Una propuesta de aprendizaje para el desarrollo orientado a objetos de proyectos software

Mª Carmen Penadés, Eliseo Marzal, Antonio Garrido
Dpto. de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera, s/n. 46022 - Valencia
e-mail: {mpenades, emarzal, agarridot}@dsic.upv.es

Resumen

Actualmente, el perfil profesional de Ingeniero del Software exige unas capacidades básicas que la universidad española debe potenciar. En este artículo se presenta una propuesta de aprendizaje para el desarrollo orientado a objetos de proyectos software como una alternativa flexible para garantizar dichas capacidades. Nuestra propuesta plasma cuáles son las habilidades que se deben satisfacer para cada capacidad, indicando además las actividades básicas para su consecución y cómo realizar su seguimiento. Tras analizar los resultados de una aplicación real de dicha propuesta, y examinar sus ventajas e inconvenientes, podemos afirmar que se trata de una propuesta eficaz y recomendable.

1. Motivación

El desarrollo de proyectos software es una de las competencias esenciales del informático con perfil profesional de Ingeniero del Software [2]. En la universidad española, esta habilidad se desarrolla en las asignaturas pertenecientes a la disciplina de la Ingeniería del Software. Así por ejemplo, en el plan de estudios de la titulación de Ingeniero Técnico en Informática de Gestión (ITIG) ofertada por la Universidad Politécnica de Valencia, UPV, (B.O.E. nº 259 de 17/10/2001), la asignatura de Ingeniería del Software de Gestión describe brevemente su contenido como: *"Diseño, propiedades y mantenimiento del software de gestión. Planificación y gestión de proyectos informáticos. Análisis de aplicaciones de gestión"*. A nuestro entender, la competencia que un alumno, futuro profesional de desarrollo de software, debe poseer es la de ser capaz de participar y desarrollar cualquiera de las actividades implicadas en las fases del ciclo de vida de desarrollo de software [8], mediante el uso de diferentes metodologías y paradigmas de desarrollo. Sin embargo, esta competencia es demasiado genérica y debe concretarse en otras más sencillas [1]:

- Saber realizar eficazmente la administración de proyectos software (gestionando tiempo, recursos, hitos, etc.)
- Saber diseñar la arquitectura del sistema a desarrollar, estableciendo sus elementos e interrelaciones.
- Tener un conocimiento amplio de las técnicas, métodos y herramientas de análisis, diseño y desarrollo.
- Saber realizar una implementación, total o parcial, utilizando distintos lenguajes de programación.
- Saber verificar y validar el producto para la aceptación del cliente, implantarlo y ponerlo en explotación.

La mayoría de las capacidades anteriores tiene un carácter predominantemente práctico. Por lo tanto, la consecución de las mismas tiene importantes implicaciones en la forma de proponer y organizar el proceso de aprendizaje: el alumno debe jugar un papel fundamental, algo cada vez más necesario en el marco de un Espacio Europeo de Educación superior (EEES) [5], y esencial en el aprendizaje del desarrollo de proyectos software.

Basándonos en nuestra experiencia como docentes en asignaturas de la disciplina de Ingeniería del Software [3][4], en este trabajo presentamos una propuesta de aprendizaje para el desarrollo de proyectos software siguiendo el paradigma orientado a objetos (OO). Hemos optado por la orientación a objetos debido a sus claras ventajas: i) mayor facilidad para modelar el

problema, ii) mayor disponibilidad de herramientas en todas las fases del desarrollo, y iii) se ha convertido en un estándar de facto ampliamente aceptado por las universidades y empresas, tanto nacionales como internacionales [1][2][7].

El artículo se organiza de la siguiente forma. El apartado 2 presenta nuestra propuesta de aprendizaje, desglosada en un conjunto de plantillas e indicando cuáles son los criterios de seguimiento y evaluación. En el apartado 3 se expone una aplicación real de dicha propuesta a una asignatura troncal que se imparte actualmente en la UPV. El análisis y discusión de los resultados se presenta en el apartado 4. Finalmente, el apartado 5 expone las conclusiones del artículo.

