Generación de requerimientos de un Objeto de Aprendizaje a partir de escenarios: un caso de estudio para un curso de Programación Inicial

Stella Maris Massa[1], Carlos Rico[1], Raquel Huapaya[1]

[1] Facultad de Ingeniería, Universidad Nacional de Mar del Plata, Argentina

smassa@fi.mdp.edu.ar, crico@crico.com.ar, huapaya@fi.mdp.edu.ar

Resumen. El siguiente trabajo describe el análisis y diseño de escenarios para un Objeto de Aprendizaje incorporado como apoyo a un Curso de Programación Inicial. Se identificaron las dificultades más frecuentes que se presentan en la enseñanza y el aprendizaje de este tipo de asignaturas y se definieron una serie de estrategias de apoyo a incorporar en el OA para mejorar su calidad pedagógica y técnica.

Palabras Clave: Objetos de Aprendizaje, Escenarios

1 Introducción

Los Escenarios son descripciones parciales del funcionamiento del sistema, que se concentran en un momento específico de la aplicación. Los Escenarios no son formales, y se los puede representar con una variedad de recursos [1].

Desde la perspectiva del Diseño Centrado en el Usuario (DCU), los escenarios han sido propuestos como descripciones detalladas del contexto, que permiten elaborar decisiones de diseño [2].

Si bien cada Escenario es una descripción parcial del comportamiento de la aplicación, ninguno es independiente del resto y cada uno tiene una relación semántica con los otros [3].

El enfoque de Leite ([4]) incluye el uso de lenguaje natural para la elicitación y construcción de Escenarios. La Tabla 1 describe un esquema de escenario según Leite.

Los escenarios se utilizan en diferentes momentos del proceso de desarrollo del software. Estimulan, por una parte, la imaginación creativa de los diseñadores, mientras que por otra proporcionan herramientas ágiles que dan soporte al razonamiento del sistema en el proceso de diseño [5]. Ayudan a escenificar problemas existentes, favoreciendo la comprensión de los mismos y sus posibles vías de solución.

Es importante que el escenario contenga la mayoría de los aspectos que directa o indirectamente intervienen durante el proceso interactivo, destacando aquellos que son claves para que su consecución futura sea posible.

Tabla 1. Esquema de escenario de Leite

Componente	Descripción
Título	Identificación del escenario.
Objetivo	Meta a ser alcanzada en el dominio de la aplicación. El escenario describe la forma de lograr el objetivo.
Contexto	Describe las acciones previas necesarias para iniciar el escenario, las precondiciones, la ubicación física y temporal.
Recursos	Identifican los objetos con los cuales los actores trabajan.
Actores	Detalla las entidades que se involucran activamente en el escenario.
Set de episodios	Cada episodio representa una acción realizada por un actor, donde participan otros actores y se utilizan recursos. Los episodios se ejecutan secuencialmente. Un episodio también puede referenciar a un escenario. Se incluyen restricciones del escenario o episodio según corresponda.
Excepciones	Menciona los casos de excepción, que pueden corresponder a otros escenarios.
Dudas:	Puntos pendientes a clarificar con el usuario.

El estándar ISO/IEC 9126-1 [6] señala que al aplicar un modelo de calidad resulta prácticamente imposible definir (para después evaluar) todos los posibles escenarios de un sistema, y tendremos, por tanto, que definir aquel grupo de escenarios que mejor represente la globalidad del sistema. El estándar explica que la evaluación de los escenarios es la mejor manera de evaluar la calidad de uso del software aunque no especifica ninguna técnica para realizarla.

