de software susceptibles de medir [27] [5] [25]. La construcción de modelos sobre diferentes aspectos del software requiere la recolección de numerosos datos procedentes de observaciones empíricas. Los avances tecnológicos actuales posibilitan la rápida obtención de grandes cantidades de datos de fuentes muy diversas, así como el almacenamiento eficiente de los mismos. Dichos datos encierran información muy valiosa que puede tratarse mediante los métodos tradicionales de análisis de datos, sin embargo estos métodos no son capaces de encontrar toda la información útil latente en la gran masa de datos que se maneja. En ese contexto, las técnicas de minería de datos surgen como las mejores herramientas para realizar exploraciones más profundas y extraer información nueva, útil y no trivial que se encuentra oculta en grandes volúmenes de datos.

En este trabajo se muestra la aplicación práctica de técnicas de minería de datos en la construcción y validación de modelos de ingeniería del software que relacionan diferentes atributos de la ERS con el objeto de predecir características del producto final y descubrir patrones y afinidades ocultas entre dichos atributos.

2 Técnicas de minería de datos

La minería de datos ha dado lugar a una paulatina sustitución del análisis de datos *dirigido a la verificación* por un enfoque de análisis de datos *dirigido al descubrimiento del conocimiento*. La principal diferencia entre ambos se encuentra en que en el último se descubre información sin necesidad de formular previamente una hipótesis. La aplicación automatizada de algoritmos de minería de datos permite detectar fácilmente patrones en los datos, razón por la cual esta técnica es mucho más eficiente que el análisis dirigido a la verificación cuando se intenta explorar datos procedentes de repositorios de gran tamaño y complejidad elevada. Dichas técnicas emergentes se encuentran en continua evolución como resultado de la colaboración entre campos de investigación tales como bases de datos, reconocimiento de patrones, inteligencia artificial, sistemas expertos, estadística, visualización, recuperación de información, y computación de altas prestaciones.

Los algoritmos de minería de datos se clasifican en dos grandes categorías: supervisados o predictivos y no supervisados o de descubrimiento del conocimiento [36].

Los algoritmos **supervisados o predictivos** predicen el valor de un atributo (*etiqueta*) de un conjunto de datos, conocidos otros atributos (*atributos descriptivos*). A partir de datos cuya etiqueta se conoce se induce una relación entre dicha etiqueta y otra serie de atributos. Esas relaciones sirven para realizar la predicción en datos cuya etiqueta es desconocida. Esta forma de trabajar se conoce como *aprendizaje supervisado* y se desarrolla en dos fases: entrenamiento (construcción de un modelo usando un subconjunto de datos con etiqueta conocida) y prueba (prueba del modelo sobre el resto de los datos).

Cuando una aplicación no es lo suficientemente madura no tiene el potencial necesario para una solución predictiva, en ese caso hay que recurrir a los métodos **no supervisados o de descubrimiento del conocimiento** que descubren patrones y tendencias en los datos actuales (no utilizan datos históricos). El descubrimiento de esa información sirve para llevar a cabo acciones y obtener un beneficio (científico o de negocio) de ellas. En la tabla siguiente se muestran algunas de las técnicas de minería de ambas categorías.

Supervisados	**No supervisados**
Árboles de decisión	Detección de desviaciones
Inducción neuronal	Segmentación
Regresión	Agrupamiento ("clustering")
Series temporales	Reglas de asociación
	Patrones secuenciales

Tabla 1. Clasificación de las técnicas de minería de datos

La aplicación de los algoritmos de minería de datos requiere la realización de una serie de actividades previas encaminadas a preparar los datos de entrada debido a que, en muchas ocasiones dichos datos proceden de fuentes heterogéneas, no tienen el formato adecuado o contienen ruido. Por otra parte, es necesario interpretar y evaluar los resultados obtenidos. El proceso completo consta de las siguientes etapas [7]:

1. **Determinación de objetivos**
2. **Preparación de datos**
   - Selección: Identificación de las fuentes de información externas e internas y selección del subconjunto de datos necesario.

- Preprocesamiento: estudio de la calidad de los datos y determinación de las operaciones de minería que se pueden realizar.

3. **Transformación de datos**: conversión de datos en un modelo analítico.
4. **Minería de datos**: tratamiento automatizado de los datos seleccionados con una combinación apropiada de algoritmos.
5. **Análisis de resultados**: interpretación de los resultados obtenidos en la etapa anterior, generalmente con la ayuda de una técnica de visualización.
6. **Asimilación de conocimiento**: aplicación del conocimiento descubierto.

Aunque los pasos anteriores se realizan en el orden en que aparecen, el proceso es altamente iterativo, estableciéndose retroalimentación entre los mismos. Además, no todos los pasos requieren el mismo esfuerzo, generalmente la etapa de preprocesamiento es la más costosa ya que representa aproximadamente el 60 % del esfuerzo total, mientras que la etapa de minería sólo representa el 10%.

3 Aplicaciones de la minería de datos en la medición del software

Las técnicas de minería de datos se están utilizando desde hace varios años para la obtención de patrones en los datos y para la extracción de información valiosa en el campo de la Ingeniería del Software [24]. Entre estas aplicaciones podemos citar la utilización de árboles de decisión en la construcción de modelos de clasificación de diferentes características del desarrollo de software [18] [29] [34], la aplicación de técnicas de "*clustering*" en la planificación del mantenimiento [20] y en la estimación de la fiabilidad del software [26] o el uso de redes neuronales en la predicción de riesgos de mantenimiento en módulos de programa [19]. La mayor parte de los trabajos realizados están dirigidos a la obtención de modelos de estimación de esfuerzo de desarrollo [32] y modelos de predicción de diferentes aspectos de la calidad del software [17]. En ambos casos, las métricas tanto de productos como de procesos juegan un papel importante, constituyendo la base para la construcción de los modelos y posterior validación de los mismos. En publicaciones recientes aparece la introducción de algoritmos de minería en la realización de validaciones de modelos obtenidos mediante otras técnicas. En estos trabajos se comprueba la validez de modelos de estimación mediante métodos de regresión, redes neuronales, algoritmos genéticos, etc. [12], se validan métricas, e incluso "*frameworks*" de medición [23].

4 Ejemplos de aplicación

En este apartado se muestran algunos ejemplos de aplicación de las técnicas mencionadas en la construcción de modelos de estimación del tamaño de un proyecto. Se ha utilizado la herramienta MineSet de Silicon Graphics sobre un conjunto de datos procedentes de experimentos llevados a cabo por Dolado y descritos en [12].

4.1 Estudio previo de los datos utilizados

Se dispone de datos referentes a 42 proyectos implementados con un lenguaje de cuarta generación (Informix-4GL). Los sistemas desarrollados son aplicaciones de contabilidad con las características de sistemas comerciales, cada uno incluye alguno de los siguientes subsistemas.

Compras, ventas, inventario, finanzas y ciclos de producción. Esta información se ha dividido en dos grupos, en el primero se ha realizado la descomposición del proyecto en módulos o componentes de tres tipos diferentes siguiendo las pautas de Verner y Tate [35] y se han obtenido atributos de cada uno de los módulos. El segundo grupo contiene datos globales de cada uno de los proyectos. En el primer caso contamos con 1537 registros con datos referentes a los módulos, mientras que en el segundo caso se trabaja con 42 registros con información correspondiente a los proyectos estudiados.

