Ingeniería del software: ¿basta con desarrollar proyectos o haría falta probar a implantar procesos de desarrollo a largo plazo?

Pablo Gervás Gómez-Navarro, Marco Antonio Gómez Martín, Antonio Sarasa Cabezuelo

Dept. de Sistemas Informáticos y Programación
Universidad Complutense de Madrid
28040 MADRID
e-mail: {pgervas,marcoa,asarasa}@sip.ucm.es

Resumen

Para la asignatura de ingeniería del software se propone un marco general de trabajo que enfrente a los alumnos a proyectos más amplios que les supongan desafíos similares al desarrollo real de software en la empresa. Toma la forma de una sucesión de concursos entre equipos de alumnos para optar a contratos para desarrollar aplicaciones. Los pliegos de condiciones de los concursos, redactados por el profesor, exigen para cada concurso un grado mayor de implementación de procesos de ingeniería del software por parte de los equipos candidatos. Este proceso se compagina con exposiciones teóricas del material necesario en cada etapa, pero siempre orientado a que cada equipo lo aproveche para hacer frente al concurso siguiente. De este modo los alumnos se ven enfrentados a cada nuevo concepto como requisito de un concurso *antes* de que les sea explicado, y desde el primer momento tienen presente el objetivo hacia el que se dirigen las sucesivas ampliaciones de las metodologías de trabajo que se les van proponiendo.

1. Motivación

La ingeniería del software surge para hacer frente a una serie de circunstancias específicas del desarrollo de software en grandes empresas que hacen necesarias técnicas y metodologías apropiadas para gestionar la complejidad en distintas escalas. Sin pretensión alguna de ser exhaustivo (para más detalles, ver [7, 11]), se pueden mencionar por ejemplo las siguientes:

- número elevado de trabajadores involucrados,
- cambios de personal durante un proyecto concreto,
- proyectos de larga duración,
- no existe una especificación de partida ni del problema ni de su posible solución sino que ésta debe generarse como parte de la labor a realizar,
- posibilidad de cambio de los requisitos mientras el proyecto está en curso,
- la calidad de trabajo debe mantenerse a lo largo del tiempo abarcando varios proyectos...

En el momento de enfrentarse a la ingeniería del software la experiencia de la mayor parte de los alumnos de carreras universitarias de informática se reduce a haber cursado asignaturas varias del tipo de laboratorios de programación en las que las circunstancias son muy distintas. Típicamente, el bagaje de experiencia como programador que aporta cada alumno puede caracterizarse (dentro del mismo espíritu de ejemplificar, no de definir) por:

- no han trabajado casi nunca en equipos de más de tres o cuatro integrantes,
- están acostumbrados a trabajar a partir de enunciados con muy pocas ambigüedades y calculados para facilitar la labor de corrección y evaluación por parte de los profesores,
- están acostumbrados a plantear su trabajo a corto plazo, en contextos muy específicos de un profesor concreto en una asignatura concreta, hasta el punto de que el alumno que mejor consiga ajustar su trabajo a las preferencias personales del profesor más probabilidades tiene de obtener buenos resultados,

- los equipos en los que han trabajado los crean los propios alumnos específicamente para trabajos concretos y se disuelven una vez realizados,
- jamás se han visto en la necesidad de retocar o reutilizar código generado por otros o por ellos mismos en el pasado...

Existe un abismo de diferencias entre estas dos situaciones. Uno de los escollos que encuentra habitualmente la docencia de la ingeniería del software es la dificultad para hacer comprender a los alumnos la necesidad (e incluso la propia utilidad) del conjunto de técnicas que se han desarrollado para hacer frente a las circunstancias específicas de la empresa, que le resultan extremadamente ajenas. El alumno asiste de manera pasiva a la exposición del contenido teórico de la asignatura, sin tener referencias concretas mediante las que anclarlo a sus experiencias pasadas. Puesto que esto suele ocurrir, dada la ubicación temporal de esta asignatura en la mayoría de los planes de estudio de las universidades españolas, en un momento en que el alumno empieza a sentirse capaz de programar (después de unos años de trabajo duro en que se ha sentido inútil, por fin consigue que sus aplicaciones funcionen), y muy tentado de dar por hecho que empieza a saber más que sus profesores, los alumnos tienden a mostrarse escépticos ante las metodologías presentadas. Por esta razón es especialmente importante encontrar maneras alternativas de plantear el acercamiento a la materia que consigan motivar al alumno y situarle en un punto de partida desde el que le resulte fácil entender y contextualizar el contenido teórico que se le va a impartir.

La presente propuesta constituye un intento de atacar este problema en el contexto de 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 del plan de estudios vigente, y es la que concentra todos los contenidos del plan de estudios sobre el tema de ingeniería del software.

