Actas XVIII JENUI 2012, Ciudad Real, 10-13 de julio 2012
I.S.B.N. 10: 84-615-7157-6 | I.S.B.N. 13:978-84-615-7157-4
Páginas 413-416

El Método "Software Factory": Acciones para realizar prácticas más realistas, usando herramientas Web de trabajo colaborativo, y trabajo a distancia.

Alejandro Bia
Centro de Investigación Operativa
Universidad Miguel Hernández
Av. Universidad s/n 03202 Elche (Alicante)
abia@umh.es

Ramón P. Ñeco
Departamento de Ingeniería de Sistemas y Automática
Universidad Miguel Hernández
Av. Universidad s/n 03202 Elche (Alicante)
ramon.neco@umh.es

Resumen

El método Software Factory (SF) consiste en llevar a cabo acciones y aplicar herramientas informáticas, con dos finalidades: (1) hacer que las prácticas de asignaturas tecnológicas (en un sentido amplio) sean lo más parecidas que sea posible al trabajo real en la empresa o la industria y (2) difundir el uso de herramientas Web que permitan nuevas formas de trabajo ágiles y dinámicas, como por ejemplo el teletrabajo, el trabajo móvil y el trabajo colaborativo a distancia. Este artículo describe las experiencias de los autores en este sentido.

Summary

The SF method is meant to take some actions and to apply software tools, with two purposes: (1) to make the practical work of technology subjects (in a broad sense) to resemble as much as possible the actual work in business or industry, and (2) to spread the use of Web tools that allow new flexible and dynamic forms of work, such as teleworking, mobile working and remote collaborative work. This article describes the experiences of the authors in this regard.

Palabras clave

Software Factory, prácticas realistas, teletrabajo, trabajo colaborativo, gestión de proyectos

1. Antecedentes

Existen algunos trabajos previos en los que se describen experiencias con métodos relacionados con el descrito en el presente artículo. Así, por ejemplo, en [1] los alumnos simulan estar trabajando en una empresa informática, formando equipos de trabajo cada uno de los cuales debe realizar proyectos reales distintos. También existen numerosos trabajos relacionados con la docencia a partir de grupos. En [6] se analiza, a partir de experiencias prácticas de los autores, los beneficios del trabajo en grupo de los estudiantes desde el punto de vista de las relaciones e interdependencias que se establecen en los mismos. En [2] se describen las competencias que aprenden los estudiantes mediante la realización de proyectos reales en grupo. Más recientemente, Jones y Issroff [5] realizan un estudio de las habilidades sociales y de actitud obtenidas con el trabajo en grupo, de forma parecida a como se utiliza en el método propuesto en el presente artículo.

2. Motivación, objetivos y herramientas

El objetivo general del método propuesto consiste en simular en las clases prácticas, con el mayor realismo posible, las condiciones de trabajo en una empresa real, preparando así mejor a los alumnos para la inserción en la vida laboral. Como objetivos adicionales, se plantea la integración de habilidades y conocimientos de varias asignaturas y el fomento del trabajo en equipo. Hemos denominado a este método *Software Factory*, en honor a aquellas iniciativas que surgieron en los ’70 y ’80 para sistematizar la producción de software, como si se tratase de una línea de montaje industrial [3].

Si bien se ha creado pensando en las clases prácticas de las asignaturas Ingeniería de Software y Gestión de Proyectos de Ingeniería de Software, esta metodología es aplicable a la enseñanza práctica de otras asignaturas. Así, por ejemplo, en el curso académico 2011-2012 también se ha aplicado a una asignatura de Automatización Industrial de la

titulación de Ingeniería Mecánica para la que hemos comprobado que la metodología SF puede aplicarse con éxito.

2.1. Herramientas

En función de la asignatura específica en la que se use el método propuesto, se usarán distintas herramientas software y/o hardware, tales como:

- Herramientas de trabajo colaborativo en red
  - Digital Humanities Workbench: herramientas en-línea para procesamiento XML: <equis.umh.es/dhw/>
  - Gogle Apps (GoUMH[1]): correo/chat, calendario, grupos, documentos de ofimática, sitios Web: <http://www.goumh.es/>
  - Dropbox: compartir ficheros mediante sincronización de carpetas: <http://www.dropbox.com/>
  - Collabtive: entorno de trabajo colaborativo: <http://collabtive.o-dyn.de/>
  - Subversion: control de versiones de software: <http://subversion.tigris.org/>
  - Moodle: entorno colaborativo educativo: <http://moodle.org/>
- Herramientas de gestión de proyectos: Gantt-Project (<http://www.ganttproject.biz/>), dotProject (<http://www.dotproject.net/>), web2Project (<http://web2project.net/>), Grindstone (<http://www.epiforge.com/Grindstone/>), project-Open (<http://www.project-open.org/>).

