LA LÚDICA DE JUEGOS EN EL APRENDIZAJE DE LA PROGRAMACIÓN ORIENTADA A OBJETOS: UN PROTOTIPO EN C#

Por

Ricardo de Jesús Botero Tabares

UNIVERSIDAD EAFIT

Escuela de Ingenierías

Marzo 2012

PROYECTO DE INVESTIGACIÓN PARA EL GRADO DE

MAGÍSTER EN INGENIERÍA (ÁREA SISTEMAS Y COMPUTACIÓN)

Director:

Dr. Helmuth Trefftz Gómez

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 trabajo expone una de ellas, consistente en la aplicación de un método con cuatro fases (Identificación de requerimientos, Diseño del diagrama de clases, Especificación de responsabilidades de las clases y Escritura de seudocódigo), combinado con un software lúdico que presenta una serie de juegos en 2D. El método aporta los cimientos teóricos inherentes a la POO, mediante el estudio paulatino de los conceptos de clase, herencia, polimorfismo, sobrecarga de métodos y sentencias de control; el software motiva el aprendizaje por medio de una serie de juegos donde los participantes interactúan con los conceptos de una manera lúdica. Para demostrar la efectividad de ésta combinación, se presentan 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.

Abstract

In todays world, it is possible to support the learning of Object Oriented Programming (OOP), by new students, with multiple academic activities. In this work, one of them is presented.

We applied a four-phase method (Requirements Elicitation, Class Diagram Design, Class-Responsibilities Identification, and Pseudo-Code Writing). The method is combined with software that presents the user with various 2D games. The theoretical principles underlying OOP are presented, through the study of the concepts of Classes, Inheritance, Polymorphism, Method Overloading and Control Sentences.

The software aims at increasing the students' motivation by encouraging their learning through a number of games in which students learn while playing.

In order to evaluate the pedagogical effectivity of this combination, we present the results of a series of user experiences comparing the learning of four groups of students: two experimental groups that used the software and two control groups that did not.

Palabras clave

Aprendizaje de la programación

Programación orientada a objetos

Juegos digitales

Keywords

Learning programming

Object-Oriented Programming

Digital games

Agradecimientos

El trabajo en equipo de un proyecto de maestría conlleva el estudio y dedicación del maestrante con el apoyo invaluable de una serie de entidades y personas a las cuales les manifiesto sentimientos de gratitud.

Entre las primeras, agradezco al Tecnológico de Antioquia – Institución Universitaria, donde me desempeño como profesor de tiempo completo de la Facultad de Ingeniería, por su apoyo en tiempo y financiación del estudio de posgrado. A la Universidad EAFIT por brindarme espacios académicos dinamizados con el valioso recurso humano de los profesores de la maestría que me asesoraron en los diferentes semestres.

Entre las segundas, agradezco la deferencia, orientación y apoyo de mi asesor, doctor Helmuth Trefftz. El apoyo de los profesores evaluadores que aplicaron las rúbricas en la fase final de las pruebas con los grupos experimental y de control fue invaluable; en particular, gracias a Raquel Anaya de la Universidad EAFIT, Adriana Reyes del Politécnico Colombiano Jaime Isaza Cadavid, Jorge Gaviria de la Fundación Universitaria Luis Amigó, Juan Camilo Giraldo del Tecnológico de Antioquia – Institución Universitaria, Liliana García de la Institución Educativa Gabriel García Márquez, y Wilmar Castañeda y Rita Ligia Osorio de la Institución Educativa Comercial de Envigado.

Que no pasen desapercibidas las palabras de ánimo de mi esposa Sara y mi hijo Rafael, quienes convirtieron el desarrollo del proyecto en una meta familiar; a ellos mi perenne amor y gratitud.

Y a Dios: gracias por tenerme con salud y vida para finalizar éste trabajo que es una continuación y también otro comienzo.

Medellín, marzo de 2012

Contenido

Lista de figuras

Lista de tablas

Capítulo 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 [GM03]. Pero, antes de continuar con este esbozo introductorio cabe preguntarse: ¿se puede mejorar el aprendizaje de la programación con el uso o el diseño de juegos de computadora?

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 [LKG07] [Alb08] 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 [KH05] es un entorno integrado de desarrollo y programación que facilita la escritura de juegos y simulaciones en lenguaje Java. Robocode [KUW03] [Har04] 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 [RM05] 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 [Pat81] 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 [Mck07] [DCP07] es un entorno de desarrollo de software tridimensional, en donde podremos aprender a programar jugando en entornos virtuales. C-jump [SS07] 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 *Minueto* [Den05], *Jeroo* [DS04], *ProBot* [MM09] y *Júpiter* [Chi08] 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– [BT11], que incentiva de manera directa los conceptos fundamentales del paradigma de programación implícito en su nombre y donde se supone que las nuevas generaciones de estudiantes de un primer curso de fundamentos de programación, pueden comprender más fácilmente los principios básicos del paradigma de programación orientado a objetos con la intermediación de juegos digitales.

Por lo anterior, el presente trabajo expone nuevos recursos para apoyar la enseñanza y el aprendizaje de la POO, aplicada en escenarios iniciales del proceso formativo de tecnólogos en sistemas, ingenieros en software y programas relacionados, donde se tengan en cuenta elementos inherentes a las ciencias de la computación y a la lúdica basada en juegos digitales para incentivar los grados de motivación discente. De ésta manera, se establecen lineamientos

curriculares innovadores que aumentan los grados de motivación en el aprendizaje de la POO, para facilitar el futuro estudio y comprensión de otras áreas en niveles superiores de la ingeniería de software.

Capítulo 2

Formulación, objetivos y diseño de la investigación

2.1. Formulación del problema

Según lo descrito en el capítulo introductorio, se plantea el problema de comparar los niveles de motivación para la comprensión de la programación orientada a objetos, en estudiantes de educación media técnica y superior, con la aplicación de un método que incluye el uso de un juego digital de creación propia.

Surgen entonces dos interrogantes por resolver:

¿El uso o el diseño de juegos digitales motiva el aprendizaje de la programación orientada a objetos?

¿Cuáles son los juegos actuales de mayor preponderancia en el mercado que incentivan el aprendizaje de la programación de computadoras y a qué público objetivo van dirigidos?

2.2 Objetivo general

Proponer un método para aprender fundamentos de programación orientada a objetos, combinado con el uso de un juego digital que aumente los grados de motivación en el aprendizaje de personas que inician estudios de tecnología e ingeniería de sistemas y programas afines.

2.3 Objetivos específicos

- Presentar un estado del arte relacionado con el aprendizaje de la programación y el desarrollo o uso de juegos interactivos de computadora.

- Desarrollar un prototipo de juego, que motive el aprendizaje de los conceptos fundamentales del paradigma de programación orientado a objetos (clase, objeto, herencia y polimorfismo).

- Realizar experimentos, pruebas o experiencias de usuario, con grupos de estudiantes de educación media técnica y superior, para identificar si el uso de juegos digitales motiva o no el aprendizaje de la programación orientada a objetos.

2.4 Diseño de la investigación

El modelo de diseño fue experimental, dado que la investigación se desarrolló por fases o etapas, a saber:

i) Dar cuenta del estado del arte de los juegos para el aprendizaje de la POO.

   A partir de la lectura de artículos publicados en revistas científicas nacionales extranjeras, se identifican una serie de juegos para el aprendizaje de la programación, dirigidos a público de diferente nivel educativo (educación secundaria y terciaria).

ii) Desarrollar un juego para aumentar los grados de motivación durante el estudio inicial de la POO.

El juego, denominado *CoquitoDobleO*, se desarrolla en Microsoft Visual Studio 2010 con el lenguaje C#. El modelo de proceso utilizado fue RUP (Proceso Unificado Rational) y el patrón de desarrollo MVC (Modelo-Vista-Controlador) [Som05].

iii) Revisar el juego para verificar su pertinencia.

El software lúdico es revisado o puesto a prueba ante estudiantes de pregrado en Ingeniería de Sistemas de la Universidad EAFIT y ante estudiantes de Tecnología en Sistemas del Tecnológico de Antioquia – Institución Universitaria. Esto conllevó correcciones al producto final presentado a los estudiantes durante la fase relacionada con las experiencias de usuario.

iv) Propiciar experiencias de usuario.

Las experiencias de usuario se realizaron ante un público conformado por estudiantes de educación media técnica y educación superior, donde se diferenció entre dos clases de grupo: experimental y de control.

v) Analizar las experiencias de usuario (resultados).

El análisis de los resultados se realizó por medio de rúbricas diligenciadas por los profesores y por encuestas aplicadas a los estudiantes.

vi) Conclusiones.

A partir de las experiencias de usuario se argumentan varias conclusiones, en respuesta a los objetivos planteados en la investigación.

Capítulo 3

Los juegos en el aprendizaje de la programación orientada a objetos

La enseñanza de la programación con el apoyo de metodologías y herramientas pedagógicas centradas en la lúdica y los juegos digitales, se ha dado en varias instituciones de educación superior del mundo. Así, en la Universidad de Málaga (España) se presenta un enfoque pedagógico desarrollado por el equipo docente de la asignatura Laboratorio de Programación de la Ingeniería Técnica de Telecomunicación [Bue01] . El enfoque, basado en el uso de juegos de ordenador, incluye tres prácticas: la primera intenta asentar los conocimientos sobre estructuras de datos simples, mediante juegos con una entrada-salida en modo texto sencilla. La segunda práctica tiene como objetivo el manejo de estructuras de datos complejas, mediante la implementación del juego de la serpiente, en el cual los alumnos comienzan aprendiendo a controlar las teclas especiales de movimiento y las coordenadas de la pantalla. La tercera y última práctica introduce el trabajo con librerías para la construcción de tipos abstractos de datos e interfaces gráficas de usuario, mediante el juego de los barquitos, donde el alumno debe manejar varias estructuras de arreglos de registros.

En la Escuela Colombiana de Ingeniería “Julio Garavito” se ha logrado experiencia en la enseñanza del paradigma orientado a objetos a través de juegos [Cad06], donde los estudiantes desarrollan un modelo de clases que represente una pista de baile, compuesta de bailarines que pueden realizar una serie consecutiva de

pasos o “estilo” de baile, haciendo uso de una serie de métodos básicos definidos en la clase Bailarín, como subir y bajar tanto los brazos como piernas; el proyecto final es el juego de invasores, cuyo propósito es implementar un marco de trabajo (framework) con toda la lógica de un juego donde una nave controlada por el jugador se enfrenta a una diversidad de elementos, como los asteroides y elementos autónomos que se desplazan con determinados patrones y atacan al jugador en caso de detectarlo.

En la Universidad Complutense de Madrid (España) [Rec06] se presenta una experiencia realizada en un curso de Laboratorio de Programación III, en la que se propone la implementación de Sudokus en Java. La experiencia permite poner en práctica conceptos de estructuras de datos y algoritmia, así como patrones de diseño y programación de dispositivos móviles.

En la asignatura de Informática Industrial de la Escuela Universitaria de Ingeniería Técnica Industrial de la Universidad Politécnica de Madrid (España) [Rod08] se ha optado por desarrollar un método docente basado en el diseño y desarrollo de una aplicación grafica interactiva como son los videojuegos, los simuladores o herramientas de ingeniería. La enseñanza de la POO se ve facilitada en gran medida por un enfoque unificado de la asignatura que engloba desde las practicas realizadas en laboratorio a las tutorías, el desarrollo de trabajos personales por los alumnos, la docencia teórica e incluso el material docente. Se cree a raíz de los buenos resultados obtenidos, que la motivación e interés que despierta este enfoque en el alumnado redunda claramente en una mejora de la docencia y en los resultados académicos.

