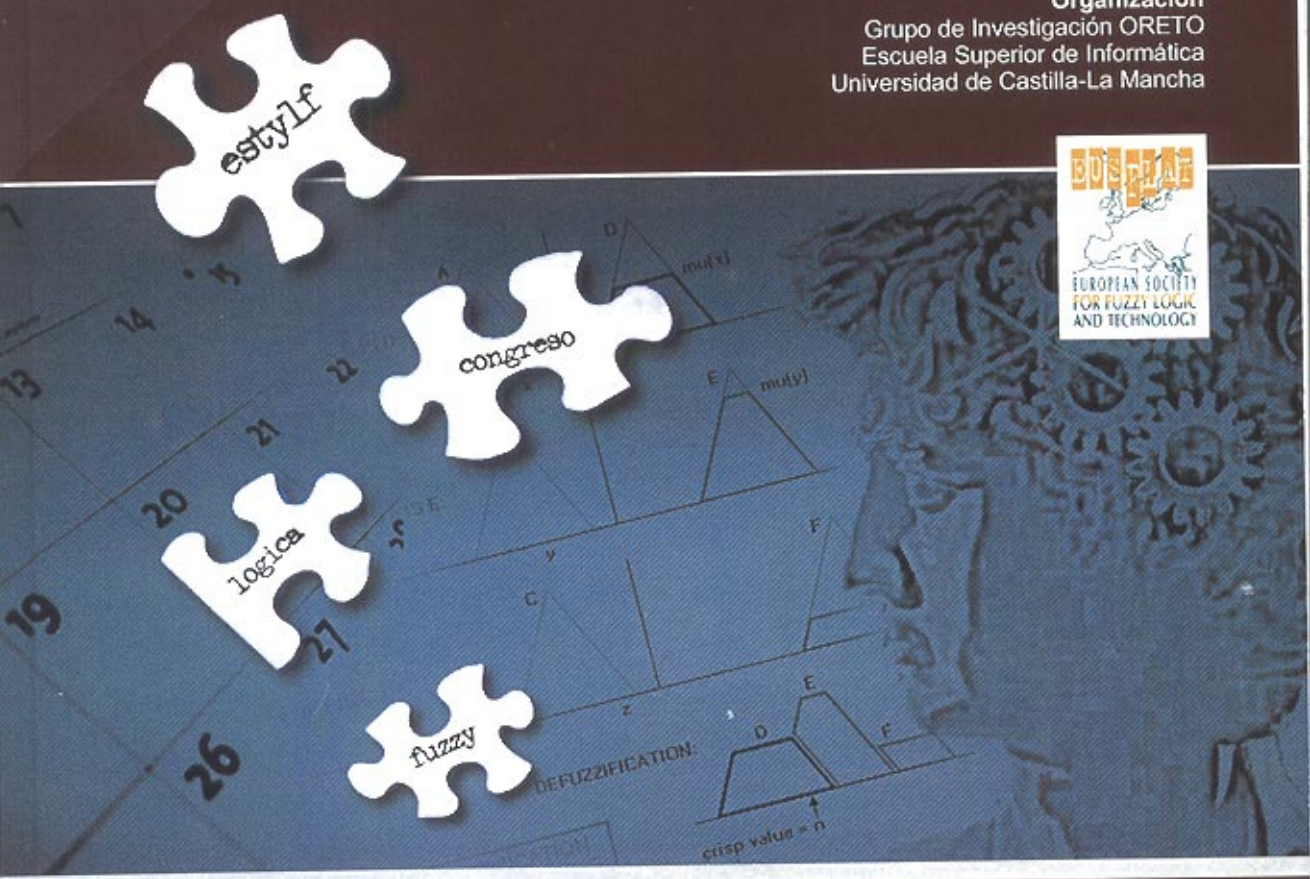
Estylf 2006

XIII Congreso Español sobre Tecnologías y Lógica Fuzzy

Organización
Grupo de Investigación ORETO
Escuela Superior de Informática
Universidad de Castilla-La Mancha



ESTYLF 2006
Ciudad Real
20/22 Septiembre

Este libro de actas recoge todos los trabajos que han sido presentados como contribuciones al *XIII Congreso Español de Tecnologías y Lógica Fuzzy* (ESTYLF) de la *European Society for Fuzzy Logic and Technology* (EUSFLAT).