2. La propuesta

El problema planteado se puede intentar resolver de varias maneras.

Una primera posibilidad correspondería a la solución tradicional de recurrir a la clase magistral para intentar hacer entender las circunstancias que motivan la aparición de la disciplina.

Otra posibilidad es proponer una componente práctica para la asignatura en la que se intenta que los alumnos vivan la experiencia de trabajar utilizando las técnicas de la ingeniería del software.

Dentro de esta posibilidad, una alternativa es establecer las técnicas y metodologías como requisitos formales para aprobar la asignatura y plantear a los alumnos pequeños trabajos, no necesariamente muy distintos de los que ya han realizado en otras asignaturas, pero insistiendo en que deben de atenerse a las técnicas y metodologías so pena de suspender. En ese caso el alumno tiende a aplicar las metodologías sin llegar a entender el sentido que tiene aplicarlas, y probablemente acabe por desarrollar una cierta aversión hacia ellas.

Otra alternativa es enfrentar a los alumnos a proyectos más amplios, que les supongan desafíos similares al desarrollo de software real en la empresa. El riesgo en este caso radica en que el tiempo disponible para llevarlos a cabo es siempre limitado, lo cual acota la magnitud posible de los proyectos acometibles. Otra dificultad añadida es que una parte muy importante del esfuerzo del alumno para esta asignatura se difumina por este camino en codificar soluciones concretas, labor más acorde para otras asignaturas del plan de estudios. Por otro lado, incluso si se consiguiesen simular las magnitudes de equipos de trabajo del desarrollo de software profesional, sigue estando fuera de alcance el simular muchos de los factores que complican esta actividad en la realidad.

La propuesta presentada aquí se basa no en intentar reproducir de manera artificial las circunstancias del desarrollo de software profesional sino en ubicar al alumno desde el principio de la asignatura ante una simulación del mercado real, competitivo, en el que las empresas de desarrollo de software son evaluadas por entidades externas para determinar su idoneidad, desde el punto de vista de la implementación de procesos de ingeniería del software, para participar en contratos de desarrollo de aplicaciones. Desde el primer momento se pide a

los alumnos que se organicen en equipos[1] - o son organizados por el profesor - y se les plantea un contexto amplio en el que más adelante se ha de desarrollar la parte práctica de la asignatura.

El esquema general que se propone correspondería a que, a lo largo del curso, el profesor ejerza el papel de entidad externa evaluadora de los niveles de implementación de procesos de ingeniería del software en cada equipo, y los equipos ejercen el papel de empresas que aspiran a ser consideradas aptas para el desarrollo de proyectos. Dentro de este marco puede tomarse como punto de partida una exposición de los criterios y procedimientos de evaluación de empresas clásicos dentro del campo de la ingeniería del software, como el Software Capability Maturity Model (SW-CMM) (ver [2], o [7, 11] del SEI [10] para descripciones más reducidas) o una adaptación que se considere apropiada. Esta elección tiene la ventaja de presentar a los alumnos con una referencia externa real, independiente del profesor. También puede servir para describir sobre datos concretos temas cruciales como la crisis del software, y la motivación pragmática de empresas grandes de desarrollo de acogerse a un programa de evaluación [8].

A partir de ahí se plantea a los equipos de alumnos como objetivo el desarrollar un modelo de proceso para el equipo que aspire a implementar la mayor cantidad posible de los procesos que se han explicado como deseables. El primer trabajo práctico que se ha de encomendar a cada equipo es el de elaborar una definición por escrito del proceso de desarrollo que se piensa seguir, detallando aquellas áreas clave del proceso (según la definición del SW-CMM) que se aspira a implementar primeramente.

A la hora de proporcionar una motivación concreta para llevar a cabo la tarea, se propone desarrollar toda la parte práctica que corresponda a la asignatura a modo de concursos públicos (ficticios) para optar a hipotéticos contratos para desarrollar aplicaciones concretas que vengan determinadas por pliegos de condiciones del concurso redactados por el profesor. Cada equipo debe presentar documentación correspondiente al proyecto, incluyendo una especificación completa y un diseño de la aplicación a desarrollar, así como una estimación de costes, una planificación temporal, y los demás artefactos que se considere adecuado solicitar en las distintas etapas del curso. De este modo, el profesor puede ir introduciendo progresivamente a lo largo del curso distintos conceptos teóricos que se vayan incorporando a los pliegos de condiciones y que por tanto los alumnos tienen que incorporar a sus equipos. A cada paso, los criterios en que se debe basar el profesor para elegir el equipo ganador del concurso se han de corresponder con los establecidos como deseables para considerar que una empresa implementa razonablemente los procesos deseados de ingeniería del software. El triunfo en el concurso puede utilizarse como mecanismo de evaluación. Con objeto de que puedan desarrollarse un número significativo de estos concursos, es posible no exigir que cada proyecto se lleve a cabo hasta obtener una aplicación en marcha. Las especificaciones y diseños que se obtenga por este método podrían resultar interesantes como subproductos de la experiencia, con vistas a utilizarlos como material docente en otras asignaturas del plan de estudios más orientadas a la programación pura y simple.

