Una propuesta de primer curso de programación basada en competencias transversales

Pedro J. Clemente, Alberto Gómez, Julia González,
Héctor Sánchez*, Encarnación Sosa
Departamento de Informática
Escuela Politécnica, *Centro Universitario de Mérida
Universidad de Extremadura
e-mail: {jclemente, agomez, juliagon, sasah, esosa}@unex.es

Resumen

Uno de los objetivos de la elaboración de un plan docente para la materia de programación en un primer curso del título de grado en Ingeniería Informática adaptado al Espacio Europeo de Educación Superior es fijar, además de los contenidos propios que estas asignaturas deben contemplar, un conjunto de actividades que permitan al alumno empezar a adquirir algunas de las competencias transversales que debe alcanzar al finalizar su proceso formativo. En este trabajo se presenta una propuesta de actividades, en el marco de un primer curso de programación, para conseguir algunas de estas competencias, tratando además que sean independientes del enfoque elegido a la hora de seleccionar los contenidos en cada materia, lo que permitirá que esta propuesta pueda ser utilizada de forma más generalizada.

1. Introducción

El proceso de integración de nuestras universidades al Espacio Europeo de Educación Superior (EEES) es un enorme reto y una gran oportunidad para avanzar hacia una educación universitaria de mayor calidad, aunque va a suponer un gran esfuerzo por parte de todos, desde las instituciones a los alumnos, pasando indiscutiblemente por los profesores.

Uno de los cambios más importantes a los que nos enfrentamos es una de las directrices fundamentales del nuevo espacio, la adopción de un modelo de educación centrado en el aprendizaje del estudiante. El papel que desempeña el profesor en este nuevo sistema ha de cambiar, no debe limitarse exclusivamente a la figura de contenedor de conocimientos y mero transmisor de los mismos. De forma general, el nuevo papel del profesor debe ser, junto con el de trasmisor, el de asesor, orientador e incluso dinamizador, adoptando nuevos métodos pedagógicos que permitan al alumno involucrarse de forma activa en su aprendizaje, proceso cuyos objetivos no son exclusivamente la adquisición de conocimientos específicos de los títulos, sino también el desarrollo de competencias, habilidades y capacidades que le permitan ejercer su profesión y continuar aprendiendo a lo largo de toda su vida.

Partiendo de estos últimos objetivos, presentamos en esta ponencia parte del trabajo realizado por los autores dentro de la I Convocatoria de acciones para la adaptación de la Universidad de Extremadura al Espacio de Europeo de Educación Superior. Estas acciones están encaminadas a la coordinación entre los profesores de una misma titulación y a la aplicación de las directrices fundamentales del EEES en la elaboración de planes docentes de distintas materias.

Como parte fundamental de la elaboración del plan docente de una materia, además de su contextualización, los objetivos, los contenidos, la evaluación, la bibliografía, etc., hay que hacer explícitos la metodología docente y el plan de trabajo del estudiante. Por tanto, se deben establecer las actividades de enseñanza-aprendizaje que se van a llevar a cabo, junto a las consideraciones relacionadas con la distribución del tiempo (y su contabilización con créditos ECTS) y las consideraciones metodológicas oportunas.

La selección y estructuración de los conocimientos en la materia de programación

siempre despierta controversia (¿programación orientada a objetos desde el principio?, ¿enfoque formal o no tanto?, ¿recursividad antes o después?, etc.; sólo hay que ver las actas de pasadas JENUI) y existen varios enfoques posibles para un primer curso. Pese a ello, consideramos que las competencias transversales [1] que debe adquirir un alumno que se inicia en la programación de ordenadores son, en gran medida, las mismas. Por ello, hemos querido diseñar un conjunto de actividades encaminadas a la adquisición de algunas habilidades y aptitudes fundamentales para nuestros estudiantes, y que se pueden aplicar de manera independiente de la selección de contenidos realizada.

