ENSEÑANZA DE ADMINISTRACIÓN Y DESARROLLO DE PROYECTOS DE SOFTWARE BASADOS EN CMM/SEI ®, BAJO LA MODALIDAD DE TALLER.

María Inés Lund[1], Sergio Gustavo Zapata[2]

Abstract *— El objetivo de este trabajo es presentar la experiencia obtenida en el proceso de enseñanza – aprendizaje, de un estándar de calidad para administrar el proceso de desarrollo de software. Esta experiencia se realiza en un curso de grado, correspondiente al último año de la carrera de Licenciatura en Ciencias de la Información, de la Universidad Nacional de San Juan. Tradicionalmente la forma de impartir este tipo de curso se basa en clases teóricas expositivas. Esta forma de enseñanza dificulta el aprendizaje por parte del alumno, cuando, como en este caso, las temáticas son difíciles de asimilar sin una experiencia concreta Por lo anterior, presentamos una propuesta de un proceso de enseñanza - aprendizaje basado en la modalidad de Taller, en donde se organiza el grupo de alumnos como una empresa productora de software de nivel 2 de CMM/SEI (Capability Maturity Model/Software Engineering Institute) y se los guía en la administración y desarrollo de un proyecto de software para un cliente externo real.*

Index Terms — Administración de Proyectos de Software, Desarrollo de Proyectos de Software, Estándar CMM, Ingeniería de Software.

INTRODUCCIÓN

El software juega un rol cada vez más preponderante en nuestra sociedad, cada vez son más las situaciones de nuestra vida diaria en las que nos encontramos interactuando con software. El software día a día incrementa su participación en actividades cada vez más complejas e importantes de nuestra vida cotidiana. Esta rápida expansión de la industria del software se vincula con la necesidad creciente de ingenieros de software bien formados [3].

El software está creciendo fuertemente en tamaño, complejidad y dominios de aplicación, pero desafortunadamente hay problemas graves en el cumplimiento de planes de costo y tiempo, y en la calidad de desarrollo de la mayoría de los proyectos de software. La mayoría de los proyectos de software de gran envergadura nunca son terminados, y muchos de los terminados no satisfacen los requisitos del usuario y son de pobre calidad [5]. Esta situación incrementa la demanda de desarrolladores de software, no solo formados en los aspectos técnicos y científicos de la computación, sino también que tengan una preparación práctica en ingeniería de software [2]-[4].

La calidad del software depende de una adecuada oferta de profesionales actualizados y competitivos. Los desarrolladores de software han sido formados en la forma tradicional. Desafortunadamente esto no ha producido la oferta y calidad de desarrolladores necesarios para satisfacer esta creciente demanda de software [6].

Tradicionalmente, la forma de enseñar ingeniería de software está basada en clases expositivas en donde se exponen una gran cantidad de conceptos, en la mayoría de los casos en forma teórica, sin experiencias prácticas concretas. Aún en los casos en donde se realizan sesiones prácticas, estas son aplicadas a ejemplos no reales, acotados y ajustados a las necesidades del curso, es decir los llamados proyectos de software de "juguete".

La propia naturaleza de la ingeniería de software, en donde la capacidad de toma de decisiones en situaciones reales es una característica fundamental, hace que este enfoque tradicional pierda efectividad dejando incompleta la formación del alumno.

Para salvar estas deficiencias proponemos una metodología de enseñanza basada en la premisa que *todos los conceptos aportados en el curso se deben aplicar en un proyecto de software real*. Además, realizamos un seguimiento tanto de las opiniones de la industria como de los propios alumnos respecto del curso. Un elemento critico en el éxito de un curso de ingeniería de software es la participación de la industria y la interacción con los estudiantes para evaluar y analizar los distintos aspectos del mismo [1].

La presente propuesta ha sido probada en un curso de Administración de Proyectos de Software, utilizando CMM/SEI como modelo de proceso de software. Uno de los aspectos claves de la misma es la participación de un cliente-usuario real, el cual demanda, a los alumnos, un software basado en necesidades reales surgidas en su medio laboral o empresa. Esta participación externa trae al menos dos beneficios inmediatos:

Motivación del grupo de alumnos: el grupo de alumnos es motivado por el contacto con un medio real, en donde los requisitos, clientes, usuarios y tecnología tienen características reales y no surgidos de un ejemplo académico de libro. La adquisición de conocimiento no técnico (que excede al software) relativo al dominio de aplicación del proyecto también es de gran interés y a la vez motivador para los alumnos.