En el caso de los Objetos de Aprendizaje (OA), el término "escenario" se utiliza para indicar una secuencia discreta de pasos dentro de un sistema de gestión del aprendizaje o de otro tipo de aprendizaje relacionada con el sistema [7]. En particular, [8] definen “Escenario de Aprendizaje” a la secuencia de pasos en una actividad de aprendizaje con un fin pedagógico especifico, el cual comprende un conjunto de reglas a seguir para que los estudiantes puedan adquirir, reforzar o asimilar conocimientos dentro de un marco de estudio**.**

2 Resultados

El caso de estudio que presentamos corresponde a un OA para un Curso de Programación inicial de la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata. Las carreras que se dictan son “No -informáticas”

El primer paso para abordar el problema de la enseñanza de la Programación en la Ingeniería es analizar qué debe aportar la informática a la formación de un ingeniero. [9] realizó un estudio sobre la contribución de la informática al perfil de un ingeniero. Dicho perfil es descrito a través de un conjunto de atributos, tales como conocimiento técnico, madurez, capacidad de planteamiento y resolución de problemas entre otras. Estos atributos [9] los agrupa en tres campos: formación profesional, desarrollo personal y desarrollo académico. La contribución de la Informática se realiza a tres niveles:

1. Desarrollo de competencias en áreas específicas tales como: aprender un determinado lenguaje de Programación o herramientas de software. También contribuirá al desarrollo de habilidades necesarias para apoyar su aprendizaje en otras áreas (como el uso de herramientas de modelado en el diseño de sistemas, hojas de cálculo, etc)
2. Desarrollo de competencias relacionadas con la comprensión de clases generales de herramientas informáticas, sin atarse a productos específicos.
3. Conocer las aplicaciones informáticas de su campo profesional.

Por otra parte, el objetivo de un curso de Programación no es únicamente que el estudiante aprenda a escribir un programa. En forma transversal se puede contribuir a la generación de otras competencias: entender un problema (abstraer, modelar, analizar), plantear soluciones efectivas (reflexionar sobre una abstracción, definir estrategias, seguir un proceso, aplicar una metodología, descomponer en subproblemas), a manejar lenguajes para expresar una solución (codificar, entender y respetar una sintaxis), a utilizar herramientas que entiendan esos lenguajes (programar, compilar, ejecutar, depurar), a probar que la solución sea válida (entender el concepto de corrección y de prueba), a justificar las decisiones tomadas (medir, argumentar), etc [10]. Estas son competencias genéricas con las que debe contar cualquier profesional en Ingeniería tal lo señala el Consejo Federal de Decanos de Ingeniería (CONFEDI) en su documento “Competencias Genéricas – Desarrollo de Competencias en la Enseñanza de la Ingeniería [11], el Proyecto Tuning [12] y el Accreditation Board of Engineering and Technology [13].

La educación en Ingeniería es un campo con fuertes tradiciones, pero diversos ejemplos de modificación de enfoques tradicionales a otros que utilizan estrategias pedagógicas que involucran a los estudiantes como agentes de generación de su propio conocimiento, han evidenciado la necesidad de cambio y la fuente de posibles soluciones frente a dificultades en la enseñanza de Ingeniería[14].

2.1 Planificación del OA

a) Caracterización de los usuarios finales (profesores y estudiantes)

Si se pretende caracterizar al docente en los espacios universitarios generalmente se lo asocia con la imagen del profesor como transmisor del conocimiento, preocupado por brindar información, esclarecer y explicar los contenidos presentados y estableciendo estrategias que le permitan evaluar desde el control que las competencias alcanzadas por los estudiantes sean las adecuadas. En este contexto el estudiante no tiene más que un rol pasivo, el proceso de aprendizaje se desarrolla en

forma individual y los docentes tienen la certeza que se realiza el proceso de enseñanza tal como estaba estipulada y queda la inquietud sobre lo que sucede con el aprendizaje.

