Interactuando con la Industria en el diseño e implementación de un modelo de enseñanza-aprendizaje de Ingeniería de Software

Aliosha Bertini, Cecilia Casanova
Departamento de Ciencias de la Computación
Universidad de Chile
{abertini, mccasano}@dcc.uchile.cl

Abstract

Este documento relata la experiencia obtenida en el proceso de adecuación de la malla curricular para la carrera de Ingeniería Civil en Computación, con especialización en Ingeniería de Software(IS), mediante el cual se ha logrado la realización de un curso-laboratorio obligatorio al final de la carrera, destinado a desarrollar en los alumnos tanto habilidades técnicas, como de gestión y de comunicación humana. Este curso-laboratorio contempla el desarrollo de proyectos de software reales de pequeña escala, propuestos por empresas locales, emulando en el entorno académico, una organización de desarrollo de software real. Con este trabajo, se ha logrado obtener un modelo de enseñanza-aprendizaje en Ingeniería de Software que ha sido diseñado, implementado y validado en forma evolutiva, acercando de este modo la enseñanza de la IS a las necesidades de la industria.

Palabras Clave: Ingeniería de Software, Educación.

1. Introducción

La participación del Departamento de Ciencias de la Computación (DCC) en el grupo SPIN-CHILE (Software Process Improvement Network) desde 1998, motivó la conformación de un grupo de interés en torno a la Ingeniería de Software (IS), integrado por investigadores del departamento y profesores jornada parcial provenientes de la industria.

Como resultado de las reuniones del grupo de interés, se inició un programa de mejoramiento de la enseñanza de IS al interior del departamento. Desde entonces, se han realizado modificaciones importantes al plan de estudios de la carrera de Ingeniería Civil en Computación, incorporando los contenidos de IS según las habilidades técnicas y no técnicas requeridas por las organizaciones de software. En particular, se han realizado modificaciones en el plan de estudios de los tres cursos que abordan el área temática de IS en el departamento. En el presente documento, se describen los cambios que se han realizado en el curso-laboratorio *Proyecto de Software*, de final de carrera, con miras a construir un modelo curricular interdisciplinario, que permita entrenar a los alumnos, tanto en la aplicación de conceptos técnicos, como modelos de ciclo de vida, gestión del proceso y estándares de calidad, y no técnicos, como la comunicación humana aplicada a la relación con un cliente real y al trabajo de equipo en el desarrollo de un proyecto.

2. Motivación y Contexto

La carrera de Ingeniería Civil en Computación del DCC, está orientada a formar profesionales con capacidades para desempeñar actividades de diseño, desarrollo, evaluación y mantención de software complejo (Ingeniero de Software) y la administración de Redes de Datos, Servidores y Seguridad de Sistemas Computacionales (Ingeniero de Sistemas).

Los conocimientos centrales considerados para la formación del Ingeniero de Software como son el proceso de desarrollo de software, la gestión del proceso de software, las normas y estándares de

calidad, el proceso de documentación y la tecnología de apoyo al proceso de software, son entregados progresivamente en los cursos *Desarrollo de Software de Aplicación*, *Ingeniería de Software*, y *Proyecto de Software*, dictados en los semestres seis, nueve y once respectivamente, como se muestra en la Figura 1.

Semestre						
5	**6**	**7**	**8**	**9**	**10**	**11**
Algoritmos y Estructuras de Datos	Fundamentos de Ciencias de la Computación	Análisis y Diseño de Algoritmos	Bases de Datos	Programación Orientada al Objeto	Comunicación de Datos	**Proyecto de Software**
	Programación de Software de Sistemas	Introducción al hardware	Sistemas Operativos	Inteligencia Artificial		
	Desarrollo de Software de Aplicación	Lenguajes de Programación	Computación Gráfica	**Ingeniería de Software**		

Figura 1. Malla Curricular Ingeniería Civil en Computación

En el primer curso, *Desarrollo de Software de Aplicación*, se entregan conceptos teóricos generales sobre el proceso de desarrollo de software, los principales modelos y las herramientas de apoyo al proceso. En cuanto a tópicos de comunicación humana, el énfasis está en el trabajo individual del Ingeniero de Software y su rol como facilitador de la tecnología de software para el que hacer de la sociedad actual.

En el segundo curso, *Ingeniería de Software*, se aplican los conceptos entregados en el primer curso sobre el proceso de software y se entregan nuevos conceptos sobre gestión, metodologías formales para las actividades clásicas (por ejemplo, análisis y diseño estructurado [9, 10], análisis y diseño orientado a objetos [5, 6, 7]), estándares de calidad [11, 14] y trabajo en equipo y relaciones interpersonales por medio de la realización de proyectos pequeños que simulan un ambiente real.

