View metadata, citation and similar papers at core.ac.uk

provided by UPCommons. P

Uso de herramientas para la adquisición de competencias transversales asociadas al desarrollo y mantenimiento de software

Carlos López, Raúl Marticorena, Pablo Santos y Jesús M. Maudes
Área de Lenguajes y Sistemas Informáticos
Universidad de Burgos
Escuela Politécnica Superior Edf. C
09006 Burgos, España
{clopezno, rmartico, psluaces, jmaudes}@ubu.es

Resumen

Las herramientas de control de versiones y planificación de tareas permiten monitorizar y recoger información sobre el proceso de desarrollo software. En este trabajo se presenta la utilización de este tipo de herramientas en asignaturas relacionadas con este campo. La idea no es tanto utilizar las herramientas como contenido *per se* de las asignaturas, sino servirse de ellas para fomentar y evaluar la adquisición de ciertas competencias transversales relacionadas con esas asignaturas. El presente trabajo selecciona esas competencias, estudia las posibles herramientas, su ámbito de utilización y conveniencia, y finaliza con el análisis del impacto de las herramientas en las competencias transversales seleccionadas a través de encuestas a los alumnos.

1. Introducción

Existen muchas asignaturas de informática que en sus actividades prácticas piden como producto a entregar por el alumno un determinado *código fuente* como resultado de la resolución de un problema.

El código fuente tiene una naturaleza cambiante a lo largo de su vida debido fundamentalmente a dos cuestiones: por un lado, la incorporación de nuevos requisitos, por otro, el asociado al mantenimiento correctivo ocasionado por errores y defectos no controlados. Para controlar y monitorizar los efectos de estos cambios existen herramientas ampliamente utilizadas en la industria, como son las herramientas de control de versiones y planificación de tareas.

En un contexto docente, la naturaleza cambiante puede venir determinada por el profesor, al incluir nuevos requisitos, y por el alumno que va transformando el código hasta alcanzar la solución final. Esta sucesión de cambios se hace más palpable si, como es habitual, la tarea de codificación tiene una vida superior a una sesión de prácticas.

Cuando un profesor tiene que evaluar un código, generalmente sólo evalúa el producto final, es decir comprueba propiedades cualitativas de la última versión [13]. En la evaluación no se considera la información asociada al proceso de desarrollo: tiempo empleado en el desarrollo, responsabilidades de los miembros de equipo, versiones de los diferentes estados evolutivos del código, etc.

Sin embargo, toda esta información: (i) suele ser relevante desde el punto de vista de la adquisición y seguimiento de ciertas competencias transversales en las propias asignaturas en las que se desarrolla el código evaluado, y (ii) puede estar disponible mediante el uso de herramientas de control de versiones y planificación de tareas.

El tipo de competencias transversales que se pretende potenciar y monitorizar con estas herramientas incluyen, por ejemplo, el trabajo en equipo, la planificación, y la capacidad de organización.

Por tanto, parece razonable que la aplicabilidad de las herramientas con este fin tome más relevancia en contextos de asignaturas donde la vida del código sea más larga; entendiendo por vida de código desde que se propone un enunciado hasta que se entrega (e.g. Trabajos Final de Carrera, Trabajo Fin de Grado o Trabajo Fin de Máster).

Sin embargo, parece interesante que los alumnos puedan adquirir las competencias mencionadas con anterioridad, a fin de que puedan aplicarlas con mayor naturalidad cuando lleguen a dichas asignaturas de final de carrera,

por lo que también parece aconsejable que al menos se introduzca la utilización de estas herramientas en asignaturas más básicas, aun cuando la vida del código en esas asignaturas sea menor.

Esa naturalidad en la utilización de las herramientas vendrá dada en gran medida por el grado de conocimiento de las mismas. Por tanto, es conveniente elegir las herramientas de forma que su coste de aprendizaje no suponga hacer un doble esfuerzo, tanto a alumnos, como a profesores.

