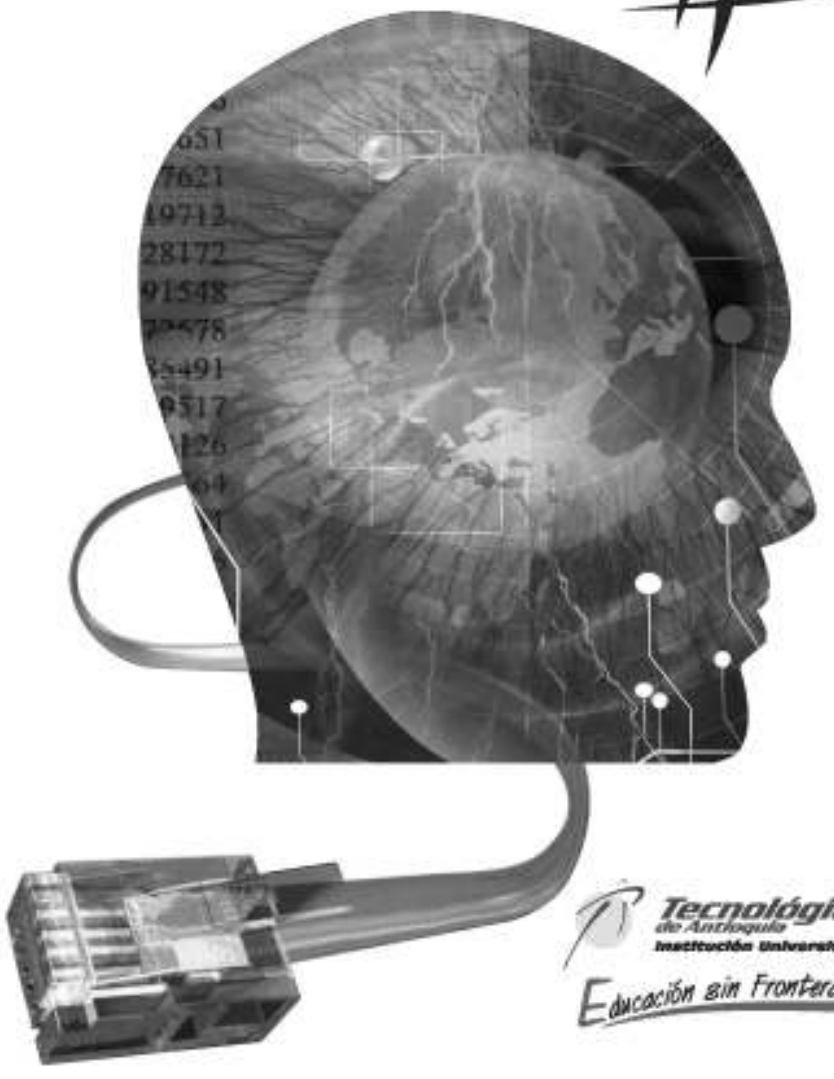
Cuaderno activa

GRUPO DE INVESTIGACIÓN
EN INGENIERÍA DE SOFTWARE
DEL TECNOLÓGICO DE ANTIOQUIA
GIISTA



Tecnológico de Antioquia
Institución Universitaria

Educación sin Fronteras

GOBERNACIÓN DE ANTIOQUIA

ISSN: 2027-8101

DISEÑO E IMPRESIÓN
Divegraficas Ltda.
Carrera 53 N.° 54 - 30 PBX: 511 76 16
Medellín, Colombia
www. divegraficas.com
Fecha de impresión: diciembre de 2011

Cuaderno

activa

GRUPO DE INVESTIGACIÓN
EN INGENIERÍA DE SOFTWARE
DEL TECNOLÓGICO DE ANTIOQUIA
EAFIT A

Contenido

Contenido

Editorial

La Revista Cuaderno ACTIVA nació con la expectativa de convertirse en un espacio académico e investigativo en las áreas de la Ingeniería de Software, la informática, la Electrónica y los campos afines a las tecnologías de la información y la comunicación, en la cual los investigadores nacionales e internacionales plasmen sus trabajos que puedan servir como un aporte significativo para la comunidad académica y un punto de partida para la comunidad científica en el área.

La primera edición de la revista Cuaderno ACTIVA se realizó en marzo de 2011 como marco del "I Seminario Internacional de Ingeniería de Software", celebrado en el Tecnológico de Antioquia en el mes de agosto del año 2010.

La Revista Cuaderno ACTIVA cuenta dentro de su Comité Científico y Editorial con profesionales nacionales e internacionales de gran prestigio académico e investigativo y con gran trayectoria en las áreas afines a la revista, todo esto evidenciado en publicaciones, investigaciones y libros.

La edición actual de la revista Cuaderno ACTIVA cuenta con una gran diversidad de artículos de alta calidad que hacen referencia a los campos de la Ingeniería de Software, la lógica de programación, la Matemática, la neumática entre otros. En esta edición participan investigadores de varias Universidades del país, incluyendo investigadores pertenecientes al grupo GIISTA del Tecnológico de Antioquia.

Con esta proyección, la revista busca constituirse como una de las más importantes en el área para contribuir a la consolidación de una comunidad científica que permee los círculos académicos e investigativos en los ámbitos regional, nacional e internacional.

Fabio Alberto Vargas Agudelo
Director grupo de Investigación GIISTA
Tecnológico de Antioquia

Recibido: 11 de junio 2011
Aprobado: 26 de octubre 2011

Transformación de lenguaje natural a lenguaje controlado para la educción de requisitos a partir de documentación técnica

Turning natural language into controlled language in order to educe requiriments from technical documentation

Bell Manrique Losada
Estudiante de Doctorado en Ingeniería, Profesora Asistente
Universidad de Medellín, bmanrique@udem.edu.co

Carlos Mario Zapata Jaramillo
Ph.D. en Ingeniería, Profesor Asociado Universidad Nacional de Colombia, sede Medellín, cmzapata@unal.edu.co

Resumen
En la ingeniería de software, la educción de requisitos es el proceso mediante el cual un analista intenta capturar los requisitos que tiene un interesado respecto de un futuro aplicativo de software. Este proceso implica una traducción de un lenguaje natural—del interesado—a un lenguaje controlado. Tradicionalmente, para obtener estos requisitos se utilizan técnicas relacionadas con entrevistas y diálogos, lo cual genera grandes esfuerzos de los analistas y pérdida de tiempo, por la complejidad que implica el procesamiento del lenguaje natural. No es común educir requisitos a partir de fuentes indirectas como documentación técnica (manuales de procedimientos, reglamentos y estatutos, entre otros). En este artículo se contextualiza la problemática asociada con la educción de requisitos y las brechas que existen entre los lenguajes controlados, para especificar requisitos, y el lenguaje natural del interesado. Finalmente, se describe el escenario que se plantea lograr con la puesta en marcha de estas propuestas, a partir de la revisión de literatura realizada.
Palabras clave: documentación técnica, educción de requisitos, lenguaje controlado, lenguaje natural.

Abstract
Analysts try to capture the stakeholder requirements related to a future software application through a requirements elicitation, a branch of software engineering. Requirements elicitation implies a translation

Recibido: 11 de junio 2011
Aprobado: 26 de octubre 2011

from a natural language—the stakeholder's—into a controlled language. Commonly, analysts use techniques related to dialogues and interviews in order to meet such requirements. These techniques demand a huge amount of analysis and a waste of time, due to the complexity of natural language processing. Eliciting requirements from indirect sources, like technical documentation (i.e., procedure manuals, regulations, and statutes) is not a common task. In this paper we contextualize the problems linked to the requirements elicitation process and the existing gaps between the controlled language—to specify requirements—and the stakeholder's natural language. Finally, we describe a research scenario proposed, based on a state-of-the-art review.
Keywords: controlled language, natural language, requirements elicitation, technical documentation.

Introducción

La educción es una de las primeras fases de la *ingeniería de requisitos* para el desarrollo de software, cuyo propósito principal es descubrir todos los requisitos que el futuro software necesita satisfacer, para que alcance los objetivos definidos (Cheng & Atlee, 2007). Se relaciona principalmente con la acción de comunicación analista-interesado, que busca recuperar información esencial y relevante acerca del dominio (expresada en lenguaje natural). Esta información se convertirá en la base de los requisitos y se captura con los interesados (cliente, usuario-final, experto del dominio, etc.; Kof, 2004). En este proceso son determinantes la exactitud y la precisión en los discursos en lenguaje natural, lenguaje en el cual, según Berry (2003), se escribe la gran mayoría de requisitos. Para el analista es muy importante identificar los conceptos y las relaciones entre conceptos que emplea el interesado, pues ellos se convertirán en la base del lenguaje común que deben entender el analista y el interesado. Lo anterior requiere, según Li *et al.* (2003), mayor intervención y transformación para las tareas posteriores de análisis y diseño del producto software.

Este proceso se puede visualizar como una 'traducción' de un lenguaje a otro, de manera tal que el analista (traductor) debe reconocer y entender símbolos expresados en un lenguaje natural de un universo de discurso (del interesado) y transformarlos en un conjunto de símbolos definidos en un lexicón de un lenguaje controlado (Castro *et al.,* 2009). Posteriormente, el analista debe representar estos símbolos en lenguajes técnicos (generalmente de tipo gráfico, como los esquemas conceptuales). Los interesados, por su parte, validan los requisitos capturados y plasmados en dichos modelos y esquemas conceptuales, a pesar de no comprenderlos suficientemente, pues los lenguajes técnicos se alejan del natural. Esta comunicación entre los participantes (interesado-analista) se torna más débil aún por las diferencias de formación y experiencia entre ellos, lo que genera problemas en la correcta validación de los requisitos, en el alcance de los modelos conceptuales generados y, finalmente, en los costos de ejecución del proceso de educción (Bolton *et al.,* 1994).

A partir de las técnicas de captura de requisitos utilizadas de forma tradicional (Christel & Kang, 1992), la intervención del interesado se suele estimar incorrectamente, pues la mayoría de las veces se realiza en forma de diálogos y entrevistas (Leite, 1987) con la resultante pérdida de tiempo, costos, coherencia, concisión, entre otros, como se verá más adelante. A pesar de los acercamientos propuestos en la literatura para reducir la brecha entre los universos de discurso del interesado y del analista, todavía se exige máxima intervención del analista en el proceso de educción, en la conversión entre la descripción de requisitos y modelos de diseño y en el método que guíe este proceso. Es necesario, entonces, lograr acercamientos entre los lenguajes controlados existentes para especificar los requisitos y el lenguaje natural del interesado. Este artículo presenta el área problemática que describe este escenario, la revisión de literatura asociada con el problema principal identificado y los resultados que se espera lograr con la ejecución de tales acciones.

El resto del artículo se organiza así: la Sección 2 describe el área problemática y el marco teórico-conceptual que lo sustenta; la Sección 3 presenta una serie de problemas identificados y, finalmente, el

problema de investigación que se propone abordar; la Sección 4 muestra, a manera de conclusiones, los resultados que se espera lograr con la ejecución de la propuesta y un acercamiento a la justificación del tipo de aporte.

Área problemática

Para abordar teórica y conceptualmente el tópico de transformación de lenguaje natural a lenguaje controlado en la educción de requisitos, es necesario partir de la definición de un objeto real de investigación. El objeto real se representa con la descripción de una necesidad de un interesado, expresada en *lenguaje natural* dentro de un documento técnico, y su traducción a un *lenguaje controlado* que especifica los requisitos (puede ser el lenguaje UN-Lencep; Zapata, 2007).

A partir del objeto real, se puede delimitar el objeto de estudio considerando dos conceptos básicos: lenguaje natural y lenguaje controlado. Estos conceptos se aplican el marco de la educción de requisitos utilizando técnicas de procesamiento del lenguaje natural y de la lingüística computacional.

Educción de requisitos

Es una de las primeras fases de la *ingeniería de requisitos* para el desarrollo de software, cuyo objetivo principal es descubrir todos los requisitos que el futuro software necesita satisfacer para que se considere de calidad. Para lograr este objetivo, se deben llevar a cabo, de manera iterativa e incremental, dos actividades primordiales: *educción de requisitos* y *análisis de requisitos*, involucrando un lenguaje natural y un lenguaje de modelado, respectivamente (Li *et al.*, 2003).

La fase de educción de requisitos tiene diferentes actividades, que incluyen: entendimiento del dominio de aplicación, captura y clasificación de requisitos, establecimiento de prioridades, resolución de conflictos y negociación de los requisitos del sistema (Robertson y Robertson, 2006). La educción de requisitos se relaciona, principalmente, con la acción de comunicación analista-interesado, la cual busca recuperar la información esencial y relevante acerca del dominio, obtener la base de los requisitos y extraerla de los interesados (cliente, usuario-final, experto del dominio, etc.).

Lenguaje natural y lenguaje controlado

Serrano (2005), citando a Ferdinand de Saussure, expresa que el lenguaje se concibe como el complemento de dos entidades: *lengua* y *habla*. El lenguaje es propiedad social, no individual, como la totalidad de los sistemas lingüísticos que emplean los miembros de una comunidad, es decir, es un sistema de signos. Así, de acuerdo con Vernengo (1996), el lenguaje se puede entender como un conjunto de oraciones gramaticalmente bien formadas conforme a reglas fonéticas, léxicas, sintácticas y semánticas correspondientes a un lenguaje natural cualquiera. En su estado normal, el lenguaje natural utiliza elementos gramaticales, como: sustantivo, verbo, adjetivo, pronombre, conjunción, preposición, adverbio y artículo.

El lenguaje, que desde Grecia se considera esencial para la naturaleza humana, resulta poco confiable cuando la comunicación requiere ciertos niveles de precisión y cuando las acciones futuras dependen de los participantes en el proceso comunicativo. En este sentido, es importante que en dominios como el de la educción de requisitos, donde son determinantes la exactitud y la precisión en los discursos en lenguaje natural, se pongan en claro reglas que determinen las relaciones entre ciertas expresiones formadas y los sentidos que ellas pretenden transmitir (Berry, 2003).

Por lenguaje natural se entiende la lengua utilizada normalmente en una comunidad de individuos para la comunicación de estos entre sí (Tendales, 2004). El lenguaje natural se caracteriza por su enorme capacidad y su riqueza comunicativa, su flexibilidad y la posibilidad de jugar con las palabras y con las expresiones, produciendo metáforas y ambigüedades. De lo anterior se deduce que, si bien el lenguaje natural es un instrumento idóneo para ciertos propósitos, no lo es igualmente para áreas científicas o ingenieriles donde se requiere un máximo de exactitud y precisión (Li *et al.*, 2003).

Según Berry (2003), la gran mayoría de requisitos se escribe en lenguaje natural. Para el analista es muy importante identificar los conceptos y relaciones entre conceptos que emplea el interesado, que constituyen la base del lenguaje común que deben entender el analista y el experto. En general, según Li *et al.*

(2003), el lenguaje natural es altamente informal por naturaleza, lo que implica mayor intervención y transformación, para las tareas posteriores de análisis y diseño de un producto software.

Un lenguaje controlado (LC), según Wojcik y Hoard (1995), es un subconjunto del lenguaje natural con sintaxis, semántica o terminología restringidas. Haller y Schütz (2001) lo definen a partir de un conjunto de reglas que debe cumplir el lenguaje, así como el glosario que se debe utilizar.

Procesamiento de lenguaje natural

La transformación, cuando se habla del procesamiento del lenguaje natural, se refiere a la traducción de la versión de un texto desde una lengua natural a otra (Moreiro, 1992). Para procesar el lenguaje natural se requiere transformar el texto en una representación semántica apta para razonar, tomar decisiones y ejecutar ciertas tareas (Lourdes, 2006). Esta representación se consigue por medio del proceso de *parsing* o construcción de un árbol de análisis a partir de una gramática (Gavaldá, 2011). Si la gramática es sintáctica, por medio de un árbol de análisis se genera información sobre las categorías gramaticales de las palabras y la función sintáctica asociada (por ejemplo, la identificación del sujeto, el verbo, el predicado, los complementos, etc.). Mientras tanto, si la gramática es semántica, el árbol de análisis ya es bastante próximo a la representación lógica que permite el razonamiento y la ejecución.

Lingüística computacional

Diferentes disciplinas dentro del campo de la *lingüística* estudian el *lenguaje*. A su vez, este campo se ocupa de todos los hechos y fenómenos relacionados con el lenguaje natural. El objetivo de la *lingüística* es producir modelos que se aproximen al comportamiento humano en sus tareas básicas: leer, escribir, escuchar y hablar. Este campo, según Castro *et al.* (2009), se enfoca en el estudio de los signos lingüísticos e incluye la semántica, la sintaxis y la pragmática. Una de las disciplinas que estudia el lenguaje es la *lingüística computacional*, cuyo propósito es desarrollar una teoría computacional del lenguaje, a partir de las nociones de algoritmos y estructuras de datos de las ciencias de la computación (Araujo, 2006).

El término *lingüística computacional* (en inglés *computational linguistics*) se refiere al campo interdisciplinario entre lingüística, fonética, ciencias de la computación, ciencias cognitivas, inteligencia artificial y lógica formal (Clegg, 2008). En otras palabras, según Cunningham (2000), la *lingüística computacional* se concentra en el estudio de los lenguajes naturales, tal como lo hace la lingüística tradicional, pero usando equipos de cómputo como herramienta para modelar fragmentos de teorías lingüísticas con un interés particular.

Problema de investigación

En el marco de la ingeniería de requisitos, para iniciar el proceso de educción, se requiere descubrir y obtener el máximo de información para el conocimiento de un contexto en cuestión. El discurso contiene esta información. Una vez se consolida un discurso que describe el dominio del problema, el analista representa, mediante un modelo, el ámbito del dominio y su solución; normalmente, se utiliza un modelo conceptual. Para desarrollar un modelo conceptual, el analista o diseñador debe identificar ciertos elementos conceptuales, identificar las relaciones entre ellos y entender esta relación, para luego representar esos elementos en un lenguaje de modelado (Gangopadhyay, 2001). Este proceso se puede visualizar como una 'traducción' de un lenguaje base a otro diferente, de manera tal que el traductor reconozca y entienda símbolos expresados en un lenguaje natural de un universo de discurso, en un conjunto de símbolos definidos en un lexicón de un lenguaje de modelado (Castro *et al.*, 2009).

Es en este proceso de traducción donde el analista, luego de capturar las necesidades y expectativas del interesado, las representa en modelos técnicos. Los interesados, por su parte, validan los requisitos capturados y plasmados en dichos modelos (que suelen ser en su mayoría gráficos), aunque no los comprenden suficientemente, porque se describen en un lenguaje técnico que se aleja del natural. Esta comunicación entre los participantes (interesado-analista) se torna más débil aún por la diferencia de formación y experiencia entre ellos (Zapata & Villa, 2008), lo que genera problemas en cuanto a la correcta validación de los requisitos, el alcance de los modelos conceptuales generados y, finalmente, los altos costos de ejecución del proceso de educción.

Por otro lado, tradicionalmente, la obtención de requisitos parte de la aplicación de técnicas de captura,

como entrevistas y diseño de aplicaciones conjuntas (Christel & Kang, 1992), u otras técnicas enfocadas hacia el análisis de escenarios, como las que describen Zapata *et al.* (2007). No es muy común educir requisitos a partir de otro tipo de fuentes, como la documentación técnica, la cual incluye información en forma de manuales de procedimientos, reglamentos y estatutos de organización, etc. Esta educción permitiría principalmente: una comprensión y descripción detallada de la propia organización y del papel que representa el sistema en este contexto (Leite, 1987), la comprensión del dominio del interesado, el diseño posterior de entrevistas, la aplicación de técnicas de análisis de requisitos y la generación de modelos iniciales del dominio del problema.

A partir de las técnicas de captura de requisitos utilizadas de forma tradicional, la intervención del interesado se suele estimar incorrectamente, pues, la mayoría de las veces, se realiza en forma de diálogos y entrevistas (Leite, 1987). En este proceso se pierde tiempo, secuencia, coherencia y concisión, dado que los interesados tienden a dilatar sus intervenciones y la entrega de información, lo que, como ya se indicó, acarrea mayores tiempos en la educción y mayores costos. El compendio de información obtenida y las descripciones del dominio de aplicación, que son resultado del trabajo con el interesado, tienen los problemas propios del lenguaje natural: mucha información, uso indiscriminado de sinónimos y ambigüedades, etc.

A pesar de los acercamientos propuestos en la literatura para reducir la brecha entre los universos de discurso del interesado y del analista, todavía se exige máxima intervención del analista en el proceso de educción, en la conversión entre la descripción de requisitos y los modelos de diseño y en el método que guíe este proceso. Es necesario lograr acercamientos entre los lenguajes controlados que existen para especificar los requisitos y el lenguaje natural del interesado, el cual se puede traducir directamente a partir de documentación técnica y así conducir a las etapas posteriores del proceso de desarrollo del producto software, que se realiza actualmente de forma automática, como propone Zapata (2007).

A partir de la descripción problemática anterior, se plantea la siguiente pregunta de investigación: **¿***Cómo especificar un proceso de transformación automático de lenguaje natural a lenguaje controlado, a partir de documentación técnica, para la educción de requisitos en el diseño de un producto de software*?

Conclusiones y resultados esperados

En la literatura no se cuenta con un modelo descriptivo que represente el proceso de transformación de las necesidades y expectativas del interesado, expresadas en lenguaje natural, en requisitos expresados en un lenguaje controlado, en la fase de educción de requisitos a partir de documentación técnica. Es necesaria una formalización de dicho proceso, a partir de las teorías que ofrecen la lingüística computacional y el procesamiento de lenguaje natural, lo que podría derivar en la propuesta de nuevos conceptos o la inclusión de conceptos de otras disciplinas del procesamiento de lenguaje, en la ingeniería de requisitos.

Es necesario realizar aportes que permitan, entre otros, generar los siguientes resultados esperados:

- Facilitar la tarea de modelado del analista, a partir de información capturada de documentación técnica que se pueda representar en un lenguaje técnico o modelo técnico, aplicando un método establecido.
- Mejorar la comprensión de los interesados sobre los modelos y esquemas conceptuales que diseñan los analistas, por medio de un lenguaje cercano al natural.
- Proveer técnicas y formalismos que permitan trasladar descripciones y conocimiento de la organización, hacia los modelos cercanos al proceso de análisis de requisitos.
- Definir un marco conceptual respecto de las variables que intervienen en el proceso de transformación de lenguaje natural a un lenguaje controlado, a partir de documentación técnica, en la educción de requisitos.
- Extender, por medio de un formalismo o procedimiento, el proceso de transformación de lenguaje natural a lenguaje controlado.
- Mostrar cómo ciertas propiedades, teorías o herramientas utilizadas en la lingüística, se pueden utilizar en el marco del procesamiento del lenguaje natural, para mejorar el proceso de transformación de lenguaje a partir de documentación técnica, en la educción de requisitos.

Agradecimientos

Este trabajo se enmarca dentro de los resultados obtenidos en el proyecto de investigación 'Revisión de Literatura en Transformación de Lenguaje Natural a Lenguaje Controlado en la Educción de Requisitos', cofinanciado entre la Universidad Nacional de Colombia, Sede Medellín, y la Universidad de Medellín, Colombia.

Referencias

Bolton, D., Jones, S., Till, D., Furber, D. & S. Green (1994). Using domain knowledge in requirements capture and formal specification construction. *Requirements Engineering: Social and Technical Issues*, Academic Press, 2ª. ed., pp. 141-162.

Castro, L., Baiao, F. & Guizzardi, G. (2009). A survey on conceptual modeling from a linguistic point of view. *Relatórios t*écnicos do departamento de informática aplicada *da Unirio,* N° 0019/2009, pp. 3-12.

Clegg, A. (2008). *Computational-linguistic approaches to biological text mining*. Tesis de PhD. Londres: Escuela de Cristalografía, University of London.

Cunningham, H. (2000). *Software architecture for language engineering*. Tesis de PhD. Reino Unido: Departamento de ciencias de la computación, University of Sheffield.

Cheng, B. & Atlee, J. (2007). *Research directions in requirements engineering*. Proceedings of future of software engineering (FOSE'07), IEEE Computer Society, USA.

Christel, M. & Kang, K. (1992). *Issues in requirements elicitation*. Technical report CMU/SEI-92-TR-012 ESC-TR-92-012. USA: Software Engineering Institute.

Gangopadhyay, A. (2001). Conceptual modeling from natural language functional specifications. *Artificial Intelligence in Engineering*, Vol. 15, No. 2, pp. 207-218.

Gavaldá, M. (2011). La investigación en tecnologías de la lengua. Research in language technology. http://quark.prbb.org/19/019021.htm [Consultado el 15 de mayo de 2011].

Haller, J. & Schütz, J. (2001). *CLAT: Controlled language authoring technology.* Proceedings of the 19th annual international conference on computer documentation, Santa Fe NM.

Kof, L. (2004). *Natural language processing for requirements engineering: applicability to large requirements documents*. Alemania: Fakultät für Informatik, Technische Universität München.

Leite, J. (1987). *A survey on requirements analysis*. Advanced software engineering project technical report RTP071. EE.UU.: Department of Information and Computer Science, University of California.

Lourdes, A. (2006). Procesamiento de lenguaje natural. Disponible http://tabasco.torreingenieria.unam.mx/gch/PLN/cap1.pdf [Consultado el 10 de mayo de 2011].

Moreiro, J. (1992). *Perspectiva documental del procesamiento de lenguaje natural*. Memorias Congreso SEPLN VIII, Universidad Carlos III, Madrid.
Serrano, W. (2005). ¿Qué constituye a los lenguajes natural y matemático? *Sapiens: Revista Universitaria de Investigación*, Vol. 6 No. 001, pp. 47-59.
Tendales (s.f.). Lógica simbólica. Lógica proposicional. http://blog.educastur.es/tendales/files/2009/12/logica-teoria2.pdf
[Consultado el 25 de mayo de 2011].

Vernengo, R. (1996). El discurso del derecho y el lenguaje normativo. *Isonomía*, No. 4, pp. 87-95.

Wojcik, R. & Hoard, J. Controlled languages in industry. http://www.cslu.ogi.edu/HLTsurvey/ch7node8.html [Consultado el 22 de mayo de 2011].

Zapata, C.M. (2007). *Definición de un esquema preconceptual para la obtención automática de esquemas conceptuales de UML*. Tesis doctoral doctorado en ingeniería. Colombia: Universidad Nacional de Colombia Sede Medellín.

Zapata, C.M., Palacio, C. & Olaya, N. (2007). UNC-ANALISTA: Hacia la captura de un corpus de requisitos a partir de la aplicación del experimento Mago de Oz. *Revista EIA*, N. 7, pp. 25-40.

Zapata, C.M. & Villa, F. A. (2008). La gramática básica de UN-Lencep expresada en HPSG. *Avances en Sistemas e Informática*, Vol.5 No.1, edición especial, pp. 81-92.

A proposal to generate the method code based on class diagram and java® meta-model[1]

Una propuesta para la generación del código de los métodos desde el diagrama de clases y el metamodelo de java®

Carlos Mario Zapata Jaramillo,
Ph.D. en Ingeniería

Andrés Felipe Muñetón Lopera,
M. Sc. en Ingeniería de Sistemas

Abstract

Code generation, one of the final phases of software development lifecycle, has been made using CASE tools. However, the method code has not been successfully accomplished in the well-known CASE tools. To meet this challenge, in several proposals researchers employ UML diagrams in order to generate some parts of the method code, even though UML diagrams are given in a non-standard use. We introduce, in this paper, a proposal to generate the method code from class diagrams (complemented with pre- and post-conditions) and Java® meta-models. We exemplify the use of this proposal with a case study.