El último curso, *Proyecto de Software*, tiene como objetivo principal, la participación del alumno en un proyecto de software real, enfatizando las actividades de requerimientos, diseño, implementación, trabajo en equipo y la documentación. Inicialmente, los alumnos debían participar en un proyecto de software real durante un semestre en un régimen de cuatro horas diarias. Al final del semestre el alumno debía entregar un informe destacando el trabajo realizado en tres de las etapas clásicas del software: requerimientos, diseño e implementación. En el transcurso y, debido a que los alumnos comienzan a trabajar muy temprano en la carrera, los objetivos del curso fueron diluyéndose de manera que los alumnos no necesariamente realizaban las etapas anteriores, limitándose a entregar un informe con el trabajo realizado en la empresa. Normalmente, este informe consistía de unas veinte páginas, en las cuales el alumno enfatizaba las tareas de implementación llevadas a cabo, sin detallar el trabajo realizado en las primeras fases, como lo son requerimientos y diseño. Esto se debía, principalmente, a que los alumnos no poseían ni los conocimientos ni la experiencia requerida para realizar tales tareas y por lo tanto, los trabajos asignados en las empresas no eran los adecuados para los objetivos académicos del curso.

En el segundo semestre del año 1997, la realización del curso *Taller de Ingeniería de Software*, con la aplicación de un nuevo modelo de enseñanza, basado en el trabajo de James Tomayko y Mary Shaw [1, 2], permitió conocer el grado de dominio de métodos y técnicas en los alumnos, y sus habilidades para abordar proyectos de desarrollo en equipos de trabajo y con desempeño de roles.

En este curso-taller, se abordaron proyectos “ficticios” bajo el modelo de ciclo de vida en cascada [3]. El profesor actuó como Cliente y Gerente de proyectos. Los equipos de trabajo fueron de cinco alumnos y cada equipo se auto-asignó los roles de Jefe de Proyecto, Ingeniero de

Requerimientos, Ingeniero de Diseño, Ingeniero de Implementación e Ingeniero de Pruebas. Uno de los equipos se destinó a actividades de Pruebas y Aseguramiento de calidad, con la tarea de revisar formalmente la documentación y el cumplimiento de los objetivos planteados para cada proyecto. Este curso significó un cambio cualitativo en el método de enseñanza-aprendizaje de la Ingeniería de Software, el que hasta ese momento había estado enfocado a la entrega de conceptos teóricos, y modificó sustancialmente la visión de los alumnos sobre las actividades involucradas en la realización de un producto de software, permitiendo experimentar y comprender mejor los alcances de cada una de las etapas del ciclo de desarrollo y sus complejidades.

Un año más tarde (segundo semestre del año 1998), se modificaron los contenidos programáticos del curso *Proyecto de Software*, incorporando un modelo de enseñanza-aprendizaje similar al *Taller de Ingeniería de Software* del año anterior, con algunas variaciones importantes. En particular, se establecieron vínculos con empresas externas de modo de asignar proyectos reales a los equipos de trabajo. En términos de comunicación humana, se enfatizó la relación con el cliente, es decir, se definió un protocolo de relación con el cliente que permitiera valorar en el desarrollo del proyecto, la satisfacción del cliente y el desarrollo técnico, permitiendo incorporar la lógica del usuario y los conceptos básicos de calidad de software en el desarrollo del proyecto.

Hasta la fecha, el curso se ha dictado en los semestres 98/2, 99/1 y 99/2, realizando doce proyectos en total como se muestran en la Figura 2.

Semestre 98/2	**(1)/(2)**	**Semestre 99/1**	**(1)/(2)**	**Semestre 99/2**	**(1)/(2)**
Sistema de Control y Monitoreo de Vehículos	CS/ DS	Mejoramiento del Proceso de Mantención de Sistemas	CS/DS	Programa de Configuración	CS/DS
Sistema de Control de Vendedores	CS/M	Mesa de Ayuda	W/S	TV Tools	A/DS
Sistema de Normalización de Calles	B/DS	Sistema de Administración y Consulta de Documentación	W/S	Bitácora de Proyectos	W/DS
Mi Mundo en mi mano	PE/DS	Sistema de Control Documentario	CS/S		
Herramienta de Publicación en Intranet	W/DS				
NOTA: (1) **Tipo de Proyecto:** *Stand Alone (SA), Cliente Servidor (CS), Aplicación Batch (B), Prototipo Experimental (PE), Web (W)* (2) **Tipo de Cliente:** *Empresa de Manufactura (M), Empresa de Desarrollo de Software (DS), De Servicios(S)*					

Figura 2. Proyectos realizados hasta la fecha

3. Trabajo Realizado

Debido principalmente a la inexistencia de una infraestructura previa adecuada a la realidad de los alumnos del departamento, el curso nos ha permitido experimentar en todas las áreas y mejorar aquellas que consideramos fundamentales. Una vez que éstas hayan sido definidas, extenderemos nuestro esfuerzo a áreas más especificas. Las áreas en las cuáles se ha enfatizado el trabajo hasta el momento son las siguientes:

- Definición de una organización de software académica
- Proceso de desarrollo
- Trabajo en equipo y asignación de responsabilidades
- Documentos de trabajo y las herramientas de apoyo
- Esquema de evaluación de los alumnos

Los tópicos anteriores se interrelacionan unos con otros y han sido diseñados e implementados en forma integrada. A continuación se describen los aspectos más relevantes de cada uno y las mejoras que se han incorporado.