En la Facultad de Ingeniería de la Universidad ORT Uruguay [Fri08], se presentan diversas líneas de trabajo en la Cátedra de Programación de la Facultad de Ingeniería, relacionadas con la enseñanza y eprendizaje de la programación. Estas líneas son el diseño de entornos de aprendizaje basados en la gestión del conocimiento, el uso de morfismos, el diseño y aplicación de “Kinesthetic learning

activity” y el uso de Scratch en dos cursos de Ciencias de la Computación 1. Esta última herramienta se utiliza en las primeras semanas de los cursos con el propósito motivar y mejorar las experiencias iniciales con la programación, tratando de detectar su influencia en las puntuaciones y las tasas de deserción escolar en comparación con los cursos normales. Además, se desarrollaron guías detalladas de laboratorio, ejercicios, pruebas y formas de preguntas. Al contrastar los resultados con los cursos normales, se encontró que los estudiantes usuarios de Scratch denotan una motivación más alta, pero no hubo evidencia estadística de las tasas de abandono ni de las puntuaciones obtenidas.

En la Universidad de Melbourne (Victoria, Australia) se tiene en cuenta que la mayoría de los estudiantes de Sistemas de Información de los primeros niveles no están motivados para la POO, por eso la idea de completar un juego de tablero 2D como materia de un semestre los animará en dicho aspecto [GB05]. Este trabajo fomenta la reutilización de componentes de software y permite a los estudiantes concentrarse en cuestiones de interfaz y diseño, sin dejar de satisfacer las necesidades de aprendizaje.

En la Universidad de Corea (Seúl, Corea del Sur) se trabaja con el Juego de la Tarjeta Smalltalk (SCG) [Kim06], que se basa en reglas de conversación de uso diario. Los participantes pueden entender los conceptos de la orientación a los objetos en forma evolutiva sin un conocimiento previo de la POO. En el transcurso del juego los participantes no utilizan términos técnicos como clase, herencia, instancia u otros asociados al paradigma, sin embargo usan escenarios que los conducen naturalmente al desarrollo de los conceptos por sí mismos y con sus propios términos.

En las universidades Northwestern (Illinois, USA) y de Victoria (Columbia Británica, Canadá) [RLG07] se propone un método para evaluar la pedagogía basada en grupos con el ánimo de afianzar las capacidades de programación de los estudiantes y buscar su inserción adecuada en la industria, donde se hace

énfasis en la incorporación de herramientas industriales para el diseño de juegos en el aula, en vez de aplicar valoraciones de las estrategias pedagógicas.

En la Universidad de Kettering (Michigan, USA), Baibak y Agrawal [BA07] sugieren diferentes clases de juegos para ser asignados en cursos de Ciencias de la Computación. Esto benefica el plan de estudios porque se generan ambientes de aprendizaje divertidos y atractivos, proporciondo una base sólida para la comprensión y creación de algoritmos. Además, se sugieren diferentes tipos de juegos que podrían ser asignados en un curso, así como los tipos de algoritmos que pueden ser aprendidos en este proceso.

Diversas técnicas de interacción se están aplicando en un intento de dar a los estudiantes experiencias atractivas y concretas con objetos. En el Laboratorio de Computación de la Universidad de Kent (Reino Unido), el entorno Greenfoot [KH05] se ha propuesto como otro paso en este aspecto. Greenfoot es un entorno integrado de desarrollo y programación que facilita enormemente la escritura de juegos y simulaciones en lenguaje Java. Si bien la introducción de la POO se mueve lentamente hacia abajo por grupos de edad - a partir de cursos superiores universitarios hacia a los cursos de introducción de primeros niveles, y ahora en las escuelas secundarias - muchos intentos se están haciendo para hacer de la introducción a la POO menos abstracta y teórica.

Aunque no se trata de un juego, el uso del ambiente BlueJ se promueve en el Departamento de Ciencias de la Computación de la Universidad del Estado de Sam Houston (Texas, USA) [Kou07], porque incluso los programadores novatos serán capaces de desarrollar una implementación funcional del juego de cartas Blackjack, a manera de ejercicio lúdico en el proceso de aprendizaje de la programación. Debido a que los conceptos se presentan en el contexto de una aplicación familiar y divertida, la tarea incrementa su nivel de compromiso. En adición, los estudiantes descubren por sí mismos las ventajas del diseño orientado a objetos. Todas las tareas se implementan con el uso del ambiente de desarrollo

integrado de BlueJ el cual reduce la sobrecarga del aprendizaje del código y ayuda a que los estudiantes creen fácilmente gráficos incorporables al proyecto. Las ventajas del ambiente de BlueJ incluyen las representaciones graficas de las clases y objetos en un proyecto con el cual pueden interactuar los estudiantes.

En la Universidad del Estado de Sam Houston también se incorpora el juego Robocode en una clase de inteligencia artificial [Har04] y se le entregan a los estudiantes herramientas para el desarrollo de versiones prácticas de los algoritmos, por consiguiente los estudiantes aprecian mejor la teoría y desarrollan una mayor confianza en su aprendizaje.

En la Universidad de Roger Williams (Bristol, Inglaterra) y en la Escuela de Negocios Gabelli de la Universidad de Fordham (New York, USA) [Mck07], en lugar de usar lenguajes tradicionales como Java o Visual Basic, se esta usando Alice, un ambiente de programación en 3D donde los objetos son humanos, animales u objetos de fantasía que habitan y actúan en mundos reales o imaginarios. Los estudiantes pueden utilizar Alice para la programación de animaciones o juegos interactivos. El método de Alice de ”arrastrar y soltar” se ha reportado como una ventaja sobre la escritura de comandos.

Keefe [Kee06] reporta un proyecto de investigación que indaga por el efecto de la introducción de varias prácticas de Programación Extrema (XP) en la enseñanza de técnicas para estudiantes nuevos en programación en la Facultad de Tecnología de la Información de la Universidad de Monash (Melbourne, Australia).

El concepto clave de XP es la práctica colaborativa entre dos programadores trabajando conjuntamente en una máquina para completar una tarea programación. Ambos desarrolladores se involucran en el proceso de codificación, uno se designa como ”el conductor“ y el otro como “el navegador”. ”El conductor “ se concentra en la tarea a mano, mientras que “el navegador” constantemente revisa el código buscando problemas futuros inminentes. Los

estudiantes que programan a dúo aprenden mas entre sí y disfrutan mas la experiencia.

En la Facultad de Ciencias de la Computación de la Universidad McGill [Den05] en Montreal, Canadá, se presenta a Minueto: marco de trabajo para el desarrollo de juegos en Java diseñado específicamente para estudiantes de pregrado. Minueto es de tipo multiplataforma con el objetivo de simplificar el proceso de desarrollo de juegos por medio del encapsulamiento de tareas complejas de programación tales como la programación de gráficos, de audio, de teclado y de mouse en objetos de manipulación sencilla. Este diseño simple permite que los estudiantes comiencen el desarrollo de juegos luego de un corto periodo de tiempo.

Jeroo [DS04] es una herramienta pedagógica que entrega una introducción adecuada y amable a la POO, utilizada en la Universidad Estatal Missouri Noroeste (USA). Esta herramienta se ha desarrollado para ayudar a los programadores novatos al aprendizaje de los conceptos básicos del uso de la POO para la solución de problemas, con el aprendizaje de métodos de escritura que soporten la descomposición funcional de las tareas y que aprendan la semántica de las estructuras de control fundamentales. La sintaxis de Jeroo provee una transición suave hacia Java, C++ o C#. La interfaz de usuario tiene una ventana en la cual todo esta visible. El sobresaltado del código fuente, las animaciones sencillas y el panel de status continuamente actualizado proveen un ambiente rico de enseñanza y aprendizaje siendo ideal para el aprendizaje de programadores novatos.

En el Departamento de Matematicas y Ciencias de la Computación de la Universidad de Paderborn (Alemania), los profesores Schulte y Niere [SN02] exponen un concepto que utiliza una secuencia de aprendizaje para fomentar actividades en los aprendices que comienzan con la exploración de un modelo orientado a los objetos a un nivel de análisis y diseño. La orientación se reduce continuamente hacia una fase del proyecto en la cual los estudiantes modelan e

implementan una pequeña aplicación orientada a los objetos. Este método usa un lenguaje grafico centrado en las estructuras de los objetos; se aplican métodos para el cambio de las estructuras de los objetos de las aplicaciones y se usan herramientas para el soporte del diseño grafico y para la especificación de aplicaciones ejecutables orientadas a los objetos.

En el Departamento de Ciencias de la Información y Tecnología de la Universidad de Pennsylvania (Philadelphia, USA) [Ryo08], se les proporciona a los estudiantes un problema desafiante y real de tamaño significativo el cual es relevante a los objetivos de aprendizaje de un curso dado relacionado con ingeniería de software. Los estudiantes son animados para que resuelvan el problema en grupo en un semestre con la minima ayuda del instructor del curso. Destacando esto ultimo, se tiene un modelo pedagógico denominado ABP (Aprendizaje Basado en Problemas) que enfatiza el rol de los problemas de la vida diaria aunado con procesos de descubrimiento colaborativos para el aprendizaje. Fuera del método tradicional de enseñanza orientado a la clase magistral, ABP hace mayor énfasis en el rol como facilitador del instructor, para la preparación de problemas significativos e interesantes y para la creación y organización de los materiales del curso, de forma que los estudiantes tengan la dosis justa de información en cada clase para que incrementalmente desarrollen la solución final.

En el Departamento de Ciencias de la Computación de la Universidad de Denver (USA) [EL08], parten de la idea que muchos de los viejos juegos de Unix basados en texto, podrían ser una parte valiosa de un plan de estudios de desarrollo de videojuegos, por eso proponen la utilización del antiguo juego Rogue en la clase de CS1. Rogue es un juego de rol (RPG), que permite una introducción a la programación, al diseño orientado a objetos y a los patrones de diseño. Además de estos conceptos, los estudiantes obtienen un conocimiento adicional de la historia del juego y experiencia en la lectura de código heredado. El proyecto también permite a los estudiantes trabajar con interfaces de usuario, inteligencia artificial, métodos de prueba y enfrentar problemas en el diseño de software.

En la Universidad de Texas, sedes de San Antonio y Austin (USA), los profesores Yuen y Liu [YL10] plantean un estudio cognitivo con un interactivo de edición multimedia (IMA), y analizan cómo afecta al equipo de estudiantes principiantes de POO. En este estudio, la edición multimedia interactiva se refiere a la construcción de un juego de rol con una plantilla de juego desarrollada con Adobe Flash CS3 y ActionScript 2.0. Tres procesos cognitivos se observaron en este estudio: desequilibrio, exploración y conciencia. El desequilibrio es un estado de conflicto cognitivo donde los alumnos no pueden adaptarse a nueva información recibida dedido a la inestabilidad producida cuando la retroalimentación visual y textual de las pruebas con los juegos, es inconsistente con las expectativas de los participantes. La exploración es el método inicial por el cual los participantes intentaron resolver sus momentos de desequilibrio, abriendo otros archivos para tratar de resolver problemas. Conciencia es el nivel en el que los participantes entienden la arquitectura orientada a objetos a partir de plantillas dadas en el juego. al seguimiento a las relaciones jerárquicas e interactivos entre clases.

En el Departamento de Ciencias de la Computación de la Universidad de Rowan (Glassboro, New Jersey, USA) [Lob00] consideran la hipótesis de que el uso de objetos del mundo real en un laboratorio de introducción a la POO aumenta significativamente el interés en los estudiantes y la comprensión del enfoque orientado a objetos. El objeto real de la investigación es un dispositivo de hardware desarrollado localmente, denominado MIPPET (Module for Input/Output Programming Projects Enhancing Teaching). La clase Mippet, desarrollada en C++, fue creada para modelar este dispositivo de hardware y posteriormente fue utilizada en un laboratorio de POO. Otra sección del mismo laboratorio de objetos se basó en un proyecto de C++ con la clase SimRobot; ningún hardware especial estuvo involucrado en esta práctica. El análisis de las encuestas diligenciadas por los estudiantes en ambas secciones, sugieren que el aprendizaje mejora cuando se utilizan objetos de mundo real para enseñar POO.