[1] María Inés Lund, Docente-Investigador del Instituto de Informática de la Universidad Nacional de San Juan. Rep.Argentina. mlund@iinfo.unsj.edu.ar
[2] Sergio Gustavo Zapata, Docente-Investigador del Instituto de Informática de la Universidad Nacional de San Juan. Rep.Argentina. szapata@iinfo.unsj.edu.ar

Interacción con problemas reales: la aparición de situaciones extra-académicas, propias de la ejecución de un proyecto de software real, entrena a los alumnos en la toma de decisiones poniendo en práctica sus capacidades ingenieriles. Estas capacidades no serían posible exponerlas en proyectos académicos de "juguete".

La propuesta aquí presentada ha sido experimentada, analizada y mejorada anualmente, a partir del año 1998. Para ello se contó con la colaboración de los propios alumnos que a través de encuestas de satisfacción dieron indicios de oportunidades de mejoras del curso, que en muchos casos luego se implementaron.

PROPUESTA

Ambiente de aplicación

Se trabaja con alumnos de 5° año de la Licenciatura en Informática de la Universidad Nacional de San Juan. Esta Licenciatura es de 5 años, por lo cual se considera que los alumnos tienen una adecuada base, adquirida a lo largo de la carrera, de conocimientos en temáticas de la Ingeniería de Software como metodologías de Análisis y Diseño, Implementación y Planificación de Proyectos.

El curso se llama 'Sistemas de Información III', pertenece al Área Sistemas y es dictado en el último cuatrimestre de la carrera. El cuerpo docente contempla un profesor titular y un jefe de trabajos prácticos.

Con respecto a infraestructura, se dispone, para los alumnos, un gabinete con 12 equipos en red NT, que es el aula donde toman clases.

Objetivos del Curso

El objetivo principal del curso, es formar al alumno en los estándares internacionales de calidad de software. El alumno, que ya está capacitado a raíz de haber aprobado otros cursos del área, en metodologías de desarrollo de software, etc, necesita adquirir conocimientos de conceptos, técnicas y estrategias de calidad de software, con el fin de alcanzar una formación profesional con alta conciencia y conocimientos prácticos con respecto a la excelencia en el proceso de desarrollo de software.

Los objetivos específicos por unidad temática son:

- **Calidad**: Lograr que el alumno adquiera un conocimiento profesional del concepto de calidad, calidad del producto y calidad del proceso.
- **Administración de Requisitos**: Lograr que el alumno valore la importancia de esta etapa dentro de ciclo de desarrollo de software adquiriendo conocimientos prácticos de formalización de documentación de requisitos, a través de un estándar utilizado para tal fin.
- **CMM**: Lograr que el alumno conozca el alcance de CMM/SEI como herramienta de evaluación y auto mejora de la calidad del proceso de software. Que el alumno conozca los fundamentos y lineamientos generales de los niveles 3, 4 y 5 de CMM/SEI, y conozca en profundidad, por medio de la experiencia práctica, los objetivos y bondades del nivel 2 de madurez de CMM/SEI.
- **Administración de Riesgo**: Lograr que el alumno comprenda, identifique y gestione factores de riesgo del producto, del proyecto y del negocio, en forma práctica sobre el proyecto que comienzan a desarrollar.
- **Inspecciones de software**: Lograr que el alumno conozca los conceptos y fundamentos de inspecciones de software, cuya práctica es desarrollada durante todo el curso.
- **Testing de Software**: Lograr que el alumno adquiera conocimientos teóricos y prácticos de las distintas técnicas de testing de software aplicadas en el proceso desarrollo de software.
- **Satisfacción del cliente:** Lograr que el alumno tenga otra perspectiva de la calidad en donde el cliente-usuario del software tenga un papel protagónico.

Metodología

La primer clase expositiva del curso es introductoria sobre las temáticas abordadas por el curso, modalidad del cursado y la forma de trabajo en el taller, enseñándoles pautas de comportamiento para lograr una adecuada interacción grupal.

Luego existe una serie de clases de exposiciones teórico-prácticas sobre Calidad, analizando diferentes bibliografías, estándares de calidad, etc. Se les brinda a los alumnos un panorama sobre la filosofía conceptual del CMM/SEI, estructura y niveles, en forma general, haciendo hincapié en los fines de cada nivel, las metas de cada área y los objetivos de las prácticas claves.

