Integración de teoría y práctica / gestión y desarrollo en la docencia de la ingeniería del software

Pablo Gervás
Dept. de Sistemas Informáticos y Programación
Universidad Complutense de Madrid
28040 Madrid
e-mail: pgervas@sip.ucm.es

Resumen

Se presenta una propuesta de integración y coordinación entre las partes teórica y práctica de la ingeniería del software destinada a: facilitar la disponibilidad de especificaciones de partida, paliar las restricciones temporales para el desarrollo de proyectos a gran escala, proporcionar a los alumnos una experiencia real de trabajo en equipo y de uso de la documentación para la comunicación dentro de un equipo.

1. Motivación

Muchas de las técnicas que se utilizan hoy en día en ingeniería del software están dirigidas a resolver el problema básico de cómo convertir la tarea de desarrollo de software en un conjunto de subtareas disjuntas que puedan asignarse a partes distintas de un equipo de desarrollo sin dar lugar a una explosión de las necesidades de comunicación interna de todo el equipo [1]. Tareas como documentar los requisitos una vez aclarados con el cliente, describir explícitamente la planificación de las tareas en forma de diagramas de PERT o de Gantt, o generar modelos detallados del diseño [8,9,12] tienen como uno de sus objetivos principales el plasmar el resultado de una de estas subtareas disjuntas en que se ha repartido el proceso de desarrollo, de modo que ese resultado pueda ser utilizado por otra parte del equipo de desarrollo sin necesidad de consultas frecuentes o prolongadas con los miembros del equipo que llevaron a cabo la subtarea original. Esta virtud también puede entenderse de forma diacrónica, de modo que el abandono de miembros del equipo que habían realizado subtareas en el pasado no suponga que el resultado de esas tareas deje de estar disponible para el resto del equipo.

Estas verdades genéricas son a menudo demasiado abstractas para ser captadas en toda su magnitud por los alumnos, que están demasiado entusiasmados con la posibilidad de poner a prueba su recién adquirida capacidad para diseñar e implementar lo que sea menester. Para el tamaño de aplicación que resulta viable desarrollar en el marco de una única asignatura, el alumno medio tiene en la mayoría de los casos capacidad suficiente para retener en la cabeza tanto una visión general de la especificación, como una idea de la planificación suficiente para organizarse, como un esbozo del diseño que le basta para irlo programando. De hecho, esta manera de "documentar" sus propios procesos de trabajo tiene la ventaja importante de llevar incorporados sus propios métodos innatos de mantener un seguimiento continuo durante el proceso de desarrollo, revisar periódicamente esa "documentación", corregir lo que sea necesario, comunicar las modificaciones realizadas a quien puedan afectar (normalmente un proceso paralelo pendiente del propio alumno), y de forma general llevar a la práctica muchas de las recomendaciones básicas de los textos de ingeniería del software, todo ello con un esfuerzo mínimo de tiempo y un gasto de papel y tecnología casi nulo. Esta es la razón de que, en determinadas circunstancias, se pueda desarrollar software con éxito sin aplicar explícitamente técnicas de ingeniería del software.

La labor de enseñar ingeniería del software es una tarea ingrata porque requiere a menudo ponerle al trabajo de los alumnos trabas especiales que se lo hagan más difícil, con el bien intencionado objeto de que aprendan maneras de

hacerle frente a dificultades de ese tipo [3]. Una versión de esta labor radica en imponer la necesidad de trabajar en equipo, lo cual obliga a hacer explícitos parte de esos procesos mentales de "documentación" que mencionaba antes. Aún así, para equipos pequeños, es factible el llegar a sincronizar "documentaciones" mentales del proceso de los participantes mediante una comunicación verbal periódica, sin necesidad de documentos formales o modelos explícitos.