Todos los juegos y entornos para el aprendizaje de la POO antes mencionados ,difieren de la propuesta del presente trabajo por varioas razones:

- La propuesta *CoquitoDobleO* se enfoca hacia la teoría básica de la programación orientada a objetos, es decir, a los conceptos de objeto, clase, paso de mensajes, sobrecarga herencia y polimorfismo. No pretende, como se presenta en la Universidad de Málaga, profundizar elementos relacionados con las estructuras de datos simples o complejos ni con las librerías o paquetes, los cuales se tratarán en otros cursos avanzados de programación e ingeniería de software.

- En el artículo del profesor Cadavid Rengifo de la Escuela Colombiana de Ingeniería “Julio Garavito” se afirma que los estudiantes realizan su primera práctica con un lenguaje estructurado, mientras que en la propuesta *CoquitoDobleO* se trabaja con independencia de cualquier lenguaje de producción comercial; de igual manera, la práctica final relacionada con el juego de invasores requiere interactuar con un juego desarrollado en un lenguaje de programación de alto nivel, cosa que debe tratarse en segundo curso o asignatura avanzada de técnicas de programación.

- El trabajo con sudokus en la Universidad Complutense de Madrid se enfoca a un curso avanzado de fundamentos de programación, mientras que la propuesta *CoquitoDobleO* se orienta a un curso básico de programación, donde no se interactúa con Java ni con dispositivos móviles.

- En el presente trabajo no se propone interactuar con videojuegos ni simuladores, como se pretende en la Escuela Universitaria de Ingeniería Técnica Industrial de la Universidad Politécnica de Madrid. La propuesta *CoquitoDobleO* consiste en un juego convencional para PC en dos dimensiones para motivar el aprendizaje de la POO.

- La propuesta de la Universidad ORT es interesante en cuanto a la utilización de la gestión del conocimiento, los morfismos y actividades kinestésicas para la enseñanza y aprendizaje de la programación, cuestiones no tratadas en la propuesta *CoquitoDobleO*.

- Lo que se logra con *CoquitoDobleO* difiere de la propuesta de Goschnick y Balbo de la Universidad de Melbourne porque no se trata de completar un juego, sino de interactuar con uno de creación propia.

- En *CoquitoDobleO* también se propone el aprendizaje de la programación de forma evolutiva, pero no con un juego de reglas conversacionales de uso diario como el Juego de la Tarjeta Smalltalk de la Universidad de Corea.

- La pedagogía propuesta en *CoquitoDobleO* es más de carácter individual y no basada en grupos, como se presenta en las universidades Northwestern y de Victoria.

- La propuesta de Baibak y Agrawal se enfoca a la creación de juegos aplicando algoritmos para la gestión de bases de datos, solución de colisiones, entradas de datos, hallazgo de trayectorias y gestión de gráficos, entre otros.
  La creación de juegos en sí no interesa en la propuesta *CoquitoDobleO*, donde se plantean juegos o retos a solucionar, para motivar el aprendizaje de conceptos fundamentales como objeto, clase, herencia, sobrecarga de métodos y polimorfismo.

- Es claro que Greenford, así como BlueJ, proporcionan entornos de aprendizaje para captar los conceptos de la POO mediante aplicaciones gráficas sencillas y atractivas, utilizando el lenguaje Java. Sin embargo, el interés de ésta propuesta de maestría consiste en aportar elementos

pedagógicos para el aprendizaje de la programación orientada a objetos con el uso de juegos y con independencia de cualquier lenguaje.

- En la propuesta para el aprendizaje de la programación realizada en el proyecto *CoquitoDobleO*, no se ha considerado el uso de BlueJ como entorno de aprendizaje de la programación. Sería interesante tenerlo en cuenta en estadios intermedios o finales del proceso de aprendizaje de la POO, ya que se tiene información, por el artículo de Kouznetsova [Kou07], sobre los beneficios y posibilidades de BlueJ.

- Cuando se toma como caso de estudio a Alice en alguna de sus versiones (3.0 por ejemplo), es indudable que pueden aumentar los grados de interés por el aprendizaje de la programación; sin embargo, antes de interactuar con este juego resulta conveniente manejar otros que presenten los conceptos de programación con diferentes matices y grados de complejidad, como ocurre con los juegos Scratch, RoboMind y *CoquitoDobleO*.

- Robocode pretende facilitar el aprendizaje de la programación con Java, y en algunos casos se puede extender este juego para impulsar el aprendizaje de la inteligencia artificial, tema que no interesa en un curso de fundamentos de programación por su carácter avanzado.

- La Programación Extrema (XP) a demostrado efectos beneficiosos en el aprendizaje de la programación y la ingeniería de software en estadios intermedios o avanzados del proceso formativo del estudiante. Aplicar XP en la enseñanza de los cimientos de la programación orientada a objetos puede resultar contraproducente, por eso en la propuesta *CoquitoDobleO* se aplicarán didácticas progresivas que podrían favorecer el trabajo con XP en cursos avanzados de otros semestres.

- Minueto, al igual que Robocode, pretende facilitar el aprendizaje de la programación con el lenguaje Java, aspecto que no interesa en este trabajo de maestría por encontrarse enfocada al aprendizaje de la POO con independencia de cualquier lenguaje de programación.

- Aunque la herramienta pedagógica Jeroo presenta animaciones e interfaces gráficas y cumple con cuatro objetivos educativos importantes (compromete a los estudiantes de manera inmediata, genera confianza, mejora la comprensión de los tópicos seleccionados y desarrolla habilidades de programación cruciales), no presenta a los juegos de computadora como una alternativa de aprendizaje de la POO. Sería interesante aplicar esta herramienta en el aprendizaje de lenguajes como C++, C# o Java, dado que Jeroo provee una leve transición hacia este tipo de lenguajes.

- El artículo de Ryoo-Fonseca-Janzen [Ryo08] presenta una estrategia pedagógica para la enseñanza de la ingeniería de software fundada en el ABP, no de la POO en el contexto de la lúdica de juegos. El ABP es un excelente punto de partida para el aprendizaje de la programación, el cual se tiene en cuenta en la propuesta *CoquitoDobleO*, donde se invita a la solución de juegos-problema.

Capítulo 4

El juego *CoquitoDobleO* en el aprendizaje de la POO

Muchos de nosotros, en Colombia y otros países latinoamericanos, aprendimos a leer y a escribir el idioma español con la cartilla de Coquito [Zap06]; ahora se propone aprender, o más bien comprender, los conceptos fundamentales del paradigma orientado a objetos de una manera teórica mediante clases presenciales apoyadas con un libro de texto, y de manera lúdico práctica con un juego: *CoquitoDobleO*.

4.1 Descripción general del juego

El juego presenta inicialmente una pantalla de bienvenida tipo “splash” que se despliega por unos pocos segundos (figura 4.1). Dicha pantalla expone el icono del juego, siempre exhibido en la esquina superior izquierda de todas las pantallas de la aplicación, y dos figuras humanas (hombre y mujer) acompañadas del nombre del juego con un subtítulo y en la parte inferior el mensaje “Juegos didácticos para aprender clases, objetos, herencia y polimorfismo”.

Figura 4.1: Pantalla inicial del juego *CoquitoDobleO*

Luego de la pantalla de bienvenida, se presenta un menú principal textual con opciones para el afianzamiento teórico de conceptos de programación y para 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, como se ilustra en la figura 4.2.

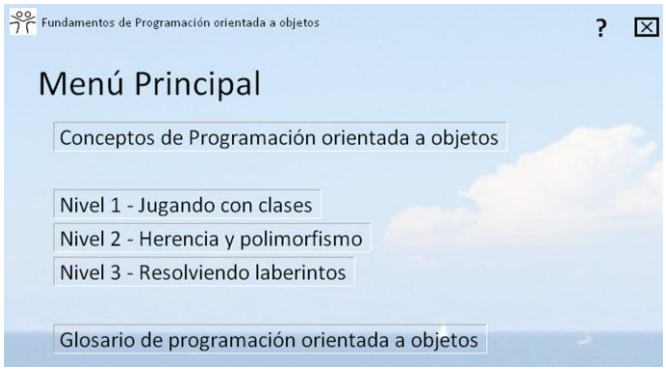

Figura 4.2: Menú principal del juego *CoquitoDobleO*.

Además, el menú principal presenta la opción “Glosario de programación orientada a objetos” que presenta un glosario básico sobre el tema.

La primera opción de menú, etiquetada como “Conceptos de Programación orientada a objetos” no contiene juegos, los cuales se presentan en las demás opciones (niveles 1, 2 y 3); ésta opción presenta dos submenús (“Selección secuencial” y “Selección múltiple”), como se observa en la figura 4.3.

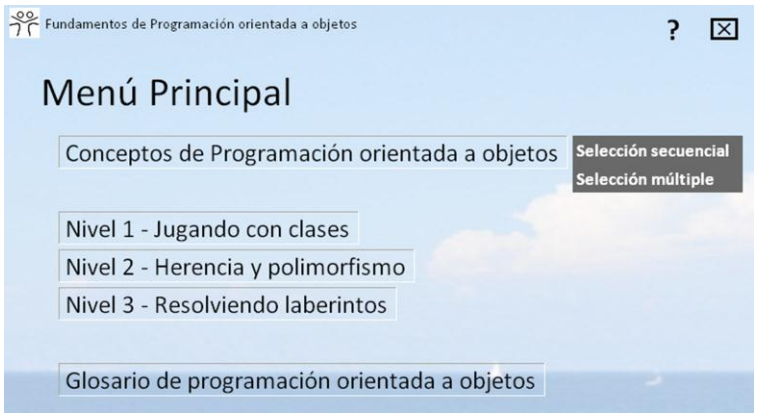

Figura 4.3: Sub-menús de la opción “Conceptos de programación orientada a objetos”

A lo largo del juego, toda pantalla presentará dos botones para obtener ayuda sobre la pantalla actual (botón “?”) y para salir de la aplicación (botón “X”).

Al escoger la opción “Selección secuencial” se presentan varios problemas sobre programación, como el que se expone en la figura 4.4, donde se añade un botón de retroceso (“^”) en la esquina superior derecha, que permite regresar al menú principal; además, se añaden dos nuevos botones en la esquina inferior derecha, el primero para reiniciar el problema en curso (en este caso la secuencia seleccionada se borra) y el segundo para verificar o evaluar las respuestas seleccionadas.

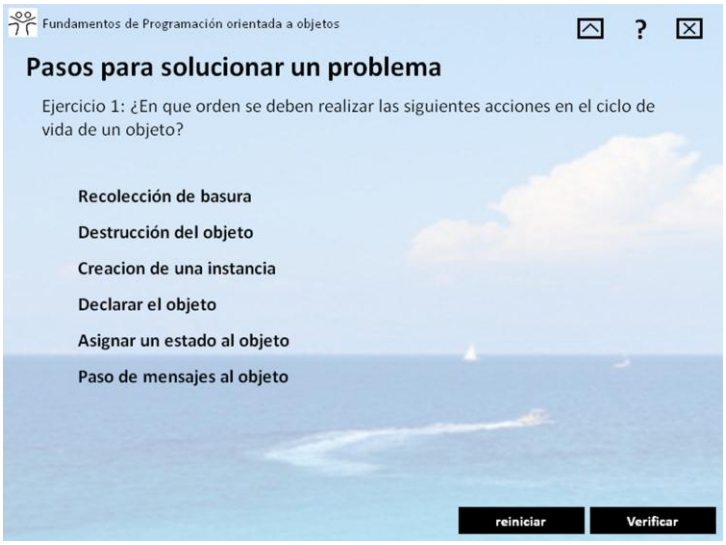

Figura 4.4: Un problema típico de selección secuencial

Al escoger la opción “Selección múltiple” se presentan problemas que permiten seleccionar una respuesta única entre varias posibles, como el que se ilustra en la figura 4.5.

