Revista de Procesos y Métricas de las Tecnologías de la Información (RPM) ISSN: 1698-2029
VOL. 1, Nº 3, Diciembre 2004, 21-28
Asociación Española de Sistemas Informáticos (AEMES)

MODELADO DINÁMICO Y APRENDIZAJE AUTOMÁTICO APLICADO A LA GESTIÓN DE PROYECTOS SOFTWARE

HERACLES[1]
Departamento de Lenguajes y Sistemas Informáticos
E.T.S. de Ingeniería Informática
Avda. Reina Mercedes S/N – 41012 Sevilla
isabel.ramos@lsi.us.es

[1] El proyecto HERACLES (CICYT TIC2001-1143-CO3) está formado por el grupo de Gestión de Proyectos Software constituido por (I.Ramos, M. Ruiz, J. Aroba, M. Mejias) y el grupo de Mineria de Datos constituido por (J.C. Riquelme, J. Aguilar, J. Mata, J.L. Álvarez, R. Giraldez, F. Ferrer y R. Ruiz).

Resumen: En los últimos años, se ha producido un avance significativo en la estimación y gestión de proyectos software, con la creación de modelos dinámicos que permiten modelar el comportamiento del proceso de desarrollo. Una de las ventajas más importantes de estos modelos está en poder "experimentar" el efecto que tendrá sobre el proyecto la aplicación de diferentes políticas de gestión. En este trabajo se presentan los resultados obtenidos al conjugar la utilización de técnicas de aprendizaje automático con los modelos dinámicos para proyectos software. Lo anterior permite obtener reglas de gestión para la estimación de los resultados deseados.Palabras Clave: Regresión lineal, k-vecinos más cercanos, estimación de proyectos, predicción, minería de datos.

1. INTRODUCCIÓN

Tradicionalmente, se ha intentado afrontar el conocido problema de la crisis del software desde el punto de vista de la tecnología de desarrollo empleada olvidándose los aspectos relacionados con la estimación y gestión. Es a principios de los 90 cuando se produce un salto significativo en:

• la evaluación y mejora de procesos software con la aparición del modelo CMM (Capability Maturity Model) del SEI (Software Engineering Institute) y las propuesta recogidas en ISO 90001 (Model for Quality Assurance in Design, Development, Production, Installation and Servicing), ISO 9000-3 (Guidelines for the Application of ISO 9000 to the Development, Supply and Maintenance of Software) e ISO/IEC 15504 (SPICE), entre otros.

• y en la estimación y gestión de proyectos de desarrollo de software (en adelante, PDS) con la aparición del primer modelo dinámico [Abdel-Hamid, 1991], que simula el comportamiento de dichos proyectos.

Esta ponencia se centrará, en adelante, en este segundo apartado recogiendo algunos aspectos relacionados con la construcción de modelos dinámicos aplicados a proyectos software y la aplicación de técnicas de aprendizaje automático a los resultados obtenidos con el modelado de proyectos.

Con la aparición de los modelos dinámicos para PDS y de potentes entornos de simulación (Stella, Vensim, iThink, PowerSim, etc.) se ha producido un avance significativo en el ámbito de las herramientas de estimación y gestión de proyectos y en el asesoramiento para el complejo proceso de toma de decisiones. Estas herramientas cuyo núcleo lo constituye un modelo dinámico se denominan, normalmente, Simuladores de Proyectos de Software (en adelante, SPS) y permiten al director de proyecto experimentar con diferentes políticas de gestión sin coste, facilitando la toma de decisiones más adecuada posible. Un SPS permite realizar análisis a priori (¿Qué sucedería si ...?), monitorización (¿Qué está sucediendo? y análisis post-mortem (¿Qué habría sucedido si...?) de proyectos software.

2. CONSTRUCCIÓN DE MODELOS DINÁMICOS

La construcción de modelos dinámicos constituye una metodología formal según la cual se pueden expresar los conocimientos sobre el sistema a

modelar. Además, el proceso de construcción del modelo por sí mismo, obliga a los expertos a tener un elevado conocimiento de cuáles son los atributos claves y como se relacionan entre sí. El carácter formal del modelo dinámico permite a los responsables de proyectos compartir sus puntos de vista de una manera no ambigua y armonizar los modelos mentales del proceso. Es a principio de los 90 cuando se aplica por primera vez al modelado del proceso de desarrollo de software [Abdel-Hamid, 1991].