Si imponemos la necesidad de documentar explícitamente como un requisito *sine qua non* del trabajo de la asignatura, independientemente de la cantidad de explicaciones teóricas que aportemos, es probable que los alumnos (siempre sometidos a presiones temporales varias) adopten la solución de llevar a cabo el trabajo de desarrollo de la manera en que les resulta más cómodo (sin ingeniería del software) y generen *a posteriori* la documentación que se les pide (ahorrándose de ese modo las distintas iteraciones sucesivas, el seguimiento, la revisión, la modificación...). Si se atienen al orden explicado en teoría generarán una serie de documentos a cada paso que reflejan esas "documentaciones" mentales que ellos se van construyendo en paralelo, pero rara vez volverán a consultarlos más adelante, resultándoles mucho más cómodo referirse a la versión que recuerdan de cuando redactaron el documento. En el mejor de los casos lo harán convencidos de que la teoría la tienen clara y, si llega el momento, sabrán aplicarla. Quizá uno de los desafíos más importantes en la docencia de la ingeniería del software hoy en día sea el conseguir que ese momento llegue, y que llegue en el marco del trabajo práctico asociado a la asignatura.

2. Problemas a resolver

Es importante combatir la visión, frecuente entre los alumnos y en parte inducida por la posición de la asignatura dentro del plan de estudios, de que la ingeniería del software es simplemente una manera rígida, dogmática y prescindible de encorsetar la programación. Se presentan por orden de importancia relativa los problemas que se pretende resolver con la propuesta.

2.1 Redundancia del trabajo de documentación

Un problema fundamental es el ya mencionado de proporcionar un marco en el que la labor de documentar explícitamente los procesos de desarrollo no sea un trabajo vano, sino que cumpla un cometido concreto con objetivos fácilmente perceptibles y evaluables por el alumno.

2.2 Desarrollo a pequeña o gran escala

Se pretende atacar de maneras diferenciadas dos problemas distintos. Por un lado los alumnos necesitan tener la oportunidad de enfrentarse a título individual a cada una de las cuestiones relevantes que se tratan en la asignatura. Esto requiere que se trabaje en equipos pequeños para que todos los integrantes de un equipo tengan ocasión de participar en los procesos que se lleven a cabo, y adquieran experiencia en el uso de las técnicas y en la resolución de los problemas. Por otro lado, las técnicas de ingeniería del software están pensadas para hacer frente a circunstancias de trabajo que son específicas de (o al menos mucho más frecuentes en) situaciones en que se trabaja con proyectos grandes, con participación de muchos trabajadores, y con una expectativa significativa de continuidad y consistencia del proceso de desarrollo más allá de un proyecto o una plantilla de personal concretos. Esto hace muy deseable que los alumnos experimenten, en la medida de lo posible, circunstancias de trabajo de estas características, para que perciban los problemas que acarrean, la dificultad de solucionarlos, y el papel de las tecnologías de ingeniería del software a la hora de resolverlos.

Las consideraciones curriculares de instituciones importantes son unánimes a la hora de recomendar la inclusión de un proyecto de desarrollo a gran escala en la formación de todo informático [10]. En [5] se presentaban distintos métodos empleados en universidades de EEUU y el Reino Unido para solventar esta cuestión. La mayor parte de ellos no son viables en el contexto que nos ocupa. La única alternativa viable sería la denominada "curso con proyecto encapsulado", consistente en que los estudiantes realicen un proyecto de desarrollo de software después de terminar la mayoría del curso. La presente propuesta pretende adelantar el momento de comienzo del trabajo práctico, mediante una reorganización adecuada de los contenidos teóricos por un lado, y la inclusión de un proyecto a gran escala para el segundo cuatrimestre que entronca directamente con prácticas a pequeña escala realizadas durante el primer cuatrimestre. De este modo se dispone de más tiempo total para

el desarrollo y, para cada concepto, se acercan más en el tiempo el momento de su exposición teórica y el de su posterior puesta en práctica.

2.3 Restricciones temporales

Es difícil desarrollar un proyecto a gran escala dentro del marco temporal que impone una asignatura concreta dentro de un plan de estudios, y que cada alumno llegue a ver finalizado el proyecto en el que ha trabajado, al menos hasta un punto suficiente como para permitirle hacer una valoración personal del rendimiento conseguido con su esfuerzo, y de la medida en que las técnicas utilizadas han servido para hacer frente a los problemas encontrados. Esta dificultad aumenta si además se pretende conjugar el desarrollo de un proyecto a gran escala con la realización de prácticas a pequeña escala.

