Re Visión

Artículo invitado

AENUI
Asociación de Enseñantes Universitarios de la Informática

Sintonizar universidades y empresas

Ricardo Galli
Departament de Matemàtiques i Informàtica
Universitat de les Illes Balears
gallir@gmail.com

Resumen

El origen de este artículo es una respuesta en forma de ensayo, *Sintonizar universidades y empresas, pero ¿qué debe saber un ingeniero?* [12] a un artículo de Martín Varsavsky, *El Paro, el problema número uno de España se agrava* [27]. Lo que más me llamó la atención, no por novedosa sino por el contexto, fue la frase: «Porque España, desde Cataluña a Andalucía, educa a los estudiantes de una manera que los hace poco atractivos para los empleadores. Hasta que no se sintonicen las carreras con lo que las empresas necesitan seguiremos con dificultades para emplear a jóvenes».

Actualmente no dudaríamos en calificar a las ingenierías como una titulación profesional y a un ingeniero como un profesional. Pero existe un gran desconocimiento sobre el significado de ambas conceptos. La ingeniería es la aplicación de los conocimientos científicos para la resolución de problemas. El hecho de ser una profesión implica que es una actividad de aplicación práctica pero con un fuerte componente intelectual. Además de ello una profesión involucra la responsabilidad de servicio a la comunidad y la definición de altos estándares para el ejercicio de sus actividades.

La formación de profesionales exige que acaben teniendo conocimientos amplios sobre su campo, pero además deben ser capaces de resolver problemas prácticos que aprenden fundamentalmente durante un entrenamiento y la propia práctica de su profesión. La experiencia y la formación continua son parte esencial de la carrera profesional.

En este artículo se hará un breve repaso a la historia de las profesiones y se analizarán los problemas y causas que hace que muchos empresarios y la sociedad en general tengan la visión de una presunta falta de sintonía exclusivamente española entre la formación universitaria y las necesidades de las empresas.

Palabras clave: Ingeniería, Profesión, Conocimientos

Recibido: 30 de Noviembre de 2008; **Aceptado:** 18 de diciembre de 2008

1. Introducción

La historia de los *gilds* (o *guilds* o gremios) en Europa es apasionante no sólo por su variedad y larga historia, sino porque de ellos derivan muchos de los beneficios sociales que damos por hechos en la mayoría de los países europeos (seguridad social, seguro laboral, días laborables y fines de semana, trabajo diurno, sindicatos). Pero también han marcado el origen de lo que hoy conocemos como profesiones.

Los primeros gremios profesionales —o *craft-guilds*— aparecieron en el siglo XI y XII en Italia, Alemania y los Países Bajos. Pocas veces se llamaban a sí mismos *gilds* o gremios: cuando aparecieron los primeros *gilds profesionales* se referían a sí mismos en documentos oficiales como *arte*, *métier* o *Zunft*.

Inicialmente los *craft-guilds* fueron asociaciones de trabajadores que mezclaban intereses particulares con los sociales. Por un lado se encargaban de formar a los aprendices, establecieron normas de trabajos —horas, días, instalaciones, etc—. Los que pasaban el período de formación y tenían una cierta experiencia eran reconocidos como másters. Por otro lado perseguían intereses sociales como cobrar una tasa para pagar atención médica, entierros, o pagar una pensión a las viudas de sus asociados.

Estos *gremios de artesanos* reclamaban el derecho a decidir qué trabajos les competían, cuánto tiempo de aprendizaje necesitaban los aprendices, y a regular el comportamiento de los *jornaleros* [1]. Aunque los *craft-gilds* surgieron como un activismo contra el poder feudal en el siglo XIV se empezaron a hacer patente los abusos de poder de los *gilds*: admitían cada vez menos másters, estos contrataban a los cada vez más numerosos *jornaleros* más allá de las cuotas permitidas, definían el precio de los productos, establecían cuotas de producción para mantener los precios elevados, establecían tasas para poder comerciar en las ciudades, impedían el uso de maquinarias para «mantener el valor artesanal», sólo formaban y admitían como másters a los hijos de otros másters, etc. Este nepotismo creó una segunda clase —muy numerosa— de trabajadores, los *jornaleros*, que empezaron a asociarse entre ellos[2] y fueron muchas veces perseguidos por ser considerados *asociaciones conspirativas*.

Este poder de los *gilds* se fue perdiendo con el nacimiento de los estados europeos, especialmente notable a partir del Renacimiento, donde se consideraba que los *gilds* limitaban el comercio y el ejercicio profesional.

[1] Aprendices que completaron el ciclo de aprendizaje pero todavía no eran reconocidos como másters.
[2] En Francia estas asociaciones se llamaban *compagnionages*.

Lo que en un principio surgió como una asociación que promovía la "libertad ciudadana" que marcaría los inicios de lo que hoy conocemos como profesión definiendo estándares de conocimiento y capacidad, *responsabilidad social*[3] y códigos de funcionamiento social y ayuda mutua se convirtieron en grupos de élites excluyentes, sostenedores de monopolios, *lobbies* políticos poderosos de fijación de precios y cuotas. A medida que aumentó el poder de los estados centralizados, estos empezaron a eliminar el gran poder local que ejercían los *gilds*, y los propios estados empezaron a crear sus propios métodos de definición de las profesiones. Esta es una de las razones fundamentales por la que tenemos títulos profesionales definidos por los estados y otorgados por las universidades.

Más información de este proceso puede encontrarse en los libros de Black [6] y de Krause [15].

1.1. Profesiones modernas

La medicina es siempre citada como la más antigua y madura de las profesiones modernas. La persona que más influyó en dar este carácter a la medicina fue Abraham Flexner. En 1915 escribió en *Is social work a profession?* [10]:

> Profesión [...] involucra un actividad intelectual y de responsabilidad personal; deducen su material directamente de la ciencia y el aprendizaje; poseen técnicas organizadas y que pueden ser enseñadas y comunicadas a otros; han evolucionado en un estatus personal y profesional muy definido; y tienden a convertirse, cada vez más claramente, en órganos para el logro de objetivos sociales más grandes.

Los trabajos y reflexiones de Flexner tuvieron una gran influencia —primero en EEUU y luego en Europa— en la definición de lo que hoy conocemos como profesión. Básicamente, Flexner define seis criterios que debe cumplir una profesión:

1. La profesión es una actividad basada en acciones intelectuales junto con una responsabilidad personal.
2. La práctica de una profesión está basado en conocimiento, no en actividades rutinarias.
3. Existen aplicaciones prácticas, no sólo teorías.
4. Existen técnicas que pueden ser enseñadas.
5. Una profesión se organiza internamente.
6. Una profesión está motivada por altruismo: sus miembros trabajan por el bien de la sociedad.

