PROPUESTA DE INCLUSIÓN DE MDA EN INGENIERÍA DE SISTEMAS DE LA UNIVERSIDAD TECNOLÓGICA DE PEREIRA

Proposal of MDA inclusion in Systems Engineering of the Universidad Tecnologica de Pereira

JORGE MARIO GÓMEZ
Estudiante de Ingeniería de Sistemas y Computación
Universidad Tecnológica de Pereira
link2mario@gmail.com

HÉCTOR FABIO SÁENZ
Estudiante de Ingeniería de Sistemas y Computación
Universidad Tecnológica de Pereira
hsaenz@gmail.com

RESUMEN

Debido a la desactualización del programa de Ingeniería de Sistemas de la Universidad Tecnológica de Pereira, y específicamente del área de Ingeniería de Software respecto a los avances, que en este campo, se han dado en los últimos años, se dispone a elaborar una propuesta de inclusión del tema "Arquitectura de software dirigido por modelos" en dicho programa. Tal propuesta tendrá un componente teórico, una temática específica y un modelo pedagógico de integración.

PALABRAS CLAVES: Aprendizaje basado en problemas, Arquitectura de Software dirigido por modelos, Desarrollo de Software, Ingeniería de Sistemas, Ingeniería de Software, Modelos, Pensum, Swebok.

ABSTRACT

Due to the obsolescence of the Systems Engineering program at the Universidad Tecnologica de Pereira, and specifically the Software Engineering area regarding the progress in this field have occurred in recent years, will draw up a proposal for the topic "Model-driven Architecture" in the program. This proposal will have a theoretical component, a specific theme and a pedagogical model of integration..

KEYWORDS: *Curriculum, Model Driven Architecture, Models, Problem based learning, Software Development, Software Engineering, Swebok, Systems Engineering.*

1. INTRODUCCIÓN

El pensum de Ingeniería de Sistemas, de la Universidad Tecnológica de Pereira, que define el perfil profesional de sus egresados no ha tenido una transformación curricular en los últimos años, a pesar de las discusiones que al respecto se han dado y que los avances tecnológicos para esta carrera así lo exigen.

La Ingeniería de Software como una de las áreas de conocimiento en el programa de Ingeniería de Sistemas de la Universidad Tecnológica de Pereira, requiere ser actualizada con las tendencias que actualmente se dan en la industria del desarrollo de software. Uno de los retos de la industria del desarrollo de software es mejorar el desempeño para maximizar la productividad y las ganancias y para lograrlo es necesario "cambiar la forma de trabajar, maximizando el reúso, no desgastándose en diseño, codificación y pruebas exhaustivas, realizando programación en el nivel de ingeniería de modelos y requisitos" [2].

Lo planteado anteriormente muestra la necesidad de proponer estrategias de trabajo que potencien el reúso a un alto nivel de abstracción. Una de los avances en esta área es la "Arquitectura dirigida por modelos" (MDA[1]) que propone formas de reúso basadas en la manera como se hace abstracción de un problema y se especifica una propuesta de solución.

La tendencia marcada al desarrollo de nuevas tecnologías basada en la ingeniería por modelos está fundamentada en MDA, la cuál es definida por el Object Management Group (OMG[2]).

OMG a través de MDA ofrece un enfoque abierto y neutral al reto presentado por los cambios en los modelos de negocio y en la tecnología, basado en sus ya reconocidos y establecidos estándares tales como UML (Unified Modeling Language), MOF (Meta-Object Facility), entre otros.

Organizaciones tales como The U.S. Government Intelligence Agency, The National Cancer Institute, ABB, Deutsche Bank Bauspar AG, entre muchos otros

[1] http://www.omg.org/mda

[2] http://www.omg.org

Fecha de Recepción:
Fecha de Aceptación:

han implementado exitosamente MDA en sus desarrollos[3].

