Identificación de competencias para el diseño de un modelo educativo en ingeniería de software

MI Patricia Parroquín, MI Karla Olmos, MC Luis F. Fernández, MI Victoria González

Introducción

En México, la Secretaría de Economía, en coordinación con organismos empresariales y empresas del ramo de las tecnologías de información, diseñó el Programa para el Desarrollo de la Industria del Software (ProSoft) con el objetivo de impulsar la industria de software y extender el mercado de tecnologías de información en el país. Las metas que se plantea este programa son las siguientes: que al término del 2013 México alcance el promedio mundial de gasto en TI, que se logre una producción anual de software por cinco millones de pesos y que el país sea reconocido como líder latinoamericano en desarrollo de software y contenidos digitales en español (Secretaría de Economía, 2007).

Este programa propone siete estrategias, interrelacionadas entre sí, para alcanzar los objetivos. En particular, la estrategia dos plantea *la necesidad de la educación y formación de personal competente en el desarrollo de software, en cantidad y calidad convenientes*, ya que los recursos humanos representan el factor crucial en la industria del software.

Los siguientes puntos plantean la necesidad de personal competente en la industria del software:

- México se percibe en el plano internacional como un país sin capacidad de desarrollar tecnología, en particular software.
- La escasez de programadores e ingenieros certificados en las últimas tecnologías y la carencia de infraestructura adecuada, dificultan la posibilidad de atraer alianzas estratégicas e inversión extranjera.

Se considera que la disponibilidad, en cantidad y calidad, de los recursos humanos depende de las instituciones educativas y de capacitación. En el caso de las instituciones de educación superior, estos planes de estudio requieren una adecuación inmediata y actualización permanente para responder a la dinámica de la evolución del sector de desarrollo de software. El problema está en que gran parte de las universidades en México no cuentan con un modelo educativo basado en competencias que coadyuve a paliar los problemas que presenta la formación de recursos humanos que la industria del software requiere.

Nuestra institución, la Universidad Autónoma de Ciudad Juárez, se localiza en Ciudad Juárez, ciudad que encuentra ubicada al norte de México, en el estado de Chihuahua y colinda con los estados de Texas y Nuevo México, que pertenecen a Estados Unidos de Norteamérica; la ciudad vecina es El Paso, Texas y a 45 minutos, la ciudad de Las Cruces, Nuevo México.

La situación geográfica de la institución, estratégica por naturaleza, implica estar sometido a una influencia de uno de los países tecnológicamente más avanzados. Esta cercanía origina que se tenga una visión diferente de la globalización comparada con el resto del país. Este escenario se presta para identificar las competencias necesarias del recurso humano para que la industria mexicana del software sea más competitiva en un mercado globalizado.

Necesidad del Modelo educativo basado en competencias.

Actualmente la UACJ cuenta con el programa de Ingeniería en Sistemas Computacionales (ISC), en la cual están adscritos aproximadamente 1000 alumnos y se está trabajando en construir el nuevo programa de estudios en Ingeniería de Software.

El documento que guía el desarrollo de los cursos de cualquier programa de estudios en la UACJ se le conoce como carta descriptiva. Las cartas descriptivas contienen los siguientes rubros: 1) identificación del programa, 2) su ubicación en el plan de estudios, 3) los conocimientos, habilidades y destrezas, y actitudes y valores que los estudiantes deben de tener antes de cursar la materia, 4) el propósito general y los objetivos formativos e informativos que se pretende que el estudiante adquiera, los cuales también se dividen en

conocimientos, habilidades y destrezas, y actitudes y valores, además de los problemas que puede solucionar, 5) las condiciones de operación, 6) el contenido, 7) las estrategias didácticas, 8) los criterios de evaluación y acreditación, 9) bibliografía y 10) el perfil del docente.

Aunque las cartas descriptivas contienen la información necesaria para guiar un curso, se han detectado tres problemas, el primero de ellos es que las competencias no están explícitas dentro del documento, el segundo es que no se ha realizado un estudio en conjunto con la industria para determinar si las competencias descritas en forma implícita son las que realmente requieren los egresados de esta carrera para ser exitosos en su vida laboral, el tercero, es que se desconoce si los estudiantes adquieren las competencias, ya que no hay un estudio formal en el que se realice una evaluación de adquisición de competencias.