En lo que sigue el artículo se estructura de la manera siguiente. En la sección 2 se seleccionan las competencias transversales a tratar, el proceso seguido para su posible monitorización y un estudio sobre trabajos similares. En la sección 3, se presenta un marco tecnológico de herramientas software que permita monitorizar el proceso. En la sección 4, se presenta el escenario concreto de aprendizaje donde se realiza la experiencia y el plan llevado a cabo. En la sección 5, se analiza una evaluación por parte de los alumnos sobre la obtención de competencias transversales mediante la herramienta utilizada. Por último, en la sección 6, se enumeran unas conclusiones y líneas futuras de actuación.

2. Selección de competencias transversales y su monitorización

El objetivo general es monitorizar el proceso de desarrollo y mantenimiento de código para comprobar la adquisición de ciertas competencias transversales.

El tipo de herramientas utilizadas es pues, el que determina qué competencias transversales son susceptibles de ser tratadas. Las competencias transversales elegidas son mostradas en la Tabla 1 y toman como referencia las descritas en el Real Decreto 1393/2007 [15] y en el Libro Blanco de Ingeniería Informática (LB). Además en la tabla se ha añadido una columna para identificar la competencia (Id).

Para conseguir el objetivo marcado el trabajo ha constado de las siguientes etapas:

1. Estudio y selección de herramientas que permitan monitorizar el proceso de desarrollo y mantenimiento de código y su relación con competencias transversales.
2. Aprendizaje y uso de herramientas por parte de profesores como alumnos en varias asignaturas.
3. Evaluación del uso de las herramientas y su relación con competencias transversales por parte de los alumnos

Descripción	**Id**
Capacidad de organización y planificación	C1
Conocimientos de informática relativos al ámbito de estudio	C2
Capacidad de gestión de la información	C3
Trabajo en equipo	C4
Planificación y gestión del tiempo	C5

Tabla 1. Competencias transversales

En la medida de nuestro conocimiento, en otros trabajos como [20], se expone una experiencia práctica de aplicación de herramientas de gestión de configuraciones, pero sin abordar un estudio como el realizado en el presente trabajo. Otras experiencias similares en cursos de ingeniería del software pueden ser consultadas en [2] y [9], enfocadas a la mejora de competencia de trabajo en grupo simulando el funcionamiento real de un proyecto. En esa misma línea en [19] se plantea el uso de estas herramientas para el seguimiento de los alumnos en las asignaturas. En [14] también se apunta la importancia en el aprendizaje basado en proyectos del uso del control de versiones y herramientas de cooperación en web. La importancia de la utilización de este tipo de soluciones es evidente con la inclusión de talleres o *workshops* para docentes [4]. En nuestro trabajo no sólo se pretende incidir en el uso de este tipo de herramientas, sino en abordar la adquisición de competencias transversales a lo largo de su aplicación en diferentes materias y asignaturas.

3. Herramientas de monitorización para el desarrollo de código

Desde un punto de vista docente, el desarrollo de código puede verse como la realización de una determinada tarea por parte de los alumnos que debe entregarse en una fecha determinada. Como consecuencia, el alumno entrega un resultado

compuesto generalmente de múltiples documentos electrónicos.

Con un enfoque generalista, una de las primeras aproximaciones que se hizo fue utilizar la funcionalidad de monitorización de los entornos de aprendizaje virtual, como Moodle [16]. La funcionalidad de gestión de grupos permite conocer la intervención de cada alumno en la tarea, y en consecuencia la intervención de cada uno en el trabajo. Pero en el contexto planteado, muchas tareas pueden referirse al mismo fichero resultado. Este modelo de trabajo no está soportado de manera natural en las herramientas de enfoque generalista.

En un ámbito más orientado al mundo profesional, la monitorización de la actividad de desarrollo de código se realiza utilizando herramientas de control de versiones y herramientas de gestión de tareas. En este sentido, parece interesante incorporar los conocimientos de este tipo de herramientas en los planes de estudio de manera transversal para que puedan ser utilizadas en diferentes asignaturas, en particular en las asignaturas de trabajos fin de grado.

