# Elaboración de Documentos de Requisitos en Asignaturas de Ingeniería del Software

Francisco J. García Peñalvo, Mª N. Moreno García
Dpto. de Informática y Automática
Universidad de Salamanca
37008 Salamanca
e-mail: fgarcia@usal.es, mmg@usal.es

Amador Durán Toro
Dpto. de Lenguajes y Sistemas Informáticos
Universidad de Sevilla
41012 Sevilla
e-mail: amador@lsi.us.es

**Resumen**

La relación entre la calidad de los requisitos y del producto final es hoy en día algo ampliamente asumido dentro de la Ingeniería del Software. Al contrario que hace unos años, en los que simplemente se hablaba del *análisis* de requisitos que se suponían proporcionados por el cliente, actualmente se reconoce como necesario un proceso mucho más complejo, la *Ingeniería de Requisitos*, en el que deben participar de forma activa tanto clientes y usuarios como desarrolladores de software. Tradicionalmente, la enseñanza de los temas relacionados con los requisitos se ha centrado más en técnicas de análisis, básicamente modelado, que en otras más importantes como puede ser la elicitación. En este artículo se presenta cómo se aborda la Ingeniería de Requisitos en la Ingeniería Informática en la Universidad de Salamanca, y el impacto que ha tenido la adopción de la metodología desarrollada en el Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Sevilla.

## 1. Introducción

La rama de la ingeniería de software conocida como *Ingeniería de Requisitos*[1] (IR) está recibiendo cada vez más atención por parte de las comunidades académica y profesional. Los resultados presentados en los informes *CHAOS* [14] y *ESPITI* [8] confirman con resultados de proyectos reales que las principales causas, tanto de éxito como de fracaso, de los proyectos de desarrollo de software están directamente relacionadas con los requisitos. En [12] se pone también de manifiesto que la formación en IR es una de las principales carencias detectadas por los profesionales de las tecnologías de la información una vez que abandonan la Universidad y comienzan su carrera profesional.

Este cambio de mentalidad respecto a la importancia de los requisitos se puede apreciar claramente en la nomenclatura utilizada para hacer referencia a las actividades relacionadas con los mismos. Hasta principios de los 90, se consideraba que la única actividad relacionada con los requisitos era el *análisis de requisitos*. Es decir, se asumía que el cliente proporcionaba los requisitos y que los *analistas* se limitaban a *analizar* las peticiones de los clientes a partir de las cuales se diseñaba y se implementaba el nuevo sistema. Una vez que se observó que está actitud *pasiva* de los analistas conducía al fracaso de la mayoría de los proyectos, se impuso la idea de que la elaboración de los requisitos no podía considerarse una responsabilidad única del cliente sino que debía ser una labor conjunta entre clientes, usuarios, desarrolladores y cualquier otra persona con algún interés en el sistema a desarrollar, lo que se conoce como *stakeholder*.

### 1.1. El Proceso de Ingeniería de Requisitos

Este cambio de actitud hace necesaria un nuevo proceso, la IR, que ya no se limita a analizar las peticiones de los usuarios, sino que incluye los siguientes subprocesos (ver Figura 1):

[1] Algunos autores sitúan a la IR como parte de la *Ingeniería de Sistemas*, en la que se desarrollan sistemas compuestos tanto por hardware como por software. En este artículo consideraremos que los sistemas de nuestro interés son compuestos básicamente por software, por lo que la IR de la que hablamos será de la *IR de sistemas software*.

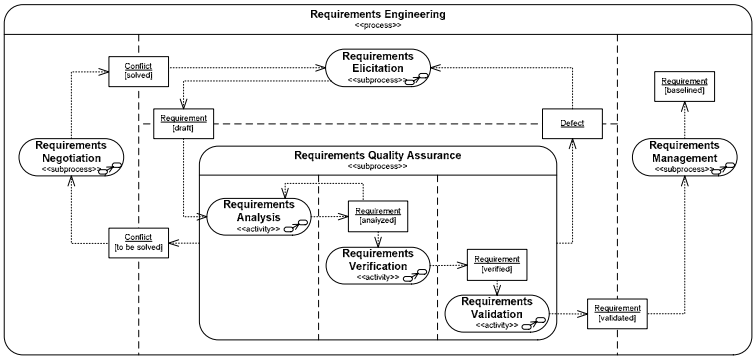


Figura 1. Proceso de Ingeniería de Requisitos (sólo se muestran algunos productos)