El objetivo de este documento es elaborar una propuesta de inclusión del tema "Arquitectura de software dirigido por modelos" en el pensum del programa de Ingeniería de Sistemas, a través de los siguientes lineamientos:

- Definir la temática de MDA para la propuesta.
- Definir un modelo de integración de MDA al pensum de la carrera.
- Adaptar un modelo pedagógico para la enseñanza de MDA en el contexto de la ingeniería de software.

2. MODEL DRIVEN ARCHITECTURE (MDA) [1], [2]

Ante el gran reto que tienen las empresas desarrolladoras de software de mejorar el desempeño para obtener un mayor índice de ganancias, se plantea la necesidad de aplicar el reúso en un alto nivel de abstracción. En este punto metodologías como la arquitectura dirigida por modelos (MDA) definida por la OMG proponen pensar tempranamente en formas de reúso basadas en la manera como se hace abstracción de un problema y se especifica una propuesta de solución.

Para los desarrolladores es un gran reto definir una estructura en la que el actor principal sean los modelos que definen la lógica del negocio y que por medio de mecanismos predefinidos y herramientas sean transformados progresivamente hasta llegar al producto final. Esto es lo que conocemos como MDA. Inicia por la idea, ya conocida y ampliamente establecida, de separar la especificación de la operación de un sistema, de los detalles sobre la forma en que dicho sistema usa las capacidades de la plataforma. Entre los aspectos más importantes en MDA, tenemos:

1) Introducción al proceso en MDA (Transformación general de modelos) [3]**.** Transformar un modelo es el proceso en el que un modelo fuente se convierte en otro destino con el uso de unas reglas de transformación determinadas. En las reglas de transformación se describe como un determinado elemento del modelo fuente puede ser transformado a un elemento del modelo destino.

2) Transformaciones de modelos [4]**.** La transformación de modelos es el proceso de convertir un modelo en otro modelo del mismo sistema. La transformación de modelos puede ser horizontal o vertical, el primero es una mejora de un modelo en un mismo nivel y en la segunda es la evolución del modelo a otro nivel dentro del modelado MDA. Aquí intervienen estándares propuestos por la OMG como QVT (Query/View/Transformation, definido por la OMG) y OCL (Lenguaje basado en restricciones).

3) Modelo Independiente de la Computación (CIM) [2], [5], [6]**.** En CIM se modelan los requisitos del sistema, se describe la situación en la que el sistema será utilizado y sirve de enlace entre los expertos en el dominio del problema y sus requisitos con los expertos en el diseño y construcción de software. Los requisitos pueden ser representados mediante diagramas de caso de uso, de actividad y de secuencia. Entre los tipos de requisitos más comunes obtenidos en CIM se encuentran los requisitos de usuario, los funcionales, los no funcionales y los organizacionales.

4) Transformación de CIM a PIM en MDA [7], [8]**.** La creación del nivel de CIM no está unificada ahora y no usa estándares unificados pero es asumido que este nivel es representado por el modelo de procesos de negocio (BPM). La transformación de CIM a PIM es presentada como el enfoque ordenado. Esta usa los diagramas de actividad de UML2 que modelan los procesos del negocio. Estas son modeladas como las tareas del usuario. Desde los diagramas de actividad, los requerimientos del sistema son especificados. A partir de los elementos del modelo de requerimientos los componentes del sistema son creados. Por último, un conjunto de prototipos de negocio ayudan a transformar los componentes del sistema a la capa PIM en detalle.

5) Modelo independiente de la plataforma (PIM) [2], [5], [6]**.** Es una visión de un sistema en la que no se incluye la plataforma donde será implementado, y se busca ese grado de independencia con el fin de ser apto para usarlo en diferentes plataformas posteriormente y surgen como resultado de análisis y diseño. Tal modelo tendrá cierto nivel de abstracción que no cambiará sin importar la plataforma que sea elegida para su implementación. Las principales características que debe tener el PIM necesarias para la transformación de PIM a PSM son:

- Formación del modelo abstracto.
- Describir el comportamiento del sistema, aspectos funcionales y no funcionales independientes del entorno de computación y tecnologías de implementación. Y que puedan ser reutilizados en múltiples plataformas.
- Los requisitos del negocio se especifican utilizando diagramas UML.
- El sistema es modelado desde el punto de vista que mejor soporte los requisitos del usuario final.
- Que sea independiente de la implementación de la plataforma/tecnología.