El uso de ambas herramientas puede hacerse de manera conjunta, de manera independiente o incluso se pueden combinar con los gestores de contenidos educativos. Las exigencias de conocimientos y de aprendizaje de las distintas herramientas varían notablemente.

En el caso de gestores de tareas, se puede introducir desde los primeros cursos de grado, y el tiempo dedicado para que los alumnos puedan utilizarlos de manera básica puede estar comprendido entre una y dos horas de prácticas. Los gestores de tareas proporcionan una visión que puede ayudar a abordar la adquisición competencias de capacidad de organización, planificación y gestión del tiempo. Aunque en este trabajo no se pretende evaluar un conjunto de herramientas de este tipo, cabe destacar que como criterio de selección se ha considerado el acceso a la herramienta a través de Internet y que se integre con otras funcionalidades de desarrollo de código. Así, la herramienta elegida es el portal XP_DEV con su planificador para el seguimiento de proyectos [22]. El portal está orientado a soportar la metodología de desarrollo de proyectos e**X**treme **P**rogramming, [11] donde las diferentes tareas del proyecto se agrupan por historias y éstas a su vez en iteraciones.

El uso de los sistemas de control de versiones requiere de unos conocimientos más avanzados por parte de los alumnos. Además, su aprendizaje y uso puede requerir varios estados de madurez adquiridos en diferentes cursos. Este tipo de sistemas pueden ayudar a monitorizar la adquisición de competencias como la capacidad de organización y de gestión de la información. La elección del sistema depende de las siguientes restricciones:

- Utilizar la misma herramienta en todas las asignaturas implicadas.
- Uso del sistema mediante aplicaciones de red cliente/servidor.
- Disponer de herramientas flexibles que permitan consultar la utilización de los repositorios para configurar la monitorización.

Bajo estas premisas se eligió un sistema comercial llamado Plastic SCM [5]. La empresa desarrolladora del sistema, Códice Software, ha proporcionado gratuitamente una licencia de servidor y soporte de mantenimiento. Respecto a la funcionalidad frente a otros sistemas de código abierto, como Subversión [17], se destaca la fácil gestión de ramas, integración con sistemas de gestión tareas, personalización de de consultas sobre el uso de repositorios y visualización gráfica de monitorizaciones.

4. Especificación de las experiencias

En este apartado se especifica el conjunto de experiencias aplicado a contextos concretos de asignaturas. En la Tabla 2 se da una visión global de las asignaturas implicadas, y de las tareas que supone para los profesores y alumnos la incorporación de dichas herramientas. La experiencia ha sido evaluada en dos titulaciones Ingeniería Técnica de Informática de Gestión (ITIG) e Ingeniería Informática (II). La estrategia seguida en ITIG ha sido utilizar el gestor de tareas del portal XP_DEV en una única asignatura de 3er curso de la titulación. En el caso de la titulación de II, la estrategia se ha basado en utilizar un gestor de control de versiones y su utilización se ha distribuido en varias asignaturas.

En las siguientes secciones se describe con más detalle el contexto de enseñanza para facilitar su comprensión y su posible reproducción en otros ámbitos similares.

Programación Avanzada, 3º ITIG	
Tareas del profesor	Tareas del alumno
TPA1 Presentación del portal XP-DEV (1 hora). **TPA2** Registros de usuario y grupos (1 hora).	**TPA_A1** Desarrollo de un mini proyecto de programación desde cero. (6 horas en grupos de 4 alumnos).
Planificación y Gestión de Proyectos, 4º II	
Tareas del profesor	Tareas del alumno
TPGP1 Instalación de servidores, uno por cada grupo de prácticas (2 horas). **TPGP2** Preparar los guiones de la prácticas con casos de estudio (4 horas).	**TPGP_A1** Operaciones básicas de sistema de SCM (2 horas en grupos de 2 alumnos). **TPGP_A2** Técnica de desarrollo "*main line development*" (2 horas en grupos de 2 alumnos). **TPGP_A3** Técnicas de desarrollo basadas en *branching*. (4 horas en grupos de 2 alumnos).
Diseño y Mantenimiento del Software II, 5º II	
Tareas del profesor	Tareas del alumno
TPDM1 Organización de las prácticas guiadas en el repositorio central (1 hora). **TPDM2** Desarrollo de dos prácticas guiadas de mantenimiento de código mediante refactorizaciones (4 horas).	**TPDM_A1** Desarrollo de prácticas guiadas en paralelo con el profesor (4 horas en grupos de 2 personas). **TPDM_A2** Mantenimiento de un código para mejorar su calidad mediante inspecciones y reestructuración de diseño. (6 horas en grupos de 4 alumnos).
Sistemas Informáticos, 5º II	
Tareas del profesor	Tareas del alumno
TSI1 Seminario de Plastic SCM impartido por la empresa Códice Software (4 horas). **TSI2** Creación de un repositorio por cada petición de proyecto.	**TSI_A1** Desarrollo del trabajo fin de carrera utilizando el repositorio para gestionar las versiones de sus productos software.

