Motivación para la Ingeniería de Computadores

Julio Ortega, Mancia Anguita, Miguel Damas, Jesús González
Departamento de Arquitectura y Tecnología de Computadores
E.T.S.I.I.T., Universidad de Granada
{jortega, manguita, mdamas, jesusgonzalez}@ugr.es

Resumen. Los estudios de Ingeniería de Computadores proporcionan competencias que permiten participar en el desarrollo de los actuales y futuros sistemas de cómputo, con una incidencia importante en el entorno socio-económico, y por tanto con posibilidades profesionales relevantes. Este artículo incluye consideraciones y argumentos para facilitar, a los profesores involucrados en la docencia de asignaturas relacionadas con la Ingeniería de Computadores, la labor de motivación de los estudiantes. En particular se proporcionan algunas estrategias docentes para la motivación y se ilustra su uso en la docencia del lenguaje ensamblador y de los modelos de consistencia de memoria.

Palabras clave: Aplicación de las teorías sobre la motivación, Docencia de lenguaje ensamblador, Ingeniería de computadores, Modelos de consistencia de memoria, Motivación intrínseca.

Abstract. Computer engineering courses and degrees provide competencies to be involved in the development of present and future computer systems, with a relevant effect in social and economical contexts, and interesting professional opportunities for students. This paper provides some considerations and arguments to make it easier for teachers involved in subjects related with computer engineering topics to motivate their students. More specifically, the use of some motivation strategies for teaching assembly languages and memory consistency models are described.

Keywords: Applications of motivation theories, Teaching assembly language, Computer engineering, Memory consistency models, Intrinsic motivation.

1 Introducción

¿Por qué en los estudios de Ingeniería Informática, las asignaturas, intensificaciones, especialidades, etc. más próximas al hardware suelen ser menos demandadas por los estudiantes? Una de las causas posibles surge de la percepción de que, en nuestro contexto socio-económico, se necesitan menos profesionales con las competencias

asociadas a esos estudios. Por otro lado, existe una perspectiva de la Ingeniería Informática que, desde una descripción del computador como una serie de capas, considera que las competencias relacionadas con el software podrían adquirirse, incluso al nivel propio de un ingeniero, prácticamente sin conocimientos del hardware o de la arquitectura del computador.

De hecho, se puede decir que en la Informática han confluido dos corrientes, la relacionada con lo que en inglés se conoce como "computer science", y la que tiene que ver con la denominada "computer engineering", o incluso "electrical and electronic engineering". Desde cierto punto de vista, el hardware se contempla como algo relacionado solo tangencialmente con la informática. Considérese como ejemplo, el uso de adjetivos tan poco científicos, como "hermosa" o "fea", que se pueden leer en la página 5 de la traducción al castellano de la tercera edición del libro "Sistemas Operativos Modernos" de Tanembaum [1]. Se indica allí que una de las tareas esenciales del sistema operativo es "ocultar el hardware y presentar a los programas (y a los programadores) abstracciones agradables, elegantes, simples y consistentes con las que trabajar […] ocultando la parte fea con la parte hermosa como muestra la figura 1.2". Sin embargo, la evolución de la tecnología de computadores ha dado lugar a una influencia cada vez más estrecha de las características del hardware y la arquitectura del computador en la programación y el desarrollo de software. Por lo tanto, se necesitan actuaciones docentes. Por otro lado, es cierto que parece haber una menor demanda de puestos de trabajo que solicitan competencias específicas propias de la especialidad de ingeniero de computadores respecto a competencias más relacionadas con el desarrollo de software. Por tanto, la justificación del interés de la Ingeniería de Computadores debe abordarse desde planteamientos que pongan de manifiesto la relevancia de los contenidos de la especialidad, no solo en el presente sino para el futuro. Es importante hacer valer las posibilidades que encierra la perspectiva de sistema que ofrece la Ingeniería de Computadores y, por supuesto, fomentar el aspecto emprendedor en los estudiantes de Ingeniería de Computadores.

En este artículo se proponen algunas estrategias docentes que, a partir de las claves que proporcionan algunas de las teorías de la motivación más relevante propuestas hasta el momento, pueden motivar el estudio de las asignaturas de tecnología, estructura y arquitectura de computadores. Si los estudiantes consideran los contenidos de estas asignaturas asequibles y útiles para su formación como Ingenieros Informáticos es más probable que la Ingeniería de Computadores sea contemplada por algunos como su profesión. Así, tras la revisión de las teorías de la motivación y del proceso de aprendizaje que proporcionamos en la Sección 2, se describen las posibilidades de la Ingeniería de Computadores para desarrollar una carrera profesional de futuro, y adquirir competencias que fomentan la iniciativa emprendedora y la innovación en el contexto de la evolución tecnológica. Para ello, en la Sección 3 se dan algunos datos acerca de las ofertas de trabajo para profesiones relacionadas con las Tecnologías de la Información y las Comunicaciones, y se analizan las posibilidades que se contemplan en un futuro tanto desde el punto de vista de las empresas como del ámbito de la investigación. Después, en la Sección 4 se describen algunas de las perspectivas de futuro para las TIC, el punto de vista desde el que la Ingeniería de Computadores contempla a la Informática, y de qué

forma puede tener un papel relevante en la evolución hacia las nuevas aplicaciones y posibilidades previamente descritas. En la Sección 5 se proporcionan algunas estrategias docentes que pueden plantearse a partir de las teorías del aprendizaje y la motivación y de lo expuesto en las secciones anteriores. También se describe en esta sección la aplicación de esas estrategias en algunos contenidos de asignaturas de Estructura y Arquitectura de Computadores impartidas en el Grado de Ingeniería Informática de la Universidad de Granada y cursadas por los estudiantes antes de la elección de especialidad. Finalmente, la Sección 6 recoge las conclusiones del artículo.

2 El aprendizaje y la motivación

Para proponer estrategias docentes que promuevan la demanda de los estudios de Ingeniería de Computadores (Sección 5) se han tenido en cuenta las conclusiones de una serie de trabajos, recientes y/o ampliamente citados, acerca de cómo y en qué circunstancias el cerebro absorbe y almacena información, incidiendo por tanto positivamente sobre el aprendizaje [2, 3]. Estas conclusiones se pueden resumir a través de siete principios:

1. Cuando inician el estudio de una asignatura, los estudiantes parten de ciertas concepciones previas de lo que les rodea, incluyendo los contenidos de la asignatura. Las ideas de partida sobre la asignatura afectan (facilitando o dificultando) a su aprendizaje.
2. Para asimilar nuevas competencias no es suficiente conocer muchos datos sino que hace falta incorporarlos dentro de un marco conceptual que organice e interconecte los hechos para que sea fácil acceder a ellos y aplicarlos. La forma según la que los estudiantes organizan el conocimiento influye en lo que aprenden y en la forma en que lo utilizan.
3. La motivación determina, orienta y mantiene el aprendizaje de los estudiantes.
4. El conocimiento propio de los expertos se consigue integrando y practicando con los elementos que constituyen el campo de trabajo de forma que se adquieren habilidades y se sabe cuando aplicar lo aprendido.
5. Las actividades prácticas orientadas a alcanzar objetivos, junto con la realimentación acerca de lo que se ha alcanzado incrementa la calidad del aprendizaje.
6. El aprendizaje se ve afectado por el clima social, emocional, e intelectual de la clase.
7. Para conseguir el aprendizaje autodirigido, los estudiantes deben ser capaces de estimar la dificultad de la tarea en cuestión, evaluar sus propias habilidades y conocimientos, planificar su trabajo, analizar su progreso y ajustar sus estrategias convenientemente. Es conveniente proporcionar información “metacognitiva” que permita al estudiante ser consciente de su aprendizaje y del logro de las metas establecidas.