3.1. Organización de Software Académica

El desarrollo de los proyectos se efectúa bajo la supervisión de una organización de software académica con la estructura que se muestra en la Figura 3. Las contrapartes están representadas por un Coordinador o Jefe de Proyecto por parte de la empresa y un Jefe de Proyecto por parte de la organización de software académica, quien está a cargo de un Equipo de Desarrollo, de entre cuatro a cinco alumnos, con dedicación equivalente a media jornada para un proyecto en particular.

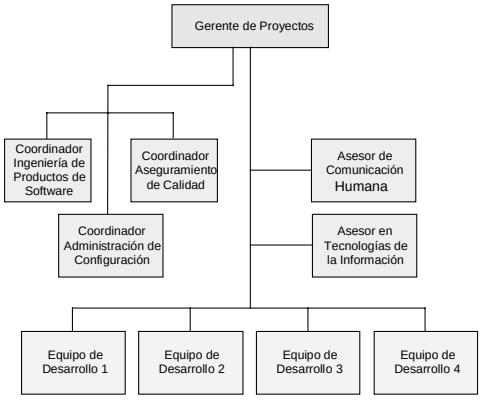

Figura 3. Organización de Software

Los roles de Gerente de Proyectos, Coordinadores y Asesores externos, son cumplidos por los docentes del curso y los equipos de desarrollo son conformados en su totalidad por alumnos del curso.

3.2. El Proceso de Desarrollo

Uno de los aspectos claves en Ingeniería de Software es la identificación de un proceso que permita la generación de un producto de software de calidad, cumpliendo los requerimientos, los plazos y el presupuesto estimado. La comprensión de que la construcción de un producto de software obedece las mismas reglas que otras actividades productivas, es fundamental para dejar de hacer artesanía de software.

En los tres semestres en que se ha dictado el curso, se han aplicado tres procesos de desarrollo distintos: espiral, cascada e incremental.

El semestre 98/2 se aplicó un proceso en espiral [12, 13] para el desarrollo de los proyectos. La poca experiencia de los alumnos en modelos formales de desarrollo, y mecanismos de evaluación inadecuados, produjeron rápidamente que el proceso se transformara en un único ciclo de un semestre, convirtiéndose indirectamente, en un proceso en cascada tradicional.

Uno de los problemas principales que se detectó fue la poca capacidad de los alumnos para diferenciar el alcance de las actividades de ingeniería del proceso: requerimientos, diseño, construcción y pruebas y los productos de trabajo de cada actividad. El resultado fue una etapa final de proyecto muy intensa, en la cual casi todos los integrantes de los equipos tuvieron que apoyar las actividades de construcción. Cabe destacar que el mejor trabajo elaborado ese semestre fue realizado por un equipo que intentó llevar a cabo las entregas en los plazos establecidos y que, además, contó con el apoyo de un cliente que comprendía las sutilezas existentes en el proceso de desarrollo de software. En este sentido, los objetivos del proyecto fueron planteados por el cliente de manera de llevar a cabo un modelo incremental [3, 15], en donde, al final de cada ciclo, se realizó la entrega de un sub conjunto de la funcionalidad contemplada para el proyecto. Los otros proyectos realizaron una actividad de requerimientos tan abstracta y amplia que, cuando fue necesario construir el sistema, no lo lograron completamente y tuvieron que recurrir a negociaciones con el cliente para re-definir los objetivos del sistema.

Tomando en cuenta los resultados del semestre anterior y dado que el proceso en cascada es completamente familiar para los alumnos, el semestre 99/1 se aplicó un proceso en cascada para el desarrollo de los proyectos. En esta situación, los alumnos intentaron cumplir con todas las etapas del proceso, sin embargo, no fueron capaces de dimensionar el esfuerzo requerido para realizar cada una de

las etapas y construir los productos. Los plazos establecidos por cada equipo no fueron cumplidos y, nuevamente, gran parte del esfuerzo tuvo que ser destinado a la etapa de construcción en un plazo de dos a tres semanas. A diferencia del semestre anterior, en que los productos de cada etapa fueron requeridos en fechas pre-establecidas, los equipos definieron las fechas de entrega de los productos de cada etapa en base a una planificación previa. Sin embargo, intentar cumplir los plazos sin un conocimiento detallado de los resultados esperados para cada etapa (documentos de trabajo y modelos), no fue posible. Debido a esto, por ejemplo, la documentación del diseño fue entregada casi en conjunto con los resultados de las actividades de construcción.

Es importante destacar el hecho que las actividades menos aplicadas, para los alumnos del departamento, como son requerimientos y diseño, no representan ni se traducen en "productos necesarios" y constituyen "exigencias externas" que no permiten el "real" avance del proyecto.

Otro hecho importante que se repitió en forma análoga al semestre anterior, es el problema en la definición de los requerimientos del proyecto. La definición de requerimientos es fundamental y condiciona todo el proyecto, pero no es valorada como tal por los alumnos. Los requerimientos son "diseñados y programados" mentalmente a medida que son recolectados, lo que produce una sensación de seguridad extrema respecto del término exitoso de los proyectos, ya que los alumnos se autoconvencen que, con unas pocas semanas de trabajo en conjunto con el cliente, el proyecto estará definido completamente, los requerimientos claros, el diseño será mínimo y la construcción no debiera quitar mucho tiempo. Con esta percepción, se dejan de lado aspectos fundamentales como la plataforma de desarrollo, las expectativas del cliente, el cumplimiento de plazos y, principalmente, los riesgos asociados al proyecto.