Tabla 2. Visión global de la experiencia

4.1. Programación Avanzada

En esta asignatura los alumnos desarrollan un pequeño proyecto de programación desde cero en grupos de cuatro personas. El objetivo es aplicar conceptos de patrones de diseño [8] y programación web con bibliotecas de Java. El caso de estudio elegido se fundamenta en [12], donde se da una visión del aprendizaje de patrones de diseño en un plan de estudios.

La práctica tiene un tiempo establecido de seis horas, distribuidas en tres sesiones, con una evaluación parcial en cada sesión. Una estimación del número de ficheros que puede contener el producto final sería entorno a diez ficheros para código fuente, dos ficheros de lotes para automatizar el proceso de desarrollo, diez ficheros para recursos y tres ficheros con manuales. Aunque la naturaleza de la práctica incide en las competencias transversales es difícil definir su evaluación. El enfoque de evaluación continua planteado con las entregas parciales, exige disponer de mecanismos que ayuden al profesor a evaluar tanta cantidad de información. En este sentido, se obliga al grupo de alumnos a que cada entrega lleve asociada una planificación de tareas, donde se deben definir y agrupar en historias de usuario. La correcta gestión de tareas de cada grupo ha supuesto un 10% de la nota final de la práctica.

En la Figura 1 se muestra una monitorización global del proceso de prácticas en la última entrega. Cada fila recoge los resultados de cada grupo de prácticas. La información de las celdas relacionada con las tareas e historias muestra el número total, y entre paréntesis las pendientes por completar. De ese modo, se observa que los grupos G03 y G06 no han terminado su planificación, y que la agrupación de tareas en historias de G03 no parece adecuada puesto que sólo ha definido una historia.

Además de esta información se permite: recoger la traza de cada tarea, la definición y organización, el tiempo estimado, los incrementos de tiempo y qué miembro del equipo realizó la tarea.

	Permission	Bugs	Stories	Tasks
Prav0910G01	Write	0 (0 open)	6 (0 open)	17 (0 open)
Prav0910G03	Write	0 (0 open)	1 (1 open)	10 (8 open)
Prav0910G04	Write	0 (0 open)	7 (0 open)	27 (0 open)
Prav0910G06	Write	0 (0 open)	4 (3 open)	10 (6 open)
Prav0910G02	Write	0 (0 open)	6 (0 open)	8 (0 open)
Prav0910G05	Write	0 (0 open)	3 (0 open)	16 (0 open)

Figura 1. Monitorización global de tareas con XP_DEV

4.2. Planificación y gestión de proyectos

En esta asignatura se introduce de manera teórica y práctica los conceptos de gestión de configuración de software (SCM) [3]. Se familiariza al alumno con conceptos

fundamentales tales como[1]: versionado de elementos, creación de líneas base (*releases, baselines*), integraciones (*merges*), desarrollo paralelo (*branching*), técnicas de integración (*big bang*, integración continua, integración controlada), relación de SCM con *testing* y con el ciclo completo de construcción de software (*testing* automático, gestión de tareas, control diario de proyecto), relación de SCM con métodos ágiles y entrega incremental.