Esta fue una de las primeras definiciones formales de una profesión, y generó largos debates, discusiones que duraron décadas. Otros autores influyentes en la definición de las profesiones fueron los ingleses Alexander M. Carr-Saunders y Paul A. Wilson que en 1933 publicaron un volumen describiendo las profesiones británicas que sirvieron de base para el capítulo *Professions* de la *Encyclopaedia of the Social Sciences* en 1934.

> [...] "profesionalismo" —los estándares que los profesionales deben seguir, estándares que fueron desarrollados en un punto para una profesión y posteriormente para otras—. La medicina fue una de las primeras profesiones en materializar y manifestar estándares de grupos que pueden servir de modelos para otros profesionales.

Talcott Parson en *Remarks on Education and the Professions* (en *International Journal of Ethics*) describe a la profesión en términos similares a Flexner y además comenta:

> [...] de tendencia universalista e independiente en la sociedad, o sea, independiente de otros grupos étnicos o geográficos o de asociaciones vecinales [...] los profesionales tiene objetivos de reputación y honor.

Logan Wilson en 1942 describió a los profesionales también de forma similar a Flexner además añadiendo otras descripciones funcionalistas:

> [...] con entrenamiento especial [...] operan bajo sus propias interpretaciones del conocimiento, [...] interés personal limitado y con obligaciones hacia la profesión y los clientes.

Para finalizar, una frase muy conocida —aseguran que es la más citada—de Francis Bacon:

> *I hold every man a debtor to his profession; from the which as men of course do seek to receive countenance and profit, so ought they of duty to endeavor themselves, by way of amends, to be a help and ornament thereunto.*
>
> (Considero que cada hombre es un deudor a su profesión. De ella buscan respeto y provecho, y por tanto tienen el deber, como compensación, a ser de ayuda y utilidad.)

Se puede decir que hay un consenso general sobre qué es la profesión: conocimientos amplios de un campo, trabajo intelectual y un límite autoimpuesto del interés personal a favor del interés social [22].

> La tercera condición de una auténtica profesión es que sus técnicas de operación se basen en principios amplios más que en recetas de procedimientos o habilidades rutinarias, aunque estas sean complejas.
> [...]
> Hemos considerado lo que yo creo son los tres requerimientos básicos de una profesión —ética, estándares [de calidad], y a falta de un término mejor, "principios generales"—.

Estos conceptos están reflejados en todos los códigos deontológicos, incluso en los de informática propuestos por la ACM [3] o en las genéricas para las ingenierías de la IEEE [13].

[3] Denominadas en su época *officium*, *Amt*, *ministerium*.

1.2. Cómo llegamos a la universidad moderna

Abraham Flexner no sólo influyó en la definición moderna de la profesión, también tuvo una gran influencia en el modelo de la universidad moderna [11]. En contra de la opinión mayoritaria de su época —finales del siglo XIX, principios del siglo XX— su modelo de universidad es la que une investigación en ciencias, la enseñanza y entrenamiento de alto nivel, el trabajo conjunto de graduados con profesores y el compromiso con el "descubrimiento intelectual". Una frase suya define su ideal de universidad:

> Una institución conscientemente dedicada a la búsqueda del conocimiento, la solución de problemas, la satisfacción crítica de los logros y el entrenamiento de muy alto nivel.

Todas las universidades públicas europeas —también las españolas— y norteamericanas han adoptado este modelo básico propugnado por Flexner. No sólo le debemos algunos de los principios de las profesiones modernas, también el modelo de universidad que nos da el título profesional.

Estas pequeñas historias que acabo de contar evidencian que el camino para definir lo que hoy conocemos como "profesión" no ha sido un camino corto ni fácil.

La universidad comenzó su camino hace casi 2.400 años en un naranjal de Academus, donde Platón montó su *Escuela de Atenas*, luego en las universidades medievales, y que tuvo otro hito importante en el siglo XI cuando el malogrado filósofo y monje[4] Pierre Abélard creó el primer campus universitario moderno autónomo llamado *Universitas* en las afueras de París[5]. Pero no fue hasta entrado el siglo XX que se acabó de definir el modelo de universidad en la que ahora estudiamos.

Como ya he comentado, la profesión, y conceptos fundamentales como días laborables y de descanso, comenzaron su andadura en el siglo XI con los *gilds*. Lo que comenzó como un interés asociativo para mejorar los *oficios* se convirtieron en grupos nepóticos y de poder que fueron combatidos por los nacientes estados europeos más centralizados.

Pasaron casi 600 años para que ese poder de los estados conjugara un nuevo modelo de universidad —investigación más docencia— con conceptos modernos de profesión como una formación de alto nivel y compromiso ético con la sociedad. En Europa además la universidad es hija directa de unos de los primeros *actos solidarios* de los nuevos estados europeos: la biblioteca pública. Por ello es que podemos disfrutar de educación pública financiada por el estado —es decir, de toda la sociedad—.

No sé si los lectores apreciarán las —desde mi punto de vista— estrechas relaciones entre la historia de la profesión y el origen del modelo de universidad española y europea. Es interesante conocer el porqué somos como *somos* y el porqué hacemos lo que *hacemos*.

2. El conocimiento de los profesionales

A efectos de simplificar los tipos de conocimientos y habilidades de un *profesional* —en el sentido laxo, no necesariamente con título profesional universitario— o *ingeniero* consideremos que tiene conocimientos o entrenamiento en tres áreas distintas:

(1) Ciencia
(2) Tecnología
(3) Herramientas

Así se pueden definir esos tres conceptos fundamentales, y además qué es la *ingeniería*.

Ciencia: Es por un lado, el proceso mediante el cual se adquiere conocimiento, y por el otro, el cuerpo organizado de conocimiento obtenido a través de este proceso.

Tecnología: Es el conjunto de saberes que permiten fabricar objetos y modificar el medio ambiente, incluyendo las plantas y animales, para satisfacer las necesidades y los deseos humanos. En la sociedad, la tecnología es consecuencia de la ciencia y la ingeniería.

Herramienta: Es frecuente usar el término *herramienta*, por extensión, para denominar dispositivos o procedimientos que aumentan la capacidad de hacer ciertas tareas. Tal es el caso de las *herramientas de programación*, *herramientas matemáticas* o *herramientas de gestión*.

Ingeniería: Conjunto de conocimientos y técnicas que permiten aplicar el saber científico a la utilización de la materia y de las fuentes de energía. O según la Wikipedia: «Es la profesión que aplica conocimientos y experiencias para que mediante diseños, modelos y técnicas se resuelvan problemas que afectan a la humanidad. [...] La ingeniería es la profesión en la que el conocimiento de las matemáticas y ciencias naturales, obtenido mediante estudio, experiencia y práctica, se aplica con juicio para desarrollar formas de utilizar, económicamente los materiales y las fuerzas de la naturaleza para beneficio de la humanidad y del ambiente. [...] Otro concepto que define a la ingeniería es el arte de aplicar los conocimientos científicos a la invención, perfeccionamiento o utilización de la técnica en todas sus determinaciones.»

