Una Experiencia de aprendizaje basado en proyectos utilizando herramientas colaborativas de desarrollo de software libre

Jose Emilio Labra Gayo, Daniel Fernández Lanvin,
Jesús Calvo Salvador, Agustín Cernuda del Río
Dpto. de Informática
Universidad de Oviedo
C/Calvo Sotelo S/N CP 33007 Oviedo, Spain
{labra, dflanvin,calvojesus,guti}@uniovi.es

Resumen

En este artículo se describe la experiencia desarrollada en el ámbito de una asignatura optativa sobre programación declarativa en la que se han utilizado herramientas colaborativas habituales en el desarrollo de software libre. Se ha creado un proyecto común entre todos los estudiantes, con el objetivo de facilitar un aprendizaje basado en proyectos.

1. Introducción

La implantación del Espacio Europeo de Educación superior ha traído consigo una creciente preocupación por un nuevo modelo de enseñanza/aprendizaje, centrado en el alumno. Aparecen una serie de palabras como competencias, destrezas y habilidades que adquieren una importancia clave en el nuevo planteamiento de la Enseñanza Superior.

En el proyecto Tuning [12] se habla de la importancia de la adquisición de competencias genéricas o transversales como la capacidad de resolución de problemas, trabajo en equipo, habilidad de comunicación oral y escrita, adaptación a situaciones nuevas, planificación y organización, etc. Dichas competencias son difíciles de enseñar y habitualmente son adquiridas, si es que llegan a serlo, de forma accidental por los estudiantes.

En ese contexto, se están desarrollando diversas iniciativas de innovación educativa que pretenden modificar las metodologías tradicionales utilizando nuevas técnicas entre las que se encuentra el aprendizaje basado en proyectos [7]. Como se indica en la siguiente sección, este tipo de aprendizaje puede favorecer la adquisición de este tipo de competencias genéricas.

Dentro de la Escuela Universitaria de Ingeniería Técnica Informática de la Universidad de Oviedo se imparte una asignatura optativa denominada *Programación Declarativa* dedicada a la enseñanza de los paradigmas de programación lógica y funcional. La asignatura es cuatrimestral de primer cuatrimestre (6 créditos) y el número de alumnos matriculados habitualmente oscila entre 20 y 40. En cursos anteriores se dividía la asignatura en 2 partes, en la primera parte se impartía el lenguaje de programación funcional *Haskell* y en la segunda parte el lenguaje de programación lógica *Prolog*. Impartir dos lenguajes de programación de dos paradigmas de programación diferentes resultaba complicado y se optaba por una enseñanza liviana de los mismos, con el principal objetivo de que los estudiantes reconociesen las posibilidades de los lenguajes, pero sin pretender que llegasen a ser desarrolladores expertos en los mismos.

A partir del curso 2002-03 se planteó una primera innovación en la asignatura mediante un esquema de prácticas basado en utilizar XML y vocabularios específicos, llegando a crear mundos virtuales infinitos mediante *octrees* [9,10]. Dicho esquema de prácticas de laboratorio estaba completamente organizado atendiendo a la secuenciación de contenidos teóricos vistos en clase y considerando el atractivo que dichos ejercicios podrían tener para los estudiantes.

Los boletines de ejercicios contenían un número determinado de problemas cortos obligatorios (habitualmente 10), seguidos de un número reducido de ejercicios abiertos opcionales (entre 3 y 5) para que los estudiantes que lo deseasen pudiesen incrementar sus notas.

Dicho esquema era bastante bien aceptado por los estudiantes, que conocían qué tenían que hacer para aprobar la asignatura y qué tenían que hacer para obtener mejores puntuaciones. Aunque se detectaron algunos abandonos y algunos casos de plagio, el porcentaje de aprobados era cercano al 90%.