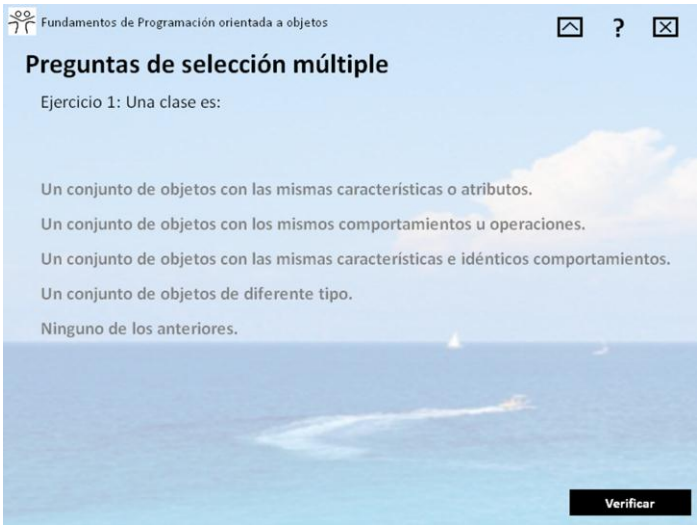

Figura 4.5: Un problema típico de selección múltiple

Al final, el botón “Verificar” permite saber si la respuesta elegida fue o no la correcta.

Los juegos de *CoquitoDobleO*, como ya se indicó, están expresos en las opciones del menú principal etiquetadas con “Nivel 1 – Jugando con clases”, “Nivel 2 – Herencia y polimorfismo” y “Nivel 3 – Resolviendo laberintos”. Veamos cada una de ellas.

4.1.1 Nivel 1 – Jugando con clases

En el primer nivel se juega con objetos de la clase Círculo y se trabaja con métodos constructores, modificadores y analizadores. En éste nivel de juego se invita al jugador a construir círculos, colorearlos, desplazarlos a través del plano cartesiano y calcular su área y perímetro.

Al hacer clic sobre el primer nivel de juego, se presentan las opciones “Métodos analizadores” y “Métodos modificadores y analizadores”, como se observa en la figura 4.6.

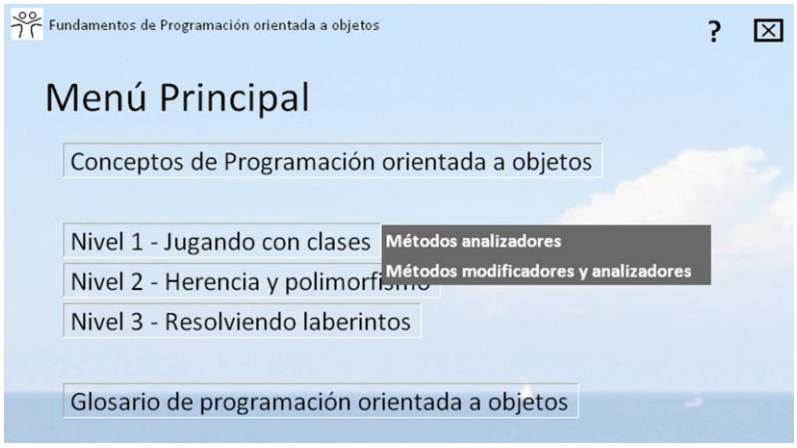

Figura 4.6: Sub-menús de la opción “Nivel 1 – Jugando con clases”

Recordar que los métodos analizadores son aquellos que requieren de una alta capacidad de abstracción y dominio de las sentencias de control por parte del analista, dado que éste tipo de métodos desarrollan las responsabilidades de las clases, es decir, realizan el trabajo fundamental para el cual la clase fue creada. Los métodos modificadores controlan el estado del objeto, equivalentes a los métodos setX( ) y getX( ) de algunos lenguajes de programación, donde X corresponde a un atributo.

Al seleccionar la opción “Métodos analizadores” se presentan juegos con la clase Círculo, que contiene dos atributos, un método constructor y tres métodos analizadores que permiten dibujar, rellenar y mover el círculo. Una pantalla típica de éste nivel de juego se observa en la figura 4.7.

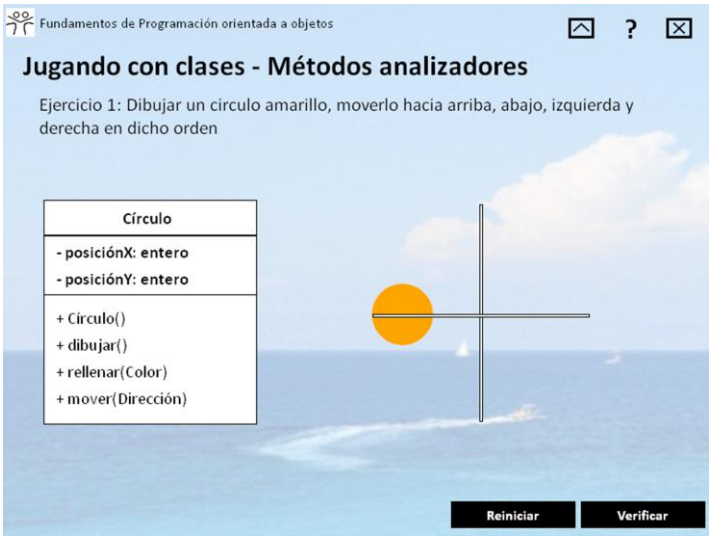

Figura 4.7: Pantalla típica de “Jugando con clases – Métodos analizadores”

Al seleccionar la opción “Métodos modificadores y analizadores” se presentan juegos con la clase Círculo con tres atributos, un método constructor, seis métodos modificadores y cuatro métodos analizadores que permiten dibujar, colorear, calcular el área y calcular el perímetro del círculo. Una pantalla típica de éste nivel de juego se observa en la figura 4.8.

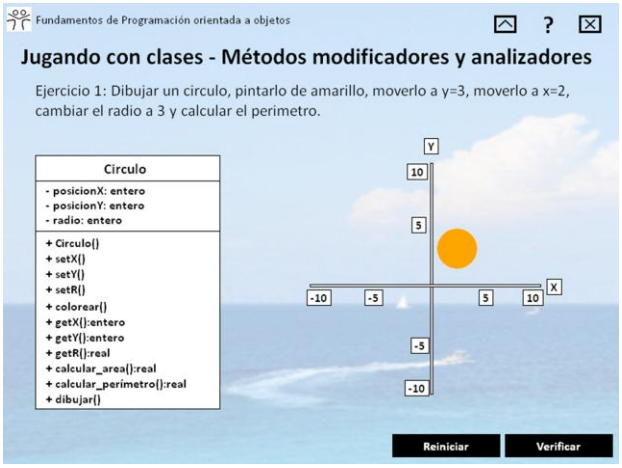

Figura 4.8: Pantalla típica de “Jugando con clases – Métodos modificadores y analizadores”

3.1.2 Nivel 2 – Herencia y polimorfismo

El nivel 2 de *CoquitoDobleO* permite interactuar con juegos que incentivan la comprensión de los conceptos de herencia y polimorfismo, como lo indica la figura 4.9.

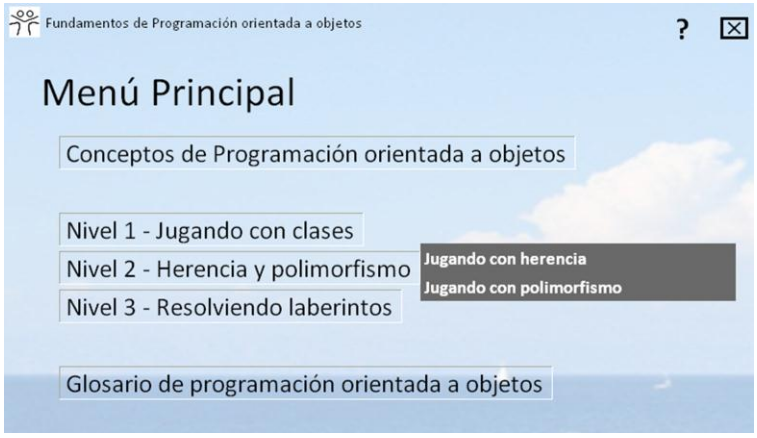

Figura 4.9: Sub-menús de la opción “Nivel 2 – Herencia y polimorfismo”

Al hacer clic sobre la opción “Jugando con herencia” se despliegan diagramas de clase jerárquicos, que contienen clases y subclases con algunos atributos y métodos, uno de éstos polimórfico que debe ser identificado por el jugador. Una de las pantallas de éste nivel se expone en la figura 4.10.

Figura 4.10: Una pantalla del nivel 2: Herencia y polimorfismo

Al hacer clic sobre la opción “Jugando con polimorfismo” se despliega una representación de la clase Vehículo con una serie de métodos, uno de éstos polimórfico que debe ser identificado por el jugador. En éste caso no se presenta un diagrama de clases que conlleve la herencia, como en el caso anterior. Una de las pantallas de éste nivel se expone en la figura 4.11.

Figura 4.11: Una pantalla del nivel 2: Polimorfismo

4.1.3 Nivel 3 – Resolviendo laberintos

El tercer y último nivel conlleva la ejecución de los métodos analizadores *avanzar()* y *girar()* para salir de un laberinto, afianzando de ésta forma los conceptos de método sobrecargado y razonamiento secuencial con toma de decisiones. Una pantalla típica de éste nivel se expone en la figura 4.12.

Figura 4.12: Pantalla típica del juego “Resolviendo laberintos”

En todos los niveles de juego, se ofrece la posibilidad de reiniciar el juego o de evaluar las jugadas realizadas hasta un determinado momento para determinar éxito o fracaso en el juego activo.

3.1.4 Éxitos y fracasos en los juegos

Cada vez que el usuario ejecuta un juego, tiene la posibilidad de evaluar si las acciones ejecutadas conducen al éxito o fracaso del juego. Así, en las figuras 4.13 y 4.14 se observan pantallazos de éxito en los juegos “Jugando con clases” y “Resolviendo laberintos”.

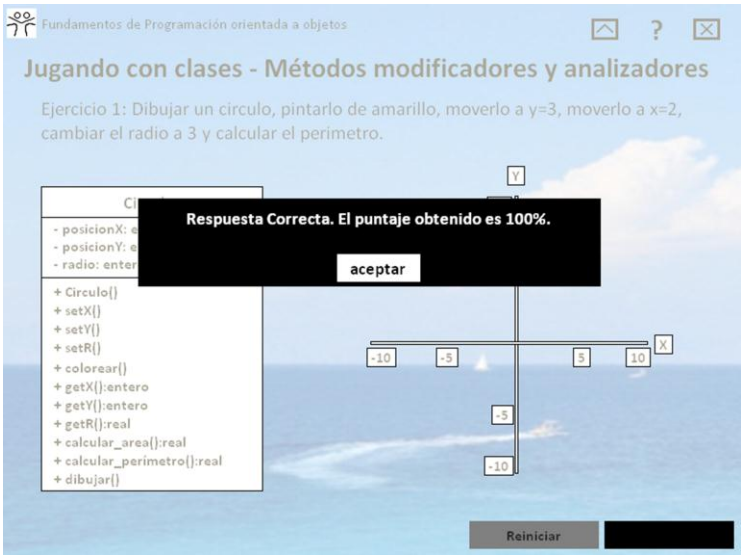

Figura 4.13: Mensaje final de éxito del juego “Jugando con clases”

Figura 4.14: Mensaje final de éxito de “Resolviendo laberintos”

De igual manera, las figuras 4.15 y 4.16 presentan pantallazos de fracaso en los juegos “Polimorfismo” y “Resolviendo laberintos”.

Figura 4.15: Mensaje final de fracaso del juego “Polimorfismo”

Figura 4.16: Mensaje final de fracaso de “Resolviendo laberintos”

La construcción del software *CoquitoDobleO* se concluyó de tal manera que el escalamiento referido a la agregación de nuevos subniveles a cada nivel de juego, se hiciera sencillo, dado que no es necesario modificar el código existente o escribir nuevas líneas en C#; sólo sería necesario adicionar los archivos de texto e imágenes pertinentes a las carpetas *Resources* generadas por *Visual Studio 2010* cuando se crea un proyecto de aplicación de *Windows Forms*.