2.4 Disponibilidad de especificaciones

A la hora de plantear prácticas y proyectos para la docencia de la ingeniería del software es difícil encontrar material de partida sobre el que los alumnos puedan desarrollar una especificación sobre la que basar ejercicios de diseño.

La solución ideal sería conseguir clientes reales que estuviesen dispuestos a ejercer como tales para algún proyecto de formación. Esta solución suele ser inviable por dificultades de disponibilidad de clientes, compatibilidad de horarios de posibles clientes y estudiantes, voluntad de los clientes de aceptar las restricciones asociadas (temporales, en la tecnología o el tamaño de las aplicaciones a realizar), y de imposibilidad de garantizar un servicio de mantenimiento o de seguimiento más allá del final de la asignatura.

Otra solución es dejar el papel del cliente en manos del profesor. Esta solución plantea varias dificultades. Por un lado, el profesor de ingeniería del software es demasiado consciente tanto de cómo debe ser la especificación ideal como de cuáles son los errores más habituales de los clientes o de los ingenieros del software, y corre el riesgo de proporcionar una entrada como cliente que no sea significativa. Por otro lado, si el número de alumnos es apreciable, el profesor deberá o dedicar un momento distinto a atender a cada grupo (con un proyecto distinto, lo que puede dificultar mucho su labor) o decidir que atiende a todos los grupos a la vez (con lo cual se desvirtúa el propósito del ejercicio). La alternativa de proporcionar la especificación por escrito reduce, a mi entender, el valor del ejercicio, puesto que restringe las posibilidades de poner en práctica el proceso de captura de requisitos tal como se describe en los textos básicos de la materia (en términos de entrevistas con el cliente, iteraciones sucesivas, negociación a partir de prototipos...[9,12]).

3. Formato de la propuesta

La presente propuesta pretende atacar estos problemas y se ha implementado este año en la asignatura de Ingeniería del Software, impartida en la titulación superior de Ingeniería Informática de la Facultad de Informática de la Universidad Complutense de Madrid. Esta asignatura forma parte del segundo ciclo, y concentra todos los contenidos del plan de estudios sobre el tema de ingeniería del software. Se trata de una asignatura de 18 créditos (12 teóricos y 6 prácticos). Los alumnos que la cursan ya han estudiado estructuras de datos, metodología y tecnología de la programación, y programación orientada a objetos.

La innovación de la propuesta radica en una manera específica de coordinar la teoría y la práctica, y en un planteamiento de la parte práctica de la asignatura orientado a dar respuesta a los problemas que se han reseñado. El orden en que se va cubriendo la parte teórica se ha definido específicamente para: garantizar que los alumnos pueden empezar a trabajar en la parte práctica en los primeros momentos del curso, y proporcionar a cada paso del desarrollo práctico aquel material teórico que es de aplicación directa en el punto alcanzado. Este paralelismo se mantiene durante los primeros meses, cuando los alumnos están absorbiendo los conceptos teóricos que necesitan para trabajar en la práctica, y se relaja ligeramente hacia el final del curso, cuando los alumnos ya están desarrollando, tienen experiencia en el proceso de desarrollo, y toleran mejor el planteamiento de alternativas, generalizaciones, o discusiones más teóricas. En este punto la teoría se desvincula de la práctica para cubrir el resto del temario contenido en el plan de estudios. La distribución temporal propuesta aparece en la figura 1. A continuación detallamos aquellas características de la propuesta que pretenden resolver los problemas descritos.