El XIII Congreso ESTYLF ha sido organizado por el grupo de investigación Oreto, del departamento de Tecnologías y Sistemas de Información de la Universidad de Castilla-La Mancha, en las instalaciones de la Escuela Superior de Informática en Ciudad Real.

Desde estas líneas queremos agradecer a todos los que han contribuido a la elaboración de estas actas, tanto en su aspecto visual como en sus contenidos. De igual forma agradecemos la colaboración de las instituciones que han participado y con su apoyo han permitido la celebración de este congreso científico bianual.

Esperando que este encuentro en Ciudad Real sirva para mostrar el potencial que esta joven Universidad de Castilla-La Mancha es capaz de generar, así como ser unos dignos representantes del carácter acogedor de esta tierra.

Entidades Colaboradoras

Universidad de Castilla-La Mancha
Vicerrectorado del Campus de Ciudad Real y Extensión Universitaria
Vicerrectorado de Investigación

Castilla-La Mancha
Consejería de Educación y Ciencia

Escuela Superior de Informática

Departamento de Tecnologías y Sistemas de Información

tsi

Diputación Provincial de Ciudad Real

Ayuntamiento de Ciudad Real

ISBN 84-689-9547-9

REQUERIMIENTOS DIFUSOS

Daniel Montenegro Arenas[1] daniel.montenegro@gmail.com
Luís Garmendia Salvador[1] lgarmend@fdi.ucm.es
Matilde Santos Peñas[2] msantos@dayca.ucm.es

[1]Departamento de Sistemas Informáticos y Programación
[2]Departamento de Arquitectura de Computadores y Automática
Facultad de Informática. Universidad Complutense de Madrid. 28040-Madrid. España

Resumen

En este trabajo se proponen un procedimiento para el manejo de requerimientos imprecisos y con incertidumbre, así como posibles estrategias de diseño e implementación que los instancien de cara a la puesta en marcha de proyectos de Ingeniería del Software. El principal objetivo de este proceso de tratamiento difuso de restricciones es reducir el tiempo en las etapas de elicitación y en el análisis de requerimientos, en beneficio del tiempo total del proyecto de desarrollo software. Se plantea su posible aplicación a motores de inferencia difusos y motores de reglas de negocios.

Palabras Clave: Requerimientos Difusos, Ingeniería del Software, Lógica Difusa, Motores de Reglas de Negocio.

1. INTRODUCCION

El estudio de requerimientos software suele hacerse utilizando el lenguaje natural, muchas veces lleno de términos imprecisos difíciles de modelar. La elicitacion y análisis de requerimientos exigen una gran cantidad de tiempo y recursos para modelar dicha imprecisión y realizar el tratamiento de la incertidumbre relacionada con ellos. Generalmente los ingenieros de requerimientos que se enfrentan a requisitos imprecisos y vagos tienen que realizar una labor de análisis exhaustiva con el cliente para poder explicitar dichos requerimientos y expresarlos en un lenguaje sin ambigüedades.

Este análisis exhaustivo de requerimientos conlleva tiempos y costos dentro del proyecto que retrasan actividades posteriores del mismo (diseño, implementación, pruebas, etc.). Estas demoras perjudican en gran medida a los proyectos, y aunque son solo una pequeña parte de los posibles retrasos y costes añadidos que sufren, pueden llegar a ser relevantes en el éxito y fracaso de los mismos.