6) Transformación de PIM a PSM [9]**.** Para llevar a cabo la transformación de PIM a PSM se utiliza el

[3] Para profundizar sobre estos y otros desarrollos: http://www.omg.org/mda/products_success.htm

estándar de transformación de modelos QVT, que a su vez presenta dos tipos de notaciones, gráfica y textual. Una transformación tiene estas características:

- Dos o más dominios: cada dominio identifica un modelo candidato (como PIM o PSM) y un conjunto correspondiente de elementos definidos por medio de patrones. Un patrón de dominio puede ser considerado una plantilla de objeto. Sus propiedades y asociaciones pueden ser asignadas, modificadas o creadas en un modelo candidato para satisfacer la relación.
- Un dominio de relación: especifica el tipo de relación entre dominios, y puede ser para verificar si el modelo contiene una correspondencia válida que satisfaga la relación, o puede ser que cuando el patrón de dominio no corresponda, los elementos del modelo destino puedan ser creados, eliminados o modificados para satisfacer la relación. Además, para cada dominio el nombre de la metamodelo subyacente es especificado.
- La sentencia when: especifica las condiciones que deben ser satisfechas para ejecutar la transformación.
- La sentencia where: especifica las condiciones que deben ser satisfechas por todos los elementos del modelo involucrados en la relación: ej. las postcondiciones.
- Una transformación contiene dos tipos de relaciones: de alto nivel y de bajo nivel. La ejecución de una transformación requiere el cumplimiento de todas las relaciones de alto nivel, mientras que las relaciones de bajo nivel requieren cumplimiento únicamente cuando son directamente o transitivamente invocadas desde la sentencia where de otra relación.

7) Modelo especifico de la plataforma (PSM) [4]**.** Es una vista del sistema para una plataforma específica. Éste combina la especificación del sistema hecha en el PIM, con los detalles que especifican la manera en que dicho sistema usa una plataforma particular. MDA no implica el usar solo UML, en su lugar, la tecnología crucial es MOF (MetaObject Facility) y la definición de metamodelos son instancias del metametamodelo MOF.
Estos metamodelos definen un lenguaje de modelado de dominio específico, que presenta una solución al modelado de distintos tipos de sistemas de software. Un PSM será una aplicación, si proporciona toda la información necesaria para construir un sistema y ponerlo en funcionamiento, o puede actuar como un PIM que se utiliza para mayor refinamiento a un PSM que puede ser directamente implementado.

8) De PSM a código e implementación. La transformación de PSM a código es análoga a la transformación de PIM a PSM, y se puede ejecutar de una manera similar. Puede ser deseable soportar algunas diferentes configuraciones de implementaciones para llevar al sistema a un entorno específico.

9) Fortalezas y debilidades de MDA [10]**.** Identificarlas puede ayudar a los desarrolladores a formular estrategias para su implementación en el desarrollo de Software, se destacan algunas de ellas en este artículo:

Fortalezas:

- Productividad: los desarrolladores no tienen que escribir mucho código, ya que gran parte de ese código se genera automáticamente a partir de los modelos PIM.
- Transformación automática: Las herramientas de transformación son responsables de implementar las transformaciones de modelos.
- Portabilidad e independencia de plataforma: Los modelos PIM muestran una vista independiente de la solución para que puedan transformarse en múltiples modelos PSM para diferentes plataformas.
- Interoperabilidad: Diferentes modelos PSM se pueden construir a partir de un modelo PIM, por lo que la interoperabilidad multiplataforma es mucho mayor.
- Aumentar el nivel de abstracción: MDA separa los modelos PIM de sus contrapartes PSM, disminuyendo la complejidad mediante la promoción de modelos a unidades de abstracción.
- Mayor facilidad de mantenimiento: En MDA, el mantenimiento es apoyado por la separación de la funcionalidad general (PIM) de las características específicas de la plataforma (PSM.
- Roles especializados: Cada fase del desarrollo puede ser desempeñado por distintos expertos en cada campo y así dividir el trabajo dejando que cada experto se encargue solo de lo que sabe.