- *Elicitación de requisitos*: este subproceso, probablemente el más crítico y el más difícil de realizar, tiene como objetivos buscar, investigar y ayudar a los clientes y usuarios a documentar sus necesidades. La documentación de los requisitos deberá hacerse siempre usando el vocabulario de clientes y usuarios, de forma que éstos puedan entenderlos, siendo lo más habitual emplear lenguaje natural. Las técnicas más comunes son las entrevistas, reuniones en grupo, estudio *in situ*, etc. [2][13].
- *Análisis de requisitos*: esta actividad, parte del subproceso de aseguramiento de la calidad, tiene como objetivo principal detectar conflictos en los requisitos elicitados, normalmente mediante técnicas de modelado conceptual (estructuradas, orientadas a objetos o formales) y de prototipado de interfaz de usuario. Los modelos generados son también una importante herramienta de comunicación con diseñadores y programadores [2][13].
- *Verificación de requisitos:* esta actividad de calidad tiene como objetivo detectar defectos en los requisitos previamente analizados, normalmente mediante técnicas como revisiones formales, listas de comprobación (*checklists*), etc.
- *Validación de requisitos:* esta tercera actividad de calidad intenta asegurar que los requisitos verificados reflejan realmente las necesidades de clientes y usuarios. Las técnicas empleadas suelen ser reuniones en las que se revisan los requisitos mediante el apoyo de prototipos de interfaz de usuario.
- *Negociación de requisitos:* el objetivo de este subproceso es buscar soluciones a los conflictos detectados que satisfagan a los distintos *stakeholders*.
- *Gestión de requisitos:* este subproceso gestiona todo el proceso, en especial las peticiones de cambios en los requisitos, el impacto de dichas peticiones, las distintas versiones de los requisitos, etc.

### 1.2. Educación en Ingeniería de Requisitos

Normalmente, los cursos de introducción a la Ingeniería del Software, en lo tocante a los tópicos relacionados con los requisitos, se centran fundamentalmente en aspectos de modelado. Esto conlleva un estudio muy somero (incluso una supresión total) de todo lo relacionado con la elicitación y documentación de los requisitos.

Esta situación provoca diversos problemas y malos entendidos a la hora de llevar a cabo una documentación de requisitos cuando el alumno tiene que enfrentarse a la realización de una Especificación de Requisitos del Software (ERS), que incluye tanto requisitos en lenguaje natural como modelos conceptuales, en proyectos software no triviales como puede ser el caso de los proyectos fin de carrera (PFC) de una Ingeniería Informática o un proyecto de desarrollo de software real.

Tratando de evitar esta situación en los estudios de Informática de la Universidad de Salamanca, se ha buscado como objetivo la introducción de una forma unificada de documentar requisitos. Existen diversas propuestas para especificar los requisitos, como por ejemplo el estándar IEEE Std. 830-1998 [10], sin embargo nosotros hemos comprobado que la metodología de elicitación y documentación de requisitos [3][5][6], desarrollada en el Departamento de Lenguaje y Sistemas Informáticos de la Universidad de Sevilla (al que nos vamos a referir como Metodología de Durán y Bernárdez, abreviadamente MDB), se ajusta mejor a los objetivos perseguidos.

Así, en el curso 1998-1999 se introdujo en la asignatura de Ingeniería del Software de la Ingeniería Técnica en Informática de Sistemas (ITIS) dicha metodología MDB, siendo ésta satisfactoriamente aceptada por los alumnos de esta titulación tanto en las prácticas obligatorias de la asignatura de Ingeniería del Software como en la documentación técnica de sus PFC.

Concretamente en este artículo se presenta la evolución del uso que ha tenido esta metodología en la titulación ITIS en la Universidad de Salamanca. El resto del artículo se organiza como sigue: la Sección 2 explica los fundamentos básicos de la metodología de elicitación y documentación de requisitos elegida; la Sección 3 presenta el uso de la metodología MDB en ITIS desde el curso 1998-1999 hasta la actualidad; la Sección 4 introduce someramente diversas herramientas CASE que ofrecen un soporte a esta metodología; finalmente, la Sección 5 cierra el artículo con las conclusiones del mismo.