Keywords: CASE tools, class diagram, code generation, meta-modeling, pre- and post-conditions.

[1] This paper is an English version of the paper "Generación del cuerpo de los métodos a partir de la semántica de las operaciones del diagrama de clases", published on the journal *Ingeniería e Investigación*, volume 28, issue 3, 2008, pp. 58-63. The authors state that, besides translation, several amendments and minor additions have been included.

[2] The authors belong to the Computational Language Group, Universidad Nacional de Colombia. E-mail: cmzapata@unal.edu.co, afmuneto@unal.edu.co

Recibido: 11 de junio 2011
Aprobado: 26 de octubre 2011

Resumen

La generación de código, una de las fases finales del ciclo de vida del software, se viene realizando mediante herramientas CASE. Sin embargo, el código de los métodos no se realiza de manera exitosa con las herramientas CASE tradicionales. Como respuesta a este problema, existen propuestas que emplean algunos de los diagramas de UML para generar porciones del código de los métodos, pero esas propuestas emplean, con este fin, modelos o elementos que se alejan del estándar de UML. En este artículo se introduce una propuesta para generar el código de los métodos a partir del diagrama de clases (complementado con pre y post condiciones) y el metamodelo de Java®. El uso de esta propuesta se ejemplifica con un caso de estudio.

Palabras clave: diagrama de clases, generación de código, herramientas CASE, Metamodelado, pre y postcondiciones.

Introduction

Software development lifecycle has the following phases: definition, analysis, design, construction, transition, and production. During the construction phase, the models defined in previous stages are used to generate an executable code of the final application. This process is partially assisted by the well-known CASE tools. Some of these tools, like Rational Rose® (Quatrani, 2000), Together® (2011), Poseidon® (2011), and ArgoUML® (Robbins *et al.*, 1997) are capable of generating code in several languages like Java® or C++. Also, some platforms are used to perform such task; for example, NetBeans 6.8 (2011).

The resulting code obtained from the above mentioned CASE tools is still immature, and there are two reasons for this: most of the tools only generate code from class diagram—given that this diagram is structurally similar to the source code—and the method contents from the generated code classes lack the method body—these CASE tools only include the head of the code. In the case of the NetBeans 6.8 platform (2011), we can use a set of predefined "templates" included inside the "code generator" feature, but only in the development phase. We need, instead, tools to generate code from the design phase.

The software engineering community has reacted to these problems with new proposals that use other diagrams or elements in the code generation process. Three examples of these proposals are Fujaba® (Niere & Zündorf, 1999), rCos (Liu & Jifeng, 2005), and B-method (Mammar & Laleau, 2006).

Fujaba® (Niere & Zündorf, 1999) uses class diagram, and a non-standard combination of state machine and sequence diagram. Keeping in mind that the use of non-standard elements in UML standard diagrams poses a major usability problem in this CASE tool, the entire process is completely linked to customized versions of the UML standard, and changes to this standard will not be reflected upon it. Also, the device selected to represent the body of the methods is a text with the same structure of the source code to be obtained. The invested effort in the task of model building with these additions is comparable to the invested effort in manually writing code.

The rCos (Liu & Jifeng, 2005) and the B-method (Mammar & Laleau, 2006) are similar proposals in that they both use UML diagram formalizations in order to generate code. Some of the formalized diagrams are class and sequence diagrams. These proposals, as in the case of Fujaba®, pose some problems:

- The method body is represented by a formal language originated before the code generation process (with an invested time substantially longer than the time employed in the manual encoding process).
- The process itself is restricted to methods with database semantics (for example, the insertion of a record in a database).
- The use of non-standard elements makes the code generation process dependent of customized versions of the artifacts.
- There are no CASE tools that use this approach.

As a way to improve the code generation process and surpass the listed problems of previous proposals, we argue, in this paper, that the method code can be obtained from class diagram and pre- and postconditions belonging to the operations of such a diagram. We also complement this approach by adding

semantic information (represented by new elements) to the Java® meta-model expecting to match the cited post-conditions. This approach is shown by means of a case study.

The rest of the paper is structured like this: first, we discuss the state-of-the-art in automated code generation; second, we propose a method to link class operations to code methods from pre- and post-conditions; third, we propose a modification to the Java® meta-model in order to include pre- and post-condition associations; next, we present the relationship between user models and Java® development platform; then, we outline a case study illustrating this proposal; and finally, we provide conclusions and future work perspectives.

Automated code generation

A code fragment belonging to a method in the Java® programming language looks like this:

```
public void saveUser(u:User) {
Connection con = DriverManager.getCo-
nection(“”); PreparedStatement ps = con.
preparedStatement(“insert into usuarios values(?,?);”);
ps.setString(1,u.getID());
ps.setString(2,u.getName());
ps.execute();
}
```

This method is used to insert a new user into an existing table, as a way to deal with database management. Figure 1 shows a possible sequence diagram matching this method.

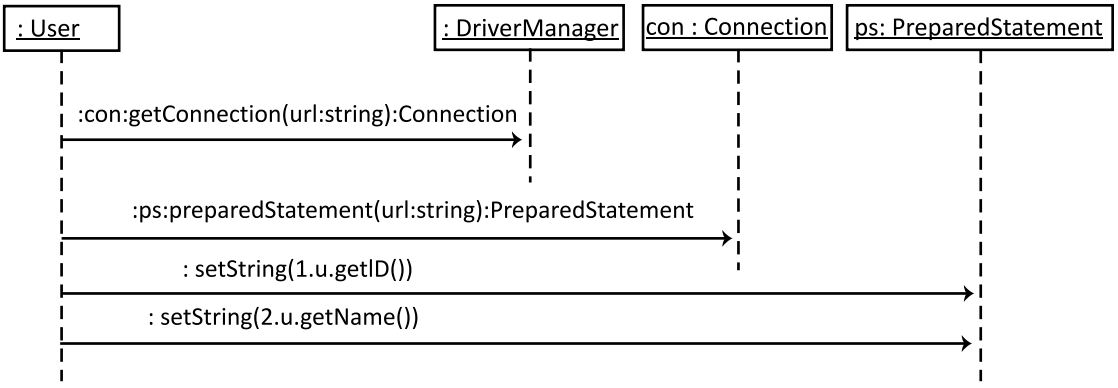


Figure 1. A possible sequence diagram for the *saveUser* method.

Most of well-known CASE tools exhibit capabilities to draw diagrams like the one in figure 1, but only Together® (2011) and Fujaba® (Niere & Zündorf, 1999) are able to generate Java® code from this diagram. The remaining tools might only generate incomplete source code from class diagram. When source code generation from sequence diagram is possible, the designer must add some special features to this code in order to complete it. In the previous example, some classes like *DriverManager*, *Connection*, and *PreparedStatement*, belonging to the Java® language, must be manually added to the resulting code. On the other hand, NetBeans 6.8 (2011) is unable to automatically generate such a code, since it does not match the predefined templates, which are: sample generator, constructor, override method, and add property.

On the other hand, Model-Driven Architecture (MDA) is an initiative of the Object Management Group (Kepple *et al.*, 2003) intended to develop standards based on the idea that modeling is a better foundation for developing and maintaining systems. MDA suggests the use of profiles to adapt the business area models (platform independent models—PIM—in the jargon of MDA) to the programming language (platform specific models—PSM—as understood by MDA practitioners). Some of the well-known CASE tools, for example Rational Rose® (Quatrani, 2000), exhibit capabilities to include

profiles in order to complement the models with information about the targeted programming language. However, even such capabilities are not enough to include the required information in the method to be generated. In the *saveUser* method, from the previous example, the information the designer must add belongs to Dynamic Link Libraries (DLL) of the Java® language, and this information is not supported by any Java® profile.

Different to the source code previously stated, other kind of source code is related to calculus; for example, the computation of the absolute value of a number or the result of an algebraic operation. Looking at the following expression of source code:

x:[x = abs y]

If we have to program a method to explain this expression, we have to know the sense of the implicit operation; for example:

```
if
y>= 0 --> x:=y
|| y < 0 --> x:= -y
endif
```

The process of recognizing the procedure to calculate the result of an operation, called refinement, is not supported by most of CASE tools. Also, other proposals as rCos (Liu & Jifeng, 2005) and B-method (Mammar & Laleau, 2006) employ a formalization of this refinement process, but this process must be explicitly defined to complete the code generation. In these proposals, formalization is an intermediate representation of the source code, and it poses some drawbacks for the entire code generation process:

- Formal languages are more difficult to manage than programming languages. The time the designer must invest in the formalization process is longer than the time devoted to manual code writing.
- Formal languages do not have standard guidelines for well-known CASE tools. Also, there are multiple notations of this kind of languages. If standardization is not possible in the code generation process, there will be a dependency on customized versions of the formal language.
- Both rCos (Liu & Jifeng, 2005) and B-method (Mammar & Laleau, 2006) are theoretical approaches, and there is no CASE tool *supporting* these theories.

Relationship among operations, from pre- and post-conditions

Every operation requires—as an executing initial condition—a set of states in the particular system it belongs to. After the execution of the operation, the system can remain the same or can become a new set of states. The initial set of initial conditions is named pre-conditions and the final set of states is named post-conditions. The *saveUser* operation, previously defined, has the following discussed conditions:

```
saveUser (u:User){
pre: not exist u:User
post:u=User +{u}
}
```

User is a set of users that has been stored in the database.

According to Morgan (1998), the specs are based on pre- and post-conditions. Those specs can be refined up to machine-understandable code, that is, a text that can be compiled and executed. When applying this concept to the *saveUser* operation, we must write down the following sequence of Java® commands to reach the post-condition (in this case, to store a new user in the database): (1) to create a connection to the database, (2) to prepare the query, and (3) to execute the query. To accomplish the initiation of the execution of the query (3rd step), the system needs to reach the states demanded by the post-condition in the second step. Similarly, the preparation of the query (2nd step) can only be initiated when the post-condition in the first step is reached.

Due to the fact that we need to express pre- and post-conditions in the Java® language, we propose an instance of the Java® meta-model, in order to describe classes belonging to the *java.sql* package. Figure 2 shows the part of this instance, including the classes used by the *saveUser* operation.

We must highlight the fact that one parameter, in figure 2, acts as a relationship connector between one method (belonging to a class) and another class. In the same line of reasoning, one role acts as a navigation connector; for example, by means of *returnParameter* the system is capable of executing a new method. If we assume that *returnParameter* is

the method post-condition, then all the methods represented in Figure 2 are linked by their post-conditions. Figure 3 shows the pre- and post-condition specs to the *saveUser* operation. In this Figure, pre- and post-conditions participate in the relationships among methods.

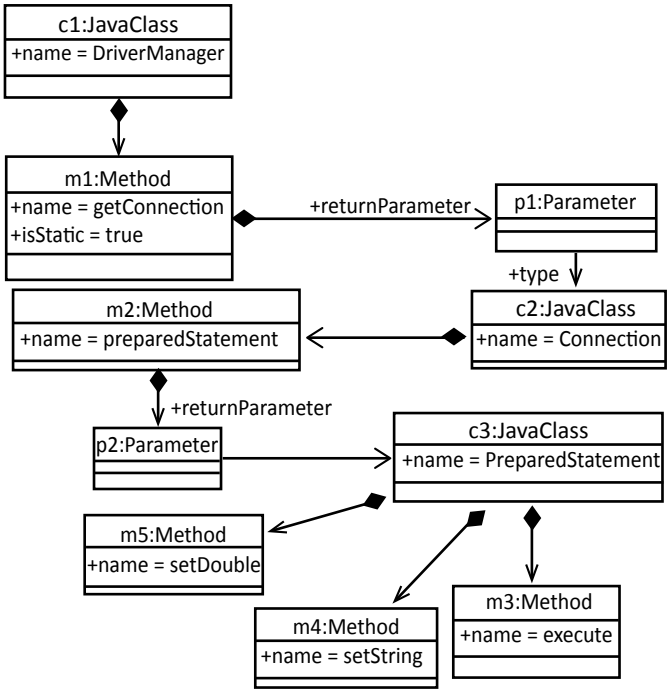


Figure 2. Part of the Java® meta-model instance used by the *saveUser* operation.

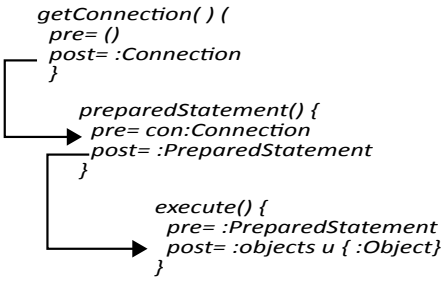


Figure 3. Pre- and post-condition specs for the *saveUser* method.

Figure 3 is the base to generate the following Java® code:

```
Connection con = DriverManager.getConnection();
PreparedStatement ps = con.preparedStatement("");
ps.execute();
```

The general process followed in order to generate this code was:

- To identify the *execute* operation as the one for saving elements in the database.
- To outline the fact that the *execute* operation is non-static, and, consequently, it has (as a pre-condition) an object creation of the class *PreparedStatement*. In addition, we must outline that *preparedStatement* operation has, as a post-condition, the creation of a *PreparedStatement* object.

- To note that *preparedStatement* operation is non-static, and it has, as a pre-condition, an object creation of the class *Connection*. Again, we have to note that the *getConnection* operation has, as a post-condition, the creation of a *Connection* object.
- Finally, to evidence that *getConnection* operation is static, so it does not require any pre-condition.

If we compare the suggested code with the one defined earlier, we discover some missing facts. For example, the *setString* operation belonging to the *PreparedStatement* class is not generated by the described process.

To summarize, we can establish that an O' operation subset can be created from an O operation set—and the O set has a post-condition named P. All the operations belonging to O' subset are linked by means of pre- and post-conditions, in such a way that is possible to reach a P post-condition. However, as we already present, the O' subset does not include all the needed operations to reach the post-condition P. As a consequence, the number of relationships among Java® operations might not be enough to generate the entire code, and we need to supply additional information to the process.

Adding pre- and post-condition associations to the Java® meta-model

The process of code generation, when manually made, is full of previous knowledge assertions—on the part of the designer—about the encoding platform. These assertions are commonly found by the designer in the platform documentation, using the revision of the examples included on this documentation.

The presence of these assertions makes difficult to automate the encoding process. In other words, if we want to automatically generate code from models, we will perhaps need to increase the contents of the meta-model with some of the designer's previous knowledge. For example, in the *saveUser* method that we describe, the designer knows that the use of the *execute* method, from the *PreparedStatement* class, needs as a pre-condition the results of a setX() operation, where X can be *String*, *Double*, and *Int*, among others.

Figure 4 shows the modifications proposed for the Java® meta-model. In order to make explicit all the possible relationships among operations, we add pre- and post-condition associations to the *Method* class and the *PreconditionGroup* class. We also add the following OCL restriction, as a way to avoid the fact of a method being itself pre- or post-condition.

self.precondition -> forAll(m | m <> self) and self.postcondition -> forAll(m | m <> self)

Pre- and post-condition associations are used by the designer to store conditions that are not explicitly defined in the Java® platform structure. In addition, *PreconditionGroup* class is used to store groups of pre-conditions needed to initiate a method. Figure 5 shows a portion of the refined Java® meta-model instance (the entire meta-model is illustrated in Figure 2). In Figure 5, we see the *PreconditionGroup* package belonging to *m3* method of *c3* JavaClass. In this case, *m3* method can be initiated by the result of either *m5* method or *m4* method.

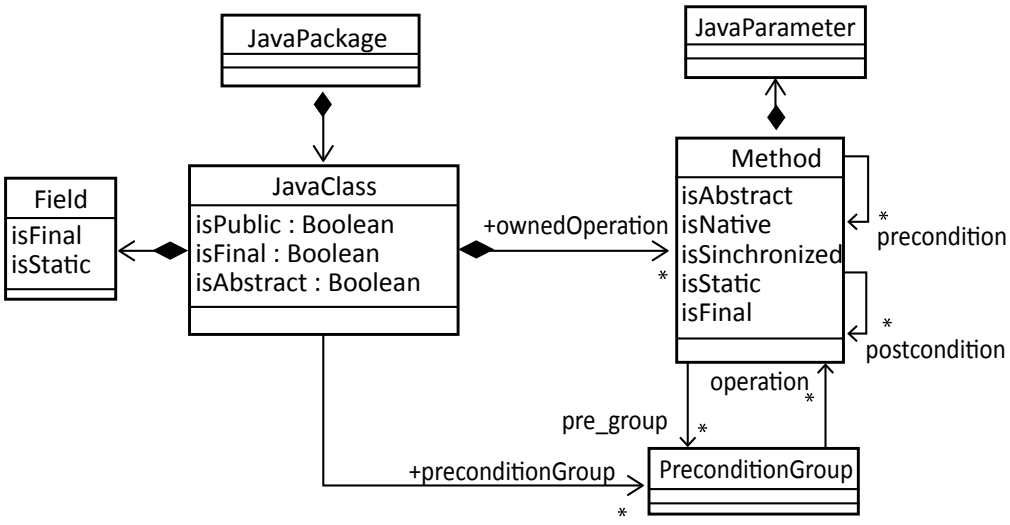


Figure 4. Modifications proposed to the Java® meta-model.

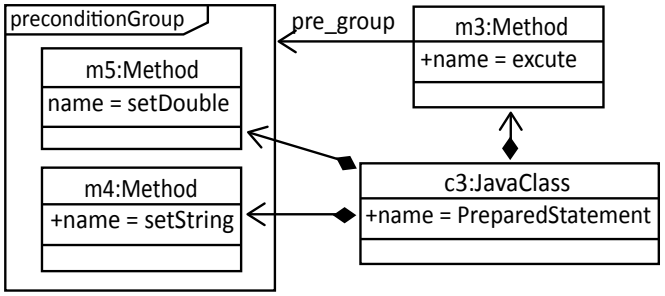


Figure 5. A portion of the Java® meta-model instance of Figure 2.

Relationship between user models and the Java® meta-model platform

As a final step in the code generation process, we propose to establish a relationship between the user model and the Java® meta-model instance. In this stage, a profile is provided in order to associate the user specs to the several kinds of database management specs. This profile can be generalized to meet another kind of specs, and figure 6 shows how it looks. In this figure, a Specification has pre- and post-conditions, represented by means of the attributes *pre* and *post*.

The *DBSpecification* class is the generalization of all possible user specs, in terms of database management, and includes the database (represented by the *DB* attribute) and the Power Set (represented by the *P(X)* attribute). This class can be specialized into *DDLSpecification* class (to modify the database) and *DCLSpecification* class (to query the database). Finally, *DDLSpecification* class can be specialized into three classes: *UpdateSpecification*, *InsertSpecification*, and *DeleteSpecification*.

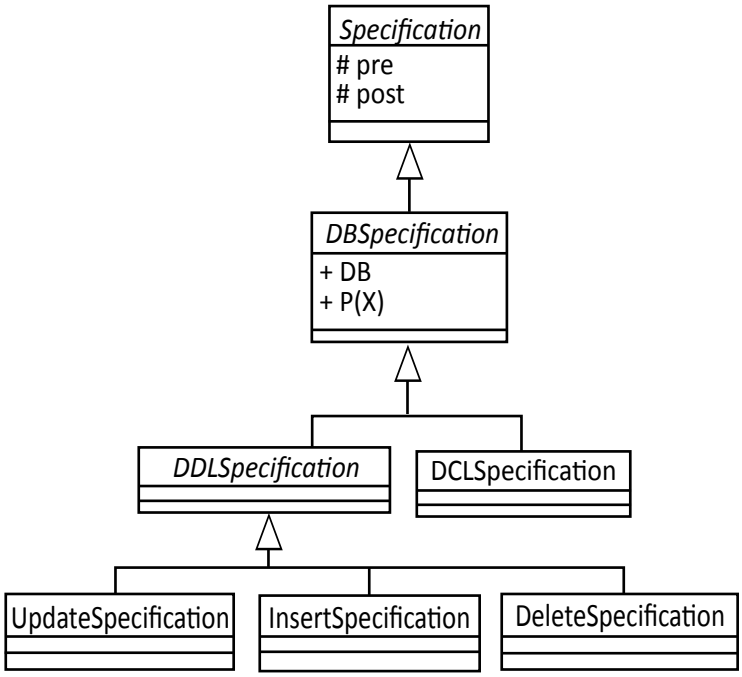


Figure 6. A *profile* to associate user specs to database management specs.

The aforementioned profile can be handled using UML stereotypes, a special feature of UML. Figure 7 shows the *ManageUser* class, belonging to the user model, which includes the *saveUser* operation. This figure also shows the *java.sql.PreparedStatement* class, which includes the *execute* method.

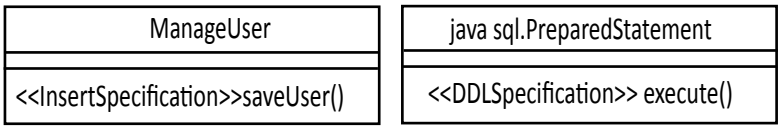


Figure 7. UML stereotypes of the operations belonging to two classes.

Figure 8 shows the relationship between the user model and the Java® meta-model instance. Due to the fact that *InsertSpecification* class is invoked by the *o1* operation, and simultaneously belongs to the specialization of the *DDLSpecification* class, we have to select this class to represent the semantics of the *saveUser* operation.

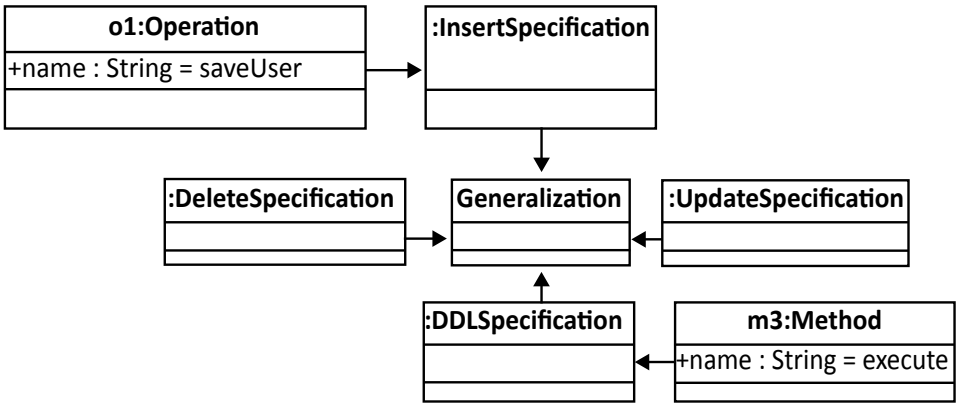


Figure 8. A meta-model instance to explain the relationship between user model and Java® meta-model instance.

A case study

Figure 9 shows two classes of a user management class diagram. In this figure, *User* and *UserManager* classes have their components stereotyped, according to the profile provided before. *UserManager* class has two operations: *saveUser*, the previously discussed stereotyped operation, and *getUser*, which has a stereotype intended to retrieve a user from database by means of his/her user identification. *User* class has only one stereotype belonging to the name of the class (<<Entity>> in figure 9.)

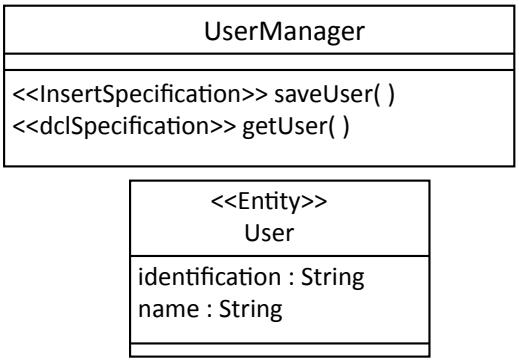


Figure 9. Two classes of the user management class diagram.

Following is the step-by-step encoding process of the *getUser* operation. The code is completed as its components are being found. Figure 10 shows an instance of the Java® meta-model with the pre- and post-conditions of the *executeQuery* method. The *DCLSpecification* of this method is also included.

As we can see in figure 10, the *executeQuery* method is associated with *DCLSpecification* object, which has the same semantics that the *getUser* operation of the *UserManager* class. Furthermore, *executeQuery* method has a *ResultSet* object as a post-condition and a *PreparedStatement* object as a pre-condition (because is non-static). Consequently, we obtain the following resulting code:

ResultSet resultSet = preparedStatement.executeQuery();

Objects of the *PreparedStatement* class are created by means of the *preparedStatement* method of the *Connection* class, as discussed earlier. This method requi-

res (because it is non-static) the creation of an object belonging to the *Connection* class. At this stage, the code of the *getUser* method is:

PreparedStatement preparedStatement = connection.preparedStatement();

ResultSet resultSet = preparedStatement.executeQuery();

getConnection method is non-static (consequently, it has no pre-conditions) and belongs to the *DriverManager* class. This method also returns a *Connection* type object. After this analysis, the code of the *getUser* method is:

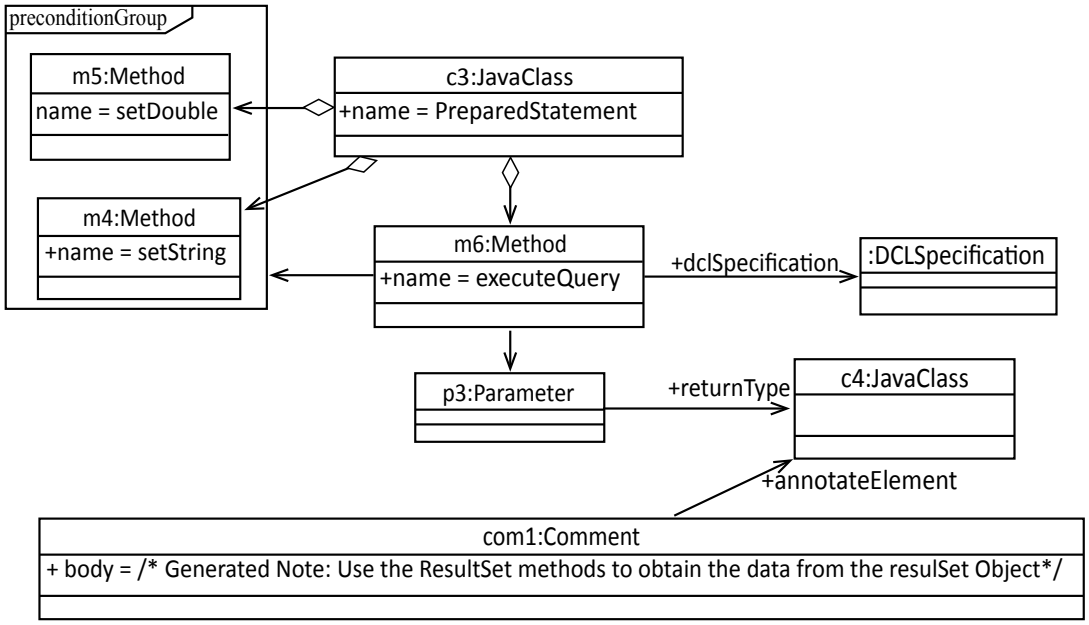


Figure 10. Java® meta-model instance of the *executeQuery* method.

```
public User getUser(User u){
Connection connection = DriverManager.getConnection();
PreparedStatement preparedStatement = connection.preparedStatement();
ResultSet resultSet = preparedStatement.executeQuery();
}
```

This is the final result of our proposal, but it is incomplete. A message is added to the code in order to warn the designer about the need for complement the method code. The source of this message is the object *com1* in Figure 10. The final code of the *getUser* method is:

```
public User getUser(User u){
Connection connection = DriverManager.getConnection();
PreparedStatement preparedStatement = connection.preparedStatement();
ResultSet resultSet = preparedStatement.executeQuery();
/* Generated Note:
* Use the ResultSet methods to obtain the data from the resultSet object
*/
}
```

We can expect—from the aforementioned code—that the *getUser* method can get a record from the database, but it is not stated that this record is a *User* object. The expected code of the *getUser* method will be:

```
Connection connection = DriverManager.getConnection();
PreparedStatement preparedStatement = connection.preparedStatement();
ResultSet resultSet = preparedStatement.executeQuery();

Usuario u = new Usuario();
```

```
    if(next()){
    u.setIdentificacion(resultSet.
getString("identificacion"));
    u.setNombre(resultSet.getString("nombre"));
    }
    return u;
```

As we previously mentioned, the designer must complete the code to have it ready to run. Before, we explored the connection between the *resultSet* method and the *setX* method, and we believe that this gap can be bridged by means of a careful analysis of the API Java® documentation. The automation of the rest of the proposal still needs some additional analysis.