El resto del artículo se estructura de la forma siguiente: en la sección 2 se introducen algunas cuestiones sobre el aprendizaje basado en competencias. En el apartado 3 se identifican las principales competencias transversales que consideramos que un alumno de Ingeniería Informática debería adquirir en su primer curso, centrando nuestra atención en aquellas que deben fomentarse desde la materia concreta de programación. A continuación, en la sección 4 se expone nuestra propuesta, basada en la identificación y planificación de las actividades, junto con los requerimientos materiales y humanos para llevarlas a cabo, y la evaluación necesaria para conseguir que el alumno adquiera las competencias transversales identificadas. Por último, en la sección 5 se discuten las principales conclusiones de este trabajo y las líneas de actuación futuras.

2. Aprendizaje basado en competencias

Según el Diccionario de la Real Academia, una de las acepciones de la palabra competencia es "Pericia, aptitud, idoneidad para hacer algo o intervenir en un asunto determinado". Según el contexto, en algunos casos se utiliza como sinónimo de destreza, habilidad, aptitud, etc.

El proyecto Tuning [2], una de las referencias esenciales en la convergencia al EEES (aunque sin referencia directa a los estudios de Informática), indica que hay un gran consenso para entender los títulos universitarios según "las actividades que los poseedores de dichos títulos estarían en capacidad de desempeñar", es decir, sus competencias. De esta manera, la evaluación del aprendizaje de los estudiantes no sólo debería estar basada en los conocimientos sobre la materia en cuestión, sino que debe incluir una "evaluación basada en las competencias, capacidades y procesos estrechamente relacionadas con el trabajo y las actividades que conducen al progreso del estudiante y a su articulación con los perfiles profesionales definidos". Esta percepción del proceso educativo basado en competencias lo acerca a los procesos productivos donde el alumnado debe desarrollar su futura labor profesional.

Esto nos sitúa, como profesores, en un escenario distinto al actual, aunque quizás no radicalmente distinto en todas las materias. Por ejemplo, la enseñanza de la programación siempre ha incluido, además de la transmisión de una serie de conocimientos, el aprendizaje de unas destrezas básicas como son la capacidad de análisis y la resolución de problemas.

El cambio fundamental para nosotros será tomar conciencia de todas las competencias que debemos fomentar, poner los medios y evaluarlas, así como coordinar nuestros esfuerzos con el resto de materias para que nuestros titulados acaben adquiriendo esas competencias que les posibilitarán ejercer su profesión.

Las competencias que un titulado en Ingeniería Informática debe alcanzar están plasmadas en el libro blanco [1], en el que se identifican y ordenan en función de los perfiles profesionales de los futuros titulados:

- perfil profesional de desarrollo de software,
- perfil profesional de sistemas y
- perfil profesional de gestión y explotación de tecnologías de la información.

Sin embargo, los mecanismos necesarios para conseguir que en la titulación se fomenten dichas competencias no están definidos. Por tanto, son las universidades y sus profesores los encargados de determinar las actividades necesarias para conseguir que sus alumnos obtengan las competencias atribuidas a la titulación.

3. Competencias transversales

Generalmente, las competencias se dividen en dos grandes grupos: las competencias *específicas* de la titulación (entre ellas, los conocimientos), y las competencias *genéricas* o *transversales*, que son comunes a casi todas las titulaciones e imprescindibles para que un titulado superior se

integre en el mercado laboral y pueda desarrollar toda su carrera profesional.

El libro blanco [1] y el proyecto Tuning [2] clasifican las competencias transversales en:

- *Competencias instrumentales*: capacidad de comprender y manipular ideas, organizar el tiempo y las estrategias de aprendizaje, tomar decisiones o resolver problemas, uso de tecnologías, comunicación oral y escrita, y de una segunda lengua.
- *Competencias interpersonales*, de interacción social y cooperación: habilidades críticas y de autocrítica, habilidades de relaciones interpersonales, capacidad de trabajo en equipo, compromiso social y ético.
- *Competencias sistémicas,* integradoras de los dos niveles de competencias anteriores: capacidad para diseñar nuevos sistemas, y planificar los cambios y mejoras de un sistema.

