Aprendizaje de la programación orientada a objetos: Experiencias en educación media técnica y superior

Ricardo de J. Botero Tabares
Tecnológico de Antioquia, Medellín, Antioquia, Colombia, rbotero@tdea.edu.co

Helmuth Trefftz Gómez
Universidad Eafit, Medellín, Antioquia, Colombia, htrefftz@eafit.edu.co

RESUMEN

En el mundo de hoy el aprendizaje de los fundamentos de programación orientada a objetos (POO) para noveles estudiantes admite múltiples alternativas académicas. Este artículo expone una de ellas, consistente en la aplicación de un método con cuatro fases, combinado con un software que presenta una serie de juegos en 2D relacionados con los conceptos de clase, herencia, polimorfismo, sobrecarga de métodos y sentencias de control. Para demostrar las bondades del aprendizaje de la POO combinando las cuatro fases –que aportan los cimientos teóricos– y el juego –que motiva el aprendizaje–, se muestran los resultados cognitivos alcanzados al exponer el juego ante dos grupos experimentales, contrastados con los obtenidos ante dos grupos de control donde se prescindió del software lúdico. Estas prácticas se llevaron a cabo en un colegio con educación media técnica y una institución de educación superior, localizados respectivamente en los municipios de Envigado y Medellín, departamento de Antioquia, Colombia.

Palabras clave: aprendizaje de la programación orientada a objetos, educación media técnica en informática, juegos y programación.

ABSTRACT

In today's world, learning the basics of object-oriented programming (OOP) for novice students academic support for multiple alternatives. This article describes one of them, involving the application of a method with four stages, combined with software that presents a series of 2D games related to the concepts of class, inheritance, polymorphism, overloading methods and control statements. To demonstrate the benefits of learning OOP combining the four phases –which provide the theoretical underpinnings– and game –learning motivated–, show cognitive outcomes achieved by exposing the game to two experimental groups, contrasted with those obtained before two control groups which disregarded the recreational software. These practices are conducted in a school with technical education and higher education institution, located respectively in the municipalities of Envigado and Medellin, Antioquia department, Colombia.

Keywords: learning object-oriented programming, technical education in computer games and programming.

1. INTRODUCCIÓN

El juego ha sido usado por el hombre para efectos recreativos y de aprendizaje desde tiempos inmemoriales, tanto que hoy, los juegos digitales se están usando pedagógicamente para el aprendizaje en varias áreas del conocimiento humano, como los motores de juego para el entrenamiento clínico simulado (Marks et al, 2008). En relación con los cimientos para ciencias de la computación, la lista de juegos para aprender a programar computadoras cada día crece más. Algunos de los juegos más representativos en el mercado del software y que se están utilizando en algunas instituciones de educación media y superior nacionales y extranjeras son Scratch, Greenfoot, Robocode, Robomind, Karel, Alice y CJump.

Scratch (Resnick et al, 2009) posibilita a niños y a jóvenes la incursión en el mundo de la programación de computadoras de una manera intuitiva, por medio de un entorno multimedia donde que propicia una comprensión temprana del paradigma orientado a objetos y de las sentencias de control secuencia, decisión y ciclo. Greenfoot (Kölling and Henriksen, 2005) es un entorno integrado de desarrollo y programación que facilita la escritura de juegos y simulaciones en lenguaje Java. Robocode (Hartness, 2004) permite la simulación de guerras de robots, donde se debe programar un tanque en Java para contender en el campo de batalla contra tanques programados por otros jugadores. RoboMind (Halma, 2005) permite el aprendizaje de la programación estructurada; mediante un abanico de ordenes (agarrar un objeto, mirar, girar, etc.) se puede generar un programa que haga cosas tan variadas como buscar un punto blanco o resolver un laberinto. Karel (Pattis, 1981) es una herramienta de aprendizaje que presenta los conceptos de una forma visual, lo cual es menos abstracto que programar en un lenguaje como Pascal o C. Alice (McKenzie, 2007) es un entorno de desarrollo de software tridimensional, en donde podremos aprender a programar jugando en entornos virtuales. C-jump (Singh, 2007) es el nombre de un juego de mesa dirigido a niños, que sienta las bases para programar en lenguajes de cómputo como Java o C++.