Con la experiencia de muchos años en el dictado de Cursos de Programación Inicial, coincidimos con: el informe CUPI2 [10], [15], [16], [17], [18] y [19] en cuanto a la caracterización de los profesores y estudiantes señaladas por los autores. Las conclusiones arribadas son las siguientes:

- Los cursos típicos de Programación están muy guiados por el eje asociado con el lenguaje.Esto hace que el curso esté basado en un recorrido de las estructuras sintácticas del lenguaje y los demás aspectos que son parte de la labor de programar, se ven de manera lateral, o sencillamente se ignoran.
- Estructurar un curso en base a un lenguaje particular hace que en muchos casos el estudiante no adquiere una visión real de lo que es programar, dándole más importancia a los elementos del lenguaje que al proceso de construir un programa.
- Al final del cursado, el estudiante tiene la impresión de que aprendió un lenguaje, pero no es consciente ni valora realmente las competencias que generó, siente que está perdiendo el tiempo.
- Los novatos pasan poco tiempo en la planificación y las pruebas del código, y cuando es necesario, tratan de corregir sus programas con pequeñas correcciones locales en lugar de ir más fondo, es decir en la reformulación de los programas.
- En un curso típico, la conceptos se presentan siguiendo un orden “de abajo hacia arriba”. Primero los elementos básicos (tipos simples, operadores, expresiones, etc.) y luego, se van introduciendo las estructuras de control, la construcción de funciones, el paso de parámetros, los vectores, las matrices, registros, etc. Sólo al final del curso el estudiante es capaz de construir un programa completo. Esta metodología en general provoca que:
  1. El estudiante no tenga una visión clara de la razón por la cual se introduce un concepto, porque no le ve una necesidad real, dificultando su proceso de aprendizaje.
  2. Muchos de los temas que se presentan resultan artificiales debido al tamaño de los problemas sobre los cuales se puede trabajar.
  3. Se necesita cubrir una gran cantidad de temas antes de poder hacer algo interesante con contexto.

b) Plataforma (posibilidades/restricciones)

Los productos incorporados en el Proyecto Integral de la Facultad de Ingeniería de la UNMDP están basados en la filosofía del software libre. Bajo este paradigma y luego de analizar las ofertas disponibles, fue seleccionada la Plataforma Educativa Moodle. El OA a desarrollar se incluirá en el Curso de Computación de dicha plataforma.

2.2 Elicitación y Especificación de requerimientos

En la Facultad de Ingeniería el curso de Introducción a la Programación (denominado Computación) es obligatorio para las 8 carreras (Eléctrica, Electromecánica, Electrónica, en Alimentos, en Materiales, Industrial y Mecánica), lo

que representa una población semestral de cerca de 300 estudiantes, repartidos en aproximadamente 5 comisiones.

A continuación se detallan cada una de técnicas utilizadas en la etapa Elicitación de Requerimientos.

a-1) Entrevistas

Se entrevistó a los integrantes de la cátedra de Computación con diferentes perfiles (profesores y ayudantes) a fin de recabar información sobre sus experiencias, opiniones y criterios referentes a la enseñanza y aprendizaje de la asignatura. En particular se preguntó cuál sería el tema que seleccionarían para el desarrollo del OA.

Consideraron que se podrían abordar temas claves como: estructuras de control, estructuras de datos, subprogramas, programación modular.

Por otra parte de estos temas claves algunos de ellos atraviesan muchos conceptos y métodos fundamentales como es el caso de los subprogramas:

a) aplicar el método de resolución de problemas sobre todo referido a la etapa de diseño del algoritmo (descomposición, refinamiento)

b) Dar valor a la reusabilidad en búsqueda de la eficiencia en la escritura el código.

c) Se avanza sobre la necesidad de mayor abstracción procedural introduciendo conceptos claves como funciones y procedimientos como la forma más adecuada de concebir los problemas en el paradigma estructurado.

d) Se definen y caracterizan los objetos claves de la comunicación entre los subprogramas: los parámetros, variables locales y globales.

a-2) Indagación Contextual

Se realizó un análisis contextual de las tareas que realizan los usuarios involucrados (profesores y estudiantes).