Por tal motivo, se plantea la necesidad de un modelo educativo basado en competencias que guie el diseño de programas educativos relacionados con las tecnologías de la información. Este enfoque basado en competencias surge como una de las respuestas al hecho de que los estudiantes al graduarse poseen un conjunto de conocimientos, válidos en la mayoría de los casos, pero que muchas veces no responden a lo que se necesita para actuar en la realidad.

De acuerdo a Deseco [2003] una competencia se define como la capacidad de resolver demandas o de realizar una tarea con éxito y se agrupan en dos dimensiones: cognoscitivas y no cognoscitivas. En este sentido, se entiende que la competencia es una construcción social compuesta de aprendizajes significativos en donde se combinan atributos tales como conocimientos, actitudes, valores y habilidades, con las tareas que se tiene que desempeñar en determinadas situaciones. Con la enseñanza basada en competencias se pretende que los alumnos adquieran: la capacidad para construir conocimiento (saber), se desempeñen eficazmente en el ejercicio de su profesión (saber hacer) y se integren de manera eficiente a la vida profesional y al ámbito social, económico y político (saber ser).

Identificación de competencias

El desarrollo de software es, en principio, un escenario que constantemente cambia; consideremos que el hardware y el software avanzan de manera vertiginosa, que el uso de las computadoras y las aplicaciones que en ellas se instalan están orientadas a resolver problemas en diferentes dominios y que muchos de estos dominios son complejos y críticos. Adicionalmente a estas características, existen otras que son impuestas por un entorno específico (se discuten en la sección 4), marcado por particularidades locales, regionales y nacionales. En este contexto cabe preguntarse ¿qué clase de conocimientos, habilidades, actitudes, etc., son necesarias para adecuarse a todo este conjunto de exigencias? Una posible respuesta es establecer, para este entorno particular, las competencias que un profesionista del desarrollo de software, es decir un ingeniero de software, debe tener para habitar exitosamente en este mundo del desarrollo de software.

Identificar, de forma preliminar, las competencias básicas requeridas en el desarrollo de software, requiere de una revisión de la literatura relacionada con las competencias y habilidades necesarias para esta disciplina. En la literatura al respecto, que se puede considerar amplia y a veces compleja, podemos encontrar desde formas de enseñar la ingeniería de software (Liu, Marsaglia, & Olson, 2002), habilidades y capacidades necesarias en los programadores (Bailey & Stefaniak, 2001), habilidades necesarias en los equipos de trabajo (Hogan & Thomas, 2005) y experiencias de la industria (Fernández, García, Camacho, & Evans, 2006) entre otras aproximaciones. No obstante que una gran parte de los artículos revisados se enfocan en los conocimientos necesarios (que podemos identificar como académicos), una posición que comparten varios autores, por ejemplo (Beard & Schwieger, 2007) y (Orsted, 2000), es que en la formación que reciben los desarrolladores de software los modelos educativos son débiles en propiciar lo que ellos mencionan como "*soft skills*"; entendidas como lo necesario para la interacción diaria: comunicación, liderazgo, motivación, etc. Aunque se revisó y analizó una cantidad considerable de artículos, el presente trabajo no contempla un reporte de esta revisión.

Una aproximación que proponemos para comprender mejor el manejo de las competencias es mediante un marco o modelo. La figura 1 muestra esta aproximación.