Sin embargo, quedaban varias dudas en el aire: ¿estaban aprendiendo algo útil los estudiantes? ¿hasta dónde eran soluciones originales y no prácticas plagiadas? Y ¿qué pasaría cuando saliesen al mercado laboral y se enfrentasen a problemas reales? ¿Cuáles de las competencias genéricas estaban siendo practicadas en la asignatura?

Ante dichas preguntas, en el curso 2005-06 se decidió modificar completamente dicho esquema de prácticas y optar por un modelo abierto. El objetivo ha sido huir de un enfoque cerrado y *lanzar a los estudiantes a la piscina* proponiéndoles la realización de un proyecto conjunto con unos objetivos iniciales poco claros y que sería desarrollado por todos de forma colaborativa siguiendo la inspiración de las técnicas que se utilizan para el desarrollo del software libre. En este artículo se describe la experiencia y se indican algunas conclusiones y sugerencias de cómo mejorar este esquema.

2. Aprendizaje basado en proyectos

El papel del profesor tradicional que impartía una materia mediante clases magistrales y posteriormente, al finalizar el curso, realizaba un examen de dicha materia ha sido cuestionado en gran cantidad de ocasiones y se buscan nuevos modelos.

A finales de los años 60, N. Postman y C. Weingartner [11] proponían un modelo de enseñanza en el que se prescindiese de las clases magistrales y se desarrollase la capacidad creativa de los estudiantes mediante el planteamiento de preguntas y problemas abiertos.

El desarrollo del aprendizaje basado en proyectos comenzó a aplicarse en el ámbito universitario en el campo de la medicina [2]. Posteriormente, se aplicó en ingenierías y específicamente en ingenierías informáticas [1, 4,5].

En este esquema, los profesores proponen uno o varios proyectos, habitualmente inspirados en problemas reales, que los estudiantes deben resolver en grupo.

Los estudiantes deben decidir cómo afrontar los proyectos y qué actividades llevar a cabo. Deben juntar información de diversas fuentes, analizándola y resumiéndola, para generar nuevo conocimiento a partir de la misma. Este tipo de aprendizaje puede tener un gran valor al fomentar el desarrollo de habilidades genéricas como la coordinación, trabajo en equipo, búsqueda de información, planificación y organización, etc.

3. Herramientas colaborativas de desarrollo de software

La creación de software es una actividad compleja que requiere de la colaboración de grandes equipos de personas. Aunque, en numerosas (quizá excesivas) ocasiones, se enseña a los estudiantes a crear pequeños programas de forma individual, en la realidad, la mayoría de los productos software son desarrollados por mucha gente que debe organizarse. No es de extrañar que una de las habilidades que valoran las empresas de un desarrollador software sea su capacidad de trabajo en equipo.

Por otro lado, el movimiento de software libre ha mostrado la posibilidad de que se desarrollen grandes productos de forma colaborativa entre grupos de personas poco organizados, con intereses muy diferentes y geográficamente distantes. Para ello, se han desarrollado herramientas que facilitan este esquema de trabajo.

Una de las herramientas más populares es *Sourceforge*[1], software que proporciona numerosas funciones para gestión de proyectos como sistema de control de versiones (cvs), listas de correo, gestión de tareas y de errores, etc. En el momento de escribir este artículo, su página web anuncia 112.764 proyectos y 1.245.763 usuarios registrados. Muchos de los proyectos más populares del software libre, son desarrollados utilizando dicha herramienta.

La utilización de sistemas de control de versiones es clave en el desarrollo de cualquier producto en el que interviene más de una persona. El sistema más popular es cvs[2], proporcionado por *Sourceforge*, aunque existen otros sistemas que están incrementando su popularidad como Subversion[3] o DARCS[4].

[1] http://www.sourceforge.net
[2] http://www.cvshome.org/
[3] http://subversion.tigris.org/
[4] http://abridgegame.org/darcs/

Otra característica habitual en el desarrollo de software por numerosos equipos es la técnica de desarrollo dirigido por pruebas [3] que promueve la creación de pruebas unitarias anteriores a la implementación del programa. *JUnit* es uno de los sistemas más populares en *Java*, aunque la popularidad de la idea ha facilitado el desarrollo de sistemas similares para otros lenguajes, como por ejemplo, *HUnit*[5] para el lenguaje *Haskell*.