Conclusions and future work

We discussed an approach to automate the code generation process by means of pre- and post-conditions of the user's model and the addition of some elements to the Java® meta-model (specifically, pre- and post-condition associations and the *PreconditionGroup* class). We also used UML stereotypes to define the semantics of the user class method. The mentioned approach has shown that it can be useful to generate the body of the methods, one of the never-accomplished promises of CASE tools.

There is some work to be done, as a way to improve the approach discussed:

- To implement a method to select the right semantics of the method, in situations where there is a "many-to-many" relationship between the stereotype of the user model and the stereotype suggested by the Java® meta-model instance.
- To extend the suggested approach to non-database management operations.
- To generalize this approach to other programming platforms.
- To incorporate this approach into a CASE tool.

References

Kepple, A., Warmer, J. & Bast, W. (2003). MDA Explained, *The Model Driven Architecture: Practice and Promise*. Indianapolis: Addison-Wesley.

Liu, Z. & Jifeng, H. (2005). Towards a Rigorous Approach to UML-Based Development. *Electronic Notes in Theoretical Computer Science*, Vol. 130, pp. 57–77.

Mammar, A. & Laleau, R. (2006). From a B formal specification to an executable code: applicational to the relational database domain. *Information and Software Technology*, Vol. 48, No. 4, pp. 253-279.

Morgan, C. (1998). *Programming from Specifications,* 2nd Edition. Hampstead: Prentice Hall International.

NetBeans Platform. "Code Generator Integration Tutorial". http://platform.netbeans.org/tutorials/nbm-code-generator.html [Consulted July 11th, 2011].

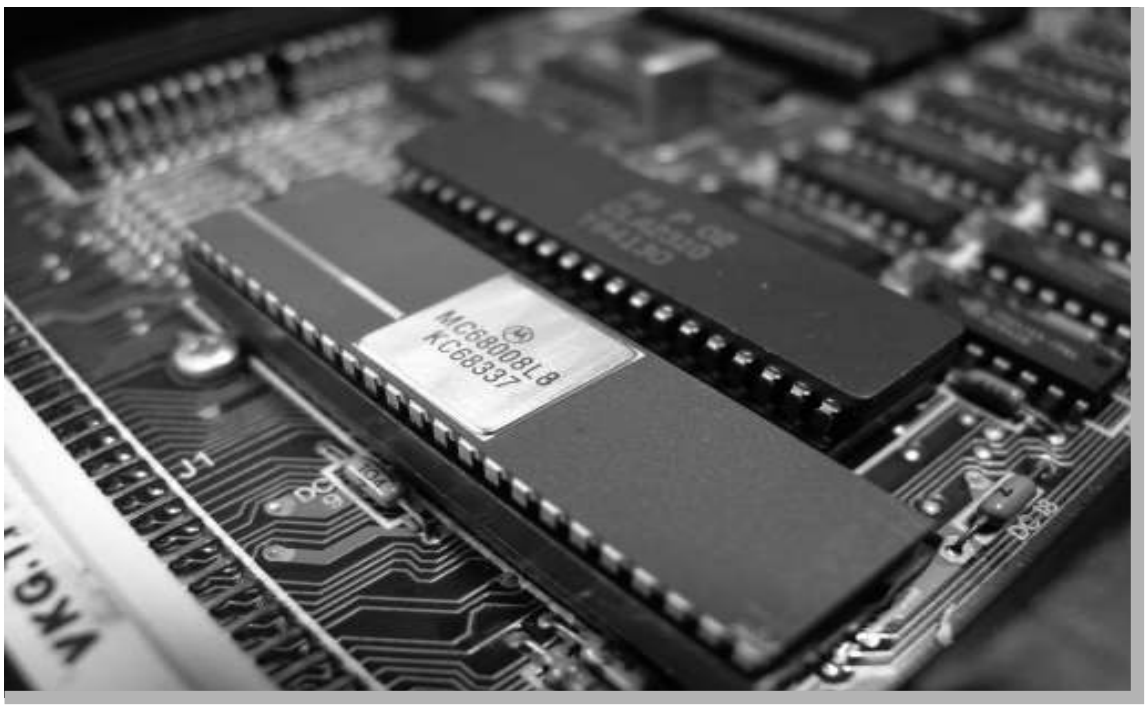
Niere, J. & Zündorf, A. (1999). Using Fujaba for the Development of Production Control Systems. *Lecture Notes in Computer Science*, Vol. 1779, pp. 181–191.

Poseidon®. http://www.gentleware.com [Consulted July 11th, 2011].

Quatrani, T. (2000). *Visual Modeling with Rational Rose 2000 and UML*. Reading: Addison-Wesley.

Robbins, J. E., Hilbert, D. M., & Redmiles, D. F. (1997). *Argo: A Design Environment for Evolving Software Architectures*. Proceedings of the 19th International Conference on Software Engineering (ICSE'97), Boston, USA, pp. 600–601.

Together, Borland Software Corporation. "Borland Together Architect". http://www.borland.com/us/products/together/index.html [Consulted July 11th, 2011].


MC68008L8
KC68337

La Ingeniería de Software en el desarrollo de aplicaciones para dispositivos móviles

Software engineering in the development of applications for mobile devices

Fabio Alberto Vargas Agudelo
Magíster en ingeniería de sistemas,
e-mail: fvargas@tdea.edu.co,
Tecnológico de Antioquia

Resumen
El artículo presenta la aplicación del ciclo de vida del software en el desarrollo de aplicaciones para dispositivos móviles, teniendo muy en cuenta las diferencias existentes cuando el desarrollo se hace pensando en aplicaciones cliente-servidor o inclusive en aplicaciones web. Se describe en cada etapa del ciclo de vida el proceso que se debe seguir a partir de la concesión de una construcción móvil.
Palabras claves: dispositivos móviles, ciclo de vida del software, desarrollo de software.

Abstract
This paper shows the application of software lifecycle in the development of applications for mobile devices, considering the differences when development is intended for client-server and/or web applications. The process to be followed, after a mobile construction being granted, is described for each stage of the lifecycle.
Keywords: Mobile devices, software life cycle, software development.

Recibido: 11 de junio 2011
Aprobado: 26 de octubre 2011

Introducción

El desarrollo de software requiere de un proceso planeado y estandarizado si se quiere generar productos de alta calidad, tanto en su documentación como en su aplicativo final. Para ello existen varios métodos de ingeniería de software que establecen un conjunto de entregables con el ánimo de dar una trazabilidad al producto, y asegurar que pueda ser interpretado, actualizado y adaptado fácilmente por los usuarios finales. Los métodos de ingeniería de software RUP (Kruchten, 1999), CDM (Oracle 2000), FDD (Coad y Lefebvre, 1999), XP (Beck, 2000) y UN-Método (Zapata *et al*., 2006) fueron concebidos para el desarrollo de aplicaciones cliente-servidor, sin tener muy en cuenta el avance de los diferentes ambientes de desarrollo móvil y web, los cuales requieren un sinnúmero de particularidades que hacen insuficientes las etapas de ciclo del vida del software para responder a las necesidades de los interesados (Zapata y Vargas, 2009). El artículo aborda el ciclo de vida del desarrollo de software sugerido en los métodos de desarrollo de software mencionados anteriormente y plantea un conjunto aspectos que deben tenerse en cuenta en cada fase (definición de requisitos, análisis, diseño, desarrollo, pruebas e implementación) para crear un aplicativo móvil de alta calidad.

Este artículo tiene la siguiente estructura: en la sección 2 se presenta una revisión de los trabajos referentes a la utilización de la ingeniería de software en aplicaciones móviles; posteriormente, en la sección 3 se realiza una descripción del proceso de cada fase de la ingeniería de software para una aplicación móvil y en la sección 4 se presentan las conclusiones.

Antecedentes

La ingeniería de software se ha aplicado al desarrollo de software desde hace treinta años aproximadamente, lo cual ha contribuido directamente a mejorar la calidad y a garantizar el cumplimiento de las necesidades de los usuarios o interesados en el producto (Lamsweerde, 2000).

Recientemente, los dispositivos móviles —como teléfonos, asistentes personales, *Ipod, smartphones,* computadores portátiles, relojes y otros— han cobrado gran popularidad en diferentes ámbitos sociales, económicos y políticos. Estos dispositivos han permitido construir aplicaciones de gran alcance, adaptadas a cambios de contexto, como las variaciones en el ancho de banda, batería, conectividad, accesibilidad de los servicios, diseño de interfaces, capacidad de memoria, tamaño, etc.

Capra *et al*. (2003) describe un *middleware* de computación móvil que explota el principio de reflexión para mejorar la adaptación y sensibilidad al contexto en la construcción de aplicaciones móviles. Sin embargo, Mascolo *et al.* (2002) plantea que la aplicación de la ingeniería de software para dispositivos móviles es tediosa y propensa a errores, debido a que se debe tener en cuenta muchos aspectos en la definición de requisitos no funcionales, tales como la heterogeneidad y tolerancia a fallos, así como aspectos de seguridad y de rendimiento.

Los desarrollos alrededor de la computación ubicua requieren el despliegue de aplicaciones soportadas en la ingeniería de software que puedan sobrevivir el uso diario, se acomoden a las necesidades del entorno, cumplan realmente funciones productivas sin convertirse simplemente en una moda (Abowd & Mynatt, 2000). El reto para la ingeniería de software es diseñar notaciones, técnicas, artefactos, métodos y herramientas para construir sistemas para dispositivos móviles (Emmerich, 2000).

Ciclo de vida del software para dispositivos móviles

Cuando se piensa desarrollar una aplicación para un dispositivo móvil en cualquiera de las plataformas y para cualquier entorno es de vital importancia reconocer y establecer condiciones que garanticen la pertinencia, la calidad, la seguridad, la eficiencia y el rendimiento del programa que se desea construir y utilizar por medio del dispositivo móvil (Fling, 2009). Por tal razón es de suma importancia seguir en forma clara las etapas generales del ciclo de vida del software (definición y análisis de requisitos, diseño, desarrollo, pruebas, y mantenimiento), pero teniendo muy presentes las grandes diferencias que existen entre el desarrollo de una aplicación para ejecutar en un PC de escritorio y el de una aplicación para ejecutar en un dispositivo móvil.

Cada etapa debe ir enfocada a establecer muy claramente qué es lo que se busca en una pieza de

software para un dispositivo móvil, que garantice la movilidad, el fácil uso y el aprovechamiento de los limitados recursos de memoria.

Definición de requisitos

La primera fase para desarrollar una aplicación móvil es establecer una buena especificación de requisitos, ya que esta es la base para el buen rendimiento, la eficiencia y la calidad de la pieza de software por desarrollar. La claridad sobre todos los requisitos de los usuarios y el problema que se desea abordar permite tomar decisiones fundamentales acerca del tipo de aplicación requerida, así como también sobre el tipo de dispositivo por utilizar, pues hay claras diferencias entre una aplicación web móvil y una residente en el dispositivo. Además, se determina si la aplicación justifica un desarrollo móvil o si es recomendable una aplicación de servidor que se ejecute en un computador de escritorio normal. Así mismo, es muy importante definir los requisitos no funcionales con respecto a la seguridad, la funcionalidad, la capacidad y la cobertura.

Lo anterior permitirá establecer y definir en un documento inicial la viabilidad de la aplicación móvil que se va a desarrollar, tras determinar su inicio o en caso de cambios de planes, y a partir de allí descartar una aplicación móvil como solución para la organización.

Análisis de requisitos

Una vez determinada la viabilidad de la aplicación móvil y definidos los requisitos del futuro sistema, se deben identificar y representar las entradas, los procesos, las salidas y los usuarios que intervienen directamente en la aplicación móvil, así como las estructuras de datos requeridas. La representación de estos aspectos se debe realizar mediante diagramas de UML (Larman, 2002), utilizando el diagrama de procesos (Villanueva *et al.*, 2005), el diagrama de casos de uso (Larman, 2002) y el diagrama de clases (Larman, 2002).

Diseño

En la fase de diseño, también conocida como de desarrollo de prototipos, se comienza a reflejar de forma muy clara los aspectos visuales y técnicos que dan vida a la aplicación móvil. Durante esta etapa, se genera una retroalimentación por parte del usuario, quien se encarga de hacer cumplir los requisitos de funcionalidad y diseño esperados —es muy importante definir aspectos funcionales con el cliente en cuanto a la navegabilidad, desplazamientos por el dispositivo y, lo más importante, la forma de carga, almacenamiento y presentación de los datos solicitados y necesarios para el proceso requerido del aplicativo móvil—.

Para representar las interfaces de usuario propuestas en el diseño se debe utilizar emuladores de dispositivos móviles presentes en un conjunto de herramientas libres, los cuales permitan ir realizando modificaciones solicitadas y sugeridas en el diseño de la interfaz, pero también se debe ir probando en dispositivos reales, de tal manera que se garantice un mayor grado de adaptabilidad a las diferentes tecnologías y dispositivos existentes.

Desarrollo

Una vez el usuario defina los requisitos y se establezcan las condiciones de diseño aprobadas en las fases anteriores se pasa a la etapa en la cual se escribe el código fuente necesario para dar funcionamiento a la pieza de software móvil. Para ello se debe tener conocimiento de un lenguaje de programación móvil adaptable a las necesidades de diseño previamente concretadas con los usuarios directos de la aplicación.

Pruebas

Una vez culminado el desarrollo de la pieza de software definida y establecida con el usuario es de vital importancia verificar y evaluar el funcionamiento y la calidad de la pieza de software resultante con el fin de evitar sobrecostos por ajustes posteriores al software. Además, debe verificarse que la pieza de software cumpla con los requisitos definidos en etapas anteriores y genere los resultados esperados por los usuarios. Para lograr este objetivo es necesario ejecutar la aplicación en el dispositivo móvil seleccionado. Igualmente, se debe realizar un conjunto de pruebas al software móvil desarrollado, que vayan desde las pruebas unitarias hasta las pruebas funcionales con el fin de garantizar la satisfacción total del usuario.

Es muy importante aclarar que un programador jamás debería entregar una pieza de software móvil sin haberlo probado, y a su vez, un usuario no debería recibir la aplicación si haber verificado dicha prueba. Para realizar la entrega de la pieza de software se deben llevar a cabo las siguientes pruebas:

Pruebas unitarias: cuando se construye un aplicativo móvil, es de vital importancia comprobar que el código fuente escrito funcione correctamente y se adapte al funcionamiento permanente del programa.

Pruebas de integración: se debe probar no solo el funcionamiento individual de la pieza de software móvil, sino también su integración con los demás componentes de la aplicación, especialmente con el servidor de datos con el cual se realiza el proceso de sincronización de la información utilizada en el dispositivo.

Pruebas de validación: este proceso tiene como objetivo determinar si la aplicación móvil cumple los objetivos para los que se construyó el producto.

Estas pruebas sirven para determinar que se hayan logrado todas las metas definidas en las fases anteriores, muy especialmente los requisitos funcionales especificados por el usuario, así como también en las características de diseño establecidas.

Prueba de recuperación: aquí se examina la capacidad del aplicativo móvil de recuperar el sistema luego de bloqueos, fallas en el dispositivo y descargas.

Prueba de seguridad: evalúa la capacidad de la aplicación móvil para protegerse contra accesos no autorizados y de limitar sus operaciones generales a un conjunto de usuarios determinados en las fases anteriores.

Prueba de compatibilidad: aquí se revisa el desempeño de la pieza de software móvil entre diferentes dispositivos que manejen diferente tecnología y apariencia a fin de garantizar su diseño y rendimiento. Es muy importante que un aplicativo móvil se acomode a una gran cantidad de dispositivos, formas de navegación y tecnologías existentes.

Instalación y mantenimiento

Luego de cumplir con todos las fases anteriores, es decir, de haber realizado una buena definición de requisitos, un análisis, un diseño acorde a las necesidades y de codificar y probar la aplicación —por ende garantizando la satisfacción de las necesidades de los usuarios del aplicativo móvil—, se procede a la instalación del software en el dispositivo (en caso de ser necesario) y a la puesta en marcha del mismo, así como a la capacitación de los usuarios.

Conclusiones

La computación móvil permite, independientemente del lugar o tiempo, realizar diferentes actividades de procesamiento en los dispositivos móviles. Se realizan desde aplicaciones de entretenimiento, mensajería, comercio electrónico móvil (*m-commerce*), entre otras que apenas comienzan a desplegarse en nuestro país, como servicios de multimedia, de vigilancia, y colaboración en servicios médicos.

La construcción de aplicaciones de software para dispositivos móviles de alta calidad demanda de los desarrolladores el uso de métodos de ingeniería de software para lograr la satisfacción total de los requisitos de los usuarios y permitir a la vez una fácil actualización y mantenimiento del software por un largo periodo de tiempo.

El software móvil es un campo de acción que aún no ha alcanzado el grado de uso que esto genera, debido a que todavía las personas y las organizaciones desconfían de la seguridad de estos recursos para administrar información personal y, especialmente, organizacional.

Es muy importante crear métodos de ingeniería de software adaptables al desarrollo móvil, que tengan en cuenta sus particularidades y permitan al desarrollador el crear productos de alta calidad.

Referencias

Abowd, G. y Mynatt, E. (2000). "Charting past, present, and future research in ubiquitous computing". *ACM Trans. Comput.-Hum. Interact.* 7, 1, pp. 29-58.
Beck, K. (2000). *Extreme Programming Explained: Embrace Change.* Boston, Addison-Wesley.

Capra, L., Emmerich, W., Mascolo, C. (2003). "CARISMA: Context-Aware Reflective mIddleware System for Mobile Applications". *IEEE Transactions on Software Engineering,* vol. 29, no. 10, pp. 929-945.

Coad, P. y Lefebvre, E. (1999). *Java Modeling in Color with UML.* Nueva York, Prentice Hall.

Emmerich, W. (2000). Software engineering for middleware: a roadmap. En A. Finkelstein, editor, *The Future of Software Engineering*. Edición especial publicada en asociación con ICSE 2000.

Fling B. (2009). *Mobile Design and Development.* O'Reilly, ISBN: 978-0-596-15544-5.

Kruchten, Ph. (1999). *Rational Unified Process —An Introduction.* Boston, Addison-Wesley.

Larman, C. (2002). *UML y patrones,* 2ª ed. Madrid, Prentice-Hall.

Lamsweerde A. (2000). Requirements engineering in the year 2000: a research perspective. Proceeding of the 22nd International Conference on Software Engineering, Limerick (Irlanda), 5-19.

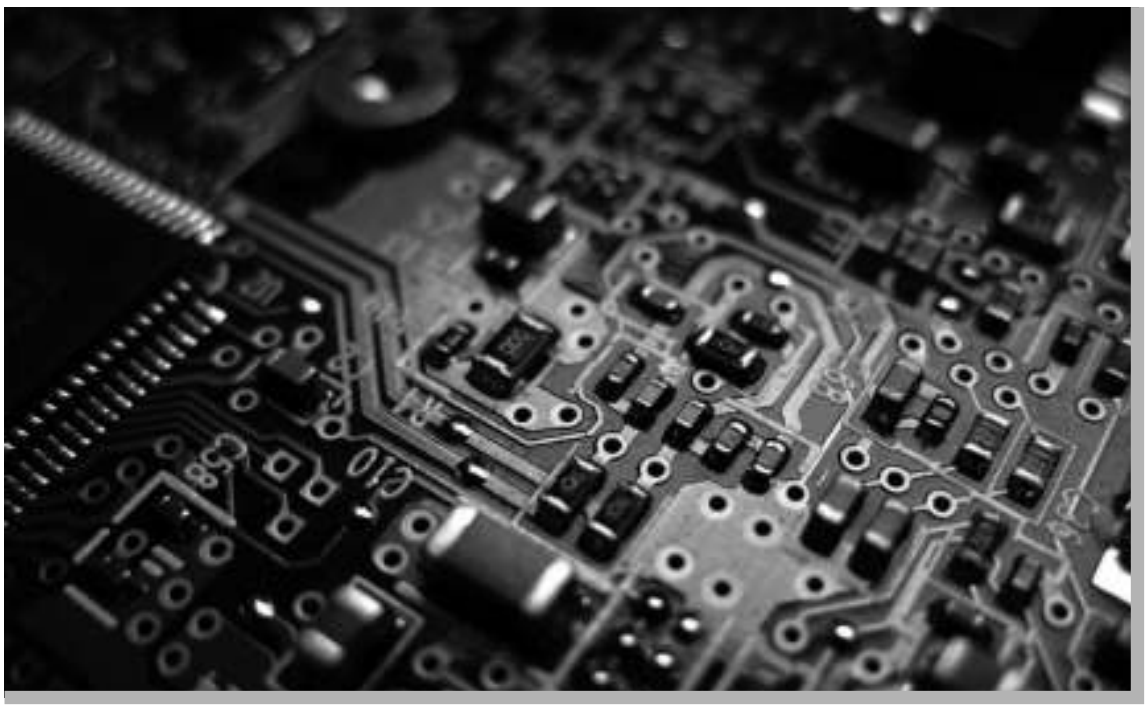
Mascolo, C., Capra, L. y Emmerich, W. (2002). Middleware for Mobile Computing (A Survey). En: E. Gregori, G. Anastasi y S. Basagni, editores, *Networking 2002 Tutorial Papers,* LNCS 2497, pp. 20-58.

Oracle® Corporation (2000). *Oracle MethodSM CDM Quick Tour.* Redwood City, Oracle Corporation.

Villanueva, I., Sánchez, J., Pastor, O. (2005). "Elicitación de requisitos en sistemas de gestión orientados a procesos", *VIII Seminario sobre Ingeniería de Requerimientos (WER'05),* Porto, Portugal.

Zapata, C., Vargas, F. (2009). Una revisión de la literatura en consistencia entre problemas y objetivos en ingeniería de software y gerencia organizacional. *Revista EIA,* Número 11, pp. 117-129.

Zapata, C., Villegas, S. y Arango F. (2006). Reglas de consistencia entre modelos de requisitos de UN-Método. *Revista Universidad Eafit,* 42(141):40-59.



Integración de TI y TIC para la construcción de un sistema que optimice el registro y control de parqueo en Colombia

Integrating IT and ICT to build a system optimizing parking registration and control in Colombia

Juan Camilo Giraldo Mejía,
Ms.C. en Ingeniería de Sistemas, ingenierojcgm@gmail.com

Paola Andrea Noreña Cardona,
Candidata a Magíster en Ingeniería de Sistemas, Universidad Nacional de Colombia. panc524@gmail.com.

El mundo *está experimentando una revolución tecnológica de primer orden, centrada en torno a las tecnologías de la información y la comunicación (TIC) y la ingeniería genética. Internet es, a la vez, el epítome y el medio más poderoso de esta revolución.*

Castells (2002).

Resumen

Las TI y las TIC quieren aplicarse para crear una solución que reduzca las falencias en el sistema de aparcamiento de nuestro país, cada vez que se requiere el servicio. TI es un término conveniente para incluir las tecnologías telefónicas y la computacional. Ésta es la tecnología que ha conducido a lo que comúnmente se ha llamado "la revolución informática ".

Estas tecnologías apoyarán la funcionalidad de un sistema de información que permita registrar y controlar la información de vehículos en el proceso de aparcamiento. Se definen algunos conceptos sobre hardware y software necesarios para concretar la propuesta de integración de tecnologías.

Palabras clave: TI, TIC, sistemas de información, parqueadero.

Recibido: 11 de junio 2011
Aprobado: 26 de octubre 2011

Abstract
IT and ICT technologies are to be used to create a solution reducing drawbacks in the Colombian parking system, whenever service is required. IT is a convenient term to include the telephone and computer technology. This is the technology that has led to what is commonly called "the information revolution." These technologies support the functionality of an information system that records and tracks information about vehicles in the parking process. Several concepts on hardware and software are defined, since they are instrumental to materialize a proposal for technology integration.
Keywords: IT, ICT, information systems, parking lot.

Introducción

El artículo describe en forma general un sistema que integra tecnología computacional de hardware y software (TI) y tecnologías de la información y la comunicación (TIC), con el objetivo de introducir disciplina y organización en estacionamientos o zonas de parqueo en diversos establecimientos, como almacenes de cadena, centros comerciales, hoteles, discotecas, empresas, aeropuertos, parques de recreación y deporte, clínicas e instituciones educativas, entre otras, tanto en la región como en el país. Igualmente, se definen algunos conceptos sobre hardware y software necesarios para concretar la propuesta de integración de tecnologías.

TI es un término que abarca todas las formas de tecnología usadas para crear, almacenar, intercambiar y usar información de diferentes maneras (escritura, dibujos, datos, conversaciones de voz, imágenes, video, presentaciones multimedia, y muchas otras, incluyendo las que todavía no han sido concebidas). El término TI es conveniente para abarcar las tecnologías telefónica y computacional. Ésta es la tecnología que ha conducido lo que comúnmente se le ha llamado "la revolución informática.

Buscando soluciones con TI y TIC

Quiere aplicarse las TI y TIC para crear una solución que reduzca las falencias en el sistema de aparcamiento de nuestro país, cada vez que se requiere el servicio. Estas falencias afectan al usuario, lo cual se manifiesta en la dificultad de localización de estacionamientos disponibles. Debería indicársele al usuario cuando alguno de los sitios frecuentados se encuentra copado. Sin embargo, lo que sucede actualmente es que la persona interesada busca directamente dónde estacionarse, y muchas veces, al no lograr su objetivo, debe retirarse. Esto genera pérdida de tiempo, pues la persona invierte tiempo para hallar un aparcamiento y después de un largo recorrido, se encuentra con la noticia de que no hay disponibilidad en un parqueadero al tope.

Las personas encargadas no tienen control permanente del flujo vehicular: en diferentes días y horas se presenta movilidad mayor o menor de acuerdo a las actividades establecidas en el lugar. En consecuencia, se pierde en algunos de estos casos el control del flujo vehicular y de la cantidad de personas que están utilizando el servicio.

Así mismo, se observa que los conductores en el país parquean indiscriminadamente: los colombianos están acostumbrados a estacionarse en cualquier lugar sin atender las normas de tránsito o de circulación peatonal o vehicular, ya que no hay un control administrativo claro respecto a la posición donde se pueden ubicar.

Descripción del sistema a partir de la integración de tecnologías

Se propone, entonces, analizar, diseñar e implementar un sistema de información llamado SoftParking, que permita registrar y controlar el proceso de ubicación en los diferentes parqueaderos de la ciudad de Medellín.

Este sistema se implementará utilizando un lenguaje de programación orientado a objetos, como lo es C#, un sistema gestor de bases de datos SGBD, SQL server. De igual modo, el sistema de información estará apoyado por un sistema físico estructurado por sensores y redes inalámbricas, las cuales permitirán el flujo de información desde el software al hardware. Toda esta tecnología de computación TI estará respaldada por las tecnologías de la información y la comunicación (ver figura 1).

