Educando hacia la Ingeniería del Software
Una reinterpretación para URUGUAY *

Ing. S/C Daniel Schwed†
Investigador visitante
Department of Information and Computer Science
University of California, Irvine, CA 92717, U.S.A.

Marzo de 1987

Dirección actual:
P.O. Box 6036
Montevideo
URUGUAY

1 Introducción

En este trabajo se analiza la situación pasada y actual de la educación en el área de Ingeniería del Software en el Uruguay[1]. Asimismo se plantean los objetivos futuros hacia los que se pretende dirigir la educación en esta área. Este trabajo resume experiencias del autor como docente del In.Co., y de docentes de la Universidad de California en Irvine.

2 Historia reciente

Las carreras de Ingeniero de Sistemas en Computación y Analista-Programador se encontraban desde el año 1974 sin modificación en su plan de estudios. Este plan de estudios

*Segun Peter Freeman en su presentación *Why Software Engineering is Important to Developing Countries* [Freeman77].

†En uso de licencia del Instituto de Computación, Facultad de Ingeniería, Montevideo, URUGUAY. Este tarbajo ha sido posible gracias a una beca brindada por la Comisión para el Intercambio Educacional entre Uruguay y Estados Unidos (Comisión Fulbright), y el apoyo del Advanced Software Engineering Project, Universidad de California – Irvine.

[1]El único centro de enseñanza de computación de nivel universitario del país, es la Facultad de Ingeniería (F.Ing.) dependiente de la Universidad de la República donde se dictan las carreras de Ingeniero de Sistemas en Computación y Analista-Programador. En la F.Ing. el Instituto de Computación (In.Co.) tiene a su cargo las labores docentes y académicas en el campo de la Informática. El In.Co. es uno de los 8 institutos que componen la Facultad de Ingeniería.

al que llamaremos Plan Viejo, estaba organizado hacia la creación de técnicos de nivel terciario.

La situación no era satisfactoria, como lo muestran los siguientes indicadores:

- La relación alumnos por docente era mayor a 130:1.
- La carga horaria por alumno era de 6.5 minutos docente/semana.
- Los alumnos inscriptos superaban los 4000, o sea mas del 50 % de los estudiantes de la Facultad de Ingeniería.
- Se disponía de solo 28 terminales para todos los alumnos.
- Creciente obsolecencia en el contenido de los cursos.

Como resultado del retorno al estado de derecho en 1985, y con el fin de tratar esta situación, se formó la Comisión Asesora del Area de Computación. Esta comisión fue designada por el Consejo de Facultad de Ingeniería [2] con el fin de llevar a cabo un replanteo de la situación en que se estaba y hacia qué metas se apuntaba.

Los primeros cambios que la Comisión logró introducir fueron:

- Revisión de los contenidos de cada materia.
- Llamados a nuevos cargos docentes.
- Redistribución de los escasos recursos existentes.

Surgió una propuesta docente de reestructura de la currícula, en dos etapas, bautizada Plan Turing 88 [PlanTuring]. Esta propuesta modifica fundamentalmente los contenidos de las materias (Plan de Emergencia) y propone una carrera sustitutiva (la que se definirá a partir del Plan Nuevo), renombrando el título final como "Ingeniero en Computación", y agregando el título intermedio "Analista en Computación", con una orientación más académica. En este momento el Plan de Emergencia se está poniendo en práctica y el Plan Nuevo se esta definiendo.

3 Estado Actual

Las primeras etapas del Plan de Emergencia fueron recientemente aceptadas por el Consejo de la Facultad de Ingeniería y entraron en vigencia en el 87. La puesta a punto de la nueva carrera (Plan Nuevo) será gradual, ya que los alumnos del Plan Viejo podrán terminar su carrera segun éste.

[2] El gobierno de la Universidad está representado por tres órdenes: docentes, estudiantes y egresados. Estos forman parte de los consejos de cada facultad y del consejo central y son elegidos democráticamente. Durante los años 74 al 84 por motivos políticos, el gobierno de la Universidad no se vio representado por los tres órdenes.

3.1 Ingeniería del Software

La idea de introducir el concepto de la Ingeniería del Software en nuestro instituto aparece durante 1985 (como resultado de la reestructura docente y los cambios que se produjeron en los contenidos de las materias indicados anteriormente). Se creó un Area de Ingeniería del Software dependiente del Departamento de Programación del In.Co. y se reestructuró la materia Procesamiento de Datos II (PDII), con esta orientación.