## 2. Visión General de la MDB

En este apartado se va a hacer una revisión de los aspectos más destacados de esta metodología con el objeto de que el lector de este artículo conozca sus fundamentos, para una información más detallada de la misma se recomienda la consulta de [3][5][6]. Las tareas propuestas en esta metodología son:

1. Obtener información sobre el dominio del problema y el sistema actual.
2. Preparar y realizar las reuniones de obtención/negociación.
3. Identificar/revisar los objetivos del sistema.
4. Identificar/revisar los requisitos de información.
5. Identificar/revisar los requisitos funcionales.
6. Identificar/revisar los requisitos no funcionales.
7. Priorizar objetivos y requisitos.

Por parte de los autores de esta metodología se recomiendan diversas técnicas para llevar a cabo las tareas anteriores, pero las más importantes son los casos de uso, las plantillas y los patrones lingüísticos.

Los casos de uso son una técnica sumamente conocida y utilizada en los métodos de desarrollo de software actuales. Esta técnica fue propuesta inicialmente por Ivar Jacobson [11] y actualmente se encuentra incluida dentro de UML (*Unified Modeling Language*) [1].

Para la correcta descripción de los casos de uso (que son una forma de expresar requisitos funcionales), los requisitos no funcionales, los requisitos de información, así como de otros requisitos del sistema, se recurre a la utilización de diversos tipos de plantillas, concretamente:

- *Objetivos del sistema.*
- *Requisitos de información.*
- *Requisitos de restricción (reglas de negocio)*
- *Actores.*
- *Casos de uso.*
- *Requisitos funcionales (expresados de forma tradicional, como texto libre).*
- *Requisitos no funcionales.*

Cada plantilla presenta los campos de información necesarios para especificar el concepto que está representando, como ejemplo de plantilla en la Figura 2 se recoge la plantilla utilizada por esta metodología para los casos de uso, que describen requisitos funcionales de una manera operacional.

## 3. Evolución del Uso de la MDB en la Universidad de Salamanca

Los estudios de Informática en la Universidad de Salamanca se organizan en un primer ciclo de tres años de duración (Ingeniería Técnica en Informática de Sistema) y en un segundo ciclo de dos años de duración (Ingeniería Informática).

Los tópicos relacionados con los requisitos se introducen en la asignatura de Ingeniería del Software, que aparece en el Plan de Estudios de la Ingeniería Técnica en su tercer año. La superación de la parte práctica de esta asignatura obliga a que los alumnos, organizados en grupos, realicen la especificación de un sistema a nivel de análisis y diseño. Esta especificación debe incluir un catálogo de requisitos, pero los estudiantes pueden elegir el método de especificación.

La MDB se introdujo en el temario de la asignatura de Ingeniería del Software en la Universidad de Salamanca en el curso 1998-1999. Desde el momento de su introducción, su uso global en las prácticas de la asignatura sólo fue menor al uso de otros métodos de especificación durante el primer curso, incrementando su presencia en años sucesivos hasta alcanzar un porcentaje de utilización del 85,4% en el curso 2002-2003.

La Figura 3a compara el número de trabajos obligatorios que, dentro de la asignatura de Ingeniería del Software, han utilizado esta metodología de obtención de requisitos frente a otros métodos, típicamente el IEEE Std. 830 [10]. Por su parte, la Figura 3b realiza la misma comparación que la Figura 3a, pero expresando dicha comparativa en términos porcentuales.

La Ingeniería Técnica en Informática de Sistemas finaliza con un PFC. Normalmente, estos proyectos requieren la realización de una documentación técnica que incluirá en alguno de sus anexos un catálogo de requisitos. La utilización de la MDB comienza a aparecer en la documentación de estos proyectos en el año 2000. Durante el año 2000, 17 de los 38 PFC defendidos en la Ingeniería Técnica utilizaron la MDB, esto supone una presencia en el 44,7% de los proyectos leídos en este año. Pero cabe destacar el espectacular incremento de uso de esta metodología en los siguientes años, alcanzando una cota máxima de utilización durante el año 2001 con una presencia en el 84,4% de los PFC defendidos en la Ingeniería Técnica. Esta tendencia se ve refrendada en el año 2003, donde el 88,9% de los proyectos leídos hasta la fecha utilizan este método (actualmente, sólo se tienen datos de la convocatoria de febrero de 2003).