2.1. Ciclos de vida

Podemos ampliar brevemente las definiciones con los *ciclos de vida* de los modelos científicos, las tecnologías y las herramientas.

Los modelos científicos pueden durar y ampliarse durante cientos de años, hasta que aparezcan modelos nuevos que falseen o simplifiquen radicalmente al anterior —*cambios de paradigmas* [16]—. Así, por ejemplo, el modelo de la gravitación universal de Newton duró doscientos años hasta que aparecieron

[4] Fue literalmente castrado por haber tenido un hijo con su alumna Heloïse, sobrina de su mecenas.
[5] Básicamente para huir de la censura de la iglesia y las quejas de los vecinos burgueses por lo ruidos y borracheras de los estudiantes.

los modelos de la relatividad o la física cuántica. En la informática tal como la conocemos ahora el modelo matemático fundamental es el modelo computacional de la máquina de Turing y la arquitectura Von Newman.

Las *ciencias de la computación* es una ciencia formal que no está del todo madura, por lo que crece rápidamente. Aunque el cuerpo del conocimiento se incremente a gran velocidad, todavía no hemos tenido *cambios de paradigmas fundamentales* como en otras ciencias naturales[6]. Los conocimientos del *campo* informático incluyen a muchos conceptos denominados *principios fundamentales* que sirven a la definición de su cuerpo de conocimiento [1]: estructuras discretas, fundamentos de programación, algoritmos y complejidad, arquitectura y organización, sistemas operativos, lenguajes de programación, gráficos, interfaces, sistemas inteligentes, ciencia computacional y métodos numéricos, etc.

Aunque cada vez aparecen nuevos conocimientos derivados del avance tecnológico, el ciclo de vida de esos conocimientos es bastante prolongado y todavía no hemos visto que hayan sido superados por modelos diferentes [1]:

> El campo, sin embargo, continúa evolucionando a una velocidad sorprendente. Tecnologías nuevas son introducidas continuamente, las existentes se hacen obsoletas casi tan rápido como aparecen. [...] Cuando se publicó el CC1991, por ejemplo, las redes no eran percibidas como un área importante [...] La falta de énfasis en redes no es particularmente sorprendente. Después de todo las redes todavía no eran un fenómeno de mercado de masas, y el WWW era poco más que una idea en la mente de sus creadores.

Los *ciclos de vida de las tecnologías* son muchos más breves que los científicos, ya que sus mejoras son influidas por tres factores fundamentales: las mejoras científicas, las mejoras técnicas y las *innovaciones* derivadas de la práctica profesional. Ejemplos de tecnologías y técnicas informáticas típicas son el desarrollo de los lenguajes de alto nivel (compilados, dinámicos), mecanismos y abstracciones de control de concurrencia (semáforos, tareas, hilos, monitores, etc), protocolos, arquitecturas de software, etc. Mientras que el modelo fundamental no ha cambiado desde los inicios de la informática moderna (en la década de 1940–1950), las tecnologías sí han cambiado mucho haciendo cada vez un mejor y mayor uso del conocimiento científico adquirido. El ciclo de vida de estas tecnologías es más breve, muchas han caído en desuso o se han vuelto obsoletas, pero se compensan con la aparición de tecnologías nuevas [1].

> La mayoría de los cambios que afectan a las ciencia de la computación provienen de los avances de la tecnología. Muchos de esos avances son parte de un proceso evolucionario de muchos años. La Ley de Moore [...] se sigue manteniendo. Como resultado hemos visto un crecimiento exponencial en la capacidad de cálculo disponible que hizo posible resolver problemas que no estaban a nuestro alcance hace sólo unos años. Otros cambios en la disciplina, tales como el rápido crecimiento de las redes después de la aparición de el WWW, son más dramáticos, lo que sugiere que los cambios también ocurren en pasos revolucionarios. Ambos, los cambios evolucionarios y revolucionarios, afectan al cuerpo del conocimiento requerido para las ciencia de la computación y el proceso educativo.

Los *ciclos de vida de las herramientas* son aún más breves. Por ejemplo, las mejoras de las tecnologías de interfaz humana[7] creó la necesidad de una forma más cómoda y rápida de introducir texto. Así nacieron los *editores de texto* —siendo emacs y ed/vi los históricos que aún perviven gracias a su evolución—. A medida que mejoró la tecnología de interfaces, los editores de texto se hicieron más potentes hasta el punto que podían ser usados para tareas genéricas que no necesariamente estaban ligadas a la programación y así evolucionaron a los *procesadores de texto*. Los grandes avances científicos y tecnológicos (gráficos, interfaces, potencia de CPU, memoria) han provocado estos cambios, aunque sigan usando todavía las mismos *conocimientos científicos* y muchas de las técnicas que ya usaban en sus inicios. Otros ejemplos de herramientas son: lenguajes/compiladores, editores HTML, herramientas de administración, programas de CAD, ofimática, etc.

Algunos se preguntarán dónde encajan las *metodologías formales* de ingeniería del software[8]. Claramente no pertenecen al *corpus* del conocimiento *científico*[9], pero no se podría asegurar si pertenecen al campo de las "tecnologías" o de las "herramientas". Siguiendo la definición «procedimientos que aumentan la capacidad de hacer ciertas tareas» son una herramienta de la ingeniería, aunque quizás parte de los modelos de procesos desarrollados en algunas metodologías puedan ser considerados como parte de las tecnologías[10].

Aunque un consenso definitivo sería muy complicado, en todo caso los ciclos de vida de las metodologías de diseño son relativamente cortos, a medio camino entre las tecnologías y las herramientas. Por ejemplo la metodología de Gane-Sarson [1977] predominaba en la década de 1980, a mediados de 1990 lo hacía Yourdon [publicada en 1991–1993, aunque desarrollada en los 70 y 80], actualmente predominan las ágiles como RAD [1991], Scrum [1995], XP [1999] y en España la Métrica 3 [2003] (aunque Métrica [1989] estuvo basada en la británica

[6] Posiblemente la informática cuántica, si tiene éxito, sea el origen de un cambio de paradigma.

[7] Fundamentalmente cuando se pasó de las tarjetas perforadas a los teletipos y pantallas con teclado.

[8] Por ejemplo las funcionales o estructuradas tales como la de Gane y Sarson, Yourdon o DeMarco, las orientadas a objetos y las ágiles como XP o Scrum.

[9] Seguramente tanto Kuhn como Popper estarían de acuerdo, no son falsables ni han supuesto una revolución en la forma de desarrollar, verificar o validar software.

[10] De hecho hay muchas herramientas comerciales todavía basadas en Gane/Sarson y Yourdon, incluso algunos libros las recomiendan para la documentación de patentes [26].

