Actas de las XX JENUI. Oviedo, 9-11 de julio 2014
ISBN: 978-84-697-0774-6
Páginas: 133-140

Aprendizaje Basado en Proyectos usando metodologías ágiles para una asignatura básica de Ingeniería del Software

Alfredo Goñi, Jesús Ibáñez, Jon Iturrioz, José Ángel Vadillo
Dpto. de Lenguajes y Sistemas Informáticos
Universidad del País Vasco
San Sebastián
(alfredo, jesus.ibanez, jon.iturrioz, vadillo)@ehu.es

Resumen

En este trabajo se presenta el proyecto docente de la asignatura Ingeniería del Software I (segundo curso, segundo cuatrimestre) siguiendo la metodología del Aprendizaje Basado en Proyectos (ABP) utilizando metodologías ágiles. Nuestra hipótesis se basa en que estas metodologías recogen en su propia esencia los fundamentos del ABP, y por tanto, su correcta utilización garantizan los beneficios de este enfoque. Para su implementación, proponemos el desarrollo de un proyecto software siguiendo el proceso unificado de desarrollo software (UP) utilizando SCRUM como metodología ágil. El enfoque iterativo e incremental de esta metodología nos permite presentar de manera natural conceptos más complejos a medida que va evolucionando el proyecto. El desarrollo del proyecto se realizará en tres iteraciones o *sprints*, donde en cada iteración se van añadiendo nuevos requisitos que implican el estudio de nuevos objetivos de aprendizaje. La realización del proyecto comienza prácticamente el primer día de clase, y su evaluación supone el 75% de la calificación final. Para finalizar, presentamos los resultados y conclusiones obtenidas utilizando esta metodología después del primer año de su implementación.

Abstract

This paper presents the teaching project for the subject Software Engineering I (second year, second semester) based on Project-Based Learning (PBL) methodology. For its implementation, we propose the use of agile techniques widely applied in the software development industry. We believe that agile methodologies do capture the very principles of PBL, and its use can therefore guarantee the acquisition of the proposed learning goals. During the course, students will develop a project using the unified software development process, or simply Unified Process (UP), encompassed in the agile methodology SCRUM. The elaboration of the project will be undertaken in teams that will need to explore the appropriate modelling tools and implementation technologies along three successive iterations or sprints. The project will be in place almost from the first day of class, and its assessment will compute for a 75% of the final grade. Additionally, the iterative structure of the project development will entail an incremental learning framework that results especially useful to assimilate the concepts involved in Software Engineering.

Palabras clave

Ingeniería del Software, Aprendizaje Basado en Proyectos, metodologías ágiles, SCRUM.

1. Introducción

La Facultad de Informática de la UPV/EHU ha desarrollado una serie de líneas estratégicas para incorporar a la oferta docente de los nuevos Grados. Así se ha puesto en marcha un programa específico de implantación de asignaturas cuya docencia se encuadre en el paradigma del Aprendizaje Basado en Proyectos (ABP) [1]. El objetivo es que los estudiantes puedan desarrollar una asignatura por cuatrimestre siguiendo dicha metodología.

La asignatura objeto de este artículo, Ingeniería del Software I, es de las que podemos llamar tempranas (se imparte en el cuarto cuatrimestre del Grado en Ingeniería Informática en la UPV/EHU) y cortas (cuenta con seis créditos). El estudiante ha cursado previamente tres asignaturas del área de programación: un curso de Programación Básica de iniciación a la programación imperativa (1er cuatrimestre), otro de Programación Modular y Orientación a Objetos (2º cuatrimestre) y otro de Estructuras de Datos y Algoritmos (3er cuatrimestre). En el momento de llegar a Ingeniería del Software I el estudiante ya cuenta con unas habilidades de

programación en Java razonablemente desarrolladas, dominio del entorno de programación Eclipse y nociones básicas de diseño de algoritmos eficientes.

Se pretende que al completar la asignatura de Ingeniería del Software I se haya producido un salto cualitativo, en el sentido de que el estudiante sea capaz de abordar y construir una aplicación siguiendo los principios de Análisis y Diseño Orientado a Objetos, utilizando de manera provechosa diversas herramientas de modelado e implementando su solución mediante una arquitectura cliente-servidor en tres niveles y un código que respete los patrones de software más habituales.

En el momento de incorporarse a la asignatura, el estudiante no tiene aún nociones de programación de interfaces de usuario ni de utilización de mecanismos de persistencia. Naturalmente, la programación distribuida también le es ajena. Esto plantea formidables obstáculos de tipo tecnológico, que hemos intentado superar maximizando el número de problemas resolubles con el mínimo esfuerzo de adaptación a las exigencias de las técnicas utilizadas. Como resultado de algunos años de experiencia con asignaturas del área hemos seleccionado un conjunto de soluciones tecnológicas que respondan lo mejor posible a un mismo modelo:

- Lenguaje de programación Java.
- Entorno de programación Eclipse.
- AWT/Swing para el diseño de interfaces gráficas de usuario (GUI).
- Db4o para implementar la persistencia en bases de datos orientadas a objetos.
- RMI para desarrollar aplicaciones distribuidas.