Todos estos juegos apoyan el aprendizaje de la programación y han sido utilizados con buenos resultados en diversas instituciones de educación media y superior; sin embargo, ninguno trata de manera agrupada los conceptos de objeto, clase, herencia y polimorfismo, fundamentales para la POO. Otros juegos como *BlueJ* (Kouznetsova, 2007), *Minueto* (Denault, 2005), *Jeroo* (Dorn and Sanders, 2003), *ProBot* (Moreno y Montaño, 2005) y *Júpiter* (Chicharro et al, 2008) también han realizado aportes importantes para la enseñanza y aprendizaje de la programación, pero tampoco tratan de manera conjunta los fundamentos del paradigma orientado a los objetos. Por tales circunstancias, surge la idea de un juego denominado *CoquitoDobleO* -Coquito Orientado a Objetos- (Botero y Trefftz, 2011), que incentiva de manera directa los conceptos fundamentales del paradigma de programación implícito en su nombre.

2. MÉTODO APLICADO CON LOS GRUPOS EXPERIMENTAL Y DE CONTROL

Para efectuar los experimentos y consecuente evaluación de las características del juego *CoquitoDobleO*, se trabajó con cuatro grupos, dos de media técnica en informática y dos de educación superior discentes de Ingeniería en Software, donde se contó en igual porcentaje con grupos de control y experimental. En los experimentos se trabajó con un método para el aprendizaje de la POO, denominado MIPSOO (Botero et al, 2010), donde se proporciona la solución de problemas mediante cuatro pasos: identificación de requerimientos, diseño del diagrama de clases, especificación de responsabilidades de las clases y escritura de seudocódigo.

Además del método MIPSOO, en los grupos experimentales se trabajó con el juego *CoquitoDobleO*, buscando medir el impacto motivacional en el proceso de aprendizaje de la programación, para luego contrastar éstos resultados con los obtenidos en los grupos de control.

2.1 APRENDIZAJE DE LA PROGRAMACIÓN CON MIPSOO

Según el método MIPSOO, la docencia en fundamentos de programación dirigida a estudiantes de los grupos experimental y de control se fundamenta en la aplicación de cuatro pasos:

i) *Identificación de requerimientos*: ésta primera etapa forma parte del análisis del problema. Los requerimientos hacen referencia a las necesidades de los usuarios, es decir, identifican los aspectos que los usuarios del programa desean resolver mediante software.

Los requerimientos se especifican en una tabla compuesta por cuatro columnas: identificación del requerimiento, descripción, entradas y salidas o resultados.

ii) *Diseño del diagrama de clases*: la abstracción de clases también forma parte del análisis del problema y es el primer escaño en el diseño de la solución. Consiste en una representación gráfica del problema - plano de software-, donde se dibujan abstracciones de la realidad relacionadas con el mundo del problema, modelables con software.

iii) *Especificación de responsabilidades de las clases*: conlleva la descripción de los métodos de cada clase

mediante contratos, que incluyen los requerimientos asociados, la precondición o estado del objeto antes de ejecutar el método, la postcondición que determina el estado del objeto luego de ejecutar el método, y el modelo verbal que consiste en una descripción en lenguaje natural de la solución planteada, algo similar al denominado algoritmo cualitativo. La identificación de responsabilidades forma parte de la documentación de la solución, futuro sistema basado en software.

iv) *Escritura de seudo código*: el seudo código especifica la solución del problema en cuanto al "cómo" se logra implementar la solución, de una manera bastante cercana al proceso de codificación en un lenguaje de programación como Java o C#. Esta fase conlleva la aplicación de pruebas para cada uno de los métodos, llevadas a cabo manualmente, similar a las denominadas "pruebas de escritorio".

