Análisis del enfoque de la asignatura Laboratorio de Software en la Sede Ushuaia de la UNPSJB

Guillermo Feierherd[1], Armando De Giusti[2], Marcela Jerez[2], Alejandro Gimenez [3]

Facultad de Ingeniería – Sede Ushuaia –
Universidad Nacional de la Patagonia San Juan Bosco

Resumen

Se analiza el enfoque y resultados de tres años de implementación de la asignatura Laboratorio de Software en la sede Ushuaia de la UNPSJB.

Las particularidades del curso (Reducido número de alumnos, Conocimientos previos de Análisis y Diseño de Sistemas, Escasa experiencia en desarrollo de proyectos en equipo) nos llevó a plantear un curso centrado en problemas ("problem based learning") buscando estimular las aptitudes de estudio e investigación de los alumnos y al mismo tiempo motivando el trabajo en equipo.

Con un tercer curso en desarrollo es interesante hacer un balance de las ventajas y desventajas apreciadas hasta el momento. En particular tratamos de diferenciar la metodología de dictado en un curso centrado en problemas y las posibilidades de evaluación individual, cuando se ha elegido una técnica que fuerza el trabajo en grupo.

Por último se hacen algunos comentarios sobre el impacto en los alumnos y docentes auxiliares y los principales errores a corregir en los esquemas "problem based".

[1] Profesor Asociado en la Facultad de Ingeniería, Sede Ushuaia de la UNPSJB.
E-mail: feierherdge@ciudad.com.ar
[2] Inv. Principal CONICET. Profesor Tit. Ded. Excl., Facultad de Informática, UNLP. Profesor Visitante en la Sede Ushuaia de la UNPSJB. E-mail: degiusti@lidi.info.unlp.edu.ar
[3] Docentes Auxiliares en la asignatura Laboratorio de Software. Sede Ushuaia. Facultad de Ingeniería. UNPSJB. E-Mail: jerezjm@impsat1.com.ar, alegimen@impsat1.com.ar

Introducción

Laboratorio de Software es una asignatura del tercer año del Plan de Analista Programador Universitario de la carrera de Informática de la Facultad de Ingeniería de la UNPSJB. Se trata de un curso cuatrimestral con 6/8 hs. de carga horaria semanal y objetivos que han variado en el tiempo, a partir de la enseñanza y experimentación en un lenguaje de programación a la consolidación de conceptos de Análisis y Diseño de Sistemas.[Pla99].

Los alumnos que la cursan habitualmente están en la etapa final de la carrera de Analista, con conocimientos previos de Programación y Análisis y Diseño de Sistemas.

La expectativa del alumnado está centrada en la futura actividad profesional como egresado. Desde ese punto de vista, consolidar y poner en práctica los conocimientos adquiridos en Análisis y Diseño de Sistemas les resulta particularmente valioso y útil. Claramente se trata de alumnos con buena formación en Programación y facilidad para resolver problemas centrados en un enfoque algorítmico procedural u orientado a objetos.

Con el contexto que hemos descripto brevemente en el párrafo anterior, se discutieron tres posibles enfoques para el curso, a partir de 1998:

1. Armar un curso clásico, *centrado en un lenguaje,* en el cual el alumno pusiera énfasis en el aprendizaje de una herramienta (que podía evolucionar anualmente o reiterarse) y resolviera problemas de relativa complejidad empleando características asociadas con el lenguaje elegido. Este enfoque tenía algunos puntos de contacto con la asignatura Programación Avanzada que ya se dictaba para los mismos alumnos.

2. Centrar el curso en la *profundización de conceptos de Ingeniería de Software,* armando una especie de “Análisis y Diseño de Sistemas II” o “Ingeniería de Software avanzada”. Si bien esto resultaba interesante por la gama creciente de temas que abarca la Ingeniería de Software, alguno de los cuáles era imposible cubrir en el curso de Análisis y Diseño de Sistemas, resultaba algo alejado del objetivo experimental y de consolidación con el que se había concebido la asignatura Laboratorio de Software.