Debilidades:

- Potencial para la inconsistencia del modelo: Un concepto puede ser presentado en diferentes niveles de abstracción y puntos de vista a través de diferentes modelos, como los modelos evolucionan gradualmente, su sincronización se convierte en una tarea compleja y difícil.
- La falta de un proceso de desarrollo específico: MDA no define ni prescribe un proceso de desarrollo concreto. Las metodologías basadas en MDA necesitan definir sus propias actividades, ciclos de vida, artefactos, roles y directrices; lo que puede resultar en procesos que se desvían de las normas de la MDA.
- Problemas de la calidad del modelo: Calidad de las herramientas para la transformación del modelo; calidad del lenguaje de modelado y sus criterios de calidad; la conciliación de los aspectos de calidad conflictivos entre los modelos; medición, mejora, gestión y control de calidad de los modelos.
- Complejidad de la trazabilidad del modelo: Debido a la multiplicidad de modelos y la calidad requerida, la trazabilidad es una característica difícil de implementar en los procesos de desarrollo basados en MDA.
- Limitaciones en la escalabilidad: A medida que avanza el proceso de desarrollo, una gran cantidad de modelos son producidos por lo que el mantenimiento y la manipulación se hacen cada vez más difíciles.

- Verificación del modelo y problemas de validación: El soporte para la generación de casos de prueba basados en modelos y la verificación o validación formal de los modelos es una crucial (aunque difícil) tarea.

11) Tecnologías utilizadas en MDA[4]. Entre las herramientas más importantes que intervienen en MDA y que sirven de soporte para su implementación, se tiene:

- Enterprise Architect.
- OptimalJ.
- ArcStyler.
- AndroMDA.
- Codagen Architect.
- IQGen.
- MagicDraw.
- Rational Software Architect for WebSphere Software.

3. MODELO PEDAGÓGICO Y DE INTEGRACIÓN AL PENSUM DE INGENIERÍA DE SISTEMAS DE LA UTP

1) Definición del modelo de integración. La línea de Ingeniería de Software del pensum de Ingeniería de Sistemas tiene como base el Swebok[11], que es la guía del conocimiento y enseñanza para la Ingeniería de Software creado por la Software Engineering Coordinating Committee y promovido por la IEEE[5].

Una vista de las materias Ingeniería de Software I y II:

Ingeniería de Software I[6].

El programa de Ingeniería de Software I está basado en los capítulos de requerimientos de software, diseño de software, construcción de software, pruebas de software y mantenimiento de software, además, agrega una introducción a la ingeniería de software y a la historia de UML.

Ingeniería de Software II[7].

El programa de Ingeniería de Software II está basado en los capítulos de gestión de ingeniería de software, gestión de la configuración de software, el proceso de ingeniería de software, y estándares de calidad de software, además se adicionaron los capítulos referencia histórica a las metodologías de desarrollo de software, personalización del software, estándares y calidad de software, e implantación de software.

Laboratorio de Software

Esta asignatura se ha orientado al desarrollo de un proyecto de ingeniería de software donde se ponen en práctica los conocimientos de las asignaturas ingeniería de software I y II. Más que hacer un balance de esta orientación lo que se presenta es la propuesta de fortalecimiento de la línea de ingeniería de software con la inclusión de al menos una asignatura, la cual entre otros temas debería abordar la arquitectura dirigida por modelos.

En el siguiente apartado presentaremos los prerrequisitos teóricos para la materia propuesta en base a la temática de MDA.