Según lo anterior, es importante tener en cuenta la visión previa que los estudiantes tienen de las asignaturas, relacionadas con la Ingeniería de Computadores, que preceden a la elección de especialidad (en el Grado en Ingeniería de Computadores de la Universidad de Granada serían Tecnología y Organización de Computadores, Estructura de Computadores, Arquitectura de Computadores, e Ingeniería de Servidores). En estas asignaturas no se estudia el computador buscando únicamente alcanzar competencias relacionadas con el diseño hardware, sino que se aportan conceptos esenciales para entender el funcionamiento del computador, de los sistemas operativos, y del desarrollo de programas eficientes.

El tercero de los siete principios indicados hace referencia a la importancia de la motivación en el aprendizaje. Dado que la motivación es un estado interno que promueve y controla la actividad de un individuo hacia un objetivo, motivar a alguien es estimularlo para que encamine sus actividades hacia ese objetivo. A continuación se resumen las características fundamentales de algunas de las teorías de la motivación más relevantes [4]: la teoría de los objetivos, la de las atribuciones, la de expectativa-valor, y la de la autodeterminación. Como se pone de manifiesto a través de dichas teorías, en la motivación influyen muchos factores.

La teoría de los objetivos relaciona la motivación con las características de los objetivos planteados. Así, dichos objetivos se califican de acuerdo con su *naturaleza* (se busca ser competente en algo o ser el que mejor calificaciones obtiene) o con su *orientación* (se busca alcanzar algo o se pretende evitar algo que no se desea). Combinando las dos posibilidades de los objetivos en cuanto a naturaleza con las dos orientaciones se tienen cuatro alternativas de motivación de la actividad. No todas esas alternativas son igual de efectivas para el aprendizaje. Así, los objetivos que pretenden alcanzar destrezas o dominar tareas producen, con mayor probabilidad, un aprendizaje más profundo que si buscan a obtener buenas calificaciones. Por otra parte, también es mejor si la acción se dirige a alcanzar algo que a huir de algo. Es decir que, según esta teoría, motiva más el deseo de adquirir una habilidad que el de evitar un suspenso.

La teoría de las atribuciones propone que la motivación de las personas está relacionada con sus expectativas de alcanzar un objetivo, ya que si alguien cree que no podrá conseguir algo, muy posiblemente no intentará alcanzarlo. Las causas del éxito o del fracaso se pueden deber a razones internas de la persona o externas a ella, que pueden cambiar con el tiempo o se mantienen invariables (dinámicas o permanentes), y son o no son controlables por la persona. Por ejemplo, mientras que el esfuerzo se puede catalogar como una causa interna, variable, y controlable; la dificultad se contempla como una causa externa, permanente y no controlable; y la suerte como externa, inestable, y no controlable. Es más probable que un estudiante persevere en una actividad y alcance el éxito en una tarea si se evita que dicho éxito se relacione con causas externas, permanentes, o incontrolables.

La teoría de expectativa-valor establece que la motivación hacia algo aumenta cuando existe la creencia de que se puede alcanzar el objetivo en cuestión (expectativa) y merece la pena hacerlo (valor). En cuanto a las expectativas de éxito en los objetivos,

éstas dependen de la percepción que se tiene acerca de la dificultad que entraña alcanzarlos, y de la capacidad que uno considera que tiene para ello. También relacionados con esta teoría, en [5] se describen tres factores importantes que determinan la motivación de las personas hacia algo: (a) la forma en que cada uno contempla sus habilidades o capacidades para hacer ese algo; (b) lo que se cree importante, interesante o útil; y (c) los sentimientos y emociones que suscita el objetivo a alcanzar. Por tanto, desde este punto de vista, la motivación se consigue poniendo de manifiesto la relevancia de los objetivos a alcanzar y creando entornos docentes en los que los estudiantes tengan la sensación de que pueden completar las tareas que se les planteen.

La teoría de la autodeterminación distingue entre dos tipos de motivación. La motivación *extrínseca*, cuando el estímulo para la acción proviene de factores externos a la persona, como por ejemplo contemplar la posibilidad de encontrar puestos de trabajo bien remunerados a la hora de elegir una determinada profesión, y la motivación *intrínseca*, en la que las razones para encaminar la actividad hacia el objetivo en cuestión están relacionadas con lo que, internamente, la persona considera interesante o importante, como por ejemplo, puede considerarse un objetivo intrínseco ser partícipe en el desarrollo de herramientas y técnicas que pueden abrir nuevos caminos y posibilidades para la humanidad. Desde hace bastante tiempo se han sucedido trabajos que buscaban las razones por las que una actividad puede ser intrínsecamente motivadora. En [6] se pueden encontrar una presentación resumida de los más relevantes, junto con las correspondientes referencias. Según algunas de las aproximaciones descritas, lo que motivaría a las personas está relacionado con el grado de control que pueden alcanzar en su relación con otras personas, con objetos, y con ellos mismos, es decir lo que se denomina *interacción efectiva*. Así, hay tres necesidades psicológicas básicas en la persona que condicionan la motivación hacia lo que las satisface [4, 7]: la autonomía (sensación de libertad y control sobre nuestra actividad); la competencia (poder ejercitar y poner de manifiesto las propias capacidades haciendo algo que se considera realmente importante para la sociedad); y la socialización (sentirse conectado socialmente). Por tanto, la relevancia de los problemas que se plantean en una disciplina y la utilidad de las soluciones constituyen la base de la motivación intrínseca hacia dicha disciplina. En trabajos heurísticos, donde interviene fundamentalmente el hemisferio derecho y en cuyo desarrollo juega un papel central la creatividad, como es el caso de las ingenierías, la motivación intrínseca juega un papel relevante frente a la motivación extrínseca [7].

A continuación, en las Secciones 3 y 4, se analizan aspectos que afectan a la motivación intrínseca y extrínseca para estudiar Ingeniería de Computadores. Así, mientras que la Sección 3 proporciona información acerca del mercado de trabajo en Informática y en Ingeniería de Computadores, como elemento importante para la motivación extrínseca, la Sección 4 proporciona algunas de las perspectivas futuras para el desarrollo profesional de un Ingeniero de Computadores, que van a ser clave para la motivación intrínseca de los estudiantes. Después, en la Sección 5 se proporcionan estrategias docentes que motiven a los estudiantes en el estudio de las asignaturas relacionadas con la Ingeniería de Computadores (las previas a la elección de la especialidad y las propias de la especialidad). Para establecer esas estrategias se

ha tenido en cuenta las conclusiones acerca de la relevancia de los contenidos y los entornos docentes de la teoría de expectativa-valor.

3 La demanda de Ingenieros de Computadores

En el Computer Curricula de 2005 de ACM e IEEE [8] se establecen cinco perfiles o disciplinas de computación: Ingeniería de Computadores, Ingeniería del Software, Tecnología de la Información, Sistemas de Información, y Ciencia de la Computación. Para establecerlos se tuvo en cuenta la existencia de guías curriculares publicadas para el perfil correspondiente por una o más sociedades científicas o profesionales. Estos perfiles coinciden (salvo por el uso del plural en el perfil de Tecnología de la Información y por el añadido de Sistemas Inteligentes y el uso de Computación en lugar de Ciencia de la Computación) con las especialidades que contempla el título de Grado en Ingeniería Informática de la Universidad de Granada (aprobado oficialmente en 2010). Una descripción detallada de las especialidades propuestas por IEEE/ACM se tiene en [8], en tanto que las características de las especialidades en el Grado en Informática de la Universidad de Granada están en [9].