Esta reestructura del curso de PDII se basó en el seminario dictado por el Prof. Lavette C. Teague, [3] sobre "Análisis y diseño estructurados", utilizando bibliografía provista por éste durante su estadía en Montevideo [DeMarco78], [Page-Jones80], [Teague85].

El curso dictado se extendió anexando un taller realizado en grupos de 6 a 8 alumnos, a los cuales se les asignó, según sus intereses, un proyecto real, con usuarios reales (dependencias de la Universidad, Facultades, Ministerios, etc.).

Como resultado de este primer curso se observaron varios puntos:

- Los grupos numerosos de estudiantes (6 a 8) no funcionan bien, debido a problemas de comunicación, falta de definición de roles, incompatibilidad de horarios y diseño por compromiso (diseño por comité). La experiencia sirvió como ejemplo de lo pernicioso que puede ser trabajar en grupos grandes

- Es deseable que los proyectos sean reales. Los proyectos fueron positivos en el sentido de la experiencia ganada, el contacto con la realidad y con los problemas que generan los usuarios, lo que no se les hubiera podido transmitir de otra forma.

- Los usuarios reales tienen poca preocupación por los plazos académicos y se dieron casos en que desatendieron a los estudiantes.

- Los tiempos calculados para los proyectos fueron sub-estimados. El rendimiento de los alumnos fue menor al esperado y los usuarios contribuyeron en esto por lo indicado en el punto anterior.

- La teoría y la práctica deben ser combinadas en forma más gradual. Los cursos no deben tener una parte teórica y una parte práctica separadas.

Es de destacar que la cátedra debió enfrentar numerosos problemas, al disponer solamente de 3 docentes con un total de 33 horas/semana para atender 264 alumnos.

3.2 Formación y Competencia de un Ingeniero del Software

Esta sección revisa los objetivos que deberían alcanzarse a fin de lograr una adecuada formación de Ingenieros del Software. Fairley en [Fairley86], enumera la formación y competencia deseable y/o requerida por un Ingeniero del Software:

[3] California State Polythechnic University, profesor visitante bajo los auspicios del programa Fulbright.

- Análisis de requerimientos
- Especificación funcional
- Diseño del software
- Implementación
- Inspecciónes, revisiones y *walkthroughs*
- Pruebas de las unidades y *debugging*
- Integración y pruebas de aceptación
- Control de calidad
- Preparación de los Manuales del usuario y Guías de mantenimiento
- Mantenimiento del software
- Dirección de proyectos

A fin de lograr una buena educación en el área de Ingeniería del Software, se debe incluir conocimientos de las siguientes cinco disciplinas básicas: ciencias de la computación, administración, habilidades de comunicación, resolución de problemas generales, y diseño [Freeman77].

Es importante una discusión mas detallada de cada una de las disciplinas mencionadas.

1. Ciencias de la computación

   Las ciencias de la computación son la base del conocimiento para un Ingeniero del Software. Un Ingeniero del Software debe conocer:

   - *Matemática discreta*
     Algebra. Teoría de grafos. Lógica.
   - *Estructuras de datos*
     Estructuras recursivas. Algoritmos recursivos. Análisis de algoritmos.
   - *Arquitectura de sistemas*
     Anatomía de los computadores. Compresión de datos, codificación. Arquitecturas de alto nivel. Administración de procesos. Arquitectura del software de alto nivel. Sistemas distribuídos.
   - *Métodos formales*
     Pruebas. Verificación. Abstracción. Especificación. Teoría de Lenguajes Formales. Análisis.

- *Metodología de programación*
  Métodos de verificación. Inspecciones. Pruebas. Codificación. Técnicas, lenguajes, y métodos de diseño.

En resumen, un Ingeniero del Software debe tener una base sólida en ciencias de la computación, pues es en ésta que se basa toda su actividad. El conocimiento amplio de estos temas le permitirá desarollarse a medida que el campo de la Ingeniería del Software avance.

2. Administración

El objetivo principal de la Ingeniería del Software es desarrollar artefactos usables (en tiempo y forma). Los principales temas que deben considerarse son:

- Técnicas de evaluación de costos
- Presupuestación
- Análisis de los requerimientos del usuario
- Técnicas del tipo de PERT, CPM, programación lineal
- Organización teórica y práctica
- Psicología industrial
- Administración de personal

3. Habilidades de comunicación

Un Ingeniero del Software debe participar en la preparación de propuestas, descripción de los requerimientos del usuario, especificación de sistemas, reportes de estado, documentación de sistemas y reportes de mantenimiento.