3. Discusión de la propuesta

Una propuesta de este tipo debe por supuesto compaginarse con sesiones teóricas en las que se impartan los materiales necesarios para que los equipos puedan acometer con éxito las tareas que se les encomiendan. En ese sentido, la propuesta no supone una reestructuración radical del modo de trabajo vigente en la actualidad en la gran mayoría de las universidades. Simplemente constituye un marco general en el que posiblemente los alumnos tengan en su poder las herramientas necesarias para entender el por qué de cada nuevo concepto *antes* de que les sea explicado, y que desde el primer momento tengan presente un objetivo último muy concreto hacia el que se dirigen las sucesivas ampliaciones de las metodologías de trabajo que se les van proponiendo.

[1] Las dimensiones de los equipos deberán establecerse de manera que permitan una estructuración más o menos compleja, con reparto de funciones y subgrupos dentro del equipo, pero sin que el tamaño excesivo reste agilidad de desarrollo por sobrecarga en la comunicación interna [3].

3.1. Puntos fuertes

De esta manera, ese marco general puede servir para dar cohesión a una serie de materias elementales difíciles de presentar como una teoría unificada cuando se intentan presentar secuencialmente a modo de temas sucesivos.
Por otro lado, el trabajo práctico se está centrando desde el primer momento en la parte del desarrollo de software (el entorno empresarial, y el programador no como individuo enfrentado a una tarea específica y localizada, sino como engranaje dentro de una maquinaria más amplia que aspira a funcionar como una factoría de software) más alejada de la experiencia anterior de los alumnos. De este modo, cada alumno toma contacto inmediatamente con los contenidos fundamentales de la asignatura. Al mismo tiempo puede obtenerse un efecto beneficioso adicional si se consigue impedir de este modo que el alumno emprenda los trabajos prácticos de ingeniería del software sin arrastrar consigo la inercia de métodos propios de trabajo adoptados durante su trabajo para asignaturas más elementales de programación.

3.2. Puntos débiles

No debe perderse de vista en este contexto que el volumen de materia a impartir en cualquier temario de ingeniería del software es grande, y que un planteamiento como el que se propone supone un esfuerzo grande por parte del profesor para reestructurar los contenidos de modo que a cada paso se esté ofreciendo a los alumnos material suficiente para realizar las prácticas que se les exigen de modo simultáneo.
Otra debilidad de esta propuesta puede radicar en el hecho de que al organizarse en equipos, puede ocurrir que cada alumno se enfrente en la práctica solamente a una faceta específica de los procesos de ingeniería del software que se estén implementando. Sin embargo todos los alumnos tendrán la vivencia de las distintas labores que está llevando a cabo el equipo, puesto que, independientemente de si ellos mismos han sido nombrados responsables de planificación, de gestión de configuración... todos ellos se ven planificados por alguien y tienen que trabajar dentro de un sistema de gestión de configuración.

3.3. Posibles problemas

Al principio de la experiencia los alumnos no disponen de los conocimientos teóricos que van a necesitar. Es el esfuerzo de los alumnos a partir de las condiciones explícitas en el concurso, trabajando sobre los materiales proporcionados, y coordinado por el profesor, el que da lugar a que los alumnos, al terminar la experiencia, se hayan familiarizado con la materia teórica.

4. Comparación con propuestas existentes

Una parte importante de las propuestas existentes en la actualidad para solventar el problema de motivar el aprendizaje de la ingeniería del software se centran en los procesos de análisis, diseño, implementación y prueba para el desarrollo de un proyecto, de maneras más o menos iterativas e incrementales, apoyándose en distintas metodologías de trabajo, con especial popularidad de las versiones orientadas a objetos [1, 4]. Este enfoque indudablemente engloba una parte importante de la materia a cubrir, proporciona al alumno un número importante de herramientas que más adelante va a necesitar en su labor profesional, y le lleva bastante más allá del punto en que se haya podido quedar al terminar las asignaturas elementales de programación. Sin embargo presenta la desventaja de que deja sin contrapartida práctica en la experiencia del alumno aquellos aspectos que incluso en esas mismas metodologías se declaran fundamentales, como son la necesidad de adaptar cualquier proceso de desarrollo que se pretenda adoptar a la circunstancia concreta de un equipo de trabajo, de una empresa de desarrollo, o incluso de un individuo. Del mismo modo, los textos seminales de estas metodologías insisten en la necesidad de cubrir de modo adicional temas que, por su naturaleza eminentemente práctica y dependiente de los recursos, plataformas, y aplicaciones, no pueden tratarse de modo universal en un texto, como pueden ser la gestión de configuraciones del software. Esto lleva a que en muchos casos temas como este se pasen por alto, o como mucho se describan desde un punto de vista teórico solamente. El método propuesto, que parte de un conjunto lo más amplio posible de procesos a implementar, e intenta de manera