El objetivo de las prácticas es familiarizar al alumno con los conceptos de SCM mostrados en teoría. Se busca que el alumno domine las técnicas básicas de gestión de configuración utilizando una herramienta concreta, como Plastic SCM. Las prácticas se organizan en tres sesiones guiadas por el profesor donde cada alumno dispone de una instalación propia del servidor de Plastic SCM.

En la primera sesión de dos horas, *conceptos básicos*, se practican las operaciones básicas de versionado de elementos: añadir elementos al control de versiones, crear modificaciones y visualizar diferencias.

En la segunda sesión de dos horas, se practica con técnicas de trabajo "*main line development*" o "*trunk development*". Se trata de la técnica más simple de SCM y se explica al alumno sus ventajas e importantes carencias. Se trabaja sobre una única rama con varios espacios de trabajo y se simulan conflictos debidos a cambios concurrentes en los mismos archivos. Se introduce el concepto de "*merge*" durante el proceso de "*checkin*". Se explica el funcionamiento del "*merge* de tres vías" (*three-way-merge*) sobre una única rama, explicando el algoritmo de cálculo de antecesor común desde una perspectiva genérica aplicable a varias herramientas.

En la tercera sesión de cuatro a seis horas, *Branching* y *merging*, se explica el concepto y utilidad del trabajo con ramas y del desarrollo paralelo como paradigma superior a "*main line development*". Se explican patrones de *branching* como "*branch per task*" y "*branch per developer*". Se destacan las ventajas de dichos patrones y su impacto en la gestión de proyecto, estabilidad de versiones, trazabilidad de cambios y relación con gestión de requisitos. Los alumnos trabajan en varias ramas y realizan cambios conflictivos en los mismos ficheros forzando integraciones posteriores. Se explica el concepto de integración controlada, la figura del integrador y las ventajas y desventajas respecto a integración continua.

[1] Debido a la falta de consenso en la traducción de ciertos términos técnicos, los autores han optado por mantenerlos en inglés para no dificultar su comprensión.

4.3. Diseño y mantenimiento del software II

En esta asignatura se tratan conceptos como la calidad basada en inspecciones mediante métricas de código [18], e inspecciones de diseño basada en patrones orientados a objetos [8]. Los resultados de las inspecciones son informes de anomalías que provocan reestructuraciones de código. En el caso concreto de esta asignatura se realizan mediante refactorizaciones de código [7].

El concepto de refactorización de código es nuevo para los alumnos. Para su explicación práctica se utilizan cuatro sesiones de prácticas guiadas, donde a los alumnos en paralelo con el profesor, realizan varios casos prácticos de refactorización utilizando el entorno de desarrollo Eclipse. El primer caso práctico dura dos sesiones y se fundamenta en el ejemplo guía que se presenta en [7]. El profesor a modo de entrenador realiza previamente las refactorizaciones desde el entorno de desarrollo mostrándolas mediante un proyector. El segundo ejercicio, se basa en el supuesto práctico disponible en [6]. Como estrategia docente se disminuye la intervención del profesor, que se limita a analizar los problemas del sistema y a sugerir refactorizaciones para mejorar su diseño.

Ambas prácticas se realizan en grupos de dos alumnos y uno de los problemas que plantea es el seguimiento de las mismas. Para abordar el problema, se usa la extensión de Eclipse para trabajar como cliente del sistema Plastic SCM, y se crea en un servidor central, un repositorio para almacenar los trabajos de la asignatura y gestionar las cuentas de usuarios. En cada sesión se crea un conjunto de ramas, una por cada grupo de prácticas. Cada una de las tareas de la sesión es enviada al repositorio de control de versiones. La Figura 2 muestra de manera gráfica la monitorización de una sesión. Cada grupo tiene su rama (G1_1….), y cada cuadrado en la línea de vida se corresponde con una operación de entrega de documentación al repositorio. Gráficamente se

observa que el grupo G1_1 puede estar perdido porque no está realizando entregas.

Posteriormente a los alumnos se les propone la realización de una práctica obligatoria donde de manera no guiada identifican y resuelven anomalías en un sistema existente. Esta práctica dura tres sesiones de dos horas y se realiza en grupos de cuatro personas.