En este documento, en primer lugar, abordaremos la problemática real de los proyectos de software en la actualidad, que viene dada fundamentalmente por la necesidad de acortar tiempos en las fases iniciales de los mismos. Para ello se verá en la Sección 3 una forma de abordar los requerimientos imprecisos para que, en etapas posteriores del proceso de desarrollo de software, puedan ser concretados por estrategias de diseño e implementación. La sección 4 estudia la aplicación de estas propuestas en los motores de inferencia difusos y en los motores de reglas de negocios. Finalmente en las conclusiones se analizan posibles mejoras de este procedimiento.

2. LA REALIDAD DEL DESARROLLO DE PROYECTOS DE SOFTWARE

Muchos estudios sobre los proyectos de software dejan una imagen desalentadora. En ellos se muestra como sólo un pequeño porcentaje de los proyectos se consideran exitosos y en la mayoría de ellos los requerimientos no se cumplen en su totalidad (Figura 1).

Entre un tercio y un cuarto de los proyectos son considerados exitosos, es decir, proyectos que desarrollan los requerimientos del cliente en el tiempo y los costes estimados. El resto de ellos se entregan con costes excedidos, fuera del tiempo, con funcionalidad incompleta o simplemente son abandonados. Por otra parte, los proyectos que terminan (sin importar el coste o tiempo empleados) sólo cumplen con una parte de los requerimientos del cliente, entre la mitad y dos terceras partes de los mismos.

Según el análisis del Standish Group [5] [6] del año 2000, los proyectos exitosos fueron tan solo el 28%, mientras que un 21,6% de ellos fueron cancelados, y el restante 50,4% fueron entregados con exceso de costes, tiempo, o

con funcionalidad parcial[1]. Resultan aun más desalentadoras las cifras en cuanto al coste y tiempo extra invertidos por las empresas: 214% de la inversión presupuestada y 239% del tiempo estimado.

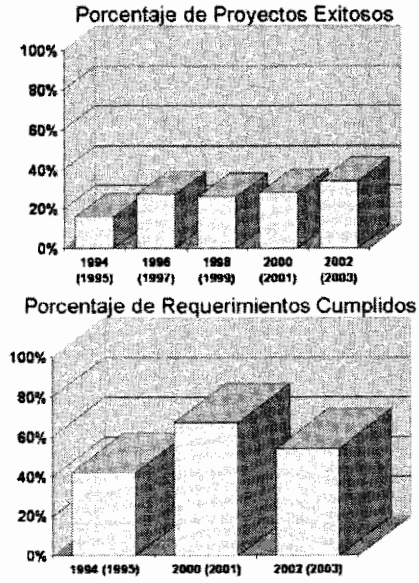


Figura 1: Resultados sobre realización de proyectos

Frente a estas alarmantes cifras, el Standish Group realizó una encuesta con los directores ejecutivos de empresas de tecnología para obtener sus opiniones acerca del éxito de los proyectos. A partir de estas opiniones se creo el *CHAOS Ten* [6], diez factores con una determinada puntuación que aseguran un nivel de confianza en cuanto al éxito del proyecto: usuarios involucrados (19), ayuda de la dirección ejecutiva (16), claridad de los requerimientos (15), planificación apropiada (11), expectativas realistas (10), hitos cortos (9), personal competente (8), pertenencia (6), claridad de visión y objetivos (3), y personal enfocado al trabajo (3). Con estos factores son calificados los proyectos para tener una visión concreta de las posibilidades de que estos sean exitosos o no.

2.1. LAS FASES INICIALES DE LOS PROYECTOS

Independientemente de la metodología que se utilice en el proyecto de software, ésta contará con unas fases iniciales de conocimiento del negocio y elaboración u obtención de requerimientos [1]. Algunas de ellas pueden realizar estas actividades una sola vez al principio del proyecto (Metodologías tradicionales), otras pueden hacerlas repetidamente a lo largo de su proceso (Metodologías cíclicas y repetitivas). Unas pueden ser estrictas en el manejo de información concreta en los requerimientos y otras, como algunas metodologías ágiles, guiar la elicitación de requerimientos como procesos con pocos formalismos pero involucrando seriamente al cliente. Lo cierto es que para todas las metodologías, las etapas de elicitación y análisis de requerimientos son fases iniciales relevantes dentro del proceso de software. Aunque existan visiones iterativas, para cada iteración los requerimientos serán siempre el inicio del proceso.

