Re Visión
Revista de Investigación en Docencia Universitaria de la Informática
AENUI Asociación de Enseñantes Universitarios de la Informática

Docencia 2.0

Juan Julián Merelo, Fernando Tricas

La ciencia, la ingeniería y la cultura

En general, en cualquier carrera se enseña la teoría y la práctica relacionada con la misma, obviando aspectos humanos, emocionales o culturales. Si se enseña un lenguaje de programación o las estructuras de datos que se usan en él o los algoritmos o, para el caso, cualquier otra cosa, no se hace énfasis en aspectos humanos, ni culturales, ni laborales. En realidad, prácticamente nada de esas habilidades horizontales con las que tanto nos tuvimos que pelear cuando elaboramos los planes de estudios de Bolonia. El problema es que, cada vez más, esos aspectos culturales son más difíciles de desligar de los puramente técnicos. Metodologías de desarrollo como SCRUM son más culturales que técnicas: se trata de una forma de organizar los equipos de trabajo e interactuar con los compañeros, más que una serie de herramientas o de técnicas. DevOps, el arte de colocar conjuntamente a los encargados de sistemas, de calidad y de desarrollo trabajando conjuntamente sobre un mismo repositorio de código es más un cambio cultural, organizativo, que técnico: pasar de tres departamentos separados a un sólo equipo de trabajo que incluye los tres aspectos. En parte, estos cambios han surgido por la adopción de sistemas basados en la nube. Sistemas no instala ya un ordenador y un servidor para los desarrolladores, provisiona instancias para diferentes tareas y tiene que hacerlo de forma continua. La adopción de métodos de integración y lanzamiento continuo (CI/CD) también hace que los ciclos de prueba del software sean rápidos y, de hecho, se hagan también en la nube en servidores de CI que ejecutan Jenkins o algún software similar. El despliegue, que podría ser antes la parcela de una parte especializada del equipo de sistemas o de desarrollo, se hace ahora usando herramientas tales como Fabric o Foreman, programadas por el mismo equipo integrado. Es complicado reflejar esto en la enseñanza universitaria, que es lo que nos ocupa, porque generalmente refleja esa compartimentalización que aparece en la empresa. Las asignaturas de "sistemas" no hablan con las de "desarrollo" y estas no hablan con las de "Ingeniería del Software". El resultado es un choque cultural de los graduados cuando se incorporan a alguna compañía moderna, una *startup* que tenga un equipo de desarrollo ágil, y se encuentre que ni se usa UML ni siempre el mismo lenguaje ni su labor es desarrollar, sino testear o escribir los *hooks* que se tienen que ejecutar cada vez que se envíe código al repo. Afortunadamente, los tiempos están cambiando. Muchos grados adoptan proyectos integrados que permiten a los alumnos desarrollar prácticas y trabajos fin de grado en un entorno más real; las asignaturas se ponen de acuerdo en integrar sus prácticas para echar abajo los cubículos en las que pueden haberse convertido. Y, finalmente, los alumnos también aprenden a ser más flexibles y a adoptar buenas prácticas desde la escritura de la primera línea de código. En resumen, los profesores estamos aprendiendo a incorporar esa cultura y a que los alumnos aprendan de ella. Poco a poco. Cuando se dan este tipo de integraciones no sólo se pueden mejorar la relación de los conocimientos que proporcionan las diferentes materias, sino que se pueden añadir las habilidades relacionadas con la organización y gestión de proyectos y también las relacionadas con la comunicación, negociación, e interacción con el "mundo exterior". No se trata sólo de las tareas relacionadas con la superación de la asignatura ante el profesor y el resto de la clase, sino también las relaciones dentro del equipo. En general, podemos recomendar equipos de tamaño "no pequeño": no valen dos o tres personas; un mínimo sería cuatro o cinco. También sería bueno que sean suficiente-

JJ Merelo es catedrático de Universidad en el área de Arquitectura y Tecnología de Computadores, y actualmente director de la Oficina de Software Libre de la UGR. Mantiene un blog desde el año 2002, y lo ha utilizado en clase desde el año 2004; también wikis y, ultimamente, agregadores y otras herramientas TIC. Últimamente le ha dado por el *flipped learning*, de lo que se informará debidamente en esta columna.

Fernando Tricas García es profesor titular de Lenguajes y Sistemas Informáticos del Departamento de Informática e Ingeniería de Sistemas de la Universidad de Zaragoza. Empezó a estudiar la blogosfera casi cuando aún no existía (allá por el año 2002) y a tratar de integrarla en los cursos y tareas docentes un poco después. Ha impartido numerosas charlas relacionadas con el tema de la Web 2.0. Es actualmente Director de su departamento.

mente diversos: a esto ayuda el tamaño (no es fácil, en cuanto el equipo crece, que todas las personas tengan relaciones de amistad); esto también refleja lo que seguramente nos sucederá posteriormente en el trabajo, donde no siempre podremos estar en el equipo con las personas más afines. Y puede ayudar al estudiantado a descubrir que hay gente interesante fuera de su círculo. También conviene incidir en los procesos formales: por ejemplo, con reuniones internas del equipo (¿habrá actas? ¿algún otro mecanismo de seguimiento de los acuerdos adoptados?) y con la persona responsable de la asignatura (preguntas relevantes sobre el proyecto, su organización, ayudar a marcar hitos o actuar como "cliente", por ejemplo). Otra pregunta puede hacerse relacionada con el momento. Naturalmente, nadie nace enseñado: en muchas universidades se vienen utilizando asignaturas de proyectos para que los estudiantes tengan una aproximación informal (y conducida por la necesidad) antes de proporcionarles las herramientas necesarias (que vendrán en asignaturas posteriores). Tal vez sería posible intentar incluir este tipo de actividades en etapas más tempranas o más adelante, como refuerzo de aspectos ya aprendidos. En cuanto a qué asignaturas coordinar es difícil dar un consejo genérico: a lo mejor podemos empezar con unas pocas (dos, o tres asignaturas que proponen un proyecto común) y ver si se puede crecer (con la ventaja de que, seguramente, no supone un cambio del plan de estudios y su correspondiente proceso de verificación); pero también se puede apostar por asignaturas dedicadas exclusivamente a esta tarea, que pueden enfocarse en materias concretas o ser mucho más transversales. En resumen, sería conveniente que los profesores tuvieran, al menos conocimiento de estas metodologías de desarrollo y trataran de que el alumno los aprendiera en entornos simulados, pero tan cercanos como sea posible a la realidad.

Todas las columnas de la serie Docencia 2.0 pueden descargarse en formato LaTeX desde `https://github.com/ReVision-Docencia-20/Columnas`