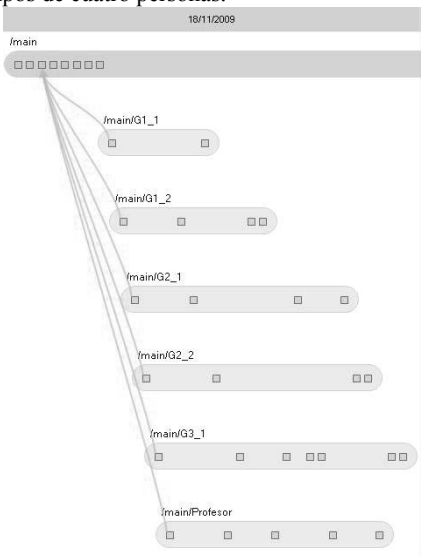

Figura 2. Monitorización de versiones en una sesión de prácticas

4.4. Sistemas Informáticos

Esta asignatura se corresponde con la asignatura de trabajos final de carrera de II. Se sigue un enfoque clásico donde los tutores plantean un proyecto que el alumno tiene que resolver [13].

Es realmente en esta asignatura donde el alumno tiene que poner en práctica muchas de las competencias adquiridas y cuando puede hacer un uso de manera libre de las herramientas proporcionadas. En el caso de solicitar utilizar el servidor de versiones a los encargados de las asignaturas, se habilita un repositorio visible también para los docentes de la asignatura. Hasta el momento, en este curso, han solicitado la creación de repositorios tres de trece proyectos en curso.

5. Evaluación de la experiencia

En las asignaturas expuestas se ha realizado una serie de encuestas anónimas a los alumnos. La encuesta está compuesta por un conjunto de preguntas (ver Tabla 3) donde se relaciona la herramienta a evaluar (XP_DEV, Plastic SCM) con las competencias transversales denotadas en la segunda columna como Ci, y seleccionadas según la Tabla 1. Cada pregunta se valoraba de 1 a 5 (1 = nada, 5 = mucho, las respuestas en blanco no se valoran).

P1. ¿Crees que el uso de *la Herramienta* puede mejorar la organización y planificación respecto a actividades asociadas al proceso desarrollo de software?	**C1**
P2. ¿Crees que el uso de *la Herramienta* puede mejorar tus conocimientos de informática relativos al proceso configuración de software?	**C2**
P3. ¿Crees que el uso de *la Herramienta* puede mejorar tu capacidad de gestión de información asociada al proceso desarrollo de software?	**C3**
P4. ¿Crees que el uso de *la Herramienta* puede mejorar tus conocimientos de planificación y gestión del tiempo?	**C5**
P5. ¿Crees que el uso de *la Herramienta* puede ayudarte a trabajar en equipo?	**C4**
P6. ¿Utilizarías *la Herramienta* en el desarrollo del proyecto fin de carrera?	**C2**

Tabla 3. Preguntas de la encuesta e identificador de competencias

La Tabla 4, Tabla 5 y Tabla 6 muestran los resultados de evaluar las experiencias con los alumnos.

5.1. Análisis de datos de las encuestas

Para cada una de las tres encuestas, cada pregunta se ha sometido por separado a un *test t* de una muestra (*one-sample t-test*) con nivel de significación del 5%, con objeto de ver si alguna de las encuestas rechaza la "*hipótesis nula de que la puntuación media global es 3*". Los casos en los que se rechaza dicha hipótesis son los siguientes. La pregunta 6 en la Tabla 5 y Tabla 6 es valorada por los alumnos por encima de como cabría esperar. Esto puede deberse a que los alumnos de la asignatura PA son de primer ciclo, y todavía no han realizado un proyecto final de carrera, por lo que les falta la experiencia de cuáles son las dificultades que entraña y en las situaciones en las que estas herramientas son útiles. Sin embargo, los alumnos de las otras dos asignaturas sí tienen esa visión y lo valoran

positivamente. Además, en la asignatura PGP, existen bastantes preguntas ponderadas por encima de cómo cabría esperar, rechazando la hipótesis nula (P2, P3 y P6). Una posible interpretación es que la asignatura PGP no utiliza la herramienta tanto a nivel de seguimiento de código, como las otras dos asignaturas. Por ello, el alumno puede que tenga una visión más optimista en lo relativo al coste de su uso integral a lo largo de un desarrollo.