2. Una propuesta de aprendizaje

Nuestra propuesta de aprendizaje va encaminada a que el alumno sea capaz de realizar, desde un enfoque OO, un proyecto de desarrollo de software. Se hace especial hincapié en las actividades básicas del proceso, sin fijar ningún modelo de proceso específico.

En esta propuesta, el alumno como sujeto activo, debe alcanzar la competencia citada anteriormente. Con este objetivo se han identificado un conjunto de conocimientos y habilidades que los alumnos deben adquirir y que hemos agrupado en capacidades. Se han identificado un total de seis capacidades básicas:

- Saber realizar un plan de proyecto software.
- Saber hacer el modelado conceptual OO de un sistema de información.
- Saber hacer el diseño OO de un sistema de información.
- Saber hacer una implementación OO de un sistema de información.
- Saber diseñar casos de prueba.
- Saber realizar una instalación y puesta en marcha de un producto software desarrollado.

El desarrollo de la propuesta se basa en seguir el esquema: competencia, capacidad, habilidad y actividad. Alcanzar una competencia supone alcanzar las capacidades requeridas, que a su vez implica desarrollar las habilidades asociadas a cada capacidad. Finalmente, se definirán un conjunto de actividades cuya realización ayudará al desarrollo de cada una de las habilidades.

En los siguientes apartados se desarrolla la propuesta, definiendo con mayor detalle cada una de las capacidades básicas identificadas. También se citan los criterios de evaluación, así como la importancia de un proceso de seguimiento.

2.1. Capacidades

Por simplicidad, hemos definido una plantilla que recoge el desarrollo de cada capacidad (ver Tabla 1). La plantilla consta de cuatro apartados. En primer lugar aparece el nombre de la capacidad a alcanzar. En segundo lugar, se indican los conocimientos previos o capacidades requeridas al alumno para poder desarrollar adecuadamente la capacidad en cuestión. El tercer apartado es el más importante pues supone el desglose de la capacidad en un conjunto de habilidades que el alumno debe ejercitar. Finalmente, aparece un apartado de resultados que indica el conjunto de artefactos que el alumno debe ser capaz de producir una vez alcanzada dicha capacidad.

Capacidad n	
Prerrequisitos	Pn.1 . . .
Habilidades	Hn.1
	Hn.2
	. . .
Resultados	

Tabla 1. Plantilla de definición de capacidades[1]

Es importante destacar en este punto que si un alumno no dispone de los conocimientos establecidos como prerrequisitos de una capacidad, existe una carencia. Ésta se debe subsanar para que no se convierta en un impedimento al desarrollar las habilidades específicas de la misma. La situación deseable es que dichas carencias se detecten lo antes posible. La solución más sencilla para que no afecten al proceso de aprendizaje de la capacidad en sí misma es que el prerrequisito pase a ser otra habilidad más a desarrollar.

[1] Tanto las capacidades como los prerrequisitos y habilidades asociadas a cada una de ellas, aparecen etiquetadas con un número para facilitar su identificación y posterior referencia.

Capacidad 1	Saber realizar un plan de proyecto software
Prerrequisitos	- -
Habilidades	H1.1 Conocer en qué consiste la administración de proyectos software y sus principales características.
	H1.2 Conocer las principales tareas que desempeñan los administradores de proyectos software.
	H1.3 Saber utilizar una herramienta de planificación para la elaboración de un plan de proyecto.
Resultados	Diagrama de Gantt, diagrama de recursos y diagrama de costes.