[11] Algunos se preguntarán por qué no menciono al ISO 12207. Éste es un complemento a la gestión institucionalizada, no es ni sustituye a metodologías formales, sólo *provee el marco de trabajo y un conjunto de bloques de construcción del ciclo de vida del software para su uso más conveniente y efectivo en las organizaciones* [17, 23].

SSADM [1981], basada a su vez en la de Gane-Sarson, Yourdon y Merise [1978])[11].

3. La ingeniería informática

El debate sobre los conocimientos que deben tener los profesionales informáticos comenzó prácticamente desde los inicios de la "informática comercial" en la década de 1950. Hasta mediados de 1960 no se habían establecido las primeras carreras universitarias de informática y es conocido el largo y persistente conflicto entre la *Association for Computer Machinery* (ACM) y la *Data Management Processing Association* (DPMA) [22] que creó dos facciones separadas.

La DPMA acusaba a la ACM de ser muy academicista y de estar más preocupada por su relación con los académicos y los programas de doctorado que por las necesidades reales de las empresas. La ACM hacía esfuerzos por establecer planes de estudios formales mientras que la DPMA lanzó un programa ambicioso de certificaciones bajo el nombre *Six Measures of Professionalism Program*. Como parte de este programa la DPMA y la entonces reconocida revista *Datamation* lanzaron en 1962 el *Certificate in Data Processing* (CDP).

El programa CDP no tuvo éxito para establecerse como un programa de certificaciones reconocido[12], las empresas empezaron a recurrir a las universidades —con muchas más relaciones con la ACM— para reclutar al personal especializado que necesitaban.

Comparada con los estudios otras ingenierías, la informática es bastante especial, como muestra el hecho que no tengamos en España una carrera de *licenciatura en ciencias de la computación*[13] —como propone la ACM [4]— a diferencia de otras que sí han tenido el *cisma* que separó la carrera académica (o *vocacional*) de la profesional.

Hasta principios del siglo XX no existía la carrera de ingeniería eléctrica. La electricidad era un tecnología cuyo uso no estaba del todo extendida pero que estaba evolucionando a pasos agigantados. Durante los primeras décadas de ese desarrollo eran los propios físicos lo que cumplían el papel de "ingenieros". A principios del siglo pasado se produce en el sistema anglosajón[14] el *cisma* que separó a la ingeniería de su origen científico. Así la licenciatura en física supo mantener su contenido altamente centrado en la ciencia mientras que la ingeniería se enfocó más a la formación de profesionales. Los ingenieros eléctricos siguen teniendo una formación y entrenamiento científico importante —aunque menor a los físicos— ya que esos conocimientos son indispensables para la cumplir con sus objetivos *profesionales*.

En cambio en la informática no hemos tenido, todavía, este cisma[15]. Desde el principio las carreras tradicionales de informática fueron una mezcla de ciencias y de ingeniería, a la inversa de la física-ingeniería eléctrica/industrial. Existen muchas razones, la juventud de las ciencias de la computación, el "tipo de ciencia", la transversalidad con otras ciencias e ingenierías, y sin duda alguna la histórica falta de consenso sobre cuál es la formación más adecuada para los profesionales informáticos.

A diferencia de la física, una *ciencia natural*, la ciencia informática es una *ciencia formal*, más cercana a las matemáticas. Pero las relaciones son más complejas. Los modelos informáticos son modelos matemáticos, pero cuya aplicación a la "naturaleza" o para resolver problemas depende de otros campos científicos e ingenierías [1]:

> La informática es un campo amplio que se extiende más allá de los límites de la ciencia de la computación [...] Los cimientos de las ciencias de la computación se derivan de una amplia variedad de disciplinas. Los estudios de grado de ciencias de la computación exige que los alumnos utilicen conceptos de diferentes campos.

3.1. Los conocimientos de la educación formal

El ideal de todos seguramente sea el modelo de renacentistas famosos como Leonardo Da Vinci, o Galileo Galilei (Figura 1), personas que llegaron a dominar una gran parte del conocimiento, tecnologías y las herramientas disponibles en su época[16].

Desafortunadamente no vivimos en la Florencia del Renacimiento. Afortunadamente para el desarrollo científico, las ciencias y tecnologías han avanzado tanto que es muy improbable que una persona pueda dominar y ser un experto en ciencias, tecnologías y herramientas dispares.

A la hora de desarrollar los planes de estudios formales para la formación de los profesionales que la sociedad necesita hay que encontrar el balance en el enfoque —científico, ingeniería, técnico o especialista— y en el tiempo que se dedicarán a esos estudios.

El ciclo de conocimientos —ciencia, tecnología o herramientas— tiene una gran influencia. Así, por ejemplo, no tiene sentido dedicar cuatro años a la enseñanza de herramientas si estas ya serán obsoletas cuando los alumnos acaben la carrera. Los conocimientos que se enseñan en las carreras se distribuyen aproximadamente como se muestra en la Figura 2[17]. La superficie total coloreada de la Figura 2 es una aproximación simplista a la cantidad de conocimiento que se puede transmitir durante el ciclo de formación. El tipo de carrera determina cómo será la distribución del conocimiento.

[12] Aunque a mediados de 1970 la DPMA se unió a otras asociaciones y crearon el *Institute for Certification of Computer Professionals* (ICCP) para intentar relanzarlo.

[13] Aunque habrá cambios con los planes de Bolonia, en general la ingeniería informática española es una mezcla de ciencias de la computación, ingeniería de software y sistemas.

[14] El ejemplo de la ingeniería eléctrica anglosajona es similar en el caso español de las ingenierías industriales. Las escuelas de ingenierías industriales se crearon en 1850, estas ingenierías fueron luego las responsables de la *ingeniería eléctrica* (llamada en sus principio electrotecnia, sus primeros trabajos formales como tal se remontan a 1852).

[15] Hay expertos y académicos que consideran que un cisma de la informática con las matemáticas aplicadas sería un gran error [14].

[16] Por ello se sigue denominando "renacentistas" a los que dominan diversas técnicas o artes.

[17] No hay una escala precisa, el área total no tiene relación con el esfuerzo o tiempo necesario para el estudio.

