Método integrado de programación secuencial y programación orientada a objetos para el análisis, diseño y elaboración de algoritmos - MIPSOO

Autores

Ricardo de Jesús Botero Tabares
Tecnológico de Antioquia
Profesor de Tiempo Completo
rbotero@tdea.edu.co

Carlos Arturo Castro Castro
Universidad de San Buenaventura, Medellín
Docente-Investigador
carlos.castro@usbmed.edu.co

José Eucario Parra Castrillón
Fundación Universitaria Católica del Norte
Coordinador de Ingeniería Informática
jeparra@ucn.edu.co

Contenido

A modo de diálogo interinstitucional y divulgación de actividades científicas, el Grupo de Investigación en Ingeniería del Software del Tecnológico de Antioquia-GIISTA, comparte con los lectores de la *Revista Virtual de la Fundación Universitaria Católica del Norte* el siguiente proyecto de investigación.

Resumen: En la presente investigación se propone una metodología para facilitar el proceso enseñanza-aprendizaje en los cursos de educación superior de lógica de programación (algoritmos), mediante la integración de los paradigmas estructurado (procedimental) y orientado a objetos.

La metodología toma características asociadas a lenguajes de programación de la familia C / C++ / Java y de Lenguaje de Modelado Unificado (UML); incluye además elementos sintácticos nuevos que posibilitan la integración entre los dos paradigmas y apoya en gran medida la enseñanza temprana de la programación orientada a objetos.

La metodología propuesta posee soporte, en primer lugar, en un trabajo de campo realizado mediante resultados de encuestas a profesores, estudiantes y administrativos de diversas instituciones de educación superior del departamento de Antioquia- Colombia; en segundo lugar, en componentes pedagógicos del modelo constructivista y en tercer lugar, en una estructura didáctica elaborada a partir de la planeación de un microcurrículo apropiado a la metodología.

Palabras clave: Algoritmos, Ingeniería del software, Lógica de Programación, Programación Orientada a Objetos.

Abstract: In the present investigation a methodology is proposed to facilitate the education-learning process in higher education

Programming Logic courses (algorithms) by means of the integration of the structured programming and object oriented paradigms.

The methodology uses characteristics associated with programming languages of the C/C++/Java family and the Unified Modeling Language (UML); it includes in addition new syntactic elements that make integration between both paradigms possible and supports the early education of the object-oriented programming.

The proposed methodology is based on field work carried out by means of surveys of teachers, students and administrative personal from diverse higher education institutions in the department of Antioquia-Colombia; secondly in pedagogical components of the constructive model and thirdly, in didactical structures elaborated from the planning of micro curriculum appropriate to the methodology.

Key words: Algorithms, Programming Logic, software Engineering, Object-oriented programming.

0. Introducción

En el contexto de la informática, la forma de resolver un problema mediante un algoritmo computacional ha trascendido de la aplicación de técnicas procedimentales estructuradas a técnicas orientadas a objetos. Este cambio se debe a la evolución de los lenguajes de programación como respuesta a las problemáticas que generó la crisis del software agudizado en las postrimerías del siglo XX.

La influencia que ha tenido el paradigma de programación orientado al objeto en los sectores académico y productivo para el análisis, diseño y desarrollo de software, es evidente, máxime cuando ha generado estándares internacionales para la producción de software escalable de alto rendimiento, basados en la aplicación de los conceptos fundamentales

asociados al paradigma: abstracción, reutilización, modularidad y extensibilidad.

La orientación a objetos forma parte del núcleo de conocimientos de la informática. En las propuestas curriculares ACM (Association for Computing Machinery), IEEE-CS (Computer Society of the Institute for Electrical and Electronic Engineers) y AIS (Association for Information Systems) [1] se consideran conocimientos obligatorios sobre los conceptos de la orientación a objetos para la escritura de programas, proceso basado en UML y patrones de diseño.

Las instituciones educativas que ofrecen programas tecnológicos o profesionales relacionados con informática o que imparten cursos de introducción a las ciencias de la computación, están abocadas a la actualización continua de sus planes de estudio teniendo en cuenta los adelantos científicos y tecnológicos.

Dichas actualizaciones se realizan por lo general a nivel de infraestructura hardware y software (modernización de los equipos de cómputo, cambio a las nuevas versiones de programas de aplicación, lenguajes y plataformas de desarrollo). Sin embargo, en ocasiones, se descuidan los aspectos curriculares o se orientan de forma indebida. Por ejemplo, muchas instituciones de educación superior imparten los cursos de Fundamentos de Programación y Estructura de Datos con los elementos teóricos del paradigma de la programación estructurada, mientras que las prácticas y laboratorios de programación se desarrollan con lenguajes diseñados para la programación orientada a objetos y a eventos, con interfaces gráficas de usuario.