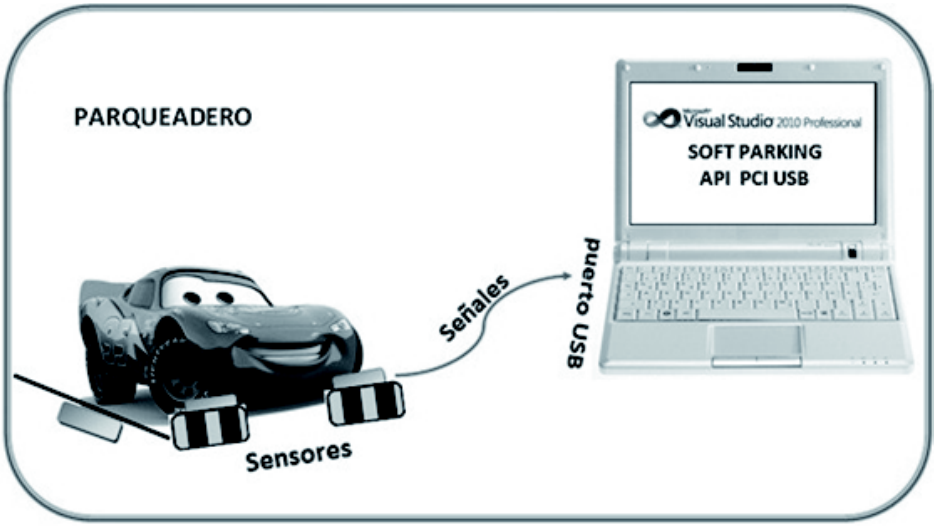


Figura 1. Integración de hardware y software

Descripción de las tecnologías de TI y TIC para apoyar el sistema de información Softparking

TI

Según Pérez (2004), TI es el uso de tecnología computacional (hardware y software) en la empresa. También se define TI como el uso de computadoras y telecomunicaciones para el procesamiento y la distribución de información digital, de audio, de video y otros medios.

Sistemas de información

Un sistema de información es una aplicación formada por una base de datos y una interfaz grafica (GUI), la cual permitirá la gestión en forma amigable de la información que se obtiene en forma de consultas a partir de la base de datos. Un sistema de información se debe caracterizar por su entorno amigable para el usuario, su flexibilidad, y portabilidad respecto a las plataformas o sistemas operativos donde se opere.

El proceso de ingeniería de desarrollo de aplicaciones informáticas orientadas a la web comprende la toma de decisiones en los aspectos de diseño e implementación que inevitablemente influyen en todo el proceso de desarrollo.

Aspectos claves para la creación de un sistema de Información

El proceso comienza con la captura de requisitos. El grupo de técnicos toma la información suministrada por los usuarios y clientes. Esta información puede provenir de fuentes muy diversas: documentos, aplicaciones existentes, entrevistas, etc. En base a esta información, el equipo de desarrollo elabora el catálogo de requisitos. Finalmente con la validación de requisitos se realiza la valoración de los mismos, comprobando si existen inconsistencias, errores o si faltan requisitos por definir. El proceso de definición-validación es iterativo y en algunos proyectos complejos resulta necesario ejecutarlo varias veces.

Resulta muy difícil establecer criterios para seleccionar técnicas apropiadas. Entre estos criterios pueden considerarse la facilidad de aprendizaje y de uso, la escalabilidad, el costo, la calidad y completitud de los resultados y el tiempo requerido para aplicar las técnicas. Los casos de uso son apropiados tanto para pequeños como para grandes sistemas, mientras que el uso de plantillas resulta menos apto para grandes sistemas (Abda, 2002).

Sistemas de información web

"El funcionamiento de un sitio web es un ejemplo típico de la arquitectura cliente-servidor, en donde múltiples clientes se conectan a un servidor en for-

ma simultánea. En general, el servidor depende de la instalación del sitio mientras que el cliente suele ser un *browser*. En todo esquema cliente-servidor debe existir un protocolo que especifique de qué forma se comunican e intercambian datos el cliente y el servidor, se utiliza el protocolo http que funciona encapsulado sobre el protocolo TCP/IP ("transmission control protocol/internet protocol")" (Busso, 2003).

Las aplicaciones hipermedia han evolucionado en los últimos años y se han concentrado mayormente en la web. Las antiguas aplicaciones distribuidas en CD dieron lugar a aplicaciones dinámicas, de constante actualización e incluso personalizables, capaces de adaptarse a los tipos de usuarios y, en casos avanzados, a cada usuario. Estas características encuentran el medio ideal en la web, ya que de otra forma sería costoso su mantenimiento y evolución.

La complejidad del desarrollo web ocurre en diferentes niveles: dominios de aplicación sofisticados (financieros, médicos, geográficos, etc.); la necesidad de proveer acceso de navegación simple a grandes cantidades de datos multimediales, y, por último, la aparición de nuevos dispositivos para los cuales se deben construir interfaces web fáciles de usar. Esta complejidad en los desarrollos de software sólo puede alcanzarse separando los aspectos de modelización en forma clara y modular (Silva, 2001).

Lenguaje de programación C#

Visual C# es un lenguaje de programación diseñado para construir sistemas de información basados en red o, mejor aun, en internet. Se trata de un lenguaje orientado a componentes, flexible, potente, claro y elegante, en el que se han escrito los servicios de la plataforma .NET (Hoffman, 2007).

Sensores

"Un sensor es un dispositivo capaz de transformar magnitudes físicas o químicas, llamadas variables de instrumentación, en magnitudes eléctricas. Las variables de instrumentación dependen del tipo de sensor y pueden ser, por ejemplo: temperatura, intensidad lumínica, distancia, aceleración, inclinación, desplazamiento, presión, fuerza, torsión, humedad, pH… La magnitud eléctrica obtenida puede ser una variación de resistencia eléctrica, de capacidad eléctrica, de tensión eléctrica, de corriente eléctrica"(Vigara, 2009).

Los sensores proporcionan información crítica de un sistema de control y realizan la medición de los cambios que se den en el mismo, por influencia de interferencias ambientales y físicas (Chaves, 2008). Así mismo, Proporcionan información analógica al sistema, la cual se acondiciona antes de pasar al microprocesador.

Sensores de posicionamiento y movimiento

Los transductores de posición se utilizan para determinar la posición de un objeto con respecto a un punto de referencia. Las posiciones pueden ser lineales o angulares.

Un sensor de posición consiste en un potenciómetro conectado al objeto que se desplaza, el cual al moverse varía la resistencia del mismo y, por lo tanto, hace posible calcular el cambio de posición que es proporcional a el cambio de posición. Estos se conocen como transductores resistivos de desplazamiento, y pueden medir movimiento tanto lineal como rotacional. Además, son relativamente económicos; el principal inconveniente para su empleo es el desgaste que se produce en el elemento móvil.

Protocolo USB

El USB es un bus punto a punto: dado que el lugar de partida es el *host* (PC o *hub*), el destino es un periférico u otro *hub*. No hay más que un único *host* (PC) en una arquitectura USB. Los PC estándares tienen dos tomas USB, lo que implica que para permitir más de dos periféricos simultáneamente, es necesario un *hub*. Algunos periféricos incluyen un *hub* integrado; por ejemplo, el teclado USB, al que se le puede conectar un *mouse* USB.

La arquitectura USB se forma por un *host* y dispositivos USB a él conectados. El sistema USB se compone de varios niveles de hardware y software. Primero, una aplicación requiere el acceso a un periférico USB igual que los periféricos comunes: llamada a funciones de la API (Application Programming Interface.). En un segundo paso, la API llama a rutinas del *driver* cliente del periférico USB instalado; este *driver* traduce los comandos de la API a comandos USB. El *driver* cliente es generalmente parte del sistema operativo o viene para ser instalado junto con el dispositivo USB**.** (Quisnancela, 2005).

PCI

Un Peripheral Component InterConnect (PCI, "Interconexión de Componentes Periféricos") consiste en un *bus* de ordenador estándar para conectar dispositivos periféricos directamente a su placa base. Estos dispositivos pueden ser circuitos integrados ajustados en ésta o tarjetas de expansión que se ajustan en conectores. Esta familia de microcontroladores contiene registros propio para el manejo de este protocolo USB.

Conclusiones

El artículo refleja la importancia de la integración de tecnologías para la creación de un sistema que apoye el proceso de parqueo, respecto a la problemática que se presenta en nuestra ciudad y en el país. Es válido afirmar entonces que tenemos la tecnología y la capacidad para mejorar la funcionalidad del proceso de parqueo en nuestro país.

Existen diversidad de elementos físicos como sensores, *drivers*, puertos, que pueden servir para el montaje del objeto físico que interactuara con el software a implementar. Por lo tanto es importante realizar una investigación más detallada de cada uno de estos componentes y determinar la compatibilidad, y funcionalidad que mejor se ajusten a las necesidades que marca el problema.

Referencias

Abda, Mercedes (2002). *Ingeniería de requisitos en aplicaciones para la web, un estudio comparativo.* Universidad de Sevilla, Departamento de Lenguajes y Sistemas Informáticos, España.

Chaves, P.A. (2008). *Sensores analógicos utilizados en la automatización industrial*. Universidad de Costa Rica, Escuela de Ingeniería Eléctrica, Costa Rica.

Castells, Manuel. (2002). Tecnologías de la información y comunicación y desarrollo global. *Revista de Economía Mundial,* Universidad Oberta de Cataluña, España.

Chio, N., Tibaduiza, D.A., Aparicio, L.C. & Caro, L.M. (2010). Redes de sensores inalámbricos. Congreso de Mecatrónica, Bucaramanga.
Hoffman, K. (2006). *Programación Visual C# 2005.* Anaya Multimedia, España.

Pérez, A. (2004). Situación actual de las pequeñas y medianas empresas consultoras de tecnologías de información (PyMES-TI) en el noroeste de México. *Memorias de la III Conferencia Iberoamericana de Sistemas, Cibernética e Informática (CISCI 2004),* Universidad de Sonora, Departamento de Ingeniería Industrial, México. Vol. I, pp. 274-279, Orlando, Florida, EE.UU.

Quisnancela, L.R, Roa D.R. (2005). Análisis y diseño de dos tarjetas de interfaz de red de datos para comunicaciones Power Line In Home (Aplicación para puerto PCI y USB). Escuela Politécnico del Ejército, Facultad de Ingeniería Electrónica, Ecuador.

Silva, D.A. (2001). Construyendo aplicaciones web con una metodología orientada a objetos. *Revista Colombiana de Computación,* vol. 2, núm. 2.

Conceptos y tecnologías para M-Learning

Concepts and technologies for M-Learning

Eucario Parra Castrillón
Ingeniero de sistemas, Magíster en Educación,
Magíster en Software Libre
eucarioparra5@gmail.com
Universidad San Buenaventura Medellín

Resumen

Los conceptos sobre e-learning y m-learning aquí expuestos son el fundamento del proyecto "Evaluación cualitativa a través de dispositivos móviles utilizando software libre". En este proyecto se utilizará software libre para crear un sistema para dispositivos móviles. El objetivo es dar herramientas a los profesores para aplicar un modelo de escala de Likert que determinar el grado de interpretación de sus estudiantes ante una situación problema. Aparte de las funcionalidades técnicas y la usabilidad requeridas en un ambiente de computación móvil, el sistema permitirá aplicar la prueba en tiempo real, en la medida en que evolucione el análisis e interpretación del estudiante. Es decir, este irá evaluando la situación presentada y en ese momento, a través del dispositivo móvil, se le presentará la escala de Likert. El aprendizaje móvil o m-learning se concibe como e-learning en dispositivos móviles. Por esa razón, es necesario establecer ciertos presupuestos, ya que la naturaleza tecnológica de estos aparatos exige particulares herramientas y la asimilación de las diferencias entre estos conceptos.

Palabras clave: e-learning, m-learning, aprendizaje móvil, dispositivos móviles, redes inalámbricas, protocolos de redes.

Abstract

The concepts of e-learning and m-learning here presented serve as a foundation to the project "Qualitative evaluation through mobile devices using free software". This project will use free software to create a system for mobile devices. This aims to provide teachers with tools to apply a Likert scale model, in order

Recibido: 11 de junio 2011
Aprobado: 26 de octubre 2011

to determine the degree of interpretation their students give to a problem situation. Besides its technical capabilities and usability that are required in a mobile computing environment, the system will allow to apply a test in real time, as the student's analysis and interpretation evolve. That is, the system will be continuously evaluating the current situation and, at that time, through the mobile device, the Likert scale will be presented. Mobile learning, or m-learning, is conceived as a kind of e-learning on mobile devices. That is why it is necessary to establish some estimates, since the technological nature of these devices demands specific tools and the assimilation of the differences between these concepts.
Keywords: e-learning, m-learning, mobile learning, mobile devices, wireless networks, network protocols.

Introducción

El siguiente estado del arte se enfocará en temáticas relativas a la transformación del *e-learning* en *m-learning*, la relevancia de algunos proyectos de *m-learning* en el ámbito mundial, referencias sobre fundamentos de las escalas de Likert, software libre y proyectos sobre evaluación de aprendizajes aplicando dispositivos móviles. Por supuesto, es muy difícil establecer delimitaciones sobre las cuales pueda converger todo lo que se haya desarrollado en estos temas, razón por la cual solo se presentan algunos proyectos destacados.

El tema dominante es el *m-learning* o aprendizaje móvil, enfocado en este proyecto de investigación desde el punto de vista tecnológico, pero sin perder de vista la incidencia de las concepciones pedagógicas y comunicativas. Los temas de evaluación de aprendizajes y escalas de Likert son también de interés, pues para automatizar esos procesos se creará un prototipo de software que permita aplicarlo en dispositivos móviles. Es de aclarar que en el proyecto no se aplicará la escala de Liker para recoger información, sino que se desarrollará un software que permita a los estudiantes responder evaluaciones de sus profesores desde un dispositivo móvil. Estas evaluaciones tendrán formato de escala de Likert y se adaptarán para medir interpretaciones en lugar de actitudes frente a una descripción, tal como se le utiliza en psicología.

El *e-learning* o aprendizaje electrónico es más que un sistema de acceso a información y de distribución de conocimientos. Además de cumplir estas dos funciones, debe proveer mecanismos de participación pedagógica para la interacción entre los participantes, teniendo en cuenta la educación como un proceso constructivo, la tecnología como soporte y medio para la creación de ambientes educativos y la comunicación como elemento esencial para la finalidad esencial, que es el desarrollo del ser dentro de los contextos en los que actúa.

El *e-learning* brinda la posibilidad de acceder al aprendizaje cuando el usuario lo necesite, en el lugar en el que lo necesite, e implica que el aprendizaje no tenga que ser una experiencia pasiva de los alumnos en un aula frente a un profesor. De otra parte, el *e-learning* puede hacer el aprendizaje interesante y convincente: con él áreas difíciles se pueden hacer agradables y atractivas.

El *e-learning* se ha venido transformando con los avances tecnológicos. Es así como las lecciones digitales venían antes empacadas en CD con un buen nivel de multimedia, pero sin posibilidad de interacción entre los usuarios y el profesor y con la imposibilidad de actualizar contenidos. Con la disponibilidad de internet, cobraron fuerza los cursos *on-line*, que brindaban la posibilidad de interacción entre los participantes, versatilidad en las comunicaciones sincrónicas o asincrónicas, actualización y adaptación de los contenidos y utilización de plataformas LMS - Learning Management System -, plataformas para la gestión de aprendizajes. Terminada la primera década del siglo XXI, con el advenimiento de la tecnología móvil, el desarrollo del *e-learning* adquirió la forma de *m-learning*, de manera acorde con la masificación de los dispositivos móviles como recursos de comunicación y acceso a información en distintos formatos.

Un ejemplo de desarrollo en la aplicación de las tecnologías de la información y la comunicación —TIC— en actividades formativas son las experiencias en el Instituto Tecnológico de Monterrey, donde ha sido notable la evolución del aula de clase. Se ha pasado por la clase presencial, la incorporación de tecnología a las aulas, la educación satelital, la educación en línea aprovechando la web, hasta llegar en el 2010 a las innovaciones con *m-learning* (García-Valcárcel, 2009).

El *e-learning* o aprendizaje electrónico es más que un sistema de acceso a la información y de distribución de

conocimientos. Además de estas dos funciones, debe proveer mecanismos de participación pedagógica para la interacción entre los participantes, considerando la educación como un proceso constructivo, la tecnología como un soporte y un medio para la creación de ambientes educativos y la comunicación como un elemento esencial para su finalidad, que es el desarrollo del ser dentro de los contextos en los que actúa.

Así entonces, el *e-learning* no abarca únicamente nuevas tecnologías para el aprendizaje, más bien puede concebirse como nuevas formas de pensar las relaciones de mediación y de comunicación.

El aprendizaje electrónico: *e-learning*

El aprendizaje electrónico, *e-learning,* es un componente esencial del modelo pedagógico para la nueva sociedad de redes y de la información. Se refiere en esencia el uso de recursos de información por medios electrónicos para contribuir al aprendizaje y a la producción de conocimiento.

Al *e-learning* corresponde un conjunto de procesos que en el contexto de las tecnologías de la información y la comunicación están cambiando dinámicamente los modelos y las estrategias de aprendizaje. El *e-learning* es una actividad social; en consecuencia, las experiencias de aprendizajes pueden lograrse, no sólo mediante contenidos estructurados, sino también por medio de comunidades y redes. De esta manera, el *e-learning* apoya el aprendizaje a través de la reflexión y la discusión.

Por otro lado, el *e–learning* puede empoderar a los alumnos en el manejo de su propio aprendizaje. Todos aprenden de diferentes maneras. *e-learning* significa que se puede tener acceso a una variedad de recursos de aprendizaje y comunidades de personas. De este modo, cada alumno puede tener una experiencia individual y personalizada de aprendizaje. Más allá, esta modalidad de aprendizaje está ayudando a incrustar el aprendizaje en los ambientes de trabajo; las organizaciones empiezan a reconocer que el aprendizaje no ocurre sólo en el salón de clases, también ocurre en el trabajo. El aprendizaje se ha movido del aula al escritorio y de allí al bolsillo. Gracias a ello, el *e-learning* facilita que las organizaciones se vuelvan ágiles y competitivas en el mercado.

Las dos dificultades técnicas más comunes que decepcionan a alumnos en programas de *e-learning* son: el bajo ancho de banda y las configuraciones incorrectas. El ancho de banda es la primera frustración que encuentran los alumnos, pero un navegador incorrecto y un PC mal configurado es la segunda causa más común de deserción. Otros obstáculos comunes son la ausencia de interacción entre el profesor y los estudiantes y la resistencia cultural.

Sin embargo, sobre estas dificultades han avanzado las investigaciones educativas y tecnológicas, con el fin de crear metodologías, mecanismos y sistemas que hagan del *e-learning* una opción formativa de calidad. Al respecto merece atención la siguiente definición, la cual resume los propósitos y puntos de atención del aprendizaje electrónico.

> Todas aquellas metodologías, estrategias o sistemas de aprendizaje que emplean tecnología digital y/o comunicación mediada por ordenadores para producir, distribuir y organizar conocimiento entre individuos comunidades y organizaciones (Bernárdez, 2007, p.5).

La aparición y la progresión del *e-learning* están indefectiblemente ligadas al desarrollo tecnológico. Esto se evidencia en la transformación del formato de contenidos: los formatos de papel se transforman a formatos electrónicos para ser expuestos en las clases, con el agregado de poder ser modificados, masivamente almacenados y fácilmente accesibles. Luego, esos formatos se suben a la web para ampliar el rango de potenciales usuarios y racionalizar las necesidades de almacenamiento. Además, con el texto se incorpora otro tipo de software, como videos o animaciones. Luego se incorporan los cursos en línea, para agregar a la capacidad de información las posibilidades de comunicación entre los participantes. Por último, se personalizan los contenidos y las comunicaciones, de acuerdo con la particularidad y la movilidad de cada usuario (Barberá, 2008).

Aprendizaje móvil: *mobile learning*

Mobile learning (*m-learning*) es una forma de aprendizaje generada a partir de la conjunción entre el *e-learning* y la aplicación de los dispositivos móviles (*smart devices*) inteligentes, como PDAs, *smartpho-*

nes, *Ipods*, *pocket PCs*, teléfonos. Se fundamenta en la posibilidad que brindan estos móviles para combinar la movilidad geográfica con la virtual, lo que permite el aprendizaje en el momento en que se necesita, en el lugar donde se encuentre y con la información precisa que se requiera.

El *m-learning* se desarrolla apoyado en las siguientes características de los servicios móviles ISEA (2009):

- Accesibilidad: tecnológicamente son pocas las limitaciones en el tiempo o en el espacio para utilizar los servicios y comunicarse.
- Conveniencia: los servicios se disponen en paquetes; se empaquetan servicios (como agenda, radio, teléfono) y se utilizan dónde y cuándo se quiere.
- Inmediatez: no existen retrasos entre el acto y la comunicación. (el acto se comunica en tiempo real).
- Localización: la movilidad geográfica se acompaña de sistemas de localización segmentados.
- Personalización: los servicios y los terminales son adaptables a las necesidades y a los gustos de los usuarios.
- Ubicuidad: permiten la comunicación y la ejecución de programas de distintas maneras y con distintas fuentes y destinos al mismo tiempo.

Ventajas funcionales del *m-learning*

También en ISEA (2009) se formulan las siguientes ventajas funcionales del *m-learning*:

- Aprendizaje en todo tiempo y en cualquier lugar: no se requiere estar en un lugar particular ni a una hora determinada para aprender. El dispositivo móvil puede usarse en cualquier parte y en cualquier momento, por lo que el aprendizaje se personaliza y adapta a los requerimientos y a las disponibilidades individuales.
- Los dispositivos móviles posibilitan la interacción en tiempo real entre los estudiantes y el profesor. Se facilita de forma automática la retroalimentación por parte del profesor.
- Mayor cobertura: la telefonía móvil está al alcance de la mayoría de la población. El dispositivo móvil es un accesorio ya común en la gente.
- Mayor accesibilidad: la tendencia tecnológica es que los dispositivos móviles tengan conexiones a las redes y servicios de acceso a la web.
- Mayor portabilidad y funcionalidad: los estudiantes pueden escribir anotaciones directamente en el dispositivo, durante el mismo tiempo que reciben los objetos de aprendizaje y las comunicaciones con sus asesores. Además, pueden grabar las sesiones para acceder de nuevo a las instrucciones más tarde.
- Aprendizaje colaborativo: la tecnología móvil favorece que los estudiantes compartan el desarrollo de determinadas actividades con distintos compañeros, creando equipos, socializando, compartiendo respuestas, compartiendo propuestas.
- Se facilita el aprendizaje exploratorio: en tiempo real, en la medida en que se recibe información se puede estar explorando, experimentando y aplicando sobre contextos y campos de práctica.

Ventajas pedagógicas del *m-learning*

Aunque en este proyecto el interés del desarrollo se concentra en los aspectos tecnológicos, por tratarse de la aplicación a un ámbito educativo, deben tenerse en cuenta los aspectos pedagógicos referenciados en ISEA (2009):

- Se potencia en los estudiantes capacidades para leer, escribir, calcular y reconocer escenarios existentes.
- Se incentivan experiencias de aprendizajes independientes y grupales.
- Se les facilita a los estudiantes ayuda y respaldo de manera inmediata, cuando lo requieran.
- Permite que los docentes envíen avisos a sus estudiantes sobre plazos de actividades o tareas, así como mensajes de apoyo y estímulo.
- Ayuda a elevar la confianza de los estudiantes en la medida la información y las comunicaciones están disponibles a toda hora.
- Se favorece la posibilidad de interactuar con la información en distinto formatos (audio, voz, video, texto, animación).
- Se favorece la posibilidad de agrupar y grabar la información para luego revisarla.
- Se permite reingresar a un mismo objeto de aprendizaje cuantas veces se quiera.
- La comunicación entre los estudiantes es inmediata y adaptable a las necesidades de cada uno.
- Fácilmente mientras se realizan otras tareas, se puede acceder a los objetos de aprendizaje, para su exploración parcial o total.

- Se puede acceder a los servicios de la web de manera rápida. El correo, el chat, la convivencia con comunidades sociales son posibilidades que elevan el valor del *m-learning*.

Desventajas del *m-learning*

Complementando lo anterior, en ISEA (2009) se hace referencia a las siguientes desventajas del m-learning, las cuales deben considerarse como un reto dentro de los procesos de desarrollo tecnológico y pedagógico:

- La conectividad no siempre depende de la capacidad del dispositivo móvil, sino de la disponibilidad de las redes locales y satelitales. Además, el ancho de banda puede variar entre un lugar y otro, siendo por esto posible que en el momento más inesperado se pierda el contacto con el profesor o con los objetos alojados en una web.
- Los dispositivos móviles presentan limitaciones asociadas a la usabilidad, ya que tienen pantallas pequeñas.
- Aunque los avances tecnológicos son notables, en algunos dispositivos es difícil leer textos en tamaño normal, pues la cantidad de información visible es limitada.
- Algunos móviles tienen diseños que los hacen demasiado compactos, lo que afecta su usabilidad.
- Aunque los dispositivos pueden ser económicamente asequibles, el costo de la comunicación, determinado por las empresas administradoras y proveedoras de redes públicas, puede llevar a que el servicio *m-learning* resulte con costos altos.

Tecnologías empleadas en el *m-learning*

El desarrollo del *m-learning* se fundamenta en la capacidad de las redes y de los dispositivos para las comunicaciones inalámbricas. A continuación se definen las más importantes:

Tecnología GPRS: GPRS es la sigla de General Packet Radio Services (servicios generales de paquetes por radio). A menudo se describe como "2,5 G", es decir, una tecnología entre la segunda (2G) y la tercera (3G) generación de tecnología móvil digital. Se transmite a través de redes de telefonía móvil y envía datos a una velocidad de hasta 114 kbps. El usuario puede utilizar el teléfono móvil y el ordenador de bolsillo para navegar por internet, enviar y recibir correo, y descargar datos y soportes. Permite realizar videoconferencias con sus colegas y utilizar mensajes instantáneos para charlar con sus familiares y amigos, esté donde esté. Además, puede emplearse como conexión para el ordenador portátil u otros dispositivos móviles (Mundotelme, 2007).

Tecnología 3G: al igual que la GPRS, la tecnología 3G (tecnología inalámbrica de tercera generación) es un servicio de comunicaciones inalámbricas que le permite estar conectado permanentemente a internet a través del teléfono móvil, el ordenador de bolsillo, el Tablet PC o el ordenador portátil. La tecnología 3G promete mejor calidad y fiabilidad, mayor velocidad de transmisión de datos y ancho de banda superior (lo que permite ejecutar aplicaciones multimedia). Con velocidades de datos de hasta 384 kbps, es casi siete veces más rápida que una conexión telefónica estándar, lo que a su vez hace posibles las video llamadas, dado que dichas llamadas se harán con una conexión directa a internet. Hay empresas de telefonía que ofrecen este servicio de manera gratuita (Mundotelme, 2007).

> Se dice que los usuarios de GPRS y 3G están "siempre conectados", dado que con estos métodos de conexión tienen acceso permanente a internet. Mediante los mensajes de texto cortos, los empleados de campo pueden comunicar su progreso y solicitar asistencia. Los ejecutivos que se encuentran de viaje pueden acceder al correo electrónico de la empresa, de igual modo que puede hacerlo un empleado de ventas, quien también puede consultar el inventario. Puede automatizar su casa o su oficina con dispositivos GPRS y 3G supervisar sus inversiones.