Descripción de los atributos de los módulos:

TYPECOMP: Tipo de componente (1: menú, 2: entrada, 3: informe/consulta)
OPTMENU: Número de opciones (sólo para componentes de tipo 1)
DATAELEMEN: Número de elementos de datos (sólo para componentes de tipo 2 y 3)
RELATION: Número de relaciones (sólo para componentes de tipo 2 y 3)
LOC: Número de líneas de código del módulo

Con los datos de este fichero se ha realizado un estudio estadístico de cada uno de los atributos de los módulos que componen el proyecto. En las figuras 1, 2 y 3 aparece la distribución de valores de atributos para cada uno de los tipos de componente. Se puede observar en todos los casos que la mayor parte de los módulos estudiados no son de gran tamaño, siendo los más pequeños los componentes de tipo menú. En el caso de los componentes de tipo 2 y 3 el número de elementos de datos y el número de relaciones tampoco es muy elevado, no obstante, los valores altos de los atributos tienen suficiente peso para influir en la clasificación.

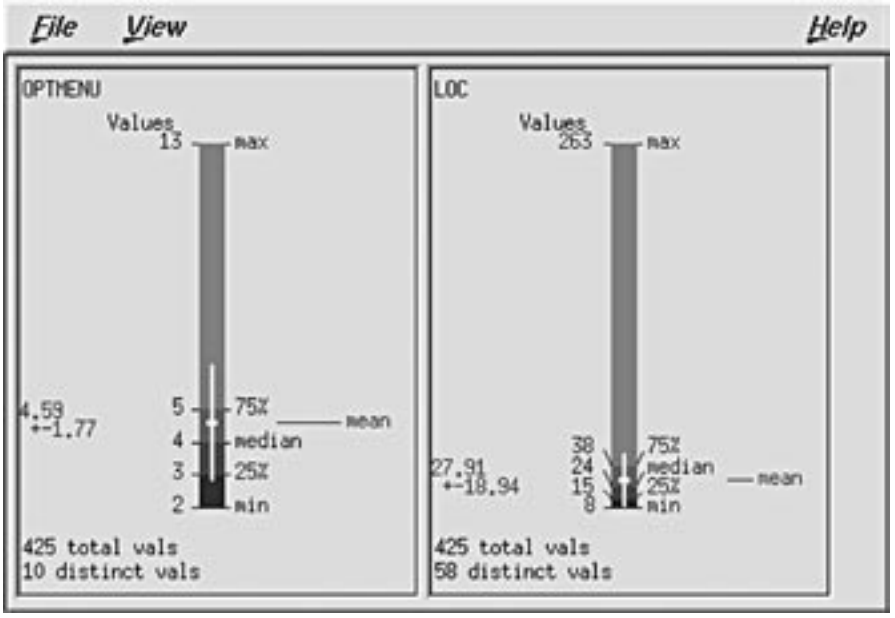

Figura 1. Estadísticas para componentes de tipo 1 (menú)

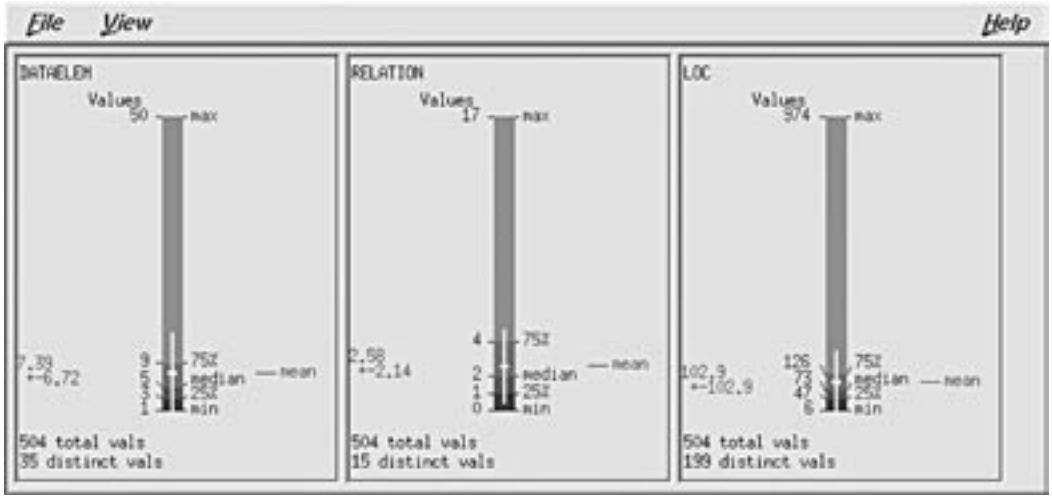

Figura 2. Estadísticas para componentes de tipo 2 (entradas)

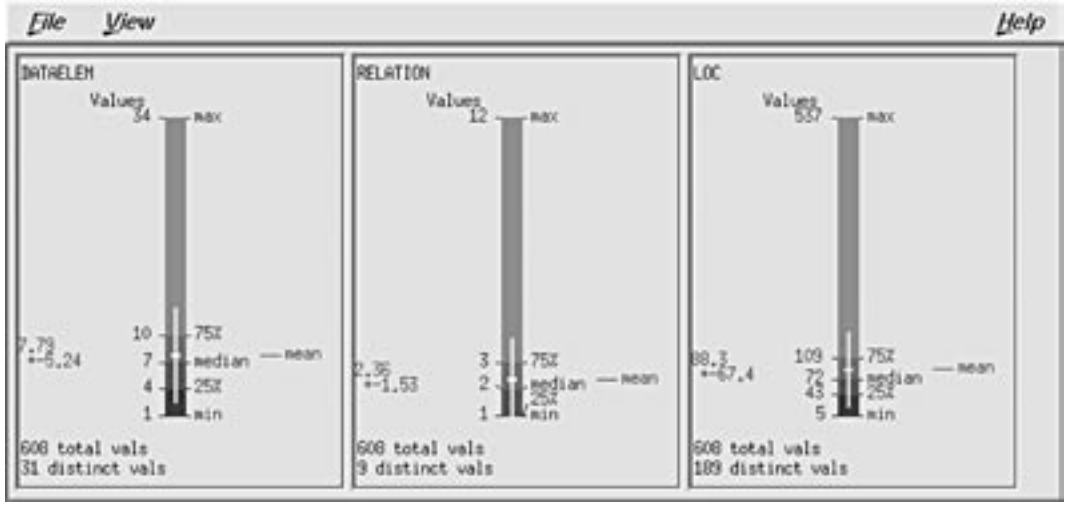

Figura 3. Estadísticas para componentes de tipo 3 (informes/consultas)

Descripción de los atributos de proyectos:

Los atributos que se describen a continuación son los utilizados en el método Mark II [33] para el cálculo de los puntos de función. El método considera el sistema compuesto de

transacciones lógicas. Una transacción lógica es una única combinación entrada/proceso/salida activada por un único evento que tiene significado para el usuario o es el resultado de una consulta o extracción de información. El concepto de entidad sustituye al de fichero lógico.