Sin embargo, además del problema tecnológico queda la parte fundamental, es decir la metodológica. También en este caso hemos establecido la metodología de desarrollo y modelado:

- Análisis y diseño orientado a objetos.
- El Proceso Unificado de Desarrollo del Software (UP) [2] como ciclo de vida de referencia, centrándonos en las disciplinas de Requisitos, Diseño e Implementación.
- Artefactos UML: diagramas de casos de uso, diagramas de clases, flujos de eventos y diagramas de secuencia.
- StarUML como herramienta de modelado.

En clase se utilizan numerosos ejemplos, tanto en el aula como en el laboratorio, para dominar y poner en práctica las distintas técnicas y conceptos. Pero el trabajo nuclear del estudiante es un proyecto software, que debe realizar en paralelo a las clases presenciales. Se le proporciona un enunciado y una solución parcial en el que se han aplicado las tecnologías indicadas. Su trabajo consistirá en completar el proyecto con nuevos casos de uso, un modelo de dominio más rico, etc.

Hasta aquí hemos descrito el planteamiento y los contenidos de la asignatura tal y como ha sido impartida hasta el curso 2011/12 usando metodologías docentes tradicionales, en las que el proyecto era complementado con la realización de ejercicios escritos individuales, acumulativos y eliminatorios. Ahora bien, el objeto de la presente contribución es describir cómo se ha implementado la misma asignatura en el curso 2012/13 utilizando el Aprendizaje Basado en Proyectos (ABP) y metodologías ágiles [3]. En ediciones anteriores de JENUI hemos encontrado propuestas similares que utilizan tanto metodologías ágiles [4] como ABP [5] para asignaturas de ingeniería del software, impartidas en cursos posteriores (tercero y cuarto), y por tanto, diseñadas para alumnos con más madurez.

El artículo está divido en 5 secciones. En la sección 2 se presenta el proyecto docente actual de la asignatura. En la sección 3 se presenta las líneas generales del proyecto. La evaluación y resultados se exponen en la sección 4. Finalmente, las conclusiones recogen las ideas principales de nuestra contribución.

2. El proyecto docente

Uno de los primeros requisitos a tener en cuenta en la asignatura consiste en incorporar técnicas de aprendizaje colaborativo y conseguir que el proyecto que constituye el núcleo de la asignatura se desarrolle en el marco del ABP. Así, si anteriormente se realizaba de manera individual, actualmente se ha concebido como una actividad para equipos de tres personas. Afortunadamente es bastante sencillo redimensionar la especificación de un proyecto software variando la cantidad y complejidad de los casos de uso propuestos.

Sin embargo, uno de los principales obstáculos para el éxito del trabajo cooperativo es que el alumnado raras veces cuenta con herramientas o habilidades básicas para gestionar el hábito de negociar, tomar decisiones y mantener una documentación efectiva que promueva el reparto de tareas.

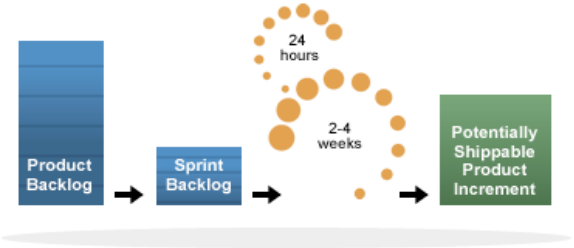

Figura 1: El proceso SCRUM.

Para salvar al menos parcialmente estos obstáculos hemos apreciado que el dotar a los estudiantes de una metodología sencilla de entender y aplicar puede ser un impulsor crucial. Es por ello que les propondremos el uso de SCRUM [6], o al menos de una versión aligerada del mismo. Esta metodología es la más popular entre todas las que compiten por el mercado del desarrollo ágil[1], por su naturaleza de proceso, en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar en grupo de manera colaborativa y obtener el mejor resultado posible de un proyecto.

La metodología está basada en roles, lo que supone una ventaja a la hora de dotar a los equipos de una manera natural de distribuir las responsabilidades y tareas. Para promover el reparto homogeneo de responsabilidades, los miembros del equipo van rotando en los roles que van adquiriendo en cada sprint.

En SCRUM un proyecto se ejecuta en bloques temporales cortos y fijos llamados *sprints*. La noción de *sprint* es absolutamente análoga a la de iteración, y el nombre únicamente recalca la importancia del tiempo y la distancia fija a una meta. Cada *sprint* tiene que proporcionar un resultado previsto y completo *en forma de incremento de producto final* que sea susceptible de ser entregado con el mínimo esfuerzo al cliente. Para mantener manejable la planificación hemos establecido *sprints* de 4 semanas.

