Revista Antioqueña de las
Ciencias Computacionales y la Ingeniería de Software

ISSN: 2248-7441

www.fundacioniai.org/raccis/index.htm
raccis@fundacion.org

"Software Engineering" is Engineering

La "Ingeniería de Software" es Ingeniería

Edgar Serna M.
Instituto Antioqueño de Investigación IAI. Medellín, Colombia. *eserna@eserna.com*

INFORMACIÓN DEL ARTÍCULO

Tipo de artículo
Reflexión

Historia del artículo
Recibido: 17-09-2011
Correcciones:
Aceptado: 29-11-2011

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education – *Curriculum*.

General Terms
Computer Science, Software Engineering, Logic or Abstraction.

Keywords
Computer Science, Engineering, Software Engineering, Software Development, Accreditation.

Palabras clave
Ciencias Computacionales, Ingeniería, Ingeniería de Software, Desarrollo de Software, Acreditación.

ABSTRACT
Determine that "Software Engineering" is engineering, is an issue that generates discussion in many universities in which, by decades, computer science programs have used the term to describe individual courses and claim it as part of their discipline. However, some engineering faculties demand it as a new and necessary specialty between traditional engineering disciplines. This article analyzed the differences between Computer Science and "Software Engineering" as engineering and is argued the necessity of a program in this last that follow the traditional engineering approach looking for a professional formation of software engineers.

RESUMEN
Determinar que la "Ingeniería de Software" es ingeniería es una cuestión que genera discusión en muchas universidades en las que, por décadas, los programas de Ciencias Computacionales han utilizado el término para describir cursos individuales y lo reclaman como parte de su disciplina. Sin embargo, algunas facultades de ingeniería lo demandan como una especialidad nueva y necesaria entre las disciplinas tradicionales de la ingeniería. En este artículo se analizan las diferencias entre las Ciencias Computacionales y la "Ingeniería de Software" como ingeniería y se argumenta la necesidad de un programa en esta última que siga el enfoque de las ingenierías tradicionales, en busca de una formación profesional de ingenieros de software.

1. INTRODUCCIÓN

Desde 1967, cuando en Alemania un grupo de profesionales provenientes de variadas disciplinas –la mayoría serían hoy identificados como científicos computacionales– se reunió para discutir acerca de la "Ingeniería de Software", los científicos computacionales sostienen que esa especialidad es una sub-área de las Ciencias Computacionales. Las facultades de ingeniería cuentan con personas especializadas en teoría de autómatas, diseño de lenguajes, sistemas operativos, demostración de teoremas, Ingeniería de Software y muchas otras áreas y los estudiantes toman cursos en una variedad de temas como compiladores, bases de datos y, también, en Ingeniería de Software que, generalmente, es un curso al que se coloca dicho nombre sólo porque sí, aunque a veces se ofrezca con extensiones caprichosas como " Orientada a Objetos" o "Basada en Componentes".

En este artículo se toma un punto de vista diferente y se enmarca a la Ingeniería de Software como un elemento del conjunto I = {Ingeniería Civil, Ingeniería Mecánica, Ingeniería Química, Ingeniería Eléctrica, Ingeniería de Software,…} y no como una sub-área de las Ciencias Computacionales. Esto no es simplemente un juego de taxonomía académica en el que se argumenta acerca de la historia o propiedades del área, la cuestión importante es el contenido y el estilo de la formación. Los programas universitarios en ingeniería son muy diferentes de los programas en ciencias, matemáticas o artes liberales. Las disparidades se derivan de las diferencias en los objetivos profesionales y los intereses de los estudiantes, muchos de los cuales se orientan por las programas en desarrollo de software y, posiblemente, aplicando un estilo de formación ingenieril, se formarían mejor de lo que les ofrece un programa de Ciencias Computacionales.

Es importante subrayar que no se trata de comparar dos áreas de la ciencia pero, del mismo modo que la Física es la base científica de la Ingeniería Eléctrica, las Ciencia Computacional son la base científica de la Ingeniería de

Software. Los intentos por distinguir entre estos dos cuerpos separados del conocimiento puede dar lugar a confusiones, de lo que se trata es de contrastar la formación en una ciencia con la formación en una disciplina de ingeniería basada en la ciencia misma y de reconocer que ambos programas pueden compartir mucho de su núcleo principal, lo que ayudará a comprender sus diferencias reales.

2. LA NECESIDAD DE UNA INGENIERÍA DE SOFTWARE

Los ingenieros son profesionales cuya formación los prepara para usar las matemáticas, la ciencia y la tecnología para crear productos y proponer soluciones a problemas sociales importantes. Debido a la variedad y complejidad de los problemas actuales y a que una sola área de la ciencia no puede brindar el conocimiento suficiente para solucionarlos, la ingeniería se ha dividido en especialidades diferentes que se centran en un tipo específico de problemas y de productos. Por ejemplo, los ingenieros civiles se especializan en estructuras físicas como carreteras, puentes y edificios; los ingenieros químicos en el diseño de plantas y procesos de fabricación para la industria química; los ingenieros eléctricos en sistemas de potencia, electrónica y dispositivos de comunicaciones y así sucesivamente.

Actualmente, cada vez es más frecuente encontrar que el software es un componente importante en una amplia variedad de productos, incluyendo muchos de las ingenierías tradicionales. Además, los ingenieros lo utilizan para diseñar otros productos no computarizados, cuya efectividad depende en parte de la correctitud del software utilizado. También hoy, los científicos computacionales dedican mucho esfuerzo al estudio de los computadores y de la creación de programas, por lo que se conoce mucho más acerca de la computación y del desarrollo de software de lo que se conocía en 1967. Gran parte de la formación que reciben los desarrolladores de software hoy, es conocimiento que no conocían los que se reunieron en las conferencias originales de Ingeniería de Software entre 1967 y 1969.

El incremento de la importancia y la complejidad del software, combinado con el incremento en el conocimiento acerca de cómo "construirlo", se traduce en una necesidad para algunos profesionales que, como los ingenieros, reciben una formación centrada en cómo diseñar y fabricar productos fiables pero poco especializada en el diseño, la construcción, las pruebas y el mantenimiento de productos software, por lo que ya no es posible "aprovechar" lo que han recibido en algunos cursos de desarrollo, impartidos en programas tradicionales de ingeniería o en programas de Ciencias Computacionales [1].