Figura 1: El modelo renacentista

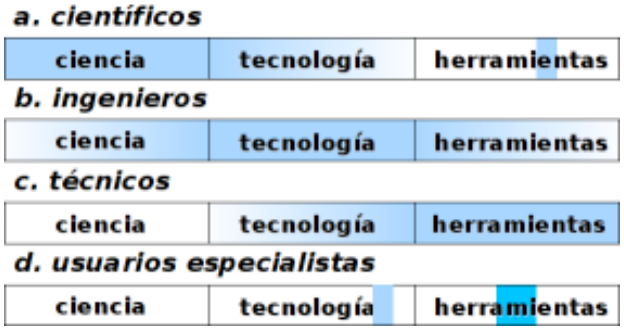

Figura 2: Conocimientos adquiridos durante la educación formal

Las carreras más científicas-vocacionales requieren de una mayor formación en conocimiento de ciencias y menos en tecnologías. Las ingenierías requieren mas formación en las tecnologías y en las herramientas. Las técnicas (FP o Ciclos formativos) prácticamente no estudian ciencia y dedican su tiempo a aprender más las herramientas (además son de duraciones más cortas que las dos anteriores). Por último existen cursos cortos, tanto de academias especializadas como de postgrados, cuyo objetivo es la de entrenar en herramientas muy específicas.

La situación real no es tan uniforme como la que se muestra en la Figura 2. En cualquier carrera, de cualquier universidad quedan inevitables *agujeros del conocimiento* (Figura 3).

Estos agujeros se producen por razones diversas:

- Culpa del profesor, por desconocimiento, no estar actualizado o no cumplir con los programas.
- Culpa de los planes de estudio, por no estar actualizados o no ser completos para el cuerpo de conocimiento requerido para una carrera.
- Culpa de los alumnos, que se han esforzado en saber sólo aquello que sabren será parte fundamental del examen, o en no hacer —es decir, copiar y plagiar— o participar en las prácticas (en general es culpa de la falta de motivación).
- Falta de tiempo, los imprevistos —enfermedad, conferencias, viajes— pueden provocar retrasos en las asignaturas que impidan enseñar o aprender adecuadamente todos los temas del programa original.

Esos agujeros no son tan graves si son pequeños. Un poco de estudio, incluso una simple lectura de sitios especializados, ayudan a cerrarlos muy rápidamente si se tiene toda la información contextual y de "hechos" [21] que sirven de soporte.

Pero en informática el problema es un poco más marcado. Además de que todavía continúan los debates sobre el conocimiento necesario, como la ciencia y la tecnología avanzan tan rápidamente los planes de estudios quedan casi obsoletos —o mejor dicho, incompletos— a los pocos años de aprobarse, tal como explican en [1]. Aunque cumplamos estrictamente los planes de la educación formal ya estamos dejando agujeros sin rellenar.

Esa distorsión de los planes de estudios se suele suplir con la buena voluntad de profesores, asignaturas optativas, cursos *ad hoc* y formación continua. Pero sobre todo se resuelve con alumnos muy motivados y un ambiente que permita el intercambio de conocimiento. La formación continua y autodidacta no sólo es importante para los profesores, también es imprescindible para los alumnos.

La formación de los académicos y profesionales no acaba al salir de la universidad. Así un científico tendrá que profundizar en ciertas áreas de su ciencia para ser capaz de comprender a fondo lo que se explica en los artículos especializados —*papers* y *journals*— y poder aportar nuevo conocimiento.

Los ingenieros informáticos tendrán que profundizar en áreas muy específicas de la ciencia, dominar determinadas tecnologías y aprender a fondo algunas herramientas (lenguajes, metodologías, etc). Además de ello deberán mantenerse al día sobre lo nuevo que está ocurriendo en los tres campos.

Así un profesional experimentado y con la especialización, entrenamiento y formación continua adecuada mantiene las "zonas coloreadas" como cuando acabó su carrera, pero además es un experto en áreas específicas (Figura 4). Aunque es un caso ideal, al alcance de muy pocos afortunados, los agujeros en el conocimiento —como los de la Figura 3— son difíciles de evitar.

Un profesional, aunque sea muy bueno y competente en su área específica, sin la formación continua imprescindible —formal o informal— a los pocos años desconocerá los últimos

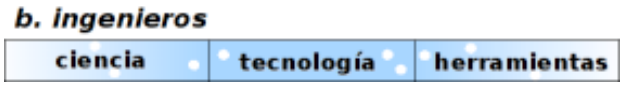

Figura 3: Hay pequeños agujeros de conocimientos

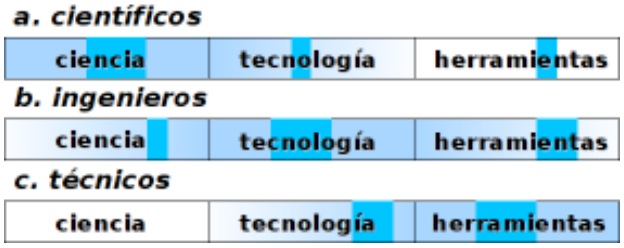

Figura 4: Los conocimientos de profesionales con experiencia y formación continua

avances y cambios del campo científico, tecnológico y hasta de las herramientas disponibles.

Para los informáticos la formación continua requiere de más esfuerzos. Influye en ello los rápidos cambios que se producen en la ciencia y tecnología, lo que hasta los expertos de la ACM denominan *pasos revolucionarios*. Pero también influye el mercado: las empresas son naturalmente conservadoras. Empresas conservadoras e ingeniería mutante hacen una combinación nada agradable. Así no es infrecuente encontrarnos con profesionales o ingenieros informáticos que no han tenido la oportunidad de recibir la formación continua adecuada.

El esquema conocimientos de estos profesionales es el que se muestra en la Figura 5.

Un buen ingeniero debería intentar tener los conocimientos distribuidos como en la Figura 4, y rechazar esquemas del tipo de la Figura 5 (es muy perjudicial para la persona y la organización para la que trabaja), aunque muchas veces la *natural disfunción metacognitiva* de los informáticos [5, 24] no nos permite evaluar la situación en que nos encontramos.

3.2. Lo que demandan las empresas

Hace unos pocos años tuvimos unos debates en la UIB sobre empresas y software privativo contra software libre. Participaban empresarios, profesores y alumnos. En una mesa redonda se discutió sobre este dilema. Uno de los empresarios de software más importante de Balears —además es licenciado en matemáticas, o sea que es conocedor del valor de la *formación científica*— afirmó:

> No sé para qué se enseña en la UIB cosas raras como sistemas operativos y motores de bases de datos: aquí nadie desarrollará un sistema operativo. Lo que tenéis que enseñar es SQL y Visual Basic. Eso es lo que necesita la empresa.

Este tipo de afirmaciones no es novedad, comenzaron desde el mismo momento en que las universidades incluyeron estudios de informática en sus programas. Ya en 1962 se podía leer [20]:

> Se observa una profundización del cisma de la programación en desarrollo en la industria, una brecha en crecimiento entre los científicos e ingenieros que hablan Algol y Fortran [...] y los de procesamiento de datos de empresas que hablan inglés y Cobol.

Aunque el nivel intelectual del debate no es comparable, sí denotan un patrón. Las empresas desean que un nuevo ingeniero en la empresa sea productivo en el menor tiempo posible. Es decir que dominen las tecnologías y herramientas particulares que su empresa usa o necesita.