Respecto a la gestión de la documentación, SCRUM evita la generación de mucha documentación superflua, pero la que se produce es extraordinariamente dinámica y es muy importante que esté accesible en todo momento para su consulta y actualización. Las estructuras que gestionan esta documentación que representa la auténtica instantánea del proyecto se denominan *backlogs*.

El *backlog del producto* es la lista de requisitos o casos de uso del sistema, normalmente llamados características (*features*) o historias de usuario (*user stories*). Este se incrementa en las reuniones de selección de requisitos. A su vez, en cada iteración se construye un *backlog de sprint*, que contiene las actividades a desarrollar durante esa iteración. Por tanto los backlogs de un proyecto son una herramienta excelente que permiten al equipo (y al profesor) apreciar el avance de cada proyecto, la contribución individual de los miembros del equipo y los riesgos y problemas identificados sin exigir ni mucho tiempo ni procesos complejos de gestión. Es también una herramienta de soporte para la comunicación directa del equipo y promueve la reflexión crítica del grupo.

Actualmente existen numerosas herramientas que permiten gestionar los *backlogs* de un proyecto. De entre las disponibles libremente hemos seleccionado *VersionOne* por su buen balance entre usabilidad, accesibilidad, funcionalidades y complejidad.

2.1. Ingredientes del aprendizaje colaborativo en el proceso SCRUM

Uno de los requisitos a tener el cuenta a la hora de planificar las actividades dentro del ABP, consiste en garantizar que se cumplen al menos los cinco ingredientes del aprendizaje colaborativo [7]. Consideramos que SCRUM recoge de manera natural tales ingredientes.

Uno de los aspectos metodológicos de SCRUM consiste en realizar reuniones de SCRUM diario para realizar el seguimiento del proyecto. Dado que los estudiantes no se dedican a tiempo completo al proyecto, en nuestro contexto “diario” significa “dos veces a la semana”, concretamente al comienzo de las clases de lunes y jueves. De esta forma fomentamos y garantizamos la *interacción cara a cara***,** uno de los ingredientes básicos mencionados. Adicionalmente solventamos el problema de encontrar tiempo compatible, podemos asistir parcialmente a esas reuniones y podemos leer el informe o resultado de las mismas. Los *backlogs* son excelentes mecanismos de *exigibilidad individual*: es poco probable que un equipo tolere que un miembro se atribuya la realización de tareas que no están completadas y por las que todo el equipo va a ser responsabilizado.

El mantenimiento del *backlog de sprint* es útil porque obliga a las personas del equipo a una *comunicación directa* para determinar qué actividades hay que desarrollar y cómo abordarlas de manera colaborativa (repartiéndose el trabajo). También somos conscientes de que durante el trabajo colaborativo inevitablemente surgirán desavenencias e incidentes, que les deberá llevar a una *reflexión crítica del grupo*, donde deberán tomar decisiones que permitirán mejorar la dinámica de equipo. Esta se verá reforzada por la reunión de retrospectiva, donde deben identificar qué es lo que funciona bien, qué es lo que no, y qué medidas podrían tomar para mejorar.

En definitiva, SCRUM por su propia naturaleza, se basa en el *compromiso conjunto* y la colaboración entre las personas del equipo. La *transparencia* entre todos es fundamental para poder entender la situación real del proyecto y así poder hacer las mejores adaptaciones que permitan conseguir el *objetivo común*. Por ello, su implantación como modelo de trabajo en el desarrollo del proyecto creemos que fomentará el ejercicio de *habilidades interpersonales y de trabajo en grupo***.**

2.2. La influencia del desarrollo iterativo en el aprendizaje

Para desarrollar el proyecto hemos intentado

[1] http://www.proyectosagiles.org/

integrar cuatro de las tendencias más importantes que han caracterizado la evolución de la Ingeniería del Software:

1. La necesidad de utilizar una metodología sólida integrada en la cultura de la organización (en nuestro caso el Proceso Unificado de Desarrollo del Software UP).
2. El uso de notaciones estandarizadas orientadas a producir software manejable, mantenible y reutilizable (en nuestro caso el lenguaje visual de modelado UML).
3. El desarrollo del software en iteraciones, por oposición a los ciclos de vida que pretenden agotar al máximo cada fase antes de proceder a la siguiente.
4. La adopción de metodologías ágiles que eviten la sobrecarga de artefactos, fomenten la discusión entre grupos de interés y produzcan versiones ejecutables y utilizables de manera continua.