Los procesos de intervención pedagógica se realizan en tres modalidades semanalmente: Clase magistral utilizando recursos como pizarrón y medios audiovisuales (2 hs) , Trabajo en laboratorio/taller de computadoras para diseñar, editar, compilar, ejecutar y depurar programas (3 hs) y Taller-Grupo operativo (resolución de problemas mediante el análisis y diseño de programas) (1 hora)

Los estudiantes concurren a los laboratorios de computación en dónde realizan las siguientes actividades: Diseño de algoritmos en papel, Codificación del algoritmo en un programa fuente Pascal (en computadora), Compilación, depuración y ejecución de programas y Resolución de problemas de creciente dificultad en computadora.

Estas actividades se llevan a cabo con guías de Trabajos Prácticos con el apoyo de profesores.

a-3) Focus Group con implicados

Con toda la documentación recopilada se convocó a integrantes de la cátedra con el fin de definir el tema del OA y los lineamientos a seguir para cumplir con los objetivos. Se arribaron a las siguientes conclusiones:

- El tema seleccionado para el OA fue “Pasaje de parámetros”.
- Presentar los contenidos principales involucrados.
- Mostrar ejemplos que involucren los contenidos con diferentes medios: texto, imagen, video, audio, etc.
- Incorporar actividades de autoevaluación y grupales

b) Especificación

b.1) Definición de los objetivos del OA:

El objetivo de esta tarea es conocer por qué se realizará el desarrollo, y por lo tanto conocer qué objetivos se esperan alcanzar mediante el sistema software a desarrollar.

Se definieron los siguientes objetivos:

Objetivo General: *Resolver problemas utilizando el pasaje de parámetros*
Objetivos específicos:

- *Entender la descomposición como forma de resolución de problemas.*
- *Dar valor a la reusabilidad en búsqueda de la eficiencia en la escritura el código.*
- *Establecer comunicación entre módulos.*
- *Comprender las ventajas de la descomposición*
- *Diferenciar los distintos tipos de pasaje de parámetros.*
- *Distinguir entre variables locales y variables globales.*

b.2) Validación de la especificación en base a los requerimientos.

La validación de los requerimientos se realizó a partir de descripciones formales de escenarios en lenguaje natural que describen los requerimientos del sistema en el contexto de las especificaciones funcionales y no funcionales, mostrando cómo se efectúan los procesos de enseñanza y de aprendizaje y qué actores o perfiles de usuario intervienen en éstos a través de las secuencias de tareas descritas para cada uno de los escenarios.

Se construyeron varios Escenarios de aprendizaje para describir los requerimientos del OA. Posteriormente se realizaron nuevas reuniones con docentes de la cátedra de Computación con el fin de ampliar información sobre aquellos escenarios cuya descripción resultara incompleta y/o confusa a partir de la información obtenida.

b.3) Selección de Escenarios

A partir de la validación de requerimientos, se propone la descripción de escenarios evaluados como etapa previa al diseño del OA. La Tabla 2 ilustra uno de los Escenarios.

2.3 Protipado de la especificación

Se construyeron prototipos de baja fidelidad en papel y maquetas digitales que fueron evaluadas por integrantes de la cátedra. Para ilustrar esta etapa, se presenta en la Figura 1 uno de los prototipos en papel y en la Figura 2 una de las maquetas digitales construidas en base al escenario presentado en la Tabla 2.

Tabla **2.** Escenario 1: video explicando pasaje de parámetros por valor con variables simples