LOC: Número de líneas de código
NOC: Número de componentes
NTRNSMKII: Número de transacciones MKII
INPTMKII: Número total de entradas
ENTMKII: Número de entidades referenciadas
OUTMKII: Número total de salidas (elementos de datos sobre todas las transacciones)
UFPMKII: Número de puntos de función no ajustados MKII

Para realizar la contabilización de entidades, éstas se dividen en primarias (aquellas para las que se construye el sistema con la intención de recolectar y almacenar datos) y no primarias (las utilizadas como referencia, validación, etc.). Todas las entidades no primarias se engloban en una única denominada entidad del sistema. Se contabiliza el numero de entidades a las que se hace referencia en cada transacción, no el número de referencias.

En cuanto al número de entradas y salidas, se cuentan los tipos de elementos de datos, no ocurrencias (aplicable también a las tablas). No se cuentan elementos de datos de cabeceras o pies de pantalla no generados específicamente para esa pantalla, tampoco se contabilizan etiquetas, cajas, etc. Los mensajes de error se consideran ocurrencias del tipo de dato "mensaje de error" (análogo para los mensajes de operador).

El estudio estadístico realizado sobre los atributos anteriores se refleja en la figura 4, en la que puede observarse que existe mayor proporción de atributos en la zona de valores bajos. El tamaño de los proyectos estudiados se encuentra entre 726 y 8.888 líneas de código.

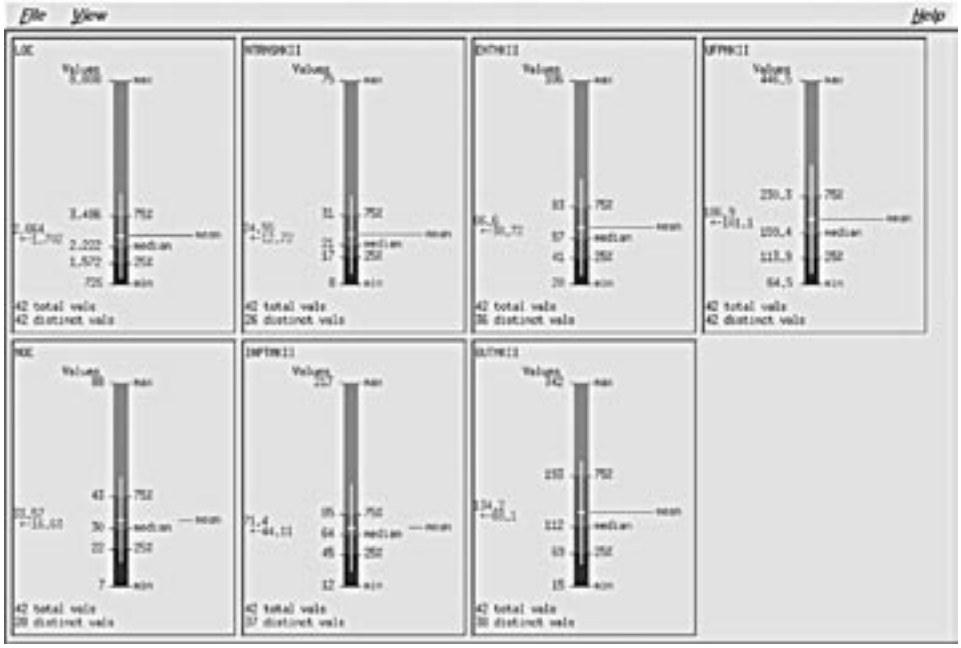

Figura 4. Estadísticas para atributos de proyectos

Además del estudio sobre la distribución de los valores de los atributos, se han realizado las pertinentes transformaciones de los datos, previas a la aplicación de los algoritmos de minería, en los casos en que la técnica a aplicar sobre ellos lo requería.

4.2 Aplicación de las técnicas de minería de datos

Los datos anteriores se han tratado con diferentes algoritmos de minería de datos. La mayor parte de las técnicas se han aplicado sobre los datos de proyectos debido a que se dispone de mayor número de atributos que en el caso de los módulos. Para estos últimos se cuenta con dos atributos (número de opciones y número de líneas de código) para los componentes de tipo menú y tres (número de elementos de datos, número de relaciones y número de líneas de código) para componentes de tipo entrada e informe/consulta. Este reducido número de atributos posibilita el uso técnicas de visualización con las que observar de forma simultánea la influencia de todos los atributos disponibles sobre el tamaño del módulo.

La construcción de los modelos predictivos consiste en la creación de un modelo de clasificación a partir de un conjunto de entrenamiento y de un ***inductor***. Los registros del conjunto de entrenamiento tienen que pertenecer a un pequeño grupo de clases predefinidas, cada clase corresponde a un valor de la etiqueta. El modelo inducido (***clasificador***) consiste en una serie de patrones que son útiles para distinguir las clases. Una vez que se ha inducido el modelo se puede utilizar para predecir automáticamente la clase de otros registros no clasificados (de etiqueta desconocida).

En la inducción de todos los modelos presentados en este trabajo se ha tomado como etiqueta el atributo LOC (número de líneas de código) y se han creado tres clases correspondientes a tres intervalos de valores de dicho atributo: $LOC < 3000$, $3000 \leq LOC \leq 5000$ y $LOC > 5000$. La elección de los intervalos se ha realizado teniendo en cuenta el tamaño de los proyectos estudiados, tanto sus valores mínimo y máximo (726 y 8888 líneas de código respectivamente) como su distribución (figura 4). Por otra parte, debido a que la principal utilidad de la estimación del número de líneas de código es la predicción del esfuerzo de desarrollo, se han seleccionado intervalos de tamaño que produzcan, mediante la aplicación del modelo COCOMO II [4], estimaciones de esfuerzo cuyos intervalos de confianza no se solapen. En todos los casos se han realizado validaciones cruzadas para crear los conjuntos de prueba y entrenamiento, dividiendo de forma iterativa los datos en subconjuntos mutuamente excluyentes del mismo tamaño aproximadamente. Los datos se han dividido de forma aleatoria en diez subconjuntos diferentes, utilizando nueve de ellos para la inducción del modelo y uno para la prueba. El proceso de inducción se repite diez veces, de manera que en cada iteración se elige un subconjunto distinto como conjunto de prueba, utilizando los restantes para el entrenamiento. Posteriormente se han realizado, también de forma aleatoria, particiones diferentes y se ha vuelto a inducir el modelo de forma iterativa.

Una de las técnicas supervisadas más intuitiva es la de los **árboles de decisión**. Los algoritmos utilizados para construir el modelo de clasificación del árbol intentan encontrar valores de los atributos que proporcionen la máxima separación en los datos del conjunto de entrenamiento. En la figura 5 se muestra un árbol de decisión que clasifica los proyectos en diferentes clases en función de los demás atributos disponibles. El atributo que más influye en la clasificación es el de mayor pureza (menor entropía) que aparece en el nodo raíz del árbol. En el ejemplo se puede observar que el número de componentes es la variable que produce la mayor diferenciación en la clasificación, consiguiendo para valores menores o iguales a 33.5 un nodo hoja con un 100% de pureza. El atributo que sirve para separar el resto de los registros no clasificados es el número de transacciones MKII, seguido por el número de entidades referenciadas. Únicamente con estos tres atributos se consigue inducir un árbol de decisión con un error estimado de 9.00 ± 4.47 % , cuyos

nodos hoja tienen una pureza del 100%, exceptuando el nodo señalado por el cursor cuya pureza es del 68.25%. Una de las consecuencias más importantes que pueden obtenerse de este modelo de estimación es su capacidad de predecir atributos del producto final (LOC) a partir de atributos que pueden obtenerse en fases iniciales del desarrollo (NOC, NTRNSMKII y ENTMKII).