La utilización de SCRUM, ya discutida en secciones anteriores, tiene una cualidad muy interesante para el proceso de aprendizaje: dado que el final de un *sprint* se ha de producir un producto software (ejecutable y funcional) que mejore el desarrollado en la iteración anterior, ello implica que en todos los *sprints* se han de trabajar todas las disciplinas (Requisitos, Diseño e Implementación). Esto permite una gran flexibilidad a la hora de distribuir los objetivos de aprendizaje, ya que cada *sprint* puede ser visto como una *pasada* sobre la metodología, de forma que en cada iteración el producto se irá enriqueciendo con funcionalidades, aspectos, casos de uso, interfaces y modelos de dominio crecientemente complejos, que estimularán al estudiante a dominar de manera progresiva las diferentes técnicas y herramientas de la Ingeniería del Software. Creemos que, además, esta estrategia permite aplicar los principios pedagógicos propuestos en [8], especialmente en el traspaso del control y la responsabilidad y control del aprendizaje a los alumnos.

Los ciclos de vida iterativos favorecen la trazabilidad y el control de cambios de los proyectos software. Es el tiempo la variable que gobierna el desarrollo, y no el producto “ideal”. Por otro lado cobra importancia el concepto de “el mejor producto que podemos producir en tiempo X”, idea muy interesante desde el punto de vista didáctico, ya que permite:

1. manejar de la misma manera los objetivos de aprendizaje, en el sentido de poder definir “el mejor dominio que podemos adquirir sobre el proceso de software en tiempo X”
2. flexibilizar dichos objetivos en función de las especificidades de los equipos de estudiantes; grupos con diferentes preparaciones, diferentes capacidades y, sobre todo, diferentes expectativas académicas, puedan sacar el mejor provecho de sí mismos sin verse negativamente afectados por las características de los restantes grupos.

Por ello hemos configurado nuestro proyecto docente inspirados en la propia materia que impartimos. Consideramos que si vemos los objetivos de aprendizaje de esta asignatura como un producto a desarrollar, podemos aplicar estas metodologías al diseño y planificación de la asignatura. Nos gustaría resaltar el aspecto novedoso de este enfoque, en el que los temas no se desarrollan secuencialmente ni dotan de estructura al curso, sino que se desarrollan de manera iterativa, revisando y profundizando cada concepto y técnica en varias ocasiones.

3. El Proyecto de la asignatura

El Proyecto para la asignatura se planteará en forma de enunciado incremental que se enriquece a medida que el alumnado va superando las iteraciones. Asimismo es muy importante la idea de suministrarles junto al enunciado una solución parcial, es decir un punto de partida que, por un lado, les limita y obliga a hacer un trabajo de análisis y comprensión, y por otro les da pistas acerca de cómo pueden abordar los problemas.

3.1. El escenario de partida

Antes incluso de la formación de grupos el planteamiento será el siguiente. Cada estudiante acaba de llegar a la importante empresa *Sinking Soft*. Inicialmente el desconcierto es total: el recién llegado no sabe qué se espera de él, y el equipo técnico que le recibe tampoco sabe hasta qué punto puede contar con él ni tiene tiempo o ganas de averiguarlo. El profesor tiene aquí que desempeñar un tercer rol: además de su función ordinaria y de su figuración como cliente, debe en ocasiones hacer las veces de responsable de la empresa de acogida.

Una de las primeras cosas que le sucederán es que le caerán los “marrones”: trabajos defectuosos o inconclusos que hay que pulir para que cumplan todas las especificaciones. Estos marrones serán de tipo individual y tienen una doble función. Primero introducen a los alumnos en las tecnologías necesarias para desarrollar el proyecto (AWT y Db4o) y por otro lado, nos servirán para detectar de manera natural quienes están interesados en realizar el proyecto.

Tras la formación de grupos (en principio libre) tendrán un encargo de mayor responsabilidad: el Proyecto, que también será un trabajo dejado a medias por otros compañeros (la solución parcial). El dominio del proyecto será el mismo para todos los grupos y consistirá en una aplicación de software que

permita el alquiler remoto de casas rurales y la gestión de las mismas por sus propietarios. El alcalde del pueblo en el que están situadas las casas es el cliente. El proyecto se desarrolla en 3 iteraciones.

3.2. Las iteraciones del proyecto

La figura 2 muestra el temario de la asignatura, así como las actividades (A), laboratorios (L) y entregables del proyecto:

- P0: Análisis de requisitos globales del proyecto.
- P1: Análisis de requisitos de la iteración en curso.
- P2: Diseño de la iteración en curso.
- P3: Implementación y prueba de la iteración en curso.