3. La tercer alternativa, que resultó el enfoque finalmente elegido fue centrar el curso en *una clase variable de problemas, utilizando los fundamentos conocidos de Ingeniería de Software e incorporando conceptos adicionales.* Elegimos un modelo de “problema como foco” buscando la motivación del alumno y al mismo tiempo pensando en la posibilidad de tener soluciones evolutivas en los sucesivos cursos.

El enfoque “problem based learning” elegido resultaba particularmente atractivo para los autores por algunas experiencias docentes previas [Deg96], [Nai96], [Lan97],[Can98],[Sig99].

Metodología propuesta para el curso

Aprovechar el número reducido de alumnos (máximo 8 alumnos) para poder trabajar con ellos como un equipo de producción de software con roles perfectamente definidos, siguiendo las ideas de talleres de Ingeniería de Software tipo “software factory” [Sig00]. En este contexto los docentes auxiliares juegan un rol dual: por momentos se ponen en el lugar del usuario, tratando de hacer que el equipo de producción ajuste sus resultados a lo especificado en los requerimientos y en otras circumstancias operan como líderes de proyecto, dando elementos técnicos de guía al grupo de alumnos.

Buscar un foco de problema que obligue a estudiar una temática “original” para el alumno. Esto se asocia con la idea de que la actividad profesional nos obliga a integrar conocimientos del “mundo real usuario” y que es conveniente conocer los fundamentos del problema sobre el cual se ha de montar una solución informática.

Integrar los temas de Ingeniería de Software necesarios para la especificación, análisis y diseño del sistema, tratando de utilizar los conocimientos ya adquiridos por los alumnos y agregando aquellos que específicamente se requieran para el sistema propuesto o que hagan a aspectos generales del proyecto de sistemas que no han sido cubiertos en el curso regular de Ingeniería de Software (por ejemplo Análisis de Riesgo, Herramientas de Planificación temporal de proyectos, Métricas orientadas a un determinado paradigma de desarrollo).

Interactuar con los alumnos a través de exposiciones de temas abiertos de estudio e investigación y defensa personalizada de la documentación y prototipos que se van desarrollando a medida que el curso evoluciona. Se trató en todo momento de tener un seguimiento similar al de una organización que debe terminar un producto de software en un plazo pre-establecido y de acuerdo a requerimientos detalladamente especificados.

Ejes Temáticos del Curso y comentarios sobre Objetivos pedagógicos.

A. El problema en el que se centra el curso (el primer año elegimos el desarrollo de un sistema distribuído de tiempo real, el segundo año un toolbox para procesamiento de imágenes, en este tercer curso una variante del toolbox creciendo en las funcionalidades de visualización y reconocimiento de patrones)

B. Temas de Ingeniería de Software de carácter general y requeridos por el desarrollo en cuestión: planificación, estimación, planificación temporal, métricas de evaluación de proyectos, métricas de estimación de un sistema de software.

C. Algunos temas de Ingeniería de Software que son variables año a año y se relacionan con el problema en el que el curso está centrado. En el primer año por ejemplo, las extensiones para especificar y verificar sistemas de tiempo real; en el segundo año las herramientas de Análisis y Diseño orientadas a objetos y en particular a ambientes “event driven”; en el tercer año las estimaciones de proyecto basadas en técnicas de Punto Función.

D. Forzar una elección de lenguaje de implementación que el alumno analice cuidadosamente y defienda justificadamente. Admitir que pueda trabajarse con más de un lenguaje en diferentes bloques del sistema. Convencer a los alumnos que el lenguaje es una “herramienta” y que un buen análisis y diseño conduce a desarrollos portables relativamente independientes del lenguaje.

E. Tratar de obtener una respuesta razonablemente similar a la estimada en la fase de planificación. Poner puntos de chequeo semanales para el seguimiento del proyecto. Convertir a los docentes auxiliares en usuarios calificados para controlar la evolución del proyecto.

F. Lograr una evaluación que refleje un conjunto mínimo de conocimientos teóricos nuevos para el alumno y al mismo tiempo verificar un desarrollo (al menos a nivel de prototipo) que funcione según lo especificado.