Capacidad 2	Saber hacer el modelo conceptual orientado a objetos de un sistema de información
Prerrequisitos	P2.1 Conocer los fundamentos del paradigma OO. P2.2 Saber pensar y razonar sobre un problema de acuerdo con el paradigma OO.
Habilidades	H2.1 Conocer los beneficios que aporta la construcción de modelos en la resolución de problemas.
	H2.2 Saber razonar sobre el sistema de información, a distintos niveles de abstracción.
	H2.3 Saber diferenciar el modelado conceptual del diseño de sistemas de información.
	H2.4 Conocer cuáles son los elementos esenciales que ha de proporcionar un método de modelado OO.
	H2.5 Conocer al menos una notación de modelado OO.
	H2.6 Saber utilizar una herramienta que dé soporte a la notación de modelado OO vista.
Resultados	Modelos que den cuenta de la estructura, comportamiento y funcionalidad del sistema.

Capacidad 3	Saber hacer un diseño orientado a objetos de un sistema de información
Prerrequisitos	P3.1 Conocer los fundamentos del paradigma OO. P3.2 Saber interpretar un modelo conceptual OO de un sistema de información.
Habilidades	H3.1 Saber construir una solución software a partir de los resultados del modelo conceptual.
	H3.2 Conocer distintos patrones arquitectónicos.
	H3.3 Comprender el diseño software como un conjunto de objetos que interactúan entre ellos.
	H3.4 Saber diseñar un sistema siguiendo una arquitectura de tres capas o niveles.
	H3.5 Saber utilizar una herramienta que dé soporte a la notación de diseño OO vista.
	H3.6 Saber utilizar una herramienta que permita el diseño de la interfaz gráfica.
Resultados	Arquitectura del sistema, diseño de objetos, diseño de mensajes y diseño de la interfaz gráfica.

Capacidad 4	Saber hacer una implementación orientada a objetos de un sistema de información
Prerrequisitos	P4.1 Conocer los fundamentos del paradigma OO. P4.2 Conocer y saber interpretar un diseño OO de un sistema de información.
Habilidades	H4.1 Conocer y saber utilizar un lenguaje OO.
	H4.2 Saber utilizar el polimorfismo y el enlace dinámico/estático en la implementación.
	H4.3 Saber construir programas y razonar sobre la ejecución y terminación de los mismos.
	H4.4 Conocer y saber utilizar un entorno integrado de desarrollo para construir un programa OO.
	H4.5 Saber implementar en un lenguaje de programación OO un diseño de objetos.
	H4.6 Saber implementar una arquitectura de tres niveles.
	H4.7 Saber razonar por analogía para entender y reutilizar código dado.
Resultados	Producto software.

Capacidad 5	Saber diseñar casos de prueba
Prerrequisitos	P5.1 Saber interpretar un modelo conceptual OO de un sistema de información. P5.2 Saber interpretar un diseño OO de un sistema de información.
Habilidades	H5.1 Conocer los principales tipos de casos de prueba.
	H5.2 Saber detectar los casos de prueba para el sistema de información.
	H5.3 Saber preparar los datos de prueba y extraer conclusiones en base a los resultados de las pruebas.
Resultados	Casos de prueba.

Capacidad 6	Saber realizar una instalación y puesta en marcha de un producto software desarrollado
Prerrequisitos	P6.1 Conocer el productos software desarrollado.
Habilidades	H6.1 Saber definir y configurar la instalación de un producto software.
	H6.2 Saber hacer un manual de usuario y un manual de instalación.
Resultados	Instalación del producto software, manual de usuario y de instalación.

Tabla 2. Capacidades a garantizar en nuestra propuesta

La Tabla 2 muestra la definición de todas las capacidades básicas identificadas en la propuesta, utilizando la plantilla definida en la Tabla 1. Para un mejor entendimiento de los prerrequisitos y habilidades que se detallan, debemos aclarar que se asume un conocimiento de los fundamentos del paradigma OO, previo al desarrollo de las capacidades básicas. Esto se ve reflejado, por ejemplo en la capacidad 2. Sus prerrequisitos son: conocer los fundamentos del paradigma OO (P2.1) y saber pensar y razonar sobre un problema de acuerdo con el paradigma OO (P2.2).