Encuestados = 13			Matriculados = 31			
	Puntuación / Frecuencia					
	1	2	3	4	5	Media
P1	7,7%	46,2%	23,1%	23,1%	0%	2,62
P2	15,4%	23,1%	38,5%	23,1%	0%	2,69
P3	0%	38,5%	61,5%	0%	0%	2,62
P4	7,7%	23,1%	38,5%	15,4%	15,4%	3,08
P5	0%	46,2%	23,1%	30,8%	0%	2,85
P6	15,4%	0%	15,4%	61,%5	7,7%	3,46

Tabla 4. Resultados de encuesta PA

Encuestados = 12			Matriculados = 20			
	Puntuación / Frecuencia					
	1	2	3	4	5	Media
P1	8,3%	0%	16,7%	50%	25%	3,83
P2	0%	0%	33,3%	66,7	0%	3,67
P3	0%	0%	50%	50%	0%	3,50
P4	0%	8,3%	41,7	50%	0%	3,42
P5	0%	25%	8,3%	41,7%	25%	3,67
P6	0%	0%	16,7%	50%	33,3%	4,17

Tabla 5. Resultados de encuesta PGP

Encuestados = 10			Matriculados = 11			
	Puntuación / Frecuencia					
	1	2	3	4	5	Media
P1	10%	40%	30%	20%	0%	2,60
P2	10%	20%	30%	40%	0%	3,00
P3	10%	10%	40%	40%	0%	3,10
P4	30%	40%	20%	10%	0%	2,10
P5	20%	40%	10%	30%	0%	2,50
P6	0%	0%	10%	40%	50%	4,40

Tabla 6. Resultados de encuesta DMSII

6. Conclusiones y líneas futuras

El trabajo se fundamenta en la necesidad de incorporar mecanismos de control para garantizar la adquisición de competencias transversales por parte de los alumnos. En este sentido, se ha propuesto un enfoque de monitorización del proceso de desarrollo y mantenimiento de código basado en herramientas.

Las herramientas genéricas de contenidos educativos deben ser adaptadas a los contextos específicos de enseñanza para mejorar la adquisición de competencias transversales. Esta característica ha sido considerada en ciertas herramientas como Moodle que permite su extensión de actividades mediante módulos. En una última etapa de maduración de la experiencia se deberían desarrollar módulos que integrasen la funcionalidad de las herramientas propuestas con alguna plataforma educativa.

En ciertas valoraciones de los alumnos no se manifiesta una relación entre el uso de las herramientas y la adquisición de competencias. A pesar de ello, es destacable que en todos los cursos encuestados, los alumnos valoran muy positivamente la posible utilización de dichas herramientas en el desarrollo de su futuro proyecto fin de carrera (P6). Es decir, valoran positivamente la competencia de adquisición de conocimientos de informática relativos al ámbito de estudio.

Además de las tareas del profesor mencionadas en la Tabla 2, la puesta en funcionamiento de estos nuevos sistemas lleva asociada una carga extra de trabajo relacionada con la administración del sistema.

Los autores son conscientes que la experiencia debe ser mejorada y madurada estableciendo un marco más preciso de monitorización que permita una evaluación más cuantitativa y que incluso forme parte de la evaluación del alumno en una asignatura.

Este trabajo entronca con otros trabajos como [10], donde se expone de forma muy detallada un sistema de evaluación basado en competencias, y más concretamente en [21] con la evaluación de competencias en los Trabajos Fin de Carrera. El aplicar dichos sistemas sobre el trabajo presentado se establece como línea de trabajo futuro.