A manera de ejemplo, solucionemos el siguiente problema aplicando las cuatro fases implícitas en MIPSOO: "La famosa ecuación de Einstein para conversión de una masa *m* en energía, viene dada por la fórmula $E = mc^2$, donde *c* es la velocidad de la luz, *c* = 2.997925 x 1010 m/s. Leer la masa de un objeto en gramos y obtener la cantidad de energía producida en ergios." (Botero, 2010)

La solución a éste problema se presenta siguiendo las cuatro etapas planteadas:

a) Identificación de requerimientos

Se identifican dos requerimientos, descritos en la Tabla 1.

Tabla 1: Requerimientos para el problema de la ecuación de Einstein

Identificación del requerimiento	Descripción	Entradas	Salidas
R1	Conocer la masa del objeto en gramos.	Un número real digitado por el usuario.	La masa del objeto está almacenada en memoria.
R2	Calcular la cantidad de energía producida por un objeto.	La masa del objeto (en gramos).	La energía del objeto (en ergios).

b) Diseño del diagrama de clases

El diagrama de clases de la figura 1 conlleva la definición de las abstracciones *Energía* y *Proyecto*, y a la reutilización de las clases de uso común *Flujo* y *Mat*.

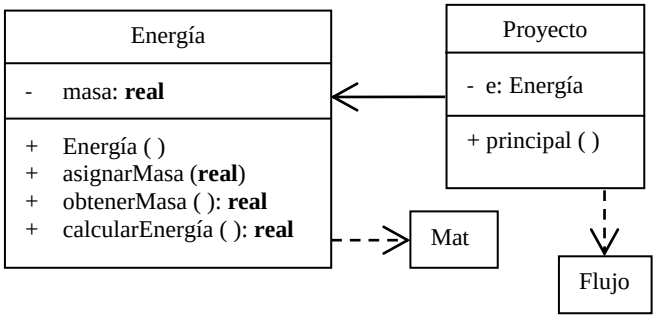

Figura 1: Diagrama de clases para el problema de la ecuación de Einstein

c) Especificación de responsabilidades de las clases

Las responsabilidades de las clases se expresan mediante los contratos de cada uno de sus métodos. En términos generales, la clase *Energía* es responsable de almacenar la masa del objeto y calcular su energía; la clase *Proyecto* es responsable de establecer comunicación con el usuario para la captura de la masa del objeto, crearlo, asignarle un estado y visualizar resultados para cumplir con los requerimientos. Las tablas 2 y 3 presentan los contratos de las clases de uso no común *Energía* y *Proyecto*.

Tabla 2: Contratos de la clase Energía

Método	Requerimiento asociado	Precondición	Postcondición
Energía	R1	No existe un objeto para calcularle su energía.	Existe un objeto en memoria listo para ser procesado.
asignarMasa	R1	El objeto tiene una masa igual a 0 (cero).	El objeto tiene una masa (número real positivo) igual a la especificada por el usuario.
obtenerMasa	R2	El objeto tiene una masa distinta de cero, desconocida para el mundo exterior al objeto.	El mundo exterior al objeto conoce la masa del objeto.
calcularEnergía	R2	Se desconoce la energía producida por el objeto.	Se conoce la energía producida por el objeto.

Tabla 3: Contrato de la clase Proyecto

Método	Requerimiento asociado	Precondición	Postcondición
principal	R1 y R2	No hay entradas de datos y no se ha efectuado proceso alguno.	Se tiene un objeto en memoria con una masa definida y una energía conocida.

Las responsabilidades de las clases de utilidad *Flujo* y *Mat*, conocidas también como clases de utilidad o de uso común, no se definen de forma explícita porque se asumen comprendidas por el analista: *Flujo* se responsabiliza de las operaciones de entrada y salida estándar y *Mat* de las operaciones con funciones matemáticas.