Como ejemplo concreto, en uno de los proyectos realizados ese semestre (99/1), los alumnos y el cliente se focalizaron en una funcionalidad específica del producto que comprendía el reconocimiento de lenguaje natural. Se buscaron estrategias, se diseñaron soluciones y se investigó en la bibliografía al respecto. Una y otra vez se analizó el problema a resolver y los alumnos no fueron capaces de orientar la atención del cliente hacia el problema real. Como resultado de esto, a tres semanas de la entrega, cliente y alumnos re-evaluaron el alcance del proyecto y establecieron que no se había contemplado ni realizado el estudio del 60% de la funcionalidad requerida y que, además, esa funcionalidad era la que realmente necesitaba el cliente.

Los resultados de los semestres anteriores no permitieron satisfacer las necesidades de los clientes ni las del equipo docente, por lo que el semestre 99/2 se aplicó un proceso incremental [3, 15] para el desarrollo con tres ciclos de un mes cada uno. Cada ciclo debía ser enfrentado como un mini proyecto, incluyendo la realización de todas las actividades de ingeniería y, además, se debía generar un producto parcial instalado en las dependencias del cliente. Este cambio, significó una mejora en el diseño del curso, requirió la generación de más material de apoyo y la re-definición del esquema de evaluación, como se describe en los puntos siguientes.

3.3. Trabajo en Equipo y Asignación de Responsabilidades

Hasta el quinto semestre de la carrera, los alumnos desarrollan principalmente trabajos individuales de desarrollo de software. Por tanto, el trabajo en equipo y la asignación de roles y responsabilidades para lograr un objetivo común, ha constituido una seria deficiencia para el éxito de los proyectos. Para entrenar a los alumnos y mejorar en ellos estas capacidades, en el curso *Proyecto de Software* se realizan dinámicas de grupo y se desarrolla una encuesta para obtener el perfil de los alumnos en términos de su experiencia práctica, dominio de conceptos técnicos, número de asignaturas cursadas, entre otros aspectos. Con los resultados de estas actividades, es el equipo docente quien define los equipos de desarrollo con el fin de simular mejor el ambiente laboral, en el cual, no siempre se elige

con quién trabajar. La asignación de roles, sin embargo, ha sido libre y cada equipo ha definido internamente el rol de cada uno de los integrantes. Desde la primera vez que se dictó el curso, los roles considerados han sido los siguientes: Jefe de Proyecto, Ingeniero de Requerimientos, Ingeniero de Diseño, Ingeniero de Implementación e Ingeniero de Pruebas.

En el semestre 98/2 en base a las dinámicas de grupo destinadas a identificar liderazgos y actitudes, se logró formar equipos de desarrollo relativamente homogéneos. Uno de los aspectos importantes que se considera, al momento de formar los equipos, es la disponibilidad de tiempo de cada alumno. La gran mayoría de los alumnos que asisten al curso trabajan en la industria, por lo que cada equipo debe equilibrarse de modo que puedan dedicar el máximo de tiempo a los proyectos. Las capacidades técnicas y la experiencia previa también son importantes, sin embargo, debido a las características de los alumnos y de los proyectos realizados, éstas no son cuantificables fácilmente. Por ejemplo, gran parte de los alumnos han desarrollado trabajos parciales de programación con herramientas del ambiente Unix (C, Perl, CGI, etc.) y la mayoría de los proyectos han sido realizados con herramientas en ambiente Windows.

Con la experiencia obtenida en cuanto al desempeño de los equipos de desarrollo, a contar del semestre 99/1, se planteó como requisito para la docencia del curso, la incorporación de un asesor en la disciplina de Comunicación Humana. Además de apoyar las actividades para la definición de los equipos de trabajo, se ha integrado esta disciplina para entrenar a los alumnos en el desarrollo de aquellas habilidades requeridas para un buen trabajo en equipo, una buena relación con el cliente y una comprensión cabal de su rol como "facilitadores" de la tecnología, para mejorar el desempeño de las organizaciones.

En cuanto a las responsabilidades de cada rol, éstas son descritas en manuales para cada uno de los roles. Sin embargo, los manuales por sí solos no han permitido transmitir bien a los alumnos sus responsabilidades y ha sido necesario reforzar el compromiso con los roles y el cumplimiento de tareas con un esquema de evaluación más riguroso.