A partir de la información recogida en la Tabla 2, sólo falta definir el conjunto de actividades concretas que cada docente realizará para conseguir que los alumnos adquieran las capacidades requeridas para el desarrollo de proyectos software. Este paso se traduce en la aplicación de la propuesta a un contexto específico que dependerá de cada Universidad, de sus planes de estudios y de los propios docentes. En nuestro caso, en el apartado 3 se detalla su aplicación en una asignatura del actual plan de estudios de la UPV.

Llegados a este punto se plantea una cuestión clave, desde el punto de vista docente: ¿cómo evaluar si un alumno ha alcanzado las capacidades exigidas? La respuesta se basa en considerar dos aspectos: el seguimiento del proceso de aprendizaje y la evaluación del mismo una vez finalizado.

2.2. Seguimiento y evaluación

Desde un punto de vista docente, pensamos que dada una capacidad o conjunto de ellas, no sólo es importante que el alumno finalice su aprendizaje adquiriendo dichas capacidades, sino que también es importante el proceso seguido para alcanzarlas.

Esta cuestión es mucho más relevante en el caso que nos ocupa, el desarrollo de proyectos software. Por ello, en nuestra propuesta, la evaluación se plantea como una evaluación de ambos aspectos. Se mide tanto los resultados parciales que se van alcanzando en el desarrollo de las distintas habilidades como el resultado final alcanzado. La nota final del alumno en el proyecto software es la suma de las notas obtenidas en la consecución de cada capacidad. De esta forma, al resultado final, es decir, al producto software, el docente le asigna una nota que no es más que otra nota a considerar en el cálculo final. Se les pueden asignar distintos porcentajes a las notas obtenidas por capacidad, en función de las actividades concretas que se realicen y el tiempo que se dedique a las mismas.

En cuanto a la forma de evaluar cada capacidad, de forma general podemos decir que consiste en la realización de una prueba objetiva que demuestre la consecución de la misma. Sin embargo, una gran mayoría de habilidades tienen que ver con la construcción de un cierto artefacto software. Por ello, la evaluación propuesta no consiste sólo en la realización de pruebas objetivas que midan el conocimiento teórico, sino también el práctico.

La evaluación se realizará atendiendo a dos criterios: conocimientos teóricos y conocimientos prácticos asociados a cada capacidad. Existirán dos tipos de pruebas para medir dichos conocimientos: i) el examen teórico que, como su nombre indica, medirá la parte más teórica de la capacidad y podrá ser oral o escrito; y ii) los entregables, que servirán para evaluar la parte más práctica y se corresponderán con el artefacto o conjunto de artefactos que se deban desarrollar dentro de cada capacidad. Ambas pruebas serán evaluadas por el docente y puesto que los aspectos son complementarios, el peso de cada uno de ellos en la evaluación de cada capacidad debería ser del 50%. No obstante, se puede variar su peso, atendiendo a condicionantes concretos, que deberá plasmar cada propuesta. En nuestro caso, en el siguiente apartado, también se detallarán los pesos específicos que se han aplicado al seguir este método de evaluación, basado en nuestra experiencia anterior en otras asignaturas [6].

3. Una aplicación real: ISG

Esta propuesta de aprendizaje se está aplicando actualmente a la asignatura de Ingeniería del Software de Gestión (ISG). Es una asignatura troncal que se imparte en 3º curso de la titulación de ITIG. Su carga lectiva es de 12 créditos, repartidos en 6 créditos teóricos y 6 de prácticas de laboratorio. La asignatura es anual y su organización docente son 4 horas semanales (2 horas de teoría y 2 horas de prácticas). La asignación docente actual es de 2 profesores en teoría y 4 profesores en prácticas. En las clases teóricas, el método docente es lección magistral y

realización de problemas en el aula. En cuanto a las prácticas, éstas se realizan por parejas y cada grupo de laboratorio tiene un máximo de 38 alumnos.