En la primera iteración se introduce el Proyecto con una iteración completa ya desarrollada (iteración P0) para un caso de uso sencillo (registrarse). El objetivo de la primera iteración consiste en que los alumnos tengan una visión general de todo el proceso de desarrollo software, desde la captura de requisitos hasta la implementación. En esta fase, para algunos casos de uso muy básicos (p.ej hacer login), los grupos definen el modelo del dominio y el flujo de eventos en la fase de requisitos, los diagramas de secuencia y el diseño de clases en las fase de diseño, y finalmente implementan una solución, donde tienen que acceder únicamente a la información ya existente en la BD (no deben modificar el nivel de persistencia). Al final de esta iteración, es fundamental que entiendan cual es el reflejo que tiene cada artefacto en las fases de desarrollo posteriores, de manera que en *sprints* posteriores sean conscientes de su repercusión. Este *sprint* tiene una duración de 4 semanas, aunque se prevé que los incumplimientos puedan ser más frecuentes y se tolerarán siempre que no indiquen una clara dejación del equipo. Durante esta fase se plantean 4 laboratorios dirigidos (L1-L4), dos para el aprendizaje de tecnología (AWT/SWING y Db4o) y otros dos para modelado (StarUML), así como varias actividades para introducir los diferentes modelos de análisis y diseño (A1-A5).

La segunda iteración se desarrolla durante las siguientes 4 semanas, y el objetivo consiste en profundizar en cada una de las fases de desarrollo. Para ello, se propone a los grupos de trabajo el desarrollo de casos de uso más complejos. Así, de manera espontánea surgen en la fase de requisitos, dependencias entre los casos de uso (*includes* y *extends*) y relaciones más complejas en el modelo del dominio (p.ej. clases asociación). En las fases de diseño se hace hincapié en las arquitecturas 3 niveles y se introduce en la actividad A6, alguno de los patrones de diseño GRASP [9] fundamentales (Controlador, Experto y Creador).

En la implementación, como novedad tienen que manejar la persistencia de los objetos. El laboratorio L6 les ayuda en esta tarea. En esta fase, los alumnos deben ser conscientes de la trazabilidad de cada artefacto durante todas las fases de desarrollo.

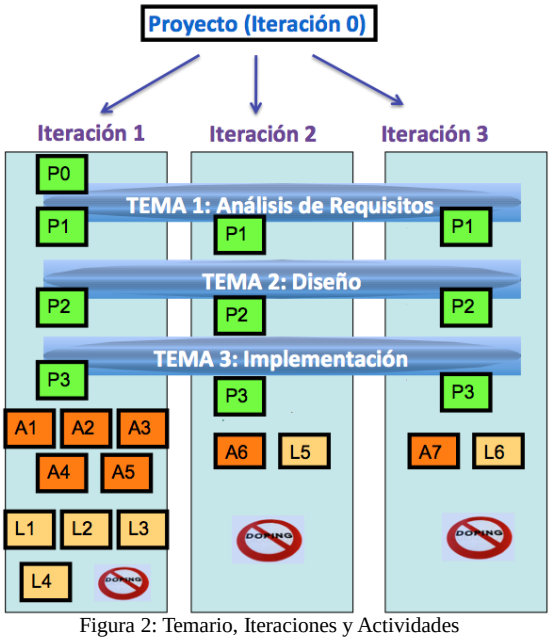

Figura 2: Temario, Iteraciones y Actividades

La tercera y última iteración abarca 4 semanas y es cuando los grupos toman protagonismo y deciden qué casos de uso van a desarrollar (con la aprobación del Dueño del Proyecto[2]). Los grupos trabajan con autonomía y el docente únicamente interviene ante las dudas planteadas o para supervisar el desarrollo del proyecto. Como respuesta a las propuestas de los alumnos surge la necesidad de introducir los patrones GRASP restantes (Bajo Acoplamiento y Alta Cohesión) (Actividad A7) y en la implementación se propone que realicen un despliegue de la aplicación en 3 niveles físicos utilizando RMI (Remote Method Invocation) siguiendo las guías del laboratorio L6.

Cabe destacar una actividad adicional denominada "Anti-doping", y cuya finalidad está enfocada a garantizar la exigibilidad individual del alumno. La valoración de la actividad se calificará como Apto o No apto, y si el alumno no supera esta actividad no podrá continuar en el proyecto. Esta actividad consiste en una prueba individual escrita donde el estudiante debe responder a unas preguntas básicas directamente relacionadas con el desarrollo de la iteración. La evaluación de las demás actividades se describe en la siguiente sección.

[2] En nuestra experiencia, los dueños del proyecto somos los profesores de los grupos. Se trata de definir los requisitos iniciales y aprobar los requisitos finales del proyecto,

3.3. La evaluación

Todas las actividades a realizar serán obligatorias, y por tanto conllevarán algún tipo de evaluación. Sin embargo, para evitar una excesiva rigidez y un número demasiado alto de notas a integrar hemos considerado tres modalidades principales y una variante adicional:

- Filtro: La valoración de la actividad se calificará como Apto o No apto , pero no tendrá capacidad aditiva sobre la calificación final. Se tolerará un máximo de tres actividades con No Apto.
- Punto Fijo: La actividad tendrá un peso concreto en la calificación final.
- Extra Bonus: Independientemente de que la actividad se evalúe según alguno de los esquemas anteriores, se premiarán las actividades que de alguna forma sean sobresalientes con puntos adicionales sobre la calificación final.