| | | |
|---|---|---|
| **UC-*<id>*** | ***<nombre/objetivo del caso de uso>*** | |
| **Versión** | *<nº de la versión actual>* (*<fecha de la versión actual>*) | |
| **Autores** | • *<autor$_1$ de la versión actual>* (*<organización del autor$_i$>*)<br>… | |
| **Fuentes** | • *<fuente$_1$ de la versión actual>* (*<organización de la fuente$_i$>*)<br>… | |
| **Dependencias** | • OBJ-*x* *<nombre del objetivo del que depende este caso de uso>*<br>• R*x-y* *<nombre del requisito del que depende este caso de uso>*<br>… | |
| **Descripción** | El sistema deberá comportarse tal como se describe en el siguiente caso de uso { abstracto durante la realización de los siguientes casos de uso: *<lista de casos de uso>*, cuando *<evento de activación>* [o durante la realización de los siguientes casos de uso: *<lista de casos de uso>*]} | |
| **Precondición** | *<precondición del caso de uso>* | |
| **Secuencia normal** | **Paso** | **Acción** |
| | $p_1$ | {El actor *<actor>*, El sistema} *<acción/es realizada/s por actor/sistema>* |
| | $p_2$ | Se realiza el caso de uso *<caso de uso (UC-x)>* |
| | $p_3$ | Si *<condición>*, {el actor *<actor>*, el sistema} *<acción/es realizada/s por actor/sistema>* |
| | $p_4$ | Si *<condición>*, se realiza el caso de uso *<caso de uso (UC-x)>* |
| | … | … |
| **Poscondición** | *<poscondición del caso de uso>* | |
| **Excepciones** | **Paso** | **Acción** |
| | $p_i$ | Si *<condición de excepción>*, {el actor *<actor>*, el sistema} *<acción/es realizada/s por actor/sistema>*, a continuación este caso de uso {continúa, queda sin efecto} |
| | $p_j$ | Si *<condición de excepción>*, se realiza el caso de uso *<caso de uso (UC-x)>*, a continuación este caso de uso {continúa, queda sin efecto} |
| | … | … |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | $p_i$ | *m <unidad de tiempo>* |
| | … | … |
| **Frecuencia** | *<nº de veces>* veces / *<unidad de tiempo>* | |
| **Importancia** | *<importancia del caso de uso>* | |
| **Urgencia** | *<urgencia en la implementación del caso de uso>* | |
| **Estado** | *<estado de desarrollo del caso de uso>* | |
| **Estabilidad** | *<estimación de la estabilidad del caso de uso>* | |
| **Comentarios** | *<comentarios adicionales sobre el caso de uso>* | |

Figura 2. Plantilla para casos de uso

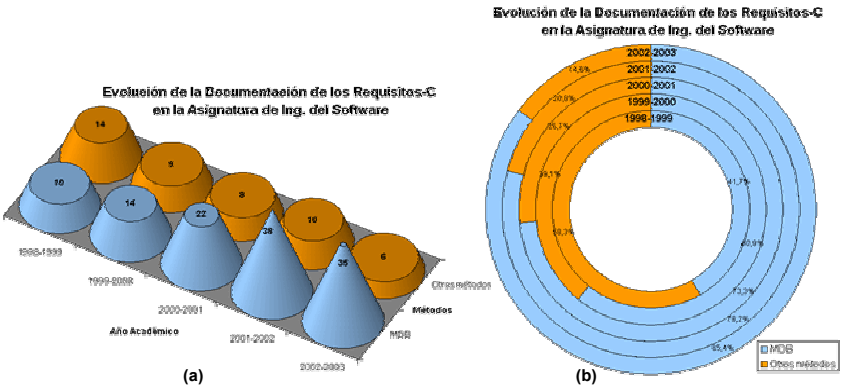


Figura 3. Comparación del Nº de trabajos de Ing. del Software que han empleado la MDB entre 1998-99 y 2002-03