Una descripción detallada de los contenidos teóricos y prácticos de ISG queda fuera del alcance de este artículo. Lo que exponemos a continuación es cómo se ha aplicado la propuesta de aprendizaje en esta asignatura, entre cuyos objetivos está el desarrollo de proyectos software.

En primer lugar, hemos clasificado las habilidades en función de un mayor contenido teórico o práctico, entendido este último como una necesidad de disponer y utilizar herramientas de apoyo para su consecución. Esta separación nos ha permitido ubicar las habilidades como parte de los contenidos vistos en las sesiones de teoría o de prácticas. Es necesario resaltar que en algunos casos la separación no ha sido trivial, puesto que en el desarrollo de software ambos conceptos guardan una estrecha relación.

La Tabla 3 muestra el resultado de dicha separación, y en ella podemos ver la correspondencia entre las habilidades y las actividades que las desarrollan. Por ejemplo, las habilidades 1 a 5 relacionadas con la capacidad 2 (H2.1 – 2.5) se desarrollan en las clases teóricas, como parte de los contenidos del Tema 4 (Modelado OO) mientras que la habilidad 6 (H2.6) se adquiere en la Práctica 6 de tres sesiones (P6-A, P6-B y P6-C).

Esta distribución de habilidades entre sesiones de teoría y prácticas supone cumplir la siguiente restricción: debe existir una sincronización temporal entre ambas sesiones. Antes de cualquier sesión de prácticas se ha visto la teoría necesaria.

En el resto del apartado nos vamos a centrar en explicar con detalle lo que consideramos de más interés en la aplicación de la propuesta: la organización de las clases prácticas, los entregables que se realizan y el porcentaje de cada uno de ellos sobre la nota final.

Durante el primer cuatrimestre de la asignatura, se realizan un total de 12-13 sesiones de prácticas (P1-P4) encaminadas a que los alumnos desarrollen habilidades directamente relacionadas con la programación OO, y el entorno de programación escogido (en nuestro caso *Microsoft Visual C# .NET,* por tratarse del entorno que posteriormente se utilizará en las asignaturas de la intensificación de Ingeniería del Software [9]). Estas sesiones también sirven para que los alumnos afiancen conocimientos ya adquiridos en otras asignaturas (prerrequisitos). En concreto, las habilidades H4.3 y H4.4 se ejercitarían en estas prácticas.

Habilidad	Actividad
H1.1 – 1.2	T3.Administración de Proyectos
H1.3	*P5. Planificación*
H2.1 – 2.5	T4. Modelado OO
H2.6	*P6-A. Diagrama de Clases* *P6-B. Casos de uso y Escenarios* *P6-C. Diagramas de Secuencia*
H3.1 – 3.4	T5. Diseño OO
H3.1, H3.5	*P7-A. Diseño de Clases*
H3.4	*P7-B. Diseño Arquitectónico*
H.3.6	*P7-C. Diseño IGU*
H4.1 – 4.2	T1. Programación OO
H4.3 – 4.4	*P1, P2, P3, P4. Programación OO*
H4.5 – 4.7 H5.1 – 5.3	*P8. Implementación*
H5.3, H6.1	*Entrega producto SW*
H6.2	- -

Tabla 3. Desarrollo de habilidades

El desarrollo del proyecto software se realiza durante el segundo cuatrimestre, y es durante este tiempo cuando realmente el alumno desarrolla las capacidades básicas identificadas en nuestra propuesta. Se les propone un caso de estudio que describe un cierto sistema de información, y los alumnos desarrollan las habilidades propuestas realizando una serie de prácticas que detallamos a continuación. Para ello se dispone de aproximadamente 13-14 sesiones.

En la primera práctica de este cuatrimestre (P5), los alumnos realizan la planificación del caso de estudio utilizando *MS Project*, analizando la viabilidad del proyecto con los recursos disponibles y costes obtenidos. Para esta práctica se dispone de 2 horas y el entregable es voluntario.