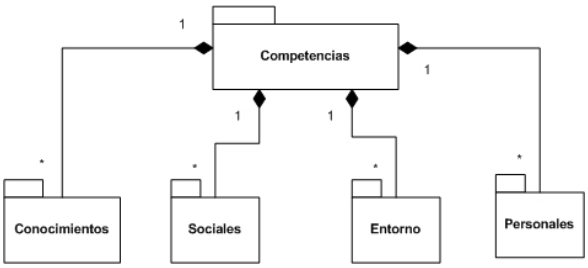

Fig. 1: Modelo para competencias

Este modelo propone a las competencias como una composición de conocimientos, características sociales y personales, y condiciones requeridas por el entorno. El modelo es un diagrama en notación UML (Lenguaje Unificado de Modelación). Mostrar el modelo en este tipo de diagrama tiene varias implicaciones; una de ellas es que cada parte es un "paquete", es decir contiene un conjunto de "cosas"; otra implicación es que la asociación entre "paquete competencias" y los otros paquetes (conocimientos, sociales, entorno, personales) es una asociación de composición, es decir que la competencia está "compuesta" por los otros paquetes.

Si retomamos la propuesta de que competencia es "*saber + saber hacer + saber ser*" podemos pensar que:

saber → conocimiento

saber hacer → conocimiento + personales + sociales

saber ser → sociales + entorno

Aunado al modelo propuesto y como resultado de esta investigación inicial, también proponemos cuatro rubros en los que habría que agrupar las competencias; esta propuesta está influenciada por algunos estudios hechos por ProSoft (Secretaría de Economía, 2007):

1) Algoritmia y resolución de problemas.

2) Trabajo en equipo.

3) Desarrollo de proyectos.

4) Administración de proyectos.

En esta etapa inicial especificamos los cuatro rubros identificados.

Algoritmia y resolución de problemas.

Una de las principales competencias de los egresados de las carreras relacionadas con el desarrollo de software es la de programar computadoras. Según Joyanes (2006), un programador es, antes que nada, una persona que resuelve problemas, por lo que para llegar a ser un programador eficaz se necesita aprender a resolver problemas de un modo riguroso y sistemático.

La resolución de un problema exige el diseño de un algoritmo, entendiéndose este último como un método para resolver un problema. En la ciencia de la computación y en la programación, los algoritmos son más importantes que los lenguajes de programación o las computadoras. Un lenguaje de programación es sólo un medio para expresar un algoritmo y una computadora es sólo un procesador para ejecutarlo. El diseño de la mayoría de los algoritmos requiere creatividad y conocimientos profundos en la técnica de programación (Joyanes, 2006).

En este sentido Schulte y Bennedsen (2006) mencionan cinco áreas de enfoque que deben considerarse en los cursos introductorios de programación

1. Orientación general: Cuál es la idea general de los programas, para que sirven y en que pueden ser utilizados.
2. El modelo abstracto de la máquina cuando ejecuta los programas.
3. Notación. La sintaxis y la semántica de los lenguajes de programación utilizados
4. Estructuras. Conocidas como planes/esquemas, soluciones para problemas estándares, un conjunto estructurado de conocimiento relacionado.
5. Pragmática. Habilidad para la planeación, desarrollo, prueba, depuración, etc.

La conclusión que Schulte y Bennedsen dan en su artículo es que "…los maestros tienden a enfocarse en el código. Este enfoque en codificación implica enfocarse en enseñar detalles concretos, por ejemplo áreas de notación en lugar de atender metas de aprendizaje más abstractas como el entendimiento general y estructura".

Esta problemática no es nueva y es un tema de interés para la comunidad interesada en la enseñanza de las ciencias computacionales, como

prueba pueden observarse los diferentes espacios dedicados a esta temática en el Grupo de Interés Especial en la Educación de Ciencias Computacionales de la *Association for Computer Machinery* (ACM- SIGCSE por sus siglas en inglés) (ACM, 2007).

En particular en el programa de ISC de la UACJ, los docentes han detectado y manifestado reiteradamente en reuniones de academia las escazas habilidades que tienen los alumnos en el desarrollo de las prácticas que involucran resolución de problemas algorítmicos complejos. De acuerdo al plan de estudios del programa en cuestión, estas habilidades deben ser adquiridas en los primeros semestres, específicamente en las materias de Introducción a las computadoras, Programación I y Programación II. Actualmente en el Departamento de Ingeniería Eléctrica y Computación (DIEC), instancia a la que pertenece el programa de ISC, no existe un estudio formal de las competencias en el desarrollo de software de los alumnos que concluyen estas materias, por lo que no se conoce con certeza la causa de la deficiencia en alumnos.