2.1 Principios básicos

Un sistema dinámico está formalizado matemáticamente mediante un sistema de ecuaciones diferenciales. Este modelo matemático formaliza un conjunto de restricciones entre magnitudes que evolucionan en el tiempo. Las ecuaciones finales son el resultado de una sucesión de refinamientos, durante los cuales el experto va comparando los resultados de la simulación del modelo con el comportamiento observado. De esta manera, el experto intenta que los resultados de la simulación del sistema dinámico concuerden con la realidad, para poder validar las hipótesis implícitas en el modelo y así explicarlas. Sin embargo, incluso al final del proceso de construcción, existe una serie de incertidumbres sobre el sistema, las cuales se refieren sobre todo a los atributos y a la forma de las funciones que intervienen en el sistema de ecuaciones diferenciales. Esto es sobre todo cierto cuando determinados aspectos de la realidad se pueden medir solo cualitativamente.

Una de las premisas en las que se basa el modelado de sistemas dinámicos es que el comportamiento o historia temporal de un sistema, compuesto por elementos o componentes, viene determinado principalmente por su estructura o conjunto de relaciones de influencia entre los elementos o componentes. Esta estructura incluye no sólo a los aspectos físicos, como el número de técnicos, sino que también acoge aspectos más importantes, como las metodologías y los procedimientos, que influyen de una manera predominante en la toma de decisiones en el interior del sistema.

Para desarrollar un modelo dinámico se debe comenzar por la identificación de un problema que se manifiesta dentro de la organización. En este momento se empieza a recoger datos relacionados con el problema a partir de diferentes fuentes y estudios específicos. Una vez disponibles estas medidas, se está en condiciones de construir un modelo formal, que inicialmente toma la forma de un conjunto de diagramas lógicos (causales), que muestran las relaciones de causa-efecto descubiertas. Seguidamente, este diagrama se traduce en un conjunto de ecuaciones matemáticas y se somete a revisiones y a posibles modificaciones. Una vez que se considera que el modelo es correcto, se procede a su simulación empleando para ello el ordenador, dadas las complejas interacciones entre variables que escapan a la capacidad humana. La representación gráfica más empleada para la construcción de un modelo mental es el diagrama causal (Figura 1) donde se recogen [Aracil, 1997]:

- las variables del sistema, cantidades que adquiere diferentes valores con el tiempo,

- las relaciones de influencia, relaciones de causa-efecto entre dos variables. Una unión se representa gráficamente como una flecha que conecta a la variable que representa la causa con la variable que percibe el efecto. Si en la relación de influencia aparece un signo positivo indica que las dos variables implicadas en la relación causal cambian sus valores en el mismo sentido. Si el signo es negativo indica que la variable situada en el origen y la situada en el destino de la flecha cambian sus valores de forma opuesta.

- Los lazos o bucles de realimentación. Consiste en dos o más relaciones de influencia conectadas de tal forma que empezando en cualquier variable, se puede seguir todo el camino de uniones y volver a la misma variable inicial. Todos los lazos de realimentación suelen contener retrasos temporales. Estos retrasos pueden proceder del tiempo que transcurre entre la toma de una decisión hasta que se empiezan a obtener sus consecuencias, o bien, en los retardos derivados de la transmisión de información dentro de la organización.