La siguiente práctica (P6) se dedica al modelado conceptual. La duración es de 6 horas y se utiliza como herramienta *Rational Rose*. Esta

práctica consta de tres sesiones (P6-A, P6-B y P6-C). En concreto, en la primera sesión (P6-A) los alumnos realizan el diagrama de clases en UML. En la segunda sesión (P6-B), realizan los casos de uso y los escenarios. Por último, en la tercera sesión (P6-C) se realizan los diagramas de secuencia. La valoración de esta práctica tiene un peso del 30% sobre la nota final.

Las cuatro siguientes sesiones se dedican al diseño de la aplicación. Las 8 horas se estructuran en tres sesiones (P7-A, P7-B y P7-C). La primera sesión (P7-A) se dedica al diseño de clases y atributos, disponiendo de 3 horas. En la segunda sesión (P7-B) se realiza el diseño arquitectónico, y más concretamente la capa de persistencia del sistema (el tiempo asignado es de 3 horas). Por último, en la tercera sesión (P7-C) se realiza el diseño de la interfaz gráfica (2 horas). El bloque de diseño de la aplicación tiene un peso del 30%.

A continuación, se realiza la implementación de los escenarios necesarios para el correcto funcionamiento de la aplicación (P8). Se dispone de 10 horas (5 sesiones) y se establece un hito en la tercera sesión con los escenarios que deberían estar implementados hasta ese momento. Al finalizar las 5 sesiones se entrega la aplicación. Esta práctica tiene un peso del 40% en la nota del caso de estudio.

El profesor evalúa cada entregable, comunicando la nota a los alumnos y proporcionando una realimentación de los errores cometidos. Además de la clara ventaja de aprender de sus propios errores, los alumnos conocen en todo momento la parte de prácticas que tienen superada.

Tras la última sesión de prácticas se realiza la prueba y evaluación de la aplicación entregada. El profesor plantea una serie de casos de prueba para evaluar el correcto funcionamiento de la aplicación. Cada grupo de trabajo dispone de 15 minutos para probar su aplicación junto con el profesor.

En resumen, el alumno realiza un entregable voluntario (el de la práctica P5) y un total de 7 obligatorios (los correspondientes a las prácticas P6, P7 y P8).

4. Análisis y discusión de resultados

El objetivo de este apartado es realizar un análisis de las impresiones de la propuesta de aprendizaje presentada en este artículo. En primer lugar, realizaremos un análisis cuantitativo basándonos en algunos de los resultados obtenidos en la aplicación de nuestra propuesta en la asignatura de ISG. Concretamente, analizaremos el grado de adquisición de las capacidades propuestas.

En la Figura 1 se muestran los primeros resultados de este análisis. Para ello, se ha comparado el número de alumnos que han adquirido las habilidades a desarrollar (considerados aptos) en cada uno de los 7 entregables obligatorios de las prácticas P6, P7 y P8, frente a los alumnos que no las adquirieron (no aptos) sobre una muestra total de 162 alumnos.

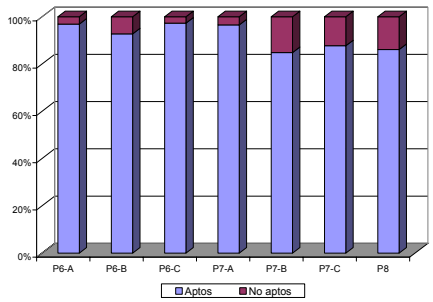

Figura 1. Análisis de resultados de los entregables obligatorios

Como puede observarse en la Figura 1, los resultados son bastante elocuentes. La inmensa mayoría (porcentajes superiores al 80-90%) de los alumnos demostraron las habilidades requeridas, obteniendo una calificación media de 7.12 (con una desviación estándar de 1.58). La calificación más baja, y por tanto la habilidad que les resultó más costosa fue la de "saber diseñar un sistema siguiendo una arquitectura de tres capas" (P7-B). Hasta cierto punto, esto es razonable pues para los alumnos resulta totalmente nuevo el planteamiento de la arquitectura de un desarrollo software. Por el contrario, la calificación más alta, y por tanto las habilidades que les resultaron más sencillas de asimilar fueron la de "saber construir una solución software a partir del modelo conceptual" y "saber utilizar una herramienta que dé soporte al diseño OO" (P7-A). Esto pone de manifiesto que la elección de una metodología OO supone un acierto a medio-largo plazo, tanto por su importante repercusión en el ámbito profesional como en el ámbito docente.