El enfoque actual de la formación para profesionalizar el desarrollo de software no es satisfactorio para nadie. Mientras muchos consumidores denuncian la mala calidad de los productos software, las empresas desarrolladoras sufren por la escasez de personal altamente cualificado [2]. Además, muchos empresarios no saben, o confunden, lo que un graduado de un programa en Ciencias Computacionales o de una ingeniería tradicional puede conocer acerca del desarrollo de software. Para cada disciplina de ingeniería existe un "cuerpo base del conocimiento" bien documentado, lo que que no existe para las Ciencias Computacionales. Por esto, no es posible identificar un componente o una técnica individual de conocimiento que se aplique siempre en todos sus programas [3]. La introducción de programas profesionales acreditados, que sigan el modelo de los programas en disciplinas de ingeniería tradicionales, contribuirá a incrementar la calidad y la cantidad de graduados bien preparados para desarrollar productos software confiables. Además, así como la introducción de programas en Ingeniería Eléctrica no le quitó importancia a la Física, los programas en Ingeniería de Software no le quitarán importancia a las Ciencias Computacionales, aunque deba modificar su enfoque formativo. De hecho, la introducción de programas en Ingeniería de Software permitirá el desarrollo de programas en Ciencias Computacionales cuyos egresados estarán mejor calificados que los actuales.

3. LOS "INGENIEROS DE SOFTWARE" SON MÁS QUE "PROGRAMADORES DE SOFTWARE"

En muchos escenarios como las ofertas de empleo, se utiliza el término "Ingeniero de Software" como un eufemismo para "programador". Muchos parecen asumir que la única responsabilidad de un Ingeniero de Software es escribir código. Estos supuestos reflejan ignorancia acerca del significado histórico y jurídico de ser ingeniero. Un Ingeniero es un profesional responsable de construir productos aptos para su consumo y, para lograrlo, debe comprender ampliamente el entorno en el que cada producto será utilizado. Consecuentemente, los Ingenieros de Software necesitan conocer muchas cosas que no hacen parte de las Ciencias Computacionales. El software no se utiliza aisladamente de otros productos de ingeniería; hace parte de sistemas que contienen componentes físicos y se aplica para computar información acerca de esos sistemas. Entre los objetivos de los programas en Ingeniería de Software se destacan: (1) Desarrollar la capacidad lógico-abstractiva e interpretativa de los estudiantes para comprender los problemas, antes de presentar una solución, (2) Especializar a sus graduados en el diseño y desarrollo de software con calidad y (3) Dotar a sus estudiantes con conocimiento suficiente acerca de otras áreas de las ciencias, que los capacite para poder solicitar la ayuda de otros ingenieros o profesionales para conformar y trabajar en equipos armónicos [4].

3.1 *Una analogía histórica*

Ya ha sucedido antes: cuando un área de las ciencias madura, las instituciones formativas desarrollan programas de ingeniería con base en esa área. Por ejemplo, cuando se logró comprender la física de los campos magnéticos y eléctricos se desarrolló una nueva especialidad, la Ingeniería Eléctrica, al tiempo que también lo hicieron los programas formativos correspondientes. Aunque se escucharon reclamos de algunos físicos quienes afirmaban que esa área pertenecía a la física y que desde ella se podía formar en programas

de física aplicada, la mayoría de universidades desarrollaron programas en Ingeniería Eléctrica, que hoy conviven junto a programas de física y a los programas previamente existentes en otras ingenierías.

Preguntas como las que ahora se hacen se formularon entonces, por ejemplo, si la Ingeniería Eléctrica se basa en la física ¿por qué se necesitan dos programas? ¿Por qué los estudiantes de Ingeniería Eléctrica no sólo estudian física? Está claro que los dos programas son necesarios, no porque existan dos áreas involucradas de la ciencia, sino porque son dos profesiones con caminos muy diferentes: una pretende graduar profesionales con la función de diseñar productos para que otros puedan usarlos, y la otra para que estudien los fenómenos que interesan a ambos grupos y para ampliar el conocimiento en esa área. No se sugiere que las Ciencias Computacionales se conviertan exclusivamente en teóricas o que los ingenieros de software nunca harán trabajos matemáticos; los físicos todavía construyen cosas y muchos ingenieros eléctricos publican artículos de matemática compleja. Más bien, es una cuestión de objetivos: de los físicos se espera principalmente, y están capacitados para ello, que profundicen en el conocimiento, mientras que de los ingenieros eléctricos se espera que desarrollen productos o técnicas para reproducirlo. El camino de cada programa atrae a un tipo diferente de estudiantes y requiere un programa formativo diferente. La mayoría de estudiantes deciden estudiar Ingeniería Eléctrica en lugar de Física, porque les gusta construir cosas; los que estudian Física a menudo están más entusiasmados por el aprendizaje que por construir. Se pueden enumerar excepciones a estas reglas, pero lo que no se puede hacer es diseñar programas formativos para cada excepción. Además, existen muchas oportunidades para que las personas se desvíen de sus planes originales, en la medida que avanzan en sus procesos formativos.

Cuando el polvo se asiente y existan dos programas diferentes de formación, estables y funcionales, se entenderá que las Ciencias Computacionales y la Ingeniería de Software son complementarias y que cooperan en muchos puntos, de la misma forma que las facultades de ciencia e ingeniería lo hacen. También, que los estudiantes pueden cambiar entre una u otra de acuerdo con sus necesidades e inclinaciones formativas.

3.2 *Formación en Ciencia vs formación en Ingeniería*

Aunque pocas personas se molestan en comparar y contrastar, la formación en ciencias es muy diferente a la que reciben los estudiantes de ingeniería. Estas diferencias no son accidentales, se basan en necesidades diferentes de dos programas muy diferentes.

Los futuros científicos, los que desean ampliar el conocimiento, se forman en:

- Lo que es cierto. Un cuerpo de conocimiento organizado acerca de los fenómenos de su interés.
- Cómo confirmar o refutar modelos del mundo.
- Cómo ampliar el conocimiento de lo que es cierto en su campo.