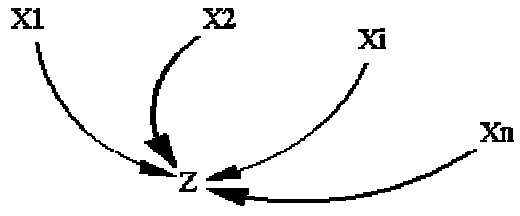


Figura 1: Extracto de un diagrama causal hipotético

Como consecuencia de la polaridad de cada unión, los lazos de realimentación también pueden tener un signo positivo o un signo negativo, dependiendo de la polaridad de las uniones que lo constituyan. Si el número de uniones negativas es impar, el bucle entero es negativo. Si, por el contrario es par, el bucle de realimentación es positivo. Normalmente a los bucles de realimentación positivos se les atribuye el mismo comportamiento que a un sistema físico

con un comportamiento inestable y carente de un controlador (crecimiento o decrecimiento exponencial). Por otro lado, se identifica el comportamiento de los lazos de realimentación negativos con el de los sistemas controlados o regulados.

Consideremos que las variables X1, X2,...Xi...Xn son variables que mantienen una relación de influencia con Z, tal y como aparece en la figura 1, y que se producen retrasos (su significado se verá a continuación) en la transmisión de información (o de material) en las variables X2 y Xn. Una propuesta para obtener la fórmula de la variable Z (hipótesis multiplicativa), suponiendo que ésta sea una variable auxiliar (información que emplean las variables de flujo para tomar sus decisiones) o de flujo (constituyen acciones o decisiones que cambian en el tiempo y que se realizan basándose en una función de decisión) es la siguiente:

$$Z = Z_n * f_1(X_1 / X_{1n}) * f_2(X_{r2} / X_{2n}) * .....* f_n(X_{rn} / X_{nn})$$

donde Zn es un valor normal de Z y X1n, X2n, ..., Xnn son valores normales de X1n, Xr2, ... ,Xrn respectivamente. En las funciones f1, f2, ..., fn se recoge el efecto de cada una de las variables que influyen en Z.

Si Z es una variable de nivel (acumulación, o integración, a lo largo del tiempo de flujos o cambios que entran y salen del mismo), la propuesta para obtener la fórmula de la variable Z (hipótesis aditiva), sería la siguiente:

$$dZ/dt = X_1 \oplus X_{r2} \oplus ..... \oplus X_{rn}$$

$$X_{r2} = retraso(X_2, tr_2) \quad y \quad X_{rn} = retraso(X_n, tr_n)$$

donde Xr2 y Xrn son retrasos y tr2 y trn son, respectivamente, los tiempos de ajuste de dichos retrasos. El signo "⊕ "corresponderá al signo "+" o el signo "-" dependiendo de que la relación de influencia sea de signo positivo o negativo.

2.2 Diagrama causal básico de un proyecto

En la Figura 2 se recoge un diagrama causal en el que aparecen los bucles de realimentación básicos que determinan el comportamiento de un gran número de PDS [Ruiz, 2001]. En la Figura 2, aparecen cuatro bucles de realimentación (dos positivos y dos negativos):

• Bucle 1: Regula el nivel de los recursos humanos asignados al proyecto, en función del progreso percibido en el mismo.

• Bucle 2: Recoge los efectos que producen la presión de plazo (o de coste) y los errores producidos por el personal sobre el progreso del proyecto.

• Bucle 3: Recoge la incidencia que tiene la contratación de personal nuevo sobre la productividad del equipo.

• Bucle4: Recoge los efectos que tiene sobre la productividad el hecho de que aparezca presión del plazo (o de coste) en el proyecto.

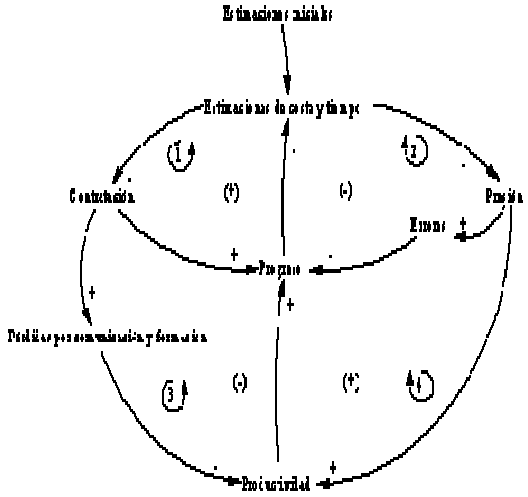


Figura 2: Diagrama Causal Básico para PDS

Las variables y relaciones causa-efecto del diagrama causal básico que recoge el comportamiento de un proyecto se van ampliando mediante un proceso incremental hasta finalizar el proceso de modelado.