Los colectivos encuestados en el libro blanco (empleadores, titulados y profesores) alcanzan un alto grado de coincidencia al ordenar las competencias transversales según su importancia. Para los tres grupos, las cuatro capacidades principales, del total de diecinueve, son las siguientes:

- Capacidad para resolver problemas
- Capacidad de análisis y de síntesis
- Trabajo en equipo
- Capacidad de organización y planificación

4. Nuestra propuesta

En esta sección se presenta un conjunto de actividades propuesto para conseguir que nuestros alumnos obtengan algunas competencias transversales que consideramos importantes y que se pueden desarrollar dentro del contexto de un primer curso universitario [3], y de forma más específica, a través de las asignaturas de programación.

Entre estas actividades, hay algunas que ya estamos llevando a cabo en las actuales asignaturas que estamos impartiendo, y con este trabajo hemos hecho explícitas las habilidades que queremos que consigan nuestros alumnos. En otros casos, sugerimos algunas actividades que pretendemos poner en práctica y que hemos visto que han obtenido buenos resultados por profesores que ya las han implantado. Para esto, las actas y la asistencia a anteriores ediciones de JENUI ha sido una ayuda inestimable.

Todas estas actividades están pensadas para nuestro entorno, teniendo en cuenta los limitados recursos actuales, tanto materiales como humanos, aunque con la esperanza de una financiación adicional para la convergencia al EEES (que si no se consigue, incidirá de forma negativa en este proceso y probablemente nos llevará a mantener un sistema educativo similar al actual).

Se han dividido las actividades presenciales según el tamaño del grupo de alumnos: grupos grandes (el tamaño adecuado sería de 60 alumnos, pero probablemente deban formarse grupos de 100 ó más alumnos, considerando los grupos de teoría actuales, cuyo tamaño oscila entre los 150 y 180 alumnos), grupos pequeños (con un número ideal de 20, pero que podría aumentarse hasta 30) y actividades tutoriales (en grupos de 4 ó 6 alumnos), con una atención personalizada. Además, se incluirán otras actividades que hemos denominado no presenciales que los alumnos pueden realizar sin la presencia del profesor, ayudados por las nuevas tecnologías de la información.

4.1. Objetivos

El objetivo de estas actividades, planificadas para un primer curso de introducción a la programación, es conseguir una serie de competencias, específicas y transversales, que le permitan al estudiante afrontar con éxito las asignaturas de cursos posteriores, tanto por los conocimientos adquiridos, como por las habilidades que hayan empezado a desarrollar.

Es fundamental que la carga de la asignatura a lo largo del curso esté uniformemente repartida, de manera que los alumnos deban esforzarse de manera continua, sobre todo sin una carga excesiva al terminar el periodo lectivo que le impida dedicar su tiempo a otras asignaturas. Por ello, el esfuerzo y su evaluación deben iniciarse desde el primer día.

4.2. Características

Estas actividades se podrían implementar en una asignatura con una temporalidad anual o bien cuyo contenido estuviese dividido en dos

asignaturas cuatrimestrales de programación en el primer curso de la nueva titulación de Ingeniería Informática.

Uno de los objetivos de las primeras materias de programación es la de enseñar a los alumnos a escribir programas e, independientemente de la secuenciación de contenidos elegida, aprender a escribir programas supone un tiempo de maduración que no es para todo el mundo igual, y donde influye mucho inicialmente el contacto previo que hayan tenido con los ordenadores.

Al menos en nuestro centro, el número de alumnos de nuevo ingreso por año en las tres titulaciones de Informática es de más de 300 alumnos (sin contar con el elevado número de repetidores). Sabemos que algunos de nuestros alumnos, en un porcentaje no excesivamente alto, poseen conocimientos previos de programación, y generalmente traen con ellos malos hábitos a la hora de diseñar e implementar programas. El número de alumnos total es bastante elevado, lo que implica que no existe posibilidad de una atención personalizada por falta de profesorado y espacio.