En [10, 11] se pueden consultar distintos análisis del sector TIC en Andalucía elaborados por SANDETEL (Sociedad Andaluza para el Desarrollo de las Telecomunicaciones). Según el informe del año 2010 [10], un 83.5% de los 32.000 puestos de trabajo del sector TIC correspondía a empleados con formación superior, y comprendía más de 1600 empresas (un 10% de esas empresas eran granadinas, lejos del 36% y del 27% correspondientes a Sevilla y Málaga, respectivamente), con una facturación de 4200 millones de euros. El 59% de las empresas realizaban actividades de I+D+i, cuya financiación promedio había sido del orden del 16.6% de la facturación, y provenía en un 68% de la propia empresa, y en un 25% de fuentes públicas. Teniendo en cuenta la distribución de actividades que hace la CNAE (Clasificación Nacional de Actividades Económicas), alrededor del 69% de las empresas se dedicaban a la "programación, consultoría y otras actividades relacionadas con la informática", el 46% ofrecían "servicios de portales web o proceso de datos", y el 17% se dedicaba a la "fabricación de productos informáticos, electrónicos y ópticos". En [9] se proporciona una descripción análoga para toda España.

Es posible tener una referencia bastante completa de la demanda de profesionales en tecnologías de la información y las comunicaciones en EE.UU. a través de la página *Bureau of Labor Statistics* (http://www.bls.gov/ooh/home.htm). En la Tabla 1 se muestran las ofertas de trabajo y los salarios anuales en 2010 para las profesiones que se recogen en dicha página dentro del campo Tecnologías de la Información y las Comunicaciones, más el campo Ingeniería de Hardware de Computador. En particular, en este último campo se especifica que la mayoría de los empleados tienen un grado en Ingeniería de Computadores. Se puede ver que profesiones con menos ofertas de trabajo suelen tener salarios mayores

Una primera cuestión que plantea la distribución de actividades de la CNAE utilizada en [10, 12], es que es difícil distinguir demandas de profesionales que correspondan a las competencias específicas de cada una de las distintas especialidades ofertadas en el grado de Ingeniería Informática de la Universidad de Granada, bastante usuales, por otra parte, en los grados de Ingeniería Informática de otras Universidades (incluso existen como titulaciones de grado en alguna de ellas): Ingeniería del Software, Computación y Sistemas Inteligentes, Sistemas de Información, Tecnologías de la Información, e Ingeniería de Computadores. Incluso en el caso de las profesiones que considera el *Bureau of Labor Statistics* en EE.UU., no existe una correspondencia clara (salvo quizá en el caso de la Ingeniería de Computadores). Así, si consideramos la clasificación de la CNAE y centrándonos en las competencias de un Ingeniero Informático con la especialidad de Ingeniería de Computadores [9] , éstas parecerían más directamente relacionadas con las del apartado de "fabricación de productos informáticos, electrónicos y ópticos", pero también tendrían cabida en todas aquellas empresas que requieren desarrollar software para productos embebidos, desarrollar software eficiente, configurar y mantener equipos informáticos, o las que proporcionan el acceso a dichos recursos desde el punto de vista del paradigma actual del *cloud* [13].

Tabla 1. Salarios y ofertas de trabajo en TIC en 2010 según el *Bureau of Labor Statistics* (EE.UU.)

Profesión	Salario Anual (2010) ($)	Ofertas de trabajo (2010)
Computer and Information Research Scientists	100.660	28.200
Computer Support Specialists	46.260	607.100
Software Developers	90.530	913.100
Information Security Analysts, Web Developers, and Computer Network Architects	75.760	302.300
Database Administrators	73.490	110.800
Computer Programmers	71.380	363.100
Computer Systems Analysts	77.740	544.400
Network and Computer Systems Administrators	69.160	347.200
Computer Hardware Engineering	98.810	70.000

Aparentemente, las consideraciones científicas acerca de los ámbitos de actividad del Ingeniero Informático no tienen una relación demasiado estrecha con las actividades que se recogen en la CNAE, poniéndose una vez más de manifiesto la necesidad de un contacto más estrecho entre el mundo universitario y el empresarial. Si cabe, dado que una gran mayoría de ofertas de trabajo buscan perfiles profesionales relacionados con la programación, en muchos casos se suelen identificar las especialidades de Ingeniería Informática que tienen que ver con desarrollo de software, sistemas de información, e incluso tecnologías de la información como las más prometedoras desde el punto de vista de las oportunidades laborales. La especialidad de Ingeniería de Computadores, se contempla como la menos relacionada con el software del computador. En cualquier caso, cuando se trata de salir de una situación de crisis económica, o se pretende impulsar un sector manifiestamente mejorable, es más

importante analizar las expectativas de futuro que puede ofrecer una titulación determinada que contabilizar las limitadas posibilidades que existen. Por lo tanto, desde el punto de vista de este artículo, lo que habría que plantearse es si tal o cual especialidad permite adquirir unas competencias próximas, no ya a las que se demandan actualmente, sino a las que se demandarán en un futuro próximo. Más que describir la situación actual respecto a un tipo determinado de profesionales, lo importante es analizar sus perspectivas de futuro.

Así, aunque es difícil establecer una predicción suficientemente precisa del mercado de trabajo para los Ingenieros de Computadores en el futuro, se pueden identificar las opciones más probables, los yacimientos donde se puede encontrar materia prima para la creatividad, la innovación, y para el emprendimiento por parte de Ingenieros Informáticos. Para ello hemos utilizado dos tipos de fuentes. Por un lado, se han analizado documentos de organismos públicos o asociaciones de empresas nacionales como SANDETEL y AMETIC (Asociación Multisectorial de Empresas de Electrónica, Tecnologías de la Información, Telecomunicaciones, y Contenidos Digitales, www.ametic.es). Por otro lado, se han tenido en cuenta los análisis realizados por redes de excelencia como HIPEAC (High Performance and Embedded Architecture and Compilation, www.hipeac.net), relacionadas con el ámbito de la Ingeniería de Computadores. Obviamente, no se trata de hacer aquí una presentación exhaustiva de los documentos a los que nos referiremos sino simplemente de proporcionar esas referencias para su consulta, y de extraer algunas conclusiones que consideramos interesantes desde el punto de vista de la motivación de nuestros estudiantes.

En [14] se indica que: "Las tecnologías de la información y las comunicaciones (TIC) se plantean el reto de ofrecer servicios, sistemas y aplicaciones respondiendo al paradigma *siempre conectado* y *en cualquier lugar*. Este paradigma se ha convertido en un objetivo que impulsa la innovación y que está promoviendo que la industria de los servicios y contenidos electrónicos se mueva a gran velocidad." En [15] también se hace hincapié en las expectativas empresariales que surgen del aprovechamiento de los dispositivos móviles y de la accesibilidad a toda la tecnología relacionada, y se sitúan las posibilidades que se abren con el acceso móvil a Internet al nivel de lo que supuso la aparición del computador personal o del propio Internet. En cualquier caso, sectores de nuestra economía como la salud, el transporte, el turismo, etc., van a encontrar nuevas posibilidades de crecimiento en el acceso, a través de dispositivos móviles, a datos y servicios promoviendo además las infraestructuras de *cloud*.