3. APRENDIZAJE AUTOMÁTICO Y MODELOS DINÁMICOS

En este apartado se mostrará cómo es posible combinar las técnicas de aprendizaje automático con los simuladores de proyectos software, basados en modelos dinámicos, para la obtención de información útil para la toma de decisiones. Se mostrarán ejemplos obtenidos al aplicar técnicas de aprendizaje automático a las bases de datos simuladas con modelos dinámicos.

3.1 Aprendizaje automático

El verdadero valor de la obtención de información radica en la posibilidad de extraer información útil

para el soporte de decisiones o la exploración y compresión del fenómeno del que provienen los datos. Tradicionalmente, el análisis de datos ha sido un proceso manual, en el que los analistas, con la ayuda de técnicas estadísticas, proporcionaban resúmenes y generaban informes. Este proceso se rompe al crecer la cantidad de datos y las dimensiones de éstos; y con la necesidad de obtener conocimiento útil con rapidez.

Al proceso de análisis o aprendizaje automático de datos se le conoce con la denominación inglesa de data mining. Hay autores que proponen un concepto más global como Knowledge Discovery in Databases (KDD) del que el data mining constituiría un paso particular. Así otras tareas asignadas al concepto de KDD serían la selección de los datos, su preparación y limpieza, la incorporación de conocimiento previo, y la interpretación y evaluación de los resultados. El concepto de KDD agrupa investigadores de campos tales como bases de datos, reconocimiento de patrones, estadística, inteligencia artificial, visión, etc. El concepto de data mining se refiere a la búsqueda de modelos adecuados o patrones determinantes de los datos de entrada. Los objetivos de un algoritmo de data mining son conseguir una predicción, una descripción o una combinación de ambas. Un algoritmo puramente predictivo se centra en lograr unas tasas de error mínimas en su habilidad de predicción, aunque para ello el modelo no refleje la realidad. Un algoritmo puramente descriptivo se centra en comprender el proceso subyacente que generó los datos, es decir que el modelo sea un reflejo de la realidad. En la práctica la mayoría de las aplicaciones de KDD usualmente demandan un porcentaje de ambos modelados.

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. Uno de los métodos más extendidos, y utilizados en nuestros primeros trabajos, es mediante la construcción previa de un árbol de decisión, en el que cada nodo interior representa una condición simple y cada nodo hoja una clase a asignar. La herramienta más extendida con esta metodología es el C4.5 [Quinlan, 1993]. Básicamente consiste en un algoritmo recursivo con técnica de divide y vencerás que optimiza la construcción del árbol en base a criterios de ganancia de información. La salida del programa es una representación gráfica del árbol encontrado (Figura 3) y una tasa de error aparente y otra estimada.

Las reglas de la figura 3 también se pueden describir como:

Si Condición 1 es V1 o Condición 1 es V2 y Condición 2 es V5 entonces Clase 1.

Si Condición 1 es V2 y Condición 2 es V3 o V4 entonces Clase 2.

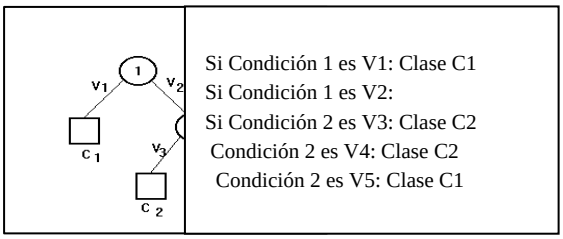


Figura 3: Ejemplo de tipo de árbol no binario y conjunto de reglas equivalentes

A continuación se verán algunos ejemplos de reglas de gestión obtenidas con reglas basadas en intervalos y en lógica borrosa. Otras técnicas aplicadas a la gestión de proyectos pueden verse en [Mata, 2003].