d) Escritura de seudocódigo

El seudo código guarda similitud con la sintaxis de lenguajes de producción como Java y Visual Basic.Net; es un buen preámbulo a la etapa de codificación en alguno de éstos u otros lenguajes. La figura 2 expone el seudo código para este problema, donde las palabras reservadas del seudolenguaje se escriben en negrita, en

contraposición a los demás identificadores para nombres de clase – *Energía* y *Proyecto* –, atributos – *masa* y *e* –, métodos –*Energía( ), asignarMasa( ), obtenerMasa( )* y *calcularEnergía( )* – , variables locales – *ener* – y objetos –*e*.

```
clase Energía
   privado real masa
   público Energía( )
   fin_metodo
   //-------------------------------
   asignarMasa(real m)
      masa = m
   fin_metodo
   //-------------------------------
   real obtenerMasa( )
      retornar masa
   fin_metodo
   //-------------------------------
   real calcularEnergía( )
      real ener
      const real C = 2.997925 * Mat.elevar (10, 10)
      ener = masa * Mat.elevar(C, 2)
      retornar ener
   fin_metodo
fin_clase
//-----------------------------------------------------------------------------------
clase Proyecto
   Energía e
   estatico principal( )
      e = nuevo Energía( )
      Flujo.imprimir ("Ingrese la masa del objeto en gramos:")
      real ms = Flujo.leerReal( )
      e.asignarMasa(ms)
      Flujo.imprimir("Energía producida = " + e.calcularEnergía( ) + " ergios")
   fin_metodo
fin_clase
```

Figura 2: Seudo código para el problema de la ecuación de Einstein

2.2 EL JUEGO COQUITODOBLEO

Muchas personas en Colombia y en otros países de lengua hispana, aprendieron a leer y a escribir el idioma castellano con la cartilla de Coquito (Zapata-Santillana, 2006); ahora se propone aprender, o mejor comprender, los conceptos fundamentales del paradigma orientado a objetos de una manera teórica mediante clases presenciales apoyadas con el método MIPSOO y de manera lúdico práctica con un juego: *CoquitoDobleO*.

El juego, desarrollado en C# bajo el entorno integrado de desarrollo Visual Studio 2010, presenta un menú principal tipo texto con opciones para el afianzamiento teórico de conceptos de programación y tres niveles de juego: el primero permite jugar con clases y objetos, el segundo con herencia y polimorfismo y el tercero con clases y sobrecarga de métodos. Algunas pantallas típicas del juego se observan en la figura 3.

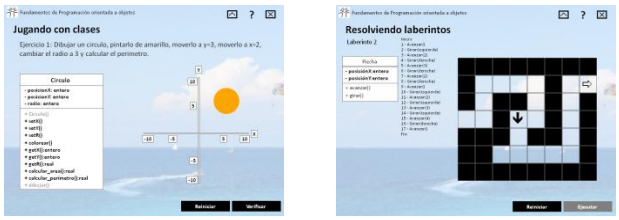

Fig. 3. Algunas pantallas típicas del juego CoquitoDobleO

En todos los niveles de juego, se ofrece la posibilidad de reiniciarlos o de evaluar las jugadas realizadas hasta un determinado momento para determinar el éxito o fracaso en el juego activo. El profesor puede añadir fácilmente nuevos juegos a los diferentes niveles, agregando a la carpeta *resources* generada por Visual Studio 2010 archivos de texto con el juego propuesto y las respectivas respuestas. Esto implica que el profesor no tiene que tener conocimientos de C#.

2.3 PRUEBAS APLICADAS EN EDUCACIÓN MEDIA TÉCNICA Y SUPERIOR

La aplicación de pruebas se llevó a cabo con cuatro grupos de estudiantes, dos del programa Ingeniería en Software del Tecnológico de Antioquia - Institución Universitaria, y dos de secundaria en grado décimo con educación media técnica en informática, procedentes de la Institución Educativa Comercial de Envigado. En las mismas, se empleó el método MIPSOO para enseñanza de la POO y el juego *CoquitoDobleO* para motivar el aprendizaje; la evaluación se efectuó con rúbricas y encuestas, las primeras diligenciadas por los profesores y las segundas por los estudiantes.

Para contextualizar la población objetivo asociada a las pruebas, cabe anotar que en Colombia se establecen seis estratos socieconómicos donde el 1 es el más bajo y el 6 el más alto. El Tecnológico de Antioquia - Institución Universitaria y la Institución Educativa Comercial de Envigado, son establecimientos públicos de educación que ofrecen sus servicios a estudiantes de los estratos sociales 1, 2 y 3, abriendo de ésta manera nuevas oportunidades de formación media y profesional a las clases menos favorecidas.