4. Proyecto único y abierto

Una vez decidida la utilización de aprendizaje basado en proyectos mediante herramientas colaborativas, era necesario establecer qué tipo de proyectos resolver y cuántos.

La primera decisión que hubo que tomar fue la cantidad de proyectos. Para simplificar el proceso, y debido a la escasez de tiempo en una asignatura cuatrimestral, se decidió que los estudiantes se enfrentasen a un único proyecto. Con el fin de aumentar la motivación se decidió que fuese un problema real y suficientemente actual para que los estudiantes pudiesen llegar a realizar aportaciones novedosas en un determinado campo.

También surge la posibilidad de dividir a los estudiantes en varios grupos encargándoles a cada uno un proyecto distinto (o el mismo, y que compitan entre sí). Sin embargo, dado que el número de estudiantes matriculados era relativamente reducido (22) se optó por realizar un único proyecto por un único equipo formado por los 22 estudiantes más los 2 profesores de la asignatura. Una ventaja de este modelo era que se evitaría directamente la posibilidad de plagio. No habría nadie a quien plagiar.

La siguiente decisión importante era acerca del tema sobre el cual realizar el proyecto. En el caso de la asignatura de Programación Declarativa, se utilizaban dos lenguajes de programación diferentes: Haskell y Prolog, lo que complicaba el aprendizaje en detalle de los mismos en un cuatrimestre. Para evitar dicho problema, se optó por utilizar un único lenguaje de programación: Haskell, que cuenta con implementaciones estables en entornos Windows y Linux y con una base de programadores relativamente numerosa, además de los correspondientes canales de soporte como listas de correo y salas *IRC* dedicados al lenguaje.

Puesto que se iba a utilizar un lenguaje funcional para la implementación del proyecto, se consideró que el proyecto podría consistir en implementar algo relacionado con la parte de programación lógica y que tuviese un cierto atractivo para los estudiantes. El tema escogido, encajaba además con los intereses investigadores de los profesores, lo cual suponía un incentivo más, tanto para los profesores, porque les interesaba aprender del desarrollo realizado por los alumnos, como para los alumnos, porque podían identificar áreas en las que no siempre existía una respuesta. En concreto, se decidió desarrollar un proyecto que implementase utilidades para la web semántica en *Haskell*. La web semántica es un campo suficientemente amplio y abierto para que los estudiantes pudiesen encontrar múltiples posibilidades de desarrollo. Además, en la actualidad, la mayoría de las herramientas están en una fase experimental y de investigación. En concreto, que se tenga conocimiento, el único sistema similar desarrollado en *Haskell* por G. Klyne, es *Swish* [8]. Dicho sistema tiene un propósito más definido y todavía no dispone de una versión oficial.

Los objetivos del proyecto eran deliberadamente ambiguos y ni los alumnos, ni los profesores sabían en un principio hasta qué nivel de detalle iban a llegar en el desarrollo. Esa ambigüedad formaba parte de los objetivos iniciales aunque, como se verá en la evaluación, no fue precisamente bien valorada por los alumnos.

5. Metodología de desarrollo

El desarrollo del proyecto comenzaba el primer día de clase con una ligera explicación de los objetivos y del tema sobre el que se iba a trabajar. La organización docente de la asignatura incluía dos sesiones de dos horas cada una a la semana, una de ellas teórica y otra de ellas en el laboratorio.

5.1. Evaluación de la asignatura

La evaluación de la asignatura se lleva a cabo a partir de los informes de trabajo confeccionados por cada alumno que serían contrastados con el

[5] http://hunit.sourceforge.net/

resto de información que el profesor podía recolectar de forma independiente: participación en clase y en la lista de correo, contribuciones al desarrollo, actas de reuniones y otras aportaciones.