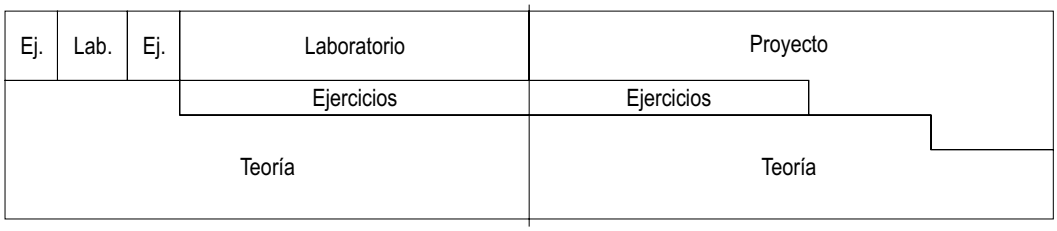

Figura 1. Distribución temporal

3.1 Parte teórica

Para hacer posible una solución del tipo que se propone, es necesario proporcionar a los alumnos directrices muy claras sobre la manera de llevar a cabo este tipo de procesos. Se ha optado por recurrir al Proceso Unificado de Desarrollo [8] de Rational como marco teórico en el que encuadrar la propuesta. Esto supone que el proceso de desarrollo a realizar en la parte práctica tendrá lugar conforme a un *plan de fase* en el que se recorran tres fases distintas: de *inicio*, en la que se reúnen los requisitos y se planifica el proceso a grandes rasgos, de *elaboración,* en la que se establece la arquitectura y se diseña el sistema, y de *construcción*, en la que se da forma concreta al sistema[1]. La cuarta fase (de *transición*) descrita en la teoría, más relacionada con la instalación de la aplicación, su depuración final y mantenimiento se obviará para hacer frente a las restricciones temporales. Cada fase tendrá una serie de *iteraciones*, cada una de las cuales dará lugar a una *construcción* del sistema. En cada iteración, según el punto del plan de fase en que se encuentre, se han de incluir proporciones variables de los siguientes flujos de trabajo: captura de requisitos, análisis, diseño, implementación y prueba (más de los primeros en las iteraciones correspondientes a la fase de elaboración, más de los últimos en las correspondientes a la fase de construcción). Además se han de aplicar las siguientes tareas de gestión de proyectos: estimación, planificación, gestión de configuración, y gestión de calidad.

El orden de exposición de los distintos contenidos teóricos mencionados se defiende más adelante en relación con la necesidad de coordinarlo con el trabajo práctico a desarrollar.

[1] Estas descripciones de cada una de las fases son a grandes rasgos y no pretenden ser exhaustivas. Para más datos ver [7,8].

Es irreal suponer que desde el primer momento los alumnos van a ser capaces de aplicar con éxito en la práctica un modelo de proceso tan detallado, casi al mismo tiempo que están descubriendo ese modelo en las exposiciones teóricas. Sin embargo se considera interesante que los alumnos tengan la experiencia de gestionar su propio trabajo. Para resolver este dilema se ha optado por establecer una distinción entre los mecanismos de gestión aplicados en las distintas partes del curso. Durante el primer cuatrimestre los alumnos trabajan en grupos pequeños gestionados por el profesor, que impone fechas de entregas, orden de trabajo, métodos a seguir...; durante el segundo cuatrimestre trabajan en grupos grandes que se autogestionan.

3.2 Parte práctica

La parte práctica que se propone consta de dos partes diferenciadas: una componente de desarrollo a pequeña escala (prácticas en equipos de 2 a 3 personas) y una componente desarrollo a gran escala (proyecto en equipos de 20 personas). La parte de desarrollo a pequeña escala se lleva a cabo durante el primer cuatrimestre (gestionada por el profesor en lo relativo a tareas a realizar, plazos de entrega...), y la parte de proyecto se lleva a cabo durante el segundo cuatrimestre (gestionada ya por los propios alumnos). Con esta división se consigue dar cabida a los beneficios discutidos en el apartado 2.2 para cada una de las alternativas (pequeña escala y gran escala). Las dos se desarrollan siguiendo el Proceso Unificado de Desarrollo.

Para evitar los problemas de restricciones temporales descritos en el apartado 2.3, se sugiere el uso de un modelo de proceso iterativo e incremental, que permita detener el proceso en el momento en que termine el primer cuatrimestre, quedando la aplicación a desarrollar en el punto en que estuviera al terminar la iteración correspondiente.