En la Figura 2 se presenta la distribución de calificaciones. En este caso, se indican los porcentajes de alumnos que han obtenido las distintas calificaciones tras realizar las pruebas para evaluar sus habilidades teóricas y prácticas. En dicha figura se observa cómo claramente las capacidades prácticas se asimilan mejor, obteniéndose mejores calificaciones. Concretamente, en las capacidades prácticas el índice total de alumnos aptos fue del 90.71%, frente al 68.52% que fue el índice en las capacidades teóricas. Además, también es significativo que el 65% de los alumnos obtuvo una calificación superior al aprobado en las pruebas de habilidades prácticas, mientras que este porcentaje fue sólo del 37% en las pruebas teóricas. De nuevo, esto pone de manifiesto que en una propuesta de aprendizaje para el desarrollo de proyectos software, fomentar el trabajo práctico resulta esencial.

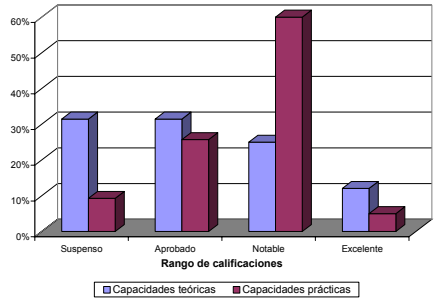

Figura 2. Comparación en el rango de calificaciones de las habilidades teóricas y prácticas

Si realizamos un análisis cualitativo de nuestra propuesta de aprendizaje, podemos diferenciar una serie de ventajas e inconvenientes. Las ventajas más destacadas son:

- Existe una realimentación del alumno al ir conociendo su evolución (y resultados de las prácticas) de forma periódica. Esto favorece e incrementa la participación del alumno en el proceso de aprendizaje, haciendo que su comportamiento sea más activo y dinámico.
- Los alumnos aprenden más y mejor. Este modo de trabajo les permite asimilar más fácilmente los conceptos necesarios para alcanzar sus capacidades, además de hacerlo de una forma eficiente.
- Los alumnos aprenden en un escenario más realista, donde existe una mayor interacción con el profesor (que actúa en el rol de cliente).
- Los alumnos conocen de primera mano el proceso de desarrollo de un proyecto software. La puesta en práctica de las actividades básicas, trabajar en equipo, respetar plazos de entrega, conseguir objetivos, obtener beneficios (no económicos pero sí en cuanto a su calificación), etc., les permitirá convertirse en futuros profesionales más capacitados.

No obstante, también hay que ser realista en la definición de la propuesta y reconocer la existencia de algunas desventajas:

- El profesor se enfrenta a una inercia importante en la forma de trabajar del alumno. Tradicionalmente, el alumno no suele jugar un papel tan activo en su proceso de aprendizaje y poner en marcha esta propuesta resulta un poco duro (e incluso frustrante algunas veces), especialmente en sus primeras etapas.
- El seguimiento y evaluación del proceso de aprendizaje en esta propuesta representa una ardua tarea para el profesor. La realización de muchas prácticas con entregables supone una sobrecarga importante en las tareas de corrección.
- Proponer una planificación tan estricta al alumno dificulta, algunas veces, el desarrollo de su capacidad de planificación. De hecho, se ha dado el caso de que prácticas con un plazo de entrega más estricto tienen un mayor porcentaje de alumnos presentados, mientras que aquéllas en las que los plazos son más relajados (sin tantos entregables intermedios), el porcentaje de alumnos presentados es menor. Por lo tanto, es importante que el profesor proporcione actividades al alumno para fomentar su *autoplanificación*.