En otras palabras, los científicos se forman en ciencia además de en los métodos científicos necesarios para ampliarla. Por lado, los futuros ingenieros, los que diseñan productos confiables, se forman en:

- Lo que es cierto y útil en la especialidad elegida. El cuerpo organizado de conocimientos.
- Cómo aplicar ese conjunto de conocimientos.
- Cómo aplicar un área más amplia de conocimientos, necesarios para construir productos completos que deben funcionar en entornos reales.
- En el diseño y el análisis disciplinar que deben seguir para cumplir con las responsabilidades que les incumbe a quienes construyen productos para otros.

En otras palabras, los ingenieros se forman en ciencia además de en los métodos ingenieriles necesarios para aplicarla.

Para los estudiantes de ciencias es esencial mantenerse al día acerca de las investigaciones recientes en su especialidad, pero corren el riego de que el conocimiento científico de alguna investigación pueda estar muy enfocado. En cambio, para los ingenieros es importante tener un conocimiento muy amplio pero, en la mayoría de los casos, basta que estén conscientes solamente del conocimiento científico y de las tecnologías que demuestran ser fiables y eficaces en sus aplicaciones.

Una ilustración de la diferencia en este énfasis la proporciona la ya mencionada disputa entre los ingenieros eléctricos y las facultades de Física, en los últimos 60 años. Algunas facultades querían revisar el curso compartido de física reduciendo la cobertura de los campos eléctricos y magnéticos, de modo que se pudiera añadir una sección con los resultados de las últimas investigaciones acerca de las partículas atómicas. Las facultades de Ingeniería Eléctrica se opusieron, señalando que sus estudiantes no necesitaban conocer acerca de las nuevas partículas sino de lo competente al diseño de dispositivos eléctricos y electrónicos. Ambas facultades pensaron correctamente acerca de las necesidades de sus propios estudiantes, pero quedó claro que las necesidades de los estudiantes de ingeniería y de los de Física no eran las mismas.

Estas diferencias tienen sentido debido a lo siguiente:

- Si la idea es realizar una investigación especializada, extendiendo la ciencia, puede darse el lujo de ser reducida pero no permitirse el incumplimiento de las fechas. Si se aplica ciencia para construir productos fiables, rara vez se necesitan los resultados de las más recientes investigaciones científicas, pero se debe tener una amplia comprensión, y ser conscientes, de los muchos factores que se deben tener en cuenta al diseñar un producto [5].

- Si se lleva a cabo una investigación científica, se puede esperar que los resultados sean verificados por quienes leen y arbitran los reportes y artículos. Sin embargo, como señalan DeMillo et al [6], el diseño de un producto no es posible someterlo al mimo tipo de control. Los maestros en Ingeniería subrayan que los ingenieros deben aceptar la responsabilidad por la fiabilidad de sus propios diseños.

- Los científicos frecuentemente trabajan en problemas o en equipos reducidos; los ingenieros, a menudo, toman la responsabilidad de diseñar completamente un producto, lo que significa que requieren de amplios conocimientos por fuera de su especialidad ingenieril. Un ingeniero mecánico podría tener que hacer un diseño de energía eléctrica, o un ingeniero eléctrico tener que mirar los aspectos mecánicos de un motor o un servomecanismo. Los programas tradicionales de ingeniería no se diseñan teniendo esto en mente. Un ingeniero requiere saber cuándo hablar con otros ingenieros y saber lo suficiente para comunicarse fácilmente con los que tienen otras especialidades [7].

Por supuesto, la disponibilidad de ingenieros de software bien formados no eliminará la necesidad de científicos computacionales. Aunque existe una gran necesidad de ingenieros especializados en diseño de software, este campo es muy joven, también hay una demanda por personas que experimenten con las herramientas y los métodos que utilizan esos ingenieros y para extender el conocimiento acerca de cómo diseñar sistemas informáticos. Muchas cuestiones quedan abiertas y su estudio requiere de científicos bien formados, no de ingenieros.

3.3 *El papel de la acreditación en ingeniería*

Por lo general, el trabajo de los científicos es juzgado por otros científicos, pero los ingenieros suelen tratar directamente con los clientes, que ni son ingenieros ni son científicos. Así, mientras nadie ha considerado necesario mantener los programas científicos bajo rígidos estándares, la acreditación ha sido siempre una consideración importante para los programas de ingeniería. En Canadá y EE.UU. la legislación limita a quienes pueden ejercer la ingeniería sólo a los acreditados por las sociedades de ingeniería profesional designadas. En estos países, esas organizaciones se unieron para crear planes de acreditación a los programas universitarios de ingeniería. Las licencias son emitidas principalmente a aquellos que se gradúan de programas acreditados, pero quienes obtienen de otra forma sus conocimientos pueden solicitar un examen individual y, si logran pasarlo, también son acreditados [8].

Los programas de ciencia están sujetos a revisión por las universidades que los ofrecen, porque las buenas instituciones siempre están preocupadas por la calidad de sus ofertas; pero, para estos programas no existe algo semejante a la acreditación del Canadian Engineering Accreditation Board o del Accrediation Board for Engineering and Technology. La Canadian Information Processing Society ofrece un programa de acreditación voluntaria, pero la documentación describe explícitamente que para los programas estatales "no existen normas rígidas". Muchas de las más importantes facultadas en Ciencias Computacionales no se interesan en la acreditación y, algunas de las que lo hacen, no la toman en serio. Por el contrario, los requisitos del Connecticut Energy Advisory Board CEAB y Adult Basic Education and Training ABET son bastante rígidos y las visitas de acreditación son eventos importantes para todas las facultades de ingeniería. Debido a que el uso del título de “Ingeniero” es restringido por la ley, los estándares son exigentes y se toman seriamente por quienes ofrecen programas de ingeniería [2].

Los procesos de acreditación para programas de ingeniería son muy eficaces para elevar la calidad de los programas formativos, para asegurar que todos los graduados se exponen a las ideas más importantes y para demostrar que saben cómo aplicarlas. La Ingeniería de Software no alcanzará el estatus de verdadera profesión hasta que tenga un sistema de acreditación similar. La forma más fácil y el mejor camino para establecer dicho sistema es tratarla como a cualquiera otra especialidad dentro de la ingeniería.