La evaluación del proyecto tiene un porcentaje en la nota final del 75%. El peso de cada una de las iteraciones va incrementándose a medida que aumenta su complejidad (15%, 25% y 35%).

El restante 25% recoge las actividades complementarias, no centradas en el modelo ABP, aunque sí estrechamente relacionadas con él. Se han agrupado las evaluaciones de las pruebas anti-doping (15%) con los laboratorios dirigidos (10%) y las actividades de tipo filtro, sin una puntuación fija.

Dado que al final de cada iteración cada grupo debe tener una aplicación operativa, aunque quizás incompleta, debería ser posible hacer una presentación de control ante el cliente. Sin embargo esto puede suponer un indeseado cuello de botella si los docentes tienen que revisar en tiempo real todos los trabajos. La solución será introducir una sesión de contraste entre pares, a realizar en un laboratorio al final de las iteraciones 1 y 2. Cada grupo tendrá como tarea asistir a la defensa del Proyecto de otro grupo y detectar posibles errores, inconsistencias o fallos de funcionamiento. Como conclusión debe escribir un *informe de inspección* que se entregará al profesor o profesora que se incorporará al *backlog* del grupo auditado, junto con una versión "congelada" del proyecto. Este contraste será una tarea de tipo filtro, pero podrá tener consecuencias posteriores. Si un fallo grave es detectado e incluido en el Informe pero no es corregido por el grupo inspeccionado, este sufrirá una penalización adicional que en casos extremos puede implicar la anulación del proyecto. Pero si un fallo grave no es detectado y se propaga a iteraciones posteriores la responsabilidad será compartida entre el grupo inspeccionado y la persona inspectora. La inspección final será la defensa del proyecto y se efectuará por el profesorado al terminar la tercera iteración. Esta tendrá forma de exposición pública ante la clase. Finalmente, en las últimas semanas del cuatrimestre se realizará un control para confirmar el nivel de exigibilidad individual adecuado en el desarrollo del proyecto (lo denominamos *control anti-doping*).

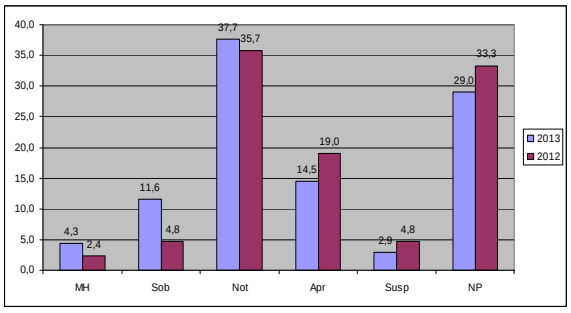

Tabla 1: Calificaciones finales (convocatoria ordinaria).

4. Resultado y análisis

4.1. Valoración de resultados académicos

Resulta complicado mostrar los resultados obtenidos en cada actividad de evaluación teniendo en cuenta que el planteamiento inicial recoge exactamente doce (12) entregables que han sido evaluados. Cuatro de ellos corresponden al proyecto (75%) de la nota y ocho a actividades complementarias (25%) de la nota final. Muchas de las actividades han sido de tipo filtro, con dos notas posibles (apto y no apto), respecto a estas actividades se puede decir que fueron superadas sistemáticamente por todos los alumnos excepto cinco[3] que fueron expulsados de la evaluación continua (tres) o bien abandonaron la misma (dos). Estos estudiantes pasan a evaluación de conjunto, que consiste en realizar un proyecto en un dominio diferente y un examen final.

Desde que se definió esta asignatura en el marco del Grado en Ingeniería Informática, tenía un claro objetivo de aprendizaje: el desarrollo por parte del alumno de un prototipo de sistema informático, en un dominio real, que comprendiera las tareas de análisis, diseño e implementación. En este sentido podemos afirmar que tanto la calidad y complejidad de los sistemas desarrollados este curso, por los grupos de trabajo, superan en más del doble los resultados obtenidos el curso anterior (2011/2012). Y esto no es algo puntual, sino que podríamos afirmar que el peor de los proyectos de este curso es bastante mejor que el mejor de los del curso pasado.