Las formas de comunicación incluyen la comprensión y preparación de trabajos técnicos y reportes de proyectos. Estas deben ser escritas, orales, formales o informales.

Un Ingeniero del Software debe poder comunicarse tanto en su lengua madre como en cualquier lenguaje técnico que use, su formación debe apuntar a ello.

4. Resolución de problemas generales

Es obvio que a una persona no se le puede enseñar a resolver problemas en general, ya que en determinadas áreas no existe el sustituto de la experiencia. Pero se puede ver que un entrenamiento adecuado puede mejorar sensiblemente la respuesta del individuo.

Las áreas en las que se debe buscar familiaridad y dominio son:

- Formulación de problemas
- Fijación de objetivos
- Estrategias de solución
- Generación de ideas
- Implementación de las soluciones
- Validación de las soluciones

5. **Diseño**

   El diseño es una actividad se que aprende realizándola. No hay sustituto al contacto con el diseño. Obtener experiencia en diseño, es fundamental en la educación de un Ingeniero del Software.

   El énfasis en la teoría es valioso siempre y cuando no desplace otras necesidades básicas de la currícula. El diseño debe ocupar un papel preponderante en la educación de un Ingeniero del Software. Se sostiene que una condición *necesaria* para un Ingeniero del Software, es estar bien entrenado en las técnicas actuales de programación y tener una considerable experiencia en ellas, pero esto no es *suficiente* para hacerlo un buen diseñador [Freeman80].

4 Soluciones a corto plazo

4.1 Prerequisitos.

El plan de estudios vigente contiene una buena raíz para la formación de Ingenieros del Software. Entre los puntos antes mencionados se cubren las áreas de ciencias de la computación y administración en detalle suficiente. Las modificaciones en estas áreas no implicarán cambios sustanciales. Sin embargo en las áreas de comunicación, resolución de problemas, y diseño hay mucho para hacer.

Para ello se deberán procurar cambios en los contenidos de las materias ya existentes, basándose en el concepto de libertad de cátedra (una reestructura de las materias implicaría una modificación tal en la currícula, que debería ser considerada por el Claustro de la F.Ing., lo que tomaría por lo menos seis meses). La Comisión Asesora del Area de Computación ha recomendado a los docentes de las materias relacionadas a los puntos antes mencionados, las modificaciones imprescindibles. La labor de la Comisión será guiar el cambio en forma conceptual y global.

4.2 Modificación del contenido de PDII.

Las modificaciones del programa de PDII ya fueron realizadas. El contenido de la materia es coherente con los objetivos planteados, pero se debe mejorar la forma en que se dicta.

La práctica debe ser revista e integrada a la teoría. Tal vez se deba considerar la inclusión de proyectos de sistemas en forma de competencia o una mayor interrelación entre los grupos de trabajo [Horning77], [Freeman76].

Los plazos de los proyectos deberán ser estrictos, y a pesar de ser deseable no se podrá depender de usuarios externos (ver punto 3.1), hasta no disponer de un plantel docente con mayor carga horaria, que le permita actuar como *buffer* entre los estudiantes y los usuarios.

Se deberá hacer un mayor hincapié en el concepto de diseño, apuntando a ampliar la experiencia de los estudiantes.

5 Soluciones a largo plazo

Las soluciones a largo plazo se basan en las propuestas del Plan Nuevo, (actualmente en estudio) que se pondrá en práctica en forma gradual, dentro de los próximos dos o tres años, luego de ser aprobado por el Claustro.

Es necesario incluir e integrar los conceptos de Ingeniería del Software al Plan Nuevo. Es recomendable preparar a los estudiantes intermedios con un claro concepto de la Ingeniería del Software [Freeman86]. Hay un numero de técnicas y conceptos de la Ingeniería del Software que pueden ser asimilados en las primeras etapas de formación universitaria. La mayor parte de los estudiantes que obtendrán el título intermedio se integrarán al mercado laboral en ese momento (esta idea esta basada en la situación actual que muestra una fuerte tendencia del alumnado a obtener solamente el título de Analista-Programador). Podríamos de esta forma, mejorar significantemente el nivel general de competencia y asimismo proveer una mejor base para la obtención del título final de Ingeniero en Computación.