Existe una coincidencia entre el planteamiento previo, que parte de las empresas, y el punto de vista de la red HIPEAC, que puede considerarse representativo del ámbito académico y científico. Así, en [16] se defiende que el futuro inmediato de los computadores vendrá determinado por la programación eficiente de arquitecturas paralelas heterogéneas, el diseño de infraestructuras que escalen adecuadamente al aumentar la complejidad y el volumen esperado de datos, y la computación ubicua fiable y predecible. Según esto, las áreas de trabajo y los objetivos del ingeniero de computadores estarían relacionados con:

- *La eficiencia de los sistemas*. Maximizar la computación por unidad de energía de las arquitecturas, explorando el diseño de arquitecturas heterogéneas y la gestión de la localidad y las comunicaciones.

- *La gestión de sistemas cada vez más complejos*. Proporcionar herramientas que permitan el desarrollo de software para las nuevas arquitecturas multi-núcleo heterogéneas, las nuevas generaciones de núcleos de procesamiento, y la optimización integral de los diseños (optimización inter-componente e inter-layer).

- *La mejora de la fiabilidad de los sistemas y las aplicaciones.* Tanto el desarrollo de arquitecturas para las aplicaciones de procesamiento intensivo en datos (*Data Deluge*), como el de plataformas fiables para la computación ubicua.

Además, estas áreas y objetivos tendrán incidencia en los sectores que se consideran estratégicos en la Unión Europea. Concretamente, los nuevos sistemas de cómputo ocasionarán cambios relevantes en el sector energético, el transporte y la movilidad, la salud, el medio ambiente, el cuidado de los mayores, la productividad, la fiabilidad, la seguridad, y la educación.

4 La Ingeniería de Computadores y la Informática

Muchas de las oportunidades que las tecnologías de la información y las comunicaciones ofrecen para contribuir a la mejora de la productividad y al progreso social, y que permiten desarrollar una carrera profesional creativa y con futuro, requieren competencias propias de la Ingeniería de Computadores. En esta sección analizaremos lo que aportan al ingeniero informático los conocimientos desde los que el ingeniero de computadores contempla los sistemas de cómputo. Así, proporcionaremos algunos ejemplos de la influencia de la arquitectura de los sistemas (y los niveles que se suelen identificar como hardware) en la programación y en las características de los lenguajes de programación, dado que con frecuencia se olvida la importancia de los niveles próximos al hardware en el desarrollo de proyectos informáticos relevantes, y se suele identificar el papel del ingeniero informático con el del desarrollo de software usando lenguajes de programación de alto nivel que permiten “aislarse” de las características del hardware.

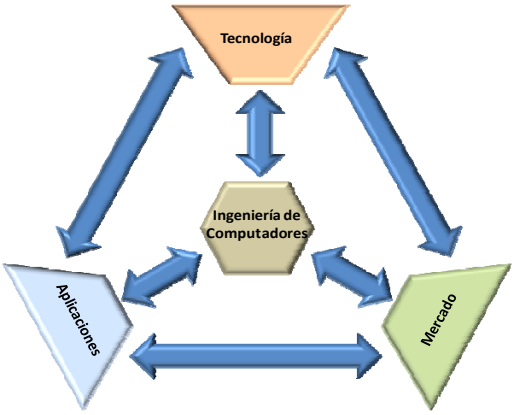

Figura 1. Interacción Tecnología-Aplicaciones-Mercado-Ingeniería de Computadores

La ingeniería de computadores aborda la informática desde un punto de vista sistémico. Integra los aspectos científicos y tecnológicos del diseño, implementación y mantenimiento de los componentes hardware y software de los sistemas de cómputo, las redes, y los equipos controlados por computador, que se utilizan en dominios de aplicación muy diversos [17] donde se necesitan conocimientos de las características hardware de la plataforma y del desarrollo de un software que aproveche eficientemente las posibilidades de la misma.

La evolución científica y tecnológica, según la teoría de las revoluciones científicas de K. S. Kunh, se produce a través de paradigmas que surgen para dar respuestas a nuevos problemas y deficiencias de paradigmas previamente establecidos. Tal y como muestra la Figura 1, en la Ingeniería de Computadores los paradigmas evolucionan a partir de la interacción entre las posibilidades que proporciona la tecnología y las características de las aplicaciones que demanda el *mercado*, definido por los factores económicos (oferta-demanda) que introducen condicionantes en las aplicaciones y en la viabilidad de las tecnologías y los sistemas de cómputo. Esta interacción entre tecnología, aplicaciones y mercado determina las perspectivas profesionales futuras de los ingenieros de computadores al definir los problemas a los que las nuevas plataformas de cómputo deben dar respuestas. Según el paradigma actualmente vigente, estamos rodeados de dispositivos y computadores, interconectados, y con capacidades de procesamiento cada vez mayores. De hecho, en todas las actividades se utilizan artefactos, aparatos, y utensilios que pueden mejorarse en cuanto a sus funciones, a su automatización, y a su control. La tecnología ha permitido la sustitución de elementos de control basados en electrónica analógica por sistemas digitales basados en microcontroladores y procesadores cada vez más potentes, de menor tamaño y menor consumo, y su uso en aplicaciones donde antes no podía pensarse en que se podrían utilizar por precio, dimensiones, consumo, o falta de prestaciones.

Las oportunidades que se pueden generar sólo van a estar limitadas por el ingenio, y las capacidades de emprendimiento y financiación. Hay que recordar que la población actual del mundo es de alrededor de $7x10^9$ personas y hay $4.3x10^9$ direcciones IP (con IPv6 hay $3.4x10^{38}$ direcciones posibles), aunque se estima que para 2020 habrá alrededor de $50x10^9$ dispositivos conectados a la red (6.58 dispositivos por persona), gran parte de ellos integrados en objetos cotidianos, como los *smartphones*, los automóviles, los electrodomésticos, o la ropa. Esta tendencia ha dado lugar a un ritmo de crecimiento considerable en el consumo de sistemas empotrados y procesadores para teléfonos móviles, en comparación con el del consumo de procesadores para lo que se reconoce por el gran público como "computador" (servidores, PCs y portátiles). Por tanto, en los próximos años surgirá una gran diversidad de aplicaciones englobadas en lo que se *denomina Internet de las Cosas*. De hecho, el volumen de mercado de los denominados *smart systems*, incluyendo aplicaciones en los sectores de la seguridad, la salud, el transporte, la educación, la automatización industrial, la energía, y el hogar, se estima en más de un billón (10^{12}) de dólares para 2016 (www.marketsandmarkets.com/). Teniendo en cuenta el volumen de dispositivos y sistemas de cómputo interconectados, se estima que en 2017 el tráfico de datos será superior a los 11.2 exabytes por mes [18]. Por otra parte, la información almacenada (y replicada) a finales de 2011 era de 1.8 Zettabytes ($1.8\ 10^{21}$ bytes = 1.8 billones de Gbytes), y aumenta el número de aplicaciones y actividades empresariales que tienen que ver con el acceso a bases de datos cada vez mayores. Las necesidades de almacenamiento, acceso, y procesamiento de esos datos también promoverán un número considerable de innovaciones y mercados, que mantendrán el interés por la supercomputación y marcarán la evolución del *cloud computing* [14, 19] y los *centros de datos*. La relevancia de la supercomputación en los ámbitos económico y social, va a mantener el interés por superar los considerables retos tecnológicos que plantea la constante búsqueda de plataformas cada vez más potentes. Así, en la lista de Noviembre de 2012 del TOP500 (www.top500.org) se superaron los 17 PFLOPS ($17x10^{15}$ operaciones en coma flotante por segundo), tan sólo cuatro años después de alcanzar el PFLOPS en 2008, pero la necesidad de más velocidad de cómputo se mantiene. En este ámbito, también es clave el consumo de energía (hay estudios que indican que en EE.UU. las TIC están en el segundo lugar como sector consumidor de energía), como pone de manifiesto la lista GREEN500 (www.green500.org). Al evaluar las prestaciones de los supercomputadores importa la velocidad, pero también la eficiencia energética.