También hay que afirmar que el tiempo dedicado a esta actividad este curso ha sido superior al tiempo

[3] El número inical de alumnos que comenzo PBL era de 50, repartidos en tres grupos de 23 (castellano), 21(euskera) y 6 (Ingés)

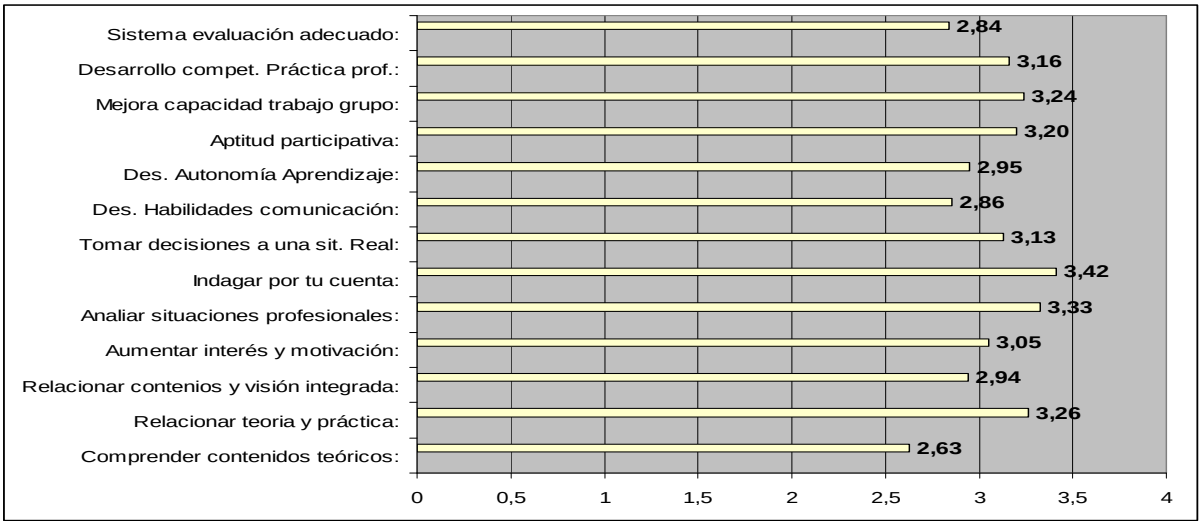

Gráfico 1: Preguntas cuestionario ABP

dedicado el curso pasado. Si el curso pasado se realizó una planificación orientada a 120 horas/curso, en este al final la planificación se ha desviado a 150 horas/curso con una media semanal de 5,5 horas de trabajo no presencial y 4,5 de trabajo presencial. Los resultados obtenidos en las calificaciones finales (convocatoria ordinaria) se muestran en la Tabla 1. En ella se pueden ver las calificaciones obtenidas por los alumnos en los dos últimos cursos académicos, tanto absolutas, como relativas al número de alumnosde cada curso lectivo. Se han acumulado los alumnos de los tres grupos existentes. La tabla muestra los porcentajes de cada curso académico. A la vista del mismo cabe destacar lo siguiente:

- Hay una disminución global del número de no presentados de un 4,3% con la implantación del ABP. Este dato es más acusado en el grupo de castellano, el más numeroso, donde se ha pasado de un 50% de no presentados el curso 2012 a un 35% este curso (15 % menos de no presentados).
- Las calificaciones en general han aumentado, hay el doble de Matriculas y de Sobresalientes. Levemente han aumentado los Notables y han disminuido los Aprobados. Esto sin duda se debe a que los proyectos desarrollados han sido mejores que la media prevista.

4.2. Valoración de las encuestas ABP[4]

Respecto a la valoración por parte de los estudiantes sobre su aprendizaje, un 45% opina que esta metodología les ayuda más que el modelo tradicional de clases magistrales y un 23% opina incluso que mucho más. En el lado opuesto, con un 16% en cada caso, consideran que les ha ayudado igual o menos respectivamente.

Cabe destacar que ocho de los trece aspectos considerados (Gráfico 1) han tenido una valoración media superior a tres (Bastante, en la escala utilizada). Han valorado positivamente el esfuerzo de trabajo en grupo, el hecho de haber analizado situaciones similares a las que se encontrarán en su desempeño profesional y el de haber adquirido competencias prácticas cercanas al mundo de la empresa. El resto de criterios han superado el valor 2,5 con lo cual su valoración es altamente positiva en todos los aspectos.

Respecto a la pregunta de si optarían por esta metodología para el próximo curso, el gráfico 2 muestra los resultados: Un 82% repetiría la experiencia.

5. Conclusiones

En este apartado comentaremos los aspectos positivos tanto desde el punto de vista del aprendizaje como desde el punto de vista de la gestión. Se describirán los aspectos a mejorar y nuestro punto de vista para la continuación de esta metodología el próximo curso (2013/14).

5.1. Aspectos positivos

Sin duda alguna es de destacar el mejor rendimiento académico obtenido tanto en la mejora de las calificaciones, los proyectos obtenidos y la disminución del porcentaje de no presentados, tal y como se han expuesto en el capítulo anterior. Respecto a las competencias logradas, las tres nucleares eran la capacidad de análisis, diseño e

[4] Esta encuesta la proporciona el Servicio de Asesoramiento Educativo de la UPV/EHU.

implementación, hay unanimidad entre los cuatro profesores de que los resultados han sido mucho mejores que los cursos precedentes. Otro aspecto que nos parece interesante es el hecho que sólo 5 estudiantes de 50 abandonaron la evaluación ABP (dos por decisión suya y tres por decisión de los profesores). De los 5, 3 optaron por la evaluación global con resultados satisfactorios.