Deberá permitirse, por medio de cadenas de materias opcionales, que el alumno se decida por la Ingeniería del Software u otra especialidad en los años finales de la carrera.

5.1 Talleres

Es de destacar la separación existente entre los objetivos de una formación en Ingeniería del Software que apunta a la práctica, con la de una orientación teórica pura.

La idea del trabajo en talleres en el Plan Nuevo está orientada hacia lograr un mayor y mejor contacto del alumno con la realidad. Esta idea, buena en su concepción, se ve empañada por el hecho que el ambiente universitario tiene diferencias muy marcadas con el mundo real, como se puede observar en la siguiente tabla [Riddle86] :

Diferencias entre los *Proyectos Educativos* y los del *Mundo Real.*

	Mundo Real	*Educación*
Tamaño	$>$ 10.000 lineas	$<$ 500 lineas
Ciclo de vida	10-15 años	"hasta que corra"
Complejidad	multiples módulos	un módulo
Desarollo	usuario $\neq$ desarrollo	usuario = desarrollo
Mantenimiento	extenso	descartable

La artificialidad de un entorno educativo (tradicional) no aporta un concepto de realidad al alumno. Como destacable se puede mencionar que el alumno no se ve enfrentado a los volúmenes de trabajo que se dan en el mundo real [Shaw76]. El ambiente educativo debe ser contrastado con la realidad, tomádose en cuenta los siguientes problemas:

- Compromisos a corto plazo y part-time de los alumnos
- Muchos estudiantes trabajando en el mismo problema
- La ausencia del usuario final y de continuidad del proyecto
- Falta de una contabilidad efectiva de recursos
- Las tensiones inherentes a un ambiente educativo

Existen distintas alternativas para organizar prácticas que sirven como aproximaciones útiles del mundo real y que proveen experiencias fundamentales en la formación en Ingeniería del Softawre. Las Empresas de Software ("Software Hut") [Horning77], por ejemplo, permiten que varios grupos compitan en un ambiente simulado de venta y soporte de productos desarrollados por ellos. Otros enfoques, [Freeman76], estan basados en el intercambio de productos entre grupos, por ejemplo, un grupo A especifica un sistema, un grupo B lo implementa y luego A lo mantiene.

No debe descartarse totalmente el trabajar con usuarios reales. Esta idea se puede llevar a cabo, disponiendo los docentes de una carga horaria suficiente, con los alumnos más avanzados de la carrera.

5.2 Nuevos cursos

Estos cursos estarán basados en una bibliografía ordenada y orientada hacia un nivel universitario [Fairley85a], [Fairley85b], [Gerhart86], [Pressman82].

En estos cursos se deberán introducir las ideas indicadas en el punto 3.2, dentro de las cuales se pueden destacar:

- El uso de formalismos en Ingeniería del Software.
- El diseño como actividad fundamental.
- La exploración de la idea de ciclo de vida en sus distintos paradigmas.
- La automatización del proceso del software.
- La formalización del conocimiento del proceso del software.

La inclusión de las cadenas de materias opcionales anteriormente mencionadas permitirán al alumno una selección y balanceo según sus intereses y aspiraciones.

5.3 *Bootstraping* de la idea de Ingeniería del Software.

La clave para lograr una buena currícula es establecer un híbrido entre el taller de artes, donde el alumno desarrolla su capacidad creativa, y el equivalente a un hospital de enseñanza, donde los médicos en formación (internos) practican sobre la realidad bajo la supervisión de expertos. Para esto es nesesario organizar laboratorios de desarrollo de software donde se logre la combinación de educación en forma de clases teóricas y el desarrollo de las habilidades prácticas del estudiante.

Estos laboratorios deberán permitir tanto la creación y desarrollo de métodos, herramientas y entornos de diseño y/o programación adaptados a la realidad y necesidades del país, como el desarrollo profesional de los alumnos. Bajo la supervisión de docentes y equipados con técnicas y herramientas actualizadas, los laboratorios permitirán al alumno ganar experiencias irremplazables [Basili86].