Se impone la restricción de que cada proyecto durante el segundo cuatrimestre debe desarrollarse a partir de una de las especificaciones que hayan resultado de las prácticas del primer cuatrimestre.

Esta restricción ya hace cobrar importancia a la documentación generada durante las prácticas del primer cuatrimestre, puesto que ha de servir más adelante para que un grupo mayor de alumnos base en ella su proyecto del segundo cuatrimestre. A pesar de eso se ha preferido insistir en potenciar la importancia de la documentación explícita, por considerar que la posibilidad de que una práctica concreta no fuese elegida para ser continuada podría dar excusas para no tomarse en serio su documentación adecuada. Para garantizar que todas las prácticas reciben la atención adecuada, se impone un mecanismo de rotación periódica de aplicaciones que se describe más adelante.

3.3 Prácticas a pequeña escala con intercambio

El orden que se propone inicialmente para estas etapas es: desarrollo de una propuesta, gestión de configuración, estimación de esfuerzo para la propuesta desarrollada, captura de requisitos (diagrama de casos de uso y requisitos no funcionales), planificación, y análisis (diagramas de clases y de secuencia).

Al final de cada etapa, cada equipo propone el material que ha desarrollado (junto con el que se recibiera al principio de la etapa como resultado de etapas anteriores) a otro equipo para que lo desarrolle en la etapa siguiente. Este intercambio se desarrolla por orden de listado de los equipos para evitar problemas. De este modo, cada equipo participa en la elaboración de material propio correspondiente a todas las etapas, y a la vez adquiere la experiencia de trabajar sobre material ajeno. A cada paso deberán tener en cuenta toda la documentación generada por los distintos grupos que hayan trabajado sobre el problema. Este proceso expone a los alumnos a la utilidad práctica real de los documentos que se les van exigiendo en cada etapa.

Como mecanismo fundamental de intercambio se propone el uso de una página web donde se encuentra disponible el material correspondiente a cada proyecto. Cada grupo actualizará al final de cada etapa el material disponible para el proyecto sobre el que ha estado trabajando. Este mecanismo de intercambio sirve adicionalmente como primera experiencia de los alumnos en temas de gestión de configuración.

A continuación se presentan detalles breves del tipo de trabajo involucrado en cada etapa y las ventajas que supone en cada una la obligación de intercambiar material:

Redacción de una propuesta de proyecto: En grupos de 3 personas, los alumnos deben establecer el contexto de gestión de un proyecto imaginario, y redactar la propuesta de proyecto. Cada grupo establece un documento preliminar sobre el proyecto a acometer, incluyendo básicamente: objetivos, usuarios, funcionalidad, requisitos de hardware, posible división en módulos. Aunque cada equipo trabaja con la participación exclusiva de los desarrolladores, mediante este ejercicio se consiguen simular muchas de las condiciones reales que se dan en una primera reunión con un cliente: tiempo limitado, imprecisión inicial muy grande respecto a los objetivos, diferencias de opinión entre los participantes, presión por conseguir un documento (por básico que sea) en el que se detalle un primer esbozo, desconocimiento del alcance real del proyecto por no haberse recopilado todavía formalmente los requisitos...

Gestión de configuración: Desde el momento en que un grupo necesita proporcionar material a otro aparecen ya problemas de formato de materiales, de protocolos de elección de nombres... Esto proporciona una ocasión para explicar la teoría relativa a esta parte de la gestión de configuración. Asimismo se introduce ya el concepto de modificación de un documento recibido como definitivo (raro es el grupo al que no le gustaría hacer modificaciones a la especificación que recibe, sea para hacerla más atractiva o más fácil de obedecer), y del proceso a seguir para anotarlo y documentarlo.

Estimación: Se pretende obtener una idea de cuánto dinero y cuánto esfuerzo (cuánta gente durante cuánto tiempo) pensamos que va a requerir el proyecto que se propone. Aunque no se disponga de datos a estas alturas para llevar a cabo una estimación válida, se trabaja en condiciones similares a las que se darían en una situación real.