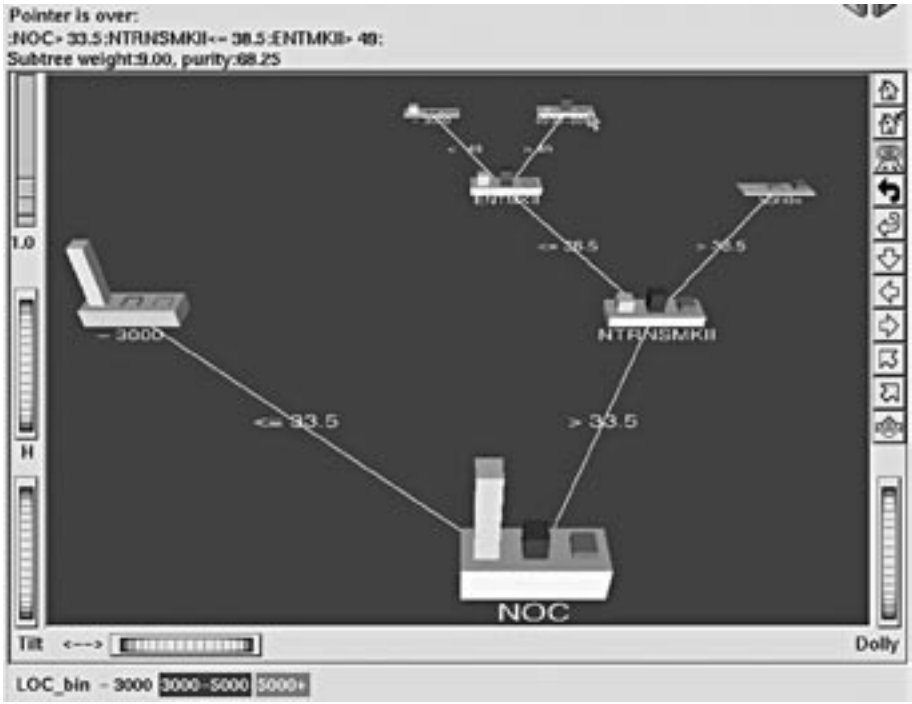

Figura 5. Árbol de decisión

Para realizar la validación del modelo inducido con los datos del conjunto de entrenamiento, aplicamos la técnica de las **matrices de confusión** que muestran el tipo de las predicciones correctas e incorrectas cuando se aplica el modelo sobre el conjunto de prueba (Figura 6). Las predicciones correctas están representadas por las barras que aparecen sobre la diagonal, mientras que el resto de las barras indican el tipo de error cometido (qué valor ha predicho el modelo y cual es el valor verdadero). La altura de las barras es proporcional al peso de los registros que representan.

En el ejemplo que se está tratando no se clasifican correctamente todos los registros, aunque el peso de las clasificaciones incorrectas es muy pequeño y además, éstas se encuentran próximas a la diagonal, lo que indica que el error cometido no es demasiado elevado. Hay que señalar que se dispone de muy pocos datos. Los resultados mejorarían sensiblemente si se contase con datos correspondientes a un mayor número de proyectos.

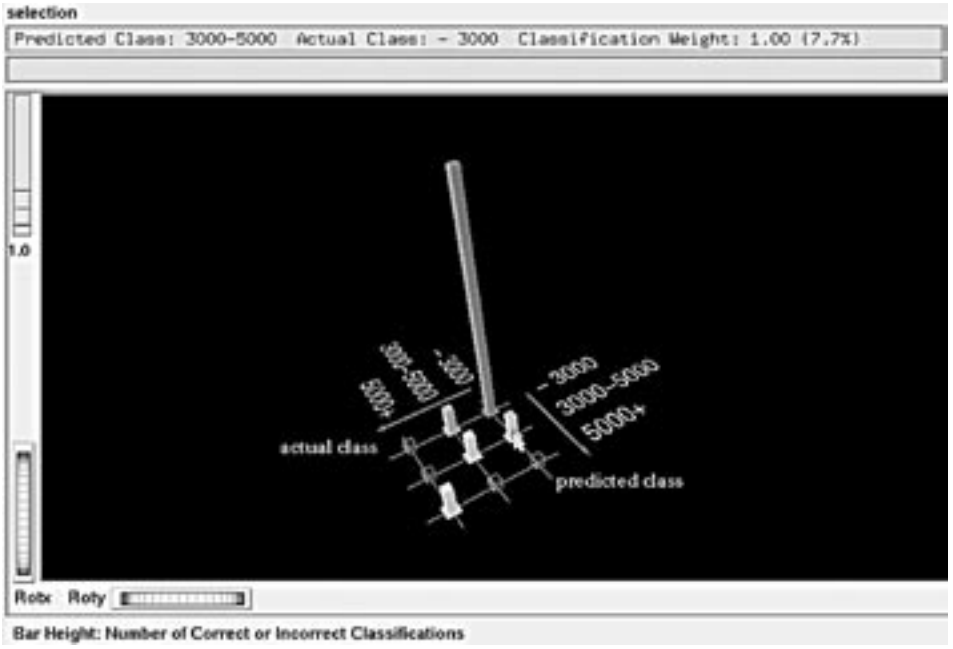

Figura 6. Matriz de confusión

Las **tablas de decisión** son otra herramienta de clasificación mediante la cual se muestran correlaciones entre pares de atributos a diferentes niveles. Los atributos continuos se separan en intervalos discretos y se induce automáticamente la tabla mediante cálculos de probabilidades a partir de los registros del conjunto de entrenamiento. Sobre los ejes de la tabla se representa el número de intervalos discretos de valores para cada par de atributos. En el interior de la tabla aparecen bloques o celdas que muestran la distribución de probabilidad para cada combinación de valores de los atributos. La proporción de colores en cada bloque indica la proporción de registros de cada una de las clases. Un buen clasificador debería conseguir el mayor número posible de celdas de un sólo color, que contendrían únicamente registros de una clase. Haciendo uso de otros atributos se puede acceder a nuevos niveles de descomposición para cada uno de los bloques. El nivel superior corresponde a los atributos que proporcionan la mejor clasificación.

En la tabla de decisión del ejemplo propuesto (figura 7) aparecen tres niveles, en el primero, con los atributos número de componentes y número de puntos de función, ya se ha conseguido clasificar la mayor parte de los registros, puesto que sólo existe una celda con registros de dos clases. Dicha celda se ha descompuesto en un nivel inferior haciendo uso de los atributos número de salidas y número de entidades referenciadas con lo que conseguimos afinar más la clasificación, pero aún queda una pequeña proporción de registros sin clasificar. En el tercer nivel la proporción se reduce con el atributo número de transacciones MKII aunque tampoco se logra discernir totalmente dos de las clases.