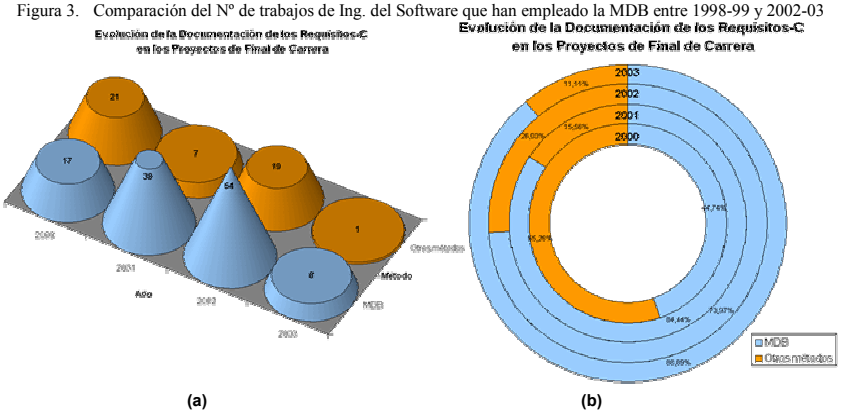


Figura 4. Comparación del Nº de PFC que han empleado la MDB entre los años 2000 y 2003

La Figura 4 presenta la misma comparación realizada con los trabajos prácticos de la asignatura de Ingeniería del Software, pero en este caso orientada a los PFC de la Ingeniería Técnica en Informática de Sistemas defendidos entre 2000 y 2003.

## 4. Soporte CASE para la MDB

Desde el curso académico 2001-2002 la metodología de obtención y documentación de requisitos está soportada por herramientas CASE.

En esta sección se van a presentar tres herramientas CASE que dan cobertura a esta metodología. La primera de ellas, denominada REM, ha sido desarrollada en la Universidad de Sevilla por uno de los autores de la metodología. Las otras dos han sido desarrolladas en el Departamento de Informática y Automática como soporte para las prácticas de la asignatura Ingeniería del Software y para los PFC.

### 4.1. REM

REM (*REquirements Manager*) [4] es una

herramienta de gestión de requisitos experimental desarrollada por uno de los autores como parte de su tesis doctoral [3] y recientemente presentada en la sesión de *research tools* de la IEEE *Joint International Conference on Requirements Engineering* celebrada en Eseen (Alemania), en septiembre de 2002 [7]. En REM, un proyecto de Ingeniería de Requisitos está compuesto por los cuatro documentos correspondientes a las cuatro vistas que pueden verse en la figura 5: requisitos (elicitación), modelos conceptuales (análisis), conflictos y defectos (análisis, verificación, validación y negociación), y peticiones de cambio (gestión).

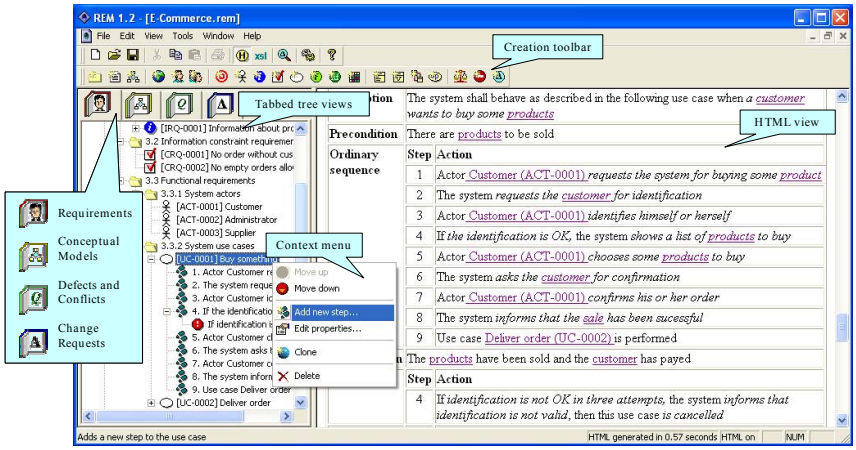


Figura 5. Interfaz de usuario de REM

REM presenta un excelente generador de informes en HTML, para lo cual internamente maneja la información en XML, aplicándole una hoja de estilo XSLT a la hora de generar el correspondiente informe. Las hojas de estilo XSLT son configurables, por lo que permite la visualización de la información de los requisitos de acuerdo a diversos estándares, incluyendo la posibilidad de aplicar heurísticas de verificación automática, tal como se describe en [7]. La Figura 5 presenta la interfaz de REM, que soporta tanto español como inglés.