Teniendo esto en cuenta, podemos apuntar como una vía sustancial de mejora en el diseño de proyectos el tercer factor del *CHAOS Ten*: *claridad de los requerimientos*. Tener requerimientos claros y precisos es una necesidad latente en todos los proyectos y su consecución está supeditada a procesos de validación, negociación y documentación exhaustiva de dichos requerimientos. La validación y negociación de requerimientos siempre involucra de una manera activa a clientes y usuarios, los cuales deben asegurar la nitidez de la información que los requerimientos reflejan. Pero la validación y negociación de requerimientos consumen gran cantidad de tiempo. Son necesarias varias reuniones con los clientes y realizar diversas estrategias de educción de la información y análisis para poder obtener un requerimiento concreto y con información nítida.

La pregunta que surge es: ¿podemos asegurar claridad en los requerimientos sin necesidad de concretarlos? No para todos los requerimientos, pero para aquellos que manejan incertidumbre. Algunos requerimientos tienen la necesidad de ser precisados completamente antes de ser pasados a fase de diseño, pero otros pueden ser imprecisos y es en esa etapa de diseño cuando son definidos. ¿Qué beneficios se obtienen con este planteamiento? Se evitan o reducen las reuniones y procesos de negociación con el cliente sobre requerimientos, que no necesitan la inversión de tanto tiempo para concretarlos, lo que supone una gran ventaja para ambas partes. De esta forma cuando ya esté el diseño, y posiblemente la implementación, se pueden realizar tareas de concretar la funcionalidad ya diseñada, pero para ese momento tanto el cliente como el analista y el diseñador están orientados en negociar la información que debe ser nítida.

3. TRATAMIENTO DIFUSO DE REQUERIMIENTOS

En esta sección se expone un bosquejo del diseño de un modelo que trate los requisitos software con términos

[1] Información de empresas pequeñas. Las empresas grandes y medianas presentan porcentajes más críticos de proyectos exitosos, cancelados, y entregados con problemas de costo, tiempo o funcionalidad.

imprecisos, procedentes del lenguaje natural de los expertos, en la definición de sistemas y elicitación de requerimientos propios de Ingeniería del Software.

Se propone para ello una metodología cuya entrada son explicaciones coloquiales, que se modelan con lógica difusa, y a las que se aplican motores de inferencia para que, finalmente, tras un proceso de "clarificación", sirva para obtener unos requerimientos software con información nítida que permita un desarrollo claro y concreto de proyectos informáticos.

3.1. REQUERIMIENTOS DIFUSOS

Los requerimientos, según el Instituto de Ingenieros Eléctricos y Electrónicos [8], son:

1) una condición o capacidad necesitada por un usuario para resolver un problema o lograr un objetivo;
2) una condición o capacidad que debe ser cumplida o poseída por un sistema o un componente de un sistema para satisfacer un contrato, estándar, especificación u otros documentos impuestos formalmente;
3) una representación documentada de una condición o capacidad como las anteriores.

Los buenos requerimientos deben ser [2, 3, 4]: necesarios, concisos, evitar ambigüedades, consistentes, completos, alcanzables, verificables y claros.

Es decir, un *Requerimiento Difuso* será cualquier condición o capacidad necesaria en un sistema que resuelva un problema o logre un objetivo, y que satisfaga unas características imprecisas que modelen el mundo real. Estos *requerimientos difusos* estarán basados en *predicados difusos*, los cuales serán variables lingüísticas calificables que describan un aspecto del requerimiento.

Tabla 1: Ejemplos de Requerimientos Difusos.