4. CÓMO FORMAR INGENIEROS DE SOFTWARE

El primer paso en el desarrollo de la Ingeniería de Software será hacer lo que se ha hecho en otras disciplinas ingenieriles: identificar un cuerpo de conocimiento base. Este proceso debe comenzar con una descripción de las tareas que sus graduados deben ser capaces de realizar. Los siguientes son los pasos clave en el desarrollo de sistemas informáticos y, en cada uno de ellos, los ingenieros de software deben estar preparados para participar [9]:

- Analizar y comprender el problema para determinar los requisitos que debe satisfacer una posible solución y registrarlos de forma precisa y bien organizada en un documento que sea fácil de comprender y utilizar.

- Participar en el diseño de la configuración del sistema informático, determinando cuáles funciones se implementarán en el hardware y cuáles en el software y seleccionando el hardware y los componentes software básicos.

- Validar y verificar el desempeño del diseño propuesto, sea analíticamente o por simulación, para asegurar que el sistema cumple con los requisitos de la aplicación.

- Diseñar la estructura básica del software, es decir, dividirlo en módulos, diseñar las interfaces entre ellos y la estructura de los programas individuales mientras documenta con precisión todas las decisiones de diseño.

- Implementar el software de la solución como un conjunto de programas bien estructurados y documentados.

- Analizar la estructura del software en cuanto a completitud, consistencia e idoneidad para la aplicación propuesta.

- Integrar el nuevo software al existente o instalarlo de forma independiente.

- Realizar pruebas sistemáticas y estáticas al software y al sistema informático integrado.

- Revisar y mejorar los sistemas software manteniendo su integridad conceptual y preservando todos los documentos completos y precisos.

Al igual que todos los ingenieros, los de software son responsables por la usabilidad, seguridad y fiabilidad de sus productos. Deben ser capaces de aplicar las matemáticas, la lógica y las ciencias, incluidas las computacionales, para asegurar que el sistema que diseñan realizará sus funciones cuando se entregue al cliente.

Muchos científicos computacionales, cuya especialidad es la Ingeniería de Software, podrán añadir otras cosas a esta lista, por ejemplo, que los ingenieros de software deben conocer cómo trabajar en equipo, hacer cronogramas, establecer plazos, calcular costos y funciones de gestión de proyectos. En algunas instituciones denominan a los cursos de gestión de proyectos como de Ingeniería de Software, sin embargo, aunque estas actividades podrían diferenciar el trabajo de los desarrolladores de software del de los académicos, no diferencian el trabajo de los ingenieros de software del de otros ingenieros. Algunos cursos acerca de gestión de proyectos pertenecen al núcleo de todos los programas de ingeniería y los pueden tomar quienes estén interesados especialmente en la gestión ingenieril.

4.1 *Diferencias entre Ingeniería de Software y Ciencias Computacionales*

Muchos maestros en las facultades de ingeniería creen que realmente forman en Ingeniería de Software. Inclusive, algunas facultades afirman que no hay necesidad de un nuevo programa y que pueden ofrecer un pregrado o especialización en Ingeniería de Software al interior de las Ciencias Computacionales. Pero, la naturaleza aborrece el vacío[10]. Las facultades de ingeniería han ignorado al software por mucho tiempo. La sociedad en general y la industria en particular necesitan desarrolladores profesionales de software y, como no los encuentran, tienen que recurrir a los llamados "programadores" [11].

- La mayoría de ingenieros y otros profesionales han aprendido autónomamente algún lenguaje de programación y se consideran así mismos "desarrolladores" de software.

- La mayoría de programas formativos incluyen cursos de "programación", al mismo tiempo que involucran al software como parte de otros cursos.

- Las facultades de Ciencias Computacionales han tratado de llenar el vacío en software mediante la inclusión de los llamados cursos de "Sistemas" o de "ciencia computacional aplicada".

Muchos de los que ahora "desarrollan" software han tenido que seguir uno de estos caminos; muchos creen que un grado en Ciencias Computacionales ofrece la mejor preparación para esta profesión. ¡No había otra opción! Actualmente, rara vez estos programas son de ciencia pura, pero tampoco son programas que puedan ganar acreditación como programas de Ingeniería de Software [12].

Siguiendo la formación tradicional en ciencias, las Ciencias Computacionales les ofrecen a los estudiantes mucha más libertad que los programas tradicionales de ingeniería. Esto en sí mismo hace más difícil la acreditación, debido a que estos comités buscan los caminos más débiles en un programa y no lo aceptarán si, incluso, un pequeño componente no cumple con sus criterios. Un problema más profundo es que en los programas tradicionales de Ciencias Computacionales, donde las ciencias experimentales y teóricas a menudo se consideran sub-campos de la competencia, existe poca conexión entre las partes teóricas y las prácticas. Algunos cursos, como la lógica de programación, se identifican como teóricos y otros, como los compiladores, se consideran prácticos; pero existen pocos programas que utilicen la teoría vista en un curso para practicarla en el otro. Esto contrasta ampliamente con los programas tradicionales de ingeniería, donde se considera importante formar en la aplicación de la teoría para la resolución de problemas prácticos, es decir, integrar en vez de separar, por ejemplo, las matemáticas y el diseño. Además, en estos programas la "práctica" muchas veces confunde tecnología con ingeniería. Algo más grave aún es que muchos cursos no subrayan sus fundamentos y se organizan alrededor de las modas del momento o de las palabras nuevas. También es sorprendente la forma en la que muchos de estos cursos forman acerca de sistemas o de lenguajes específicos aunque están llenos de material que será obsoleto antes que el estudiante se gradúe [13].

La necesidad de la acreditación en el área de software no ha escapado a la atención de la industria y de los académicos con inclinación práctica. Desde hace varios años en EE.UU. un comité de científicos computacionales se viene preguntando cómo crear un mecanismo similar para los desarrolladores profesionales de software [14]. Sin embargo, el progreso es poco visible. La pregunta que surge es ¿por qué se necesitan nuevas leyes y mecanismos de acreditación, cuando se puede aprovechar los existentes para satisfacer las necesidades actuales? Sería mucho más fácil identificar y aceptar a la Ingeniería de Software como una "ingeniería", que crear una acreditación y un sistema de concesión de licencias completamente nuevas.