4.3. Actividades para grupos grandes

Como se acaba de mencionar, la limitación de recursos obliga a realizar una serie de actividades para grupos grandes que, sin lugar a dudas, son más efectivas cuanto menor sea el número de alumnos.

Clases magistrales

Nos referimos a las clases tradicionales, con un papel predominante del profesor como transmisor de conocimientos, aunque no sólo con esa función. Además de conceptos, el profesor debe exponer también las competencias transversales que deben adquirirse, junto con las normas básicas que permitan desarrollarlas y los criterios que vayan a usarse para evaluarlas. Por ejemplo, no es suficiente proponer problemas para enseñar la habilidad de resolver problemas, sino que hay que explicar las distintas formas de afrontar un problema y diversas técnicas de resolución. Usando una analogía, hasta ahora, en muchas ocasiones, enseñábamos a nuestros alumnos a nadar tirándoles directamente a la piscina con unas instrucciones básicas; ahora debemos ser mucho más conscientes de que hay que proporcionarles la información y la formación suficientes para que, cuando se tiren, consigan salir sin problemas, y quieran seguir nadando.

Debe incluirse, como parte fundamental, la resolución de problemas, tanto propuestos con anterioridad, como analizados sobre la marcha. En estas actividades, el profesor actúa como coordinador y moderador de la participación activa de todos los estudiantes.

No es necesario establecer una diferencia entre clases teóricas y de problemas; cada sección tiene sus propias características y el profesor debe tener la libertad suficiente para ajustar los tiempos según sea necesario.

En estas clases deben incluirse, por supuesto, todos los recursos y técnicas innovadoras que puedan hacerse en un grupo grande (críticas y debates, exposiciones breves, etc.) y que estimulen el interés por la materia. Es imprescindible también contar con unos medios técnicos adecuados, por ejemplo, para poder utilizar presentaciones, software de visualización de programas, etc.

Desgraciadamente, aunque muy útil, vemos difícil la posibilidad de ir cambiando el número de horas de dedicación a esta actividad a la semana según las necesidades que se planteen, puesto que sería muy difícil la coordinación con otras materias y la logística en los centros.

Las competencias transversales que se adquieren (además de las competencias específicas) son la capacidad de gestión de la información (el alumno debe discernir aquella fundamental de entre toda la expuesta por el profesor); la resolución de problemas, la toma de decisiones, destinadas al análisis correcto de un problema y a la realización de un diseño para solucionar el mismo. El razonamiento crítico se consigue a través del debate y la propuesta razonada de varias soluciones que permitan al alumno distinguir las ventajas e inconvenientes de cada una de las propuestas.

4.4. Actividades para grupos pequeños (laboratorios)

La utilización de los laboratorios como herramienta fundamental para fomentar en el alumnado un conjunto de destrezas, habilidades y aptitudes no es una novedad en las asignaturas de programación. Es más, debido a que la programación requiere un proceso de maduración, es aquí, en los laboratorios, donde el número de

alumnos es reducido (entre 20 y 30 alumnos) cuando estos se enfrentan con el desafío de desarrollar su propio software.

El conjunto de actividades que proponemos en este entorno es el siguiente:

Desarrollo de software en el aula por parejas

El desarrollo de software por parejas es una de las actividades que no sólo fomentan la competencia asociada al trabajo en equipo, sino que desde un punto de vista educativo les acerca a metodologías de desarrollo en auge. Algunas metodologías actuales, como es el caso de Programación Extrema (XP) [4], hacen de esta práctica una de sus principales señas de identidad. Los alumnos comparten el mismo ordenador (cpu, pantalla y teclado), de modo que, por turnos, uno de ellos codifica mientras que el otro analiza el problema, propone soluciones, prepara juegos de prueba, y por supuesto, revisa la codificación de su compañero. Podemos observar que este mecanismo de trabajo favorece además que el código obtenido sea de mayor calidad. Actualmente en nuestras asignaturas hemos implantado este modelo de trabajo por parejas, junto con un trabajo individual que se realiza fuera del aula.