Tecnología GSM: GSM es un sistema digital de comunicación que transmite voz y datos. Se lo considera de segunda generación (2G), ya que a diferencia de la primera generación de celulares, utiliza tecnología digital y la división de acceso de transmisión múltiple (TDMA). GSM digitaliza y comprime la información y luego divide cada canal de 200MHZ en ocho espacios de tiempo de 25MHZ. Este sistema opera en las bandas 900MHZ y 1800MHZ en Europa, África y Asia, y en las bandas 850MHZ y 1900MHZ en los Estados Unidos. La banda 850MHZ también se utiliza para GSM y 3GSM en Canadá, Australia y en varios países de Latinoamérica.

> Dos de las grandes ventajas del GSM es que permite la transmisión de datos a velocidades de hasta de 9,6 kbt/s facilitando el servicio de mensajes cortos (SMS, por sus siglas en inglés). Otra de sus grandes ventajas es el *roaming* internacional, que permite usar un celular en cualquier país del mundo donde exista la tecnología (Serrano, 2007).

Tecnología UMTS: la tecnología UMTS (Universal Mobile Telecommunications System) es el término utilizado en Europa para referise a las redes y servicios móviles de tercera generación. Permite transmitir datos a una velocidad máxima de 384 kbps, superior a las líneas RDSI y ADSL estándar. A través del UMTS es posible ofrecer nuevos servicios multimedia, tales como videotelefonía, descarga de ficheros a gran velocidad o juegos interactivos y multijugador, desde dispositivos móviles.

UMTS es el estándar europeo y de otros muchos países de la llamada tercera generación de la telefonía móvil. La primera fue la analógica (estándares ETACS o TDMA), y la segunda, la más extendida actualmente, se conoce como GMS o GMS/GPRS (Terra, 2004).

Tecnología Wi-Fi: la sigla para Wireless Fidelity (Wi-Fi) significa literalmente fidelidad inalámbrica. Es un conjunto de redes que no requieren de cables y funcionan en base a ciertos protocolos previamente establecidos. Si bien fue creado para acceder a redes locales inalámbricas, hoy es muy frecuente que sea utilizado para establecer conexiones a internet., que está a WiFi es una marca de la compañía Wi-Fi Alliance cargo de certificar que los equipos cumplan con la normativa vigente (que en el caso de esta tecnología es la IEEE 802.11). Esta nueva tecnología surgió por la necesidad de establecer un mecanismo de conexión inalámbrica que fuera compatible entre los distintos aparatos. En busca de esa compatibilidad fue que en 1999 las empresas 3com, Airones, Intersil, Lucent Technologies, Nokia y Symbol Technologies se reunieron para crear la Wireless Ethernet Compability Aliance (WECA), actualmente llamada Wi-Fi Alliance (Mis Respuestas.com, s.f.).

Tecnología Wi-Max: WiMAX está diseñado como una alternativa inalámbrica para DSL y cable para el acceso de banda ancha por última milla y como una forma de interconectar puntos de acceso Wi-Fi en una red de área metropolitana. Los usos reales de WiMAX se superponen a los de la red local inalámbrica móvil hasta el nivel de red de área extensa. En teoría, WiMAX puede proporcionar conectividad a los usuarios dentro de un radio de 31 millas, aunque no haya una línea directa a la vista. Sin embargo, las pruebas reales de campo muestran que los límites prácticos parecen estar a la vuelta de 3 a 5 millas. Según los defensores de WiMAX, la tecnología puede proporcionar velocidades de datos compartida de hasta 70 Mb/s. Es suficiente 60 conexiones tipo T1 en forma simultánea y más de mil casas funcionando a 1 Mb/s de conectividad DSL. Las tasas máximas de datos útiles en las pruebas reales de campo muestran sólo puede ir entre 500 kb/s hasta 2 Mb/s y es muy dependiente de las condiciones en un lugar determinado. (Compute-rs.com, s.f.)

Tecnología Bluetooth: *l*as comunicaciones inalámbricas están presentes en muchas de nuestras actividades diarias. Su uso ha llegado a ser tan común que perdemos la percepción de lo útil y a veces indispensable que pueden llegar a ser. Las redes celulares para transmitir voz y datos han surgido para proveer la movilidad y disponibilidad de la comunicación que exige el ritmo acelerado de vida en las grandes urbes. La utilización de sensores infrarrojos y de radiofrecuencia provee la comodidad de controlar y operar a distancia aparatos electrónicos volviendo más sencillo nuestro quehacer diario. Asimismo, la creación de estándares de comunicaciones inalámbricas en las redes de transmisión de datos ha abierto oportunidades de desarrollo de estas tecnologías, aprovechando la utilización de interfaces aéreas operadas bajo frecuencias no licenciadas.

Bluetooth forma parte de las tecnologías creadas para proveer comunicación inalámbrica en áreas de uso personal. Sin embargo, su uso va más allá de la eliminación de cables, ya que es lo suficientemente flexible para permitir la creación de aplicaciones que abren un mundo con límite en la imaginación. Esta tecnología desarrollada por Ericsson en 1994 hace factible la conectividad inalámbrica entre dispositivos a corta distancia, los cuales pueden llegar a formar redes con diversos equipos de comunicación: computadoras móviles, radiolocalizadores, teléfonos celulares, PDA, e, inclusive, electrodomésticos.

El estándar Bluetooth se compone de dos capítulos; uno de ellos describe las especificaciones técnicas principales, mientras que el otro define perfiles específicos para aplicaciones. Estos últimos aseguran la interoperabilidad de dispositivos Bluetooth entre

fabricantes. Algunos de estos perfiles son: de acceso genérico, identificación de servicio, puerto serial, acceso a LAN sincronización y de dispositivo de información móvil (MIDP).

La IEEE ha desarrollado un protocolo equivalente denominado Wireless Personal Área Network (WPAN), 802.15, con el objetivo de lograr la interoperabilidad con otros dispositivos inalámbricos. Las siguientes son algunas de sus características:

- Tecnología inalámbrica. Reemplaza la conexión alámbrica en distancias que no exceden los 10 metros, alcanzando velocidades del rango de 1mb/s.
- Comunicación automática. La estructura de los protocolos que lo forman favorece la comunicación automática sin necesidad de que el usuario la inicie (Velásquez, 2004).

Tecnología RFID: la identificación por radiofrecuencia, o RFID por sus siglas en inglés (*radio frequency identification*), es una tecnología de identificación remota e inalámbrica en la cual un dispositivo lector, o *reader,* vinculado a un equipo de cómputo, se comunica a través de una antena con un *transponder* (también conocido como *tag* o etiqueta) mediante ondas de radio. Esta tecnología, que existe desde los años 40, se ha utilizado y se sigue utilizando para múltiples aplicaciones, incluyendo casetas de peaje, control de acceso, identificación de ganado y tarjetas electrónicas de transporte.

En los últimos años, la tecnología RFID ha entrado al *mainstream* tecnológico gracias a su creciente difusión en aplicaciones de cadenas de suministro motivadas por las iniciativas de las cadenas de autoservicio y departamentales (Egomexico, s.f.).

Conclusiones

Los avances de las tecnologías de la información y la comunicación - TIC - le imponen a la educación virtual retos nuevos y puntos de transformación. Con base en este concepto son varias las inquietudes que encierran el desarrollo del proyecto, las cuales se pueden clasificar en tres dimensiones: la aplicación del software libre, la utilización de los dispositivos móviles para fines educativos y la evaluación dentro del *e-learning*. Mediante.

la planeación, el desarrollo y la divulgación de investigaciones aplicadas que utilicen software libre para crear soluciones tecnológicas innovadoras con impacto comercial y social, se puede contribuir a la consolidación de una industria de información especializada en programas de código y estándares abiertos y de libre distribución.

El software libre en Colombia requiere, como en el caso de Extremadura, en España, de una política pública sobre software libre que comprometa a los sectores educativos, de infraestructura, empresariales y sociales. Una forma de motivar esta política es mostrando soluciones prácticas a problemáticas estratégicas de impacto amplio. Hacia esto apunta la propuesta de una solución con software libre para el *m-learning*.

Podría entonces decirse que el *m-learning* es una oportunidad para que se aplique software libre y de esta forma se abran oportunidades empresariales para el desarrollo de software o el soporte de aplicaciones. La divulgación y puesta en escena del software libre es el primer problema que justifica el desarrollo del proyecto de investigación presentado.

Un segundo problema concreto es el *e-learning*. La inquietud está en la adecuación de plataformas tecnológicas para las intenciones pedagógicas y comunicativas que se encierran en los ambientes educativos. Las etapas de los ambientes de *e-learning* como meros repositorios de contenidos o como plataformas LMS - Learning Management System- para la gestión de cursos están superadas. Se requieren soluciones para la atención en tiempo real, la adaptación de contenidos, la convergencia en la mensajería, la construcción y la integración a comunidades sociales digitales, así como el despliegue de sistemas adaptativos que consideren las circunstancias temporales y espaciales de los usuarios. Por esta necesidad se justifica el concepto de *m-learning* como solución tecnológica para la educación.

La propuesta de ampliar la gama de oportunidades tecnológicas para los ambientes virtuales de aprendizaje justifica la propuesta de aprendizaje móvil o *m-learning*. Además, porque con el desarrollo de internet, la convergencia de las telecomunicaciones, las comunicaciones integradas de voz y datos sobre IP y los dispositivos móviles evolucionan las oportunidades para la creación de ambientes pedagógicos centrados en las interacciones y situaciones constructivas de los estudiantes.

Un tercer problema tiene que ver con la evaluación educativa. - tal vez lo que más dudas generan los ambientes de aprendizaje virtuales, ya que se presentan dos problemas: el de la autenticidad y el de la retroalimentación. En el primer caso, no es fácil establecer controles para que quien responde a la prueba lo haga sin alterar las condiciones planteadas, por lo cual es necesario orientar el tipo de preguntas de manera que se fomente el autocontrol del proceso mismo. Esto es, según los problemas o preguntas planteadas se pueden controlar las actitudes de los evaluados para que respondan con autenticidad. Otra opción es crear alternativas tecnológicas que aseguren privacidad, rastreabilidad y respuesta en tiempo real.

Referencias

Barberá E. (2008). *Aprender e-learning*. Barcelona: Editorial Paidós.

Barroso E. (2008). *Creación de entonos adaptativos móviles: recomendación de actividades y generación dinámica de espacios de trabajos basados en información sobre usuarios, grupos y contextos.* Memoria presentada para optar al título de Doctora en Ingeniería Informática. Departamento de Ingeniería Informática, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Madrid.

Bernárdez M. (2007). *Diseño, producción e implementación de e-learning: Metodología, herramientas y modelos*. Bloomington: AuthorHouse.

Compute-rs.com (s.f.). *Usos de la tecnología WiMAX*. Extraído el 20 de noviembre de: http://www.compute-rs.com/es/consejos-267387.htm

Conde, M., Muñoz, C. y García, F. (s.f.). *Sistemas de adaptación de contenidos para dispositivos móviles*. Salamanca: Universidad de Salamanca, Dpto. de Informática y Automática.

Egomexico (s.f.). ¿Cómo funciona la tecnología de identificación por radiofrecuencia RFID?. Extraído el 20 de noviembre de http://www.egomexico.com/tecnologia_rfid.htm

Fontán T. y Basdos L. (2003). *La evaluación con tecnologías de la información y comunicación: recomendaciones didácticas*. Universidad de Las Palmas de Gran Canaria. Comunicación en el II Congreso Internacional de EducaRed (2003). Extraído el 21 de noviembre de 2010 de: www.educared.net/congresoii/comunicaciones/

García, V. (2009). Innovación en Servicios Empresariales Avanzados ISEA-. (2009). *Mobile learning, análisis prospectivo de las potencialidades asociadas al Mobile Learning*. Madrid: Plan Avanza.

Iguartua J. y Humanes L. (s.f.). *El método científico aplicado a la investigación en comunicación social*. Extraído el 26 de noviembre de 2010 de: http://www.portalcomunicacion.com/esp/pdf/aab_lec/6.pdf

Millán, R. (2006). *Nuevas comunicaciones móviles*. Extraído el 4 de septiembre de 2010 de http://www.ramonmillan.com/tutoriales/wap.php.

Mis respuestas.com (s.f.). *¿Qué es Wiki?* Extraído el 20 de noviembre de: http://www.misrespuestas.com/que-es-wifi.html.

Mundotelme (2007). *Tecnología 3G.* Extraído el 20 de noviembre de: http://mundotelme.com/tecnologia3g.php
Rivero López, S. (2009). *Evaluación con TIC*. Extraído el 28 de noviembre de: http://aprendizaje20.blogspot.com/2009/12/evaluacion-con-tic.html.

Soto, Sergio A. (s.f.). *Sistema web con acceso a bases de datos multiplataforma a través de teléfonos celulares*. Universidad Nacional del Nordeste. Corrientes (Argentina). Extraído el 4 de septiembre de 2010, de http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos.

Serrano, C. (2007). *Tecnología GSM ¿Cómo funciona?.* Extraído el 20 de noviembre de: http://celularesarg.blogspot.com/

Terra. (2004). *Qué ofrece la tecnología UMTS.* Extraído el 20 de noviembre de: http://www.terra.es/tecnologia/articulo/html/tec11333.htm

Serrano C. (2007). Tecnología GSM ¿Cómo funciona?. Extraído el 20 de noviembre de: http://www.terra.es/tecnologia/articulo/html/tec11333.htm.

Velásquez, L. (2004). *Bluetooth más que una conexión inalámbrica*. *Enter@te en línea.* Año 3, Número 33.

La inteligencia lógico-matemática y el aprendizaje para desarrollar algoritmos

Logical-mathematical intelligence and learning to develop algorithms

Jaime Alberto Acosta Gómez
Magíster en Educación y Desarrollo Humano
Docente Asociado Facultad de Informática
jacosta@tdea.edu.co

Fabio Franco Martínez
Magíster en Educación y Desarrollo Humano
Docente Titular Facultad de Informática
ffranco@tdea.edu.co

Resumen

El desarrollo de algoritmos requiere de acciones inteligentes y grandes cantidades de conocimiento frente al tema a tratar con el fin de permitir diferentes soluciones desde el razonamiento. La inteligencia lógico matemática podría decirse está relacionada directamente con la resolución de problemas y la razonamiento, provee a los individuos el entusiasmo de aprender y descubrir las matemáticas con el fin de desarrollar competencias de abstracción para detectar patrones, razonar deductivamente y pensar de manera lógica en situaciones que requieran soluciones científico técnicas.

Existen razones para considerar importante y positivo con lo que plantean teóricos de la inteligencias Múltiples como Howard Gardner en cuanto a la experiencia misma del acto educativo en que los estudiantes deben aprender a pensar y aprender a aprender de muchas maneras diferentes. Esto indica realizar una reflexión sobre los contenidos de los microcurrículos específicamente del área de la Algoritmia y de las prácticas pedagógicas requeridas para apoyar los procesos de aprendizaje a partir específicamente de las habilidades desarrolladas en la inteligencia lógico matemática dentro del aula a partir de la abstracción y el razonamiento.

Palabras Claves: Algoritmos, Aprendizaje, inteligencia lógico matemático, Razonamiento.

Recibido: 11 de junio 2011
Aprobado: 26 de octubre 2011

Abstract

The development of algorithms and intelligent action requires large amounts of knowledge in front of the issue to be addressed in order to allow different solutions from the reasoning. Logical mathematical intelligence could say is directly related to problem solving and reason, provides individuals with the enthusiasm to learn and discover mathematics to develop skills of abstraction to detect patterns, reason deductively and think logically, in situations that require scientific-technical solutions.

There are reasons to consider important and positive with the points theorists like Howard Gardner's Multiple Intelligences in the very experience of educational activity in which students must learn to think and learn to learn in many different ways. This indicates a reflection about the contents of microcurrículos specifically the area of Algorithms and pedagogical practices needed to support processes of learning from the skills developed specifically in mathematical logical intelligence in the classroom from the abstraction and reasoning.

Keywords: Algorithms, Learning, logical-mathematical intelligence, Reasoning.

Introducción

Los seres humanos con gran capacidad lógico matemática son capaces dentro de su proceso de evolución cognitiva resolver problemas de una manera rápida a partir de muchas variables, creando teorías que son evaluadas sucesivamente y posteriormente son aceptadas o rechazadas de acuerdo al contexto o al problema a analizar. Ello conlleva concebir implícitamente una serie de capacidades propias de identificar modelos, calcular, formular y verificar hipótesis, utilizar el método científico en procesos de alta rigurosidad investigativa y en el desarrollo de razonamientos inductivo y deductivo para la resolución de problemas propios de la algoritmia. Pero no solamente es la única inteligencia en la población humana, Howard Gardner, profesor de la Universidad de Harvard, distingue otras 6 inteligencias fundamentales de carácter hereditario, pero que podrían ser desarrolladas en un contexto histórico – social. Esta teoría ha implicado cambios en los contextos educativos a partir de su concepción basándose en la individualidad que supone el desarrollo de las diferentes inteligencias en cada sujeto y que operan en sí mismas de acuerdo a variables propias en el aprendizaje.

Por otro lado, el desarrollo de sistemas informáticos constituye un reto para crear software para las más diversas actividades humanas. Es el aula u otros ambientes de innovación tecnológica el lugar por excelencia para desarrollar dichos recursos necesarios para generar condiciones de alta competencia. En este contexto cabe decir que el desarrollo de las inteligencias para la producción de software es de vital importancia. Por ello el articulo trata de plantear respuestas a ¿Cómo contribuye la inteligencia lógica – matemática al desarrollo de la creatividad algorítmica?

Para el Siglo XXI se conciben para la formación de individuos unas habilidades básicas, de pensamiento y personales (Mascwitz, 2008). Dichas habilidades enfocan mas su interés especifico en leer, escribir, realizar operaciones aritméticas, matemáticas y razonar. Otra habilidad que se debe propender a desarrollar es la capacidad de resolución de problemas y lo más importante aprender a aprender lo nuevo, comprenderlo y aplicarlo. Ello permite entender que el estudiante en proceso de formación requiere un desarrollo muy exhausto de una actitud de análisis para resolver problemas que le permitirá a partir de las habilidades planteadas construir acciones en la solución de algoritmos. Ello parte en identificar el problema o sea entenderlo con claridad, una vez identificado a partir de un análisis juicioso definir las posibles caminos de solución, teniendo como base la construcción de posibles estructuras que permitan el dar solución al problema planteado.

La inteligencia lógico-matemática y el aprendizaje para desarrollar algoritmos

Para analizar el concepto de inteligencias múltiples debemos precisar en primera instancia la definición de inteligencia como la capacidad que diferencia al ser humano de los animales, asociándose con el razonamiento y el pensamiento lógico, estas últimas, bases fundamentales para adquirir destrezas en la

resolución de problemas algorítmicos, tema fundamental para los estudiantes de la tecnologías e Ingeniería de Software. Ello posibilita una caracterización que conlleva a un pensamiento más maduro y lógico que en su esencia se determina en el análisis, la argumentación, y el razonamiento como competencias básicas para la construcción de algoritmos de alto rendimiento. Estos se definen como la capacidad de resolver en un orden lógico un problema paso a paso de manera eficiente y construida a la luz de resolver un problema especifico.

Con el nacimiento de la idea de que no existe sólo un tipo de inteligencia, sino una multiplicidad de ellas por medio de la cual se desarrolla sólo el intelecto humano, en una progresión lineal y acumulativa, los desafíos que se le abren en la enseñanza de la cátedra de algoritmos son enormes ya que una de las inteligencias que posee el ser humano es la Lógica matemática, definida esta en la capacidad para resolver problemas con base en el cumplimiento de requisitos basados para su propósito en resolver un problema. Esto implica, modificar las prácticas pedagógicas de una manera notable, para que el alumno en dicho proceso desarrolle el pensamiento abstracto utilizando la lógica y los números para establecer relaciones entre distintos datos. Destacan, por tanto, en la resolución de problemas, en la capacidad de realizar cálculos matemáticos complejos y en el razonamiento lógico para adquirir las siguientes competencias:

- ⇒ Relacionar conceptos y Teorías
- ⇒ Razonar de forma deductiva e inductiva
- ⇒ Analizar conceptos abstractos que presenten objetos concretos.

Para lograr dichas competencias el alumno en compañía con el docente debe realizar prácticas o actividades que le permitan:

- ⇒ Deducir reglas y conceptos a partir de la información escrita.
- ⇒ Relacionar información con la cotidianidad de su vida diaria.
- ⇒ Utilizar claramente el esquema de los mapas conceptuales para lograr construir teorías con conceptos articulados que apoyen una definición o una regla determinada.
- ⇒ Desarrollar ejercicios de visualización abstracta, esto desarrolla gran capacidad de abstracción. Esta capacidad se determina además, en realizar talleres de competencia lógica y racional compleja, donde el estudiante adquirirá habilidades para resolver eficientemente algoritmos de mayor nivel en su estructuración.

¿Qué caracteriza a los que tienen dones matemáticos? De acuerdo con Adler, rara vez los poderes de los matemáticos se extienden mas allá de la frontera de la disciplina. Rara vez los matemáticos tienen talento para las finanzas o el derecho. Lo que caracteriza al individuo es su amor por trabajar por la abstracción. (Gardner, 1999)

Según Gardner, es evidente que el aprendizaje, es diferente en los seres humanos, que por lo tanto es absurdo que sigamos insistiendo en que todos nuestros alumnos aprendan de la misma manera.

Descubrir sus inteligencias, entender cuales tienen más desarrolladas y buscar soluciones a través de estas, son parte del conocimiento de sí mismo para poder elaborar un comprometido proyecto de vida. (Mascwitz, 2008)

Cada persona tiene por lo menos siete inteligencias, cada una con un desenvolvimiento propio y distinto, en el cual intervienen los elementos de la dotación biológica del individuo, de su interacción con el mundo circundante y por supuesto, la valoración cultural que recibe de su experiencia personal. Estas inteligencias se combinan, se entrecruzan y las usamos en diversas formas e intensidades, pero siempre de una manera personal y única. Habrá además que desarrollar un nuevo concepto y sistema de evaluación. No podemos seguir evaluando a la persona multinteligente a través de una única inteligencia. El ser humano es mucho más completo y complejo. Hoy lo sabemos.

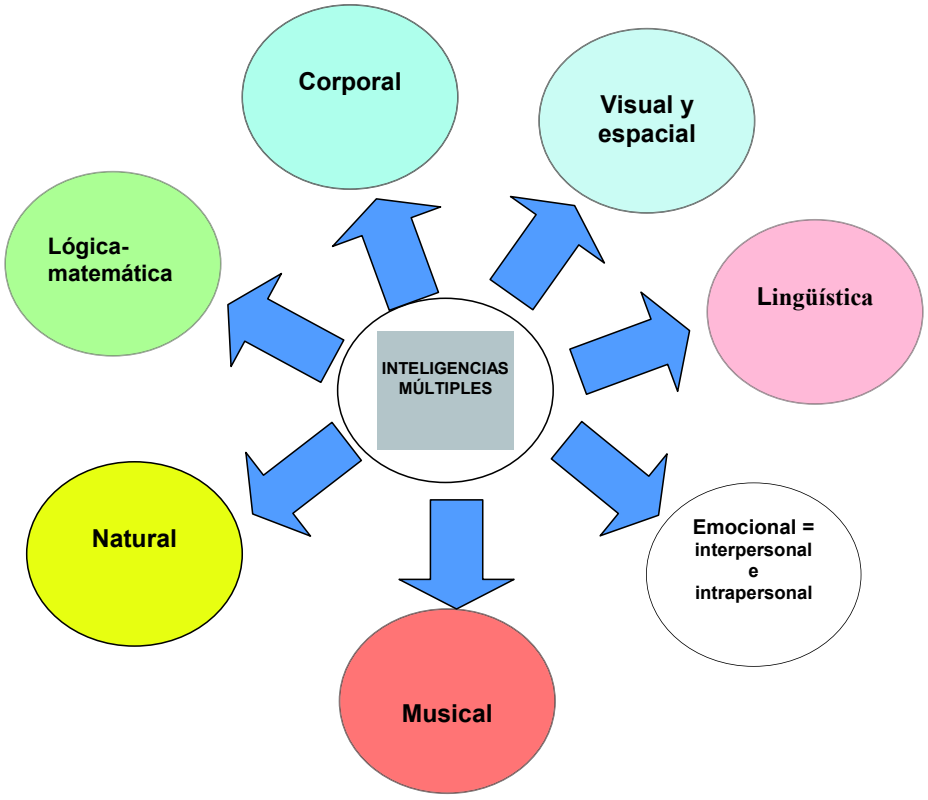


Figura 1. Las siete inteligencias del individuo

Como en toda tarea, existen diferentes pasos a seguir para transformar el proceso de aprendizaje, lo primero es aprender la teoría de inteligencias múltiples Es imprescindible que los docentes sean voluntarios en este proceso de cambio. En forma general habrá que seleccionar y capacitar a los actores involucrados en el proceso de aprendizaje en la asignatura de algoritmos que entiendan que para la enseñanza de ésta, los alumnos deben trabajar con conceptos abstractos o argumentaciones de carácter complejos y además realizar esquemas y relaciones lógicas, desarrollar afirmaciones y proposiciones.

De igual manera por parte de los docentes se debe:
⇒ Desarrollar estrategias didácticas
⇒ Desarrollo de nuevos métodos de evaluación.
Una vez planteada una propuesta didáctica y evaluativa en el proceso de aprendizaje, el estudiante de algoritmos deberá desarrollar la inteligencia lógico matemática con actividades de confrontación con los objetos y con la capacidad de una cuantificación de los objetos que va desde lo concreto hasta lo abstracto. Esto indica que debe desarrollar una dimensión mental de utilizar los números en forma efectiva para razonar en forma lógica. En otras palabras el docente debe propender comenzar a estimular acciones pedagógicas desde un modelo sociocritico en cuanto a la capacidad de razonar con altos niveles de abstracción, resolución de situaciones de gran complejidad matemática y establecer hipótesis desde un método inductivo como estrategia de enseñanza aprendizaje.

Dicho proceso subyace en la capacidad del estudiante en percibir patrones lógicos y las relaciones que se establecen con ellos, desarrollando la posibilidad de realización de aseveraciones y proposiciones; relaciones abstractas con patrones abstractos como contar de dos en dos, o hacer cálculos cotidianos en forma rápida y exacta, hacer conexiones o relaciones entre trozos de información aparentemente desconectados o diferentes.

El desarrollo de las habilidades del pensamiento lógico-matemático en la cátedra de Algoritmos es in-

quietante en la medida que no todos los estudiantes aprenden a desarrollar un nivel de abstracción que le permita garantizar la capacidad de razonar para resolver problemas en un orden lógico. La enseñanza de los Algoritmos está exigiendo nuevas respuestas al Curriculum ya que impone la revisión profunda en todos sus componentes, entre estos: los contenidos temáticos, las estrategias de enseñanza y los estilos de aprendizaje de los alumnos, con el fin de diseñar estrategias que den respuestas a las exigencias de los mismos.

En este orden de ideas, la enseñanza de los algoritmos se propone como un proceso en el cual el alumno construye tejidos lógicos y niveles de abstracción a través del despliegue de su actividad cognoscitiva.

En efecto, a pesar de los numerosos intentos por mejorar el rendimiento académico de los alumnos en dicho módulo, se evidencia hoy por hoy una creciente limitación al momento de identificar hacia donde se debe orientar las potencialidades de los alumnos y cual es su estilo de aprendizaje para adquirir lo que se pretende. Es así entonces, que se observa con preocupación como un alumno que tiene un elevado potencial de las habilidades manuales (precisión óculomanual, coordinación y planeamiento en el espacio), estudia una carrera de corte técnico que exige de él un razonamiento abstracto.