Este intento de los programas en Ciencias Computacionales por cubrir ambas funciones, es lo que dificulta ver por qué se necesita un programa separado en

Ingeniería de Software. En retrospectiva, se puede observar que la separación de la Ingeniería Eléctrica de la Física fue buena idea y permitió que ambos programas realizaran mejor su trabajo. No tiene que ser un compromiso. En la próxima década se podría llegar a la misma conclusión acerca de la Ingeniería de Software y de las Ciencias Computacionales.

El aislamiento de las facultades de Ciencias Computacionales de las facultades de ingeniería dificulta la comprensión de la necesidad de un programa en Ingeniería de Software. Pocos ingenieros continúan en Ciencias Computacionales por lo que no se han dado cuenta de la cantidad de material útil que se ha desarrollado en los últimos 30 años. Para muchos ingenieros, la base científica compartida de la Ingeniería de Software y los programas en Ciencias Computacionales incluye mucho material que no conocen personalmente y, en consecuencia, no pueden apreciar su importancia para la ingeniería [1].

Algunas personas tienen la impresión de que los programas de Ingeniería de Software hacen hincapié en el software que se utiliza en las aplicaciones de la ingeniería tradicional. Muchos ingenieros tradicionales parecen creer que su trabajo se limita a la construcción de productos físicos, algo que dejó de ser cierto desde hace varias décadas. Actualmente, muchos de los productos más importantes son intangibles. Hace 30 años, diseñar una antena direccional significaba "cortar hojalata", hoy significa escribir programas [2].

A los ingenieros se les forma en cómo aplicar la ciencia y las matemáticas para diseñar productos que otros utilizarán. Esto es especialmente importante en situaciones donde la seguridad y el bienestar de la población dependen del correcto diseño de esos productos. Sin embargo, muchos ingenieros también trabajan en otros productos, que hacen parte de sistemas por fuera de las áreas de la ingeniería tradicional. Los computadores son herramientas de propósito general y los profesionales en Ingeniería de Software deben ser igualmente flexibles. Los principios básicos del diseño de software y las técnicas que se aplican a todo tipo de desarrollo son los mimos y los profesionales en software deben estar lo mejor dispuestos para trabajar, sea en un banco o en una empresa siderúrgica.

A continuación se describen las principales diferencias entre estos dos programas:

- *Diferencias en la filosofía del currículo*. Estas diferencias se observan en la Tabla 1.

Tabla 1
Diferencias en la filosofía del currículo

Ingeniería de Software	Ciencias Computacionales
Se debe diseñar para acreditarse como un programa de ingeniería.	No tiene por qué estar sujeto a restricciones.
Será relativamente rígido con pocas opciones técnicas.	Debe continuar ofreciendo las posibilidades tradicionales de especialización.
Al igual que muchos programas de ingeniería, podría no exigir a los estudiantes la elección de su especialidad ingenieril hasta el segundo año y puede compartir el primer año con otros programas de ingeniería.	Los estudiantes pueden empezar a presentar el material de su especialidad desde el primer año.
Debe hacer hincapié en la amplitud, es decir, estar diseñado para asegurar que sus egresados tienen cierta familiaridad con los temas más importantes de ingeniería.	Debe continuar ofreciendo a los estudiantes una oportunidad de ser impulsados por la curiosidad cuando seleccionen cursos y permitir su especialización.
Debe incluir gran cantidad del material básico que aceptan muchos otros estudiantes de ingeniería.	Ese material básico podría no ser necesario para los estudiantes.
Debe insistir en la aplicación de las Ciencias Computacionales en una variedad de áreas.	Debe hacer hincapié en la comprensión de las propiedades inherentes de los sistemas informáticos y centrarse en el soporte de software y en el desarrollo de herramientas.
Debe colocar más énfasis en la enorme cantidad de material que ya se ha demostrado en la práctica.	Puede dedicar más tiempo a discutir áreas de investigación que todavía no son rutinarias o incluso estar dispuesto a utilizarlas.
Al mismo tiempo que es fuerte en teoría debería hacer hincapié en su aplicación práctica.	Debe permitirles a los estudiantes estudiar la teoría por sí mismos y prepararlos para trabajar en su extensión o refinamiento.
Debe tener un fuerte énfasis en aplicaciones de usuario final.	Debe preparar a sus egresados para desarrollar nuevas herramientas para desarrolladores de software, para reemplazar las primitivas y las *ad hoc* que están disponibles.
La teoría y las consideraciones prácticas deben estar integradas.	Se espera que los cursos especializados tradicionales continúen.

- *Diferencias en la cobertura*. Muchos temas en las Ciencias Computacionales son interesantes y desafiantes, pero aún no han encontrado aplicación práctica –la semántica denotacional de los lenguajes de programación es una de ellas. Se puede afirmar que es posible construir sistemas software de sonido sin ningún conocimiento en este campo. Por el contrario, difícilmente se podría llamar ingeniero de software a alguien que no tenga cierta familiaridad con este campo. Además, es posible hacer comentarios

similares acerca de las Redes Neurales, mucha parte de la Inteligencia Artificial y de algunos aspectos de la computabilidad y la teoría de autómatas. Algunas discusiones sobre esos temas deben incluirse en los programas de Ciencias Computacionales, pero un graduado de un programa de Ingeniería de Software debe comprender aspectos de la comunicación, la teoría de control y el diseño de interfaces, que raramente se ven en otros programas [15].

Cada uno de estos programas formativos es un compromiso y cualquier tema que sea esencial en uno sería de potencial interés para los estudiantes en el otro. Sin embargo, la extensión limitada de los programas universitarios puede forzar a tomar decisiones con base en las prioridades. En un programa de Ingeniería de Software la prioridad será la utilidad y la aplicabilidad; para otros será importante dar prioridad al interés intelectual, al futuro desarrollo del área y a la formación en los métodos científicos que se utilizan en el estudio de los computadores y del desarrollo de software.