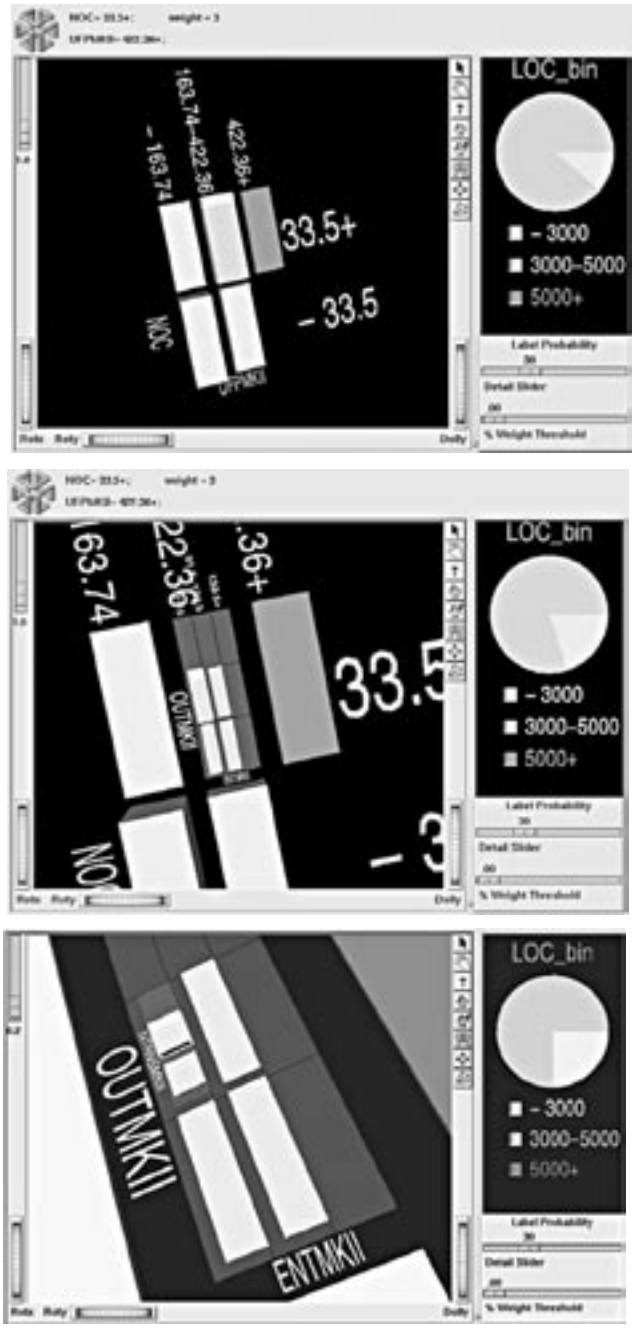

Figura 7. Tabla de decisión

Con esta técnica se ha obtenido un error de 12.75 ± 5.21 %, mayor que con el árbol de decisión pese a que se han utilizado más atributos. No obstante, de su aplicación puede deducirse que el número de puntos de función propuestos en el método Mark II son buenos indicadores del tamaño del sistema en número de líneas de código, además vuelve a aparecer el atributo número de componentes como uno de los más influyentes en la clasificación.

Otro método útil para conocer la importancia de los valores de un atributo específico para la clasificación o la probabilidad de que un registro pertenezca a una clase cuando se conocen pocos atributos de dicho registro es el **clasificador de evidencias** que muestra la distribución de registros por valores de los atributos (figura 8). Las probabilidades de pertenencia a una clase para cada intervalo de valores de un atributo se realiza con independencia de los demás. Del análisis de esta última experiencia se puede concluir que los atributos que de forma aislada sirven mejor para diferenciar las clases son UFPMKII y NTRNSMKII y el único para el que no se encuentra relación entre sus valores y el número de líneas de código es INPTMKII. Esto concuerda con los modelos inducidos anteriormente, en los que este último atributo no aparece. Otra observación importante es la influencia del número de componentes (NOC), con el que se consigue aislar en una sola clase (LOC < 3000) a todos los registros con valores inferiores a 33.5, pero no se obtienen otros intervalos que clasifiquen los registros restantes. Tanto en los árboles como en las tablas de decisión dicho atributo aparece en el nivel de influencia más alto, lo que indica, que su combinación con otros como UFPMKII o NTRNSMKII es muy efectiva en la estimación del tamaño del software.

Figura 8. Clasificador de evidencias

Los ejemplos anteriores se encuadran en la categoría de técnicas predictivas o supervisadas. Otro grupo de técnicas son las no supervisadas o de descubrimiento del conocimiento, entre las que se encuentra el análisis de asociación, que persigue el establecimiento de relaciones entre registros individuales o grupos de registros de la base de datos. Dos especializaciones del análisis de asociación son las reglas de asociación y el descubrimiento de patrones secuenciales. Una regla de asociación tiene la forma "**Si X entonces Y**", donde:

- ***Cuerpo de la regla***: X (LHS)
- ***Cabeza de la regla***: Y (RHS)
- ***Factor de confianza*** *o* ***previsibilidad***: frecuencia con que tiene lugar la regla en relación con las veces que ocurre X.
- ***Factor de soporte*** *o* ***prevalencia***: relación entre el número de veces que tiene lugar la regla y el número total de transacciones.
- ***Previsibilidad esperada***: número de veces que ocurre Y.

En la figura 9 se muestra la visualización de los conceptos anteriores relativos a las reglas de asociación que relacionan los atributos de los proyectos que se están estudiando. En los ejes se representan los valores de los atributos correspondientes al cuerpo (LHS) y a la cabeza de la regla (RHS). La altura de las barras indica la *previsibilidad* de la regla, el disco la *previsibilidad esperada* y el color la división de ambos valores. No se muestran las reglas cuya previsibilidad esperada sea mayor que su previsibilidad, es decir, reglas en que la frecuencia de que ocurra Y sola sea mayor que la frecuencia con que ocurren X e Y juntas (el valor mínimo del cociente entre previsibilidad y previsibilidad esperada es uno). Para la regla señalada con el cursor obtenemos los siguientes datos:

Regla: Si NOC = 55-62 entonces LOC = 2495-35299
Prevalecia: 2,38% (la regla se cumple en el 2,38 % de los proyectos estudiados)
Previsibilidad: 100 % (El 100% de los proyectos con NOC = 55-62 tienen LOC = 2495-35299)
Previsibilidad esperada: 19,05% (Por cada 100 ocurrencias de NOC = 55-62 existen 19,05 ocurrencias de LOC = 2495-35299 para NOC ≠ 55-62)
Previsibilidad /Previsibilidad esperada: 5,25 (Proporción entre las veces que ocurre la regla y el número de veces que ocurre LOC = 2495-35299 para NOC ≠ 55-62)