3.2 Modelos dinámicos

Los modelos dinámicos para PDS incluyen una serie de atributos 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. Una vez definidos los atributos del modelo, el gestor de proyecto debe decidir cuáles son las variables que se van a analizar (tiempo de entrega, coste, productividad, errores producidos, etc.), es decir, los objetivos del proyecto. Por tanto, las políticas de gestión que el proceso de aprendizaje automático puede hallar serán aquellas que relacionen valores de los atributos con las variables sobre cuya influencia queremos establecer reglas. Cada vez que se asigna un valor a cada atributo del modelo, éste queda completamente definido y mediante su simulación se obtendrán unos valores para estas variables. Los pasos seguidos en la obtención de reglas de gestión se indican a continuación (Figura 4):

1. Definir:
• Los intervalos de valores que pueden tomar los atributos del proyecto.
• Los objetivos del proyecto que se desean obtener. Por ejemplo, se desea 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. Para obtener la base de datos de la que el sistema aprende, se ha utilizado un simulador cuyo núcleo está constituido por el modelo dinámico propuesto en [Ruiz, 2001], denominado Modelo Dinámico Básico (http://www.lsi.us.es/docs/informes/mdr.pdf) e implantado en el entorno de simulación Vensim . Este modelo tiene un total de 35 atributos lo que equivale a decir que se pueden combinar simultáneamente hasta un total de 35 políticas de gestión diferentes sobre el proyecto.

3. Generación automática de una base de datos simulada. En cada simulación se obtiene un registro de la base de datos en los que se guardan los valores obtenido para los atributos y los valores finales de las variables que nos interese analizar (coste, tiempo, calidad, etc.). Cada registro de la base de datos representa un escenario posible del proyecto.

4. Aplicación de técnicas de aprendizaje automático. A partir de la base generada, el sistema aprende examinando la muestra de casos resueltos y proponiendo un conjunto de reglas para la toma de decisiones.

5. Obtención de reglas de gestión.

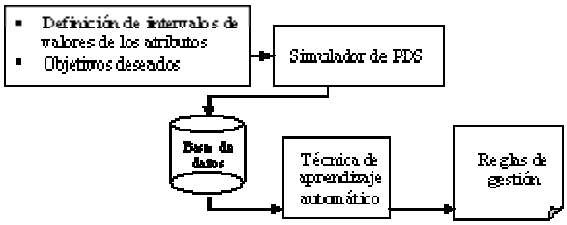


Figura 4: Pasos a seguir para la obtención de reglas de gestión

El responsable del proyecto sólo debe realizar el paso 1 de la Figura 4, el resto de los pasos descritos se realizan automáticamente. Una vez obtenidas las reglas de gestión será el responsable del proyecto el que decida qué regla o reglas son las más adecuadas.

3.3 Reglas intervalares

En este apartado se presentan algunas reglas obtenidas con C4.5 siguiendo los pasos comentados en la Figura 4 partiendo del conocimiento obtenido de un proyecto ya finalizado (proyecto post-mortem) [Ramos, 2000] . Por tanto las reglas obtenidas, del árbol que proporciona el C4.5, nos indican qué políticas de gestión tendríamos que haber aplicado para haber mejorado los resultados obtenidos en el proyecto (coste, tiempo, productividad, etc., o todas simultáneamente). Conocidos los valores finales del proyecto para el esfuerzo (coste) y tiempo, 410 t-d y 151 días respectivamente, se etiquetan como BUENOS cualquier valor menor o igual a los obtenidos. Los valores mayores se han etiquetado como MALOS. Siguiendo los pasos recogidos en la Figura 4, se puede decir que las reglas de gestión que habrían hecho BUENOS simultáneamente el coste y el tiempo de entrega, es decir habrían mejorado los resultados que se obtuvieron, son las siguientes [Aguilar, 2001]:

$DEDIC > 0{,}87;\ RETAR > 14$ (R.1)

$RETAR \leq 9;\ RETRA > 8$ (R.2)

DEDIC, dedicación media de los técnicos. RETAR, retraso medio en la reasignación de técnicos. RETRA, retraso medio en la incorporación de nuevas tareas

Se han obtenido dos reglas, con tan sólo 5 casos de un total de 200 simulaciones. La regla R2 nos indica que se habrían obtenido resultados BUENOS simultáneamente para el coste y el tiempo del proyecto si:

"La dedicación media de los técnicos hubiese sido mayor del 87 % y el retraso medio en la reasignación de los técnicos hubiese sido mayor de 8 días".