Pero las necesidades de cada empresa son tan diversas que son imposibles de elaborar planes de estudios que las satisfaga inmediatamente aunque la universidad tuviese la intención de llevarlos a cabo. El tipo de ingeniero que necesita FON[18] —conocimientos de software de sistemas operativos, específicamente GNU/Linux, empotrados, telecomunicaciones y cifrado— no es el mismo que puede necesitar una empresa que hace software de gestión de hoteles, la de los bancos, las que hacen sistemas webs, o los que necesita Airbus para la aviónica.

Si hiciésemos caso de lo que piden las empresas, deberíamos redistribuir el conocimiento, y nos quedaría algo como la Figura 6. Un técnico debería ser experto en todas las herramientas que usan las empresas y conocer las tecnologías. Un ingeniero debería dominar las mismas herramientas y además ser igual de experto en las tecnologías.

Hay una aberración importante: para poder hacer que los ingenieros sean expertos en las tecnologías y herramientas se debería ahorrar tiempo en «cosas que no necesitan», como los conocimientos fundamentales. Pero aún hay otra más, cuatro o cinco años no son suficientes para enseñar a fondo todas las tecnologías y herramientas, así que hay dos soluciones:

- Formar ingenieros como los de la Figura 5, especialización de las carreras para «cubrir las necesidades regionales». Aunque en principio no es una mala idea, también tiene

[18] Empresa fundada por Martín Varsavsky, fabrican puntos de acceso WiFi y dan servicios en diversos países, FON `http://fon.com`

Figura 5: Conocimientos de un ingeniero informático que no recibe formación continua

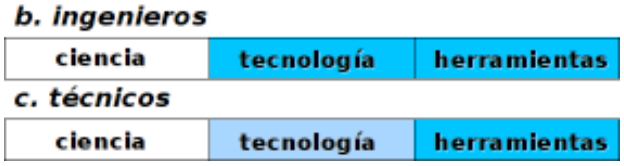

Figura 6: Lo que suelen pedir las empresas

sus problemas: ¿qué debería enseñarse? Las empresas de la misma región no tienen necesidades tan uniformes y habría dificultades para la movilidad profesional. De todas formas no es descabellada sino bastante habitual la idea de formar profesionales mejor adaptados a las necesidades regionales y con conocimientos no *externalizables* [18].

- Formar ingenieros como los de la Figura 6, lo que significaría aumentar el número de años de carrera. No tiene sentido tampoco, porque a la velocidad que progresan las tecnologías cuando hayan acabado la larga carrera la mitad de lo aprendido ya será obsoleto. Las quejas de los empresarios serían exactamente las mismas.

3.2.1. La infantilización de los planes de estudios

Hace ya veinte años Dijkstra se quejaba de que las universidades ya que en su intento por ofrecer carreras más atractivas por su salida profesional, estaban cediendo a las presiones de la industria [8]:

> [...] veo sobrecogedoramente la deprimente imagen de «lo de siempre»[19]. A las universidades les seguirá faltando el coraje de enseñar ciencia dura, continuará orientando mal a los estudiantes, y cada nuevo escalón de infantilización del currículum será exaltado como progreso educativo.

En 2005 la ACM también denunciaba la aparición de títulos de grado que cedían antes las *fuerzas del mercado* [2]:

> Las fuerzas del mercado impactan en los programas académicos de diferentes formas [...] Cuando instituciones que otorgan títulos oficiales se asocian con programas de proveedores, la integridad académica se convierte en un tema importante a tener en cuenta. El lector debería ser consciente que tales asociaciones son una invitación a la controversia acerca de la ética e integridad académica [...]
>
> El hecho de que aparezcan nuevas titulaciones para solucionar necesidades sociales muestra que el dinamismo de nuestra sociedad no se diluye en la academia. [...] Lastimosamente la misma emergencia de nuevos programas de grado también son fuente de ejemplos de lo que se puede hacer mal cuando las iniciativas académicas están dirigidas por intereses políticos, imperativos de la administración y la propaganda exagerada de los medios más que por el reconocimiento que hace falta innovaciones sustantivas para dar respuestas a las preocupaciones sociales.

Este tipo de discusiones no son nuevas ni exclusivas de la "academia". El reconocido programador y emprendedor Joel Spolsky también hacía una autocrítica y se quejaba del mismo problema en *The Perils of JavaSchools* [25]:

> Ojalá no me hubiesen escuchado [...] Allí reside el debate. Los años de lloriqueo de estudiantes de informática vagos como yo, combinado con las quejas de la industria sobre la formación de los graduados informáticos de las universidades norteamericanas hicieron mella, y en la última década un gran número de antes buenas facultados se han hecho 100 % Java. Es la moda, parece gustar a los reclutadores que usan *grep* para evaluar los curriculums, y lo mejor de todo es que no hay nada especialmente malo en [el lenguaje] Java que obligue abandonar a los programadores que no tengan idea de punteros o recursión, así el abandono es más bajo, los departamentos de informática tienen más estudiantes y mayores presupuestos, todo va bien.

Spolsky describía así a ingenieros informáticos con un esquema de conocimiento similar al de la Figura 5 graduados en universidades que se centraron más en tener planes de estudios atractivas y con un fuente componente de Java —de allí el nombre *javaschools*—.

[19] *Business as usual*

3.3. El problema español

Como ya es evidente, el problema de la aparente *falta de sintonía* no es un problema sólo español. Tampoco creo que haya excesivas diferencias en la formación y esquema de conocimientos adquiridos por los alumnos, al menos en lo que se refiere a la *distribución del conocimiento*. Cualquiera que haya visitado otras universidades habrá visto que los temas son muy similares, incluso los libros de texto son los mismos en la mayoría de los casos. Sólo que como vivimos aquí nos parece un problema creado y padecido exclusivamente en nuestras universidades.

Afortunadamente, con excepciones y matices, la mayoría de las facultades informáticas españolas han podido resistir a la tentación de convertirse en *javaschools*. Pero no estamos libres de problemas relacionados con la formación. Por los recientes debates sobre Bolonia y las peticiones de regulación de la ingeniería informática parece claro que en muchas facultades se ha hecho un abuso del *razonamiento por analogía*.

En nuestro deseo por parecernos más a las las ingenierías tradicionales no sólo hemos adoptado el controvertido nombre de *ingeniería del software*[20] [8] sino que además en nuestros discursos y debates académico-profesionales nos hemos habituado a extralimitarnos aún más con la analogía. Hemos asumido como verdad absoluta la similitud entre el proceso de desarrollo de software con el de diseño y construcción de un proyecto arquitectónico. Así muchos graduados informáticos parecen haber asumido su papel de *arquitecto* (el *ingeniero*) contra el *albañil programador* (el *picacódigos*).