Componente	Descripción
Nombre	Video 1: ejemplo variables simples
Objetivo	Proporcionar información por diversos medios: texto , imagen y sonido sobre el pasaje de parámetros por valor utilizando variables simples.
Contexto	El usuario debe tener un cuenta Moodle del curso y una computadora con lo siguiente: • Complemento de Adobe Flash Player 10.0.22 o posterior. • Firefox 1.1 o posterior, Safari 1.0 o posterior, Google Chrome u Opera. • Conexión de banda ancha de un mínimo de 500 Kbps .
Recursos	• Video
Actores	Estudiante
Set de episodios	1. El usuario hace clik en el link que hace referencia al video ejemplo del pasaje de parámetros – variables simples. 2. Se despliega una ventana pop-up que presenta el objetivo del video y debajo el video embebido. 3. El usuario clickea en el video y comienza la reproducción. El video tiene controles para avanzar, retroceder, parar y pasar a pantalla completa. 4. El video presenta el problema sencillo “swap” a resolver que trabaja con variables simples. 5. El video tiene la pantalla dividida en a) el código fuente Pascal que resuelve el problema, b) la recreación de la memoria c) La pantalla de salida del programa. 6. El usuario observa como se ejecutan cada una de las sentencias del programa, las cuales están señaladas con un color. Simultáneamente en la otra parte de la pantalla se van mostrando los cambios en la memoria y en la pantalla de salida. 7. El usuario escucha a un locutor que explica los pasos más relevantes. 8. El usuario observa que los momentos claves son señalados además con globos explicativos. 9. Se resalta con el relato del locutor y visualmente el tipo de pasaje de parámetros por valor, en el sector de la memoria se destaca que no cambian los valores de los parámetros actuales cuando cambian los valores de los formales y que en realidad no se resuelve el problema con este tipo de pasaje de parámetros. 10. Se repiten los pasos 6 a 9 en donde el usuario tiene la libertad de “avanzar”, “retroceder” o “parar” el video. 11. El usuario presiona el botón “parar” que finaliza la reproducción del video.
Casos alternativos	Si el usuario no tiene los complementos necesarios, el sistema se lo indicará, podrá descargarlos y luego ver el video.
Dudas:	a) Enunciado del problema a resolver. b) Momentos claves a destacar visual y auditivamente en el programa Pascal. c) Metáforas para recrear la memoria