A la vista de las reglas anteriores el análisis realizado por el responsable del proyecto sobre las reglas obtenidas fue el siguiente: la regla que podría haberse aplicado en este proyecto habría sido la R.2 ya que RETAR se ajusta al valor estimado inicialmente y los valores de DEDIC y RETRA se podrían haber modificado con facilidad.

Por otro lado y para facilitar la obtención e interpretación de las reglas de gestión, en las Figuras 5 y 6 se presentan interfaces de salida de SEGESOFT. SEGESOFT es un prototipo desarrollado dentro del proyecto CICYT TIC99-0351, cuyo objetivo fue la integración de diferentes módulos para la ayuda en la formación en la gestión de proyectos software [Tuya, 2001]. Mediante la pantalla de entrada de datos (Figura 5) para el módulo de generación de reglas de decisión se introducen los datos necesarios. A partir de estos datos esté módulo transforma la bases de datos simulada en una base de datos etiquetada, de forma que cada registro está formado por los valores de los atributos del modelo y una etiqueta de BUENO o NO BUENO, en función de que los valores de las variables esfuerzo (effort) y tiempo (delivery time) se encuentren en los intervalos señalados en la

interfaz. Una vez realizado este preprocesado, se ha diseñado un nuevo algoritmo de aprendizaje que nos permita encontrar reglas para proyectos BUENOS. Es decir, normalmente los algoritmos de aprendizaje supervisado buscan reglas para todas las etiquetas existentes en la base de datos, sin embargo en el caso que nos ocupa, el gestor normalmente sólo estará interesado en reglas para la etiqueta BUENO.

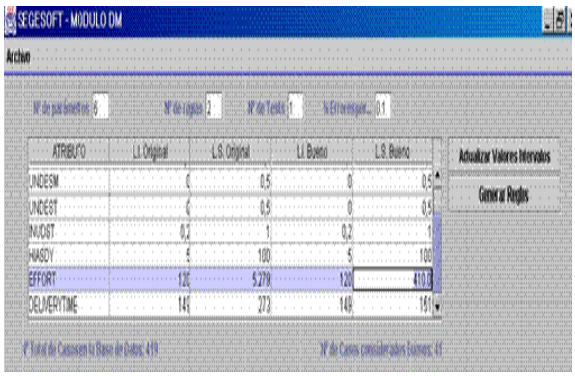


Figura 5: Interface del módulo de aprendizaje automático

En la Figura 6, se muestra un ejemplo de la salida gráfica que se obtiene en el proceso de aprendizaje utilizando SEGESOFT.

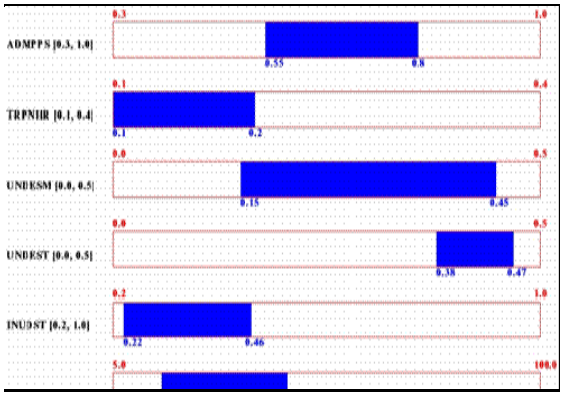
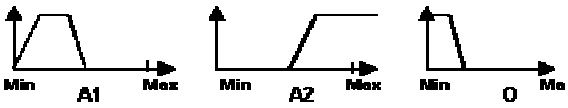