2) Prerrequisitos teóricos. En primera instancia, los conceptos básicos de Ingeniería de Software. Entre los conocimientos necesarios se pueden destacar la gestión de los requerimientos y el entorno empresarial en que estos son concebidos.

El reúso de software como una de las prácticas modernas apropiadas en la industria del desarrollo de software y que facilitan la especialización y depuración de partes de un sistema. Modelado de software con UML, ya que la base de MDA son los modelos.

Arquitecturas y diseño de software, patrones de diseño y de arquitectura, programación orientada a objetos también son importantes para el aprendizaje de MDA.

El uso de herramientas CASE para el modelado y las de última generación para el soporte de transformaciones, tema fundamental en MDA y XML como base para realizar transformaciones. El lenguaje de restricciones OCL para aumentar la información de las vistas del sistema a través de un modelo.

Refactorización como una forma de optimizar el código, muy usado en la automatización en la generación de código fuente.

Estos presupuestos teóricos se proponen sean vistos dentro del temario de MDA, aunque no hacen parte únicamente de este. Se han unificado en el numeral denominado “Fundamentos de MDA” en el contenido propuesto en el siguiente apartado.

3) Contenido de la asignatura propuesta para la incorporación de MDA en el pensum.

---

[4] Para conocer más herramientas se puede acceder a este enlace de la OMG http://www.omg.org/mda/committed-products.htm

[5] http://www.ieee.org

[6] Ingeniería de Software I, Universidad Tecnológica de Pereira: http://isc.utp.edu.co/pensum/IS714%20Ingenieria%20de%20software%20I.pdf

[7] Ingeniería de Software II, Universidad Tecnológica de Pereira: http://isc.utp.edu.co/pensum/IS813%20ingenieria%20de%20software%20II.pdf

Capítulo 1, Introducción al Desarrollo Dirigido por Modelos (MDA) en el que se habla de motivación y conceptos, MDE y MDA (visión de la OMG), el principio básico de MDA (lenguajes de modelado o específicos del dominio, DSL), generación automática de código y casos de éxito en MDA.

Capítulo 2, Fundamentos de MDA, se proponen los temas: concepto de metamodelo, lenguajes de metamodelado, el lenguaje de restricciones OCL, los perfiles UML y MOF.

Capítulo 3, Modelos, se proponen los temas: Modelos Independientes de la Computación (CIM), Modelos Independientes de la Plataforma (PIM), Modelos Específicos de la Plataforma (PSM) y ejemplos de estos.

Capítulo 4, Transformaciones de modelos, se proponen los temas: Propiedades de los modelos, Lenguajes de transformación de modelos (Características y Clasificación), QVT, Ejemplos de transformaciones

Capítulo 5, Herramientas MDA, se proponen los temas: Herramientas de modelado CIM, PIM, PSM y Herramientas para transformaciones

Capítulo 6, Aplicaciones del MDA, Generación de código a partir de modelos vs. Obtención de modelos a partir de código, Automatización del uso de framework

Se propone una materia que vista en 16 semanas tendría una intensidad horaria de 3 horas semanales.

4) Modelo pedagógico para la enseñanza de MDA en la ingeniería [12], [13], [14], [15]. Con el fin de generar en el estudiante de Ingeniería de Software habilidades propias como capacidad de análisis y síntesis, de planeación y organización, comunicación, identificación y solución de problemas, investigación, trabajo en equipo, aprendizaje auto-dirigido, creatividad y liderazgo se propone un modelo pedagógico para la enseñanza de MDA centrada en el estudiante y en su aprendizaje, en donde el estudiante deje de ser un ente pasivo en este proceso y pase a dirigir su propio aprendizaje con una asesoría constante por parte del profesor.

El Aprendizaje Basado en Problemas (Problem Based Learning – PBL) es considerado como uno de los métodos adecuados para los nuevos modelos de educación superior basados en el aprendizaje y diversas universidades utilizan PBL como el núcleo de su estrategia de formación [12].