Agradecimientos

Este trabajo ha sido realizado por el Grupo de Innovación Docente de la Universidad de Burgos DIGIT. Y ha sido financiado parcialmente por la Agencia para la Calidad del Sistema Universitario de Castilla y León y el Banco Santander. Clave Orgánica [18M9.M2].

Referencias

[1] Agencia Nacional de Evaluación de la Calidad y Acreditación, Libro Blanco. Título de grado en Ingeniería Informática. 2004.

[2] Beck, John. Using the CVS version management system in a software engineering course. *J. Comput. Small Coll.* 20, no. 6 (2005): 57–65.

[3] Berczuk, S.P. y B. Appleton, Software Configuration Management Patterns: Practical Teamwork, Effective Integration. 2002: Addison Wesley.

[4] Cigas, John. An introduction to version control with Subversion: tutorial presentation. *J. Comput. Small Coll.* 25, no. 5 (2010): 213–213.

[5] CódiceSoftware. Plastic SCM. Sistema de control de versiones. 2009; Disponible en: http://www.codicesoftware.es/.

[6] Demeyer, S., S. Ducasse, y O. Nierstrasz, Object-Oriented Reengineering Patterns. 2003, Elsevier Science.

[7] Fowler, M., Refactoring: Improving the design of existing code 1999: Addison-Wesley.

[8] Gamma, E., et al., Patrones de diseño: elementos de software orientado a objetos reutilizable 2006, Addison Wesley.

[9] Glassy, Louis. Using version control to observe student software development processes. *J. Comput. Small Coll.* 21, no. 3 (2006): 99–106.

[10] Jiménez, F., et al., Un sistema de evaluación basado en competencias: Ejemplo para la asignatura Tecnología de la Programación del título de Grado en Ingeniería Informática por la Universidad de Murcia, en Actas de las XV Jornadas de Enseñanza Universitaria de la Informática (JENUI 2009). 2009, Barcelona. p. 193-200.

[11] Larman, C., Agile and Iterative Development: A Manager's Guide. 2004: Addison-Wesley Professional

[12] López, C. y Marticorena, R., Inclusión de patrones de diseño en un plan de estudios de Ingeniería Técnica en Informática, en Los estudios de ingeniería informática en EEES contexto y realidad en la comunidad autónoma de CyL, F.J. García Peñalvo, Editor. 2006, Universidad de Salamanca. Colección Aquilafuente. p. 17.

[13] López, C., et al., Proceso de Gestión de Trabajos Fin de Carrera, en Actas de las XV Jornadas de Enseñanza Universitaria de la Informática. 2009: Barcelona. p. 413-420.

[14] Milentijevic, Ivan, Ciric, Vladimir, y Vojinovic, Oliver. Version control in project-based learning. *Comput. Educ.* 50, no. 4 (2008): 1331–1338.

[15] Ministerio de Educación y Ciencia, Real Decreto 1393/2007, Gobierno de España, 2007, BOE nº 260.

[16] OpenSource. Moodle. Entorno de Aprendizaje Virtual. 1999, Disponible en: http://moodle.org/.

[17] OpenSource, Subversion. Software de sistema de control de versiones 2000. Disponible en: http://subversion.apache.org/

[18] Piattini Velthuis, M.G., Calidad en el desarrollo y mantenimiento del software, 2002: Ra-Ma.

[19] Reid, Karen L., y Wilson, Gregory V. Learning by doing: introducing version control as a way to manage student assignments. *SIGCSE Bull.* 37, no. 1 (2005): 272–276.

[20] Ruiz-Bertol, F.J. y F.J. Zarazaga-Soria. El Control de Versiones en el aprendizaje de la Ingeniería Informática: Un enfoque práctico, en Actas de las XIII Jornadas de Enseñanza Universitaria de Informática. 2007. Teruel.

[21] Valderrama, E., et al., La evaluación de competencias en los Trabajos Fin de Carrera, en Actas de las XV Jornadas de Enseñanza Universitaria de la Informática. 2009: Barcelona. p. 405-412.

[22] XP-Dev. Subversion Hosting integrated with Trac Hosting & Project Tracking for Open Source and Proprietary projects. Disponible en http://www.xp-dev.com/