Se solicitaron 2 informes personales de participación en el proyecto, en los que los estudiantes tenían que resumir cuáles habían sido sus contribuciones al mismo.

5.2. Herramientas utilizadas

Como se ha indicado, se optó por utilizar herramientas de desarrollo colaborativas. En concreto, se creó un proyecto dentro del gestor de proyectos *Sourceforge* denominado WESO[6]. Dicho proyecto es visible públicamente y requiere que el software tenga una licencia de código abierto.

Las principales herramientas que se han utilizado de *Sourceforge* son:

- Sistema de control de versiones (CVS). Este tipo de sistemas son fundamentales cuando se desea desarrollar programas por varias personas. La mayoría de los estudiantes no habían trabajado nunca con este sistema, y de hecho, hubo que dedicar una pequeña cantidad de tiempo a explicar su funcionamiento básico.
- Gestión de usuarios. Se asignó a todos los estudiantes todos los privilegios (salvo el de administrador) para que pudiesen realizar todas las tareas que creyesen convenientes.
- Listas de correo. Se indicó a los estudiantes que la aclaración de cualquier duda debería ser realizada a través de las listas de correo alojadas en dicho proyecto. Dichas listas disponen de un sistema de archivo y se configuraron para que fuesen públicas, lo que permitía a cualquier persona consultar el histórico de mensajes anteriores.
- Archivo de documentación. El sistema permite incluir documentación de los proyectos, aunque la gestión a través de la página Web puede ser un poco tediosa.
- Publicación de versiones (*releases*). El sistema permite publicar versiones, gestionando las descargas de las mismas incluyendo también estadísticas de número de descargas.
- Espacio Web: El sistema facilita el alojamiento de espacio Web. De hecho, la página Web del proyecto ha sido desarrollada también por los propios estudiantes.
- Seguimiento de *bugs* y solicitud de características. El sistema incluye módulos que permiten realizar un control y asignación de *bugs* y de características a desarrollar.
- Control de subproyectos y tareas. Se admite la posibilidad de definir subproyectos y tareas dentro de los mismos, las cuales pueden ser asignadas a diferentes grupos.

Además de las herramientas anteriores, disponibles todas ellas en *Sourceforge*, se utilizaron las siguientes herramientas:

- Canal IRC. El primer día de clase se comentó la posibilidad, y uno de los estudiantes creó un canal IRC en `irc.freenode.net` denominado `#weso`, en el que estudiantes, profesores y cualquier otra persona interesada en el proyecto podrían participar.

5.3. Grupos de trabajo

Como se ha indicado, el proyecto se llevaría a cabo de forma colaborativa mediante un único equipo de desarrollo formado por estudiantes y profesores.

Siguiendo la práctica habitual del trabajo en equipo, se decidió que en todas las reuniones, incluidas las clases prácticas de laboratorio, una persona se encargaría de tomar nota y realizar un acta de las principales decisiones tomadas. Dicha persona debería añadir el acta al archivo del proyecto en *Sourceforge*.

En las diferentes fases de desarrollo del proyecto, se decidió ir creando pequeños subgrupos que se responsabilizasen de ciertas tareas, pero siembre formando un equipo dentro del proyecto. La creación de dichos subgrupos fue realizada por el profesor intentando mezclar a los estudiantes. Una de las características de una asignatura optativa es que los estudiantes matriculados están en situaciones muy diversas. De hecho, había desde estudiantes que empezaban segundo curso por primera vez hasta estudiantes que estaban a punto de finalizar su proyecto fin de carrera. Por ese motivo se consideró que sería aconsejable mezclar a los estudiantes intentando compensar los grupos.