Requerimiento Difuso	Términos con Predicados Difusos
Calificar películas según la cantidad de espectadores que las ven.	• Calificación* • Espectadores*
Registrar medidas barométricas cuando el clima es adverso para la navegación.	• Clima*
Diagnosticar enfermedades respiratorias dependiendo de la tos y la temperatura corporal	• Tos* • Temperatura Corporal*
Evaluar estudiantes por su conocimiento, creatividad y motivación	• Conocimiento* • Creatividad* • Motivación*

En la Tabla 1 se dan algunos ejemplos de *requerimientos difusos* y los posibles términos de los que se pueden encontrar *predicados difusos*. Estos términos serán *conjuntos borrosos* que no tienen una frontera precisa y su función de pertenencia indica un grado de cumplimiento de ese atributo [7]. Se utiliza (*) como notación para aquellos términos que pueden llegar a ser calificados con *predicados difusos*.

En el primer ejemplo identificamos dos términos que pueden tener *predicados difusos*. El primero de ellos, calificación, puede ser bien una escala o simplemente predicados como buena, aceptable, mala, etc. El segundo termino, cantidad espectadores, podría contener predicados como muchos, pocos, etc. En el segundo ejemplo, el termino clima puede contener predicados como despejado, nublado, tormentoso, etc. El resto de ejemplos podríamos identificar esos predicados con los cuales trabajaremos más adelante.

Para reconocer un requerimiento que pueda ser denominado como difuso debemos tener en cuenta la naturaleza de la información que maneja. Si el requerimiento depende de información concisa para ser validado, o es una funcionalidad que incluye valores determinados para su ejecución, no es un buen candidato. Los requerimientos que utilicen en su descripción términos a los cuales pueden aplicarse predicados calificativos que presenten antonimia, o la información de la que dependen está supeditada a este tipo de términos calificables, tenemos grandes posibilidades de incluirlos como *requerimientos difusos*.

4. REQUERIMIENTOS DIFUSOS EN EL PROCESO DE DESARROLLO DE SOFTARE

El ciclo de vida de un *requerimiento difuso* es el mismo que el de un requerimiento normal. La diferencia radica en que el ingeniero de requerimientos debe identificar únicamente los *predicados difusos* que definen los términos del requisito, y documentar estos predicados junto a los requerimientos dentro de las especificaciones que para esta fase estén planteadas. Es en este punto donde se obtiene el beneficio de esta metodología. Al tener que definir sólo unos pocos predicados que describan el comportamiento de un término, los ingenieros de software podrán centrarse específicamente en el negocio, y no en las cifras, cantidades o límites numéricos que dificultan la labor de especificación de requerimientos. Un *requerimiento difuso* es entonces descrito empleando menos tiempo que uno normal, y puede pasar a etapas posteriores del proceso de desarrollo de software con mayor rapidez.

Una vez el requerimiento difuso ha sido elicitado, debe realizarle el análisis del mismo para comprobar su difusidad. Si es necesario, un requerimiento deberá clarificarse y seguirá el ciclo de vida tradicional de los requerimientos nítidos. Generalmente las labores de elicitación de requerimientos y el análisis son etapas independientes realizadas por personas diferentes, pero muchos proyectos integran estas etapas, por lo cual esta verificación en análisis puede obviarse. Al tener el requerimiento analizado, validado, negociado y documentado, y este se encuentra en una especificación de funcionalidad (la más conocida de ellas es la especificación de casos de uso), puede pasar a la etapa de diseño. En ella se decidirá de qué forma va a implementarse este tipo de requerimientos. En este punto, y con una especificación de funcionalidad donde están claramente identificados los predicados difusos, podemos evaluar la posibilidad de utilizar los Motores de Reglas de Negocio para modelar los requerimientos difusos. De la misma forma que en análisis debe verificarse la naturaleza difusa de la funcionalidad planteada, y si es necesario, realizar un proceso de clarificación que convierta la información contenida en los requerimientos y el análisis, en datos nítidos para que el diseño pueda realizarse con normalidad. Finalmente al llegar a la fase de implementación, y cuando el diseño de los requerimientos plantea una solución basada en lógica difusa, estos requerimientos serán implementados apoyándose en los Motores de Inferencia Difusa. Aun en este punto es necesario realizar la ultima verificación sobre la difusidad de la información contenida, y de la misma forma, si se necesita, clarificar requerimientos, análisis y diseño de una forma nítida que permita la implementación de la funcionalidad por medios tradicionales (Figura 2).