4.2 Proceso de desarrollo del software *CoquitoDobleO*

El juego *CoquitoDobleO* se desarrolló en el lenguaje C#, bajo el ambiente integrado de desarrollo Microsoft Visual Studio 2010. Como se observa en el ítem anterior, el juego presenta una serie de pantallas 2D que despliegan las diferentes alternativas de juego, obtenidas a partir de una aplicación de Windows Forms.

El juego debe cumplir con los siguientes requerimientos de usuario, que en éste caso sería un profesor de un curso de iniciación a la programación orientada a objetos:

- Aclarar los conceptos fundamentales de programación orientada a objetos: clase, objeto, herencia, polimorfismo, atributo, métodos (constructores, modificadores y analizadores) y sobrecarga de métodos.
- Presentar juegos que permitan:
  - Crear objetos.
  - Modificar el estado de los objetos.
  - Ejecutar métodos analizadores.
  - Trabajar con herencia y polimorfismo.
  - Fortalecer el razonamiento secuencial.

4.2.1 Análisis

El análisis para el desarrollo del software lúdico se realizó en dos etapas:

- ✓ Atención a los requerimientos del cliente-profesor por medio de la aplicación de pruebas de software.
- ✓ Impacto en los estudiantes destinatarios del software por medio de la aplicación de rúbricas y encuestas.

Los requerimientos del cliente-profesor se resumen en diseñar juegos de computador que clarifiquen los conceptos de clase, objeto, sobrecarga de métodos, herencia y polimorfismo. Las pruebas realizadas se centraron en el análisis de riesgos respecto a los casos de uso del sistema:

- Un sub-juego no cumple con su objetivo específico.
- El resultado de un juego no refleja el conjunto de jugadas realizadas.
- Las salidas al menú principal o la finalización del juego no funcionan.

- El reinicio de un juego no blanquea todas las variables y resultados del juego anterior.
- El sistema de ayuda en o fuera de línea no funciona adecuadamente.

El control de estos riesgos asegura la fiabilidad de *CoquitoDobleO.*

La Tabla 4.1 presenta los requerimientos del juego.

Tabla 4.1: Requerimientos del juego *CoquitoDobleO*

ID	Descripción	Entradas	Salidas
R1	Aclarar y motivar el aprendizaje de los conceptos fundamentales de la programación orientada a objetos.	Archivos de texto con cuestionarios y respuestas.	Dos cuestionarios: uno con selección secuencial y otro con selección múltiple.
R2	Presentar juegos que permitan crear objetos, modificarles su estado y ejecutar métodos analizadores.	Recursos para los juegos: archivos de sonido, imagen y texto.	Juegos que interactúen con objetos.
R3	Presentar juegos para trabajar con herencia y polimorfismo.	Recursos para los juegos: archivos de sonido, imagen y texto.	Juegos con herencia y polimorfismo.
R4	Presentar juegos para fortalecer el razonamiento secuencial.	Recursos para los juegos: archivos de sonido, imagen y texto.	Juegos con razonamiento secuencial.

Los casos de uso base del sistema se observan en la Tabla 4.2.

Tabla 4.2: Casos de uso base del juego *CoquitoDobleO*

N°	Caso de uso	Descripción
1	Iniciar el juego.	El usuario (o jugador) hace clic en el archivo ejecutable de la aplicación.
2	Terminar el juego.	El jugador hace clic en el botón “X” del menú principal o de cualquier pantalla expuesta por el software, y responde “Si” al mensaje de verificación de salida del juego.
3	Seleccionar desde el menú principal la opción “Conceptos de programación orientada a objetos”.	El jugador decide no jugar, sino estudiar los conceptos de programación orientada a objetos.
4	Seleccionar desde el menú principal la opción de juego: “Nivel I – Jugando con clases”.	El usuario hace clic en la opción del menú principal que le permite jugar con clases.
5	Seleccionar desde el menú principal la opción de juego:”Nivel 2 – Herencia y polimorfismo”.	El usuario hace clic en la opción del menú principal que le permite jugar con Herencia y polimorfismo.
6	Seleccionar desde el menú principal la opción de juego: “Nivel 3 – Resolviendo laberintos”.	El usuario hace clic en la opción del menú principal que le permite jugar con laberintos.
7	Seleccionar desde el menú principal la opción “Glosario de programación orientada a objetos”.	El usuario hace clic en la opción del menú principal que le permita observar la ayuda disponible a nivel local o en línea.
8	Ver la ayuda.	El jugador hace clic en el botón “?” del menú principal o de cualquier pantalla expuesta por el software, con el objetivo de observar la ayuda disponible con respecto a la pantalla activa.
9	Reiniciar un juego.	El jugador hace clic en el botón “Reiniciar” para volver a ejecutar desde el principio un juego puesto en marcha.
10	Verificar las respuestas para un juego de nivel 2 ó 3.	El jugador verifica si cumplió con el objetivo del juego, haciendo clic en el botón “Verificar”.
11	Ejecutar las instrucciones de un juego de nivel 3.	El jugador ejecuta las instrucciones seleccionadas del juego de nivel 3 (Laberinto), haciendo clic en el botón “Ejecutar”.
12	Regresar al menú principal.	El usuario hace clic en el botón “^” para regresar al la pantalla del menú principal.

Existen algunos casos de uso que se extienden de los básicos; éstos se especifican en la Tabla 4.3.

Tabla 4.3: Casos de uso extendidos en el juego *CoquitoDobleO*

N°	Caso de uso extendido	Descripción	Caso base del cual se extiende
3.1	Escoger el subnivel “Selección secuencial”.	El jugador decide trabajar con preguntas de selección secuencial, es decir, selecciona el orden de los pasos para la solución de un problema.	Seleccionar desde el menú principal la opción “Conceptos de programación orientada a objetos”.
3.2	Escoger el subnivel “Selección múltiple”.	El jugador decide trabajar con preguntas de selección múltiple, es decir, sólo una opción es la correcta.	Seleccionar desde el menú principal la opción “Conceptos de programación orientada a objetos”.
4.1	Seleccionar el subnivel 1 del caso de uso N° 4 de la Tabla 4.2.	El usuario decide jugar con clases a un nivel básico, sin cambiar el estado de los objetos.	Seleccionar desde el menú principal la opción de juego: “Nivel I – Jugando con clases”.
4.2	Seleccionar el subnivel 2 del caso de uso N° 4 de la Tabla 4.2.	El usuario decide jugar con clases a un nivel avanzado, con la posibilidad de modificar el estado de los objetos.	Seleccionar desde el menú principal la opción de juego: “Nivel I – Jugando con clases”.
5.1	Seleccionar el subnivel 1 del caso de uso N° 5 de la Tabla 4.2.	El usuario decide jugar con herencia y polimorfismo.	Seleccionar desde el menú principal la opción de juego:”Nivel 2 – Herencia y polimorfismo”.
5.2	Seleccionar el subnivel 2 del caso de uso N° 5 de la Tabla 4.2.	El usuario decide jugar con Polimorfismo.	Seleccionar desde el menú principal la opción de juego:”Nivel 2 – Herencia y polimorfismo”.

Según la numeración de los casos de uso dada en las tablas 4.2 y 4.3, el diagrama de casos de uso para el juego *CoquitoDobleO* se observa en la figura 4.17.

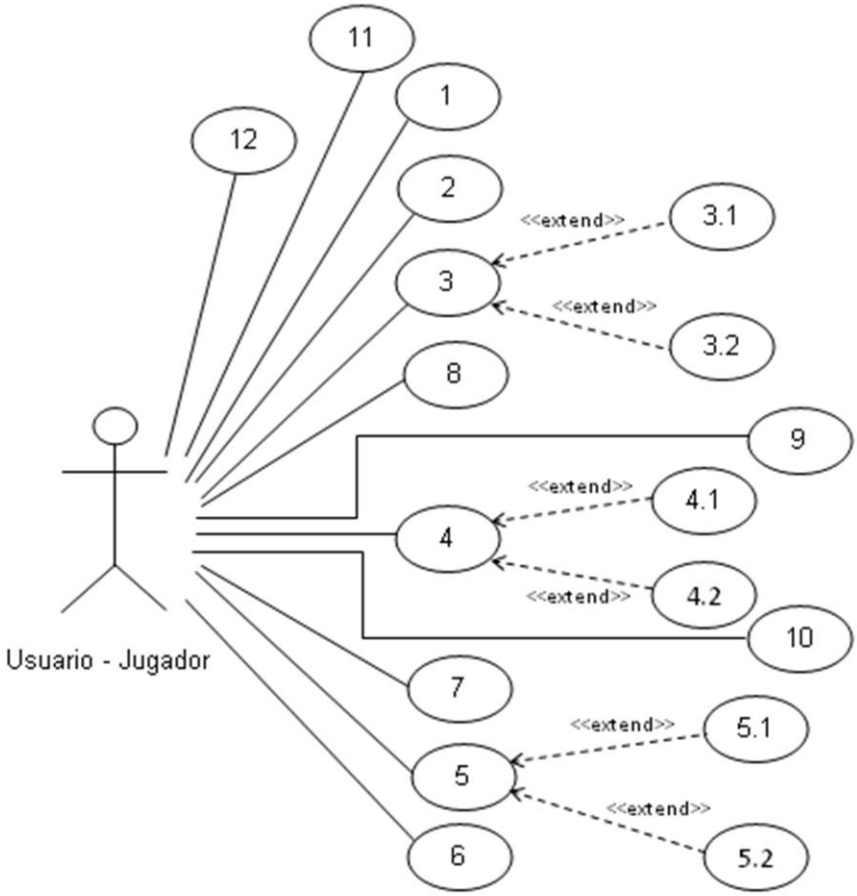

Figura 4.17: Diagrama de casos de uso para la aplicación *CoquitoDobleO*

4.2.2 Diseño y desarrollo

El desarrollo del software lúdico *CoquitoDobleO* se basó en una aplicación de Windows Forms escrita en lenguaje C# bajo el entorno integrado de desarrollo

Microsoft Visual Studio 2010, versión 10.0.30319.1 RTMRel, y con el Microsoft .NET Framework versión 4.0.30319 RTMRel, software disponible para estudiantes y profesores de la Facultad de Ingeniería del Tecnológico de Antioquia – Institución Universitaria, por el convenio de campus Agreement con Microsoft Colombia y la suscripción MSDN Academic Alliance.

El diagrama de clases conlleva a la clase principal de la aplicación, denominada Program, que contiene al método main( ), desde el cual se crea un objeto de la clase Form_presentacion, que lanza la aplicación *CoquitoDobleO*. El diagrama de clases de la aplicación generado desde el mismo entorno Visual Studio 2010, se observa en la figura 4.18.