Se puede considerar que la Ingeniería de Computadores proporciona el eslabón entre las prestaciones y el funcionamiento correcto del sistema. Cuando además del resultado importan las restricciones de tiempo establecidas para obtenerlo y/o el consumo energético, los conceptos que aborda la Ingeniería de Computadores son esenciales. Por ejemplo, la necesidad de obtener cada vez más velocidad de procesamiento con un consumo energético limitado ha orientado la evolución de los microprocesadores hacia microarquitecturas que integran varios núcleos de procesamiento junto con memorias cache, controladores de memoria, interfaces con buses de E/S e interconexiones, etc. Estas microarquitecturas multi-núcleo están presentes en prácticamente todos ámbitos de aplicación de los dispositivos de cómputo. Por ejemplo, se pueden usar varios núcleos para mantener el nivel de servicio con menos consumo de energía: los teléfonos móviles con arquitecturas

multi-núcleo podrían trabajar a menor frecuencia y consumir menos batería. El dominio de conceptos de concurrencia y paralelismo es una necesidad cada vez más acuciante. Para los programadores, la mejora de las prestaciones ha dejado de estar servida en bandeja sin esfuerzo con cada nueva generación de microprocesadores. Ahora hay que escribir software que extraiga explícitamente el paralelismo que ofrecen las arquitecturas multi-núcleo (en muchos casos heterogéneas) para cuya programación no existen herramientas totalmente adecuadas hasta el momento. Los códigos que no sean capaces de aprovechar la presencia de varios núcleos de procesamiento, se encontrarán en una clara desventaja competitiva.

En [20,21] se pone de manifiesto cómo las características de las arquitecturas multi-núcleo pueden hacerse visibles para el programador de alto nivel. Por ejemplo, en una arquitectura multi-núcleo la computación puede ser bastante menos costosa que la comunicación entre núcleos (aunque los anchos de banda entre núcleos pueden ser del orden de los 3 GB/s, las latencias pueden llegar casi al centenar de ciclos), o un acceso a la memoria compartida, que debe mantener la coherencia. En las arquitecturas secuenciales usualmente se almacenan los datos que se van obteniendo para evitar tener que recalcularlos, dado que esto suele costar más tiempo que acceder a registros locales donde se pueden almacenar los resultados. En arquitecturas con memoria físicamente distribuida entre los procesadores, no está tan claro que el acceso de un núcleo a un dato calculado por otro núcleo sea más rápido que volverlo a calcular. Sería deseable que los lenguajes de programación permitiesen diseñar programas en los que la decisión entre almacenar o recalcular se pudiera retrasar más allá del momento de la compilación. Con esto se podría mejorar la portabilidad de los programas entre distintas arquitecturas.

Otra muestra de la influencia de la arquitectura en los lenguajes de programación se encuentra en el paso de parámetros, por referencia o valor, en las llamadas a las funciones. La opción de utilizar el paso por referencia, usualmente preferida en una arquitectura con un solo núcleo, puede no ser la alternativa más eficiente en una arquitectura multinúcleo, donde pueden producirse penalizaciones considerables si el dato al que se hace referencia está en memoria compartida y se tiene un número elevado de núcleos. Según esto, y en la misma línea de lo indicado en relación con las alternativas de almacenar o recalcular, sería conveniente que los lenguajes permitieran la decisión retrasada del procedimiento para el paso de parámetros (late-binding) según las características de la arquitectura.

Estos ejemplos ponen de manifiesto que la tendencia actual en el diseño de nuevas arquitecturas hace que se necesiten lenguajes de programación que hagan énfasis en la concurrencia y permitan que, para desarrollar códigos eficientes de forma natural, los programadores tengan presente ciertas características de las arquitecturas. El ingeniero informático que tenga una mayor comprensión de estas situaciones no solo podrá desarrollar proyectos más competitivos sino que, además, tendrá más posibilidades de proponer innovaciones en el diseño de nuevas herramientas y plataformas. En este contexto, la especialidad de Ingeniería de Computadores contribuye, además, a desarrollar en el estudiante la perspectiva del aprovechamiento eficiente de los sistemas.

5 Estrategias docentes para la motivación

En esta sección se proponen algunas estrategias de enseñanza/aprendizaje para promover la demanda de asignaturas relacionadas con la Ingeniería de Computadores en estudios de grado teniendo, en cuenta lo descrito en las secciones previas. Estas estrategias deben aplicarse ya en las asignaturas obligatorias relacionadas que se cursan con anterioridad al momento en que se elije una u otra especialidad. Concretamente, y como se ha indicado, en el grado en Ingeniería Informática de la Universidad de Granada, las asignaturas Tecnología y Organización de Computadores, Estructura de Computadores, Arquitectura de Computadores, e Ingeniería de Servidores son las asignaturas obligatorias que los estudiantes deben cursar antes de elegir entre las distintas especialidades (sus contenidos se pueden consultar en [9].

Tabla 2. Estrategias docentes según las teorías de la motivación resumidas en la Sección 2

Teoría de la motivación	Estrategias docentes
Objetivos (ser competente, alcanzar los objetivos)	• Despertar la curiosidad planteando preguntas sobre aplicaciones prácticas de la materia que se irán respondiendo a lo largo de la asignatura • Evitar referencias que puedan convertir a las calificaciones en objetivos de aprendizaje
Atribuciones (causas internas, dinámicas, y controlables)	• Proponer relaciones de problemas de dificultad creciente, comenzando con problemas de resolución muy directa a partir de las leyes o expresiones matemáticas estudiadas en la asignatura • Evaluar el esfuerzo: proponer varias actividades voluntarias y tener en cuenta el número de actividades completadas.
Expectativa-Valor (metas alcanzables y relevantes)	• Mostrar que los conceptos que se abordan en la asignatura son importantes para ser profesionales competentes. • Realizar pruebas de evaluación sobre habilidades concretas o contenidos específicos tratados en clase con detalle.
Autodeterminación (autonomía, competencia, socialización)	• Plantear prácticas y ejercicios individuales que pongan de manifiesto la utilidad de los contenidos de la asignatura. • Promover el trabajo cooperativo en equipo • Describir posibilidades profesionales que abren las competencias alcanzadas con los contenidos de la asignatura

La Tabla 2 propone, en función de las distintas teorías de la motivación resumidas en la Sección 2, algunas estrategias docentes para motivar el aprendizaje en una asignatura. Las propuestas de la Tabla 2 ponen de manifiesto dos objetivos generales respecto a los contenidos de las asignaturas: (1) que sean relevantes para la profesión, y (2) que se contemplen como asequibles con un esfuerzo razonable. Precisamente uno de los 10 mandamientos de la docencia según el profesor Yale Patt [22] es no bajar el listón de los objetivos de aprendizaje. Buscar que los estudiantes asimilen y sientan que controlan el aprendizaje de una asignatura, y que son capaces de alcanzar las competencias correspondientes no debe suponer la renuncia a abordar conceptos o problemas complejos. Precisamente esos problemas son los que hacen que el estudiante llegue a ser consciente de la utilidad de la asignatura para su futuro profesional. Los que le convencen de que el esfuerzo realizado le ha permitido

avanzar en su conocimiento del computador y de aspectos de su profesión que no hubiera ni imaginado si no es por la asignatura estudiada.