Esto es necesario atenderlo si se considera que el Modelo Instruccional que se ha venido empleando para la enseñanza de los algoritmos, ya no responde a los requerimientos de construcción de los alumnos, ni al perfil del ingeniero y del tecnólogo que se pretende formar. Hasta ahora, el profesor de forma magistral explica problemas modelos, luego entrega una guía o recomienda un taller para que el alumno lo resuelva; de allí, de todos los ejercicios el profesor, en muchos casos coloca una evaluación extraída de una bibliografía desconocida por el alumno. Evidentemente, las circunstancias actuales deben orientar al docente hacia la búsqueda de nuevos modelos que garanticen un verdadero aprendizaje, alejado de la desmotivación y la frustración que representa para el alumno el hecho de que su esfuerzo intelectual sea infructuoso.

Basándose en esto, se hace necesario planificar y aplicar estrategias de aprendizaje apoyado en la inteligencia asociativa, racional e inductiva, que atienda a los intereses de los estudiantes.

Conclusiones

Si bien puede haber algunas temas importantes alrededor de las inteligencias múltiples en materia de educación, tal como lo ha planteado Howard Gardner, estas temáticas han ayudado a un número significativo de educadores a la pregunta de su quehacer docente para animarles a mirar más allá de los estrechos límites de los discursos arbitrarios de la cualificación, el currículo y evaluación , esto permitiendo establecer dinámicas que accedan definir estrategias de enseñanza aprendizaje acorde a la forma como aprenden y en qué momento lo hacen los estudiantes. Específicamente en el área del aprendizaje para desarrollar algoritmos es importante determinar que las prácticas pedagógicas estén orientadas a desarrollar la inteligencia lógico matemática a partir de:

- análisis de problemáticas asociadas a situaciones reales;
- desarrollo de la inducción y la deducción de acuerdo una información previa;
- construcción de mapas conceptuales con el fin de construir reglas determinadas y asociadas a una información;
- desarrollo permanente de ejercicios abstractos, realizando talleres que permitan ejercitar análisis complejos para llegar a soluciones efectivas;
- talleres lógico-matemáticos que ejerciten la dimensión mental con el fin de utilizar los números en forma efectiva; y
- retroalimentación entre los estudiantes y el docente, con el fin de que los profesores obtengan información valiosa sobre los análisis que hagan los estudiantes sobre situaciones específicas.

De igual manera para desarrollar destrezas para el análisis, el razonamiento lógico y la abstracción es fundamental, desde una concepción pedagógica y de conocimiento, que los estudiantes:

- desarrollen procedimientos de cálculo en múltiples pasos;
- Propongan soluciones a un problema matemático;
- interpreten el significado de las operaciones y las relaciones entre las operaciones;
- propongan métodos de solución ante ciertas situaciones problémicas;
- realicen tareas de cálculo a la luz de la lógica matemática pero insertadas en contextos reales;

- adopten una actitud propositiva e intuitiva en el aula para dar sentido a ideas lógicas con base en las matemáticas; esto con el fin de desarrollar las capacidades de abstracción para dar solución a problemas en los modelos o prototipos propios de la construcción de algoritmos; y
- desarrollen talleres, por su importancia para la solución de problemas no cotidianos, que inquietan al aprendiz en cuanto al qué y al cómo aplicar sus conocimientos a partir del análisis.

Referencias

Bayon, M. I., & Saldaña, M. A. (1999). *Proyecto de Inteligencia "Hardvard".* Madrid: CEPE.

Gardner, H. (1999). *Estructura de la Mente La teoria de las Inteligencias Multiples.* New York: Fondo de Cultura Económica.

Mascwitz, E. M. (2008). *Inteligencias Multiples en la Educacion de la Persona.* Buenos Aires: Cooperativa Editorial Magisterio.

Samper, J. d. (2002). *Teorias Comtemporáneas de la Inteligencia y la excepcionalidad.* Bogota: Editorial El Magisterio.

Umbral de problemas en ingeniería de sistemas y programas afines: una visión desde lo cotidiano

Problem threshold in systems engineering and related programs: a vision from the everyday life

Ricardo Botero Tabares
Candidato a Magíster en Ingeniería (Área Sistemas y Computación)
Especialista en Didáctica Universitaria
Ingeniero de Sistemas
Profesor Asociado Facultad de Informática
Tecnológico de Antioquia – Institución Universitaria
rbotero@tdea.edu.co

Resumen
Este artículo, producto de una revisión de tema, narra el origen de algunos problemas típicos en ingeniería de sistemas, ingeniería de software y programas afines, causados por situaciones cuotidianas que con el trascurrir del tiempo, se convierten en modelos de estudio en el proceso formativo de los noveles ingenieros.
Palabras clave: problemas típicos en ingeniería, estudio de la ingeniería.

Abstract
This article, resulting from a subject review, traces back the origin of some typical problems caused by everyday situations in systems engineering, software engineering and related programs, those programs become as time goes by, models to study in the junior engineer learning process.
Keywords: typical problems in engineering, engineering study.

Recibido: 11 de junio 2011
Aprobado: 26 de octubre 2011

Introducción

Estudiar programas académicos relacionados con ciencias de la computación o áreas afines conlleva el encuentro con currículos que incluyen asignaturas como lógica de programación, estructura de datos, sistemas operativos, análisis y diseño orientado a objetos, teoría de redes, entre otras, en las cuales ciertos temas de tinte estrictamente científico se asemejan a hechos reales, en ocasiones rutinarios, hogareños, citadinos o idiosincrásicos, que generan situaciones problemáticas solucionadas en gran parte mediante el uso de los modelos matemáticos inherentes a la investigación de operaciones, las matemáticas discretas, la geometría analítica, la física o el cálculo diferencial e integral. Esto convierte al estudio, momentáneamente, en un juego capcioso que requiere del ingenio de las personas en búsqueda de una solución óptima, haciendo de lo cotidiano una eclosión de descubrimientos que pueden verterse en nuevos desarrollos científicos.

Son célebres los casos de Newton y el árbol de manzano para el análisis de la fuerza gravitacional; de Colón observando naos en Génova o Cádiz para deducir la esfericidad terráquea antes oteada por Copérnico y Galileo; de Arquímedes limpio y ufano en los baños públicos de la antigua Roma con su *eureka* por el cálculo de un volumen cesáreo; de Darwin y su teoría expuesta en la obra sobre el origen de las especies por medio de la selección natural (Darwin, 2006); de Palamedes o los persas o los chinos por el gallardo juego del ajedrez, sólo por citar algunos casos.

Ocupémonos ahora de situaciones no tan perogrulladas, por tratarse de casos relacionados con las ciencias informáticas, enfatizando en su origen y expresando su solución sin los diamantinos formalismos matemáticos.

Problemas célebres

Cuando se estudia una determinada área del conocimiento es común tratar casos o problemas que no se han originado en un laboratorio, en un centro de investigación o en algún entorno tecnológico o científico, sino que se han producido por algún hecho o necesidad del diario vivir. Por ejemplo, desde que el ser humano emergió en el planeta ha enfrentado el conflicto, tanto, que al ser parte natural de nuestra vida se han ideado formas de solución desde las más primitivas hasta las más elaboradas, dando origen a la teoría del conflicto social (Romero Gálvez, s.f.), un tema desarrollado en sociología, pero aplicable a la economía y al derecho. En medicina, el tratamiento de pacientes con sepsis severa de origen abdominal se puede tratar con la técnica del "abdomen abierto", ideada en 1984 en el Hospital San Juan de Dios de Bogotá, cuando se cubrió un defecto en la pared abdominal de un paciente sometido a varias intervenciones, con una lámina plástica (polivinilo), hoy conocida en la literatura internacional como "bolsa de Bogotá" y en nuestro medio como "bolsa de Borráez" (Borráez, s.f.).

Al estudiar ingeniería, y en concreto ingeniería de sistemas y programas asociados, se presentan una serie de problemas cuyo origen parte de un caso de la vida real, como se expone en los cuatro problemas siguientes.

Acomodación de discos en las Torres de Hanoi

De niños, nuestros padres posiblemente adquirieron este juego, con el ánimo de avivar el razonamiento espacial de sus párvulos herederos. El origen del juego de las torres de Hanoi se remonta a la Europa del siglo XVIII, donde hace su aparición como un material sin duda apócrifo, en el cual se explicaba que el juego representaba una tarea que estaba realizándose en el templo de Brahma, inmerso en algún recóndito lugar del Valle del Ganges (Kruse, 1988). Según dicho material, en el momento de la creación del mundo los sacerdotes brahmanes recibieron una plataforma de bronce sobre la cual había tres agujas de diamante (origen, auxiliar y destino), según se observa en la figura 1; en la primera aguja estaban apilados sesenta y cuatro discos de oro puro, cada uno ligeramente menor que el ubicado debajo. Una versión menos exótica se distribuía en la Europa de aquel tiempo: constaba de ocho discos de cartón y tres postes de palo.

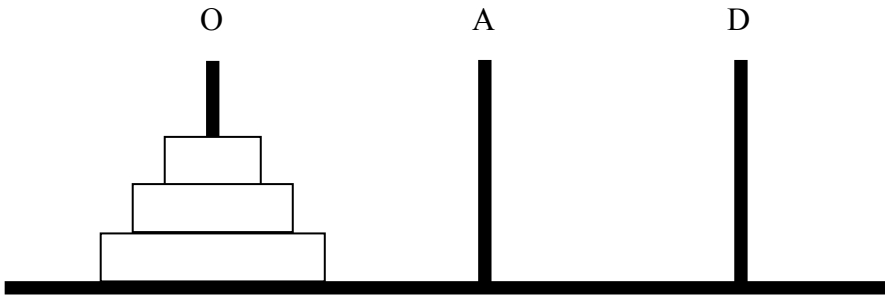


Figura 1. Discos y agujas de las torres de Hanoi, para n = 3

A los sacerdotes brahmanes se les encomendó la tarea de pasar todos los discos de la primera aguja a la tercera, bajo dos condiciones: sólo puede moverse un disco a la vez y ningún disco podrá ponerse encima de otro más pequeño. Se dijo a los sacerdotes que, cuando hubieran terminado de mover los sesenta y cuatro dorados discos, llegaría el fin del mundo.

La solución recursiva planteada en Kruse (1988) no se concentra en el primer paso (mover el disco de la cima hacia la aguja A o D), sino en el paso más difícil: mover el disco del fondo.

Sintetizando los pasos que se deben seguir para culminar la tarea, los sacerdotes deben proceder a:

1. Mover sesenta y tres discos desde la aguja O a la aguja A, usando la aguja D como almacenamiento intermedio.
2. Mover un disco de la aguja O a la aguja D.
3. Mover sesenta y tres discos desde la aguja A hasta la aguja D, usando la aguja O como almacenamiento intermedio.

La idea anterior se repite para los sesenta y tres discos restantes, hasta llegar al movimiento trivial de un solo disco.

Suponiendo que un sacerdote realiza el traslado de un disco a la frenética velocidad de un segundo, se requieren de 2^{64} -1 movimientos. Si hay cerca de 3.2 x 10^7 segundos en un año, la tarea total tardará unos 5 x 10^{11} años. Los astrónomos estiman la edad del universo en diez mil millones de años (10^{10}). Por tanto, según esta historia, el mundo durará cincuenta veces más de lo que tiene de existencia.

Recorrido por los puentes de Königsberg

Múltiples problemas en matemáticas, computación, investigación de operaciones, medicina y otras áreas, conducen a planteamientos que requieren el uso de grafos dirigidos y no dirigidos. Un grafo es una figura amorfa que consta de un conjunto de nodos o vértices conectados entre sí por un conjunto de aristas, tramos o arcos, tal como lo ilustra la figura 2, donde los vértices se representan mediante círculos y los tramos por segmentos de línea.

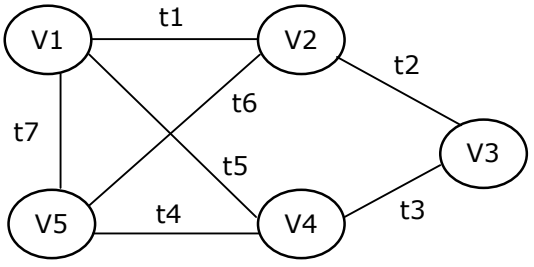


Figura 2. Grafo no dirigido

Su imaginación lo puede transportar a la variabilidad de aplicaciones de esta estructura: sistema de transporte donde los nodos son ciudades y los arcos carreteras o líneas aéreas; red de comunicación donde los vértices son servidores y las tramos flujos de datos; camión repartidor que debe visitar distintos puntos de una ciudad —vértices— para dejar su producto en determinado lapso de tiempo —tramos—; ga-

soducto donde los vértices son domicilios y los tramos la tubería; cada neurona de un ser vivo haría las veces de nodo y los mensajes entre ellas serian las aristas. De manera similar, se podría mencionar una amplia gama de casos adicionales.

El iniciador del trabajo con grafos para el modelamiento de problemas fue el matemático suizo Leonard Euler, a quien en 1736 le propusieron determinar una manera de recorrer exactamente una vez cada uno de los siete puentes que atraviesa la ciudad de Königsberg (hoy Kaliningrado, Rusia), terminando en el mismo punto de partida. Los puntos se encontraban dispuestos sobre el río Pregal como se muestra en la figura 3.

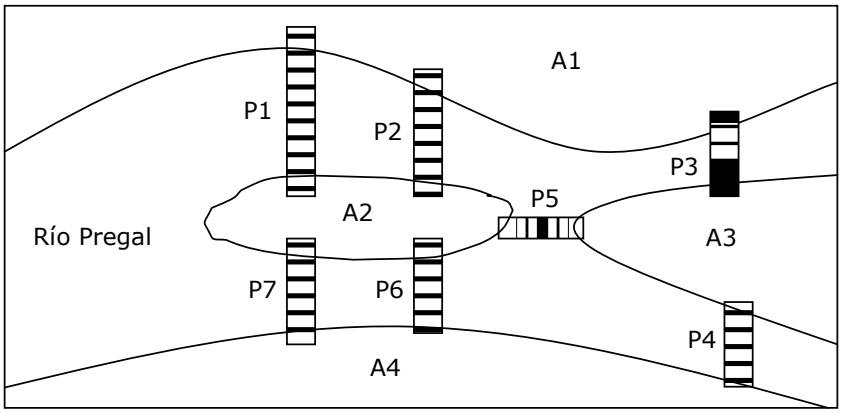


Figura 3. Problema de los puentes de Königsberg

Euler demostró que no era posible encontrar dicha ruta, fundamentado en que para cada sector de la ciudad confluye un número impar de puentes. El modelamiento de Euler consistió en un grafo, donde los sectores o áreas A1, A2, A3 y A4 son los nodos y los puentes p1 a p7 constituyen las aristas (ver figura 4).

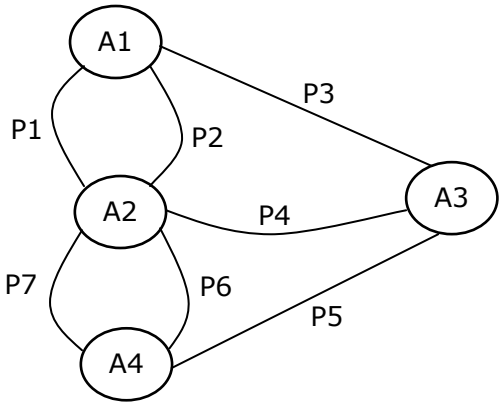


Figura 4. Modelo de Euler para el problema de los puentes de Königsberg

Penetración de sistemas informáticos en base a gusanos, virus y caballos de Troya

Gran parte del género humano siente aversión por la especie espeluznante de invertebrados llamados gusanos. Según el contexto, estos animales pueden ser benéficos o perjudiciales al hombre; *verbi gratia*, la sanguijuela es un anélido que puede causar anemia al ganado cuando no se le controla, pero en medicina es de gran utilidad para sangrías locales. Las larvas de los insectos lepidópteros causan estragos en la agricultura, no obstante en su hábitat natural forman parte de la conservación del ecosistema. Por otro lado, las picaduras de algunos de estos contráctiles seres pueden ser dolorosas, a veces letales, pero las utilidades manifiestas de la lombriz de tierra y el gusano de seda son evidentes para el bienestar de los seres humanos. Así mismo, hay gusanos planos parásitos como las duelas y las tenias que viven en el intestino del hospedero. También hay gusanos de vida libre como la acuática y carnívora planaria.

El parasitismo vegetal se manifiesta en fanerógamas, hongos, bacterias y virus. Los virus son el origen de múltiples enfermedades en el hombre (viruela, rabia, fiebre tifoidea, poliomielitis, escarlatina, tifus exantemático), en animales (brucelosis) y en plantas. Incluso virus de laboratorio son encubiertos al mundo para proyectarlos como armas biológicas.

Dejando a un lado los gusanos y parásitos, recordemos que Homero en su Ilíada describe la epopeya de la toma de Troya por los soldados griegos escondidos en el vientre de un caballo de madera gigantesco, que los ciudadanos troyanos confundieron con una ecuestre dádiva del enemigo (Homero, 2010).

Estos aspectos biológicos e históricos se extienden al estudio de los mecanismos de seguridad y protección para los sistemas informáticos. El empleo creciente de computadores en las actividades de comercio, gobierno y milicia, reverbera a diario en grandes transacciones electrónicas de fondos, notificaciones de propiedad intelectual, datos comerciales estratégicos, registros sobre individuos e innúmeros sucesos adicionales que pueden ser revelados sin autorización o destruidos por usuarios ilegítimos.

Existen varios mecanismos mediante los cuales se puede intentar la penetración de un sistema computarizado, entre ellos, los gusanos, los virus informáticos y los caballos de Troya (Milencovic, 1994).

Un gusano informático es un programa que puede invadir las computadoras, generalmente a través de una red, y denegar servicio a los usuarios legítimos utilizando cantidades desproporcionadas de recursos de procesamiento y comunicación para su auto propagación. El nombre fue acuñado como analogía del gusano solitaria, criatura parásita que vive en los intestinos de los seres humanos y otros vertebrados infectados. Los primeros gusanos informáticos eran benignos, pues fueron diseñados como mecanismos para extender grandes computaciones a través de una red de computadores, con el fin de hacer uso del tiempo de máquina inactivo. Sin embargo, como sus homónimos, los gusanos informáticos pueden convertirse en parásitos cuando la especie prolifera.

Son bien conocidos ciertos casos de gusanos artificiales que han conseguido extenderse a través de redes públicas, llegando a infectar a miles de computadores *ipso facto*. La línea habitual de ataque son las listas de correo de usuarios que contienen los nombres y direcciones de otras máquinas alcanzables; tras obtener acceso a una lista de correo, el gusano envía copias de sí mismo a algunas o todas las máquinas listadas, replicando el proceso de propagación sobre las copias recientes. La infección se extiende con rapidez y ahoga el sistema entero al consumir tiempo de proceso y ancho de banda de la red, de tal modo que se impide la realización de todo trabajo o comunicación.

Las salvaguardias frente a los gusanos informáticos están dadas por programas que refuerzan la seguridad del sistema, especialmente en las áreas de utilidades de correo, contraseñas y control de acceso a las listas, y por puntos de comprobación en el sistema de comunicación que eviten la propagación de los gusanos.

Los virus informáticos son trozos de código que infectan otros programas, generando eliminación de archivos o corrupción del bloque de arranque de un disco. Un virus informático, como su homónimo biológico, no puede funcionar por sí mismo, sino que debe introducirse en un programa anfitrión. Tras infectar una máquina o programa, puede permanecer aletargado durante un lapso de tiempo para evitar sospechas. La infección se extiende habitualmente a través de la adquisición de programas obtenidos directa o indirectamente de redes informáticas y boletines electrónicos en forma de programas de dominio público, aunque se tiene noticia de casos en los cuales se han adquirido programas corrompidos a través de software comercial legítimo, como resultado de la contaminación de algunos distribuidores y vendedores de software.

A diferencia de los gusanos, los virus no se propagan por sí mismos a través de las redes; por lo general se extienden del programa portador infectado que se ejecuta en diversas máquinas. La propagación de virus es posible en cualquier computador programable, sobre todo en computadoras personales con sistemas operativos desprovistos de protección debido su instalación ilegal. No hay una vacuna general conocida para los virus informáticos. Las medidas preventivas comunes incluyen el filtrado preventivo de todo software de reciente adquisición, las copias de respaldo frecuentes y una combinación de utilidades, tales como comprobadores de integridad, programas de vigilancia y supresores de virus.

Un programa caballo de Troya es aquel que puede ocultar intencionalmente parte de su funcionalidad,

con frecuencia dañina, buscando transmitir datos o derechos de acceso del usuario a otro que hace las veces de intruso (Milencovic, 1994). Un sencillo programa caballo de Troya se puede usar para robar contraseñas de usuario imitando al programa legítimo de apertura de sesión y reproduciendo fielmente la secuencia y diálogos de presentación normal. Estos programas son fáciles de implantar en sistemas cuyas terminales se hallan en recintos públicos, dejando una copia activa en un terminal y haciendo que simule las pantallas de presentación y despedida. Otras versiones sofisticadas de esta tipología de programas anulan totalmente la utilidad que están suplantando.

Orden en la mesa de los filósofos comensales

Las computadoras modernas pueden ejecutar varias tareas al mismo tiempo, lo que da origen en el ámbito de los sistemas operativos al término multitarea. Si reemplazamos el vocablo tarea por proceso, surge el término multiproceso, y si consideramos un proceso como un programa de computadora en ejecución, entonces multitarea se convierte en multiprogramación.

La multiprogramación realmente no se presenta, porque, en rigor, en cualquier instante de tiempo la unidad central de procesamiento (CPU) está ejecutando sólo una tarea. La ilusión de paralelismo se presenta por la rapidez de ejecución de las instrucciones.

Un caso patético de paralelismo se presenta cuando se espera la apertura de una página web desde internet y se decide abrir un archivo de hoja electrónica, para ser impreso de inmediato; en éste y otros casos, el sistema operativo se verá obligado a tramitar una comunicación entre procesos y a solucionar todos los impasses que se puedan presentar para finiquitar con éxito todas las tareas pendientes.

La literatura sobre sistemas operativos está repleta de interesantes problemas clásicos de comunicación entre procesos, uno de ellos conocido como el problema de los filósofos comensales (Tanenbaum, 1988). El planteamiento de la situación data de 1965, cuando Edsger Dijkstra formula y soluciona el problema de manera sencilla y peculiar, imaginando a cinco filósofos sentados en torno a una mesa circular, dispuestos a ingerir cinco platos de delicioso espagueti singularmente resbaladizo, tanto, que un filósofo necesita de dos tenedores para poder comer. Cada filósofo debe controlar su voraz apetito pues sólo se cuenta con igual número de tenedores ubicados entre cada plato, tal como lo ilustra la figura 5.

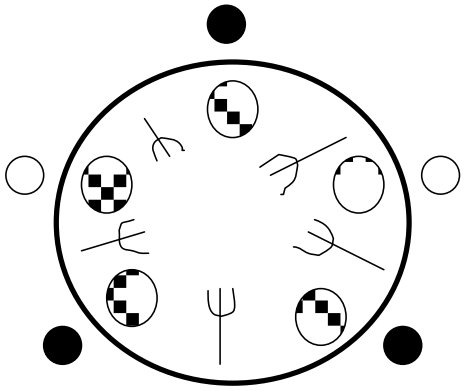


Figura 5. Vista superior del quíntuplo de platos, tenedores y filósofos, dos de ellos rapados o posiblemente calvos

El problema conduce a imaginar comensales italianos, pero también se puede pensar en matemáticos árabes comiendo kebab, o vegetarianos hindúes consumiendo Maha Prasada, o místicos orientales disfrutando el arroz chino —aquí los tenedores se convierten en palitos—, o alguna otra situación gastronómica casual del avatar terreno.

Prosiguiendo con el problema, la vida de un filósofo consta de periodos alternos de comer y pensar, supuesto en tanto insulso pero necesario en el esquema. Cuando alguien siente hambre, intenta asir dos tenedores, el de su izquierda y el de su derecha, en cualquier orden. Si logra tomar ambos tenedores, engulle unos bocados, baja los cubiertos y sigue filosofando.

La solución obvia e inmediata para un número N de filósofos, tomando la variable i como el i-ésimo comensal, es la siguiente:

```
problema_filósofos (N) {
        repita para cada filósofo i = 1, 2, ..., N
        filósofo (i)
        fin_ciclo_repita
}

filósofo(i) {
        repita mientras exista apetito {
           El filósofo i está meditando
           Tomar tenedor de la izquierda
           Tomar tenedor de la derecha
           Disfrutar del espagueti
           Colocar tenedor izquierdo sobre la mesa
           Colocar tenedor derecho sobre la mesa
        }
}
```

La solución anterior por lo obvia es equivocada: si los cinco filósofos toman los tenedores a la vez, ninguno de ellos podrá tomar el tenedor derecho y habrá estancamiento.

Conclusión

El estudio de la ingeniería de sistemas, la ingeniería en software, la ingeniería informática y demás titulaciones relacionadas, así como otras áreas del conocimiento, conllevan problemas que planteados desde su origen cotidiano, hacen del estudio algo más agradable, con menos traumatismo cognitivo, para conducir en conjunto a formalismos tecnológicos y científicos que cualifican el proceso formativo del futuro profesional.

Referencias

Borráez, Oswaldo (s.f.) Abdomen abierto. Utilización del polivinilo. ***Revista de Cirugía. Estudios analíticos.*** [documento en html]. Consultado el 3 de junio de 2011 en: http://www.encolombia.com/medicina/cirugia/cirugia16101-abdomen.htm

Darwin, Charles (2006). *El origen de las especies.* Prólogo de Faustino Cordón. Madrid: Editorial EDAF.

Homero. (2010). *La Ilíada. Clásicos de Grecia y Roma.* Madrid: Alianza Editorial.

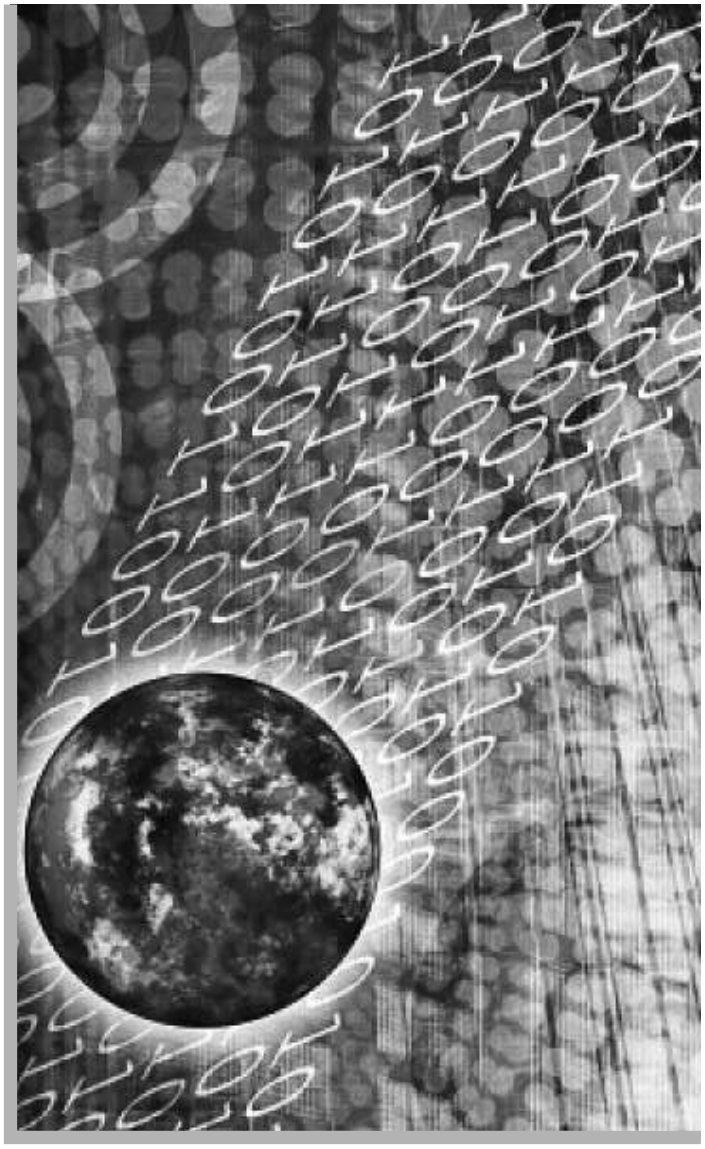
Kruse, Robert L. (1988). *Estructuras de datos y diseño de programas.* Naucalpán de Juárez, Prentice-Hall Hispanoamericana.