Es claro que para este rubro la competencia involucra conocimiento, pero enseñado y adquirido por el estudiante de una manera diferente a la que actualmente se utiliza.

Desarrollo de proyectos.

Uno de los principales problemas en la industria de desarrollo de software en México es que en la mayoría de las empresas dedicadas a este rubro, el software se produce de forma artesanal. Este concepto se contrapone al de fábrica de software, que según Peñaloza (2007), es "una organización capaz de manufacturar productos con calidad aceptada en el ámbito mundial bajo criterios de rentabilidad, planificación, diseño y organización". Generalmente en una empresa de tipo artesanal no se lleva a cabo un proceso formal de desarrollo. Lo que ocasiona que las empresas desconozcan en qué parte del proceso están fallando y no pueden mejorar sus procesos.

A nivel mundial una forma de diagnosticar la capacidad de una empresa es el modelo CMMI, sin ser un estrictamente un estándar, rápidamente se ha posicionado como tal: CMMI es un modelo que mide la capacidad de las empresas de software para incrementar el rendimiento de sus procesos de negocios. El modelo describe cinco niveles de madurez, cada uno de los cuales indica el nivel de rendimiento de la empresa. Según Peñaloza (2007) para que México se posicione en el mercado internacional es necesario que las empresas cuenten con al menos el nivel 3 de CMMI.

Una respuesta nacional ha sido la creación de MoProSoft, un modelo de procesos de software que a partir del año de 2005 es una norma nacional no obligatoria (NMX-059- NYCE-2005) bajo el nombre: Tecnología de la Información-Software-Modelos de procesos y de evaluación para desarrollo y mantenimiento de software. Este modelo propone procesos a tres niveles: Alta Dirección, Gerencia y Operación.

A nivel nacional no son muchas las empresas en algún nivel reconocido del CMMI y aún menos las que han incorporado MoProSoft; sin embargo esto ha ido cambiando sustancialmente. Es por eso que se considera que se deben hacer esfuerzos en las instituciones educativas para que los estudiantes egresen con las competencias necesarias que les permita entender la importancia de seguir procesos de software, implementarlos en una empresa de desarrollo y contar con la habilidad de adaptarse a un proceso de desarrollo en caso de que la compañía ya cuente con uno.

Para las instituciones educativas, este rubro de competencias conlleva dos problemas, el primero es que actualmente el programa de ISC está más encaminado a la tecnología y el segundo, es que los estudiantes, generación identificada con el avance en el desarrollo tecnológico, difícilmente ven la importancia de aprender acerca de los procesos de desarrollo.

Asociamos a este rubro competencias de conocimiento, sociales y personales.

Trabajo en equipo.

La creciente complejidad de los productos de software implica que la gran mayoría de estos productos exijan ser realizados por equipos de trabajo. Es prácticamente imposible imaginar aún que un programador de manera individual pueda construir un sistema complejo y crítico de manera completa. Como en cualquier otra organización, el capital humano es, en la industria del software, crítico tanto en su trabajo individual como en su trabajo en conjunto.

Por ejemplo, los modelos de procesos que en la actualidad marcan la pauta en la industria como

lo son Rational Unified Process (RUP) o Team Software Process (TSP) se caracterizan por el énfasis en el trabajo en equipo. En estos procesos se considera que las personas que conforman un equipo de desarrollo jueguen diversos roles que les permita alcanzar las metas establecidas. Es decir, entregar el producto a tiempo, con el presupuesto asignado y con la funcionalidad que espera el cliente.

Aunado a esto, el crecimiento de aplicaciones en internet facilita la realización de proyectos en forma asincrónica y sin necesidad de estar en el mismo lugar. Esto conlleva a que los individuos desarrollen habilidades de trabajo en equipo utilizando las tecnologías de la información. La utilización de herramientas como los foros de discusión, el correo electrónico, etc. y habilidades de comunicación específicas como el de comunicación oral y escrita, y trabajo colaborativo en red.