Figura 6: Representación gráfica de una regla

El significado de la regla que aparece en la Figura 6 es el siguiente: si el atributo ADMPPS (dedicación media de los técnicos) está entre el 55% y el 80% y TRPNHR (dedicación media de los técnicos expertos a formación) está entre el 10% y el 20% y UNDESM (infraestimación inicial del esfuerzo estimado) está entre el 15% y 45% y UNDEST (infraestimación inicial del número de tareas estimadas) está entre el 38% y 47% e INUDST (infraestimación inicial del número de técnicos estimados) está entre el 22% y 46% e HIASDY (retraso medio en la contratación y adecuación de los técnicos nuevos) está entre el 16 días y 44 días entonces el proyecto evolucionaría según el criterio definido como BUENO por parte del responsable del mismo. Además las reglas pueden filtrarse considerando sólo los atributos cuyos intervalos no recogen los valores de la estimación inicial para la simulación. Es decir, si el valor inicial del atributo INUDST se encuentra en el rango [22%, 46%] entonces la condición sobre este atributo puede desaparecer de la regla anterior, simplificándose su lectura.

3.4 Reglas borrosas

Las reglas de gestión borrosas obtenidas al aplicar a la base de datos generada por el SPS un algoritmo de lógica borrosa, fuzzy clustering [Sugeno, 1993], se representa de la siguiente forma:



En esta regla el conjunto borroso asignado a cada atributo del proyecto, es representado por un trapecio. En el eje X de cada conjunto borroso se representan los valores del atributo y en el eje Y el valor de pertenencia a un cluster. Esta regla de gestión borrosa se interpretaría de la siguiente forma:

Si A1 es bajo Y A2 es medio o alto Entonces O es muy bajo

La aplicación de técnicas de lógica borrosa nos va a permitir añadir dos nuevas mejoras en el uso de un SPS [Aroba, 2003]:

1. La obtención de reglas de tipo cualitativo, lo cual facilita la interpretación de las reglas obtenidas.
2. El responsable del proyecto no tiene que definir previamente los resultados u objetivos deseados.

En la Figura 7, se muestra los pasos a seguir para obtener este nuevo tipo de reglas. Se observa (paso 1) cómo el responsable del proyecto no tiene que definir los objetivos del proyecto. En este caso, la base de datos 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 en formato

gráfico, que permiten al responsable interpretar los datos del proyecto de forma rápida y clara.

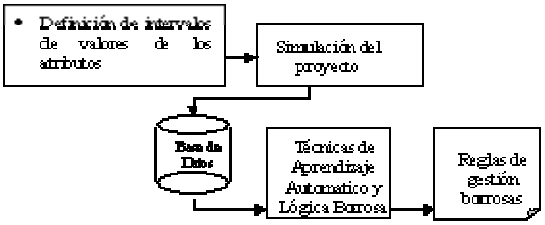


Figura 7: Pasos a seguir par la obtención de reglas borrosas

En la Figura 8 se muestran reglas de gestión borrosas obtenidas aplicando el algoritmo PreFuRGe (Predictive Fuzzy Rules Generador), desarrollado en [Aroba, 2003], a la base de datos cuantitativa obtenida como resultado de la simulación de un proyecto. En esta figura se muestran las reglas de gestión obtenidas a partir de una serie de datos cuantitativos con dos variables de salida, tiempo de entrega del proyecto (TIME) y coste (COST). A partir de esta representación gráfica de las reglas, el responsable del proyecto tiene una información de tipo cualitativo, fácil de interpretar, que le permite tomar decisiones sobre el proyecto.

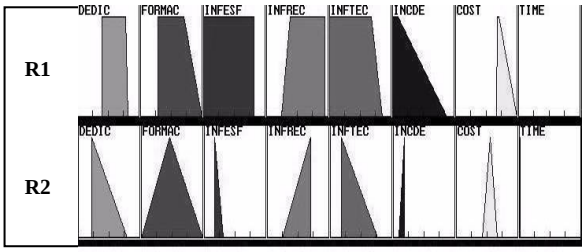


Figura 8: Pasos a seguir par la obtención de reglas borrosas

La segunda (R2) regla indica que:

"El coste toma valores medios y el tiempo toma valores muy bajos, simultáneamente, cuando la dedicación media (DEDIC) es media-baja, la dedicación a formación (FORMAC) es media, la infraestimación del esfuerzo necesario (INFESF) es baja, la infraestimación del tamaño del proyecto (INFREC) es media-baja, el número de técnicos que comienza el proyecto (INFTEC) es medio-alto y el tiempo medio de incorporación de los técnicos en el proyecto (INCDE) es bajo".