Requisitos: Deben realizarse la identificación de actores y casos de uso, la representación de los mismos en UML, y la redacción de descripciones de flujos de eventos de un caso de uso. Se utiliza

una herramienta software para facilitar la representación.

Planificación: En el momento en que hay una serie de casos de uso y una estimación de esfuerzo disponibles puede empezarse a planificar. El intercambio de material supone una dificultad añadida tan solo en el sentido de que cada equipo planifica, no su propio trabajo futuro, sino el del equipo que va a heredar el material que está preparando. Las restricciones de personal correspondientes deben ser tenidas en cuenta. Se utiliza una herramienta sofware para la planificación de proyectos.

Análisis: Se elaboran diagramas de clases y de interacción a partir de los diagramas de casos de uso dados (también con ayuda de herramienta software). Cada grupo procesa diagramas de casos de uso sobre proyectos que no ha visto hasta ese momento, para enfatizar que toda la información necesaria para el desarrollo debe haberse detallado durante la especificación (sea en los diagramas o en las descripciones textuales en forma de requisitos adicionales).

Aunque sería posible (e incluso deseable) continuar con este patrón de trabajo hasta completar un recorrido completo de los flujos de trabajo de desarrollo (diseñando, implementando y probando los proyectos), nuestra experiencia hasta la fecha es que el proceso de compaginar la exposición teórica y la puesta en práctica reduce el ritmo al que se puede cubrir la materia, y, teniendo en cuenta que se ha de partir de un cierto marco teórico imprescindible antes de poder empezar el trabajo práctico, que el mecanismo de intercambio requiere un proceso de "digestión" de la documentación recibida antes de que el grupo pueda empezar a producir, y que a cada paso los alumnos deben aprender a dominar las técnicas nuevas que se van introduciendo, no es realista aspirar a cubrir más etapas prácticas en un solo cuatrimestre. Por otro lado esto tiene la ventaja de poder dedicar las últimas semanas de exposición teórica del cuatrimestre a proporcionar a los alumnos los conceptos básicos sobre diseño (y en menor medida gestión de calidad) que van a necesitar para poner en marcha con buen pie el proyecto a gran escala en el segundo cuatrimestre.

3.4. Proyecto a gran escala

Al término del primer cuatrimestre, las prácticas realizadas han dejado como resultado una página web con una serie de proyectos con propuestas, estimación, especificación, planificación, y modelo de análisis. De todo este material, el correspondiente a las primeras etapas de las prácticas ha sido revisado por varios equipos, y cada uno ha generado no sólo versiones mejoradas sino documentos adicionales de justificación de las modificaciones. Todo este material se aprovecha como punto de partida para los proyectos a gran escala a desarrollar en el segundo cuatrimestre, paliando así parcialmente tanto las restricciones temporales que supondría el empezar un proceso de desarrollo a gran escala cuando ya solo quedan tres meses de tiempo útil, como la disponibilidad de especificaciones.

Todos los alumnos han tenido ocasión de familiarizarse con la elaboración de los distintos materiales. Aunque cada grupo ha realizado un ejemplo de planificación, el trabajo de todos los grupos hasta ese punto ha sido planificado por el profesor, en el sentido de que ha impuesto las tareas a realizar (elaboración de los distintos materiales) y los hitos (fechas de entrega de cada etapa). Asimismo, los alumnos han operado bajo un mecanismo básico de gestión de configuración (propuesto por el profesor) en forma de página web.

A partir de este punto se establecen grupos mucho más grandes (en torno a 20 personas por grupo) para continuar el trabajo práctico durante el resto del curso. Cada uno de estos grupos tiene la responsabilidad de gestionar el desarrollo de su proyecto. Esto supone tanto la planificación del trabajo, como la gestión de configuración.

El único requisito impuesto por el profesor es que se debe planificar el trabajo en tres iteraciones (una por cada mes restante de curso), y que al terminar cada iteración deben revisarse no solo los productos concretos obtenidos sino los procedimientos empleados para conseguirlos.

La exposición teórica se desvincula del desarrollo concreto de cada proyecto, pero sigue estando en las primeras etapas orientada a proporcionar conceptos necesarios.