- *Diferencias en el estilo y contenido de los cursos*. Después de analizar a estudiantes de programas de Ingeniería de Software y de Ciencias Computacionales en varias universidades americanas y europeas, no fue posible encontrar diferencias sustantivas entre ellos [16]. La mayoría de los estudiantes en Ciencias Computacionales son relativamente pacientes y dispuestos a explorar temas, sólo porque son interesantes; mientras que la mayoría de estudiantes de Ingeniería se impacientan si no se les muestra cómo aplicar lo que están aprendiendo. Para muchos estudiantes de ingeniería decir que un curso "es demasiado teórico" es una fuerte crítica; mientras que para muchos estudiantes de Ciencias Computacionales es enaltecerlo. Estas diferencias deben quedar reflejadas en los programas y en los lineamientos de los cursos.

  Se deben incluir muchos temas en los programas de Ingeniería de Software y de Ciencias Computacionales, pero se tienen que impartir en cursos muy diferentes. Por ejemplo, la lógica matemática es un tema fascinante que evidentemente es importante para ambos grupos de estudiantes. En la formación en lógica para los estudiantes de Ingeniería de Software, es esencial enfatizar el uso de la lógica para describir las propiedades de los sistemas y las propiedades de los estados; también destacar su papel en el control de las especificaciones y programas en cuanto a completitud y coherencia; debatir acerca de las pruebas y lograr que los estudiantes tengan la oportunidad de utilizar software de demostración de teoremas, pero no será necesario dedicar mucho tiempo a las diferencias entre los tipos de lógicas.

  Otro ejemplo podrían ser los cursos de sistemas operativos. Es interesante y útil formar en una taxonomía de sistemas operativos, clasificándolos según las características y las propiedades, de la misma forma en la que un biólogo clasifica a los insectos. Incluso, se puede penetran en los estudios evolutivos analizando cómo se mueven las ideas de uno a otro lado. Tal curso podría ser importante para alguien que va a investigar cómo construir nuevos sistemas operativos o a desarrollar nuevos modelos y teorías. Sin embargo, ese supuesto no sería la mejor forma para trabajar con estudiantes de Ingeniería de Software. Ellos estarían menos interesados en la historia o la anatomía comparada de los sistemas, pero querrán saber cómo seleccionar un sistema operativo y cómo utilizarlo en un contexto problemático específico. Ningún programa de ingeniería puede darse el lujo de detallar la información acerca de un sistema operativo en particular debido a que podría quedar obsoleto antes que los estudiantes se gradúen; pero se pueden formar en principios básicos que les ayuden a tomar buenas decisiones y a utilizar eficientemente cualquier sistema operativo. Aunque es cierto que algunos ingenieros terminan diseñando nuevos sistemas operativos, están mucho más propensos a usar los existentes. Sin embargo, mucho del contenido de un curso de sistemas operativos es relevante para el diseño de otros sistemas interactivos y de tiempo real. Ese material se debe incluir en un curso de diseño avanzado de software.

- *Diferencias en los objetivos formacionales*. Muchos programas de Ciencias Computacionales, al igual que otros programas científicos, forman a los estudiantes para continuar estudios en un programa de postgrado, mientras que los programas tradicionales de ingeniería se centran en prepararlos para entrar inmediatamente al mundo laboral, luego de completar su programa de pregrado.

5. UNA PROPUESTA DE PLAN DE ESTUDIOS

Un programa de Ingeniería de Software que se pueda agregar al ya mencionado conjunto I, debería estar integrado por:

- Cursos básicos, adoptados de otras disciplinas de ingeniería. Muchos ingenieros de software trabajarán en equipo con otros ingenieros, por lo que deben compartir con ellos una base común de conocimientos.

- Cursos para Ingenieros de Software, que proporcionen una visión general de las cuestiones de ingeniería básica. El ingeniero de software no puede ser un ingeniero universal, el programa no puede hacer frente a ciertos temas de ingeniería con la misma profundidad que otros programas, sin embargo, debe proporcionar buenas bases de formación que le permitan conocer y plantear discusiones alrededor de variados campos de la ingeniería.

- Cursos sobre los fundamentos matemáticos de la Ingeniería de Software, destacando sus aplicaciones en el desarrollo de software. Cada uno de estos cursos introduce un área importante de las matemáticas para la Ingeniería de Software, que no siempre se trabaja en otras ingenierías. En cada uno de estos cursos, los

ejemplos y tareas deberán mostrar cómo utilizar las matemáticas para diseñar software. Por otra parte, en lo posible, se utilizarán las herramientas matemáticas para darles a los estudiantes una experiencia práctica en el uso de los conceptos.

- Cursos de diseño de software, orientados a demostrar cómo utilizarlos para diseñar productos software exitosos. Estos cursos son el núcleo del programa y, en parte, representan material de las Ciencias Computacionales.

- Además, se deberá incluir el conjunto habitual de cursos en estudios complementarios.

A continuación se describe una propuesta de plan de estudios para un programa en Ingeniería de Software.

Cursos compartidos con otras disciplinas de ingeniería
- Química general para ingeniería.
- Matemática para ingeniería.
- Cálculo para ingeniería.
- Mecánica introductoria.
- Ingeniería del diseño y la comunicación.
- Formación en seguridad básica.
- Ondas, electricidad y campos magnéticos.
- Introducción a la programación para ingenieros.
- Principios de economía.
- Habilidades comunicativas: escritas y verbales.

Cursos de introducción a otras áreas de ingeniería
- Introducción a la estructura y propiedades de la ingeniería de materiales.
- Introducción a la dinámica y control de sistemas físicos.
- Principios de sistemas digitales y diseño lógico para ingenieros de software.
- Arquitectura de computadores y multiprocesadores.
- Introducción a la termodinámica y a la transferencia de calor.

Cursos de matemáticas aplicadas
- Aplicaciones de la lógica matemática en la Ingeniería de Software.
- Aplicaciones de la matemática discreta en la Ingeniería de Software.
- Métodos estadísticos para ingenieros de software.
- Métodos formales.