2.3.1 LAS RÚBRICAS

Los estudiantes de los grupos experimental y de control realizaron una exposición sobre los conceptos de programación estudiados, en presencia de profesores del área procedentes de otras instituciones de educación media y superior, quienes diligenciaron las rúbricas con seis criterios a evaluar y cuatro niveles de comprensión. La selección de criterios para las rúbricas se realizó atendiendo cuatro dimensiones de la comprensión: aspectos de conocimiento/contenido, formas de comunicación, propósito y método (Blythe, 2006), y fueron los siguientes:

- Distinción entre los conceptos de clase y objeto, y comprende el ciclo de vida de éste último. (Conocimiento/contenido)
- Aplicación de los conceptos fundamentales del paradigma de programación orientado a objetos (Conocimiento/contenido).
- Seguridad en la exposición, uso de vocabulario apropiado, buena pronunciación y modulación. (Formas de comunicación).
- Calidad de las diapositivas. (Formas de comunicación).
- Ejemplo sobre el tema expuesto. (Propósito)
- Solución del ejemplo. (Método)

Los niveles de comprensión de la rúbrica se clasificaron en *Ingenuo*, *Novato*, *Aprendiz* y *Experto*, con una ponderación de 1.25, 2.50, 3.75 y 5.00, respectivamente; además, se evaluó uno de tres temas concretos de la programación orientada a objetos: arreglos de objetos, herencia o polimorfismo.

Para la evaluación de cada criterio, el evaluador marca con una X un círculo dentro de la fila respectiva y le asigna un puntaje de 1 a 5; también puede marcar *No aplica* (N.A.).

Para determinar la puntuación total se trabajará con la siguiente fórmula:

$pt = \sum_{i=1}^{6} n_i * p_i$, donde

$pt: puntuación\ total.$

$i: número\ de\ criterios,\ i = 1, 2, \ldots, 6.$

$n_i: nivel\ seleccionado\ para\ el\ criterio\ i.$

$p: puntaje\ asignado\ por\ el\ evaluador\ para\ el\ criterio\ i. p_i \in Z, 1 \leq p_i \leq 5.$

2.3.2 LAS ENCUESTAS

Finalizado el proceso académico con los grupos experimental y de control, es decir, después de aplicada la evaluación final del curso, se entregó a cada estudiante de ambos grupos una encuesta con ocho aserciones a evaluar según cuatro escalas de actitudes tipo Likert (Fernández de Pinedo, 2007) y dos preguntas abiertas. Cabe anotar que las aserciones relacionadas con el software *CoquitoDobleO* no fueron planteadas al grupo de control.

Las aserciones planteadas fueron las siguientes:

1) Los conceptos estudiados sobre POO (objeto, clase, método, sentencias de control, herencia y polimorfismo) quedaron claros y comprendidos.
2) La metodología de clase magistral teórica es clara y suficiente en un proceso de aprendizaje de la programación orientada a objetos.
3) Resulta más conveniente para el aprendizaje estudiar POO de forma teórico-práctica, con el uso de herramientas de apoyo al aprendizaje como el juego *CoquitoDobleO* (u otro tipo de juego).
4) La evaluación con exámenes teóricos individuales y en equipo es apropiada para un curso de Lógica de Programación I.
5) Los talleres impresos y remitidos por correo electrónico contribuyeron a mejorar el proceso de aprendizaje de la asignatura.
6) Los problemas resueltos en clase sirvieron para mejorar la comprensión de la programación orientada a objetos.
7) El juego *CoquitoDobleO* ofrece elementos para mejorar el aprendizaje de la programación orientada a objetos.
8) Las herramientas virtuales basadas en TIC (Tecnologías de la Información y la Comunicación) como las páginas web, foros, chat, wiki, evaluaciones en línea, etc., serían convenientes para la mejora en el aprendizaje y comprensión de la programación orientada a objetos.