[6] http://weso.sourceforge.net

El trabajo dentro de dichos subgrupos sería gestionado por ellos mismos, con la pretensión de que aprendiesen a organizarse y planificarse y se solicitó que realizasen un acta de todas las reuniones, fuesen presenciales o no. Dichas actas deberían ser almacenadas en el archivo del proyecto.

Otra decisión tomada acerca de los grupos, fue la de que tuviesen una duración corta. Es decir, se les encargaba una determinada tarea durante un tiempo de unas 2 ó 3 semanas. Una vez finalizada la tarea se procedía a una nueva reorganización de los grupos con asignación de una nueva tarea. El objetivo era que los estudiantes se mezclasen aún más entre sí y que no se especializasen demasiado en una determinada parte del proyecto. Como se verá, esta decisión ha sido de las peor valoradas por los estudiantes.

6. Desarrollo del Proyecto

6.1. Identificación del tema

La primera labor que se encargaría a los estudiantes sería seleccionar dentro del campo de la web semántica un tema de interés especial. El objetivo de esta tarea era familiarizar a los estudiantes con el ámbito del proyecto y que comenzasen a buscar y recopilar información sobre el tema. Dicha tarea se realizaría durante 3 semanas en grupos de 4 personas. Puesto que la mayoría de los estudiantes no conocían nada sobre la web semántica, los profesores escogieron 5 posibles aplicaciones sobre las que grupos de 4 estudiantes realizarían una pequeña investigación. Las aplicaciones eran: EARL[7], RDF/RSS[8], Open Directory[9], RDFCalendar[10] y FOAF[11]. Además, se dedicó a un grupo de otros 4 estudiantes a investigar sobre otras posibles aplicaciones interesantes de la Web semántica.

El objetivo de esta fase era que los estudiantes fuesen investigando por su cuenta acerca del tema de la web semántica. Además, en este momento, los estudiantes apenas conocían el lenguaje *Haskell* con el que iban a implementar el proyecto, por lo que se consideró preferible que su trabajo inicial fuese de investigación de aplicaciones.

Al finalizar el plazo (10 de noviembre), una persona elegida al azar por el profesor presentaría a toda la clase en qué consistía la aplicación que se les había encomendado. También se decidió que al acabar dichas presentaciones se realizaría una votación para elegir en cuál de dichas aplicaciones se profundizaría en el resto del tiempo de la asignatura.

Uno de los grupos propuso un tema propio acerca de la creación de un mapa de cursos que relacionase las diferentes asignaturas entre sí.

En la votación hubo un empate entre FOAF (*Friend of a Friend*), un proyecto de vocabulario basado en RDF para identificar personas y relaciones entre personas, y la propuesta del mapa de cursos. Se decidió avanzar por ambos ya que podrían estar relacionados. Al fin y al cabo, dentro de los cursos hay estudiantes y profesores, que son personas y se relacionan entre sí, por lo que el mapa de cursos podía considerarse una especialización de FOAF.

6.2. Primer prototipo

Una vez decidida la aplicación de la web semántica que se iba a tomar como objetivo se realizó una nueva división de grupos gestionada por el profesor con los criterios anteriores e intentando en esta ocasión que en los nuevos grupos no volviesen a coincidir miembros que hubiesen estado juntos en los grupos anteriores.

A cada grupo se le asignó un módulo concreto en el lenguaje *Haskell*. Se partía de una versión previa implementada por los profesores con los esqueletos de cada módulo y con varias pruebas unitarias. Se indicaba que cada grupo debía implementar pruebas unitarias antes de cualquier función y se les dejaba a los grupos que decidiesen qué funcionalidad añadir a los diferentes módulos. El objetivo, nuevamente, era que el desarrollo de cada módulo formase parte de un software común de todo el equipo.