Esta errónea simplificación derivado del abuso de la analogía no sólo crea una errónea y perjudicial dicotomía diseño-programación que nos ha convencido que el resultado de aplicar una receta —la documentación aplicando alguna metodología formal de diseño— es el resultado que esperan de nosotros, también nos impide contemplar la perspectiva y el detalle de las complejidades y diferencias de nuestra profesión comparada con las tradicionales. Mientras en nuestro proceso de construcción tenemos que diseñar y escribir complejas fórmulas de cálculo de predicado sin evaluar —un complejo programa sin métodos conocidos de validación formal—, la arquitectura "sólo" tiene que lidiar con objetos cuyas combinaciones y propiedades bien estudiadas durante siglos están muy limitadas por las inviolables leyes físicas. Como dice Jack Reeves [19], una analogía más razonable entre la informática y la arquitectura es que nuestro documento final de diseño es el programa.

Por otro lado, las universidades españolas, sobre todo las de "provincias" como la UIB, tienen problemas estructurales importantes:

- Problemas de **mercado**: la industria del software española es relativamente muy pequeña, hay pocas empresas, pocas inversiones en este campo, con I+D+I casi inexistente. Obviamente afecta a las universidades, su relación con empresas, falta de profesores con experiencias en grandes proyectos empresariales, y las aspiraciones y motivación de los alumnos.

- Problemas **geopolíticos**: La "gran industria" informática está muy centralizada —sus índices de concentración son muy superiores a la de los fabricantes de coches, las segundas más elevadas según el censo norteamericano—. No es lo mismo ser una universidad en Balears o Murcia que una cuyo campus esté insertada en Silicon Valley. Las industrias de sus entornos son incomparables —unas cincuenta universidades—, la atención y fondos directos e indirectos son diferentes, lo que nos lleva al siguiente problema.

- Problemas de **recursos**: mientras que una universidad norteamericana de primer orden —de la *Ivy League*— tiene un presupuesto por alumno ronda los 30.000 €, una universidad española difícilmente supera los 6.000 €. Afecta a los recursos disponibles y sobre todo a la masificación de clases. Afortunadamente el hardware bajó mucho de precio, también el software que hace falta —casi se puede enseñar toda una carrera sólo con software libre—, pero los recursos humanos son los más caros, alrededor del 70 % de los ya reducidos presupuestos son dedicados a este apartado. Además se suma otro problema, los salarios de profesores son en general sustancialmente más bajos de los que paga la industria con una formación similar, lo que hace que sea muy difícil convencer a los mejores estudiantes para que continúen una carrera académica.

Creo que esos últimos problemas no justifican por sí mismos una especial *falta de sintonía* con las empresas, o al menos no parece que la mejora de las situaciones mencionadas anteriormente solucionen el problema automáticamente. No podríamos intentar aumentar recursos sin mejorar el mercado que a su vez necesitará de ingenieros de primer orden. No podremos tener nuestros Silicon Valleys sin una buena dotación de empresas, científicos, ingenieros y grandes cantidades de capital riesgo.

Los problemas que sí estamos en disponiblidad de solucionar probablemente sean de otra índole, que afecta a profesores y alumnos pero no son exclusivos de las universidades:

1. **Falta de motivación** (o de pasión por la informática) de alumnos y profesores, pero fundamentalmente de los primeros que son los al fin y al cabo los que saldrán al mercado laboral.

2. **Falta de confianza** de los graduados sobre sus capacidades personales y la seguridad que «no hay misterios, es sólo ciencia y matemáticas, si otros lo pudieron hacer yo también puedo hacerlo». Este problema no sólo afecta a cómo se enfrentan y resuelven los problemas dentro de una empresa, sino en que hay muy pocos ingenieros que inicien proyectos innovadores por cuenta propia y con objetivos globales. La causas fundamentales podrían ser:

    - La falta de *entrenamiento mínimo* de desarrollo en grupo de programas complejos y de varios cientos de miles de líneas .

[20] Mencionado por primera vez en 1965 por Presper Eckert, se popularizó desde que fue usado por la OTAN en la *NATO Conference on Software Engineering* en 1968 [9].

- Muy en línea con [8, 22, 25], la falta de formación, práctica y comodidad en ciencia dura o matemáticas aplicadas [7].
- Carencia de mentalidad emprendedora.

3. **Las empresas no desean invertir** en el entrenamiento de sus nuevos profesionales. Es un problema empresarial, la competencia, la *externalización* y los mercados reducidos hacen que parezca muy caro que las empresas se hagan cargo del entrenamiento inicial y luego de la formación continua de sus ingenieros.

4. Conclusión

La profesión informática todavía no ha tenido el tiempo de maduración de otras ingenierías que le haya permitido definir los estándares de calidad técnicos y éticos habituales en profesiones más antiguas. Todavía nos espera un largo camino en este sentido, las facultades podrían facilitar y favorecer el debate introduciendo asignaturas o estudios específicos y así evitar falsas expectativas y debates vacíos y estériles (como reclamar regulaciones legales de la profesión informática, en contra de la experiencia de la historia, la doctrina europea y las tendencias de los países más desarrollados tecnológicamente).

Los debates de la falta de sintonía entre universidad y empresa no es sólo español. La *fuerza del mercado* en muchos casos ha generado que se *infantilicen* los planes de estudio de universidades obligadas a aumentar o mantener las matriculación de alumnos. Aunque este fenómeno no parece haber sido reproducido masivamente en la universidad pública española, sí que hay una cierta preocupación por la falta de formación en ciencia dura y matemáticas que son habituales en universidades prestigiosas.

La *falta de confianza* es quizás un problema más social y económico que académico. No existe la cultura emprendedora ni el entorno de empresas tecnológicas que ha sido la que ha generado regiones poderosas como Silicon Valley (y últimamente Irlanda debido a una dinámica política de atraer este tipo de empresas con grandes beneficios fiscales).

Una parte fundamental de cualquier profesión es el entrenamiento para la práctica profesional responsable. En algunas carreras —como en medicina— este entrenamiento mínimo está asegurado en los largos planes de estudio y especialización. En otras como abogacía o arquitectura es habitual que el nuevo profesional se integre y trabaje en equipos de trabajo más maduros antes de asumir responsabilidades profesionales más delicadas. En informática por el contrario se espera que el profesional sea productivo e independiente en muy poco tiempo. Quizás la universidad debería favorecer o mejorar este *entrenamiento profesional especializado,* pero es un tema a discutir con empresarios para analizar y financiar cursos de formación o posgrados más específicos[21].