Además de las teorías de la motivación, en la Tabla 2 también se han tenido en cuenta las tres categorías para los niveles de comprensión de un contenido: se conoce, se usa, y se asimila y se adapta [26]. Aunque el tercer nivel requiere un tiempo considerable (en [23] se indica que alcanzar el nivel de experto en un ámbito puede requerir del orden de 10000 horas), alcanzar los dos primeros sí que puede conseguirse en menor tiempo y pueden constituir el objetivo de una asignatura de grado de 6 créditos ECTS (150 horas). En este contexto, el modelo de enseñanza-aprendizaje en espiral [15] es muy adecuado para mejorar las expectativas de los estudiantes acerca de la consecución de los objetivos del aprendizaje, buscando la competencia y la utilidad de los conocimientos adquiridos. En dicho modelo se suceden secuencias de lección – demostración – práctica. Las clases teóricas o lecciones magistrales, sobre todo si se incluyen técnicas de aprendizaje activo que hacen posible la participación del estudiante [24], permiten transmitir conocimiento, pero el trabajo práctico es esencial para transformar el conocimiento en habilidades o competencias. Las demostraciones también son importantes para ilustrar experiencias que no pueden abordarse en prácticas, y para contrarrestar ideas preconcebidas por los estudiantes. Para ello, las herramientas de simulación, el software, y el material audiovisual accesible a través de Internet constituyen una ayuda inestimable. Por ejemplo, se puede empezar la presentación de una asignatura como Arquitectura de Computadores ejecutando diversas versiones de un mismo programa como el que se proporciona como material de apoyo en [25]. Este programa aborda una tarea de presentación de imágenes y las distintas opciones de optimización de código dan como resultado una clara mejora en la velocidad de presentación de las imágenes.

Las estrategias docentes de la Tabla 2 deben concretarse para cada asignatura describiendo su uso en los contenidos que aborda. Para completar esta sección utilizaremos como ejemplos de aplicación de esas estrategias la docencia del lenguaje ensamblador (Sección 5.1), y de la consistencia de memoria (Sección 5.2).

5.1. Enseñanza/aprendizaje del lenguaje ensamblador

El aprendizaje del lenguaje ensamblador suele concitar considerables reticencias entre los estudiantes. Respecto a esta cuestión, está ampliamente aceptado que hay que evitar que los estudiantes lo contemplen como una herramienta de programación, dado que no resiste ninguna comparación con cualquier lenguaje de alto nivel que estén utilizando para desarrollar sus programas. Es imprescindible cambiar esta perspectiva incidiendo sobre dos aspectos importantes del ensamblador. Por un lado está su faceta de lenguaje para describir lo que ocurre en la máquina al nivel adecuado para entender el efecto de su funcionamiento sobre las prestaciones del computador. Por otro lado está su utilidad a la hora de optimizar el uso que los programas hacen de la máquina. A través del lenguaje ensamblador se puede comprender cómo trabaja el procesador, cómo ve la memoria, cómo interactúan los programas con el sistema operativo, cómo se representan internamente los datos y cómo se accede a ellos, cómo se pueden optimizar los programas a alto y bajo nivel, etc. El comentario editorial del

libro [26] en la correspondiente página web de Amazon [www.amazon.com] indica que la diferencia entre un programador mediocre y un buen programador es que éste comprende el lenguaje ensamblador que, al fin y al cabo es el lenguaje del computador y permite entender como se ejecutan los programas. También se hace referencia a un comentario de Joel Spolsky [27] para el que programar sin entender como trabaja la CPU es como practicar la medicina sin saber anatomía. Si no se conoce ensamblador, no se sabe realmente lo que está ejecutando la máquina. Existen por lo tanto razones más que suficientes para convencer a los estudiantes de la importancia del ensamblador en su formación como ingenieros informáticos.

Queda el problema de abordar la enseñanza del ensamblador desde una perspectiva adecuada. Para ello, se pueden aplicar las estrategias docentes que se indican en la Tabla 2. Según ellas, habría que realizar una aproximación gradual al aprendizaje del lenguaje ensamblador en la que, por este orden: (1) se proponga la realización de programas para algoritmos sencillos de ordenación, operaciones aritméticas, etc., a través de los que se ejerciten los modos de direccionamiento, se aprendan los tipos de datos, y se tenga presente el efecto de la ubicación de los datos en la estructura del programa (en principio, no habría que incluir aspectos más complejos como el paso de parámetros); (2) se lean y analicen códigos en ensamblador generados por el compilador a partir de programas sencillos que no generen llamadas al sistema (por ejemplo bucles) y que permitirían introducir las cuestiones relativas al paso de parámetros; (3) se ensayen modificaciones en los códigos en ensamblador y se compruebe su efecto en las prestaciones del programa; y (4) se aborde finalmente algún programa más complejo en ensamblador para una aplicación más realista, donde el uso del lenguaje ensamblador proporcione alguna ventaja en cuanto a las prestaciones. Respecto a esta cuestión, la programación de algún juego sencillo suele resultar muy efectiva y motiva considerablemente a los estudiantes. Además, en las distintas asignaturas relacionadas con la Ingeniería de Computadores, el ensamblador debe utilizarse como una herramienta descriptiva más del comportamiento del computador.

5.2. Enseñanza/aprendizaje de la consistencia de memoria

La arquitectura de computadores se suele definir como la interfaz entre software y hardware. Entender la interacción del software con el hardware del computador implica tener ciertas nociones de cómo se procesan las instrucciones máquina: conocimientos del lenguaje de transferencia de registros, del aprovechamiento del paralelismo entre instrucciones, de la interacción con la memoria y los periféricos, y de sus efectos en el tiempo de procesamiento, etc., y otros contenidos que se imparten en asignaturas de Tecnología, Estructura y Arquitectura de Computadores. Otra forma de poner de manifiesto la utilidad de la arquitectura de computadores, es la optimización de código. El conocimiento de los detalles del hardware que permiten entender la forma de generar códigos más eficientes en cuanto a prestaciones y consumo, se justifica el interés de los conceptos propios de la tecnología, y la estructura y arquitectura de computadores para el desarrollo de programas eficientes.

En esta línea, el modelo de consistencia de memoria que utiliza un lenguaje de programación determina las optimizaciones que un compilador (o el programador) puede realizar manteniendo la ejecución correcta de los programas.

En esa línea, un concepto interesante para poner de manifiesto la interrelación entre el hardware y el software, de gran relevancia en el contexto actual del uso extensivo de microprocesadores multinúcleo, es el de la consistencia de memoria. El modelo de consistencia de memoria es fundamental en la semántica de un programa de memoria compartida ya que define los valores que puede devolver una instrucción de acceso a memoria (un *load*) al establecer las restricciones en el orden de acceso a memoria por parte de los distintos procesos/hebras del programa. El modelo de consistencia responde preguntas relativas a, por ejemplo, la sincronización que se necesita para asegurar que una escritura por parte de una hebra se produce antes que la lectura realizada por otra, el valor final de una posición de memoria sobre la que distintas hebras han escrito, etc.