También se presenta el caso de orientar los cursos inferiores de fundamentos de programación (algoritmos) y estructuras de datos con técnicas estructuradas y los cursos intermedios de Ingeniería del software y bases de datos con técnicas orientadas a objetos generando una gran cantidad de dificultades al proceso enseñanza-aprendizaje, requiriendo así mayor esfuerzo de trabajo independiente del estudiante para apropiarse de

los fundamentos de la filosofía objetual y del docente para fomentar el cambio de paradigma.

El Grupo de Investigación en Ingeniería del Software del Tecnológico de Antioquia-GIISTA, dirigido por el Ingeniero Ricardo Botero, apoyado por los Ingenieros Eucario Parra, Carlos Castro y los Estudiantes de Tecnológico de Antioquia, Miguel Ángel Valencia, Juan David Maya, Andrés Felipe Atehortúa y Elizabeth Tobón, adoptó como proyecto de investigación la creación de una metodología para facilitar el proceso enseñanza-aprendizaje en los cursos de educación superior de lógica de programación (algoritmos), mediante la integración de los paradigmas estructurado (procedimental) y orientado a objetos.

La metodología consta de un lenguaje de modelado que posee características asociadas a lenguajes de programación de la familia C / C++ / Java y al Lenguaje de Modelado Unificado (UML). Posee elementos sintácticos nuevos que posibilitan la integración entre los paradigmas secuencial y orientado objetos y presenta además una didáctica propia con su propio microcurrículo.

La problemática ya ha sido tratada por diversos autores entre los que se tienen las investigadoras Graciela Elena Alvarado y Ana María Ferraro de Velo, de la Universidad Tecnológica Nacional de Argentina, Facultad Regional Buenos Aires, quienes tienen inscrito un proceso de investigación titulado **Mejoramiento de la enseñanza de la programación orientada a objetos** [2], en el cual se determinó el estado actual en la enseñanza de la POO en universidades y los requerimientos en el campo laboral, además de proponer contenidos y su secuencia para la enseñanza de la programación.

Los investigadores Ricardo A. Gómez Castro, Álvaro H. Galvis Panqueva y Olga Mariño Drews, trabajan el **proyecto Ingeniería de software educativo con modelaje orientado a objetos: un medio para desarrollar micromundos interactivos** [3], adscrito al proyecto LUDOMÁTICA, el cual es una alianza estratégica entre la Universidad de los Andes, la Fundación Rafael Pombo y el Instituto Colombiano de Bienestar

Familiar.

Heidi Wolf, en el artículo publicado en ComputerWorld, **Programación orientada a objetos (¿más sencillo o más complicado?)** [4], hace una serie de reflexiones en torno a la fundamentación que el estudiante debe recibir al incursionar en el mundo de la programación orientada a objetos.

El doctor Ricardo Devis Botella, presidente de la Asociación española para la Promoción de las Tecnologías Orientadas por Objetos, en su artículo titulado **Gestión de proyectos orientados a objetos: una incursión cruenta** [5], muestra el estado actual de la tecnología de objetos respecto de su aplicación en proyectos reales de desarrollo de software.

El ingeniero Juan Diego Zapata, investigador del proyecto CONEXIONES y profesor de lenguajes de programación de la Universidad EAFIT, Medellín-Colombia, en su artículo **"El Cuento" y su papel en la enseñanza de la orientación por objetos** [6], presenta una metodología de interés pedagógico que pretende servir de herramienta para facilitar la comprensión al abordar un problema usando la orientación a objetos.

Los profesores Daniel Gallo, Agustín Cernuda, Juan Manuel Cueva, Marina Díaz, Maria Pilar Almudena y José Manuel Redondo, del Departamento de Informática de la Universidad de Oviedo, han propuesto en su artículo **Reflexiones y experiencias sobre la enseñanza de POO como único paradigma** [7], argumentos a favor de una enseñanza temprana de la POO, así como los contenidos teóricos y prácticos impartidos en una asignatura de introducción a la programación.

El ingeniero Fabián Ríos Castrillón, profesor vinculado a la Facultad de Ingeniería de la Universidad de Antioquia, viene trabajando con el grupo SICOSIS una herramienta didáctica de iniciación a los objetos denominada **LÉXICO** [8]. La principal característica de este producto es que procesa seudo código orientado a objetos, permitiendo al estudiante interactuar con el paradigma antes de redactar código fuente en algún lenguaje de programación.

Otros investigadores que tocan directamente el tema son: Ricardo Botero [9], Carlos Castro et al [10], Inés Kereki [11].

Este artículo está organizado de la siguiente manera: se plantea la justificación, los objetivos generales y específicos, la metodología empleada para obtener los resultados de la investigación, algunos aspectos sintácticos y didácticos del lenguaje de modelado, la estructura microcurricular propuesta, conclusiones y referencias bibliográficas.