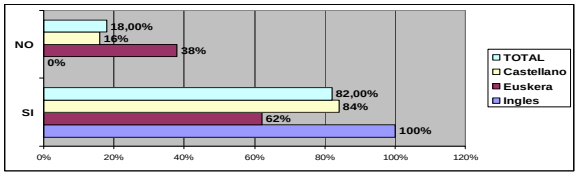

Gráfico 2: Satisfacción global. ¿Repetirías la experiencia?

Esto último indica que la información que tienen los profesores de estos alumnos que pierden el tren es mucho mayor y por tanto es mucho más fácil orientarles hacia la consecución de las competencias. En nuestra implementación había dos aspectos novedosos, la realización del proyecto en tres iteraciones y la "imposición" de SCRUM como metodología de trabajo en equipo. Ambos aspectos sin duda creemos que han sido beneficiosos para la asignatura. Nos han ayudado a replanificar contenidos, a ver los problemas de los estudiantes al construir un proyecto, a regular y flexibilizar las exigencias grupales y particulares y sobre todo a conseguir que el funcionamiento de los grupos se base en una metodología concreta. Los resultados de las encuestas corroboran estas opiniones.

5.2. Aspectos a mejorar

Quizás llevados por un exceso de celo en la aplicación del ABP hemos intentado minimizar las clases de conceptos teórico/prácticos y esto parece que los estudiantes no lo han asimilado del todo bien. Once de los alumnos han indicado en la encuesta que se deberían impartir más clases teóricas y cinco de ellos la realización de más clases prácticas. Creemos que esto ha sido motivado por las peticiones de "marrones" que les hemos hecho analizar y solucionar a partir de sesiones planteadas en los laboratorios. Pretendemos que descubran las soluciones a los problemas y esto en algunos casos les ha supuesto bastante esfuerzo. La solución es revisar estos trabajos, intentar guiarles un poco más hacia la solución y dividirlos en problemas más pequeños.

Otro aspecto a mejorar sería hacerles ver desde el principio que deben aprender a valorar los riesgos (*Risk Orientation*). Algunos grupos han propuesto funcionalidades que suponen una carga de trabajo en implementación muy fuerte (gestión de imágenes, integración del sistema con redes sociales, gestión de alquileres diarios y por lotes, etc..). Debemos ser capaces de enseñarles a identificar estos riesgos y a valorar si realmente con su disponibilidad a la asignatura como estudiantes van a ser capaces de asumirlos.

5.3. Implicaciones para el futuro

En vista de los resultados académicos, del seguimiento realizado y de la opinión de los alumnos estamos motivados para continuar el curso próximo con esta metodología. Revisando las tareas previas ("marrones"), trabajando a nivel de grupos en la valoración de riesgos y aplicando la planificación prevista en la guía del docente, consideramos que podemos resolver la mayoría de los problemas que nos hemos encontrado este curso. Si bien es de recibo decir que no hemos detectado ningún problema grave que haga temblar los cimientos de este proyecto académico.

Referencias

[1] Savery, John R. (2006) "Overview of Problem-based Learning: Definitions and Distinctions". Interdisciplinary Journal of Problem-based Learning: Vol. 1: Iss. 1, Article 3.

[2] Jacobson, I.; Booch, G.; Rumbaugh, J. The Unified Software Development Process (Addison-Wesley Object Technology Series). Addison-Wesley Professional: 1999.

[3] Goñi A., Ibáñez J., Iturrioz, J. y Vadillo J.A.: ABP aplicado a la asignatura Ingeniería del Software: Guías del docente y del estudiante. Informe Interno UPV/EHU/LSI/TR 03-2012.

[4] Letelier P. y Penadés M.C. Una estrategia para la enseñanza de metodologías ágiles. JENUI 2013 pp. 217-224.

[5] Sánchez P. y Blanco C. Implantación de una metodología de aprendizaje basada en proyectos para una asignatura de Ingeniería del Software. JENUI 2012 pp.41-48

[6] Rubin, Kenneth S. "Essential SCRUM: A Practical Guide to the Most Popular Agile Process". Addison-Wesley Signature Series (Cohn). 2012

[7] Bará J., Dominguez J. y Valero M. Técnicas de Aprendizaje Cooperativo Basado en Proyectos. Taller de Formación. Univ. Polit. de Cataluña. 2011

[8] Miró J. El diseño de una asignatura a partir de principios pedagógicos. JENUI 2013 pp. 111-118.

[9] Larman C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3ª edición) Pearson Education, 2004.