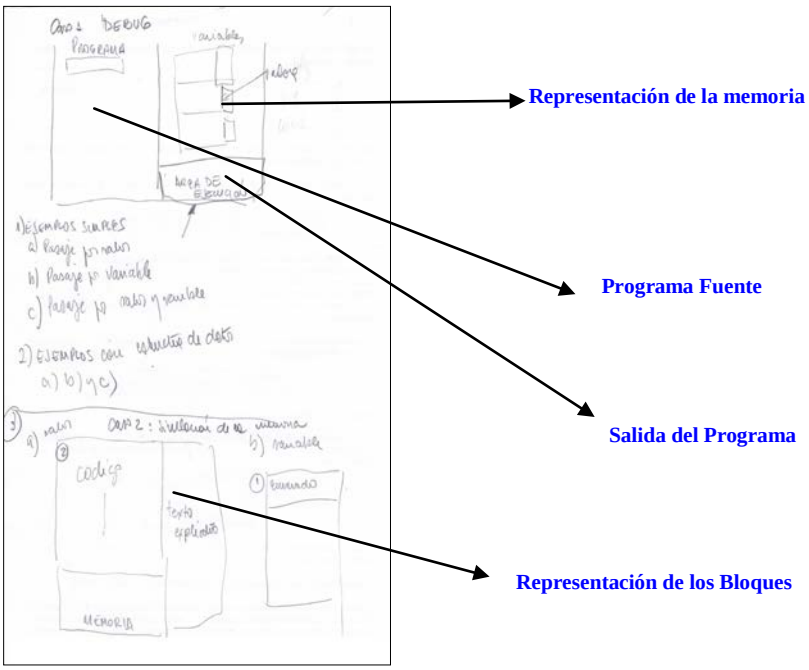

Fig. 1. Prototipo 1 en papel para videos

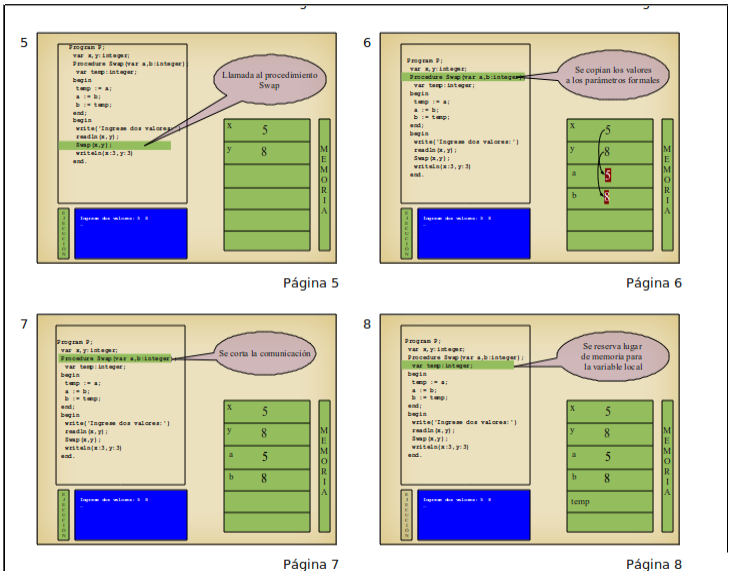

Fig.2. Maqueta digital para el video 1

Se puede observar en el prototipo en papel, algunas de las metáforas para representar la memoria, y el lugar en que se ubicarán cada uno de los elementos en el video.

En la maqueta digital de la Figura 2, se muestra como fueron plasmados los pasos 5, 6 y 8, descriptos en el escenario correspondiente al video 1 que se muestra en la Tabla 1.

3. Conclusiones y trabajo futuro

Desde la perspectiva del DCU, los escenarios; se utilizan para demostrar conceptos, informarse sobre los problemas y sus posibles soluciones y además recoger las impresiones del usuario para reflejarlas en el diseño de la interfaz.

Las actividades planteadas permitieron reflexionar acerca de las problemáticas presentes en la enseñanza y en el aprendizaje de la programación y revelaron posibles mejoras en las estrategias para superarlas.

Con este trabajo pretendemos proporcionar una metodología para el desarrollo de OA que mejore su calidad y sistematizar cada una de las actividades implicadas en su creación.

Referencias

1. Hadad G., Kaplan G., Oliveros A., Leite J.C.S.P., (1997). Construcción de Escenarios a partir del Léxico Extendido del Lenguaje. In Proceedings 26 JAIIO, Sociedad Argentina de Informática y Comunicaciones. Argentina.
2. Carroll, J. (1995). Scenario-Based Design: Envisioning Work and Technology .In Carroll, J. (ed), Introduction: The Scenario Perspective on System Development, John System Development.Wiley & Sons, New York.
3. Booch, G. (1991). Object Oriented Design with Applications, The Benjamin/Cumming Publishing Company, Inc., Redwood City.
4. Leite, J.C.S.P., Hadad, G.D.S., Doorn, J.H. & Kaplan, G.N. (2000), A Scenario Construction Process. Requirements Engineering Journal 5. Disponible en http://Fwww-di.inf.puc-rio.br/~julio/lct-pub/rej2000.pdf. Recuperado el 20 de abril de 2012.
5. Carroll, J. (2000). Making use: Scenario-based design of human-computer interactions. MIT Press.
6. ISO/IEC 9126-1 (2001) Software Engineering - Product quality - Part 1: Quality model . Disponible en http://webstore.iec.ch/preview/info_isoiec9126-1{ed1.0}en.pdf. Recuperado el 14 de febrero de 2012.
7. Sicilia M. A., Lytras Miltiadis D. (2005). Scenario-oriented reusable learning object characterizations. International Journal of Knowledge and Learning, 1,332-341. Disponible en http://www.cc.uah.es/msicilia/papers/Sicilia_IJKL_2005.pdf. Recuperado el 25 de abril de 2012.
8. Luna-Pérez, H., Mezura-Godoy, C., Benítez-Guerrero, E., García-Gaona. A. (2001). Modelado de escenarios colaborativos para e-ISO/IEC 9126-1 (2001) Software Engineering - Product quality - Part 1: Quality model learning. En Memorias del XXII Congreso Nacional y VIII Congreso Internacional de Informática y Computación (CNCIIC