El método propuesto en este artículo es aplicable a la enseñanza práctica en asignaturas principalmente tecnológicas o de tipo empresarial, donde se puedan simular casos reales de trabajo o proyectos de la industria o la empresa. Así, durante el presente curso académico se ha aplicado a las siguientes titulaciones y asignaturas: (1) Grado en Ingeniería Informática en Tecnologías e la Información, asignaturas “Ingeniería de Software”, “Gestión de Proyectos de Ingeniería de Software”, “Seguridad en Sistemas Informáticos”; (2) Ingeniería Técnica en Informática de Gestión, asignatura “Proyectos de Aplicaciones de Gestión”; (3) Ingeniero Técnico Industrial, Especialidad En Mecánica, asignatura “Automatización Industrial”.

[1] GoUMH es la versión de Google-Apps adaptada para la Universidad Miguel Hernández: https://sites.google.com/a/goumh.umh.es/goumh-going-google/

3. Descripción del método propuesto

El método tradicional de la enseñanza de resolución de problemas de informática en clase, consiste en el planteamiento de un enunciado donde se describe la situación de una empresa ficticia. Mediante este enunciado se presentan los requisitos del problema. En las empresas reales a los informáticos no se les da un enunciado narrado, sino que deben recopilar los datos del problema a resolver por su propia cuenta, con mucho ingenio y esfuerzo. A esta tarea de la ingeniería de software se le llama *captura de requisitos*, la cual es una fase crucial en el proceso de diseño de software (de hecho, muchas veces se habla de ingeniería de requisitos, hasta el punto de que hay muchos libros con ese título). En el método propuesto en este artículo se propone *no dar enunciados de problemas*, sino que el profesor haga las veces de empresario o cliente y plantee “sus necesidades” o requisitos de forma verbal e incompleta, como sucedería en una empresa. Luego es tarea de los equipos de desarrollo (grupos de alumnos) el aplicar las técnicas de captura de requisitos para verificar, refinar y completar la lista de requerimientos del problema a resolver, y elaborar con ella una especificación de la solución.

En las secciones siguientes se describen las características del método propuesto.

3.1. Trabajo en equipo y asignación de roles

En proyectos medianos y grandes de desarrollo de software existe la necesidad de trabajar en equipo (a diferencia del desarrollo de pequeños programas). Al trabajar en equipo surgen problemas de comunicación, entendimiento, coordinación, colaboración, y relación interpersonal. Por ello en el método software factory se fomenta el trabajo en equipo, donde todos estos problemas de comunicación e interacción personal se potencian. Además, en las empresas reales, los informáticos tienen que trabajar en diferentes proyectos con diferentes compañeros de equipo. Por esto motivo

en el método propuesto también se fomenta la realización de prácticas distintas con diferentes compañeros de equipo.

El estudio de las razones por las que algunos grupos funcionan muy bien y otros no tanto, es un problema muy interesante desde el punto de vista de la ingeniería de software. En este sentido, como parte del método propuesto, y después de la entrega del proyecto, se propone que los miembros de cada grupo hagan una lista ordenada según los méritos percibidos de sus compañeros de equipo. El profesor dispone así de una lista de la percepción grupal subjetiva del esfuerzo y aportes que ha hecho cada miembro del grupo para el éxito del proyecto, que se mantiene en secreto. Después de realizar esta evaluación del grupo, se puede hacer una discusión amigable de lo que ha fallado, de lo que se ha hecho bien, y de cómo se podría hacer mejor.

En relación con el trabajo en equipo, en el método software factory se asignan *roles* a los miembros del grupo, similares a los que existen en la industria informática. Es común que cada miembro del grupo tenga más de un rol, y asuma diferentes responsabilidades. Algunos roles usados en asignaturas de Ingeniería del Software son: un coordinador de proyecto (*project leader*), analistas de sistemas, diseñadores, programadores, encargados de base de datos (*DBManager*), encargado del servidor (*WebMaster*), o de la plataforma de desarrollo (*System Administrator*).

3.2. Hardware y software utilizado

En las universidades, es habitual que los laboratorios de informática dispogan de ordenadores que son configurados y mantenidos por un técnico de laboratorio. Además, normalmente se elimina cualquier dato o programa instalado por los alumnos cuando se reinician los equipos.

Sin embargo, en las empresas reales, no hay aulas informáticas o laboratorios. Los informáticos encargados de un proyecto deben desarrollarlo partiendo de un ordenador vacío, que ellos mismos tienen que montar, conectar e instalar los programas necesarios. En el métdodo Software Factory se hace algo parecido. Para ello, se asigna un ordenador a cada grupo de prácticas (equipo de proyecto), y el grupo debe hacerse responsable de configurar e instalar todo lo que necesiten para llevar a cabo su proyecto. El técnico de laboratorio de la Universidad sólo les provee de software necesario, cuando el software esta bajo licencia. Cuando necesiten software libre, cuyo uso se fomenta en la metodología Software Factory, lo deberán descargar de Internet e instalarlo ellos mismos. El mantenimiento de este ordenador será responsabilidad de quien cumpla el rol de WebMaster o System Administrator. Es importante destacar que, si bien las asignaturas en las que se ha aplicado la metodología propuesta tratan sobre el análisis y diseño de alto nivel (por medio de diagramas de diseño), se intenta que los alumnos construyan los programas en concreto, con la finalidad de dar una visión integrada de lo que es un proyecto completo de desarrollo de software.