1. Justificación

La industria moderna del software tiene en la programación orientada a objetos toda una filosofía para conceptuar y representar la realidad tangible e intangible, debido a la versatibilidad que proporciona a los programadores y a la flexibilidad que demuestra para el desarrollo de sistemas de información. Si el paradigma de programación secuencial estudiado en los procesos iniciales de formación de ingenieros y tecnólogos informáticos, difiere del paradigma orientado a objetos utilizado en los procesos formativos intermedios y finales, se necesita entonces de un método integrador, para evitar los inconvenientes expuestos en el planteamiento del problema en la introducción.

Algunas universidades han generado mecanismos de solución desde la propedéutica del sistema curricular, que incluyen conceptos de la orientación a objetos en cursos de iniciación a las ciencias de la computación. Sin embargo, no existe un método formal de carácter general, aceptado y estandarizado, para desarrollar la capacidad de generar algoritmos codificables de forma expedita en un lenguaje orientado a objetos.

El desarrollo de este proyecto aportará elementos innovadores en cuanto a los métodos y estrategias adoptadas para la solución de problemas susceptibles de implementar en computadora mediante la concepción de algoritmos. La innovación se centra en el área de la informática relacionada con los fundamentos de la programación orientada a objetos y el diseño de algoritmos, con la intención de crear una estructura cognitiva que se

manifieste en la concepción de soluciones óptimas a problemas simples y complejos que involucren instancias de clase -objetos- en los algoritmos, sin rupturas conceptuales evidentes entre los paradigmas imperativo y objetual, ubicando en un mismo nivel de abstracción estos paradigmas. Esto puede lograrse si desde los procesos de formación iniciales y hasta el final del ciclo académico universitario o tecnológico, se crean estrategias para que el estudiantado conciba abstracciones basadas en objetos que integren algoritmos y datos, no pensando en ellos de manera independiente, como ocurre con la algoritmia imperativa.

Desde el punto de vista de los procesos mentales en el aprendizaje y construcción de soluciones, puede existir diferencia entre la concepción de la realidad integrada por objetos y la misma representada por estructuras secuenciales. De hecho, las cosas y actividades del quehacer cotidiano se interpretan naturalmente como conjuntos de objetos que interactúan, y no como ejecuciones realizadas en estricta secuencia sobre ellas.

La combinación de técnicas de programación orientada a objetos y conceptos de estructuras algorítmicas, permitirá plantear una metodología que garantice, desde los procesos formativos en cierne, un profesional competente, consciente del proceso evolutivo de las tecnologías aplicadas al software.

Con la adopción del Método Integrado de Programación Secuencial y programación Orientada a Objetos para el análisis, diseño y elaboración de algoritmos, se logrará en los desarrolladores de software:

La comprensión del paradigma de programación orientado a objetos desde el inicio de su formación en el área informática.

El análisis y diseño de algoritmos orientado a objetos (integra datos y algoritmos) y no de los datos y algoritmos concebidos de manera independiente.

2. Objetivos

General

Proponer un método que integre técnicas de la programación secuencial y de la programación orientada a objetos, para el análisis, diseño y elaboración de algoritmos.

Específicos

Presentar un compendio de problemas representativos donde se aplique el método propuesto, teniendo en cuenta grados de complejidad y diferentes formas de almacenamiento de datos.

Analizar los resultados obtenidos de la aplicación del método propuesto de acuerdo con los indicadores de evaluación.

Definir una simbología para aplicar el método en la solución de problemas.

Identificar los procesos cognoscitivos que se presentan en la resolución de problemas con las técnicas estructuradas y las orientadas a objetos.

3. Metodología

Las estrategias metodológicas para llevar a cabo la investigación, correspondientes con los objetivos específicos, implican los siguientes procedimientos presentados en su respectivo orden:

Análisis del estado del arte, relacionado con el tratamiento de la algoritmia en instituciones de educación superior.

Recolección y análisis de información asociada al tema de algoritmos, a partir de la revisión bibliográfica y de entrevistas a docentes universitarios y profesionales vinculados al sector productivo.

Estudio de la programación orientada a objetos y programación

estructurada, en búsqueda de:

Precisar cuáles elementos de estas teorías se incluirán en el método a desarrollar.

Comparar el desarrollo de algoritmos orientados a objetos y secuenciales sobre muestras de población diferentes.

Planteamiento de un método formal que permita elaborar algoritmos integrando la programación secuencial estructurada y la programación orientada a objetos, a partir de una gramática o simbología especial.

Tratamiento de un conjunto seleccionado de problemas típicos que cubran tópicos específicos de la programación, aplicando el método propuesto, con la intención de lograr generalidad.

Aplicación del método en pruebas piloto a grupos de estudiantes matriculados en los niveles iniciales e intermedios de Ingeniería de Sistemas en el Valle de Aburrá (Antioquia, Colombia).