A través del modelo de consistencia de memoria se define una interfaz entre el programa y el hardware o el software que transforma ese programa (un compilador o una máquina virtual por ejemplo) de forma que no es posible razonar sobre el programa con un modelo de consistencia ambiguo. Como se indica en [28], es difícil enseñar programación multihebra sin claridad en los modelos de memoria. Por otro lado, las características del modelo de consistencia de memoria deben procurar un equilibrio entre la concurrencia que puede aprovecharse al reducir las restricciones en el orden de los accesos a memoria, y la complejidad de la implementación y del modelo de programación que debe utilizarse. Por ejemplo, en un modelo de consistencia de memoria débil, en el que se minimizan las restricciones en el orden de los accesos de lectura o escritura a la memoria, necesita que el software proporcione información sobre los eventos de sincronización de accesos a memoria para que, en los casos en los que sea necesario, se respete cierto orden. Esta información de sincronización la proporciona bien el programador a través del correspondiente lenguaje paralelo, bien el compilador o su sistema de *runtime*. Si el software fuese capaz de proporcionar información sobre los patrones de acceso esperados para las variables compartidas, se podrían aplicar opciones de optimización que mejorasen las prestaciones del sistema. Poner de manifiesto este compromiso entre prestaciones y programabilidad, como otros compromisos que aparecen continuamente en el diseño y uso de los computadores, es relevante para la formación de un Ingeniero Informático.

La incidencia de los modelos de consistencia en el programador se puede poner de manifiesto de forma inmediata si en la asignatura donde se imparten dichos modelos se realizan prácticas con programas paralelos realizados con herramientas como OpenMP. Precisamente la sección 1.4.1 del documento "OpenMP Application Programming Interface" [29], correspondiente a las especificaciones de la versión 3.0 de OpenMP, se dedica al modelo de memoria de OpenMP y comienza indicando: "OpenMP provides a relaxed-consistency, shared-memory model". Según esto, el programador debe saber qué son los modelos de consistencia de memoria, qué problemas que plantean, y cómo afecta al desarrollo de los programas paralelos. A

medida que se extiende el uso de los microprocesadores multinúcleo, los multiprocesadores NUMA, y los clusteres, es más importante disponer de una especificación clara del modelo de memoria y un uso correcto del mismo en los programas paralelos. En [29, 30] se pueden encontrar descripciones del modelo de consistencia de memoria OpenMP y de su relación con otros modelos de consistencia previamente propuestos. Concretamente, el modelo de memoria de OpenMP considera que existe una *memoria*, accesible a todas las hebras, para almacenar datos y acceder a ellos. Además, cada hebra dispone de una imagen de esa memoria (*memoria temporal*), que puede utilizar para almacenar datos temporalmente cuando otras hebras no necesitan acceder a ellos. Los datos pueden moverse entre *memorias temporales* y *memoria* pero para pasar de una memoria temporal a otra tienen que hacerlo a través de la memoria. Una variable utilizada en una región paralela puede ser compartida o privada. Cada referencia a una variable compartida dentro de la región paralela se refiere a la variable original del mismo nombre. En el caso de una variable privada, una referencia a dicha variable dentro de una región se refiere a la variable del mismo tipo y tamaño original pero privada para una hebra (no accesible a las demás).

Como se ha dicho, el modelo de consistencia de OpenMP es un modelo relajado (weak ordering) en el que, dentro de una hebra, se permite reordenar los accesos a variables diferentes a no ser esos accesos estén separados por una directiva FLUSH referida a ambas variables. Así, los órdenes que garantiza este modelo de consistencia son los S→W, S→R, R→S, W→S, y S→S, donde S es una operación de sincronización, R es una lectura, y W una escritura. La directiva FLUSH garantiza que las operaciones de memoria previas a la directiva sobre las variables a las que se refiere dicho FLUSH deben haberse completado y tener copias en memoria antes de que acabe el FLUSH, y además, las operaciones con memoria posteriores a FLUSH no deben comenzar antes de que la directiva termine. Además, todos los valores que queden en las memorias temporales se descartan, de forma que las lecturas que se produzcan al terminar el FLUSH toman los datos de memoria. La directiva FLUSH es la única que permite que se pueda pasar un dato de un thread a otro thread. Para ello, en la hebra origen se hace una escritura a la variable compartida y luego un FLUSH a la variable. En la hebra destino debe hacerse un FLUSH de la variable y luego la lectura.

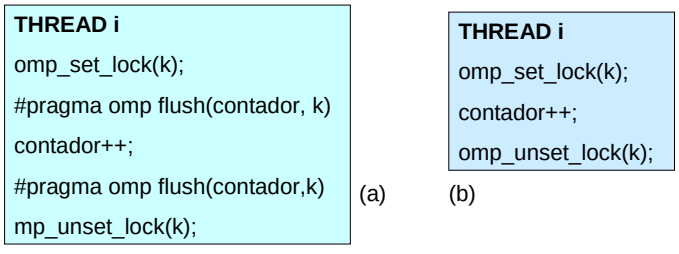

Figura 2. Implementación de un incremento de un contador en una sección crítica: (a) OpenMP 2.0, sin FLUSH implícito en el LOCK; (b) OpenMP 2.5, con FLUSH implícito.

La interacción de la arquitectura y la programación que tiene lugar debido al modelo de consistencia se pone de manifiesto al comparar la actualización de una variable mediante dos hebras sincronizadas a través de un LOCK en dos versiones diferentes de OpenMP. En la Figura 2.a se muestra la situación para la versión 2.0 donde es necesario incluir explícitamente FLUSH, mientras que en la versión 2.5, el FLUSH está implícito en el LOCK.

Según las estrategias de la Tabla 2, para motivar el aprendizaje de los modelos de consistencia de memoria es conveniente: (1) utilizar de ejemplos sencillos de accesos a memoria generados por procesos/hebras diferentes, que puedan ocasionar problemas según los distintos modelos de consistencia; y (2) el estudio de programas paralelos de memoria compartida con OpenMP que generen resultados diferentes si no se tiene en cuenta el modelo de consistencia de memoria que se usa. En [31] se hace referencia a las implicaciones del modelo de consistencia de memoria de OpenMP en los errores de programación, dado que cuando se lee una variable compartida sin anteponer un FLUSH, no está garantizado que el valor que se obtenga esté actualizado. Además, también habría que utilizar FLUSH en las hebras que han escrito en dicha variable con anterioridad. Afortunadamente, en muchos casos los errores no son aparentes en los programas porque muchas construcciones de OpenMP realizan FLUSHes implícitos. En muchas ocasiones, no se observan errores porque las variables se declaran como "volatile", con lo que existe un FLUSH implícito delante de cada lectura y detrás de cada escritura de la variable en cuestión. Sin embargo, esto supone que el compilador no podrá aplicar muchas optimizaciones posibles a esas variables (por ejemplo, las relacionadas con la reordenación de accesos). Por eso, lo ideal es que el programador sea consciente del problema y utilice FLUSH (bien explícitamente a través de la correspondiente directiva o de una construcción OpenMP que la incluya implícitamente) solo cuando sea imprescindible.

Un aspecto a tener en cuenta respecto a la inclusión de los modelos de consistencia en la docencia de asignaturas de Estructura y Arquitectura de Computadores es que no se trata de un problema totalmente resuelto [28]. A los estudiantes se les puede enseñar alguna de las propuestas actuales, pero también se puede razonar en torno a sus limitaciones para poner de manifiesto que existen posibilidades de innovación dentro de este ámbito de conocimiento. De esta forma, también se incide positivamente en su motivación.