Finalmente, teniendo en cuenta las ventajas, inconvenientes, apreciaciones personales tras su aplicación real y resultados anteriores, consideramos que esta propuesta de aprendizaje resulta una alternativa altamente recomendable

para el aprendizaje del desarrollo OO de proyectos software. El estudiante adquiere conocimientos básicos, sobre los cuales, posteriormente puede ir asimilando más fácilmente conocimientos más específicos (dependientes del proceso seguido y de la metodología y/o tecnología utilizada) que cursará en otras asignaturas relacionadas con la materia de ingeniería del software.

5. Conclusiones

En este artículo hemos presentado con detalle una propuesta para facilitar el aprendizaje del desarrollo OO de proyectos software. La viabilidad de esta propuesta viene contrastada por los resultados obtenidos y por este motivo sigue actualmente en curso. Una clara ventaja de esta propuesta es que resulta flexible y abierta a cambios. Por ejemplo, en un curso académico como el actual en el que se disponen de menos semanas lectivas, su adaptación es sencilla, permitiendo una aplicación prácticamente directa que no conlleva serios problemas.

Las prácticas propuestas permiten desarrollar correctamente las habilidades necesarias. Esto permite cumplir las expectativas de lo que el mercado laboral espera de los futuros ingenieros del software. Adicionalmente, la forma en la que se desarrollan dichas prácticas se aproxima a la idea de un marco de educación superior (EEES): prácticas en grupos, mayor independencia por parte del alumno, autosuficiencia en el proceso de aprendizaje, mejor seguimiento (entregables realizados) y una realimentación constante de los resultados obtenidos (corrección y comentarios de los mismos). No obstante, es importante resaltar que para una mejor aplicación de esta propuesta, el ratio profesor-alumnos debería ser mayor, contando con una menor cantidad de alumnos por profesor.

Como ampliación a corto plazo, se desea desarrollar una plataforma común de comunicación para fomentar el trabajo colaborativo entre alumnos-profesor e intercambiar información y conocimientos.

Referencias

[1] ACM-IEEE. *Computing Curricula 2001*. En: http://www.computer.org/education/cc2001.

[2] Agencia Nacional de Evaluación de la Calidad y Acreditación (ANECA). *Libro Blanco del Título de grado en Ingeniería Informática*. En: http://www.aneca.es/modal_eval/conver_docs_titulos.html

[3] Canós, J.H.; Penadés, M.C.; Pelechado, V.; Sánchez, J. *La Ingeniería del Software en los planes de estudio de la Escuela Universitaria de Informática de la UPV*. Actas de las II Jornadas de Ingeniería del Software (JIS'97), 1997.

[4] Canós, J.H.; Penadés, M.C.; Pelechado, V.; Sánchez, J.; Ramos, I.; Gonzalez, A. *La Ingeniería del Software en los planes de estudio: ¿dos perspectivas de una disciplina o más de lo mismo?*. Actas de las IV Jornadas de Enseñanza Universitaria de la Informática (Jenui'98), 1998.

[5] European Ministers of Education, *The European Higher Education Area - Bologna Declaration*, Bologna on the 19th of June 1999.

[6] Garrido, A.; Penadés, M.C.; Pelechano, V. *Un modelo de evaluación de prácticas en Laboratorio de Ingeniería del Software.* Actas de las VII Jornadas de Enseñanza Universitaria de la Informática (Jenui 2001), 2001.

[7] Marcos, E.; Pérez-Chirinos, C. *La orientación a objetos hoy*. Monografía revista Novática, 143, 2000.

[8] Thayer, R.H. *Software System Engineering: A Tutorial. Computer*, 35(4), pp. 68-73, 2002.

[9] ETSIA. *Titulaciones Oficiales. Plan 2001 (Intensificación en Ingeniería del Software).* En: http://www.eui.upv.es/webei/la_escuela/titulaciones/ITIG01/optativas.php#INS