3.4. Documentos de Trabajo, Manuales y Herramientas de Apoyo

A diferencia de otros cursos de la carrera, *Proyecto de Software* es un curso en el cual los equipos deben generar un número importante de productos de trabajo como son:

- Especificación de Requerimientos
- Especificación de Diseño
- Especificación de Pruebas, archivos y datos de prueba
- Estándar de Codificación
- Código fuente y ejecutables
- Manual de Usuario y de Instalación
- Empaquetamiento del producto (Caja y CD's)

Para regular la elaboración de estos documentos y productos de trabajo, actualmente el curso cuenta con manuales para apoyar las actividades del Ingeniero de Requerimientos, de Diseño y de Pruebas. Estos manuales han sido construidos con un formato similar a los “How To”, en los que se describen las técnicas y metodologías más importantes que pueden ser utilizadas para llevar a cabo las actividades de cada rol, y están destinados a ser una ayuda-memoria sobre conceptos que deben haber sido adquiridos en cursos previos del área de IS. Además, cada manual incluye plantillas para los documentos de trabajo, acorde a los estándares IEEE para IS, y un “checklist” con el cual será revisado el documento. Estos “checklists” han permitido mejorar los resultados obtenidos del trabajo de cada rol.

En el semestre 98/2, se generó un manual para el Ingeniero de Requerimientos y solamente las plantillas para los documentos de trabajo del Ingeniero de Diseño y del Ingeniero de Pruebas. El manual de requerimientos contenía, inicialmente, descripciones básicas sobre cómo realizar la especificación de la arquitectura [3], del modelo de datos [3], de los DFD´s [9, 10, 8] y de los casos de uso [5, 6, 7]. La descripción incluía conceptos básicos de cada modelo y diagramas de ejemplo. En cuanto a las plantillas, el manual contenía una plantilla base para la Especificación de Requerimientos y plantillas anexas para análisis funcional, análisis orientado a objetos y una propuesta para análisis de aplicaciones web. Con ayuda del manual de requerimientos, los productos generados por los Ingenieros de Requerimientos fueron bastante más completos que los productos generados por el Ingeniero de Diseño y de Pruebas que, como se indicó anteriormente, disponían únicamente de las plantillas.

Cabe destacar que los alumnos no poseen experiencia de trabajo en ambientes normados o bajo un modelo de calidad de software (CMM [11] o ISO9001 [14]), por lo que la documentación generada ese semestre se ajustó literalmente a las plantillas entregadas, incorporando información irrelevante con la simple intención de satisfacer la plantilla. En el semestre 99/1, los manuales y plantillas de apoyo no fueron modificados y los resultados fueron similares.

Después de dos semestres de trabajo, fue necesario apoyar de mejor forma a cada rol y entregar pautas más completas para la realización de las actividades. Por esta razón, se generaron completamente los Manuales de Diseño y de Pruebas y se mejoró el Manual de Requerimientos. Actualmente, los manuales contienen descripciones y figuras de ejemplo de las técnicas y métodos que deben ser utilizados en cada actividad del ciclo de vida. Los contenidos principales de los manuales de requerimientos corresponden a técnicas y métodos de análisis y diseño estructurado y de análisis y diseño orientado a objetos. Para la especificación de sistemas orientados a objetos se ha adoptado el estándar UML [5, 6]. Además de lo anterior, se han generado guías para la descripción de contenidos que no necesariamente se abordan con las técnicas señaladas anteriormente. Ejemplos de éstas guías comprenden la descripción de reportes, del prototipo, de la lógica del negocio y, en el caso de las aplicaciones web, comprenden la generación de mapas de navegación lógicos y físicos y la descripción de páginas. Además de los manuales, se dejaron disponibles a los alumnos, las plantillas de los documentos con el formato requerido y bajo la herramienta estándar de documentación. Las plantillas no proveen ninguna característica especial y simplemente apoyan de mejor manera el concepto de un ambiente normado.

Con la ayuda de los manuales y el proceso de desarrollo aplicado el semestre 99/2, los productos generados fueron superiores a los entregados en los semestres anteriores. En particular, estos productos cumplieron de mejor manera las exigencias del curso en términos del formato, contenidos y objetivos planteados para el proyecto. Lo anterior permite establecer que, para apoyar las actividades de enseñanza en el área de Ingeniería de Software es fundamental entregar pautas y ejemplos del trabajo a realizar. Esto se debe principalmente al hecho de que la no aplicación de metodologías y técnicas de Ingeniería de Software no obedece necesariamente a problemas de capacidades y/o de formación, sino al desconocimiento completo de que éstas existen y los contextos en las cuales deben ser aplicadas.

El material de cada semestre está disponible como referencia para los alumnos del curso y no puede ser publicado directamente en el web debido a que contiene información sensible de las empresas participantes, sin embargo, en el futuro se podrían generar extractos con los elementos más importantes, permitiendo construir una biblioteca de componentes de Ingeniería de Software con patrones de análisis, de arquitecturas, de modelos de datos y de diseño entre otros.

En cuanto a las herramientas de software de apoyo al proceso, están: Ms-Project para la planificación de los proyectos y Ms-Office para la generación de documentación y presentaciones. Para apoyar el desarrollo del curso, se construyó un sitio web que permite la difusión de las actividades y la

publicación del material de apoyo y de la documentación de los proyectos. El sitio ha ayudado además, a la comunicación con los equipos de trabajo, sobretodo cuando sus integrantes trabajan en empresas de la industria gran parte de la jornada.