6 Conclusiones

Tras resumir algunas de las últimas conclusiones sobre el proceso de aprendizaje y después de revisar las características fundamentales de las teorías de la motivación más relevantes se ha puesto de manifiesto la importancia de la visión previa que los estudiantes tienen cuando inician el estudio de una asignatura y la diferencia entre motivación intrínseca y motivación extrínseca. De esta forma, las necesidades psicológicas de autonomía, competencia y socialización hacen que, si bien una fuerte

demanda de profesionales que prácticamente garantice un puesto de trabajo sea un factor importante de cara a motivar a los estudiantes (motivación extrínseca), también lo sean la relevancia de los problemas que se plantean en la disciplina en cuestión y su utilidad (motivación intrínseca).

Teniendo en cuenta esta distinción entre motivación intrínseca y extrínseca, las Secciones 3 y 4 se han dedicado a proporcionar argumentos que justifiquen el interés por la Ingeniería de Computadores. Dado que el análisis realizado en la Sección 3 ha puesto de manifiesto una cierta dificultad de proporcionar motivaciones extrínsecas intensas para los estudios de Ingeniería de Computadores frente a otras especialidades, en la Sección 4 se han analizado las condiciones de la evolución en las tecnologías de la información y las comunicaciones que pueden fundamentar la motivación intrínseca de los estudiantes hacia la Ingeniería de Computadores porque las competencias que ofrece permiten intervenir, de forma decisiva, en la implantación de nuevas soluciones a problemas esenciales de nuestra sociedad.

Para terminar el artículo, se han propuesto una serie de estrategias docentes útiles para impartir los contenidos de las asignaturas relacionadas con la Ingeniería de Computadores. Se trata de estrategias inspiradas en las teorías sobre la motivación resumidas en la Sección 2 cuya aplicación han ilustrado utilizando, como ejemplos, la docencia del lenguaje ensamblador y la consistencia de memoria. Se hace hincapié en que los estudiantes deben contemplar los contenidos de la asignatura como relevantes para la profesión, y como asequibles, sin tener por ello que renunciar a impartir conceptos o problemas complejos que, por otra parte, son los que hacen que el estudiante llegue a ser consciente de la utilidad de la asignatura para su futuro profesional.

Referencias

1. Tanenbaum, A.S.:"Sistemas Operativos Modernos" (3ª Edición). Pearson Prentice Hall. México 2009. (ISBN 978-607-442-046-3).
2. Ambrose, S.A.; Bridges, M.W.; Di Pietro, M.; Lovett, M.C.; Norman, M.K.:"How learning works: Seven research-based principles for smart teaching". Jossey-Bass, San Francisco, 2010.
3. Bransford, J.D.; Brown, A.L.; Cocking, R.R. (Eds.): "How People Learn: Brain, Mind, Experience, and School". Nat. Academy Press, 2001.
4. Torres-Ayala, A. T.; Herman, G.L.:"Motivating learners: A primer for Engineering Teaching Assistants". American Society of Engineering Education (ASEE) Annual Conference, San Antonio, TX, June 10-13, 2012.
5. Pintrich, P. R.:"Motivation and Classroom learning". Handbook of psychology, pp.103-122, Wiley Online Library, 2003.
6. Oudeyer, P.Y.; Kaplan, F.: "What is intrinsic motivation?. A typology of computational approaches". Frontiers in Neurorobotics, Vol.1, pp. 1-14. Noviembre, 2007.
7. Pink, D.H.:"La sorprendente verdad sobre qué nos motiva". Gestión'2000, 2010.
8. ACM/AIS/IEEE-CS (The Joint Task Force for Computing Curricula 2005): "Computing Curricula 2005. The Overview Report". Septiembre, 2005. (http://www.acm.org/education/curricula-recommendations)
9. Grado en Ingeniería Informática de la Universidad de Granada: http://grados.ugr.es/informatica/pages/infoacademica
10. SANDETEL, Análisis del Sector TIC Andaluz en 2010: http://www.sandetel.es/index.php?option=com_content&task=view&id=39&Itemid=65.

11. SANDETEL, Análisis de mercado para áreas de actividad relacionadas con las TIC: http://www.sandetel.es/index.php?option=com_content&task=view&id=42&Itemid=68.
12. Asociación de Empresas de Electrónica, Tecnología de la Información y Telecomunicaciones de España (AETIC): "Informe anual del sector español de Electrónica, Tecnologías de la Información y Telecomunicaciones", 2010.
13. Buyya, R.; Yeo, C.; Venugopal, S.:"Market-oriented cloud computing: Vision, hype, and reality for delivering IT services as Computing Utilities". Proc. of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC'08), pp.5-13, 2008.
14. SANDETEL, Interoperabilidad de las tecnologías de telecomunicación en escenarios de convergencia: http://www.sandetel.es/index.php?option=com_content&task=view&id=162
15. AMETIC, Accenture: "Communications, Media & Technology. Un estudio publicado por el Centro de Alto Rendimiento de Accenture (CAR)", 2012.
16. HIPEAC Roadmap 2011/2012: http://www.hipeac.net/system/files/hipeac-roadmap2011.pdf
17. McGettrick, A.; Theys, M.D.; Soldan, D.L.; Srimani, P.K.:"Computer engineering curriculum in the new millennium". IEEE Transactions on Education, Vol. 46, No.4, pp.456-462. Noviembre, 2003.
18. CISCO Visual Networking Index (VNI): www.cisco.com/en/US/netsol/ns827/networking_solutions_sub_solution.html
19. Cooney, M.: "Gartner: Top 10 strategic technology trends for 2013". Octubre, 2012: http://www.computerworlduk.com/news/applications/3406773/gartner-top-10-strategic-technology-trends-for-2013/
20. Mycroft A.:"Programming Languages Design and Analysis motivated by Hardware Evolution". H. Riis and G Filé (Eds.), SAS 2007, LNCS 4643, pp.18-33, 2007.
21. Bernstein, D.:"Software enablement for multicore architectures". http://ce.et.tudelft.nl/~carlo/File_index/Symposium/02_Bernstein_[Vassiliadis_Symposium_2007].pdf, 2007.
22. Patt, Y.:"Ten Commandments for Good Teaching". http://users.ece.utexas.edu/~patt/Ten.commandments/, 2013.
23. Litzinger, T.A.; Lattuca, L. R.; Hadgraft, R.G.; Newstetter, W.C.:"Engineering education and the development of expertise". Journal of Engineering Education, Vol. 100, No.1, pp.123-150. Enero, 2011.
24. McConnell, J.J.:"Active and cooperative learning: tips and tricks (part I). SIGCSE Bulletin, 37, 2, pp. 27-30, 2005.
25. Gerber, R.:"The Software Optimization Cookbook. High Performance Recipes for the Intel Architecture". Intel Press, 2002.
26. Barlett, J.: "Programming from the ground up". www.barlettpublishing.com, 2003.
27. http://joelonsoftware.com/
28. Adve, S. V.; Boehm, H.-J.:"Memory models: a case for rethinking parallel languages and hardware". Communications of the ACM, Vol. 53, no.8, pp.90-101. Agosto, 2010.
29. OpenMP Architecture Review: "OpenMP Application Programming Interface" (Version 3.0), www.openmp.org, 2008.
30. Hoeflinger, J.P.; Supinski, B.R.:"The OpenMP Memory Model". Proceedings of the First International Workshop on OpenMP (IWOMP 2005), 2005.
31. Süss, M.; Leopold, C.:"Common Mistakes in OpenMP and how to avoid them. A collection of best practices". OpenMP Shared Memory Parallel Programming, LNCS vol. 4315, pp. 312-323, 2008.