Al finalizar estas exposiciones se realiza un breve relevamiento en donde cada alumno expresa sus experiencias, conocimientos y preferencias que tienen, con respecto a las etapas, metodologías, técnicas y herramientas, dentro del ciclo de desarrollo de software.

Los alumnos realizarán una presentación en detalle de cada área del nivel 2, agrupándose por afinidad entre ellos en 6 grupos en donde a cada grupo se le asigna un área clave. La presentación de cada área se realiza en clase frente a todos los compañeros. Se evalúa a los alumnos durante la presentación, por conocimiento, claridad y esfuerzo demostrado en la realización de la presentación.

En este punto del curso se toma una evaluación para controlar los conocimientos adquiridos y nivelarlos.

En base a la cantidad de alumnos y a los datos obtenidos en el relevamiento previo, se dividen a los alumnos en 1 o 2 grupos y se les asignan roles, ya sea del área de CMM/SEI nivel 2 en la que deben trabajar, como rol dentro del equipo de desarrollo de software. Cada alumno tiene responsabilidad por dos roles, en un área y en el equipo de desarrollo del software. Cada grupo se transforma en una empresa desarrolladora de software, a la que se le asigna un proyecto real y deben trabajar en pro de ello, durante los 3 meses de cursado que restan.

Durante la ejecución del proyecto, se van dando clases teórico-prácticas, de las diferentes temáticas que se van necesitando a lo largo del mismo. Se les entrega a los alumnos el Manual de Procedimientos de cada área, manual que respeta el estándar CMM/SEI y que fue generado y mejorado por los mismos alumnos en años anteriores.

La primer temática que se aborda es la Administración de Requisitos, en donde, además de prepararlos en técnicas de entrevistas, se enseña y se aplica el estándar de la ESA (Agencia Espacial Europea) para la formalización de Requisitos de Usuarios y Requisitos de Software.

A este punto, los alumnos están listos para recibir la visita del 'cliente real', todos los alumnos del grupo participan, pero son los analistas los encargados de guiar la entrevista. Esta primer visita se realiza en el aula, con el objetivo que todos los participantes del grupo se sientan involucrados en el proyecto desde los inicios y conozcan el objetivo del sistema a desarrollar, pedido por el cliente.

Las entrevistas subsiguientes que sean necesarias, ya son pactadas entre los analistas y el cliente y el lugar de reunión no necesariamente sea la facultad. Los analistas deben elaborar un completo y consistente documento de requisitos de usuario, planteando todo aquello que el cliente necesita y respetando los lineamientos del estándar de la ESA.

Paralelamente, el resto de las Areas se van involucrando en las tareas que deben realizar, estudiando el Manual de Procedimientos que se les entrega para tal fin. Las áreas comienzan a realizar las tareas asignadas. También se les da una clase teórica de Inspecciones de Software, en donde se lo instruye en la forma de realizarlas y de los beneficios consecuentes de aplicar dicha técnica a lo largo de todo el proyecto.

Se realiza inspección al documento de requisitos de usuario. Cuando este es aprobado los analistas deben generar el documento de requisitos de software, planteando todo lo que el sistema realizará, para satisfacer las necesidades enumeradas en el documento anterior y presentar la matriz de trazabilidad correspondiente. Este documento también pasa por el control de una inspección.

El área de Administración de la Configuración debe presentar su plan, generado para este proyecto y acatando lo previsto en el Manual de Procedimientos, para que todas las áreas conozcan los mecanismos con los que se deberán manejar para solicitar información o para entregar documentación para configuración.

Todas las inspecciones son registradas en actas, y estas son ingresadas al área de Administración de la Configuración, bajo el procedimiento formal consignado en el Plan de Administración de la Configuración,

El área de Planificación, ya puede ir comenzando a acordar y establecer tiempos y esfuerzos, a esta altura se les brinda clases teórico-prácticas sobre Administración de Riesgos, y sobre la técnica de Puntos de Función, con el objetivo de que puedan realizar una planificación con calendario y agenda lo más ajustadamente posible.

Las áreas de Seguimiento y Control y de Administración de la Calidad también deben presentar su respectivos planes para el proyecto y deben informar semanalmente, a todo el grupo, sobre los informes de seguimiento y actas de calidad realizados, de acuerdo a lo planificado, con el objetivo de determinar si es necesario tomar medidas correctivas.