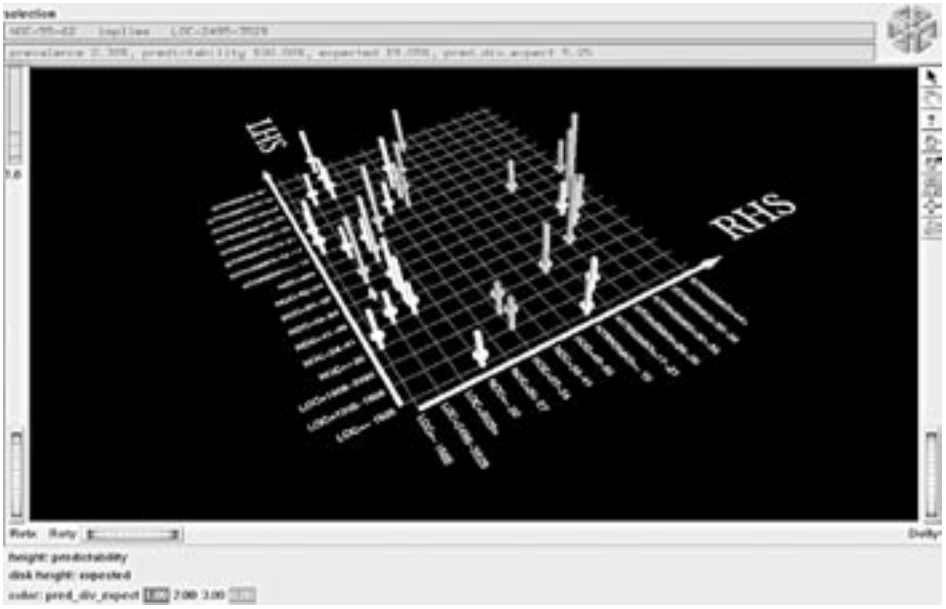

Figura 9. Visualización de reglas de asociación

Las reglas de mayor utilidad serán las representadas por las barras de mayor altura y de color rojo. En nuestro caso dichas reglas muestran la correspondencia entre el número de componentes y el número de transacciones MKII y entre ambos atributos con el número de líneas de código.

La visualización es una de las técnicas más potentes para identificar patrones ocultos en los datos. Con estas técnicas se pueden detectar fenómenos que ocurren en los datos mediante representaciones n-dimensionales de datos sobre pantallas bidimensionales. En la figura 10 podemos observar dos gráficos de dispersión que muestran simultáneamente la influencia de cuatro atributos en el número de líneas de código. Sobre los ejes aparecen representados los valores de tres de los atributos, utilizando la opacidad para representar un cuarto atributo y el color para representar el tamaño del proyecto. En el gráfico de la izquierda se corrobora la escasa relación del atributo número de entradas con el número de líneas de código, mientras que en el de

la derecha se muestra la relación de los atributos más importantes de los modelos de clasificación con el número de líneas de código.

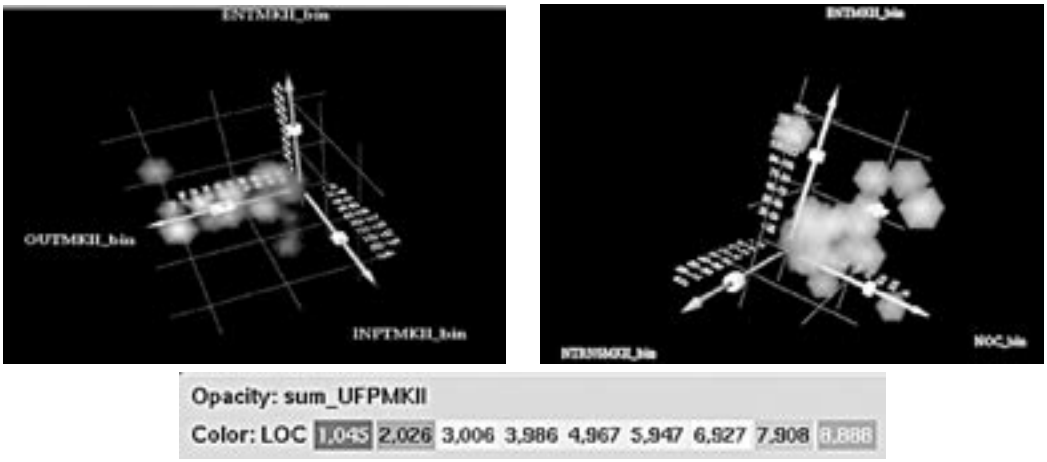

Figura 10. Gráfico de dispersión tridimensional

En la figura 11 se representan con tres colores diferentes, las tres clases tratadas en los modelos predictivos y su relación con los valores de los tres atributos de los ejes y con el número de puntos de función (tamaño de los cubos). Nuevamente se demuestra la importancia de dichos atributos en la clasificación y la correspondencia entre el número de líneas de código y el número de puntos de función. Mediante esta técnica de visualización también se pueden observar desviaciones de los registros del comportamiento esperado, es el caso del cubo azul señalado por el cursor, que pertenece a la clase de menor tamaño, y sin embargo se encuentra muy alejado del origen de coordenadas.

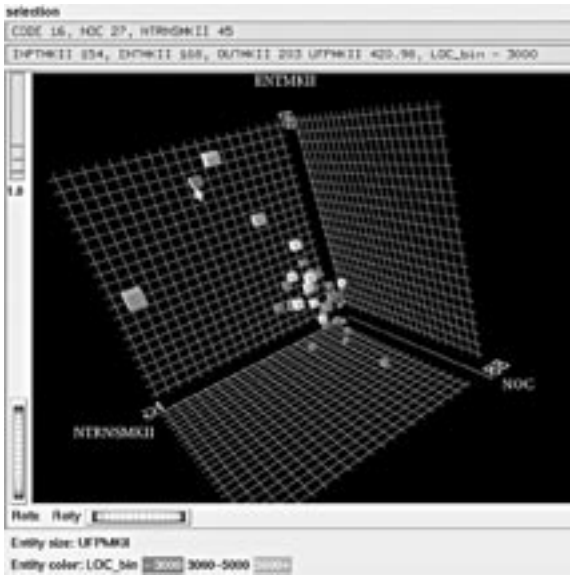

Figura 11. Visualización multivariable

Seguidamente se muestra un gráfico de dispersión (figura 12) que relaciona los atributos correspondientes a un tipo de módulos descrito en el apartado 4.1. En él se puede observar que los valores de los atributos número de elementos de datos y número de relaciones tienen una clara influencia en el número de líneas de código.