La encuesta finaliza con dos preguntas abiertas:

- ¿Qué fue lo que más le agradó del curso? Y en éste mismo sentido, ¿Qué fue lo que más le desagradó?
- ¿Qué recomendaciones tiene para mejorar el curso de Lógica de Programación I?

La caracterización de ambos grupos, relacionada con su tamaño, género de sus integrantes y edad, se observa en la Tabla 4.

Tabla 4. Caracterización de los grupos experimental y de control

Grupo	Tamaño	Hombres	Mujeres	Edad promedio
Experimental – Educación superior	36	28	8	19.7
De control – Educación superior	28	21	7	20.2
Experimental – Educación media técnica	18	9	9	17.1
De control – Educación media técnica	16	10	6	16.8

3. RESULTADOS DE LAS PRUEBAS

En la Tabla 5 se presentan los resultados obtenidos por cada grupo en educación media técnica y superior. Los resultados, relativamente bajos según lo indican los promedios obtenidos, son justificables teniendo en cuenta que se trata de noveles estudiantes que continuarán profundizando éstos temas en educación superior, con el estudio

asignaturas relacionadas con la construcción de elementos de software, lógica de programación avanzada e ingeniería de software (Tecnológico de Antioquia, 2011).

Tabla 5. Resultados de las rúbricas para los grupos de control y experimental

Tema	Puntaje promedio del tema (Grupos de control)		Puntaje promedio del tema (Grupos experimentales)	
	Media técnica	Educación superior	Media técnica	Educación superior
Arreglos de objetos	1.50	1.52	2.00	2.53
Herencia	1.15	1.17	1.85	2.08
Polimorfismo	1.52	1.64	1.80	1.82
Promedio del grupo	1.39	1.44	1.88	2.14

El consolidado estadístico de las encuestas se presenta en la Tabla 6, donde se exponen dos números separados por guión; el número de la izquierda representa la cantidad de respuestas del grupo en educación superior y el derecho al grupo en media técnica. Las convenciones manejadas en la tabla son:
A: Número de la aserción, **C. de A.**: Completamente de acuerdo, **D. A.**: De acuerdo, **I.**: Indeciso, **E. D.**: En desacuerdo, **C. en D.**: Completamente en desacuerdo, NA: No aplica.

Tabla 6. Resultados consolidados de las encuestas en los grupos experimental (GE) y de control (GdeC)

	C. de A.		D. A.		I.		E. D.		C. en D.	
A	GE	GdeC	GE	GdeC	GE	GdeC	GE	GdeC	GE	GdeC
1	7-3	11-2	23-9	17-6	6-3	0-2	0-1	0-0	0-0	0-0
2	10-6	16-6	17-6	12-4	8-4	0-0	1-0	0-0	0-0	0-0
3	22-6	NA-NA	11-7	NA-NA	3-1	NA-NA	0-0	NA-NA	0-2	NA-NA
4	23-4	25-3	12-8	3-7	1-4	0-0	0-0	0-0	0-0	0-0
5	12-5	20-6	17-8	6-2	5-2	1-2	2-0	0-0	0-1	0-0
6	18-4	23-7	16-7	5-2	2-5	0-1	0-0	0-0	0-0	0-0
7	18-4	NA-NA	15-9	NA-NA	3-2	NA-NA	0-0	NA-NA	0-1	NA-NA
8	16-0	21-6	12-2	7-3	8-4	0-1	0-0	0-0	0-0	0-0

Respecto a las preguntas abiertas, agradó en todos los grupos la metodología utilizada en el curso para la solución de problemas mediante los cuatro pasos mencionados en la introducción de éste artículo, los talleres remitidos por correo electrónico con ejercicios propuestos y resueltos, la solución de problemas en clase y las evaluaciones escritas con notas abiertas. Un aspecto que agradó en los grupos experimentales fue el trabajo en salas de informática con el software lúdico *CoquitoDobleO*. Un aspecto que desagradó en los grupos de control fue la falta de prácticas en laboratorio con un lenguaje de programación; además en todos los grupos desagradaron aspectos como la intensidad horaria del curso que consideran insuficiente, la celeridad con que el profesor explica los temas – factor subjetivo pues depende del docente a cargo- y el arduo ambiente que a veces generaba una fracción de estudiantes.