En esta ocasión se puso una fecha tope (17 de diciembre) en la cual cada grupo presentaría al resto de la clase qué habían hecho y se publicaría la primera versión (*release*) en la página del proyecto. Dicha versión se publicó a nivel global y podía ser descargada por cualquier persona interesada. En esta fecha, un poco antes de las

[7] http://www.w3.org/WAI/ER/
[8] http://web.resource.org/rss/1.0/spec
[9] http://rdf.dmoz.org/
[10] http://www.w3.org/TR/rdfcal/
[11] http://www.foaf-project.org/

vacaciones de Navidad, se solicitó el primer informe de contribución al proyecto que debía ser realizado por cada estudiante.

6.3. Corrección de la primera versión

Como se ha indicado, la primera versión se publicó un poco antes de las vacaciones de navidad realizada por los grupos de estudiantes encargados cada uno de un módulo concreto.

A partir de ahí, se abrió un nuevo plazo, que finalizaría a mediados de enero, en el que se creó una nueva distribución de grupos asignando nuevos roles a los estudiantes para depurar la versión publicada. Había grupos que realizarían el mantenimiento de los módulos anteriores, pero además, se creó un grupo para publicación de versiones (*releases)*, otro para localización y asignación de errores (*bugs*), otro para creación de ficheros de ejemplos, otro para mantenimiento de las pruebas unitarias y otro del programa principal y sus opciones. Con estos nuevos grupos más los correspondientes a los módulos anteriores, se creó una granularidad bastante fina, lo que hacía que cada grupo fuese de 2 personas.

El objetivo de esta fase era publicar una versión corregida del proyecto a mediados de enero junto con la entrega de un nuevo informe de participación de cada estudiante.

6.4. Ampliaciones

El trabajo durante la última fase no cumplió con los objetivos previstos. Los estudiantes mostraron su malestar por el exceso de reasignación de tareas dentro del proyecto, lo que les impedía centrarse en ciertas tareas, así como por la llegada de los primeros exámenes parciales y entrega de prácticas en otras asignaturas obligatorias, lo que les limitaba el tiempo disponible para la asignatura. Solicitaron una ampliación de los plazos de entrega hasta después de los exámenes de febrero, y también indicaron que en la época de exámenes iba a ser más complicada la coordinación entre los diferentes grupos. Por ese motivo, se decidió solicitar, además del segundo informe de contribución al proyecto, un pequeño trabajo individual sobre la web semántica que sirviese de complemento al resto de valoraciones.

7. Evaluación del proceso

El último día de clase de Enero, se presentó una encuesta a los estudiantes. La encuesta contenía 2 primeras preguntas en las que se pretendía averiguar los motivos más importantes por los que habían escogido la asignatura y lo que más les había gustado de la misma. Se incluía la lista de 7 posibles criterios que debían ordenar de 1 a 7 según su importancia (1 indicaría muy importante y 7 menos importante). También se incluyeron 3 líneas en blanco para otros criterios que no se hubiesen tenido en cuenta. En el caso de que un criterio no hubiese sido rellenado (indicaría que no fue influyente) se le asignaba el valor 9. Los resultados a la pregunta de los motivos por los que se matricularon en la asignatura fueron:

Criterio	Media	δ
Quería aprender cosas de programación declarativa	2,94	1,92
Quería aprender cosas interesantes	3,55	2,14
Por el profesor	3,66	1,69
Quería participar en un proyecto colaborativo	4,88	2,25
Por el método de evaluación	5,0	2,82
Quería aprender cosas de web semántica	5,05	2,09
Quería aprender a desarrollar proyectos de software libre	5,16	1,83

Tabla 1. Criterios de matriculación

Según la encuesta, parece que los principales motivos por los que se matricularon de la asignatura fueron aprender cosas de programación declarativa, cosas interesantes y el profesor. Además de los motivos pre-establecidos, hubo 3 estudiantes que añadieron que la escogieron porque pertenecía a la intensificación del plan de estudios que querían obtener y otro estudiante que la escogió por recomendación de un amigo.