El método específico utilizado en nuestras clases prácticas ha sido el siguiente:

Antes de la sesión práctica el alumno recoge un guión previo de la sesión. En este guión se realizan preguntas sobre el tema que se va a tratar en el laboratorio, y se proponen un conjunto de ejercicios (2 ó 3) no muy extensos para que el alumno afiance los conocimientos teóricos ya presentados. En la mayoría de los casos estos ejercicios se corresponden con otros muy similares planteados y resueltos en las clases magistrales de la asignatura, donde, como ya se ha comentado, se aborda también la resolución de problemas. Este guión es corregido por el profesor durante la sesión práctica de forma individual, tomando notas sobre su resolución y comentando la solución con los alumnos.

Además, durante la sesión práctica se proponen, mediante un guión de clase, nuevos ejercicios a los alumnos, los cuales ya parten de la base obtenida por el trabajo realizado previamente. La característica principal de estos ejercicios es que el alumno no sólo debe codificarlos, sino que previamente debe realizar un análisis del problema indicando sus precondiciones y postcondiciones (en el caso de que deba desarrollar una función), así como preparar un juego de pruebas adecuado al problema a resolver.

Con el objetivo de obviar inicialmente las características específicas del lenguaje de programación utilizado y centrarnos en los algoritmos, sobre todo en los primeros ejercicios realizados, éstos son implementados primero en pseudocódigo. Los algoritmos en pseudocódigo deben ser validados, aplicando el juego de pruebas en papel. Posteriormente, el algoritmo en pseudocódigo ya validado es pasado al lenguaje de programación correspondiente, en nuestro caso, C++.

Finalmente los alumnos entregan al profesor este guión de clase resuelto, para su posterior corrección, el cual es devuelto en la siguiente sesión con el fin de que exista una retro-alimentación sobre el trabajo realizado. Por una parte, el profesor detecta aquellos fallos más comunes, los cuales pueden ser explicados posteriormente o presentar ejercicios para facilitar a los alumnos su detección y corrección. Por otra parte, el alumnado recibe constante información sobre su proceso de aprendizaje así como información específica sobre sus fallos y aciertos, lo cual lo debe motivar a mejorar en tu tarea.

Los resultados obtenidos en la aplicación de este método han sido satisfactorios. Hemos observado como dos alumnos trabajando tienen menos problemas en desarrollar las prácticas de programación que uno sólo. Además, el trabajo realizado es continuo, permitiendo afianzar progresivamente los conocimientos adquiridos. Esta conclusión también la ratifican los propios alumnos, los cuales, en sus comentarios sobre el modelo de desarrollo adoptado, hacen referencia a que "cuatro ojos ven más que dos". Por otra parte hay que destacar el esfuerzo que el profesor realiza al aplicar esta metodología de trabajo, tanto en la preparación de las sesiones como en su pronta corrección.

Competencias transversales que se adquieren: trabajo en equipo, resolución de problemas, toma de decisiones, capacidad para organizar y planificar el trabajo.

Lectura y compresión de código por parejas o en grupos reducidos

Esta actividad resulta muy importante desde un punto de vista didáctico, ya que permite

mostrar, por un lado, cómo se debe codificar, cómo utilizar estructuras de datos, objetos, ficheros, construcción de algoritmos, documentación interna, estilo de codificación, etc. Además, el alumno desarrolla una actitud crítica sobre el código que no cumple con los ejemplos entregados inicialmente. Este factor, el ejemplo como vía para mejorar, es fundamental en una disciplina como es la programación, donde en muchas ocasiones es necesario apoyarse en ejemplos de desarrollo para abordar nuevos proyectos.

Competencias transversales que se adquieren: capacidad de análisis y síntesis, motivación por la calidad, capacidad para organizar y planificar el trabajo, reforzar la capacidad crítica del alumno.