Una de las dificultades principales que encuentran los alumnos en este punto es la falta de criterios para elegir entre los distintos proyectos propuestos. Puesto que tienen que elegir sobre todo en función de la documentación que hay generada, esto supone un trabajo serio de revisar la documentación disponible, y esto se aprovecha

para impartir la teoría relativa a la calidad del software y los proceso de gestión de calidad en el desarrollo. Estos conceptos se presentan ligados a la planificación en tres iteraciones y a la necesidad de revisar los procedimientos de trabajo en cada iteración, con objeto de mejorar la calidad.

Para hacer esto posible se presenta el concepto de *revisión formal técnica* (*Formal Technical Review* o FTR) como el principal método de validación de la calidad de un proceso o de un producto. Un grupo debe examinar parte o todos los resultados de su trabajo (planes, especificaciones, modelos, diseños, código, casos de pruebas, documentación,...) para buscar problemas potenciales. Se insiste en que este proceso tiene ventajas adicionales a la hora de garantizar que todos los miembros del grupo, por haber participado en revisiones del trabajo de sus compañeros, estarán familiarizados con él y podrán hacerse cargo de él en caso de ausencia, abandono, o enfermedad del compañero.

Así mismo, se explica el concepto de *métrica* como medida específica que se toma de un proceso para poder comprobar objetivamente si está funcionando como es debido. Se exige a los alumnos que en cada iteración diseñen las métricas que consideren necesarias para evaluar sus procedimientos de trabajo y recopilen los datos pertinentes. Estos datos se utilizarán en las revisiones técnicas formales de los procedimientos establecidos que se han de realizar al principio de cada nueva iteración.

Finalmente se introduce el concepto de sesión de evaluación de un proyecto, en la que cada grupo debe hacer un resumen de la situación del proyecto en el que trabaja, incluyendo los procedimientos que están empleando, los problemas que han tenido, el ajuste a la planificación inicial, y datos objetivos que tengan disponible sobre su rendimiento hasta la fecha. Este tipo de sesiones cumplen un objetivo múltiple. Por un lado, permiten a cada grupo enterarse de los problemas y soluciones que está experimentando los demás, maximizando el número de experiencias concretas a las que se ve expuesto cada alumno, aunque sea indirectamente. Por otro lado, obligan a los alumnos tanto a realizar un seguimiento del desarrollo del proyecto como a plantearse la manera en que el desarrollo observado puede presentarse a terceras personas de manera coherente e informativa. Por último, permite al profesor mantener un seguimiento y evaluación de la labor de los alumnos. Esto garantiza que se pueda detectar y remediar a tiempo posibles errores de concepto (que constituyen un riesgo importante cuando se está dejando la labor de gestión a los alumnos), o problemas serios en la interacción personal dentro de los grupos. Al mismo tiempo, simulan en cierta medida la necesidad real en el mundo de la empresa de rendir cuentas sobre el desarrollo de proyectos a la superioridad.

5. Valoración y conclusiones

Los problemas que se han planteado como objetivos son conocidos en la literatura sobre enseñanza de ingeniería del software [10,5,4]. Las distintas soluciones que se han propuesto históricamente han ido dando forma a los planes de estudios de ingeniería del software durante los últimos años, incluyendo cada vez más habitualmente una componente de desarrollo de proyectos. Aún así, es bastante habitual el fragmentar la docencia de la materia en dos partes disjuntas: una que concentra la materia relacionada con la gestión de proyectos, y otra centrada en la especificación y diseño de software. La presente propuesta está diseñada considerando que el proceso de desarrollo es una tarea compleja cuyas máximas dificultades residen no tanto en las subtareas concretas en que se puede subdividir sino precisamente en el proceso de división del trabajo original y de posterior integración de los resultados obtenidos. En este sentido puede entenderse la labor de especificación y diseño como una división de la funcionalidad (caso de uso) y la arquitectura (módulos) de una aplicación, y la labor de gestión como una división del trabajo en tareas que se asigna a distintos momentos y distintos miembros del equipo. En ambos casos el desafío mayor no es el establecer una división inicial, sino el integrar los resultados. Los problemas de integración se multiplican cuando además de integrar a ese nivel hay que integrar la división de gestión con la división de especificación y diseño. En esta propuesta se pretende integrar desde el primer momento todo lo posible estas subtareas, y facilitar al alumno un ejemplo de cómo todas las tareas que se le proponen interactúan en un proceso de desarrollo real.