Cursos relacionados con Software
- Ingeniería de Requisitos.
- Diseño de software.
- Arquitectura de software.
- Sistemas de información.
- Diseño y selección de lenguajes de programación.
- Métodos computacionales para ciencia e ingeniería.
- Diseño y selección de algoritmos computacionales y estructuras de datos.
- Métodos de optimización, modelos gráficos, investigación y técnicas de depuración.
- Técnicas de gestión de datos.
- Lógica y abstracción computacional.
- Software y responsabilidad social.
- Diseño de sistemas en tiempo real y sistemas de control computarizado.
- Fundamentos computacionales.
- Análisis de rendimiento de sistemas informáticos.
- Diseño de interfaces hombre-máquina.
- Diseño de sistemas y cómputo en paralelo y computación distribuida.
- Software en sistemas de comunicaciones.
- Redes de computadores y seguridad informática.
- Desarrollo de sistemas de calidad.
- Pruebas del software.
- Diseño y administración de bases de datos.

6. CONCLUSIONES

- *La Ingeniería de Software es diferente a las Ciencias Computacionales*. La revisión de la propuesta que aquí se describe demuestra que un programa para Ingeniería de Software es muy diferente de un programa en Ciencias Computacionales que titula "Ingenieros de Software". Los cursos clásicos de las Ciencias Computacionales, como compiladores y sistemas operativos, no están en este programa porque es probable que los egresados no los diseñen; pero el material que se puede utilizar en otras aplicaciones, como algoritmos neurales, máquinas para representación de estructuras de datos o la sincronización de actividades concurrentes, se distribuyen en otros cursos. Los programas de Ciencias Computacionales tienden a enfocarse en las áreas centrales del software, pero existe una creciente necesidad por desarrollar software para nuevas aplicaciones, donde el software remplaza o complementa a las tecnologías de ingeniería tradicional. Aproximadamente en la mitad de los cursos sugeridos no se forma a los estudiantes de Ciencias Computacionales, pero son importantes para un número creciente de aplicaciones de software. Esta propuesta se centra en los principios fundamentales del diseño que se pueden aplicar en las áreas clásicas de las Ciencias Computacionales y en gran variedad de aplicaciones, en las que son necesarios para formar desarrolladores profesionales de software.

- *Los ingenieros de software deben estar acreditados*. Los productos software son muy importantes para la seguridad y el bienestar de la población, por lo cual es importante tener cierta garantía de que quienes practican la Ingeniería de Software son egresados de un programa que utiliza el material básico más importante para su formación. Acreditar el ejercicio de los Ingenieros de Software es tan importante como Acreditar el de los Ingenieros Civiles.

- *La formación en Ingeniería de Software debe centrarse en lo fundamental*. Cuando un estudiante de Ingeniería Eléctrica comienza su proceso formativo, se sorprende al notar que muchos de los manuales y libros que posee ya no le sirven para nada y que ninguno de sus maestros aborda las temáticas que contienen. Cuando pregunta el motivo, lo que obviamente recibe como

respuesta es que los dispositivos y tecnologías que fueron muy populares antes ya no tienen interés. En su lugar, lo forman en física fundamental, en matemáticas y en una forma de pensar que le será útil para entender las necesidades de hoy y del futuro cercano. Claramente, la experiencia práctica es esencial en todos los programas de ingeniería, porque le permite a los estudiantes aprender a aplicar el conocimiento que reciben. Como se han dado cuenta, quienes conocen de software, en las líneas precedentes no se mencionan temas actuales como Java, Orientación por Objetos, Tecnologías Web, Orientación a Componentes, XP, Programación Ágil o frameworks. Pero, aunque se debe formar en muchas de las buenas ideas que subyacen a estos enfoques y herramientas, son paradigmas de moda que pueden desaparecer antes que el estudiante se gradúe. Los ejercicios de laboratorio y otros proyectos les deben proporcionar a los estudiantes la oportunidad de utilizar las herramientas más populares y de experimentar con algunas nuevas. Sin embargo, se debe recordar que estos temas son el remplazo actual de los que estaban de moda ayer. Es responsabilidad de los maestros recordar que, en esta ingeniería, lo que se estudia hoy tiene una durabilidad de no más de una década y que deben identificar los fundamentos que serán válidos y útiles en este período y enfatizar dichos principios en los contenidos. Muchos programas pierden de vista el hecho de que la formación en un determinado sistema o lenguaje sólo es un medio de aprendizaje y no un fin en sí mismo.

- *Se necesitan nuevos cursos, no una nueva combinación de los existentes*. En algunas instituciones, el debate acerca de la Ingeniería de Software se ha tratado como un conflicto de competencias entre facultades de Ciencias Computacionales y de Ingeniería. El conflicto, a veces, se resuelve mediante la inclusión de algunos cursos de cada facultad en el nuevo programa. Este tipo de arreglo producirá graduados que ni son ingenieros ni son científicos computacionales. Es esencial que el material de las Ciencias Computacionales se imparta al estilo de la ingeniería.

- *Debe cambiar el estilo para impartir y organizar los cursos*. Es importante que los cursos de software se impartan de forma diferente a los cursos convencionales de las Ciencias Computacionales. En los cursos de ciencia, es bastante razonable formar acerca de las cosas, pero en los cursos de ingeniería se forma en cómo hacer las cosas. Con los estudiantes de ingeniería no se puede simplemente llenar el tablero con las derivaciones y las pruebas; cada curso debe integrar teoría y práctica y subrayar cómo aplicar la teoría cuando se diseña un producto. Mucho de lo que un maestro imparte es lo que le fue impartido a él y lo imparte de la forma cómo se le impartió. Debido a que no abundan graduados en Ingeniería de Software, del tipo que aquí se propone, es necesario trabajar con material desconocido o con material familiar de una manera desconocida. Hasta que estos programas estén bien establecidos habrá que ser muy cuidadosos y detallados al especificar los contenidos de los cursos, así como implementar una cuidadosa coordinación y supervisión.