Las recomendaciones para mejorar los cursos *Introducción a la Programación* en educación media técnica y *Lógica de Programación I* en educación superior, se concentraron en el uso de un mayor número de tecnologías de la información y la comunicación para facilitar el aprendizaje, el desarrollo de prácticas en un lenguaje de programación con soporte para la orientación a objetos y el aumento de talleres grupales y exposiciones para incentivar el aprendizaje colaborativo.

4. CONCLUSIONES

El proceso docente educativo en las asignaturas *Introducción a la Programación* y *Lógica de Programación I* con la metodología de cuatro pasos descrita en (Botero et al, 2010), ha surtido buenos efectos en el aprendizaje, no obstante se detectan problemas de motivación en los jóvenes que forman parte de la denominada "generación digital" (Gibson et al, 2008), por lo cual se hacen necesarias nuevas alternativas que motiven el aprendizaje, entre ellas los juegos de ordenador. El trabajo lúdico complementario con el software *CoquitoDobleO* presentado ante dos grupos experimentales, contrastado con el trabajo realizado ante los grupos de control donde se prescindió del software, conlleva varias conclusiones:

- ✓ Es positivo incluir juegos digitales, de forma gradual y permanente, en el currículo y la evaluación de asignaturas como *Introducción a la Programación* y *Lógica de Programación I*, aserto extensible a materias de otras ciencias distintas a la computación.
- ✓ Según la Tabla 5, el tema con mayor dificultad de comprensión para los grupos de control es la herencia, mientras que para los grupos experimentales es el polimorfismo. Los promedios por tema y grupo son más altos en los grupos experimentales, lo cual indica que la utilización del juego *CoquitoDobleO* incentiva el proceso de aprendizaje de la programación orientada a objetos.
- ✓ Según la Tabla 6, las respuestas *C. de A. (Completamente de Acuerdo)* y *D. A. (De Acuerdo)* fueron las más seleccionadas por los grupos de ambos niveles educativos, se concluye que hay aceptación general por las temáticas y metodología del curso. Además, según lo demuestra el número de selecciones *C. de A.* y *D. A.* para las aserciones 3 y 7, en el grupo experimental resultó de una buena aceptación el juego *CoquitoDobleO* como apoyo a los procesos de aprendizaje de la programación orientada a objetos.
- ✓ Los juegos digitales apoyan las teorías de aprendizaje cognitivistas y constructivistas (De la Cueva, 2000) porque llevan un componente práctico dentro del proceso de cognición interno y propio de cada individuo, promueven el aprendizaje significativo y activo, proporcionan retroalimentación inmediata, facilitan la enseñanza personalizada y desarrollan nuevas formas de comprensión, aumentando así los grados de motivación.
- ✓ La aplicación de nuevas pruebas a grupos experimental y de control procedentes de otras instituciones de educación media o superior, es necesaria y pertinente. De hecho, se está desarrollando una nueva fase experimental con estudiantes de educación secundaria en media técnica -programación y desarrollo de software-, que permitirá difundir la metodología para el aprendizaje de la POO mediante un juego y ampliar el periplo evaluativo hacia otros estudiantes y profesores.

REFERENCIAS

Botero R. de J., Trefftz H. (2011). "Medra para el aprendizaje en lógica de programación orientada a objetos mediante un juego". *II Encuentro Internacional y VI Nacional de Investigación en Ingeniería de Sistemas e Informática – EIISI 2011*. UPTC, Tunja, Colombia.

Botero R. de J., Castro C., Maya J., Taborda G. y Valencia M. (2010). *Lógica y programación orientada a objetos: un enfoque basado en problemas*. Tecnológico de Antioquia, CITIA, Medellín.

Botero, R. de J. (2010). "Patrones Grasp y Anti-Patrones: un Enfoque Orientado a Objetos desde Lógica de Programación. *Entre Ciencia e Ingeniería*, ISSN 1909-8367, Año 4. No. 8, pp 161 – 173.