Observemos que en este tipo de reglas el responsable del proyecto puede comprobar directamente el rango de valores en el que se mueven las variables de salida (objetivos).

4. CONCLUSIONES

En este trabajo se ha presentado el modelado dinámico de proyectos software como una alternativa a la estimación y gestión tradicional de dichos proyectos. Las facilidades que aportan los actuales entornos de simulación han permitido, a partir de modelos dinámicos para proyectos software, crear entornos de trabajo que facilitan al responsable de proyectos simular el comportamiento de su proyecto en cualquier estado.

Estos entornos de trabajo permiten, también, generar diferentes escenarios del proyecto creando una base de datos sobre la que pueden aplicarse diferentes técnicas de aprendizaje automático para obtener reglas de gestión para la ayuda en la toma de decisiones. Igualmente estas técnicas podrían aplicarse sobre bases históricas de proyectos de una organización de desarrollo. De esta manera se podrían obtener reglas de gestión basadas en la experiencia del trabajo realizado hasta la fecha por dicha organización. Si es así, se puede decir que la experiencia que se ha ido adquiriendo con la realización de los proyectos en una organización de desarrollo servirá a los responsables de proyectos para estimar y gestionar futuros proyectos y no será exclusiva de una persona.

Un aspecto a tener en cuenta es la utilidad de los modelos dinámicos en la formación del personal novel y el entrenamiento de los directores de proyectos ya que pueden simular situaciones reales y comparar diferentes alternativas para mejorar los resultados del proyecto sin ningún riesgo.

Resaltar que la facilidad para la construcción de modelos dinámicos para entornos y proyectos específicos y su utilización va a depender, en gran medida, del nivel de conocimiento que la organización de desarrollo tenga de sí misma.

Finalmente, comentar que el trabajo presentado se ha realizado con el apoyo de los proyectos CICYT TIC99-0351 y CICYT TIC2001-1143.

REFERENCIAS

Abdel-Hamid, T., Madnick, S.; Software Project Dynamics: an integrated approach, Prentice-Hall, 1991.

Aguilar, J., Ramos, I., Riquelme, J.C; An Evolutionary Approach to estimating software development projects, Information and Software Technology, 43, 875-882, 2001.

Aracil J., Gordillo F.; Dinámica de Sistemas. Alianza Editorial, Madrid 1997.

Quinlan, J. C4.5: Programs for Machine Learning, Morgan Kaufmann Pub. Inc., 1993.

Aroba, J.; Avance en la toma de decisions en proyectos de desarrollo de software. Tesis Doctoral codirigida por Ramos, I.; Riquelme, J.C.. Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Sevilla.

Mata, J., Álvarez, J.L., Riquelme, J.C. y otros; Visualization Techniques of management rules for software development projects, Proceeding of the Third International Conference on Quality Software (QSIC), 360-367, Dallas (Texas, USA), 2003.

Ramos, I., Aguilar, J., Riquelme, J.C., Toro, M., A new method for obtaining software project management rules. International Conference Software Quality Management VIII (SQM2000). pp. 149-160. Greenwich (London, UK), 2000.

Ruiz, M., Ramos, I.; A Simplified Model of Software Project Dynamics , The Journal of Systems and Software, 59, 299-309, 2001.

Sugeno, M., Yasukawa, T.; A Fuzzy-Logic Based approach to qualitative Modeling. IEEE Transactions on Fuzzy Systems, Vol. 1, 7-31, 1993.

Tuya, J., Fernández, P., Prieto, M.A., Aguilar, J., Ramos, I., Riquelme, J.C., y otros, Integration of Information in a Training Environment for Software Project Management. International Conference on Software Quality Management IX (SQM), 31-42, London (UK), 2001.