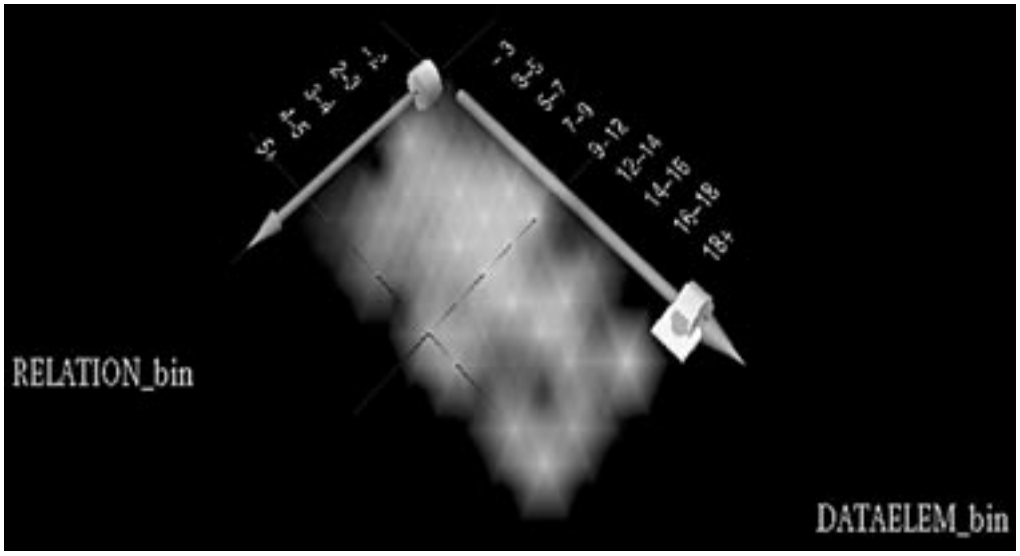

Figura 12. Gráfico de dispersión para componentes de tipo 3 (informes/consultas)

Las herramientas de visualización más modernas no sólo ofrecen gráficos estáticos como los anteriores, sino que también incorporan utilidades de animación. Estas características permiten visualizar el movimiento provocado por el cambio en el comportamiento de los registros de datos a medida que van cambiando determinadas variables seleccionadas para la animación. En la figura 13 se muestran tres momentos de la animación de un gráfico de dispersión en la que se ve como van variando los valores de los atributos LOC, INTMKII, OUTMKII, ENTMKII a medida que aumenta el número de puntos de función, variable que controla la animación y que está representada por el deslizador de la derecha. El aumento en el número de puntos de función va produciendo un alejamiento de los registros del origen de coordenadas, a la vez que van apareciendo proyectos de mayor tamaño representados por los cambios de color desde el azul hacia el rojo.

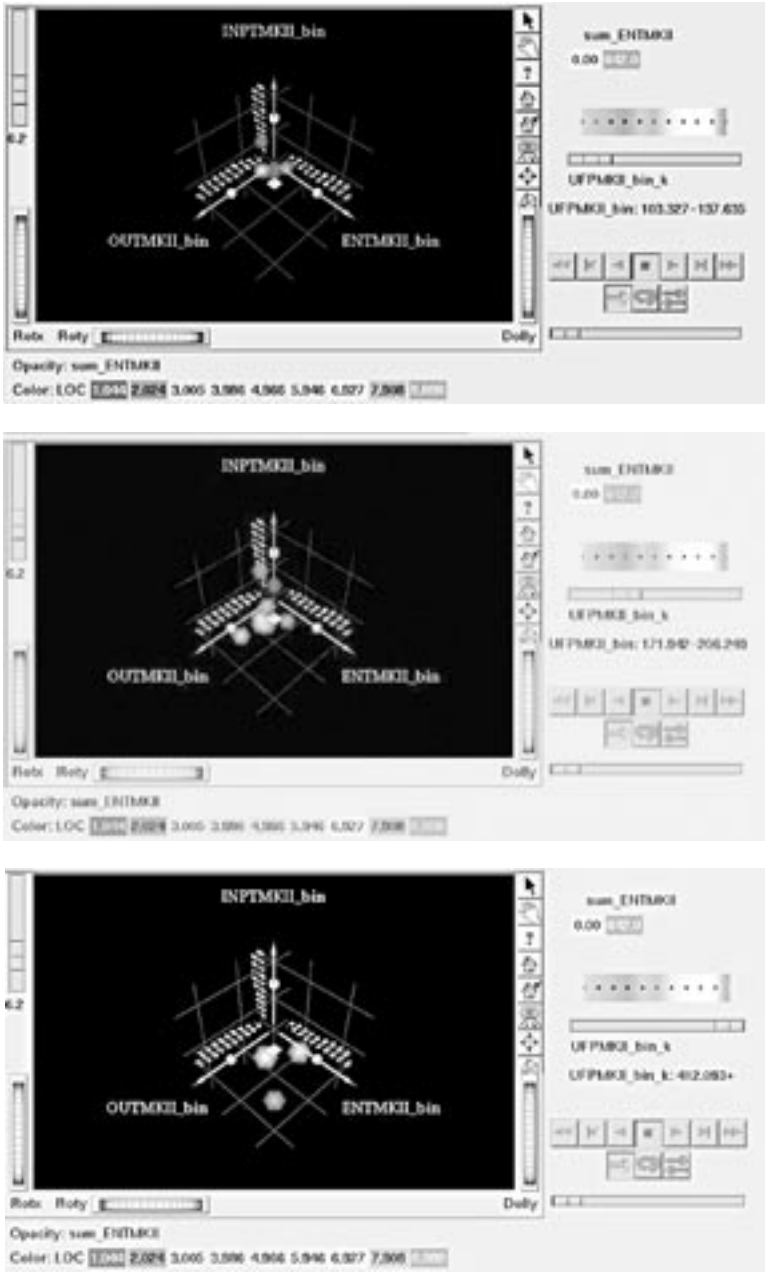

Figura 13. Animación de un gráfico de dispersión

5 Conclusiones

En el apartado anterior se han examinado los resultados de la aplicación de varias técnicas de minería de datos en la construcción de modelos predictivos y asociativos con datos obtenidos en etapas iniciales del proyecto, no obstante las posibilidades que ofrece este nuevo enfoque de

tratamiento de datos son mucho mayores, ya que el número de técnicas que engloba es mucho más amplio. Por otra parte, los métodos de minería de datos llevan asociados una serie de mecanismos (estimación de errores, matrices de confusión, matrices de pérdida, curvas de esfuerzo y aprendizaje, análisis sensitivo de entradas...) que permiten realizar una mejor validación empírica de los modelos y un análisis de resultados más completo y fiable que el que ofrece el enfoque clásico.

Las experiencias realizadas han servido para revelar qué atributos de los modelos realizados en las primeras fases del proyecto, o qué combinaciones de atributos, contribuyen en mayor medida a predecir el tamaño del producto final. Se han utilizado dos tipos de técnicas: predictivas y de descubrimiento del conocimiento.

La aplicación de técnicas predictivas ha puesto de manifiesto que es posible obtener fácilmente modelos de estimación del tamaño del software a partir de un numero reducido de atributos procedentes de mediciones realizadas en el ámbito de la especificación de requisitos. Los modelos se construyen utilizando datos históricos de proyectos terminados y posteriormente se usan para realizar predicciones de tamaño de proyectos en curso. En este estudio se ha encontrado que el atributo NOC (número de componentes), establecido en el método de Verner y Tate [35], es uno de los más influyentes en la determinación del número de líneas de código. El resto de los atributos usados en la estimación provienen del método MarK II [33]. La técnica predictiva que ha producido mejores resultados ha sido la de *árboles de decisión*, ya que es la que presenta menores errores en la fase de prueba y la que requiere menor número de atributos para la clasificación; únicamente conociendo el número de componentes, el número de transacciones y el número de entidades referenciadas se puede realizar la estimación. No obstante, las otras dos técnicas aplicadas (*tablas de decisión* y *clasificador de evidencias*) nos han servido para corroborar la importancia del atributo número de componentes y para demostrar la relación existente entre el número de puntos de función del método Mark II y el número de líneas de código, lo que indica que el cálculo de los mismos constituye una buena base para la estimación del esfuerzo y coste del proyecto.