Según Hogan y Thomas (2005) el trabajo en equipo es una de las principales competencias que deberían tener los egresados de las carreras de desarrollo de software. En especial menciona las competencias de comunicación y manejo de tiempo. Pero, en general, los desarrolladores de software deben aprender a trabajar efectivamente en equipo. Sin embargo, las instituciones educativas contraponen esta tendencia al premiar y favorecer el trabajo individual. Sobre todo en las materias relacionadas a la algoritmia y resolución de problemas. Donde, por lo regular, se enseña a programar individualmente y el trabajo en equipo es considerado una forma de "trampa" y penalizado por los profesores.

Frecuentemente se olvida que los productos de software no tendrían sentido sin la "gente": lo necesita la gente, lo hace gente y va dirigido a gente". El factor humano y todo lo que ello incluye, es vital en el desarrollo de software.

Es claro que las competencias se ubican en conocimiento, sociales y personales. Una buena aproximación se puede encontrar en (Acuna, Juristo, Moreno, & Mon, 2005).

Administración de proyectos.

La administración de proyectos es un área estudiada con bastante profundidad, y cabría la pregunta de en qué es diferente un proyecto de desarrollo de software a algún otro. La respuesta es simple pero a la vez tiene implicaciones profundas, tiene que ver con una de las características del software, es intangible; y la inmensa mayoría de proyectos terminan con un producto tangible. Una práctica común es que los puestos de administración de proyectos de software sean ocupados por personas que tienen los conocimientos de administración pero a nivel de empresas no relacionadas con el desarrollo de software. Se da por sentado que cualquier administrador puede ser adecuado para este tipo de proyectos, por muy bueno que sea, la verdad es que es necesario tener otro tipo de conocimientos que no se adquieren en un programa de estudios de administración de empresas tradicional.

J. Fernando Naveda and Stephen B. Seidman (Pyster Arthur B., 2005) escriben acerca de la emergente certificación de los ingenieros del software, siguiendo los patrones establecidos para sus miembros por el Project Management Institute y el International Council on Systems Engineering.

Si el trabajo en equipo es de suma importancia para el desarrollo de software, la administración de estos equipos y por lo tanto del talento humano y sus respectivas actividades lo es también. Muchos de los problemas que se presentan en el desarrollo de software no son técnicos, algunos autores como Watts Humphrey señalan que el 80% de los fracasos en los proyectos de software se deben a problemas de relaciones humanas a una mala administración del recurso humano.

En la actualidad, una tendencia es la distribución geográfica del trabajo; cada vez más es posible encontrar desarrollo de productos de software construidos en diferentes partes del mundo. Se requiere la habilidad de administrar este tipo de proyectos.

Según Clealand (1998) "La administración de proyectos es la aplicación del enfoque de sistemas para la administración de tareas tecnológicas complejas o de proyectos cuyos objetivos se establecen explícitamente en términos de tiempo, costos y parámetros de realización"

Un nivel de competencia deseable en los administradores de proyectos es que demuestren un avanzado conocimiento de los métodos y técnicas para la gestión de productos de software complejos. Este conocimiento también se utilizará en el apoyo y formación de los métodos y técnicas de gestión de software.

Es evidente que para cada nivel habría que agregar la gestión del recurso humano. Las

competencias, para este rubro, se ubican en conocimiento, sociales y personales.

Entorno

En el paquete de entorno, proponemos agrupar todas aquellas características que se identifiquen como necesarias y suficientes, y que conjuntamente con las agrupadas en los otros paquetes permitan alcanzar los objetivos que se plantean en los diferentes programas y planes como los señalados (ProSoft, Plan Estratégico de la Ciudad)

Al inicio de la sección 3 mencionamos que desde nuestro punto de vista, el entorno impone ciertas condiciones. Brevemente podemos señalar lo siguiente; a nivel local existe un plan estratégico que incluye considerar a la ciudad como posible polo de atracción de empresas transnacionales desarrolladoras de software y a la vez crear un clima que también propicie el nacimiento de empresas locales dedicadas a este ramo; es decir que favorezca la inversión local y nacional en generar empleos de mejor calidad que los existentes. A nivel estatal, al igual que a nivel nacional, la política económica considera a las tecnologías de información y en particular a la industria de desarrollo de software, como un camino que el país debería tomar.