2009). Ensenada, México. Disponible en http/www.somece.org.mx/ simposio/ memorias/documentos/F140.doc. Recuperado el 25 de abril de 2012.
9. Lowe, D. (2000).A skills development framework for learning computing tools in the context of engineering practice. European Journal of Engineering Education 25, 45-56. Disponible en http://services.eng.uts.edu.au/~dbl/archive/2000-Low00d.pdf. Recuperado el 19 de junio de 2012.
10. Proyecto CUPI2 (2009) . Una solución integral al problema de enseñar y aprender a programar.Universidad de los Andes. Disponible en http:// www.colombiaaprende.edu.co/ html/mediateca/1607/articles-205832_recurso_1.pdf. Recuperado el 28 de diciembre de 2011.
11. CONFEDI (2006). Desarrollo de competencias en la enseñanza de la Ingeniería Argentina. Documento del Consejo Federal de Decanos de Ingeniería. Argentina. Disponible en http:// www.confedi.org.ar/ sites/files/privado/ DESARROLLO% 20DE%20COMPETENCIAS %20GENERICAS.pdf. Recuperado el 21 de marzo de 2009.
12. Proyecto Tuning (2007). Reflexiones y perspectivas de la Educación Superior en América Latina. Informe Final. Proyecto Tuning América Latina. 2004-2007. Disponible en http://tuning.unideusto.org/tuningal/index.php?option = com_docman&task = docclick&Itemid =191&bid=54&limitstart=0&limit=5. Recuperado el 19 de junio de 2012.
13. ABET (2011).Criteria for Accrediting Engineering Programs. Engineering Accreditation Commission. Effective for Reviews During the 2012-2013 Accreditation Cycle. Disponible en http: //www.abet.org /uploadedFiles /Accreditation /Accreditation_Process/ Accreditation_Documents/Current/eac-criteria-2012-2013.pdf. Recuperado el 10 de mayo de 2012.
14. De Graff, E. & Kolmos, A. (2007) .Management of Change. Implementation of Problem-Based and Projec-Based learning in Engineering. Sense Publishers.
15. Kurland, D., Pea, R., Clement, C. & Mawby, R. (1986). A Study of the development of programming ability and thinking skills in High school students. Journal educational computing research 2(4) . Disponible en http://telearn.archives-ouvertes.fr/ docs/ 00/ 19/ 05/39/ PDF/A29_Kurland_etal_86.pdf. Recuperado el 6 de octubre de 2011.
16. McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B.,Laxer, C., Thomas, L., Utting, I. & Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. SIGCSE Bulletin, 33(4), 125-180. Disponible en http://www.deepdyve.com/lp/acm/a-multi-national-multi-institutional-study-of-assessment-of-R6JwiLwJXz?key=citeulike. Recuperado el 15 de octubre de 2011.
17. Kay, J., Barg, M., Fekete, A., Greening, T., Hollands, O., Kingston, J. & Crawford, K. (2000). Problems-based learning for foundation computer science courses. Computer Science Education, 10(2), 109-128.Disponible en http:// www.cs.usyd.edu.au/~judy/PBL /tr_cse_pbl99.pdf. Recuperado el 19 de de setiembre de 2011.
18. Fernández, J. & Millán J. (2003) .CGRAPHIC: Educational Software for Learning the Foundations of Programming. Computer Applications in Engineering Education ,11(4), 167-178. Wiley and Sons, Inc. Disponible en www.lcc.uma.es/ ~afdez/ Papers/ cae2004.pdf. Recuperado el 6 de octubre de 2011.
19. Robins, A., Rountree, J. & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. Computer Science Education, 13(2), 137-172. Disponible en home.cc.gatech.edu/csed/uploads/2/robins03.pdf. Recuperado el 20 de junio de 2012.