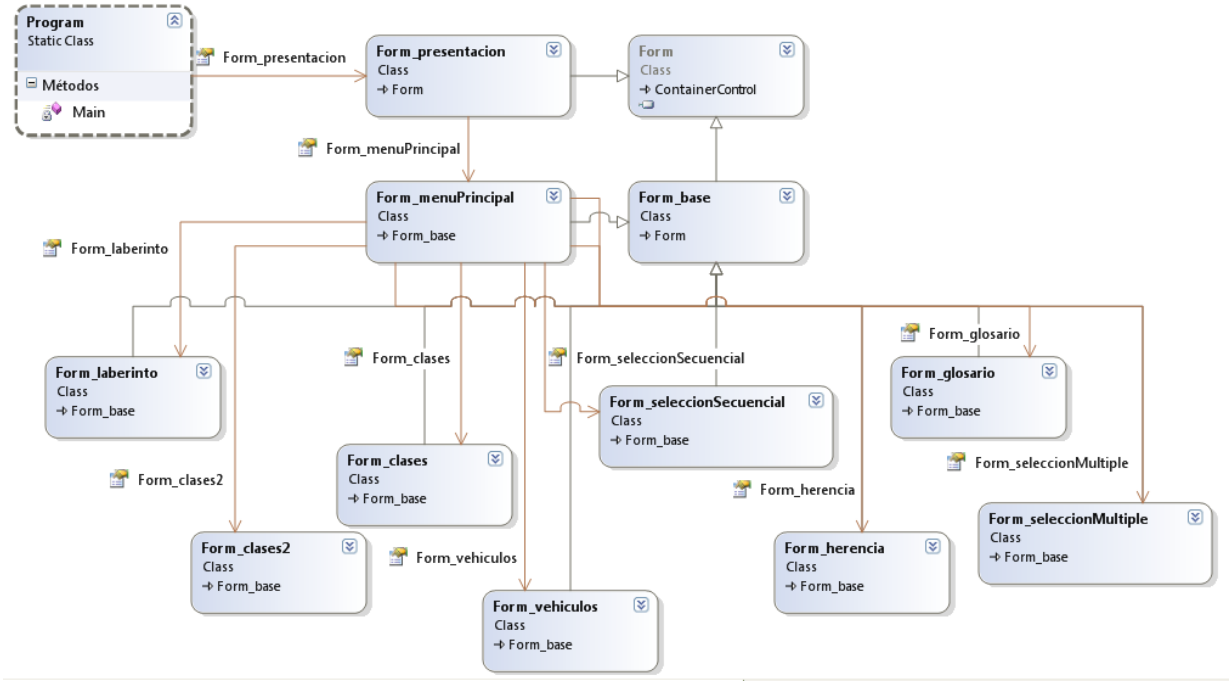

Figura 4.18: Diagrama de clases para la aplicación *CoquitoDobleO*

Capítulo 5

Experiencias de usuario con el juego *CoquitoDobleO*

Las experiencias de usuario con el juego *CoquitoDobleO* fueron efectuadas 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 éstas experiencias se trabajó con un método para el aprendizaje de la POO, denominado MIPSOO [Bot09], 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.

Otras propuestas metodológicas para la enseñanza y consecuente aprendizaje de la POO, han sido sustentadas por López [Lop07], Gayo et al [GCC03] y Villalobos-Casallas [VC07]. ¿Qué diferencias presentan éstas propuestas respecto a la expuesta en el presente documento, que combina MIPSOO y *CoquitoDobleO*?

Respecto al anterior interrogante, las tres propuestas carecen del valor agregado de la lúdica de juegos para aumentar los grados de motivación en el aprendizaje de la programación. Además, aunque la propuesta del profesor López maneja un seudo lenguaje parecido al planteado en MIPSOO, carece del diseño fundado en

los diagramas de clase y del análisis de las responsabilidades. El profesor Gayo y su equipo de colaboradores sustentan la adopción de la POO como primer y único paradigma en procesos de enseñanza-aprendizaje de la programación de computadoras, por ello plantean un interesante temario teórico-práctico para la asignatura “Introducción a la Programación” cimentado en el paradigma orientado a objetos; sin embargo, tampoco presentan las bondades de los diagramas y responsabilidades de las clases. Los profesores Villalobos y Casallas aplican un enfoque novedoso desde el punto de vista pedagógico y moderno a partir de lo tecnológico; sin embargo en su libro de texto “Fundamentos de programación. Aprendizaje activo basado en casos”, no enfatizan en el seudocódigo para la solución de problemas porque para la comprensión del texto no se requiere una formación específica previa y porque las competencias generadas con dicho libro se pueden enmarcar fácilmente en cualquier perfil profesional.

5.1 Descripción del experimento

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 = mc2, 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.” [Bot10]

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

a) Identificación de requerimientos

Se identifican dos requerimientos, descritos en la Tabla 5.1.

Tabla 5.1: Requerimientos para el problema “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 5.1 conlleva la definición de las abstracciones Energía y Proyecto, y a la reutilización de las clases de uso común Flujo y Mat.

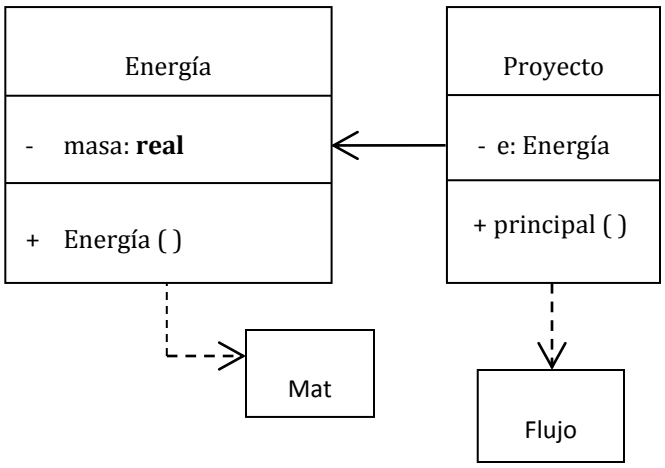

Figura 5.1: Diagrama de clases del problema “Ecuación de Einstein”

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 5.2 y 5.3 presentan los contratos de las clases de uso no común Energía y Proyecto.

Tabla 5.2: 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 5.3: 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 5.2 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 5.2: Seudo código del problema “Ecuación de Einstein”

5.2 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 [BT12]. 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.

5.2.1 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. Estos profesores desconocían el tipo de grupo a evaluar, es decir, no sabían si los alumnos que valoraban pertenecían al grupo de control o al experimental.

Los pares evaluadores para el desarrollo de las pruebas con los grupos de educación superior fueron la doctora Raquel Anaya de la Universidad EAFIT y los magíster en ingeniería Adriana Xiomara Reyes del Politécnico Colombiano Jaime Isaza Cadavid y Juan Camilo Giraldo del Tecnológico de Antioquia Institución Universitaria; con los grupos de educación media técnica colaboraron los ingenieros Liliana García de la Institución Educativa Gabriel García Márquez y

Wilmar Castañeda y Rita Ligia Osorio de la Institución Educativa Comercial de Envigado.

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 [PB05], 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)

La estructura completa de la rúbrica se ilustra en la Tabla 5.4.

Tabla 5.4. Rúbrica para evaluar una exposición

TECNOLOGICO DE ANTIOQUIA
Lógica de Programación I
Rúbrica para evaluar una exposición
Mayo de 2011

Nombres: _____ , _____ , _____ .

Criterios	Ingenuo (1.25)	Novato (2.50)	Aprendiz (3.75)	Experto (5.00)	N. A.	Puntaje (1 a 5)
1. Diferencia entre los conceptos de clase y objeto, y comprende el ciclo de vida de éste último.	No distingue entre los conceptos de objeto y clase. ◯	Tiene en cuenta (o considera) los conceptos de objeto y clase, pero no sabe distinguirlos. ◯	Define clases e instancia objetos, pero a veces se confunde en el proceso del ciclo de vida de un objeto. ◯	Establece una clara diferencia entre una clase y sus objetos, y comprende el ciclo de vida de un objeto. ◯		
2. Aplicación de conceptos fundamentales del paradigma de programación orientado a objetos (arreglos de objetos ó herencia ó polimorfismo).	No aplica los conceptos fundamentales del paradigma de programación orientado a objetos. ◯	Tiene en cuenta (o considera) los conceptos de la orientación a objetos, pero no sabe aplicarlos. ◯	Considera los conceptos de la teoría orientada a objetos, pero le falta alguna claridad al aplicarlos. ◯	Considera y aplica con claridad y pertinencia los conceptos fundamentales del paradigma de programación orientado a objetos. ◯		
3. Seguridad en la exposición, uso de vocabulario apropiado, buena pronunciación y modulación.	Expone con inseguridad, utiliza vocabulario inapropiado, pronuncia y modula sin claridad. ◯	No siempre es seguro, utiliza un vocabulario limitado, pronuncia y modula con poca claridad. ◯	Es seguro pero utiliza un vocabulario inapropiado. A veces no pronuncia o modula de forma adecuada. ◯	Actua con seguridad en la exposición, utiliza un amplio vocabulario, pronuncia y modula correctamente las palabras. ◯		
4. Calidad de las diapositivas.	No presenta diapositivas ni otro material didáctico. ◯	Presenta diapositivas con un contenido inadecuado y un diseño poco atractivo para la audiencia. ◯	Presenta diapositivas con un contenido inadecuado, pero atractivas en diseño para la audiencia (o viceversa). ◯	Las diapositivas son ricas en contenido y con un diseño atractivo para la audiencia. ◯		
5. Ejemplo sobre el tema expuesto.	No presenta un ejemplo o tema donde se aplique el tema. ◯	Presenta un ejemplo, pero no aplica adecuadamente los conceptos relacionados con el tema. ◯	Presenta un ejemplo donde aplica los conceptos relacionados con el tema, pero le falta claridad. ◯	Presenta un ejemplo claro y pertinente, donde se aplican los conceptos del tema. ◯		
6. Solución del ejemplo.	No aplica las fases para la solución de un problema. ◯	Aplica de manera inapropiada los pasos para la solución de un problema. ◯	Aplica los pasos para la solución de un problema, pero falla en alguno(s) de ellos. ◯	Aplica los pasos para la solución de un problema, de manera clara y metódica. ◯		

Puntuación total: _____

Tema de la exposición: ◯ Arreglos de objetos ◯ Herencia ◯ Polimorfismo

Firma del evaluador: _____

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.).

En el diligenciamiento de las rúbricas los expertos evaluadores tuvieron en cuenta las siguientes consideraciones:

- ✓ Se calificará a un equipo de expositores (dos o tres estudiantes) y no de manera individual. Esto implica que el evaluador debe clasificar a un equipo de expositores en uno de los cuatro niveles de comprensión para un determinado criterio.

- ✓ El puntaje asignado en la última columna será un entero entre 1 y 5, de tal manera que el criterio evaluado con más alto valor será *Experto* con un puntaje de 5, y el peor evaluado será *Ingenuo* con puntaje de 1. Si un criterio se evalúa en el nivel *Ingenuo*, *Novato* o *Aprendiz* con un puntaje alto, significa que se encuentra cercano al próximo nivel de comprensión. Por ejemplo, si un criterio se califica como *Ingenuo* con puntaje 5, significa que está cercano a *Novato* con puntaje 1, si se evalúa como *Aprendiz* 5 estaría cercano a *Experto* 1, que constituye el nivel de experto más "novato". Un *Experto* 5 es un "experto de expertos". Un puntaje de 3 en cualquier nivel se puede interpretar como el estado intermedio del nivel; así, un *Aprendiz* 3 es un aprendiz "estándar".

- ✓ 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, $i=1, 2, ..., 6$.

n_i:nivel seleccionado para el criterio i.

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

Los criterios fueron seleccionados atendiendo aspectos de conocimiento/contenido (criterios 1 y 2), formas de comunicación (c. 3 y 4), propósito (c. 5) y método (c. 6).

La caracterización de ambos grupos, relacionada con su tamaño, género de sus integrantes y edad, se observa en la Tabla 5.5.

Tabla 5.5. 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

5.2.2 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 [Fer07] y dos preguntas abiertas, como se observa en la Tabla 5.6. Cabe anotar que las aserciones relacionadas con el software *CoquitoDobleO* no fueron planteadas al grupo de control.

Tabla 5.6. Encuesta aplicada al grupo experimental

TECNOLÓGICO DE ANTIOQUIA
Facultad de Informática

Encuesta sobre el curso Lógica de Programación I, grupo 02
Mayo de 2011

Nombre: _____

Edad: _____ Género: M ☐ F ☐

Marque con una X el rectángulo que considere apropiado.

Aserción	Completamente de acuerdo	De acuerdo	Indeciso	En desacuerdo	Completamente en desacuerdo
1. Los conceptos estudiados sobre programación orientada a objetos (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 programación orientada a objetos 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	○	○	○	○	○

9. ¿Qué fue lo que más le agradó del curso? Y en éste mismo sentido, ¿Qué fue lo que más le desagradó?

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 descripción completa de las dos encuestas, dirigidas a los grupos experimental y de control, se presenta en el apéndice B.