Es posible distinguir la necesidad de una cultura empresarial aunada a una cultura de innovación tecnológica; agentes de cambio, no solo para las empresas que ya existen, sino para atraer y fomentar la generación de empresas

Conclusión

En este acercamiento preliminar se identificaron cuatro categorías de competencias: algoritmia y resolución de problemas, trabajo en equipo, desarrollo de software y administración de proyectos. Lo que permite tener una visión panorámica de los diferentes aspectos que debieran cumplir un ingeniero de software. El proceso de identificación de competencias no solo involucra la identificación de competencias relacionadas con el conocimiento, sino que también se deben tomar en cuenta las características sociales y personales, así como condiciones requeridas por el entorno, este último no ha sido hasta el momento referido explícitamente en ningún trabajo anterior.

Como trabajo futuro creemos que debemos abarcar al menos tres líneas: una es madurar el modelo o marco propuesto y que esto permita ser una referencia de mayor apoyo y trascendencia; otra es configurar un vocabulario que de más expresividad, por ejemplo evitamos en lo posible, por el momento, utilizar términos como destreza, habilidad, aptitud, actitud, capacidad. etc., lo que implica definir términos de manera clara para este dominio en particular; la tercera línea es desarrollar propiamente el modelo educativo basado en competencias y a la par un modelo de evaluación que permita diagnosticar si dicho modelo produce lo que se espera.

Palabras Clave:
Competencias, Modelo Educativo, algoritmia, trabajo en equipo, desarrollo y administración de proyectos.

Referencias

ACM. 2007. *Association for Computing Machinery.* Recuperado el 12 de Octubre de 2007, de www.acm.org

Acuna, S., Juristo, N., Moreno, A., & Mon, A. 2005. A software process model handbook for incorporating peoples´s capabilities. *Springler* , 12-16.

Bailey, J., & Stefaniak, G. 2001. Industry perceptions of knowledges, skills and abilities needed by computer programmers. *SIGCPR* , 93-99.

Beard, D., & Schwieger, D. 2007. Incorporating Soft Skills into Accounting and MIS Curricula. *SIGMIS-CPR* , 179-185.

Beaver, J., & Schiavone, G. 2006. The effects of Development Team Skill on Software Product Quality. *ACM SIGSOFTS Software Engineering Notes* , 1-5.

Bennedsen, S. y. 2006. What do teachers teach in introductory programming. *IECR'06, ACM* .

Clealand, D. 1998. *Manual para la administración de proyectos.* México: Continental.

Fernández, J., García, M., Camacho, D., & Evans, A. 2006. Software Engineering Industry Experience - The Key to Succes. *Journal of Computing Sciences in Colleges* , 230-236.

Hogan, J., & Thomas, R. 2005. Developing the software engineering team. *Australasian Computing Education Conference* , 203-210.

James, H., & Thomas, R. 2005. Developing the software engineering team. *Australasian Computing Education Conference* , 203-210.

Joyanes, L. 2006. *C++, Algoritmos, estructuras de datos y objetos.* España: Mc Graw Hill.

Liu, J., Marsaglia, J., & Olson, D. 2002. Teaching software engineering to make student ready for the real World. *JCSC 18* .

Orsted, M. 2000. Software development engineer in Microsoft: a subjective view of soft skills required. *ICSE 2000* , 539-540.

Peñaloza, M. (25 de Octubre de 2007). *La industrica del Software, una oportunidad para México.* (UNAM, Ed.) Recuperado el 5 de Noviembre de 2007, de Enterate en línea: http://www.enterate

.unam.mx/Articulos/2002/enero/software.htm

Pyster Arthur B., T. R. 2005. Software Engineering Project Management 20 Years Later. *IEEE Software* , 18-21.

Secretaría de Economía. 2007. *Industria y Comercio*. Recuperado el 18 de Octubre de 2007, de http://www.economia.gob.mx/?P=1128