3.5. Esquema de Evaluación

Una de las tareas más difíciles ha sido la definición de un esquema de evaluación adecuado para los alumnos. En estricto rigor, el esquema de evaluación del desarrollo de un proyecto debiera considerar de algún modo, los mismos factores que intervienen en el proceso de desarrollo de software, es decir, factores del producto, del proceso, tecnológicos y humanos. Dentro de estos últimos no hay que olvidar los componentes emocionales y subjetivos. Sin intentar eliminar estos factores, que simplemente corresponden a los mismos con los que serán evaluados los alumnos en su vida profesional, se han intentado establecer criterios básicos para una evaluación lo más objetiva posible.

El esquema de evaluación debe permitir evaluar el trabajo individual, fomentar el trabajo en equipo y, por sobretodo, promover la satisfacción de las necesidades del cliente. Para promover la satisfacción del cliente, una vez que el producto ha sido entregado, el cliente realiza una evaluación formal del mismo equivalente al 50% de la nota final del equipo. Lo anterior ha permitido ubicar en niveles similares de exigencia la satisfacción del cliente y la del equipo docente. Esta evaluación contiene apreciaciones generales sobre la calidad de los productos, el compromiso mostrado por los alumnos, el cumplimiento de los objetivos y la formalidad de la relación, es decir, número de reuniones realizadas, minutas generadas, presentación de los resultados, etc.

En las evaluaciones del semestre 99/1, uno de los comentarios más relevantes mencionado por uno de los clientes, fue el hecho que los alumnos no habían sido capaces de generar un producto utilizable. El producto estaba bien construido, cumplía con los requerimientos mínimos y, sin embargo, "no era útil". Cuando el cliente estableció que el producto no era utilizable, quiso destacar el hecho que los alumnos habían centrado su atención en el producto, y no en la organización y los usuarios de la aplicación. Así, por ejemplo, la aplicación no contaba con un instalador ni manuales adecuados y no proveía una interfaz fácil de usar y de acuerdo con los estándares internos de la organización. Estas "omisiones" no son casuales y su ocurrencia se debe a la escasa empatía de los alumnos con el cliente y al poco sentido de exploración y de investigación. La mayoría de ellos no considera "clave" en el desarrollo del proyecto, la participación del usuario final, limitándose a la visión entregada por el interlocutor válido de la empresa, es decir, el Jefe de Proyecto asignado, quien por lo general no es el usuario final del producto de software. Este hecho gatillo en el equipo docente, la necesidad de sensibilizar más a los alumnos, en conceptos básicos de calidad de software que apuntan a satisfacer la totalidad de las necesidades del cliente [3, 4].

En el semestre 98/2, cada integrante de un equipo de desarrollo, obtenía una nota por cada documento de trabajo generado. Esta nota correspondía a una ponderación de algunos criterios como: cumplimiento de normas mínimas (ajuste a las plantillas en términos de contenido y de formato), alcance y contenidos de los documentos y utilización de técnicas adecuadas de ingeniería de software, entre otros. Los alumnos obtenían la nota y luego debían realizar las correcciones en los documentos, indicadas por el equipo docente. Este esquema, aplicado en los semestres 98/2 y 99/1, tuvo el inconveniente que sólo permitía evaluar el trabajo realizado, una única vez durante el desarrollo del proyecto. De esta manera, independiente de la nota obtenida, el desempeño correcto de los ingenieros en su rol, se veía claramente afectado una vez que entregaban los productos: el Ingeniero de Requerimientos realizaba su trabajo sólo al principio del semestre, el Ingeniero de Diseño durante todo el semestre, el Ingeniero de Construcción en las últimas semanas y, por último, el Ingeniero de Pruebas alcanzaba sólo a especificar y no a ejecutar las pruebas.

El mayor problema del esquema descrito es que de acuerdo al modelo de proceso utilizado, no existía una instancia válida para realizar un refinamiento de los productos e intentar aclarar conceptos generales sobre las metodologías y las técnicas aplicadas. Para lograr lo anterior, no bastaba con cambiar el esquema de evaluación incorporando revisiones parciales, sino que también, era necesario cambiar el modelo del proceso de desarrollo aplicado. En este sentido, si los alumnos visualizan el proceso de desarrollo como una serie de etapas secuenciales, no es factible pedir revisiones parciales del diseño si aún no se ha terminado la etapa de requerimientos, o revisar los productos del Ingeniero de Pruebas si no se ha terminado la actividad de Construcción.

En la versión del curso del semestre 99/2, como se mencionó anteriormente, se aplicó un proceso incremental. Acorde con la dinámica y velocidad de este proceso, se definieron entregas parciales de productos al final de cada ciclo. Al siguiente ciclo, la entrega debía incorporar las correcciones del ciclo anterior y el trabajo realizado en el ciclo. De esta forma, el trabajo de cada alumno en su rol es evaluado tres veces en el semestre, permitiendo no sólo la corrección de errores y la aclaración de conceptos mal aplicados, sino también refinar la documentación en términos de sus contenidos y alcance. Este nuevo esquema de evaluación ha permitido derivar y reconocer la importancia de un esfuerzo constante y homogéneo por parte del equipo de desarrollo, necesario para un proceso incremental. A continuación, se describe la fórmula aritmética para obtener la nota final de cada alumno como parte del esquema de evaluación.