5.3 Resultados y evaluación

En la Tabla 5.7 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 [TA11].

Tabla 5.7. 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

Como es de esperar, los puntajes de los grupos experimental y de control en educación superior superan los de media técnica. Un análisis de estos puntajes por tema, discriminados según el nivel educativo, se presentan en las tablas 5.8 y 5.9, donde se han agregado algunas medidas de tendencia central como la media aritmética y la desviación típica, además del resultado de la prueba t de Student para la comparación de los promedios obtenidos.

Tabla 5.8. Puntajes promedio por tema en educación superior

Tema	Grupo de control	Grupo experimental
Arreglos de objetos	1,52	2,53
Herencia	1,17	2,08
Polimorfismo	1,64	1,82
Promedio:	1,44	2,14
Desviación estándar:	0,24	0,36
Prueba t de Student:	0,03	

Tabla 5.9. Puntajes promedio por tema en educación media técnica

Tema	Grupo de control	Grupo experimental
Arreglos de objetos	1,50	2,00
Herencia	1,15	1,85
Polimorfismo	1,52	1,80
Promedio:	1,39	1,88
Desviación estándar:	0,21	0,10
Prueba t de Student:	0,02	

La prueba t de Student para ambos niveles educativos se aplicó con los siguientes parámetros:

Prueba_T (GC, GE, 1, 3), donde

GC: primer conjunto de datos, correspondiente al grupo de control.

GE: segundo conjunto de datos, correspondiente al grupo experimental.

1: número de colas de la distribución.

3: indica una prueba t de Student heteroscedástica, porque las muestras tienen varianzas diferentes.

Estos resultados se encuentran representados gráficamente en los diagramas de barras en las figuras 5.3 y 5.4.

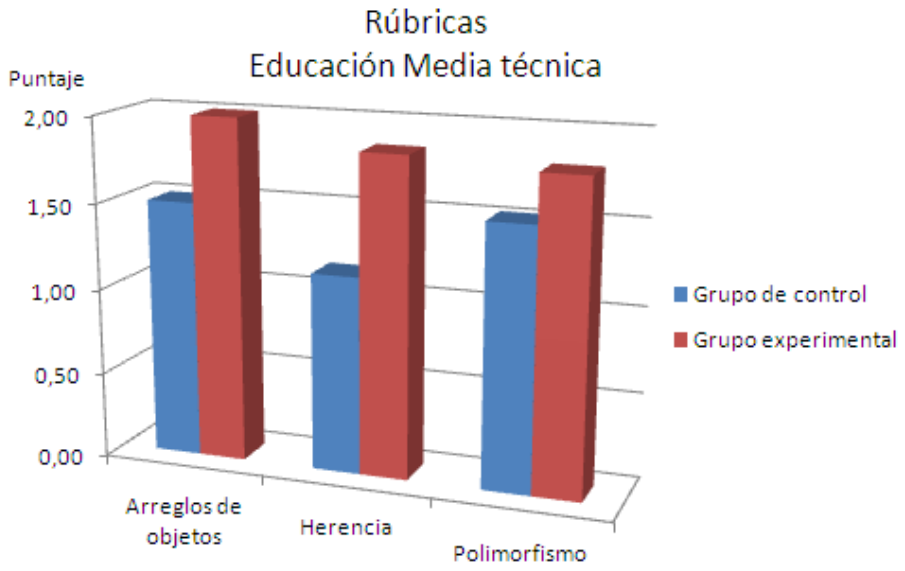

Figura 5.3: Resultados de las rúbricas para educación media técnica

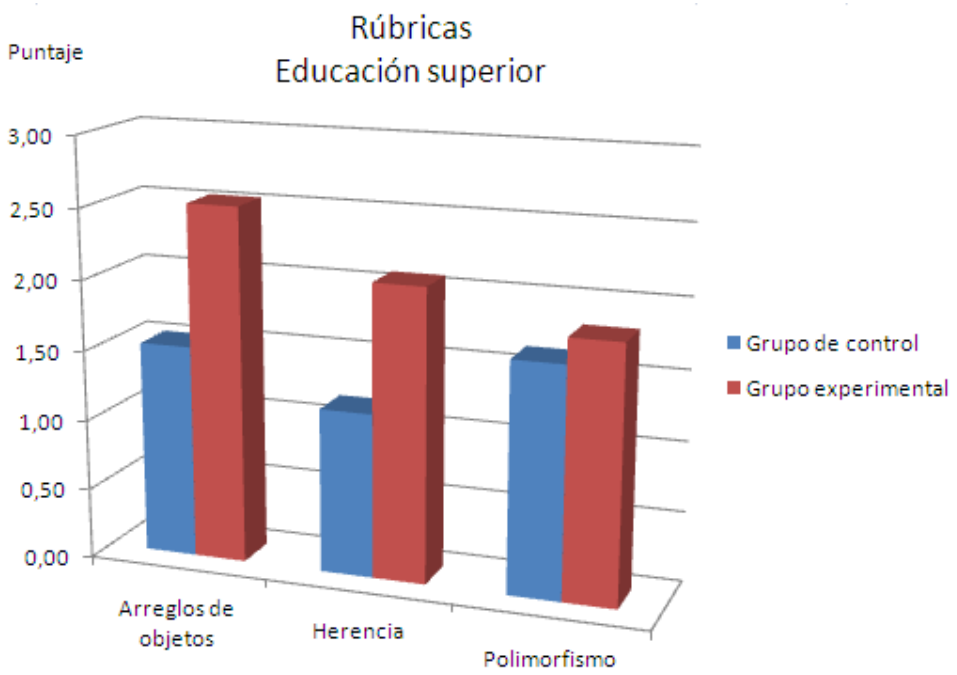

Figura 5.4: Resultados de las rúbricas para educación superior

El consolidado de respuestas a las encuestas se presenta en la Tabla 5.10, donde se exponen dos números separados por guion; 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 5.10. 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	21-6	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	16-10	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 documento, 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.

Algunas respuestas a las dos preguntas abiertas fueron las siguientes, donde cada respuesta *Ri, i = 1, 2,…, 10*, corresponde al mismo estudiante.

- *¿Qué fue lo que más le agradó del curso? Y en éste mismo sentido, ¿Qué fue lo que más le desagradó?*

  R1/ Para mí no hubo nada desagradable, el profesor hacía las cosas todo interesante.

  R2/ Pienso que lo que más me agradó fue que entendí bien los conceptos de programación orientada a objetos y voy preparado para el segundo semestre.

R3/ Todo fue muy agradable, los métodos de aprendizaje y el dinamismo que manejó el profesor.

R4/ Me agradó aprender nuevas cosas. Me desagradó la velocidad con que el profesor dicta sus clases.

R5/ Me agradó la temática y metodología utilizada por parte del docente. Me gustaría utilizar más elementos tecnológicos que faciliten el aprendizaje.

Profe: felicitaciones! Excelente su labor.

R6/ Lo mejor fue el conocimiento recibido por parte del profesor. Lo que no gusta mucho es la falta de práctica en las salas de sistemas.

R7/ Lo que más me agradó fue el desarrollo de ejercicios en clase para el mejor aprendizaje y lo que más me desagradó fue la falta de práctica, pues sin esto la clase es monótona y aburridora.

R8/ Me agradó el tema de arreglos, polimorfismo, y programación orientada a objetos.

R9/ Me agradó mucho los compañeros y las clases del profe en las que utilizamos ejemplos.

R10/ Me gustó que era un tema nuevo interesante y fundamental para nuestra carrera. En desacuerdo, nada.

- *¿Qué recomendaciones tiene para mejorar el curso de Lógica de Programación I?*

R1/ Seguir con la misma metodología, que siga siendo así de interesante.

R2/ Más clases en sistemas para coger más práctica

R3/ No tengo recomendación, todo me pareció excelente.

R4/ Me gustaría que el curso fuera más práctico y menos teórico.

R5/ Utilizar un mayor número de elementos tecnológicos.

R6/ Más prácticas, más lúdicas para mejorar el aprendizaje.

R7/ Más práctica y ayudas esenciales en la clase sobre el tema.

R8/ Pues que hagamos más talleres.

R9/ Más práctica en sala, aplicando los seudocódigos al programa.

R10/ Que usáramos más programas como “Java”, “C++”, etc.

Capítulo 6

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 [Bot09], 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”, 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 5.7, el tema con mayor dificultad de comprensión para los grupos de control es la herencia, mientras que para los grupos experimentales es el polimorfismo. Un análisis de los datos esta de las tablas 5.7, 5.8 y 5.9, muestra que 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. Los resultados de las pruebas t de Student aplicadas también apoyan ésta conclusión.

- ✓ Según la Tabla 5.10, 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 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.

- ✓ Es conveniente ampliar el periplo experimental del método para la enseñanza y consecuente aprendizaje de la programación orientada a objetos, que comprende cuatro fases, combinado con el juego digital *CoquitoDobleO*. El método, hoy aplicado por la Facultad de Ingeniería de la Institución Universitaria Tecnológico de Antioquia, se puede extender a otras instituciones de educación superior que ofrezcan programas de Tecnología en Sistemas, Ingeniería en Software o programas afines. El juego *CoquitoDobeO* se puede promover desde los semilleros de investigación, donde también se interactúe con otros juegos como *Scratch* y *RoboMind*, o con entornos integrados de desarrollo como *Greenfoot*, *BlueJ* o *Robocode,* que también contienen elementos lúdicos para el aprendizaje de la programación de computadoras.

Apéndice A

Rúbrica para evaluar una exposición

Cada profesor de educación superior como de educación media técnica, diligenció la siguiente rúbrica para evaluar la exposición de los estudiantes.

TECNOLOGICO DE ANTIOQUIA
Lógica de Programación I
Rúbrica para evaluar una exposición
Mayo de 2011

Criterios	Ingenuo (1.25)	Novato (2.50)	Aprendiz (3.75)	Experto (5.00)	N. A.	Puntaje (1 a 5)
1. Diferencia entre los conceptos de clase y objeto, y comprende el ciclo de vida de éste último.	No distingue entre los conceptos de objeto y clase. ◯	Tiene en cuenta (o considera) los conceptos de objeto y clase, pero no sabe distinguirlos. ◯	Define clases e instancia objetos, pero a veces se confunde en el proceso del ciclo de vida de un objeto. ◯	Establece una clara diferencia entre una clase y sus objetos, y comprende el ciclo de vida de un objeto. ◯		
2. Aplicación de conceptos fundamentales del paradigma de programación orientado a objetos (arreglos de objetos ó herencia ó polimorfismo).	No aplica los conceptos fundamentales del paradigma de programación orientado a objetos. ◯	Tiene en cuenta (o considera) los conceptos de la orientación a objetos, pero no sabe aplicarlos. ◯	Considera los conceptos de la teoría orientada a objetos, pero le falta alguna claridad al aplicarlos. ◯	Considera y aplica con claridad y pertinencia los conceptos fundamentales del paradigma de programación orientado a objetos. ◯		
3. Seguridad en la exposición, uso de vocabulario apropiado, buena pronunciación y modulación.	Expone con inseguridad, utiliza vocabulario inapropiado, pronuncia y modula sin claridad. ◯	No siempre es seguro, utiliza un vocabulario limitado, pronuncia y modula con poca claridad. ◯	Es seguro pero utiliza un vocabulario inapropiado. A veces no pronuncia o modula de forma adecuada. ◯	Actua con seguridad en la exposición, utiliza un amplio vocabulario, pronuncia y modula correctamente las palabras. ◯		
4. Calidad de las diapositivas.	No presenta diapositivas ni otro material didáctico. ◯	Presenta diapositivas con un contenido inadecuado y un diseño poco atractivo para la audiencia. ◯	Presenta diapositivas con un contenido inadecuado, pero atractivas en diseño para la audiencia (o viceversa). ◯	Las diapositivas son ricas en contenido y con un diseño atractivo para la audiencia. ◯		
5. Ejemplo sobre el tema expuesto.	No presenta un ejemplo o tema donde se aplique el tema. ◯	Presenta un ejemplo, pero no aplica adecuadamente los conceptos relacionados con el tema. ◯	Presenta un ejemplo donde aplica los conceptos relacionados con el tema, pero le falta claridad. ◯	Presenta un ejemplo claro y pertinente, donde se aplican los conceptos del tema. ◯		
6. Solución del ejemplo.	No aplica las fases para la solución de un problema. ◯	Aplica de manera inapropiada los pasos para la solución de un problema. ◯	Aplica los pasos para la solución de un problema, pero falla en alguno(s) de ellos. ◯	Aplica los pasos para la solución de un problema, de manera clara y metódica. ◯		

Firma del evaluador: _____

Puntuación total: _____

Tema de la exposición: ◯ Arreglos de objetos ◯ Herencia ◯ Polimorfismo

Apéndice B: Encuestas

B.1 Encuesta grupo experimental

TECNOLÓGICO DE ANTIOQUIA
Facultad de Informática

Encuesta sobre el curso Lógica de Programación I, grupo 02
Mayo de 2011

Nombre: _____

Edad: _____ Género: M ☐ F ☐

Marque con una X el rectángulo que considere apropiado.