Intercambiar el trabajo de grupos de trabajo (2 o más)

Esta es una de las actividades que está enfocada a mejorar la motivación por el trabajo que el alumno realiza. Puesto que habitualmente los alumnos trabajarán en pareja, el hecho de intercambiar su código con otra pareja con el fin de probar dicho software incentiva en el alumnado la competitividad, en el sentido disponer de un software de calidad (algoritmos eficientes, código bien documentado, estilo de codificación depurado, juegos de pruebas más completos, etc.). Esta actividad puede realizarse cada dos semanas de prácticas, de modo que los alumnos de forma habitual lean, prueben, y reflexionen sobre el código que sus compañeros les han entregado.

Competencias transversales que se adquieren: trabajo en equipo, motivación por la calidad, capacidad de análisis y síntesis, y resolución de problemas además de desarrollar una aptitud crítica.

Desarrollo de prácticas

La implementación de prácticas de programación es una de las actividades que los alumnos habitualmente desarrollan durante los primeros cursos de las titulaciones de informática. El desarrollo de la competencia de programación por parte de los alumnos está ligado precisamente a que sean ellos los que lleven a cabo todo el proceso, apoyados siempre por el profesorado. Este desarrollo de una práctica debe pasar por un conjunto de fases: Análisis, que ayuda al alumno a mejorar su capacidad de análisis y síntesis. Diseño, que ayuda al alumno a plantear soluciones a los problemas propuestos. Codificación, donde el alumno debe no sólo plantear y resolver el problema, sino también discernir entre las posibles soluciones la más eficiente, desarrollando la competencia basada en la toma de decisiones.

Además, está la necesidad de que sean los propios alumnos los que hagan todo el trabajo, sin ayudas externas no autorizadas (alumnos de cursos superiores y academias que hacen las prácticas, copias y plagios, etc.). La consecuencia directa de esta situación es que el alumno no desarrollará las competencias específicas y transversales previstas, aunque sí podría obtener una calificación positiva. En este sentido, debe desarrollarse un código ético claro de cara a los alumnos, indicando lo que está permitido o no, y por supuesto, castigando a los que no lo cumplan.

Competencias transversales que se adquieren: trabajo en equipo, toma de decisiones, razonamiento crítico, capacidad de análisis y síntesis, resolución de problemas, capacidad para organizar y planificar el trabajo, además de fomentar las capacidades creativas.

Estas actividades deben llevarse a cabo de forma continua, de modo que el alumno madure los conocimientos adquiridos mediante el desarrollo de actividades de aplicación de los conceptos aprendidos durante las clases magistrales y las clases de problemas.

En cuanto a los materiales necesarios para llevar a cabo estas actividades, sería necesario disponer de un ordenador para cada dos alumnos, los cuales sería conveniente que estuviesen conectados mediante una red local para facilitar el intercambio de información entre ellos y el profesorado. Además sería necesario disponer de herramientas software como entornos de desarrollo integrado (editor, compilador, depurador, etc.) dependientes de las características de los lenguajes de programación con los que se desee trabajar.

4.5. Actividad tutorial

Además de las actividades de tutorización general del alumno en el entorno de la titulación y de las explicaciones individuales de los conceptos que no han quedado claros para el estudiante, creemos fundamental la celebración de algunas sesiones a lo largo del curso en grupos muy pequeños, de 4 ó 6 personas, o incluso individuales.

Probablemente, no se puedan llevar a cabo más de 3 ó 4 reuniones en el curso, por lo que deben ser útiles para el alumno y para el profesor. En ellas deben tratarse las principales dificultades que hayan surgido en el desarrollo de la asignatura y también es un buen momento para hacer el seguimiento y una evaluación inicial de algún trabajo que deba realizarse de forma individual o en grupo.

Para el alumno, estas sesiones en grupos muy pequeños deben servirle como realimentación de sus actividades y para conocer exactamente aquellos aspectos que debe mejorar. Al profesor le pueden servir para evaluar algunas de las competencias transversales fundamentales, como la comunicación oral, la capacidad de organización y de gestión de la información y de trabajo en equipo, así como para guiar la realización de algunos trabajos adicionales.