3.3. Teletrabajo

En la metodología software factory se fomenta el trabajo a distancia, o teletrabajo. Esta es una forma muy interesante de trabajar, que brinda muchas ventajas a los trabajadores y a las empresas, al permitir a los trabajadores realizar sus tareas desde donde estén, sin necesidad de desplazarse a un centro de trabajo. En la metodología software factory hacemos uso del teletrabajo para que los grupos de desarrollo puedan mantenerse en contacto y compartir ficheros cuando no coinciden en clase.

3.4. Entrega y defensa finales. Inspecciones periódicas

La entrega final del proyecto se realiza en dos fases. Si la primera entrega presenta fallos menores, carencias o está incompleta, se le da al grupo la posibilidad de reparar esos fallos y entregar de nuevo el proyecto en unos días. La última defensa del proyecto se hace la penúltima semana del curso, dando la posibilidad de reentregar el proyecto corregido la última semana. En esta clase post entrega y defensa, se discute lo que se ha hecho bien y lo que se ha hecho mal, promoviendo el debate y las comparaciones entre los diferentes grupos de proyecto.

Es importante destacar que, antes de la entrega final del proyecto, se realizan inspecciones periódicas del mismo, mediante las que el profesor lleva un seguimiento semanal de la evolución de

cada grupo de prácticas (equipo de desarrollo). Mediante estas inspecciones se persigue que los alumnos aprendan a desarrollar los proyectos de forma gradual y continua, siendo el método de trabajo idóneo en un proyecto real. En el entorno docente, estas inspecciones periódicas son además una buena técnica de evaluación continua.

4. Resultados y conclusiones

Para la evaluación de los resultados obtenidos con el uso del método propuesto se han planteado las siguientes técnicas: (1) Realización de encuestas de opinión a los alumnos; (2) Uso de métricas de variables de cada proyecto de prácticas; (3) Realización de informes y toma de notas durante las inspecciones periódicas; (4) Debates alumnos-profesor sobre los logros, fallos y resultados de los proyectos a su finalización; (5) Debate y juicio crítico entre los profesores involucrados. Las principales conclusiones obtenidas tras la aplicación del método descrito pueden resumirse como sigue:

- Respecto a evitar el uso de enunciados explícitos y detallados, algunas veces, hemos encontrado cierta resistencia y quejas por parte de los alumnos. Sin embargo, si se les explican las razones por las que se hace esto, las entienden, y al poco tiempo se les ve inmersos en la captura de requisitos.
- El uso de nuevas tecnologías y la experimentación con software libre y colaborativo motiva a los alumnos. Además, la disponibilidad de software portable les permite trabajar en similares condiciones desde cualquier lugar.
- La discusión post-entrega resultó ser una herramienta muy útil que permite reafirmar los conocimientos prácticos y habilidades adquiridas, así como repasar los errores y discutir las distintas decisiones de diseño o el uso de distintas plataformas entre los grupos.
- Algunas veces se han detectado problemas de cooperación en los grupos de prácticas. La técnica de votación secreta descrita en el artículo ha resultado ser muy útil para poner de manifiesto la “conciencia de grupo”.
- El trabajo en grupos grandes, de 4 o más alumnos, y el uso de un ordenador dedicado a cada grupo, han facilitado el reparto de tareas, de roles y de responsabilidades.

Algunas líneas de trabajo futuro incluyen: la extensión y consolidación del método en asignaturas de otras ingenierías y la definición de una medida de esfuerzo de adaptación de los estudiantes a las empresas a partir de la cual los docentes podrían mostrar la línea base antes de aplicar su metodología, y compararla con los resultados obtenidos después de su aplicación.

Agradecimientos

Este trabajo ha sido parcialmente financiado por la Universidad Miguel Hernández: proyecto PN1238-ILZU02 de la Convocatoria 2011 para Proyectos de Investigación en Humanidades y Ciencias Sociales.

Referencias

[1] Bermejo, M., Fernández, R. *Alumno Rupérez, está usted despedido*, XI Jornadas de Enseñanza Universitaria de la Informática, JENUI, Madrid, 2005.

[2] Colbeck, C. L., Campbell, S. E. and Bjorklund, S. A., *Grouping in the dark: what college students learn from group projects*, The Journal of Higher Education, 71(1), p: 60-83 (2000)

[3] Cusumano, M.A., *The software factory: a historical interpretation*, IEEE Software, 6(2), pgs. 23-30, ISSN: 0740-7459, DOI: 10.1109/MS.1989.1430446, IEEE (1989)

[4] Dougiamas, M., *Moodle: Module Object-Oriented Dynamic Learning Environment*. URL: http://www.moodle.org

[5] Jones, A. and Issroff, K., *Learning technologies: affective and social issues in computer-supported collaborative learning*. Computers and Education, 44(4), p: 395-408 (2005)

[6] Nam, W.C., and Zellner, R.D. *The relative effects of positive interdependence and group processing on student achievement and attitude in online cooperative learning*, Computers and Education, 56(3), April 2011, p: 680-688 (2011)