Milencovic, Milan. (1994). *Sistemas operativos*. Madrid, McGraw-Hill.

Romero Gálvez, Antonio (s.f.). *Teoría del conflicto social.* Consultado el 3 de junio de 2011 en: http://www.gestiopolis.com/recursos4/docs/ger/tenegouno.htm

Tanenbaum, Andrew S. (1988). Sistemas operativos: diseño e implementación. Prentice-Hall Hispanoamericana, Naucalpan de Juárez, México.

Villalobos, Jorge A. (1996). *Diseño y manejo de estructuras de datos en C.* Santafé de Bogotá, McGraw-Hill.



Los números vistos de una manera fácil

Numbers the easy way

Juan Pablo Jiménez Benjumea
Ingeniero mecánico
Especialista en Gerencia de Proyectos
Email: jeanzull20@hotmail.com

Resumen
¿Cuando te encuentras con una ecuación te asustas? o ¿si ves un fraccionario te espantas? ¿Qué hacer en estos casos?; este artículo te mostrará una forma fácil de entender las matemáticas desde la simplicidad de las cosas y lo básico de sus soluciones.
Palabras clave: matemáticas, álgebra, geometría, trigonometría.

Abstract
Do you get scared in front of an equation?; or does a fractional gets you frightened? What to do in these cases? This article will show you how to understand mathematics from the simplicity of things and the basics of their solutions.
Keywords: math, algebra, geometry, trigonometry.

Recibido: 11 de junio 2011
Aprobado: 26 de octubre 2011

Introducción

A continuación, se analizará un método que puede ser útil al momento de encontrarse con una ecuación aritmética o lineal, una ecuación polinómica de segundo grado o parábola y, en general, cualquier tipo de ecuación. Además, entenderá el porqué de los números y la importancia de no verlos simplemente como la unión de curvas y segmentos, sino más bien desde el punto de vista físico.

La historia

Remontémonos siglos atrás, a los humanos nos esperaban los números al alcance de nuestras manos. Allí estaban y no lo sabíamos. Los dedos fueron la primera herramienta que se utilizó para entenderlos y darles sentido a las cosas que se encontraban en el espacio que habitábamos; los números pasaron de las manos a tablillas de arcilla, papiros o cortezas de árboles. Empezamos a aislar los números en grupos de familias de objetos semejantes; así, posiblemente el número cinco se aisló de estos grupos, pensando que nuestros dedos se terminaban allí, pero no nuestra mente y sentidos. Las manos ayudaron a entender la cantidad de objetos en nuestro entorno y a su vez nos confundieron, pues no había palabras ni dedos para contar o identificar más objetos iguales (Oñate, 2000). En varios idiomas hablamos hoy en día de diferentes tribus en África, América y Oceanía, en donde se puede desconocer el nombre de los números, pero esto no es limitante para poder identificar las cosas que nos rodean. Para hacerlo, se utilizan elementos corporales como los dedos de la mano o los pies y en ocasiones otras cosas.

En niños pequeños se puede apreciar que aprenden a contar viendo objetos iguales o de las mismas características, utilizando sus dedos para contar estos objetos. Más adelante aprenden a sumarlos y restarlos. Que no nos dé vergüenza; es preferible utilizar los dedos de las manos para contar que utilizar una calculadora para obtener un resultado tan simple.

El método

Paso uno

Analice el problema, mírelo, antes de comenzar, venza sus miedos, tómelo con calma.

Paso dos

No tome la calculadora, aún; pregúntese: ¿qué significa esto?, ¿qué quiere decirme la ecuación? Antes de comenzar piense en posibles soluciones. No importa el procedimiento que utilice, dibuje. En ocasiones veremos que dibujando el problema podremos entender mejor cómo solucionarlo.

Paso tres

Antes de comenzar a solucionar el problema, y luego de haber dibujado la situación en un papel, tenga en cuenta los siguientes aspectos:

$$AX + B = Y \quad \textbf{(a)}$$

Anote en su memoria la siguiente tabla de signos:

+	x	+	=	+
+	x	-	=	-
-	x	+	=	-
-	x	-	=	+

Figura 1. Equivalencias de signos

Y tenga en cuenta que los signos de suma, resta, multiplicación y división cambian al pasar de un lado a otro de la igualdad, es decir, del símbolo (=),

así:

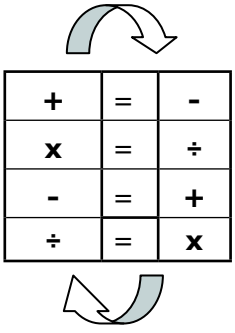


Figura 2. Cambio de los signos a ambos lados de una ecuación

Despejando X de la ecuación **(a)** se tiene:

Tabla 1. Ilustración del cambio de equivalencia a los lados de la ecuación

$AX + B = Y$	Ecuación original
$AX = Y - B$	"B" estaba sumando y pasa a restar.
$X = \frac{(Y - B)}{A}$	Se quiere obtener el valor "X", y "A" nos estorba. Recuerde que "A" multiplica a "X"; por ende pasa a dividir.
$X = S/n$	Se obtiene la solución de "X".

Paso cuatro

No pretenda solucionar el problema de una vez; considere los aspectos iniciales de los cálculos realizados, tome otra hoja y verifique posibles soluciones a las que ya planteó. En este proceso NO piense en la calculadora: eso hace que el cerebro se bloquee.

Paso quinto

Las unidades que acompañan los números son muy importantes: no las olvide. Recuerde que estas le indican de qué está hablando —metros, centímetros, milímetros, pulgadas, pies, kilos, libras, gramos, horas, minutos, segundos, newtons, kilonewtons, pascales, megapascales, etc.—. En el resultado del problema o su solución coloque siempre las unidades y, por favor, sea ordenado; esto le evitará perderse en los números.

Paso sexto

Recuerde el álgebra de bachillerato: le será útil para desarrollar sus problemas. Piense en esta como una herramienta útil para solucionar ecuaciones y factorizar problemas complejos:

En la tabla 2 y en la figura 3, se resumen algunas herramientas que podemos utilizar para resolver problemas. Tenga presente que deberá consultar e investigar nuevas soluciones y procedimientos.

Tabla 2. Resumen operaciones matemáticas básicas

	Propiedad	Ecuación	Solución
Muy Básico	Asociativa	(a+b)+c	a+(b+c)
	Conmutativa	(a+b)+(c+d)	(a+c)+(b+d)
	Distributiva	a*(c+d)	(a*c)+(a*d)
Básico	Potenciación	a ^ 2	a x a
	Radicación	√a ^ 2	a
	Logaritmos	Ln (X * Y)	Ln X + Ln Y
		Ln (X / Y)	Ln X - Ln Y
		Ln X ^ n	n * Ln X, donde n ∈ Re.
Muy Fácil	Trigonometría (Ver figura Inferior)	sen θ	a/h
		cos θ	b/h
		tan θ	a/b
		Cot θ	b/a
		Sec θ	h/b
		Csc θ	h/a
	Geometría euclidiana (ver figura inferior)	Teorema ángulos interiores	θ1+θ2+θ3=180°
		Consulte más teoremas	
	Geometría analítica (ver figura inferior)	Alternos internos	3=6 y 4=5
		Alternos externos	1=8 y 2=7
		Opuestos por el vértice	1=4, 2=3, 5=8, 7=6
		Consulte más teoremas	
Fácil	Algo más (ver figura inferior)	Ley del seno	Sen 1/b = Sen 2/a = Sen 3/c
		Ley del coseno	a ^ 2 = b ^ 2 + c ^ 2 - 2b*c(Cos θ 2)
			b ^ 2 = a ^ 2 + c ^ 2 - 2a*c(Cos θ 1)
			c ^ 2 = a ^ 2 + b ^ 2 - 2a*b(Cos θ 3)

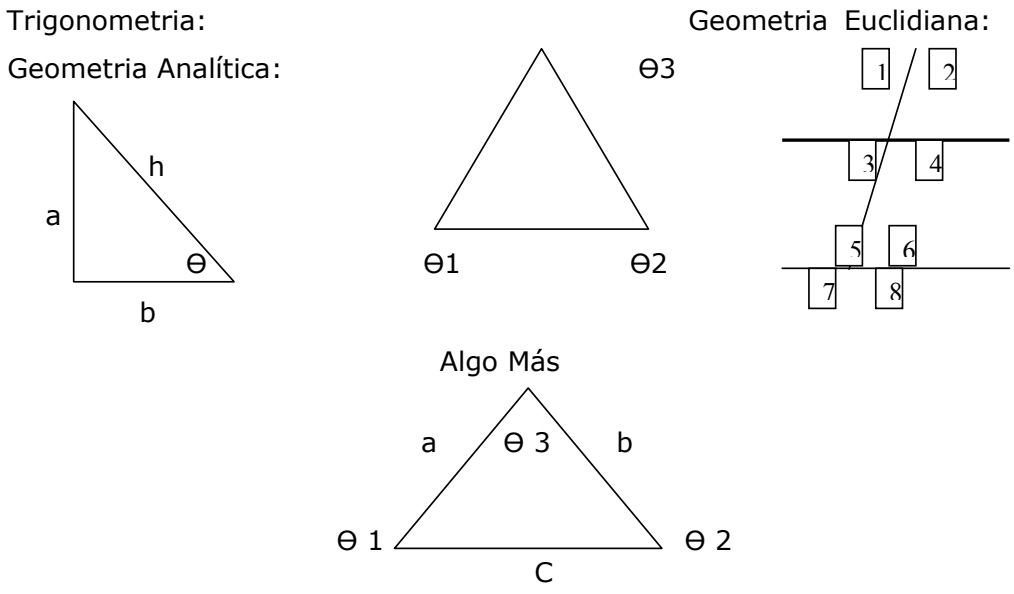


Figura 3. Ilustración operaciones matemáticas complejas

Paso séptimo

Si ya obtuvo la solución o las soluciones a su problema, busque su sentido; la habilidad para interpretar sus cálculos dependerá en buena medida de su experiencia. Hay diferentes maneras en que puede interpretar un número: una de ellas, que puede serle útil a la hora de interpretar su resultado, es la siguiente: amplíe o reduzca el número resuelto o su solución; esto le ayudará a obviar razones o a darse cuenta de su interpretación.

Utilice también un dibujo o una gráfica, recurra al sentido visual. Este le ayudará a interpretar las soluciones o los resultados. Y recuerde siempre, un número sólo tiene sentido cuando se compara con otro número.

Paso octavo

Lea nuevamente el paso No. 1.

Paso noveno

Si la confusión continúa llame a su psicólogo, pues el problema no son las matemáticas sino su miedo para enfrentarlas. Pídale que le ayude a superar sus miedos.

Paso décimo

Confíe en usted y vuelva a comenzar.

Conclusiones

Los números forman parte de nuestra existencia, pues nos permiten entender el entorno en el que habitamos. De la misma forma, las matemáticas deben ser vistas como un medio de solución a diferentes problemas algebraicos; debemos investigar nuevas formas de encontrar resultados y analizar de manera práctica y didáctica los resultados obtenidos.

En nuestro entorno se ven los números como una dificultad u obstáculo para la educación; en ocasiones, los estudiantes se encuentran limitados para entenderlos, e indican que los mismos les causan dificultades para el desarrollo de sus objetivos. Los números deben ser enseñados de manera práctica y didáctica, utilizando nuevos métodos y motivando a los estudiantes a ver su aprendizaje y utilidad.

Referencias

Ball, W.W. Rouse & Coxeter, H.S.M. (1987). *Mathematical recreations and essays*. Nueva York: Dover Books.

Campos Laclaustra, Javier. (2001). *Estructuras de datos y algoritmos.* Zaragoza: Universidad de Zaragoza.

Dedekind, Richard. (1998). ¿Qué son y para qué sirven los números? Madrid: Alianza Editorial.

Gardner, Martin. (1961). *Nuevos pasatiempos matemáticos.* Madrid: Alianza Editorial.

Mirta Rosenberg. (1986). *Matemáticas para divertirse.* Barcelona: Granica, trad.: Mirta Rosenberg.

Oñate, E. (2000). *El bucle de los números.* Centro internacional de métodos numéricos de ingeniería —Cimne—. Publicación No. 192, septiembre; Barcelona.

Globos virtuales frente a sistemas de información geográfica: un análisis descriptivo de las herramientas y formatos soportados

Virtual gobles vs. geographic information systems: a descriptive analysis of supported tools and formats

María Isabel Marín Morales,
Candidata a Magíster en Ingeniería de Sistemas, Universidad Nacional de Colombia.
mimarinm@unal.edu.co
Investigadora de la Escuela de Sistemas y de la Escuela de Geociencias y Medio Ambiente, Universidad Nacional de Colombia.

Resumen

Los datos obtenidos por medio de satélites, radares, sensores remotos, estaciones de medición y en general instrumentos de captura de datos geo-referenciados requieren herramientas eficientes para su almacenamiento y gestión. Esta necesidad se presenta debido al incremento de estos datos en la última década. Actualmente, dos grupos de sistemas gestionan la información geo-referenciada: a) los sistemas de información geográfica (SIG) y b) los globos virtuales. La elección del SIG o globo virtual adecuado para un proyecto específico de análisis o toma de decisiones permite potencializar el uso de los datos. Por lo anterior, en este artículo se presenta un paralelo entre globos virtuales y sistemas de información geográfica en términos de herramientas de procesamiento y análisis, y los formatos soportados para el ingreso y salida de los datos. Adicionalmente, se realiza un análisis descriptivo de los SIG y globos virtuales encontrados y se plantean algunas sugerencias sobre los usos de cada uno.

Palabras clave: globos virtuales, modelo de datos ráster, modelo de datos vector, sistema de información geográfica.

Recibido: 11 de junio 2011
Aprobado: 26 de octubre 2011

Abstract

Data obtained from satellites, radars, remote sensors, measuring stations and, generally speaking, instruments to capture geo-referenced data require efficient tools to be storaged and managed. This need arises from the increase of these data in the last decade. Currently, two groups of systems manage geo-referenced information: a) geographic information systems (GIS) and b) virtual globes. Choosing the right GIS or virtual globe for any given project of analysis and/or decision-making may leverage the use of data. This is why we are presenting a parallel between globes and GIS in terms of processing and analysis tools, and formats supported for data input and output in this paper. Additionally, a descriptive analysis of GIS and virtual globes is made, and their uses are suggested.

Keywords: virtual globes, raster data model, vector data model, geographic information system.

Introducción

El acceso a la información geo-referenciada se facilita por la masificación de los instrumentos que la captura como los satélites, los radares, los sensores remotos, las estaciones de medición, entre otros. El incremento en la cantidad de estos datos en las bases de datos de corporaciones regionales y empresas requiere que su gestión se realice de una manera eficiente (Yoon & Hyeong, 2004). Esto, es con el fin de optimizar el espacio en disco de las fuentes de datos espaciales; el acceso, búsqueda y recuperación al interior de estas; y su despliegue, procesamiento y pos-procesamiento en los sistemas destinados para ello (Chesnaux *et al.*, 2011).

Actualmente, los sistemas de información geográfica (SIG) y los globos virtuales se presentan como las dos alternativas principales para la gestión de datos geo-referenciados. Los SIG presentan un visor para los mapas en dos dimensiones, de manera que permiten representar solamente las porciones de la tierra requeridas. Por otro lado, los globos parten de un modelo tridimensional de la tierra para realizar sobre esta la visualización o análisis requerido (Zhang, Zhao & Li, 2010).

Los sistemas tienen ventajas y desventajas en su uso según el tipo de procesamiento que se va a realizar y según los formatos de almacenamientos con los que se trabaja. Por ello es importante conocer las bondades de cada tipo de sistemas para realizar una correcta elección en el proceso de análisis y toma de decisiones en un proyecto dado.

Por lo anterior, en este artículo se presenta un paralelo entre globos virtuales y sistemas de información geográfica en términos de herramientas de procesamiento y análisis, y los formatos soportados para el ingreso y salida de los datos. Adicionalmente, se realiza un análisis descriptivo de los SIG y globos virtuales encontrados y se plantean algunas sugerencias sobre los usos de cada uno.

Este artículo se estructura de la siguiente manera: en la sección 2 se expone el marco teórico en donde se acerca al lector a los conceptos de información geo-referenciada, SIG y globos virtuales; en la sección 3 se realiza el análisis descriptivo de los principales SIG y globos virtuales encontrados en la literatura y se hacen algunas recomendaciones sobre sus usos; y finalmente en la sección 6 se presentan las conclusiones y el trabajo futuro que se puede derivar de este trabajo.

Marco teórico

Información geo-referenciada

La información geo-referenciada corresponde a cualquier dato alfanumérico o binario que tenga asociado una coordenada geográfica con la que define su localización en el espacio terrestre. La geo-referenciación se realiza con base en un sistema de coordenadas y un datum (Yahya *et al.*, 2010). La información almacenada en cualquier fuente de datos, independiente del domino, se puede enriquecer agregándole la coordenada geográfica del lugar en el espacio al que representa. Sin embargo, el vínculo entre los datos y la referencia geográfica normalmente está dado directamente mediante su obtención a través de instrumentos de posicionamiento global como los satélites, los radares, sensores remotos, entre otros (Li *et al.*, 2002).

Sistemas de información geográfica

Los SIG posibilitan la captura, el almacenamiento, el procesamiento y el despliegue de información georeferenciada. Cobraron auge en la última década, gracias a la masificación de los instrumentos de captura de esta información. Permiten solucionar problemas de planificación y gestión en muchas áreas del conocimiento, tales como: sociología, geografía histórica, cartografía, *marketing*, hidrología, climatología, hidráulica, entre muchas otras (Elangovan, 2006). En la figura 1 se presenta el aspecto tradicional de un SIG.

Los datos en un SIG pueden gestionarse a partir de dos modelos de datos básicos: el modelo vector y el modelo *raster*. El primero representa los datos a través de primitivas geométricas como puntos, líneas y polígonos. Están pensados para almacenar información discreta en el espacio, como casas, calles, cuerpos de agua, entre otros. En estos modelos, la información se almacena en tablas de atributos en donde cada tupla está asociada a una geometría específica. Por otro lado, los modelos de datos *raster* son matrices de datos en donde cada celda tiene un valor representativo de una región. La región es definida por un rectángulo dx-dy, que especifica la resolución espacial del mapa. Este modelo está pensado para la gestión de información continua en el espacio como la precipitación, la temperatura, la evapotranspiración, entre otras (Bolstad, 2005).

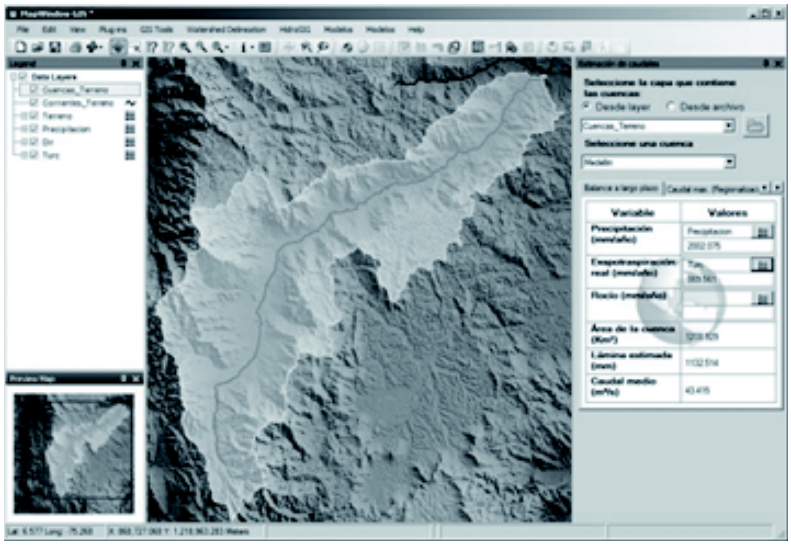


Figura 1. Aspecto tradicional de un SIG. Tomado del manual de usuario de HidroSIG 4.0 sobre MapWindow 4.6 (HidroSIG, 2011).

Globos virtuales

Los globos virtuales son representaciones de la tierra en 3D. Ofrecen al usuario la capacidad de moverse en todo el entorno virtual cambiando el ángulo de visión y la posición. Permiten superponer vistas predeterminadas de la tierra, tales como: el relieve, los ríos, las calles, entre otros (Bailey & Chen, 2011). En la figura 2 se presenta el aspecto tradicional de un globo virtual.

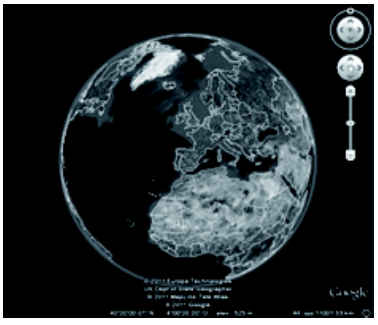


Figura 2. Aspecto de un globo virtual. Imagen tomada por los autores del globo virtual Google Earth.

Comparación y análisis

En esta sección se comparan y analizan descriptivamente los dos grupos gestores de información georeferenciada mencionados (SIG y globos virtuales). La comparación se hace inicialmente al interior de cada grupo y al final en términos generales entre los dos grupos.

Sistemas de información geográfica

Para el análisis y la comparación se revisaron los SIG ArcGIS, GRASS (Geographic Resources Analysis Support System), GeoDA, GvSIG, MapWindow GIS y SAGA. Estos sistemas se eligieron, ya que son ampliamente citados en la literatura. La descripción se hizo con base en cinco variables: formatos de lectura de datos, formatos de salida de datos (ver tabla 1), herramientas (tabla 2), licencia y sistemas operativos soportados. La síntesis de la revisión se registra en la tabla 3.

Tabla 1. Convenciones para los formatos de lectura y salida de datos en los SIG

No.	Formato de lectura y salida de datos
1	ESRI Shapefiles
2	Raster ASCII
3	Imágenes (png, jpg, bmp, gif, tiff)
4	NetCDF
5	USU Binary
6	DBF
7	GML
8	ESRI HDR/BIL Images
9	WMS
10	WFS

Tabla 2. Convenciones para las herramientas en los SIG

No.	Herramienta
1	Convertir un modelo a otro
2	Proyectar mapas
3	Re-proyectar mapas
4	Remuestrear *rasters*
5	Unir *shapes*
6	Unir *Shapefiles*
7	Exportar *shapes* seleccionados a nuevo *shapefile*
8	Crear *bufer* sobre *shapes*
9	Análisis de redes
10	Análisis hidrológico
11	Procesamiento DEM
12	Conexión de GPS
13	Configuración de impresión
14	Análisis estadístico sobre *vector*
15	Análisis estadístico sobre *raster*
16	Extensible

Tabla 3. Comparativa de sistemas de información geográfica

SIG	Formatos de lectura	Formatos de salida de datos	Herramientas	Licencia	S.O.
ArcGIS	1-10	1-3, 5,6,8	1-16	Propietario	Window, Linux, Unix
GRASS	1, 3, 9, 10	1, 3, 9, 10	1-13, 16	GNU GPL	Windows, Mac OS, Linux, Unix
GeoDA	1	3	5, 7, 14, 15	GNU GPL	Windows
GvSIG	1-3, 7, 9, 10	1-3, 7, 9, 10	1-16	GNU GPL	Windows, Mac OS, Linux, Unix
MapWindow	1-3, 5, 6, 8	1-3, 5, 6, 8	1-13, 16	MPL	Windows
SAGA	1-4, 8	1-4, 8	1-16	GNU GPL	Windows, Mac OS, Linux, Unix

Las primeras ocho herramientas listadas en la tabla 1 son las más básicas que se pueden encontrar en un SIG y están orientadas a la edición de los formatos *vector* y *raster*. Las otras herramientas dotan de mayor capacidad y especialidad a los SIG. A partir de esto y con base en la tabla 3, se puede notar que ArcGIS y GvSIG presentan una mayor funcionalidad y, teniendo en cuenta que el segundo es un SIG gratuito, se sugiere este para utilizar en proyectos educativos y de gestión de bajo presupuesto. Por otro lado, ArcGIS presenta mayor versatilidad e interoperabilidad por la amplia aceptación de formatos de almacenamiento de ingreso y salida de datos; esto hace que sea un SIG opcionado para proyectos de gran envergadura.
GeoDA exhibe especialidad en los modelos de datos *vector*, lo que lo hace un SIG liviano y rápido. GRASS, MapWindow y SAGA son opciones similares a GvSIG, pues presenta menor funcionalidad. Adicionalmente, MapWindow incorpora la limitante de correr solo en sistemas Windows.

Globos virtuales

Se analizaron y revisaron los globos virtuales Nasa World Wind, Google Earth, Marble, ArcGIS Explorer, SkylineGlobe y Virtual Earth 3D, ya que son citados ampliamente en la literatura. La revisión se hizo con base en cinco variables: formatos de lectura y salida de datos (tabla 4), herramientas (tabla 5), capas (tabla 6), licencia y sistemas operativos soportados. La síntesis de la revisión se registra en la tabla 7.

Tabla 4. Convenciones para los formatos de lectura y salida de datos en los globos virtuales

No.	Formato de lectura y salida de datos
1	World Wind XML
2	KML/KMZ
3	ESRI Shapefile
4	WMS
5	WFS
6	Imágenes (jpg, png, gif, bmp)
7	GPX
8	COLLADA
9	Fly
10	Oracle
11	ArcSDE
12	GeoTiff

Tabla 5. Convenciones para las herramientas en los globos virtuales

No.	Herramienta
1	Soporte de GPS
2	Gestión de direcciones
3	Extensible
4	Búsquedas
5	Simulador de vuelo
6	Simulador de la luz solar
7	Medición de distancias
8	Generador de videos

Tabla 6. Convenciones para las capas por defecto en los globos virtuales

No.	Capas
1	Elevación del terreno
2	Batimetría del océano
3	Red hídrica
4	Calles
5	Centros poblados
6	Imágenes satelitales y aéreas
7	Edificios 3D
8	Clima
9	Topografía
10	Reporte de tráfico en tiempo real
11	Escuelas, restaurantes, hoteles

Tabla 7. Comparativa de globos virtuales

Globo Virtual	Formatos de lectura y salida de datos	Capas	Herramientas	Licencia	S.O.
NASA World Wind	1-6, 8	1-3, 6,8,9	1, 3, 4 (Ciudades), 6-8	Open-Source	Windows, Mac OS, Linux, Unix
Google Earth	2,4,6-8	1,3-11	1, 2 (muchos países), 3, 4 (Ciudades y direcciones), 5-8	Open-Source	Windows, Mac OS, Linux, Unix
Marble	2,6	4,6, 8 (nubes), 10	3,6,7	LGPL	Windows, Mac OS, Linux, Unix
ArcGIS Explorer	1-12	1, 6	2 (estados unidos), 4 (Ciudades y direcciones), 7	Propietario	Windows, Mac OS, Linux
SkylineGlobe	2-5,8-12	1, 4, 6-9,11	1, 2 (estados unidos),3, 4 (Ciudades y direcciones), 5-8	Propietario	Windows, Mac OS, Linux, Unix
Virtual Earth 3D	2,6	1, 6	2 (muchos países), 4 (Ciudades y direcciones), 7	Propietario	Windows/ Internet explorer

Los globos virtuales NASA World Wine, Google Earth y SkylineGlobe se perfilan con mayor capacidad. NASA World Wind presenta la visualización de batimetrías marinas, lo que puede ser útil para proyectos de las ciencias de la atmósfera y el mar. Por otro lado, Google Earth incorpora la mayor cantidad de direcciones, lo que lo acerca más al usuario tradicional. SkylineGlobe presenta la limitante de ser un software propietario igual que ArcGIS Explorer y Virtual Earth 3D.