Una vez que los documentos de requisitos de usuario y requisitos de software son formalmente aprobados, los diseñadores, que ya vienen avanzando sobre este aspecto, deben tratar de llegar a un diseño con el mayor detalle posible. Generan un documento en donde se describe como hará el sistema para obtener lo especificado en el documento de requisitos de software. Este documento no solo refleja la arquitectura del software y diccionario de datos, sino también el diseño de las interfaces, el mapa de navegación y detalle de programas particularmente complejos, este documento es una versión adaptada de los documentos de diseño del estándar de la ESA.

Una vez aprobado formalmente, el documento de diseño, con una inspección, los implementadores, que previamente se han familiarizado con el lenguaje y entorno de programación, e incluso haber realizado algún prototipo, comienzan a trabajar fuertemente en implementar todo lo indicado en el documento de diseño y se planifican las fechas de inspecciones de código.

En este punto se da una clase de Testing de Software y técnicas de testing. Los encargados de realizar el testing ya pueden comenzar a planificar el testing del software, tomando como base los documentos anteriores aprobados. Este plan de testing debe ser presentado en clase.

Una vez que el sistema es terminado y se realiza la presentación formal, los encargados de testing deben comenzar a testear el software y presentar un informe de resultados de testing.

A esta altura, nos encontramos al final del período de cursado de la materia, se les enseña otra forma de medir calidad, como es la medición de la satisfacción de los usuarios del software.

Se hace un cierre de la materia con una clase en donde los exponen su experiencia, sus sugerencias, su grado de satisfacción y proponen 'mejoras' que se podrían realizar a los Manuales de Procedimientos, de acuerdo a la experiencia obtenida por su uso.

Consideraciones generales

No está de más aclarar que todas las clases teóricas y teórico-prácticas y exposiciones se realizan para todos los alumnos, no solos los involucrados en las temáticas. Las inspecciones también son practicas obligatorias para todos los alumnos del curso.

Algunas veces se puede plantear que en lugar de realizar presentaciones de los planes de las áreas, pueden realizarse Inspecciones a los planes. Esto lo hemos realizado, sobre todo, con el Plan de Planificación del Proyecto.

Los alumnos deben satisfacer un mínimo de 75% de asistencia a clases y exposiciones tórico-prácticas, como así también a las presentaciones de planes, inspecciones, sistema, etc. El resto del tiempo asignado a la materia se destina para que los alumnos trabajen en las tareas que les han sido encomendadas, en aula o en el lugar que ellos dispongan. Estos días no se computan en la asistencia.

CONCLUSIONES

El resultado de aplicar la modalidad de taller para este tipo de currícula, es realmente satisfactorio. Los resultados obtenidos son de muy buen nivel académico y de desarrollo. Los alumnos mismos ven superadas sus expectativas.

No es una tarea fácil, requiere de un dedicado seguimiento al trabajo de los alumnos, muchas veces exigir y tratar de hacer respetar la planificación original, para que vaya de acuerdo a la restricción temporal del cursado.

Las encuestas que miden el nivel de satisfacción reflejan un grado de satisfacción que va del bueno al elevado.

La participación de un cliente externo al ámbito académico es un factor importante de motivación para los alumnos, imponiéndoles restricciones reales en donde las capacidades ingenieriles de los alumnos se pueden llevar a la práctica.

REFERENCIAS

[1] Bagert D. J. , at all., "Guidelines for Software Engineering Education Version 1.0 (CMU/SEI-99-TR-032)", *Pittsburgh, Pa: Software Engineering Institute*, Carnegie Mellon University, October 1999.

[2] Denning, P.J. ,"Educating a New Engineer", *Communications of the ACM*, 35, 12, December 1992, 83-97.

[3] Ellis, H.J.C., Mead N.R., Moreno A., Mac Neil P., "Can Industry and Academia Collaborate to Meet the Need for Software Engineers?", *Cutter IT Journal*, June 2001.

[4] Ford, G. & Gibbs, N.E., Mature Profession of Software Engineering (CMU/SEI-96-TR-004, ESC-TR-96-004), *Pittsburgh, Pa: Software Engineering Institute*, Carnegie Mellon University. URL: http://www.sei.cmu.edu/publications/documents/96.reports/96.tr.004.html, 1996.

[5] Gibbs, W.W., "Software's Chronic Crisis", *Scientific American*, 271, 3, September 1994, 86-95..

[6] Shaw M., "Software Engineering Education: A Roadmap", *The Future of Software Engineering, New York: ACM Press,* 2000, 371-380.