Blythe, T. (2006). *La enseñanza para la Comprensión. Guía para el docente*, 3ºedición Ed. Paidós, Argentina.

Chicharro M. E., García J. J., Ramírez R. V. (2008). "Júpiter: Un entorno web para el aprendizaje basado en juegos competitivos". *XII Congreso Internacional de Internet y Sociedad de la Información*, Málaga, España.

De la Cueva, Víctor. (2000). "El modelo educativo constructivista ABC2: aprendizaje basado en la construccion del conocimiento". *Memorias del Primer Congreso Nacional: Retos y Expectativas de la Universidad en México*. Universidad de Guadalajara, Guadalajara, Jal.

Denault, A. (2005). "Minueto, an Undergraduate Teaching Development Framework". Master's Thesis, McGill University.

Dorn B. y Sanders D. (2003) "Using Jeroo to Introduce Object-Oriented Programming". En FIE '03: *Proceedings of the 33rd annual Frontiers in Education conference*, volumen 1, pp. T4C 22-27.

Fernández de Pinedo, I. (2007). "Construcción de una escala de actitudes tipo Likert". http://www.insht.es/InshtWeb/Contenidos/Documentacion/FichasTecnicas/NTP/Ficheros/001a100/ntp_015.pdf, 01/25/2012.

Gibson T. L., Koontz D. A. y Van Den Hende M. (2008). "The Digital Generation. Teaching to a Population That Speaks an Entirely New Language". The Chair Academi, Denver, CO (USA).

Halma A. Robomind. (2005). http://www.robomind.net/en/index.html, 02/27/2012.

Hartness, K. (2004). "Robocode: Using games to teach artificial intelligence". *The Journal of Computing Sciences in Colleges*, 19 (4).

Kouznetsova S. (2007). "Using BlueJ and Blackjack to Teach Object Oriented Design Concepts in CS1". *Journal of Computing Sciences in Colleges*, Volume 22 Issue 4.

Kölling M. y Henriksen P. (2005). "Game Programming in Introductory Courses with Direct State Manipulation". *Proceedings of ITiCSE'05*, Lisboa, Portugal.

Marks S., Windsor J., Wünsche B. (2008). "Evaluation of Game Engines for Simulated Clinical Training". In *New Zealand Computer Science Research Student Conference 2008*. Christchurch, New Zealand.

McKenzie W. B. (2007). "An Alternative Approach to Introductory Object Oriented Programming: a Case Study in Programming with Alice". *The Third International Computing Education Research Workshop Georgia Institute of Technology*, Atlanta, GA, USA.

Moreno J., Montaño E. (2005). "ProBot: Juego para el aprendizaje de lógica de programación". J. Sánchez (Ed.): *Nuevas Ideas en Informática Educativa*, Volumen 5, pp. 1-7, Santiago de Chile.

Pattis R. (1981). *Karel the Robot: A Gentle Introduction to the Art of Programming with Pasca*l. John Wiley & Sons, Inc.

Resnick M., Maloney J., Monroy A., Rusk N., Eastmond E., Brennan K., Millner A., Rosenbaum E., Silver J., Silverman B., Kafai Y. (2009). "Scratch : Programming for Everyone". *Communications of the ACM*.

Singh J., Kumar S., Woods P. (2007). "Learning Computer Programming Using A Board Game – Case Study on C-Jump". *MMU International Symposium on Information and Communications Technologies*.

Tecnológico de Antioquia – Institución Universitaria (2011). Facultad de Ingeniería. Malla curricular del programa Ingeniería en Software, plan 02. http://www.tdea.edu.co, 02/16/2012.

Zapata-santillana, E. (2006). *Coquito clásico. Lectura in*icial. Ediciones Coquito, USA, LLC.

Autorización y Renuncia

Los autores autorizan a LACCEI para publicar el escrito en las memorias de la conferencia. LACCEI o los editores no son responsables ni por el contenido ni por las implicaciones de lo que esta expresado en el escrito