La siguiente pregunta pretendía indagar acerca de lo que más les gustó de la asignatura y de nuevo se les pedía que ordenasen los criterios. Los resultados fueron:

Criterio	Media	δ
Aprender cosas de web semántica	2,44	1,41
Aprender cosas interesantes	3,27	1,87
Participar en un proyecto colaborativo	3,83	2,24

Aprender cosas de programación declarativa	3,88	2,07
Las explicaciones del profesor	4,77	1,65
Aprender cómo realizar proyectos de software libre	5,16	1,97
El método de evaluación	5,94	2,24

Tabla 2. Criterios que más han gustado

Además la encuesta también les pedía que valorasen la técnica de enseñanza utilizada en la asignatura. En este caso, se planteaban varias afirmaciones y tenían que indicar valores entre 1 (poco de acuerdo) y 5 (muy de acuerdo) a cada afirmación. Los resultados fueron:

Afirmación	**Media**	δ
La técnica permite fomentar el trabajo en grupo	3.94	0.84
Hubiese sido preferible un examen final	1.33	0.66
Es difícil trabajar en grupo en tan poco tiempo	3.11	1.19
Esta técnica obliga a un mayor esfuerzo	3.55	0.83
He aprendido más cosas	3.44	0.76
Me ha permitido investigar por mi cuenta	3.94	1.17
Me ha permitido reconocer mis limitaciones	3.77	0.85
Hubiese preferido un método más tradicional	2.33	1.10
Es difícil compaginar con otras asignaturas similares	2.94	0.97
Hubiese preferido más técnicas no presenciales	2.5	1.16
Hubiese sido mejor un proyecto cerrado	3.83	1.30
Un proyecto abierto permite investigar	3.77	0.97
Es interesante depender del trabajo de otros compañeros	2.66	1.15
Me he desmotivado porque otros compañeros trabajaban poco	2.44	1.38
Me he desmotivado porque otros compañeros trabajaban mucho	3.44	1.34
Hubiese preferido más organización por parte del profesor	4.11	0.87
Me ha defraudado la actitud de mis compañeros	2.61	1.20
Me hubiese gustado poder dedicarle más tiempo	4.27	0.65
El método es bueno, pero debe mejorarse	4.66	0.57
El método es malo	1.5	0.60
Es interesante trabajar en un proyecto real	4.38	0.67
Voy a recomendar la asignatura a otros compañeros	3.44	1.01
Ha sido un fracaso	1.38	0.93

Tabla 3. Opinión sobre la técnica de enseñanza

Es destacable que la afirmación en la que más están de acuerdo es en que el método es bueno pero hay que mejorarlo. También están de acuerdo claramente en que les ha parecido importante trabajar en un proyecto real y en que les hubiese gustado poder dedicarle más tiempo.

Además, destacar que no están de acuerdo en que la técnica haya sido un fracaso ni en que la técnica sea mala, ni en que hubiesen preferido un método más tradicional. También indican que no les ha defraudado la actitud de sus compañeros, ni les ha desmotivado que otros compañeros trabajasen poco. Al contrario, sí hubo más respuestas indicando que se desmotivaron porque otros compañeros trabajaban mucho. Una sensación que fue percibida por el profesor. Parecía que el hecho de que algunos de los alumnos aportasen muchas cosas al proyecto hacía que otros no se sintiesen motivados para hacerlo.

La encuesta también incluía unas líneas en blanco en las que se pedía que indicasen que fue lo que menos les gustó de la asignatura y sugerencias para otros años. El aspecto que menos ha gustado entre los estudiantes fue la falta de objetivos claros desde el principio. También indicaron que no les gustó el reparto de grupos, especialmente, los cambios de grupos en mitad del curso. Otros aspectos criticados fueron el propio sistema *Sourceforge* (no por el sistema en sí, sino porque los hay mejores, que soportan Subversion), el hecho de partir de pocos conocimientos iniciales de *Haskell* (algunos sugirieron algunas prácticas iniciales de introducción al lenguaje) y el hecho de tener que coordinarse con otros compañeros en tan poco espacio de tiempo.