Existen propuestas recientes [3,13] que pretenden dar solución a los mismos problemas. En [3] se propone asignar a cada equipo (de 3 personas) la gestión de proyectos y un presupuesto de puntos de evaluación que han de utilizar para contratar programadores de entre los miembros de otros grupos. En [13] se proponen una serie de 'trucos sucios' que se emplean para dificultar la tarea de los alumnos con objeto de que el proceso de desarrollo sea más afín a la experiencia real que han de sufrir luego. La presente propuesta proporciona ventajas similares sin necesidad de imponer criterios mercantilistas ni de que el profesor se presente como antagonista del alumno.

Esta propuesta es un refinamiento de otra anterior [6] en la que se proponía utilizar el modelo de madurez de la capacidad de software (SW CMM) [2] como referencia para que los alumnos intentaran a lo largo del curso constituir una 'empresa' de desarrollo con sus propios procedimientos de trabajo documentados. Se proponía intentar alcanzar el nivel 3 (cada grupo debía en el proyecto definir su propio proceso). La experiencia resultó positiva por cuanto que los alumnos desarrollaron por escrito versiones muy razonadas de los procedimientos de trabajo, pero no quedó claro hasta qué punto habían conseguido aprender de verdad a implantarlos en la práctica. En la propuesta de este año se ha introducido la idea de que la primera parte del proceso es gestionado por el profesor, lo que libera a los alumnos durante ese periodo para que puedan centrarse en dominar las técnicas. En términos prácticos se está tomando como referencia el nivel 2 (la asignatura, entendida como una colección de proyectos bajo la misma gestión general - la del profesor - sigue un proceso más o menos establecido - el que se ha discutido en teoría - pero que no está definido específicamente en ningún sitio - no hay manual de gestión de proyectos para la asignatura). Hasta la fecha (mediados de mayo) se vienen cumpliendo las previsiones. Está por ver si los resultados académicos ratifican la validez pedagógica del método.

Referencias

[1] Frederick P. Brooks, Jr. *The Mythical Man-Month.* Addison-Wesley, 1995 (1ªed. 1975).

[2] Carnegie Mellon University, SEI. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison-Wesley Publishing Company, Reading, MA, 1995.

[3] Ray Dawson, Twelve Dirty Tricks to Train Software Engineers, ICSE 200, ACM.

[4] Joint IEEE/ACM Task Force. *Computing Curricula 2001. A Vision of the Overview Volume*, 2001, Steelman Draft

[5] Ford, G. 1991 SEI Report in Graduate Software Engineering Education, SEI, Carnegie Mellon University, Pittsburgh, Tech. Rep. CMU/SEI-91-TR-2

[6] P. Gervás, M.A. Gómez, A. Sarasa. "Ingeniería del software: ¿basta con desarrollar proyectos o haría falta probar a implantar procesos de desarrollo a largo plazo?", JENUI 2001.

[7] P.Kruchten. *The Rational Unified Process. An Introduction. Second Edition*. Addison - Wesley, 2000.

[8] I. Jacobson, G. Booch, J. Rumbaugh. *UML. El proceso unificado de desarrollo de software*. Ed. Addison Wesley Iberoamericana, 2000.

[9] R.S.Pressman. *Software Engineering: A Practitioner's Approach*. European adaptation, 5th edition. McGraw-Hill, 2000.

[10] Software Engineering Coordinating Committee (SWECC). *Guide to the Software Engineering Body of Knowledge*. http://www.swebok.org/

[12] I.Sommerville. Software Engineering. 6th edition. Addison-Wesley, 2001.

[13] K. Todd Stevens, Experiences Teaching Software Engineering for the First Time, ITiCSE 2001, ACM.