Las notas individuales corresponden a la evaluación de los productos de trabajo entregados en los tres ciclos como se muestra a continuación:

$$\textit{Nota Individual Jefe de Proyecto} = (G_1 + G_2 + G_3) / 3$$
$$\textit{Nota Individual Requerimientos} = (R_1 + R_2 + R_3) / 3$$
$$\textit{Nota Individual Diseño} = (D_1 + D_2 + D_3) / 3$$
$$\textit{Nota Individual Construcción} = (C_1 + C_2 + C_3) / 3$$
$$\textit{Nota Individual Pruebas} = (P_1 + P_2 + P_3) / 3$$

La nota del proyecto contempla más variables como se describe a continuación:

$$\textit{Nota Entrega Parcial} = NEP_i = (G_i + R_i + D_i + C_i + P_i) / 4 \quad (1 \leq i \leq 3)$$
$$\textit{Nota Empaquetamiento Producto} = NEP$$
$$\textit{Nota Entrega Final} = NEF = (NEP_1 + NEP_2 + NEP_3 + NEP) / 4$$
$$\textit{Nota Evaluación Cliente} = NEC$$

Como muestra la fórmula, para incentivar el trabajo en equipo, se definió una nota para la entrega parcial de cada ciclo (NEP_i). Esta nota corresponde al promedio de las notas de cada uno de los roles para un ciclo determinado (*G=Gestión, R=Requerimientos, D=Diseño, C=Construcción y P=Pruebas*). De este modo, si un alumno entrega un producto de trabajo deficiente, no solamente perjudica la nota individual de su rol, sino que perjudica la nota del equipo. La nota del empaquetamiento del producto (*NEP*) corresponde a la evaluación de la entrega final ante el equipo docente. La entrega final contempla la generación de una caja con los productos mencionados en 3.4, que simule un producto final real. La nota de la entrega final (*NEF*) corresponde a la evaluación académica del proyecto. La nota de la evaluación del cliente (*NEC*) corresponde a la nota con la que el cliente califica el trabajo realizado.

Las notas finales del proyecto y de cada rol son las siguientes:

Nota Final de Proyecto = *NFP* = *NEF*50% + NEC*50%*
Nota Individual Rol = *NIR* = $(X_1 + X_2 + X_3) / 3$ ($X \in \{G,R,D,C,P\}$)

Por último, las condiciones de aprobación son las siguientes:

NFP ≥ 4.0 y *NIR* ≥ 4.0

No se puede establecer aún, si este esquema de evaluación permite fomentar de mejor manera el trabajo en equipo, sin embargo, se ha apreciado una mejora sustancial en los productos de trabajo entregados en los ciclos siguientes por aquellos alumnos que obtuvieron evaluaciones deficientes en los ciclos previos. Esto, indirectamente, contribuye a la obtención de mejores resultados finales y al éxito de los proyectos. Por el momento, sigue siendo una necesidad, identificar mecanismos para disminuir la subjetividad en las evaluaciones parciales. Por otro lado, el hecho de que la nota final del proyecto incorpore la nota de la evaluación del cliente con una ponderación del 50%, no ha causado en los alumnos el efecto deseado. La idea de incorporar esa ponderación a la nota de evaluación del cliente es básicamente para fomentar la satisfacción del mismo. Los alumnos han percibido esto como un riesgo importante pero, en términos generales, no han tomado las precauciones necesarias.

4. Conclusiones y Trabajo Futuro

Considerando que la motivación inicial para este trabajo fue acercar la enseñanza académica a la industria en el área de Ingeniería de Software, transcurridos ya tres semestres en los cuales se ha evolucionado el modelo de enseñanza-aprendizaje inicial implementado en el curso *Proyecto de Software*, este curso-laboratorio ha permitido, además, experimentar con modelos de desarrollo de software, en particular, en todas las actividades de Ingeniería, enfatizando y mejorando aquellas que consideramos fundamentales: requerimientos y diseño. En cuanto a las actividades de gestión, sólo se ha podido experimentar en las actividades de dirección y organización y en las siguientes versiones del curso se ampliarán los esfuerzos a las actividades de gestión más específicas como son la planificación, estimación, control y seguimiento de los proyectos.

Es clave destacar que la construcción de un modelo de enseñanza-aprendizaje para entrenar en la aplicación de conceptos de IS a través del desarrollo de un proyecto de software, requiere de un espacio de trabajo conjunto con empresas de la industria para conocer y vivenciar los problemas reales del desarrollo de software. De este modo, es posible validar y mejorar dicho modelo y asegurar que el proceso de enseñanza-aprendizaje sea exitoso.