Durante el desarrollo de la asignatura se produjeron numerosas anécdotas, entre las que se puede destacar la propuesta improvisada por un par de estudiantes de un tema nuevo de aplicación de la web semántica destinado a la creación de mapas de cursos y titulaciones para el Espacio Europeo de Educación Superior. Otra anécdota fue la petición de colaboración de un estudiante de la Universidad de Oxford. Esa petición fue realizada gracias al ámbito público del proyecto y a la filosofía colaborativa del Software libre. Este tipo de anécdotas no se habrían producido con un método más tradicional.

8. Conclusiones

La utilización de técnicas de aprendizaje basadas en proyectos ha supuesto una novedad considerable para los profesores de la asignatura, los cuales carecían de formación en este tipo de técnicas y han tenido que improvisar en numerosas ocasiones. Sin embargo, la experiencia puede considerarse positiva; posiblemente los estudiantes hayan aprendido menos cosas de *programación declarativa,* pero sí parece claro que han aprendido más de otras habilidades como trabajo en equipo, búsqueda de información, uso de herramientas colaborativas, gestión del tiempo, etc. La duda que se plantea es si el equilibrio alcanzado es suficiente. Sería interesante desarrollar un estudio comparativo entre alumnos sometidos a distintos tipos de aprendizaje.

Respecto al profesor, creemos que es importante plantear si merece la pena el esfuerzo antes de embarcarse en esta aventura. Puede ser reconfortante y en numerosas ocasiones el profesor también aprende cosas, pero puede haber momentos de desánimo en los que apetece volver a un método más tradicional (para esos casos recomendamos la lectura de [6]). Incluso la popularidad del profesor puede verse mermada en este tipo de experiencias, ya que los alumnos comprueban los límites del conocimiento del profesor y pueden llegar a defraudarse cuando ven que el profesor no tiene respuestas para todo. Claro que ¿es necesario tenerlas?

Referencias

[1] Barg, M, K Crawford, A Fekete, T Greening, O Hollands, J Kay, J Kingston, (2000) *Problem-Based Learning for Foundation Computer Science courses,* Computer Science Education 10(2), 1—20

[2] Barrows, H. S., and R. M. Tamblyn. *Problem-Based Learning: An Approach to Medical Education*. New York: Springer, 1980.

[3] K. Beck, *Test driven-development*, Addison-Wesley, 2002

[4] C. Catalán, R. Lacuesta, A. Hernández, *Cambio de modelos basados en la enseñanza a modelos basados en el aprendizaje: una experiencia práctica.* I Simposio Nacional de Docencia en Informática, SINDI'05, Granada, 2005

[5] P. Dart, L. Johnston, C. Schmidt. Enhancing *Project-Based Learning: Variations on Mentoring*, Proceedings 1996 Australian Software Eng. Conference, pp. 112-117, 14-18 July 1996.

[6] R. M. Felder, *We never said it would be easy*, Chemical Engineering Education, 29 (1) 32-33 (1995)

[7] Solomon, Gwen (2003), *Project-Based Learning: a Primer*. Technology and Learning, 23(6), 20-30.

[8] G. Klyne, *Using datatype-aware inferences with RDF*. Noviembre, 2003 `http://www.ninebynine.org`

[9] J. E. Labra, *Representaciones gráficas y mundos virtuales infinitos en las prácticas de programación lógica y funcional.* IX Jornadas de Enseñanza Universitaria de la Informática, Cádiz, Julio, 2003

[10] J. E. Labra, *Programación Declarativa utilizando XML, representaciones gráficas y mundos virtuales infinitos.* I Simposio Nacional de Docencia en Informática, SINDI'05, Granada, 2005

[11] N. Postman, C. Weingartner, *Teaching as a subversive activity*, Delacorte Press, New York, 1969

[12] *Tuning Education Structures in Europe*. Final Report, Pilot project Phase I and II