práctica incorporar la mayor cantidad de ellos en los mecanismos de trabajo de cada equipo, utilizando para ello recursos [5, 6] y fuentes de documentación [2] disponibles públicamente, sin duda tampoco va a conseguir que los alumnos acaben la asignatura habiendo experimentado el espectro completo de las áreas clave del proceso definidas en el SW-CMM. El espectro es demasiado amplio para que siquiera grandes empresas con infinidad de recursos pudieran abarcarlo en tan corto periodo de tiempo. Sin embargo se garantizaría que cada alumno haya vivido al menos la angustia del directivo que aspira a conseguirlos todos, que tiene que barajar los recursos disponibles para establecer un plan viable a corto plazo que le proporcione la mayor cantidad posible de ventajas, pero que no pierda de vista el objetivo final hacia el que camina su empresa. Semejante oportunidad no debe perderse, especialmente porque las condiciones de implantación de estos procedimientos son mínimas en la mayoría de las empresas que el alumno va encontrar en el mercado de trabajo que le es inmediatamente accesible al terminar la carrera. Esto quiere decir que el alumno actual en España tendrá la oportunidad de vivir esta experiencia o durante sus años universitarios a modo de visión de futuro, o en el futuro muy lejano si llega a verse como directivo de una empresa suficientemente grande y suficientemente lúcida para plantearse implementar estos procedimientos.

5. Conclusión

La pregunta que sirve de título a esta ponencia encierra una disyuntiva problemática de resolver. Está claro que la mayoría de los temarios, el consenso de la comunidad de ingenieros del software [9], e incluso los defensores de metodologías concretas [4] consideran que la experiencia de intentar implantar procesos de desarrollo a largo plazo es más deseable que el limitarse a desarrollar proyectos en equipos coordinados exclusivamente para la duración del proyecto. Las consideraciones que determinan que esto segundo sea más frecuente que lo primero como parte práctica de asignaturas de ingeniería del software están relacionadas más bien con restricciones temporales y de viabilidad que con objetivos docentes. La propuesta que se plantea aquí, precisamente porque no supone un aumento radical en el volumen de trabajo a realizar por los alumnos, sino simplemente una reorientación desde el principio, procurando que la parte del temario que más poso debiera dejar en el alumno se plantee al comienzo de la asignatura, puede proporcionar una posibilidad de reorientar el trabajo práctico para conseguir el objetivo más prioritario con un coste en esfuerzo y trabajo dedicado por los alumnos más o menos equivalente.

Esta propuesta es de reciente implantación y todavía es pronto para poder comentar sobre los resultados que se puedan obtener a largo o medio plazo en términos de formación obtenida por los alumnos. No obstante, si que se ha observado desde que se planteó por primera vez una mejora pronunciada en la motivación que muestran los alumnos y su disponibilidad para enfrentarse de motu propio al material escrito sobre los temas que se plantean.

Referencias

[1] G. Booch. *Análisis y diseño orientado a objetos con aplicaciones.* Segunda edición. Addison-Wesley/Díaz de Santos, 1996

[2] Carnegie Mellon University, Software Engineering Institute (Mark C. Paulk, Charles V. Weber, Bill Curtis, and Mary Beth Chrissis). *The Capability Maturity Model: Guidelines for Improving the Software Process.* Addison-Wesley Publishing Company, Reading, MA, 1995. También disponible en linea en http://www.sei.cmu.edu/cmm/obtain.cmm.html

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

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

[5] P.Miller, Aegis 3.23, transaction-based software configuration management system, http://www.canb.auug.org.au/~millerp/aegis/aegis.html

[6] P.Miller, fhist 1.7, file history tool ``fhist", a file comparison tool ``fcomp", and a file merging tool ``fmerge" http://www.canb.auug.org.au/~millerp/fhist.html

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

[8] SEMA - Maturity Profile. *Process Maturity Profiles of the Software Community*. SEI, Carnegie Mellon University, http://www.sei.cmu.edu/sema/pdf/2000aug.pdf

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

[10] Software Engineering Institute, Carnegie-Mellon University, http://www.sei.cmu.edu/sei-home.html

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