Resultados Positivos y Negativos

Mencionaremos a continuación 5 aspectos positivos de este enfoque “problem based” y 5 aspectos negativos. Los mismos son autoexplicativos y no requieren demasiado análisis para el lector con experiencia en educación:

Positivos

P1- Alta motivación para llegar a concretar el objetivo de cada curso.

P2- Aprendizaje de nuevos temas por el alumno.

P3- Fomento del trabajo en equipo, la investigación y el autoaprendizaje.

P4- Aproximación a una formación profesional muy valorada por el alumno.

P5- Claridad en los objetivos y en lo que el alumno sabe que debe alcanzar.

Negativos

N1- Dificultades con los alumnos poco creativos.

N2- Dificultad para diferenciar el rendimiento individual

N3- Diferencias notorias en los tiempos planificados y reales (un alumno tiene una capacidad de trabajo promedio muy variable)

N4- Diferencias notorias entre el aprendizaje de conceptos y la implementación de prototipos. Los equipos “saben” lo que tienen que hacer pero les cuesta hacerlo funcionar sin errores y de acuerdo a documentación.

N5- Dificultad para el trabajo en equipo. Hay una tendencia al individualismo.

Conclusiones

El objetivo de estimular el progreso y el desarrollo de la capacidad para resolver problemas de manera autónoma y creativa, es un requisito básico que se espera de la enseñanza universitaria. Asimismo la resolución de sistemas relativamente complejos integrando un equipo de trabajo es altamente beneficiosa para el futuro profesional de los alumnos.

El enfoque centrado en problemas para un Laboratorio de Software parece ser muy adecuado cuando se dan las situaciones de contexto de este caso (reducido número de alumnos, suficiente equipamiento, buena relación docente-alumno y estímulo por la cercanía laboral).

Bibliografía Básica

[Cha00] Champredonde R., Palacios A., Ainchil V. "Programming Teaching Based on Thinking Skills". Publicado en First International Congress on Tools for Teaching Logic, Universidad de Salamanca, Junio 2000.

[Cor96] Córsico, M. “Los aprendizajes en la educación superior”. Encuentro constitutivo de la cátedra UNESCO sobre Pedagogía Universitaria. Presente y Perspectivas. Montevideo: Universidad de la República, septiembre de 1996

[Can98] Canup J., Shakelford R. “Using software to solve problems in large computing courses”. Proceedings of the 29th SIGCSE Technical Symposium on Computer Science Education. Pag 135-139

[Deg96] De Giusti, Bertone,. et al “Introducción a la Programación Concurrente en un Seminario de Lenguajes” Proceedings del II Congreso Argentino de Ciencias de la Computación. Pag. 541-548. San Luis. Argentina. Noviembre 1996

[Lag00] Lage, F., Cataldi Z. Et al “Evaluación contectualizada de Software Educativo”. Informe Técnico Magister en Tecnología Informática aplicada en Educación , UNLP 2000.

[Lan97] Lanzarini L., Naiouf M., De Giusti A “Enseñanza de software de sistemas de tiempo real” Proceedings del III Congreso Argentino de Ciencias de la Computación. Pag. 844-853. La Plata. Argentina. Septiembre 1997.

[May86]Mayer, R. E. “Pensamiento, resolución de problemas y cognición”. Buenos Aires: Paidos, 1986.

[Nai96] Niaouf M., De Giusti A. “Experiencias en la enseñanza de programación paralela” Proceedings del II Congreso Argentino de Ciencias de la Computación. Pag. 549-558. San Luis. Argentina. Noviembre 1996

[Pla99] Plan de Estudios de Analista Programador y Licenciado en Informática de la Facultad de Ingeniería de la Universidad Nacional de la Patagonia San Juan Bosco. Comodoro Rivadavia 1999.

[Sig00] “Using large projects in a Computer Science Curriculum”. SIGCSE Bulletin. Vol 32. Num. 1. Marzo 2000.

[Sig99] “Problem Based Learning”. SIGCSE Bulletin. Vol 31. Num. 1

[Ste86] Stermberg J. Robert. “Las capacidades humanas. Un enfoque desde el procesamiento de la información”. Labor Universitaria, España, 1986.