4.6. Actividades no presenciales

El actor principal del proceso de aprendizaje es el propio estudiante, y uno de nuestros objetivos es guiarle para que pueda seguir aprendiendo a lo largo de su carrera profesional. Sin embargo, esto no quiere decir que deba ser él mismo el que tenga toda la responsabilidad. Creemos que en este primer curso debe empezar a adquirir esas competencias que le permitirán el aprendizaje autónomo posterior, pero bajo la supervisión constante del profesor.

Por este motivo, en todas las actividades no presenciales debe establecerse una evaluación junto con un mecanismo de control que permita comprobar que el trabajo se está realizando de forma adecuada.

Entrega de documentos, bibliografía y código

El alumno debe documentarse sobre los contenidos impartidos en la asignatura, aunque debe facilitársele inicialmente documentos base, con una bibliografía adecuada y código asociado (por ejemplo, de software libre) que el alumno puede utilizar.

Con el fin de poder evaluar o contrastar las referencias que los alumnos utilicen, éstas deben figurar en los informes de cualquier tipo que presenten.

Competencias transversales que se adquieren: capacidad de análisis y síntesis, capacidad de organización y planificación, gestión de la información, resolución de problemas, toma de decisiones, razonamiento crítico.

Escritura de informes y de exámenes

Tras proporcionales unas referencias básicas con normas de escritura de textos técnicos, el alumno debería escribir un trabajo breve sobre un tema concreto (por ejemplo, un tema ya explicado, una comparación entre programas, etc.).

Otro trabajo muy útil supone la resolución por escrito de exámenes (por ejemplo del curso anterior), con la posterior corrección por parte del profesor, haciendo hincapié en la claridad en la exposición, errores frecuentes, etc. De esta manera, el alumno puede mejorar su capacidad de comunicación escrita y saber cuál es la mejor forma de exponer sus conocimientos. (La valoración y discusión deberían hacerse durante las reuniones tutoriales.)

*Competencias transversales que se adquieren: co*municación oral, capacidad de análisis y síntesis, organización y planificación, razonamiento crítico.

Foros y repositorios de problemas

Como medio para compartir información y cooperar, puede instalarse un sitio web donde los alumnos puedan consultar y resolver sus dudas, y proponer y discutir soluciones a problemas propuestos. La moderación o no de estos ámbitos por parte del profesorado puede responder a la evolución particular de cada sistema.

Competencias transversales que se adquieren: trabajo en equipo, razonamiento crítico, creatividad, resolución de problemas.

5. Conclusiones y trabajo futuro

Pese al entusiasmo de muchos profesores a los que les interesa su trabajo, las limitaciones en los recursos materiales y humanos de las universidades pueden frenar una implantación profunda del modelo educativo promovido por el EEES. Con todo y con ello, se han propuesto un conjunto de actividades que, enmarcadas en el primer año de docencia de la titulación de grado en Ingeniería Informática, creemos que facilitarán la adquisición paulatina de algunas de las competencias transversales necesarias para el ejercicio de la profesión.

Algunas de las actividades propuestas ya se han implantado en cursos anteriores, aunque sin haber hecho explícito el objetivo de adquisición de competencias transversales, sino simplemente como vehículo para alcanzar los conocimientos básicos de programación.

Para el próximo curso pretendemos implantar la planificación de actividades propuesta anteriormente, y comenzar de este modo una transformación gradual de la docencia hacia el modelo basado en competencias que se implantará definitivamente con el cambio de los planes de estudios dirigidos hacia el Espacio Europeo de Enseñanza Superior.

Agradecimientos

Este trabajo está financiado parcialmente por la I Convocatoria de acciones para la adaptación de la Universidad de Extremadura al EEES.

Referencias

[1] Libro blanco del título de grado en Ingeniería Informática. ANECA 2005. http://www.aneca.es

[2] Proyecto Tuning. 2003. http://www.relint.deusto.es/TuningProject/index.htm

[3] Fermín Sánchez, Ricard Gavaldá. "Objetivos formativos y estrategias docentes para el primer curso de las ingenierías informáticas". Jenui, Alicante 2004.

[4] Sitio web sobre Extreme Programming. http://www.extremeprogramming.org