Asercion	Completamente de acuerdo	De acuerdo	Indeciso	En desacuerdo	Completamente en desacuerdo
1. Los conceptos estudiados sobre programación orientada a objetos (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 programación orientada a objetos 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	◯	◯	◯	◯	◯

9. ¿Qué fue lo que más le agradó del curso? Y en éste mismo sentido, ¿Qué fue lo que más le desagradó?

10. ¿Qué recomendaciones tiene para mejorar el curso de Lógica de Programación I?

Responda las preguntas abiertas 9 y 10 detrás de esta hoja.

B.2 Encuesta grupo de control

TECNOLÓGICO DE ANTIOQUIA
Facultad de Informática

Encuesta sobre el curso Lógica de programación I, grupo 01
Mayo de 2011

Nombre: _____

Edad: _____ Género: M ☐ F ☐

Marque con una X el rectángulo que considere apropiado.

Aserción	Completamente de acuerdo	De acuerdo	Indeciso	En desacuerdo	Completamente en desacuerdo
1. Los conceptos estudiados sobre programación orientada a objetos (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. La evaluación con exámenes teóricos individuales y en equipo es apropiada para un curso de Lógica de Programación I.	◯	◯	◯	◯	◯
4. Los talleres impresos y remitidos por correo electrónico contribuyeron a mejorar el proceso de aprendizaje de la asignatura.	◯	◯	◯	◯	◯
5. Los problemas resueltos en clase sirvieron para mejorar la comprensión de la programación orientada a objetos.	◯	◯	◯	◯	◯
6. 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	◯	◯	◯	◯	◯

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

Para las respuestas a las preguntas abiertas 7 y 8, utilice el envés de la hoja.

Bibliografía

[Alb08] Alba, R. *Iniciándose en la programación con Scrath*, Observatorio Tecnológico, Gobierno de España, Ministerio de Educación., 2008. Online: http://recursostic.educacion.es/observatorio/web/ca/software/programacion/619-iniciandose-en-la-programacion-con-scratch

[BA07] Baibak, T., Agrawal, R. *Programming games to learn algorithms*. 2007 ASEE Annual Conference & Exposition. Honolulu, Hawaii, 2007.

[Bot09] Botero, R. et al. *Lógica y programación orientada a objetos: un enfoque basado en problemas*. CITIA, Tecnológico de Antioquia, Medellín, 2009.

[Bot10] Botero, R. *Patrones Grasp y Anti-Patrones: un Enfoque Orientado a Objetos desde Lógica de Programación*. Revista Entre Ciencia e Ingeniería, Universidad Católica de Pereira, año 4. No. 8, 2010.

[Bue01] Bueno, D. et al. *Aprendizaje lúdico en laboratorio de programación. VII Jornadas sobre la Enseñanza Universitaria en Informática*. Palma de Mallorca, 2001.

[BT11] Botero, R., Trefftz, H. *Medra para el aprendizaje en lógica de programación orientada a objetos mediante un juego*. Revista Investigación en Ingeniería de Sistemas e Informática N°1, Universidad Pedagógica y Tecnológica de Colombia – UPTC, Tunja, Boyacá, 2011.

[BT12] Botero, R., Trefftz, H.. *Aprendizaje de la programación orientada a objetos: experiencias en educación media técnica y superior.* LACCEI 2012, Ciudad de Panamá. In press, 2012.

[Cad06] Cadavid,H. *Desarrollo de juegos como base para la compresión de temas fundamentales de la programación orientada a objetos*. VIII Congreso Colombiano de Informática Educativa. Cali, Colombia, 2006.

[Chi08] Chicharro et al. *Júpiter: Un entorno web para el aprendizaje basado en juegos competitivos***.** Actas de Mundo Internet 08, XII Congreso Internacional de Internet y Sociedad de la Información, Málaga, 2008.

[DCP07] Dann, W., Cooper, S., Pausch, R. (), *Learning to Program with Alice*, Pearson Prentice Hall, Upper Saddle River, 2007.

[Den05] Denault, A. *Minueto, an undergraduate teaching Development framework*. Tesis presentada en cumplimiento parcial de los requisitos del grado de Maestría en Ciencias, Facultad de Ciencias de la Computación de la Universidad McGill, Montreal, 2005.

[DS04] Dorn, B., Sanders, D. *Using Jeroo to introduce object-oriented programming.* 33rd ASEE/IEEE Frontiers in Education Conference, Boulder, Colorado, 2004.

[EL08] Edgington, J., Leutenegger, S. *Using the ancient game of rogue in CS1*. J. Comput. Small Coll. 24, 1 (October 2008), 150-156.

[Fer07] Fernández de Pinedo, Ignacio. *NTP 15: Construcción de una escala de actitudes tipo Likert*. Centro de Investigación y Asistencia Técnica – Barcelona. Ministerio de Trabajo y Asuntos Sociales – España, 2007.

[Fri08] Friss, I. *Scratch: Applications in Computer Science 1.*38th ASEE/IEEE Frontiers In Education Conference, Saratoga Springs, NY, 2008.

[GB05] Goschnick, S., Balbo, S. *Game-first programming for information systems students*. . In *Proceedings of the second Australasian*

conference on Interactive entertainment (IE 2005). Creativity \& Cognition Studios Press, Sydney, Australia, Australia, 71-74.

[GCC03] Gayo D., Cernuda del Río, A., Cueva J. M., Díaz, M., García, Mª P. A., Redondo J. M. *Reflexiones y experiencias sobre la enseñanza de POO como único paradigma*. IX Jornadas de Enseñanza Universitaria de la Informática (JENUI 2003), Universidad de Cádiz.

[GM03] Gómez – Martín, M. et al. *Aprendizaje basado en juegos*. Departamento de Sistemas Informáticos y programación, Universidad Complutense de Madrid, 2003.

[Har04] Hartness, K. *Robocode: using games to teach artificial intelligence*. Journal of Computing Sciences in Colleges. Volume 19 Issue 4, April 2004.

[Kee06] Keefe, K. et al. *Adopting xp practices for teaching object oriented programming*. ACE '06 Proceedings of the 8th Australasian Conference on Computing Education - Volume 52, 2006.

[KH08] Kölling, M. *Greenfoot: a highly graphical ide for learning object-oriented programming*. In Proceedings of the 13th annual conference on Innovation and technology in computer science education (ITiCSE '08). ACM, New York, NY, USA, 327-327. 2008.

[Kim06] Kim, S. et al. *Smalltalk Card Game for Learning Object-Oriented Thinking in an Evolutionary Way*. In *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications* (OOPSLA '06). ACM, New York, NY, USA, 683-684. 2006.

[Kou07] Kouznetsova, S. *Using BlueJ and Blackjack to teach object-oriented design concepts in CS1***.** Journal of Computing Sciences in Colleges. Volume 22 Issue 4, April 2007.

[KUW03] Kobayashi, K., Uchida, Y., Watanabe K. *A study of battle strategy for the Robocode*. SICE Annual Conference in Fukui, Fukui University, 2003.

[Lob00] Lobo, A. et al. *Using real-world objects to motivate OOP in a CS1 lab*. In Proceedings of the fifth annual CCSC northeastern conference on The journal of computing in small colleges (CCSC '00), John G. Meinke (Ed.). Consortium for Computing Sciences in Colleges, , USA, 144-156. 2000.

[Lop07] López, L. *Metodología para la enseñanza-aprendizaje de la lógica de la programación orientada a objetos*. XIII Jornadas de Enseñanza Universitaria de la Informática (JENUI 2007), Teruel.

[LKG07] Scratch. The Lifelong Kindergarten Group , MIT Media Lab, http://scratch.mit.edu/, 2007.

[Mck07] McKenzie, W.B. *An alternative approach to introductory object oriented programming: a case study in programming with Alice.* Proceedings of the AIS SIG-ED IAIM 2007 Conference, Montreal, Quebec, 2007.

[MM09] Moreno, J., Montaño, E. *ProBot: Juego para el aprendizaje de lógica de programación.* En J. Sánchez (Ed.): Nuevas Ideas en Informática Educativa, Volumen 5, pp. 1-7, Santiago de Chile, 2009.

[PB05] Perkins, D., Blythe, T. *Ante todo, la comprensión*. Revista Magisterio, Educación y Pedagogía. Cooperativa Editorial Magisterio. N° 15, abril 2005.

[Pat81] Pattis, R. *Karel the Robot: A Gentle Introduction to the Art of Programming with Pascal*, John Wiley & Sons, Inc., ISBN: 0471597252, 1981.

[Rec06] Recio, J. et al. *Aprendizaje de técnicas avanzadas de Programación Orientada a Objetos mediante programación de juegos*. XII Jornadas de Enseñanza Universitaria de la Informática (JENUI 2006), Bilbao.

[RLG07] Rankin, Y., Lechner, T., Gooch B. *Team-based pedagogy for CS102 using game design***.** The 34th International Conference and Exhibition on Computer Graphics and Interactive Techniques – SIGGRAPH '07, San Diego, California. 2007.

[RM05] RoboMind. *Programming structures*. Research Kitchen, http://www.robomind.net/en/docProgrammingStructures.htm, 2005.

[Rod08] Rodríguez-Lozada, D. *Enseñanza de programación orientada a objetos mediante el desarrollo de aplicaciones graficas interactivas*. VIII Congreso TAEE Universidad de Zaragoza, España, 2008.

[Ryo08] Ryoo, J. et al. *Teaching Object-Oriented Software Engineering Through Problem-Based Learning in the Context of Game Design*. Proceedings of the 21st onference on Software Engineering Education and Training - Charleston, SC.. Apr. 2008.

[SN02] Schulte, C. and Niere, J. *Thinking in Object Structures: Teaching Modelling in Secondary Schools*. In: Proceedings of the ECOOP Workshop on Pedagogies and Tools for Learning Object-Oriented Concepts, Spain, 2002.

[Som05] Sommerville, I. *Ingeniería del Software*. Séptima edición. Pearson Educación, S. A., Madrid 2005.

[SS07] Singh R. and Singh, J. Learning Computer Programming Using A Board Game - Case Study On C-Jump. Masters thesis, Multimedia University, 2007.

[TA11] Tecnológico de Antioquia – Institución Universitaria. Online: http://www.tdea.edu.co

[VC07] Villalobos, J., Casallas, R. *Fundamentos de programación. Aprendizaje activo basado en casos*. Pearson – Prentice Hall, México, 2007.

[YL10] Yuen, T., Liu, M. 2010. *How interactive multimedia authoring transforms object-oriented thinking*. In Proceedings of the 41st ACM technical symposium on Computer science education (SIGCSE '10). ACM, New York, NY, USA, 426-430. 2010.

[Zap06] Zapata-Santillana, E. Coquito clásico. Lectura inicial. Ediciones Coquito, 2006.