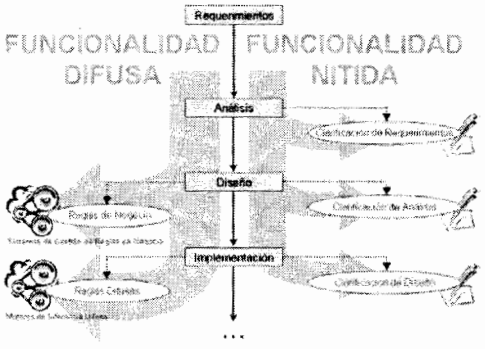


Figura 2: Método de Desarrollo de Requisitos Difusos

Los motores de inferencia difusa son herramientas matemáticas orientadas a implementar reglas difusas. Son conocidos ampliamente en sectores como la electrónica y van muy ligados a dispositivos inteligentes de control. Como componente software son herramientas poco difundidas en el ámbito comercial, pero conocidas en el ámbito académico. Pueden encontrarse tanto en herramientas comerciales, como Mathlab[2], o también en paquetes de libre distribución, como XFuzzy[3].

Por otro lado, un motor de reglas de negocio es un software potente, robusto y caro, donde son modeladas las funcionalidades como reglas en lenguaje natural o propietario, utilizando términos de negocio acotados por cantidades especificas, las cuales pueden ser modificadas en cualquier momento, incluso en etapas de producción. Son ampliamente conocidos como complementos para herramientas de Inteligencia de Negocio, Gestores de Procesos de Negocio e Integradores.

La elección de uno u otro motor de inferencia dependerán de la arquitectura presente en la aplicación y de la posibilidad de manejar los predicados difusos como variables del sistema que cambiarán mientras éste se encuentre en producción. Si se toma la decisión de utilizar el motor de inferencia difuso, será necesario en la etapa de diseño acotar los predicados por cantidades significativas para darle coherencia al término y sus calificativos, es decir, identificar los valores límite de los predicados dentro del conjunto borroso que los contiene. Por el contrario, si es elegido el motor de reglas de negocio, esta identificación de valores puede esperar tranquilamente, incluso hasta momentos de producción, ya que estas herramientas cuentan con la funcionalidad necesaria para incluir estos acotamientos en cualquier momento.

De cualquiera de las dos formas, existirá un proceso de negociación con el cliente para acordar estos valores limites, pero estas reuniones se harán en etapas posteriores (diseño, implementación e incluso producción), lo cual será menos crítico que hacerlo en la fase de requerimientos.

4.1. BENEFICIOS DEL USO DE REQUERIMIENTOS DIFUSOS

Según [9], uno de los problemas principales con lo métodos de diseño de software es que para transformar los requerimientos en modelos de diseño, era necesario tener un conjunto de requerimientos nítido, libre de ambigüedades y estable. Pero la realidad reflejaba que los requerimientos contenían generalmente conflictos, eran vagos y cambiaban con el tiempo. Esto podría contraponerse a uno de los factores de éxito identificado en el CHAOS TEN [6]: claridad de los requerimientos. Muchos proyectos fracasan porque su conjunto de

[2] http://wwwmathlab.com/
[3] http://www.imse.cnm.es/Xfuzzy/

requerimientos presenta problemas y conflictos. Pero esta es la naturaleza de los requerimientos.