Desde el punto de qué se puede hacer desde la universidad creo que el mejor camino a corto plazo es mejorar la motivación de los alumnos. En este sentido hay diversos reclamos de un cambio estructural importante de la universidad pública española. Todas las propuestas alternativas se reducen a dos modelos fundamentales:

Universidades de élite. Aunque el impacto social sería enorme, la propuesta es sencilla. Si de 100 alumnos dejamos entrar sólo a los 10 mejores en vez de a 80, nos aseguramos de entrada que reclutamos a los mejores y más motivados. El problema de este modelo es el fuerte impacto social negativo que generaría al generar una escasez de profesionales —lo sufre Estados Unidos—. Tampoco serviría para cubrir las necesidades de las empresas locales, luego deberíamos importar ingenieros informáticos de otros países para cubrir las vacantes.

Matrículas altas y más becas. La idea es mantener un "balance neutral" —como el de ahora— pero los estudiantes "regulares" deberían pagar de matrícula un porcentaje elevado de los costes, dinero que sería invertido en más becas y becas-salario a esos buenos alumnos que en muchos casos empiezan a trabajar antes de acabar sus estudios —bien por necesidad o porque reciben muy buenas ofertas— que termina afectando muy negativamente a su rendimiento académico. El hecho de tener que dedicarse más al estudio para no tener que pagar, o pagar menos, haría que su buen rendimiento tuviese consecuencias económicas inmediatas.

Parece obvio que el modelo de mejores becas con efectos económicos más inmediatos es el más razonable. Los resultados no podrán verse a corto plazo y requiere un fuerte compromiso político y empresario previo a todos los niveles sin la pretensión de convertir a la universidad en una *escuela de la herramienta de moda*.

Referencias

[1] ACM. *Computing Curricula 2001 Computer Science*. The Joint Task Force on Computing Curricula IEEE Computer Society, 2001.

[2] ACM. Computing Curricula 2005: The Overview Report. `http://www.acm.org/education/education/curric_vols/CC2005-March06Final.pdf`

[3] ACM. *Software Engineering Code of Ethics (Spanish)*, URL: `http://www.acm.org/about/se-code-s/`

[4] ACM. *Curricula Recommendations*. URL: `http://www.acm.org/education/curricula-recommendations`

[5] Phillip G. Armour. *The five orders of ignorance*. Communications of the ACM Volumen 43, número 10. October 2000.

[6] Antony Black. *Guilds and Civil Society in European Political Thought*. Methuen young books, 1983.

[21] El problema central en estas discusiones es lo que popularmente se conoce como la *mercantilización de la universidad*, que básicamente viene a significar la tendencia en la formación de usos de herramientas específicas y no en la enseñanza de conocimientos científicos y tecnologías fundamentales.

[7] Rodolfo Carpintier. *España, la California de Europa*. http://rodolfocarpintier.com/post/2008/12/14/espana-california-europa

[8] Edsger W. Dijkstra. *On the Cruelty of Really Teaching Computer Science*. EWD1036-0. 1988. URL: http://www.smaldone.com.ar/documentos/ewd/sobre_la_crueldad.html

[9] Nathan L. Ensmenger. *The "Question of Professionalism" in the Computer Fields*. IEEE Annals of the History of Computing. pp. 56–74. Octubre - diciembre 2001.

[10] Abraham Flexner. *Is social work a profession?* Proceedings del National Conference on Charities and Correction, 1915.

[11] Abraham Flexner, Clark Kerr. *Universities: American, English, German*. Transaction Publishers, 1994.

[12] Ricardo Galli. *Sintonizar universidades y empresas, pero ¿qué debe saber un ingeniero?* URL: http://fon.gs/conocimientos-ingenieros/

[13] IEEE. *IEEE Code of Ethics*, URL: http://www.ieee.org/portal/pages/iportals/aboutus/ethics/code.html

[14] Janusz Kowalik. *The Applied Mathematics and Computer Science Schism*. Computer, Volumen 39, número 3, pp. 103 – 104. Marzo 2006

[15] E.A. Krause. *Death of the Guilds: Professions, States and the Advance of Capitalism, 1930 to the Present*. Yale University Press, 1999.

[16] Thomas S. Kuhn. *La estructura de las revoluciones científicas*. Breviaros, Fondo de Cultura Económica. ISBN 84-375-0046-X

[17] Jim Moore. *ISO 12207 and Related Software Life-Cycle Standards*. The MITRE Corporation. URL: http://www.acm.org/tsc/lifecycle.html

[18] Mike O'Neal. *Restructuring Computing Programs to Meet Employment Challenges*. IEEE Computer, Volumen 37, número 11. Noviembre 2004

[19] Jack W. Reeves. *Code as Design: Three Essays*. http://www.developerdotstar.com/mag/articles/reeves_design_main.html

[20] C. Shaw, *Programming Schisms*. Datamation, Volumen 8, número 9, p. 32. Septiembre 1962.

[21] Mary Shaw. *Prospects for an Engineering Discipline of Software*. IEEE Software. Volumen 7, número 6, pp. 15 – 24. Noviembre 1990.

[22] C. M. Sidlo. *The making of a profession*. Communications of the ACM. Volumen 4, número 9, pp. 366 – 367. Septiembre 1961

[23] Raghu Singh. *An Introduction to Iinternational Standard ISO/IEC 12207 Software Life Cycle Processes*, 1999. URL: http://www.abelia.com/docs/12207cpt.pdf

[24] Deborah K. Smith, Trevor Moores, Jerry Chang. *Prepare your mind for learning*. Communications of the ACM. Volumen 48, número 9. Septiembre 2005.

[25] Joel Spolsky. *The Perils of JavaSchools*. Diciembre, 2005. URL: http://www.joelonsoftware.com/articles/ThePerilsofJavaSchools.html

[26] Gregory A. Stobbs. *Software Patents, Second Edition*. Aspen Publishers, 2008.

[27] Martín Varsavsky. *El Paro, el problema número uno de España se agrava* URL: http://fon.gs/paro-agudiza/

Ricardo Galli es Doctor en Informática, profesor de la Universitat de les Illes Balears. Fundador y programador de Menéame, desarrollador y promotor de software libre. Tiene más de 40 publicaciones científicas nacionales e internacionales en el área de Informática Gráfica 3D, trabajo colaborativo y temas relacionados con GNU/Linux y software libre. Ha participado y dirigido varios proyectos de I+D europeos, también ha sido contratado por la Comisión Europea como experto independiente para la evaluación de proyectos de I+D. Es portavoz oficial de la Free Software Foundation, miembro y ex presidente de Bulma, la asociación de GNU/Linux de Balears. Socio fundador en 1995 del ISP Atlas Internet. Ha puesto en marcha periódicos digitales y emisoras de radios con software libre. Ha dirigido varios cursos y postgrados de software libre. En 2003 ganó el premio de investigación "Sa Nostra 2003" con el proyecto *El Software Libre y la regionalización de los beneficios de la Sociedad de la información*. En diciembre de 2005 puso en marcha Menéame, cuyo código está publicado también como software libre. Desde 1998 escribe en el weblog colectivo de Bulma, desde 2004 tiene su propio blog personal orientado a temas de software libre.