Asimismo esto podrá crear una fuente de ingresos por medio de contratos de desarrollo con empresas y organismos nacionales e internacionales. Un modelo de este tipo de cooperación ha sido implementado en California – MICRO program[4]. En el contexto de MICRO, los investigadores universitarios proponen proyectos de investigación aplicada. El costo del proyecto es subvencionado en un 50% por compañías interesadas y la Universidad de California. Debe aclararse que "costo" comprende todos los recursos del proyecto, incluyendo sueldos de investigadores, costo de hardware, software, etc. En nuestro país, dadas las limitaciones presupuestarias de la Universidad, las empresas podrían aportar un 100% de los costos, y el In.Co. como contrapartida contribuiría con los docentes y/o alumnos (que trabajarían en estos proyectos para aprobar sus cursos). Estos ingresos se podrían usar para beneficio del In.Co. en la compra de nuevos equipos, software, herramientas o financiando viajes de educación, seminarios dictados por profesores invitados, etc.

[4]Microelectronics Innovation and Computer Research Opportunities.

6 Resumen

De lo expuesto anteriormente, a fin de lograr una buena educación en el área de la Ingeniería del Software, se debe:

- Incluir e integrar los conceptos de Ingeniería del Software en todos los niveles del Plan Nuevo.
- Realizar talleres que apunten al contacto con el mundo real, con suficiente dedicación docente.
- Crear cadenas de materias opcionales que permitan la especialización coherente del alumno.
- Investigar nuevos paradigmas en el área de Ingeniería del Software.
- Generar herramientas que se adapten a los requerimientos del país.
- Obtener financiamiento a través de contratos de desarrollo con empresas y organismos nacionales e internacionales.

Estos puntos permitirán acercarnos a las metas a las que apunta el Instituto de Computación: la formación de profesionales y la creación de un buen ambiente académico.

[Basili86] Basili, Victor R., *The Experimental Aspects of a Professional Degree in Software Engineering,* Software Engineering Education: The Educational Needs of the Software Community, Springer-Verlag, Junio 1986.

[DeMarco78] DeMarco, Tom, *Structured Analysis and System Specification,* Yourdon Press, 1978.

[Fairley85a] Fairley, Richard, *Software Engineering Concepts,* McGraw-Hill, 1985.

[Fairley85b] Fairley, Richard, *Core Course Documentation, Master's Degree Program in Software Engineering,* Technical Report TR-85-17, Wang Institute of Graduate Studies, Setiembre 1985.

[Fairley86] Fairley, Richard, *Software Engineering Education: An Idealized Scenario,* Software Engineering Education: The Educational Needs of the Software Community, Springer-Verlag, Junio 1986.

[Freeman76] Freeman, Peter, *Realism, Style and Design: Packing it into a constrained course,* ACM SIGCSE Bulletin, vol. 8, num. 1, Febrero 1976.

[Freeman77] Freeman, Peter, *Why Software Engineering is Important to Developing Countries,* Proceedings of the International Conference on Computer Aplications, Bangkok, pp. 1489-1502, Agosto 1977.

[Freeman80] Freeman, Peter, *The Central Role of Design in Software Engineering,* Software Engineering: Research Directions, Academic Press, 1980.

[Freeman86] Freeman, Peter, *Essential Elements of Software Engineering Education Revisited,* Software Engineering Education: The Educational Needs of the Software Community, Springer-Verlag, Junio 1986.

[Gerhart86] Gerhart, Susan L., *Skills versus Knowledge in Software Engineering Education: A Retrospective on the Wang Institute MSE Program,* MCC Technical Report STR-202-86, Junio 1986.

[Horning77] Horning, J.J. and Wortman, D.B., *Software Hut: a Computer Program Engineering Project in the Form of a Game,* IEEE Transactions on Software Engineering, Vol. 3, No. 4, Julio 1977.

[Page-Jones80] Page-Jones, Meilir, *The Practical Guide to Structured Systems Design,* Yourdon Press, 1980.

[PlanTuring] Instituto de Computación, *Plan Turing/88,* Documento interno del In.Co., Agosto 1986.

[Pressman82] Pressman, Roger, *Software Engineering: A Practitioner's Approach,* McGraw-Hill, 1982.

[Riddle86] Riddle, William and Williams, Lloyd , *Technology Selection Education for Software Engineers,* Software Engineering Education: The Educational Needs of the Software Community, Springer-Verlag, Junio 1986.

[Shaw76] Shaw, Mary., *Making Software Engineering Issues Real to Undergraduates,* Software Engineering Education: Needs and Objectives, Springer-Verlag, 1976.

[Teague85] Teague, Lavette and Pidgeon, Christophrer, *Structured Analisis Methods for Computer Information Systems,* Science Research Associates Inc., 1985.