### 4.2. GESCAT

GESCAT (http://tejo.usal.es/~fgarcia/) es una herramienta CASE que ofrece un soporte a la realización de las plantillas propias de la MDB.

Esta herramienta no pretende cubrir toda la funcionalidad incluida en REM, sino más bien ser una herramienta de uso docente, que facilite la labor a los alumnos de ITIS en el proceso de gestionar y documentar requisitos.

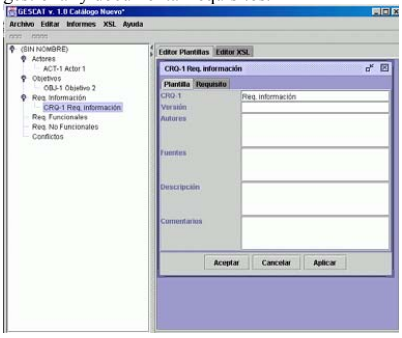

Figura 6. Interfaz de usuario de GESCAT

Esta herramienta se ha desarrollado en Java buscando una independencia de plataforma. Además, y en relación con la posibilidad de generación de informes, GESCAT hace un

profuso empleo de los lenguajes de marcado XML/XSLT, que facilita generar los informes en diferentes formatos como PDF o HTML. La Figura 6 muestra la interfaz de usuario de esta herramienta.

### 4.3. Left CASE

Left-CASE (http://zarza.usal.es) es una plataforma CASE para entornos de escritorios GNOME 1.4 basada en componentes, que ofrece un amplio soporte a las técnicas de modelado del software que se estudian en la Ingeniería Informática, fundamentalmente en las disciplinas relacionadas con la Ingeniería del Software.

Left CASE está basada en un entorno CASE extensible, donde las técnicas que soporta pueden ampliarse, pero siempre bajo las restricciones de compatibilidad con el entorno.

Para lograr este objetivo se diseña una arquitectura formada por un *framework* base o contenedor y un conjunto de componentes específicos. El contenedor hace las veces de "anfitrión" para los componentes, ofreciéndoles los servicios comunes necesarios, así como el conjunto de interfaces necesarias para que dichos componentes puedan incorporarse al entorno construido, mientras que cada uno de los componentes debe proporcionar todas las características propias de una técnica de modelado concreta [9].

Uno de los componentes desarrollados para integrarse en la plataforma Left CASE es el componente para el modelado de diagramas de casos de uso de acuerdo con UML [1], que para la especificación de cada caso de uso emplea plantillas compatibles con las definidas en la MDB. Dado que Left CASE está desarrollada para entorno GNOME 1.4, la plataforma de ejecución por excelencia será Linux. La interfaz de esta herramienta soporta tanto español como inglés, y en la Figura 7 se aprecia la interfaz de Left CASE con el componente para el modelado de casos de uso cargado.

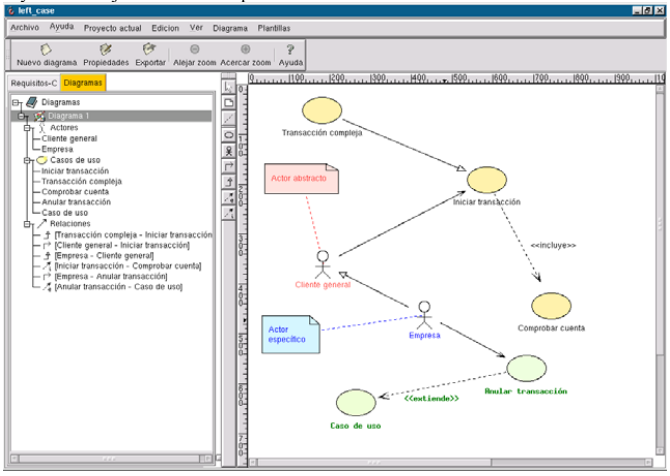


Figura 7. Interfaz de usuario de Left CASE con el Componente CU

## 5. Conclusiones

Este artículo analiza el uso de una metodología concreta para la obtención y documentación de requisitos en la titulación de Ingeniería Técnica en Informática de Sistemas en la Universidad de Salamanca. Dicha metodología ha sido definida dentro del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Sevilla.

La metodología en cuestión se basa en una unidad básica de documentación, las plantillas, que se emplean para especificar los diferentes

conceptos que se manejan en la fase de Ingeniería de Requisitos, como por ejemplo objetivos, casos de uso, requisitos funcionales, etc.