Con el uso de los requerimientos difusos la clarificación de los requerimientos no es necesaria, ya que el diseño, y la implementación de estos requerimientos puede realizarse con la información difusa existente. Por supuesto que no todo requerimiento con información difusa podrá diseñarse e implementarse, pero avanzar en el ciclo de vida del requerimiento difuso podrá restringir el requerimiento lo suficiente para que al momento de volver la información vaga en nítida, se cuente con más herramientas para hacer este proceso mucho más rápido y efectivo. Los beneficios que se obtendrían con el uso de requerimientos difusos serían:

- Menor tiempo en Requerimientos: acortar el tiempo de elicitación, documentación y negociación de requerimientos, y con esto acortar el tiempo de entrega de la funcionalidad implementada. El hecho de ganar tiempo en requerimientos es evidente por la ausencia de reuniones para clarificar requerimientos, pero que este tiempo, o una parte de el, se gane al final del ciclo de vida es una asunción. Como bien puede presentarse el caso en que muchos de los requerimientos puedan ser implementados con la lógica difusa, podría presentarse el caso donde todos los requerimientos tuvieran que ser clarificados y transformar toda la información a datos nítidos. En este último caso podría no presentarse ganancia de tiempo al finalizar el ciclo de vida.
- Responsabilidad de definición nítida posterior: se cuenta con una metodología que solo especifica información nítida cuando es estrictamente necesario, y para ello cuenta con las actividades de clarificación de requerimientos, de análisis y de diseño.
- Nitidez a partir del conocimiento del negocio: con el conocimiento adquirido del negocio en el tiempo que se demora en convertir la información del requerimiento en datos nítidos, se está en capacidad de definir la información difusa sin necesidad de recurrir al cliente para preguntarlo.

5. CONCLUSIONES

La principal aportación original de este trabajo es el planteamiento de definición de un modelo de proceso de ingeniería de software basado en requerimientos con información difusa. A su vez esta metodología podrá contener estrategias de clarificación en distintas etapas del proceso de desarrollo para convertir la información difusa de los requerimientos en información nítida para su implementación. El uso de este modelo metodológico conllevará beneficios a la hora de afrontar un proceso de desarrollo, al acortar tiempos críticos en los procesos que involucran al cliente (específicamente los requerimientos) amortizando estos tiempos en procesos donde esta participación no resulta relevante.

El mejoramiento de este modelo de trabajo para *requerimientos difusos* debe realizarse en base a experimentos en el mundo real. Su uso en un proyecto que maneje este tipo de información ayudara a identificar las carencias que el modelo pueda tener, y dar pie a nuevas interpretaciones de los términos *requerimiento difuso* y *predicado difuso*, de cara a soportar las dificultades que puedan encontrar los ingenieros de software al identificar este tipo de requerimientos.

Agradecimientos

Agradecemos el soporte del proyecto de Innovación educativa PIE 2003/7 de la UCM, el proyecto MCyT MTM2005-08982-C04-01 y el proyecto CAM Ref. 910149.

Referencias

[1] Deb Jacobs. Requirements Engineering So Things Don't Get Ugly. *CrossTalk, The Journal of Defense Software Engineering*. Octubre 2004.

[2] Shari Lawrence Pfleeger. Ingeniería de Software, Teoría y Práctica. *Pearson Education*. 2002.

[3] Roger S. Pressman. Ingeniería del Software, un Enfoque Práctico. *Mc Graw Hill*. 5ª Edición. 2002.

[4] Ian Sommerville.Ingeniería del Software. *Pearson Education*. 7ª Edición. 2005.

[5] The Standish Group. CHAOS: A Recipe for Success. *West Yarmouth, MA: The Standish Group International, Inc.* 1999.

[6] The Standish Group. T23E-T10E Standish Group Report. *West Yarmouth, MA: The Standish Group International, Inc.* Octubre 2000.

[7] Lotfi A. Zadeh. Fuzzy Sets, Information and Control, 8, 338-353, 1965.

[8] Institute of Electrical and Electronic Engineers. IEEE Software Engineering Standards Collection: 1994 Edition. Washington, DC. 1994.

[9] Noppen, Joost; Van den Broek, Pim; Aksit, Mehmet. Dealing with Fuzzy Information in Software Design Methods. *2004 Annual Meeting of the North American Fuzzy Information Processing Society*. 2004.