Google Earth es similar a NASA World Wind, sin embargo, la resolución de las imágenes es mayor en el primero que en el segundo, y las imágenes de NASA World Wind son de dominio público mientras que las de Google Earth no.

Globos virtuales frente a los sistemas de información geográfica

Las principales diferencias que se identificaron entre los dos grupos gestores de información geo-referenciada son: a) los SIG presentan distorsión en la visualización de los mapas ya que la representación del planeta es una proyección en el plano, algo que no ocurre con los globos virtuales; b) la personalización de la información es más fácil de gestionar en un SIG, pues las capas son generadas por el usuario, mientras que en el caso de los globos virtuales, estas capas son predefinidas. Esta situación también se puede convertir en una ventaja, pues la adquisición de la información no representa un problema; c) las herramientas son diferentes, lo que orienta el uso de los SIG hacia el análisis, la gestión y planificación y el de los globos virtuales hacia la exploración. Finalmente, los globos virtuales pueden ser fuente de información para los SIG, ya que incorporan capas de información que pueden ser descargadas y usadas en un SIG. Esto es posible porque existe coincidencia entre formatos, como es el caso del ESRI Shapefile.

Conclusiones y trabajo futuro

Este artículo presenta un paralelo entre los SIG y los globos virtuales, así como una descripción de los principales programas encontrados en cada grupo. Esto permitió identificar algunos usos recomendados. La funcionalidad de los SIG está más enfocada hacia el análisis, la planificación y la gestión, es decir, están orientados hacia el soporte de proyectos concretos. Por otro lado los globos virtuales se recomiendan para la exploración. Este uso puede ser potencializado por empresas que requieran la utilización diaria de direcciones como es el caso de las agencias de envíos o domicilios. Adicionalmente, los globos virtuales pueden ser fuentes de información para los SIG, pues incorporan datos por defecto, tra-

dicionalmente obtenidos mediante satélites y sensores remotos.

Como trabajo futuro se plantean las siguientes líneas de investigación:

- Agregar al análisis descriptivo otros parámetros como: fuentes de datos, soporte de estándares, nivel de interoperabilidad, entre otros.
- Explorar otros gestores de información georeferenciada como es el caso de los servidores de mapas, en donde se clasifican Google maps, Yahoo maps, Bing Maps, MapQuest, OpenStreetMap, entre otros. Estos servicios no están sujetos a sistemas operativos ya que corren sobre navegadores web.

Referencias

Bailey, J.E. & Chen, A. (2011). The role of virtual globes in geoscience. *Computers & Geosciences*, Vol. 37, No. 1, pp. 1-2.

Bolstad, P. (2005). *GIS fundamentals: a first text on geographic information systems.* 2a. ed. White Bear Lake, MN: Eider Press, 543 pp.

Chesnaux, R., Lambert, M., Walter, J., Fillastre, U., Hay, M., Rouleau, M., Daigneault, R., Moisan, A. & Germaneau, D. (2011). Building a geodatabase for mapping hydrogeological features and 3D modeling of groundwater systems: Application to the Saguenay-Lac-St.-Jean region Canada. *Computers & Geosciences*, En imprenta.

Elangovan, K. (2006). *GIS: fundamentals, applications and implementations*. New India Publishing Agency, Nueva Delhi, 208 pp.

HidroSIG home page. http://www.minas.medellin.unal.edu.co/~hidrosig/ [Consultado 1 de Julio de 2011].

Li, L., Im, E., Connor, L.N. & Chang, P.S. (2002). Detecting ocean surface winds using TRMM precipitation radar. *Geoscience and Remote Sensing Symposium, 2002. IGARSS '02. 2002 IEEE International*, Vol. 2, pp. 748-750.

Yahya, A., Karnadi, M.S., Bohari, S.N. & Suldi, A.M. (2010). Evaluating GPS for datum transfer in hydrography. *Signal Processing and its Applications (CSPA), 2010 6th International Colloquium*, pp.1-4.

Yoon, C. & Hyeong, P. (2004). Development of a web-based geographic information system for the management of borehole and geological data. *Computers & Geosciences*, Vol. 30, No. 8, pp. 887-897.

Zhang, Y., Zhao, H. & Li, H. (2010). Three-dimensional terrain visualization based on 3S technology. *Computer Application and System Modeling (ICCASM), 2010 International Conference,* Vol. 6, pp. V6-534-V6-537.

La automatización neumática en procesos industriales

Pneumatic automation in industrial processes

Tomás de Jesús Moreno Murillo
Ingeniero metalúrgico. Universidad Libre
e-mail:tomasmoreno1951@yahoo.es

Resumen

La creciente necesidad de automatizar los distintos procesos de fabricación en la industria hoy día, ha elevado la especialidad neumática hasta costos insospechados. El éxito fulgurante de esta tecnología en los últimos años se debe, sin duda alguna, a la facilidad de implantación de los sistemas manipulados con aire comprimido, a la rapidez de los movimientos de sus mecanismos, y a que en no pocos casos y en automatismos de cierta complejidad, estos sistemas son autosuficientes.
Palabras clave: automatización, industria, neumática, procesos, programación.

Abstract

The increasing need to automate the various manufacturing processes in industry today has unexpectedly raised costs of the air-operated technologies. This technology's overwhelming success in recent years is undoubtedly due to the ease of displaying compressed-air engineered systems, to their speed of movement, since, in many cases, and rather complex automation, these systems are self-sufficient.
Keywords: automation, industry, pneumatics, processes, programming.

Recibido: 11 de junio 2011
Aprobado: 26 de octubre 2011

Introducción

A lo largo de los últimos treinta años, la mecanización cuya finalidad primera era reemplazar al hombre por la máquina para liberarlo de un trabajo físico innecesario fue dando lugar a sucesivos y significativos perfeccionamientos, basados en el empleo de dispositivos neumáticos, hidráulicos, eléctricos y electrónicos. El ordenamiento de las sucesivas operaciones de trabajo, su control, la corrección permanente de los errores cometidos, la realización de los cálculos, el tratamiento de la información, etc., han liberado al hombre de múltiples tareas, que aunque pudieran considerarse de orden intelectual, no dejaban de resultarle engorrosas e ingratas.

Etapas de la automatización

Tras una relativa corta evolución se han llegado a definir tres etapas fundamentales de la automatización industrial:

-Existe primeramente una mecanización del proceso de producción; por ejemplo, mediante máquinas transfert o trenes de máquinas transfert, y la mecanización del tratamiento de la información mediante su registro, almacenamiento, transformación, presentación.

-A continuación se pasa a la automatización propiamente dicha, valiéndose de dispositivos automáticos de control y de gobierno mediante servomecanismos autocorrectores y autorreguladores, y la introducción de material que permita abordar los problemas de programación y de gestión.

-Se llega finalmente a la optimización de los procesos de producción y de administración con la ayuda del calculador electrónico que permite:

el ordenamiento de las operaciones a partir de ecuaciones matemáticas (mecanizado con máquinas herramienta de mando numérico); el análisis de las propiedades de una operación; la utilización de las características de la información (multiprogramación, problemas complejos de gestión y de previsión). La automatización afecta casi todas las actividades; sí, las industrias y la administración de empresas se valen de automatismos para asegurar su competitividad. También en vida diaria, utiliza el hombre cada vez mayor número de automatismos; en consecuencia, hoy conocemos la existencia de distribuidores automáticos, reguladores de temperatura, ascensores con memoria, la regulación automática de la iluminación, el guiado automático de automóviles, la automatización de las tareas de distribución y facturación en almacenes, etc. Y n lo que respecta a las actividades profesionales del hombre, ya desde hace años están siendo auxiliadas en grado creciente, por automatismos de todo tipo.

De todos los automatismos, los más numerosos, y que seguramente seguirán más tiempo vigentes son los relativos a las etapas 1 y 2 de la clasificación anterior, ya que se trata de los más económicos, tanto en su proyecto como en su explotación. La mayoría de los dispositivos que permiten resolver problemas o fracciones de problemas dados, pertenecen al dominio de los automatismos de secuencia. Estas técnicas simples de automatización se basan en la lógica combinatoria y/ó en la lógica secuencial.

El estudio del álgebra de Boole constituye la base científica de los automatismos. Todo dispositivo, cualquiera que sea la técnica aplicada (eléctrica, electrónica, neumática, hidráulica, oleoneumática) puede reflejarse en una ecuación. Los esquemas de Boole, que resultan de las técnicas de aplicación escogidas, nos llevan necesariamente a los esquemas de utilización que tienen en cuenta la tecnología de los aparejos.

Resumiendo, la resolución de un problema de automatización consta de tres fases esenciales:

Primera: definición precisa del estudio por realizar, enunciando las condiciones de funcionamiento y las limitaciones previsibles.

Segunda: tratamiento de la información para el desarrollo de matrices.

Tercera: confección de las matrices y de salidas para el establecimiento de ecuaciones y de esquemas que lleven a soluciones seguras y económicas.

Así como la segunda fase solo requiere un mínimo de sentido común, de orden y de precisión, la primera y la tercera aguzan la imaginación y la capacidad de análisis, al requerir tanteos para fijar las secuencias deseadas, así como capacidad de síntesis para conju-

gar armónicamente los resultados y desembocar en soluciones concretas válidas, y capacidad de decisión para determinar la técnica que debe emplearse y el dispositivo tecnológico apropiado para cada finalidad. La tecnología neumática juega un papel importante en la mecánica desde hace mucho tiempo, entre tanto, es incluida cada vez más en el desarrollo de aplicaciones automatizadas (Croser, 1991).

Funciones con la neumática

La neumática se utiliza para ejecutar las siguientes funciones:
-detección de estados mediante sensores;
-procesamiento de información mediante procesadores;
-accionamiento de actuadores mediante elementos de control;
-ejecución de trabajos mediante actuadores.

Algo de historia

El aire comprimido es una de las formas de energía más antiguas que conoce el hombre. El primero de quien se sabe con seguridad que se ocupó de la neumática (utilizó el aire comprimido como elemento de trabajo) fue el griego Ktesibios. Hace más de dos mil años, construyó una catapulta de aire comprimido (Meixner & Kobler, 1980).

De los antiguos griegos procede la expresión pneuma, que designa la respiración, el viento y, en filosofía, el alma. Como derivación de la palabra pneuma se obtuvo, entre otras cosas, el concepto de neumática, que designa los movimientos y procesos con aire. Los sistemas de aire comprimido proporcionan un movimiento controlado con el empleo de cilindros y se aplican en herramientas, válvulas de control y posicionadores, martillos neumáticos, pistolas para pintar, motores neumáticos, sistemas de empaquetado, elevadores, herramientas de impacto, prensas neumáticas, robots industriales, vibradores, frenos neumáticos, etc. (Creus, 2007)

Las ventajas que presenta el uso de la neumática son el bajo costo de sus componentes, su facilidad de diseño e implementación y el bajo par o la fuerza escasa que puede desarrollar a las bajas presiones con que trabaja (6 bar), lo que constituye un factor de seguridad. Otras características favorables son el riesgo nulo de explosión, su conversión fácil a movimientos giratorio lineal, la posibilidad de transmitir energía a grandes distancias, una construcción y mantenimiento fáciles y la economía en las aplicaciones.

Referencias

Croser, P. (1991). Neumática. Esslingen: Festo Didactic.
Creus Sole, Antonio (2007). Neumática e hidráulica. México: Alfa Omega

Meixner, H. & Kobler, R. (1980). Introducción a la neumática. Manual de estudio. Esslingen: Festo Didactic.

Serrano Nicolás, Antonio (2003). Neumática. España. Thomson-Paraninfo, 5ª. e

Matemáticas para todos: la relación de igualdad y sus implicaciones en la solución de ecuaciones

Math for everyone: the relation of equality and its implications in equation solving

Leonardo Ceballos Urrego
Magíster en Educación; Especialista en Estadística y Licenciado en Matemáticas
lceballos@tdea.edu.co; lceu0457@gmail.com
Docente de planta del Tecnológico de Antioquia

Resumen
En la didáctica de las ciencias acostumbran los profesores a utilizar estrategias de enseñanza apoyadas en la memoria, la habilidad mental o física, la experiencia del docente, etc. Al recurrir a ellas, se ocultan los procesos lógicos y las técnicas adecuadas en las que se sustentan muchos de los resultados fundamentales de las ciencias básicas. Uno de los casos más frecuentes corresponde a la enseñanza de la solución de ecuaciones elementales, cuyo fundamento está en la comprensión de las propiedades de la relación de igualdad, y el cual se sustituye por un procedimiento nemotécnico de traslación de términos que, a la postre, no genera ningún aprendizaje en el educando, y en cambio sí se convierte en un obstáculo difícil de superar a la hora de comprender todos los procesos relacionados con las ecuaciones y sus innumerables aplicaciones. Con este ensayo se pretende reorientar el camino adecuado para el aprendizaje de los conceptos engendrados en las ciencias básicas, particularmente con el tema expuesto.
Palabras clave: ecuaciones, igualdades, ley uniforme, métodos, soluciones,

Abstract
In the didactics of science, teachers are used to follow teaching strategies supported in memory, physical and mental ability, or the teacher's experience, etc. In recurring to it, logical processes and appropriate solving techniques supporting many of the essential results in the basic sciences remain hidden. One of the most common cases is the teaching of the solution to elemental equations which base is in the understanding of the equality relation, which is replaced by a mnemonic procedure of the term's transfer that doesn't generate any learning in the student but produces a hard to overcome obstacle when it comes to understand all the processes related with the equation and its many applications. The aim of this essay is to redirect the concepts produced in the basic sciences, particularity in the exposed topic.
Keywords: equations, equalities, uniform law, methods, solutions.

Recibido: 11 de junio 2011
Aprobado: 26 de octubre 2011

Introducción

Uno de los mayores obstáculos, aunque por supuesto no el único, para que la gente aprenda matemáticas básicas es la falta de comprensión y manejo de la relación de igualdad y de sus propiedades, a pesar de que desde los primeros años de la primaria la diferenciamos y permanentemente nos repiten el nombre de su principal característica: "La ley uniforme de la relación de igualdad". Con humildad, los docentes de matemáticas tenemos que aceptar que contribuimos significativamente a tal ignorancia, ya que en muchas ocasiones carecemos de la claridad conceptual y de los recursos pedagógicos necesarios para el cabal manejo de los conceptos que con frecuencia enseñamos. Con el ánimo de acercarlos al entendimiento de los procesos y propiedades que sustentan la solución de una ecuación, y de mostrarles que no hay nada especial que alguien no pueda entender, se presenta el presente ensayo.

Desarrollo

Cuando nos piden resolver una ecuación, como por ejemplo 2x-1=5, aquellos que estén familiarizados con las matemáticas seguramente dirán que basta con pasar el 1 a sumar (2x=5+1; 2x=6) y luego pasar a dividir por 2, (x=6/2) con lo que se llega al resultado x=3, el cual corresponde, efectivamente, a la solución de la ecuación. Realmente no parece que haya nada raro en el proceso y de hecho es la forma como, seguramente, a todos nos enseñaron: "Lo que está restando pasa a sumar", "lo que está multiplicando pasa a dividir", etc.

Si bien, efectivamente el procedimiento empleado conduce a la respuesta correcta, de manera rápida y cómoda, el ejercicio de la docencia nos lleva a descubrir que muchos métodos, supuestamente pedagógicos, funcionan para obtener resultados, pero su aplicación mecánica elimina u oculta elementos y procesos fundamentales para el aprendizaje de los conceptos, convirtiéndose así en el principal obstáculo para el entendimiento y la utilización adecuada de los mismos. El caso de la relación de igualdad y su conjunto de propiedades, dentro de las cuales está la ley uniforme, son un buen ejemplo de cómo los procesos nemotécnicos y mecanicistas nos impiden, desde temprana edad, *aprender* conceptos sencillos y aplicables, como la solución de ecuaciones y la demostración de identidades matemáticas.

La relación de igualdad involucra tres conceptos básicos, a saber: igualdades, ecuaciones e identidades. *Una igualdad* es una proposición de la forma a=b, de la que, según los valores numéricos que adopten a y b, se podrá afirmar si es verdadera o falsa. *Una ecuación* es una igualdad que contiene términos desconocidos, a los que se denomina variables, por ejemplo: 2x-1=5. El objetivo de las ecuaciones es hallar los valores de las variables, dentro de un conjunto numérico, que hacen que la igualdad sea una proposición verdadera; en este caso solo el valor x=3 hace que 2x-1=5, se convierta en una proposición cierta. Por otro lado, *una identidad* es una igualdad que se cumple para cualquiera de los valores que se les den a las variables que contiene; por ejemplo, en álgebra, todos los llamados "productos notables" corresponden a identidades. Así, por ejemplo, en $a^2 - b^2 = (a-b)(a+b)$ la igualdad siempre será una proposición verdadera cuando se le asignen valores a a y b, respectivamente. Para los más "viejitos", las únicas identidades que conocíamos eran las identidades trigonométricas, pues pocas veces se nos ocurrió pensar que algunos de los métodos de factorización y varias de las fórmulas que se aplican en ciencias como la física, la química, la biología, etc. correspondían a identidades o ecuaciones; eso es algo de lo que no tenemos porqué avergonzarnos ya que muy probablemente nunca, o muy pocas veces, nos hicieron hincapié en tales diferencias conceptuales.

Pero volvamos al problema de la solución de ecuaciones. Empecemos por recordar qué dice la ley uniforme de la relación de igualdad y cómo se aplica en la solución de ecuaciones, con las otras propiedades de las operaciones suma y producto usuales (clausura, asociatividad, módulos, existencia de neutros y conmutatividad)

"Ley uniforme de la relación de igualdad: Dada una igualdad cualquiera, si a ambos lados de la igualdad se suma, resta, multiplica o divide por una misma cantidad, la igualdad no se altera." (Camus-Massara; 1995)

(Debe tenerse en cuenta que la división por 0 no existe o no se define.)

Una primera aclaración tiene que ver con, lo que se ha remarcado al final de la ley, que en mi opinión de-

bería sustituirse por: *el resultado continúa siendo una igualdad;* ya que se presenta una sutileza semántica que puede traer confusión a quienes traten de entender la propiedad, ya que, por ejemplo, si a 3=2+1 le sumamos 4 a ambos lados, el resultado será 3+4= 2+1+ 4, que aunque implique que 3=3, y 7=7, no representan la misma igualdad, porque 3 es distinto de 7, pero sigue existiendo igualdad, que es, en última instancia, lo que pretende enfatizar la ley (no solo para la suma sino para todas las demás operaciones en las que la propiedad se aplica).

En segundo lugar, la potencia de la ley uniforme radica en su eficiencia para solucionar cualquiera de los tipos de ecuaciones que se puedan presentar. Retomemos el ejemplo inicial y veamos cómo con las otras propiedades de las operaciones básicas en los conjuntos numéricos (clausura, asociatividad, módulos, existencia de neutros y conmutatividad) se llega, no solo a la solución de la ecuación, sino además a entender porqué se pasa a restar, a dividir, a multiplicar, a sumar, etc. al otro lado de la ecuación: Ecuación original: 2x-1=5; sumando a ambos lados 1 (Ley uniforme), queda:

(2x-1)+1=5+1; que se transforma por asociatividad en
2x+ (-1+1)= 6; y por existencia del inverso aditivo, en:
2x+0=6; ahora, por ser el 0 el módulo de la suma, se resume en:
2x=6; finalmente, si se divide a ambos lados por 2 (que es lo mismo que multiplicar a ambos lados por 1/2); entonces queda: (2x/2)=(6/2); que simplificando se transforma en x=3.

Conclusiones

En este punto es de esperar que se piense como la mayoría de los estudiantes: ¡¡que cosa tan enredada, pero si es más fácil como lo hizo antes, es decir pase a sumar el 1 y luego a dividir el 2 y listo!!. Justamente es lo que se pretende resaltar: En *la enseñanza de conceptos básicos* la aplicación de métodos memorísticos o mecánicos conlleva, usualmente, *a la pérdida de los elementos sustanciales* para la comprensión de dichos conceptos, por lo que no resultan recomendables para ser usados por los docentes. En cambio, cuando se tiene claridad en los procesos, la aplicación sistemática de los mismos y el ejercicio continuo, se llegará indefectiblemente, no solo a comprender realmente los conceptos, sino incluso, y como resultado de la habilidad que se adquiere, a utilizar estrategias nemotécnicas y a mecanizar los procedimientos, con la diferencia de saberse qué es lo que se está haciendo.

Referencias

Buriticá T., B. (2010). Álgebra y trigonometría. Medellín: Ed. Cátedra Litografía

Camus-Massara, (1995). *Matematica*. Buenos Aires

Díez, L. H. (1988). *Matemáticas operativas.* Medellín: Tipográficas Vane.

Jiménez, B. (1963). *Aritmética.* Medellín: Bedout.

Reunión de Profesores. (1954). *Aritmética*. Medellín: Bedout.

Spitbard & Bardel. (1976). Álgebra y trigonometría. EE.UU.: Limusa.

Guía del autor

1. Sobre la revista

El Cuaderno ACTIVA de la Facultad de Informática del Tecnológico de Antioquia Institución Universitaria nació con el objetivo de brindar un espacio académico, investigativo y científico en los campos de la Informática, la Electrónica y las áreas referentes a las tecnologías de la información y comunicación. En el Cuaderno ACTIVA se busca la contribución de todos los miembros de la comunidad académica del Tecnológico de Antioquia, así como la participación de otras instituciones educativas y del sector productivo.

2. Tipología de los artículos

De acuerdo con la siguiente descripción formulada por Colciencias, los autores pueden presentar para publicación las siguientes tipologías.
Es de anotar que la Revista dará prelación para la evaluación externa y publicación los resultados de investigación, como requisito básico de indexación nacional e internacional.

Artículo de investigación científica y tecnológica. Documento que presenta, de manera detallada, *los resultados originales de proyectos terminados de investigación.*

Artículo de reflexión resultado de investigación. Documento que presenta *resultados de investigación* terminada desde una perspectiva analítica, interpretativa o crítica del autor, sobre un tema específico, recurriendo a fuentes originales.

Artículo de revisión. Documento *resultado de una investigación terminada* donde se analizan, sistematizan e integran los resultados de investigaciones publicadas o no publicadas, sobre un campo en ciencia o tecnología, con el fin de dar cuenta de los avances y las tendencias de desarrollo.

Artículo corto. Documento breve que presenta *resultados originales preliminares o parciales de una investigación científica o tecnológica*, que por lo general requieren de una pronta difusión. La estructura del artículo puede tener los siguientes cuatro apartes: *introducción, metodología, resultados, discusión, conclusiones y Bibliografía* (sólo se incluye bibliografía citada en el texto)

Reporte de caso. Documento que presenta los resultados de un estudio sobre una situación particular con el fin de dar a conocer las experiencias técnicas y metodológicas consideradas en un caso específico. Incluye una revisión sistemática comentada de la literatura sobre casos análogos.

Revisión de tema. Documento resultado de la revisión crítica de la literatura sobre un tema en particular.

3. Requisitos Artículo

Tipo de letra verdana, tamaño 12, el artículo debe ser máximo de 14 hojas.

- **Título (en español y en inglés)**
- **Datos autor(es)**
  Nombres y apellidos completos
  Formación académica
  Correo electrónico
  Filiación institucional (nombre de la entidad a la cual está vinculado)
- **Resumen y Abstract**
  Máximo 250 palabras.
- **Palabras clave y keywords**
  Se deben presentar de 3 a 5 palabras clave en estricto orden alfabético.
- **Introducción**
- **Cuerpo de trabajo**
- **Conclusiones**
- **Referencias** (sólo se incluye la bibliografía citada en el texto)

Figuras en Escala de grises, numeradas y tituladas.

Figura N°1. Productividad Inicial

Tablas y formulas numeradas y tituladas.

4. Citación

A continuación se presente un ejemplo de citación:

"La baja comprensión que tienen los analistas en relación con el discurso del dominio y el exiguo conocimiento que exhiben los interesados para entender esquemas conceptuales o representaciones sintácticas o semánticas del discurso" (Zapata & Olaya, 2007).

5. Bibliografía

A continuación se presentan ejemplos de las fuentes de información, normas APA:

LIBRO: Hopkins, K. D, Hopkins, B., & Glass, G. (1997). *Estadística básica.* México: Prentice-Hall.

ARTÍCULO: Zapata, C. M., Gelbukh, A. & Arango, F. (2006). Pre-conceptual Schemas: A Conceptual-Graph-Like Knowledge Representation for Requirements Elicitation. *Lecture Notes in Computer Science*, Vol. 4293, pp. 17–27.

EVENTO: Zapata, C. M. & Chaverra, J. (2010). *Generación automática de interfaces gráficas de usuario a partir de esquemas preconceptuales*. Memorias del Quinto Congreso Colombiano de Computación (5CCC), Cartagena.

PÁGINA: CMMI Transition Plan. http://www.sei.cmu.edu/cmmi/background/transplan.html [Consultado 10 de Agosto de 2010].

6. Carta de autorización

El autor (es) debe (n) enviar junto con el artículo y sus datos personales, una carta firmada (escaneada) en la cual certifique (n) que:

El autor o autores autorizan a la Revista y a la Institución para editar y divulgar/publicar el artículo por cualquier medio nacional y/o internacional, impreso o electrónico.

El artículo es original: no ha sido publicado, ni aceptado ni presentado para publicación en otra revista o sitio web en internet.

El artículo es original: el texto es producto de un proceso de investigación del autor y ha sido valorado previamente por colegas expertos antes de ser presentado a publicación.

El autor o autores aceptan las políticas editoriales y los lineamientos de la guía.

CORREO DE ENVÍO:
cuadernoactiva@gmail.com

Revista Activa
Tecnológico de Antioquia
ISSN 2027-8101

Tecnológico de Antioquia
Institución Universitaria

Educación sin Fronteras

CUPÓN DE SUSCRIPCIÓN Suscripción Suscripción Estudiantes Tecnologico Renovación Fecha Suscripción a partir del número
Nombre
C.C. No.
Dirección
Ciudad
País
Teléfono
E-mail
Firma
Valor suscripción anual Colombia $50.000 Fuera de Colombia US$45

Para suscriptores de Colombia enviar cheque a nombre del Tecnológico de Antioquia mediante correo certificado o consignación a la cuenta de ahorros en formato de recaudo a nombre del Tecnológico de Antioquia, cuenta corriente No. 18201017-3 Banco Popular.