Con las técnicas de descubrimiento del conocimiento, como las *reglas de asociación* o la *visualización*, se han detectado patrones ocultos en los datos. En el caso objeto de este estudio, las asociaciones más fuertes se han encontrado entre el número de componentes y el número de transacciones MKII y entre ambos atributos con el número de líneas de código. El conocimiento extraído de los datos mediante estos métodos constituye una información valiosa que conduce a los gestores de proyectos a una efectiva toma de decisiones respecto a la planificación y asignación de recursos.

Referencias

[1] A.J. Albrecht, Measuring application development, Proc. *IBM Applications Development Joint SHARE/GUIDE Symposium*, Monterey, CA, 83-92, 1979.

[2] J.D. Arthur y K.T. Stevens, Assessing the adequacy of documentation through document quality indicators, Proc. *the IEEE Conference of Software Maintenance*, 40-49, 1989.

[3] R. Binder, Testing object-oriented systems, *American Programmer*, 7(4): 22-29, 1994.

[4] B.W. Boehm; B. Clark; E. Horowitz et al., Cost models for future life cycle processes: COCOMO 2.0, Annals *Software Engineering* 1(1): 1-24, 1995.

[5] L.C. Briand; J.W. Daly y J.K. Wüst, A unified framework for coupling measurement in object-oriented system, *IEEE Transaction on Software Engineering*, 25 (1): 91-121, 1999.

[6] B. Brykczynskki, A survey of software inspection checklist, *ACM Software Engineering Notes*, 24(1): 82-89, 1999.

[7] P. Cabena; P. Hadjinian; R. Stadler; J. Verhees y A. Zanasi, Discovering data mining. from concept to implementation, Prentice Hall, 1998.

[8] S.R. Chidamber y C.F. Kemerer, A Metrics Suite for Object-Oriented Design , *IEEE Transactions of Software Engineering*, 20(6): 476-493, 1994.

[9] N.I. Churcher y M.J. Shepperd, Towards Conceptual Framework for Object-Oriented Metrics, *ACM Software Engineering Notes*, 20 (2): 67-76, 1995.

[10] A. Davis et al., Identifying and measuring quality in a software requirements specification Proc. *First International Software Metrics Symposium*, Baltimore, 141-152, 1993.

[11] T. DeMarco, Controlling software projects, Yourdon Press, 1982.

[12] J.J. Dolado, A validation of the component-based method for software size estimation, *IEEE Transactions on Software Engineering* 26(10): 1006-1021, 2000.

[13] B. Farbey, Software Quality metrics: considerations about requirements and requirements specification, *Information and Software Technology*, 32 (1): 60-64, 1990.

[14] J.C. French; J.C. Knight y A.L. Powell, Applying hipertext structures to software documentation, *Information Processing and Management*, 33 (2): 219-231, 1997.

[15] M. Genero; M.E. Manso; M. Piattini y F.J. García, Assessing the quality and the complexity of OMT models, Proc. *2nd European Software Measurements Conference-FESMA 99*, Amsterdam, Netherlands, 99-109, 1999.

[16], M. Genero; M. Piattini, y C. Calero, Una propuesta para medir la calidad de los diagramas de clases en UML, *IDEAS´2000*, Cancun, México, 373-384, 2000.

[17] T.M. Khoshgoftaar; E.B. Allen; J.P. Hudepohl y S.J. Aud, Neural networks for software quality modeling of a very large telecommunications system, *IEEE Trans. on Neural Networks*, (8)4: 902-909, 1997.

[18] T.M. Khoshgoftaar y E.B. Allen, Modeling Software Quality with Classification Trees. En: *Recent advances in reliability and quality engineering, Hoang Pham Editor.World Scientific, Singapore*, 1999.

[19] T.M. Khoshgoftaar y D.L. Lanning, A neural network approach for early detection of program modules having high risk in the maintenance phase. *J. Systems Software*, 29(1): 85-91, 1995.

[20] U. Krohn y C. Boldyreff, Application of cluster algorithms for batching of proposed software changes, *J. Softw. Maint: Res. Pract*. 11: 151-165. 1999.

[21] F. Lehner, Quality control in software documentation: Measurement of text comprehensibility, *Information and Management*, 25: 133-146, 1993.

[22] M. Lorenz y J. Kidd, Object_oriented Software Metrics, Prentice Hall 1994.

[23] M.G. Mendonça y V.R. Basili, Validation of an approach for improving existing measurement frameworks, *IEEE Transactions on Software Engineering* 26(6): 484-499, 2000.

[24] M.G. Mendonça, y N.L. Sunderhaft, Mining software engineering data: A survey, Technical Report, DoD Data and Analysis Center for Software, DACS-SOAR-99-3, 1999.

[25] M.N. Moreno; F.J. García; M.J. Polo; V. López y A. González, Marco de referencia para la gestión de la calidad de las especificaciones de requisitos, Proc. *QUATIC'2001*, Lisboa, Portugal, 2001.

[26] A. Podgurski; W. Masri; Y. McCleese y F.G. Wolff, Estimation of software reliability by stratified sampling. *ACM Trans.on Soft.Eng.and Methodology*, 8 (3): 263-283, 1999.

[27] G. Poels, Towards a size measurement framework for object-oriented especifications, Proc. *1st European Software Measurement Conference - FESMA'98*, Antwerp, 379-388, 1998.

[28] G. Poels, On the measurements of event-based object-oriented conceptual models, Proc. *4th International ECOOP Workshop on Quantitative Approaches in Object Oriented Software Engineering*, Cannes, France, 2000.

[29] A.A. Porter y R.W. Selby, Empirically guided software development using metric-based classification trees, *IEEE Software* , 7(2): 46-54, 1990.

[30] T. Roth; P. Aiken y S. Hobbs, Hypermedia support for software developmemt: a retrospective assessment, *Hypermedia*, 6 (3): 149-173, 1994.

[31] W.B. Samson; D.G. Nevill y P.I. Dugard, Predictive software metrics based on a formal specification, *Software Engineering Journal*, 5(1), 1990.

[32] K. Srinivasan, y D. Fisher, Machine Learning Approaches to Estimating Software Development Effort, *IEEE Transactions on Software Engineering*, 21(2): 126-137, 1995.

[33] Symons, C.R. "*Software Sizing and Estimating MKII FPA*". John Wiley and Sons, 1991.

[34] J. Tian y J. Palma, Analyzing and improving reliability:A Tree-based Approach, *IEEE Software*, 15(2): 97-104, 1998.

[35] J. Verner y G. Tate, A software size model, *IEEE Transaction of Software Engineering*, 18 (4): 265-278, 1992.

[36] S.M. Weiss y N. Indurkhya, Predictive data mining. A Practical Guide, Morgan Kaufmann Publishers, San Francisco, 1998.