Las plantillas utilizan el lenguaje natural como base de su contenido, haciéndolas muy comprensibles tanto a los usuarios de las mismas como a sus destinatarios.

Gracias a su sencillez, este método resulta más familiar y fácil de aplicar y comprender que otras propuestas más complejas como por ejemplo el IEEE Std. 830 [10]. La simplicidad de este método es el punto clave con el que se puede justificar la gran acogida que ha tenido por parte de los alumnos. Esta aceptación queda reflejada en las Figuras 3 y 4 que muestran cómo esta metodología está siendo ampliamente utilizada tanto en los trabajos prácticos de la asignatura Ingeniería del Software como en los PFC.

Su utilización en la asignatura Ingeniería del Software es un resultado que no resulta sorpresivo dado que esta metodología aparece citada en diversas ocasiones a lo largo del programa impartido en la parte teórica de la asignatura (aunque también se menciona y se describe someramente el IEEE Std. 830). Sin embargo, su amplia utilización en los PFC si que representa un claro síntoma de aceptación, porque en dichos proyectos las influencias de los tutores son diversas y el alumno tiene plena libertad para elegir las metodologías que mejor se ajusten a la temática de sus trabajos.

La existencia de herramientas CASE compatibles con esta metodología, disponibles y de libre uso, es sin duda alguna otra ventaja añadida para el éxito y aceptación de esta propuesta metodológica.

Desde el punto de vista de su utilización en docencia, la mayor fuente de errores que comenten los alumnos es su falta de rigor a la hora de rellenar los campos de la plantilla relacionados con el Rendimiento, la Frecuencia, la Importancia y la Urgencia, fundamentalmente debido a que sus proyectos tienen tamaños limitados, donde no les resulta obvio aportar dichos datos.

## Agradecimientos

Este trabajo ha sido parcialmente subvencionado por la Consejería de Educación y Cultura de la Junta de Castilla y León mediante el proyecto US23/02.

## Referencias

[1] Booch, G., Rumbaugh, J., Jacobson, I. *The Unified Modeling Language User Guide.* Addison-Wesley, 1999

[2] Brackett, J.W. *Software Requirements.* SEI Curriculum Module SEI-CM-19-1.2. Software Engineering Institute. January 1990

[3] Durán, A. *Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información.* Tesis Doctoral, U. de Sevilla, 2000

[4] Durán, A. *REM Web Page.* U. de Sevilla. http://klendathu.lsi.us.es/REM/. 2002

[5] Durán, A., Bernárdez, B. *Metodología para la Elicitación de Requisitos de Sistemas Software (versión 2.3).* Informe Técnico LSI-2000-10, U. de Sevilla. Abril 2002

[6] Durán, A., Bernárdez, B., Ruiz, A., Toro, M. *A Requirements Elicitation Approach Based on Templates and Patterns.* En *Proceedings of WER'99*, Buenos Aires, Argentina. 1999

[7] Durán, A., Ruiz, A., Corchuelo, R., Toro, M.. *Supporting Requirements Verification Using XSLT.* En *Proceedings of the IEEE Joint International Conference on Requirements Engineering*. Essen, Alemania. 2002

[8] European Software Institute. *ESPITI – European User Survey Results*. Informe Técnico ESI–1996–TR95104, 1996

[9] García, F. J., Álvarez, I. *Plataforma CASE Basada en Componentes para la Docencia de Ingeniería del Software.* En las *Actas del IE-2002*. 2002

[10] IEEE. *IEEE Software Engineering Standards Collection 1999 Edition. Volume 4: Resource and Technique Standards*. IEEE Computer Society Press, 1999

[11] Jacobson, I. *Object Oriented Development in an Industrial Environment.* En *Proceedings of the OOPSLA'87*. ACM, 1987

[12] Lethbridge, T.C. *Priorities for the Education and Training of Software Engineers. Journal of Systems and Software,* 53. 2000

[13] Raghavan, S., Zelesnik, G., Ford, G. *Lecture Notes on Requirements Elicitation.* Educational Materials. CMU/SEI-94-EM-10. Software Engineering Institute. March 1994

[14] The Standish Group. TSG. *The CHAOS Report*. 1995