4.1. Problemas encontrados

Entre los escollos más relevantes en el desarrollo del curso *Proyecto de Software*, se encuentran:

- La falta de compromiso de trabajo por parte de los equipos de desarrollo, originado, principalmente, por la jornada de ocupación laboral de los alumnos.
- Debido a que el régimen de cursos de la carrera es anual, la distancia entre lo aprendido y su puesta en práctica, genera grandes vacíos conceptuales en los alumnos.
- La duración del semestre académico (16 semanas), es un tiempo muy limitado para el entrenamiento dirigido en Ingeniería de Software

4.2. Resultados obtenidos

En relación a la actividad académica propiamente tal, este trabajo ha generado un ámbito de estudio sobre Ingeniería de Software, su aplicación y enseñanza. En cuanto a la interacción Universidad-Empresa, se ha logrado un mayor acercamiento de la Universidad hacia la empresa, más que de la empresa hacia la Universidad, con el fin de identificar mejor las necesidades de la industria.

En cuanto a los resultados obtenidos por la implementación de este modelo de enseñanza-aprendizaje, los principales han sido:

- De los doce proyectos desarrollados, seis terminaron exitosamente
- En cuanto al modelo del proceso de software utilizado, éste no ha sido comprendido a cabalidad por los alumnos y su definición y documentación no ha asegurado la aplicación correcta del mismo.
- Los documentos de trabajo no son valorados por los alumnos y sólo algunos clientes han reconocido su real importancia.

4.3. Trabajo Futuro

Después de realizar el curso tres semestres, las líneas de trabajo futuro que se visualizan son las siguientes:

- Definir una metodología para nivelar y actualizar conocimientos de IS en los alumnos, previo al inicio de los proyectos, con el fin de mejorar la productividad.
- Continuar con la definición y documentación del proceso de desarrollo de software, para cubrir las áreas claves de proceso estipuladas en CMM [11] (Nivel 2 y 3), e incorporar cambios acorde a las necesidades que vayan generando los nuevos proyectos. En esta línea de trabajo, se avanzará hacia la implementación de un Manual de Calidad del curso.
- Mejorar el esquema de evaluación académico con el fin de disminuir la subjetividad en las evaluaciones de los productos y en la del cliente. Para lograr esto, no basta con revisar la definición de las características esperadas para los productos, sino que se deberán estudiar esquemas de evaluación mediante puntajes y/o la formación de un equipo de evaluación, que no sólo incorpore muchos puntos de vista distintos, sino que asegure homogeneidad en los criterios.
- Definir un esquema de evaluación del modelo del curso, en cuanto a la estructura de la organización de desarrollo de software y el trabajo de equipo.
- Iniciar el proceso de medición de proyectos y generar un repositorio de métricas, para evaluar históricamente los proyectos realizados, y calcular métricas de productividad.
- Generar una biblioteca de componentes, con los productos de trabajo de los proyectos que abarquen desde mecanismos de representación, técnicas de documentación, patrones de diseño y modelos de datos, arquitecturas, entre otros. En el futuro, con ayuda de esta biblioteca, los alumnos dispondrán de un catálogo de componentes de software de manera de mejorar su productividad, por medio de su re-utilización. El concepto de componentes es mucho más amplio al usual y contempla todos aquellos productos que podrían ser re-utilizados.

5. Referencias

1. J.E. Tomayko, *"Teaching a Project-Intensive Introduction to Software Engineering"*, Technical Report CMU/SEI-87-TR-20, October 1996.
2. M. Shaw, J.E. Tomayko, *"Models for Undergraduate Project Courses in Software Engineering"*, Technical Report CMU/SEI-91-TR-10, August 1991.
3. R. S. Pressman, *"Software Engineering—A practitioner´s Approach"*, McGraw—Hill, New York, 1997.

4. R.J. Dawson, R.W. Newsham, *"Introducing Software Engineers to the Real World"*, IEEE Software, September 1997.
5. J. Rumbaugh, I. Jacobson, G. Booch, *"The Unified Modeling Language Reference Manual"*, Addison—Wesley, 1999.
6. M. Fowler, K. Scott, *"UML Distilled—Applying the Standard Object Modeling Language"*, Addison—Wesley, 1997.
7. I. Jacobson, *"Object-Oriented Software Engineering—A Use-Case Driven Approach"*, Addison-Wesley, 1992.
8. M. Page-Jones, *"The Practical Guide to Structured Systems Design"*, Prentice—Hall International, 1988.
9. T. DeMarco, *"Structured Analysis an System Specification"*, Yourdon Press, 1979.
10. E. Yourdon, *"Modern Structured Analysis"*, Yourdon Press, 1989.
11. M. Paulk, B. Curtis, M Beth Chrissis, Charles V. Weber, Technical Report CMU/SEI-93-TR-024 ESC-TR-93-177, *"Capability Maturity Model SM for Software, Version 1.1"*, 1993.
12. B. Boehm, *"A spiral model of software development and enhancement"*, IEEE Computer, pp. 61-72, May 1988.
13. B. W. Boehm, *"Software Engineering Economics"*, Prentice Hall, 1981.
14. International Standards Organization, ISO 9000-3, *"Guidelines for the Application of ISO9001 to the Development, Supply and Maintenance of Software"*, 1991.
15. C. Larman, *"Applying UML and Patterns. An Introduction to Object-Oriented Analysis and Design"*, Prentice Hall, 1997.