PBL propone que el profesor le presente al estudiante un problema (por lo general presentado intencionalmente sin estructura alguna para dar una libertad de interpretación al estudiante) para que éste tome la iniciativa definiendo precisamente el problema, averiguando lo que saben y lo que tienen que determinar y cómo proceder para solucionarlo.

El estudiante formula y evalúa soluciones alternativas, seleccionando la mejor de todas y evaluando lo aprendido, ya cuando identifica la necesidad de la instrucción para nuevo material, el instructor es la guía para que aprenda la información requerida por sí mismo.

Mediante esta metodología, los estudiantes asumen una gran responsabilidad y libertad de acción llevando a cabo un proceso de auto-aprendizaje mientras que el profesor se encargará de asegurar la participación y motivación de los estudiantes, ayudando a los estudiantes a conocer las fuentes a consultar así como identificar las dificultades encontradas dentro del proceso, en esta metodología claramente el papel del profesor obtiene una nueva función y es orientar y servir de mentor al estudiante para que tenga un papel más activo en el proceso de aprendizaje, sin ser un facilitador.

Las diferencias de esta propuesta con la enseñanza tradicional impartida actualmente en la Universidad son notorias, sobre todo cuando en el proceso de aprendizaje los estudiantes deben dominar primero los principios y las teorías de la disciplina antes de que se les pida resolver los problemas de fondo en esa disciplina, enfocando al estudiante como un individuo que trabaja mejor solo que mediante trabajo cooperativo. Este cambio demandaría una adecuada preparación de los docentes que imparten la materia sobre lo que es aprendizaje activo y PBL, para que sepan aplicarlo en sus cursos.

4. CONCLUSIONES

- Después de realizar este proyecto podemos concluir que en la formación de un Ingeniero de Sistemas, en el área de la Ingeniería de Software y con las tendencias que en este campo se desarrollan en el mundo, es necesario incluir la arquitectura dirigida por modelos.

- La naturaleza del pensum de Ingeniería de Sistemas de la Universidad Tecnológica de Pereira, el cual está basado en la guía propuesta por la IEEE para la enseñanza de Ingeniería de Software, permite una mejor adaptación de MDA a través de una tercera materia, en la cual, y con base en los conocimientos dados en las dos primeras ingenierías de software, se construirán los conocimientos necesarios de MDA.

- En la manera de enseñar Ingeniería de Software en el programa de Ingeniería de Sistemas de la Universidad Tecnológica de Pereira en la cual “Los temas se exponen en forma magistral y se complementan con

ejemplos presentados por el profesor y ejercicios propuestos por el profesor y desarrollados por los estudiantes"[8] , y ante los modelos pedagógicos que en la actualidad son apropiados por las más importantes universidades del mundo, nos parece pertinente implementar el aprendizaje basado en problemas expuesto anteriormente en este documento.

- Se cumplieron los objetivos del proyecto, al definir una temática para la propuesta de MDA, y un modelo de integración, que consiste en una tercera materia para la línea de Ingeniería de Software del programa de Ingeniería de Sistemas de la Universidad Tecnológica de Pereira. Además de proponer un modelo pedagógico para la enseñanza del mismo.

N. BIBLIOGRAFÍA

[1] OBJECT MANAGEMENT GROUP. OMG Model Driven Architecture [En línea]. [citado en 2011-09-02]. Disponible en Internet: <http://www.omg.org/mda/>.

[2] QUINTERO, Juan Bernardo y ANAYA, Raquel. El MDA y el papel de los modelos en el proceso de desarrollo de software [en línea]. Medellín (Colombia). Escuela de Ingeniería de Antioquia. Diciembre 2007. Disponible en Internet: <http://revista.eia.edu.co/articulos8/Art.10.pdf>. ISSN 1794-1237.