- *El profesorado es un problema crítico*. Encontrar maestros apropiados para este tipo de programas es crítico y difícil. Los estudiantes que optan por la Ingeniería de Software como carrera profesional, son personas que quieren capacitarse para diseñar y analizar sistemas reales. Es importante que los maestros sean personas que sepan cómo hacer esas cosas, que estén interesados en la construcción de software y que tengan experiencia profesional aplicando el tema que van a impartir. Muchos de los científicos computacionales de hoy, incluso aquellos que identifican su área de interés en la Ingeniería de Software, están interesados en abstracciones y son reacios a involucrarse en el diseño de productos e incluso a mirar de cerca lo que se está haciendo en la práctica actual. Por otra parte, algunos ingenieros saben lo suficiente de Ciencias Computacionales como para impartir estos cursos de forma adecuada. Como es fundamental tener experiencia en el desarrollo de productos software y debido a que se tiene un tiempo limitado, es preciso utilizar diferencias en la contratación. Se tarda mucho más tiempo producir un producto software que escribir un documento. No se puede comparar numéricamente a la experiencia de personas prácticas con la de quienes han sido investigadores puros. En algunos casos, la experiencia y el conocimiento pueden valer más que un postgrado. En ingeniería, la experiencia práctica debe valorarse como una característica importante, más que para algunas de las ciencias.

- *Una cooperación real servirá verdaderamente a los estudiantes.* En algunos lugares e instituciones, al hablar sobre este tema, se encuentra una profunda, sincera y decidida oposición a la idea de que la Ingeniería de Software se trate como a una nueva ingeniería. Por el lado de las Ciencias Computacionales se observa la falta de reconocimiento a la gran cantidad de conocimiento que se ha acumulado acerca de cómo desarrollar software, y muchos ingenieros parecen creer que no es más que el aprendizaje de lenguajes y convenciones de sistemas operativos. Algunos sostienen que cualquier ingeniero que escriba código es un ingeniero de software y creen que no se necesita ninguna habilidad especial o experiencia para diseñar, estructurar y mantener productos software. Otros creen firmemente que no se puede tener una disciplina de ingeniería cuya base científica se encuentra por fuera de las ciencias físicas; consideran que programas como el propuesto aquí “reduce” y que es demasiado denso en software. La mayoría de ingenieros que han revisado el programa propuesto comentan que se deben sustituir muchos cursos de software, y remplazarlos con cursos adicionales de introducción a otras áreas de ingeniería para ingenieros de software. En otras palabras, todavía no consideran a la Ingeniería de Software como una disciplina ingenieril.

Por el lado de las Ciencias Computacionales las reacciones son menos negativas, pero las razones son más complejas. Un factor importante, aunque equivocado, para los científicos computacionales es la sensación de que se les está robado algo; sienten que ellos, no las facultades de ingeniería, son quienes deberían diseñar y controlar algún programa de Ingeniería de Software [15]. Igualmente serio es la falta de comprensión acerca de cómo se diferencia la formación en ingeniería de la formación que recibieron. Muchas personas, que han realizado con éxito la transición desde otras disciplinas, como la física o las matemáticas, a un programa de ingeniería son testigos de lo mucho que tiene que cambiar su estilo en el aula y el contenido del curso. Pocos científicos computacionales han hecho esta transición o reconocen la necesidad de hacerlo. Además, son pocos los maestros de Ciencias Computacionales que reconocen que para ser un buen ingeniero de software se debe ser mucho más que un buen programador. En consecuencia, la mayoría de graduados en Ciencias Computacionales que han revisado esta propuesta comentan que gran parte del material en los cursos para introducir otras áreas de ingeniería, no son pertinentes y que se deben sustituir por cursos adicionales de software.

La formación de los ingenieros de software, que estén dispuestos a trabajar como desarrolladores profesionales, no la pueden realizar científicos computacionales o ingenieros que trabajen aislados. Trabajar en equipo es su responsabilidad para con los estudiantes y cada grupo debe estar preparado para aprender del otro.

7. REFERENCIAS

[1] Mitchell, W. 2004. Is software engineering for everyone? Proceedings of 2nd annual conference on Mid-south College computing, 53-64. April 02-03, Little Rock, USA.

[2] Thompson, J. B. 2001. A Long and Winding Road (Progress on the Road to a Software Engineering Profession). Proceedings of 25th International Computer Software and Applications Conference on Invigorating Software Development, 39-45. October 08-12, Chicago, USA.

[3] Ford, G. & Gibbs, N. E. 1996. A Mature Profession of Software Engineering. Office, 2(January), 55-62.

[4] McConnell, S. & Tripp, L. 1999. Professional Software Engineering: Fact or Fiction? IEEE Software, 16(6), 13-18.

[5] Cupp, J. W. 2001. Reviewing the professionalization of software engineering: can small colleges remain viable? Journal of Computing Sciences in Colleges, 17(1), 132-146.

[6] DeMillo, R. A.; Lipton, R. J. & Perlis, A. J. 1979. Social Processes and Proofs of Theorems and Programs. Communications of the ACM, 22(5), 271-280.

[7] Musa, J. D. 1985. Software Engineering: The Future of a Profession. IEEE Software, 2(1), 55-62.

[8] Glass, R. L.; Vessey, I. & Ramesh, V. 2002. Research in software engineering: an analysis of the literature. Journal of Information and Software Technology, 44(8), 491-506.

[9] Mahoney, M. S. 1990. The roots of software engineering. CWI Quarterly, 3(4), 325-334.

[10] Spinoza, B. 1982. The Ethics of Spinoza: The Road to Inner Freedom. Carol Publishing Group.

[11] Harold, F. G. 1988. The Two Cultures in Computing. Proceedings of the ACM SIGCPR conference on Management of information systems personnel SIGCPR '88, 188-191. April 07-08, Maryland, USA.

[12] Horn, E. & Kupries, M. 2003. A Study Program for Professional Software Engineering. Proceedings of the 16th Conference on Software Engineering Education and Training, 398-409. March 20-22, Madrid, Spain.

[13] Estublier, J. & García, S. 2006. Concurrent Engineering support in Software Engineering. Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering, 209-220. September 18-22, Tokyo, Japan.

[14] Boehm, B. K. 1994. The IEEE–ACM Initiative on Software Engineering as a Profession. IEEE Computer Society Software Engineering Technical Council Newsletter, 13(1), 1.

[15] Manterea, T. & Alanderb, J. T. 2005. Evolutionary software engineering, a review. Applied Soft Computing, 5(3), 315-331.

[16] Naveda, J. F. & Lutz, M. J. 1997. The Role of Software Engineering in Undergraduate Education. Proceedings of the 27th Annual Conference. Frontiers in Education Conference, Teaching and Learning in an Era of Change, 2, 750. November 05-08, Pittsburgh, USA.