[3] KARDOŠ, Martin y DROZDOVÁ, Matilda. Analytical Method of CIM to PIM Transformation in Model Driven Architecture (MDA). [En Línea]. Universidad de Zilina (Eslovaquia). Mayo 2010. [Citado en 2011-09-22]. Disponible en Internet: <http://hrcak.srce.hr/file/83906>.

[4] PÉREZ RUIZ, Jose Manuel Francisco y PIATTINI, Mario. Model Driven Engineering Aplicado a Business Process Management. Departamento de Tecnologías y Sistemas de Información, Universidad de Castilla-La Mancha. Marzo 2007. Madrid (España). [Citado en 2011-09-15]. Disponible en Internet: <www.uclm.es/dep/tsi/pdf/UCLM-TSI-002.pdf>.

[5] OBJECT MANAGEMENT GROUP. MDA Guide Version 1.0.1 [En línea]. Junio 2003 [citado en 2011-08-31]. Disponible en Internet: <http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf>.

[6] SINGH, Yashwant y SOOD, Manu. The Impact of the Computational Independent Model for Enterprise Information System Development. [En Línea]. Diciembre 2010. India. [Citado en 2011-09-21]. Disponible en Internet: <www.ijcaonline.org/volume11/number8/pxc3872153.pdf >.

[7] OBJECT MANAGEMENT GROUP. Model Driven Architecture (MDA) FAQ [En línea]. Agosto 2010 [citado en 2011-09-05]. Disponible en Internet: <http://www.omg.org/mda/faq_mda.htm>.

[8] CLEMENTS, Paul. A Survey of Architecture Description Languages. Proceedings of the International Workshop on Software Specification and Design. Alemania. 1996.

[9] SOLER, Emilio, et al. A set of QVT relations to transform PIM to PSM in the Design of Secure Data Warehouses. [En Línea]. 2007. [Citado en 2011-09-13]. Disponible en Internet: <http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4159859>.

[10] GHOLAMI, Mehdi Fahmideh y RAMSIN, Raman. Strategies for Improving MDA-Based Development Processes. [En Línea]. Tehran (Iran). 2010. [Citado en 2011-09-18]. Disponible en Internet: <http://ieeexplore.ieee.org.ezproxy.utp.edu.co/stamp/stamp.jsp?tp=&arnumber=5416102>.

[11] BOURQUE, Pierre, et al . Swebok. Guide to the Software Engineering Body of Knowledge. IEEE Computer Society. California (Estados Unidos). 2004.

[12] LACUESTA, Raquel; PALACIOS, Guillermo y FERNANDEZ, Luis. Active Learning through Problem Based Learning Methodology in Engineering Education. [En Línea]. Universidad de Zaragoza. Zaragoza (España). 2009. [Citado en 2011-10-15]. Disponible en Internet: <http://fie-conference.org/fie2009/papers/1338.pdf>

[13] BULLARD, Lisa G. y FELDER, Richard M. A Student-Centered Approach To Teaching Material And Energy Balances. [En Línea]. North Carolina State University. 2007. [Citado en 2011-10-16]. Disponible en Internet: <http://www4.ncsu.edu/unity/lockers/users/f/felder/public/Papers/StoichPap-pt1.pdf>.

[14] UNIVERSIDAD POLITÉCNICA DE MADRID. Aprendizaje Basado en Problemas. Guías rápidas sobre nuevas metodologías. [En Línea]. Madrid (España). 2008. [Citado en 2011-10-18]. Disponible en Internet: <http://innovacioneducativa.upm.es/guias/Aprendizaje_basado_en_problemas.pdf>.

[15] PRINCE, Michael y FELDER, Richard. The Many Faces of Inductive Teaching and Learning. [En Línea]. National Science Teachers Association (NSTA). 2007. [Citado en 2011-10-19]. Disponible en Internet: <http://www4.ncsu.edu/unity/lockers/users/f/felder/public/Papers/Inductive(JCST).pdf>.

[8] Modelo pedagógico propuesto en el programa de la materia: http://isc.utp.edu.co/pensum/IS714%20Ingenieria%20de%20software%20I.pdf