Las pruebas piloto constituirán la fáctica del método propuesto, teniendo en cuenta la comparación entre los paradigmas imperativo y orientado a objetos.

Análisis de las diferencias en los procesos cognitivos al resolver problemas con el método secuencial y el método propuesto, que se hará sobre la información de entrevistas con expertos en educación, psicología y neurodesarrollo.

4. Algunos aspectos sintácticos y didácticos del lenguaje de modelado propuesto

Los elementos sintácticos por utilizar en la solución de problemas con el método propuesto (MIPSOO), se presentan en un orden que obedece al mapa conceptual de la figura 1.

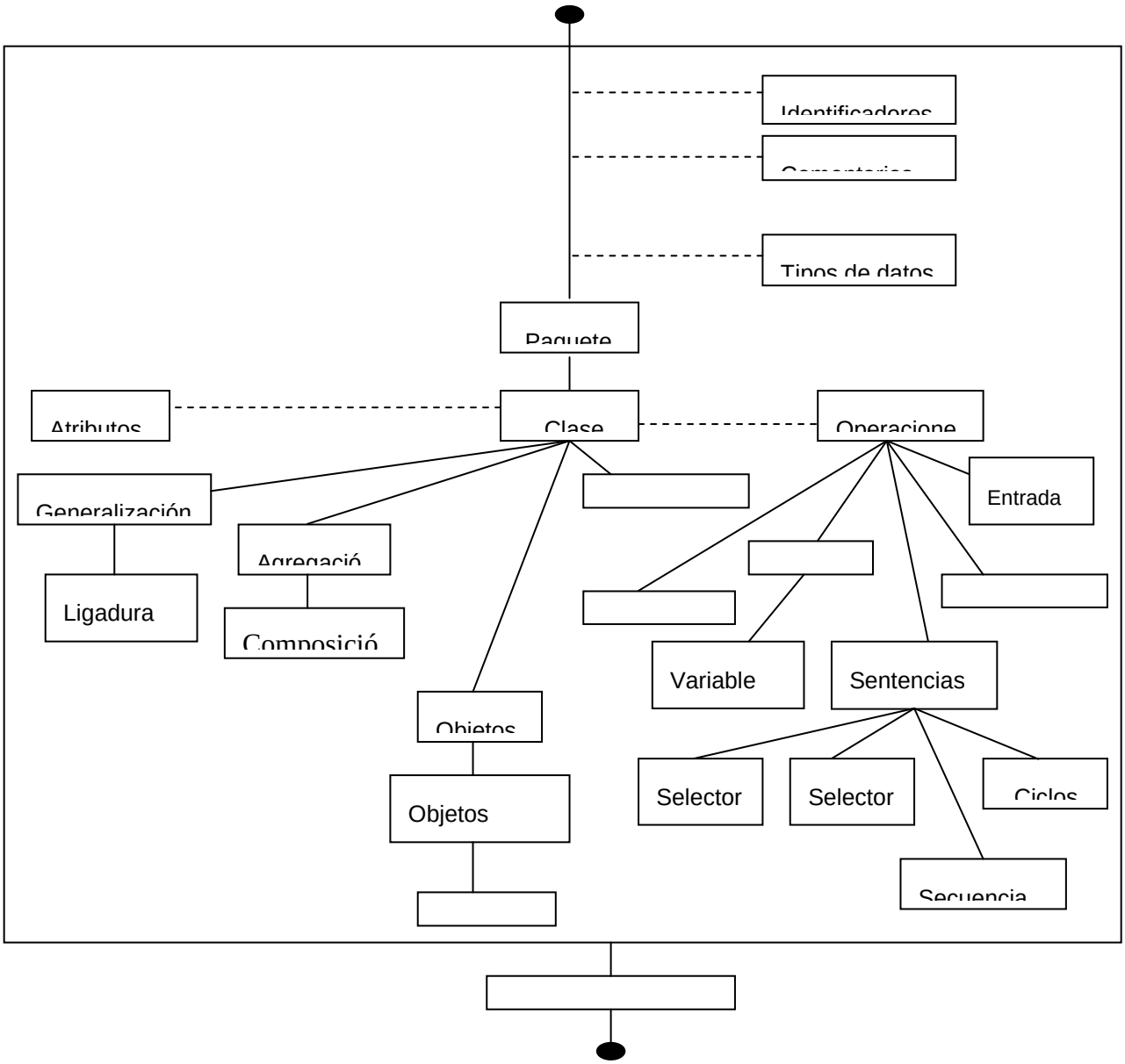

Figura 1. Elementos sintácticos utilizados en el MIPSOO

Algunos de los elementos mostrados en la figura 1, se indicaran más adelante en este artículo.

La estructura de una clase se presenta en la figura 2.

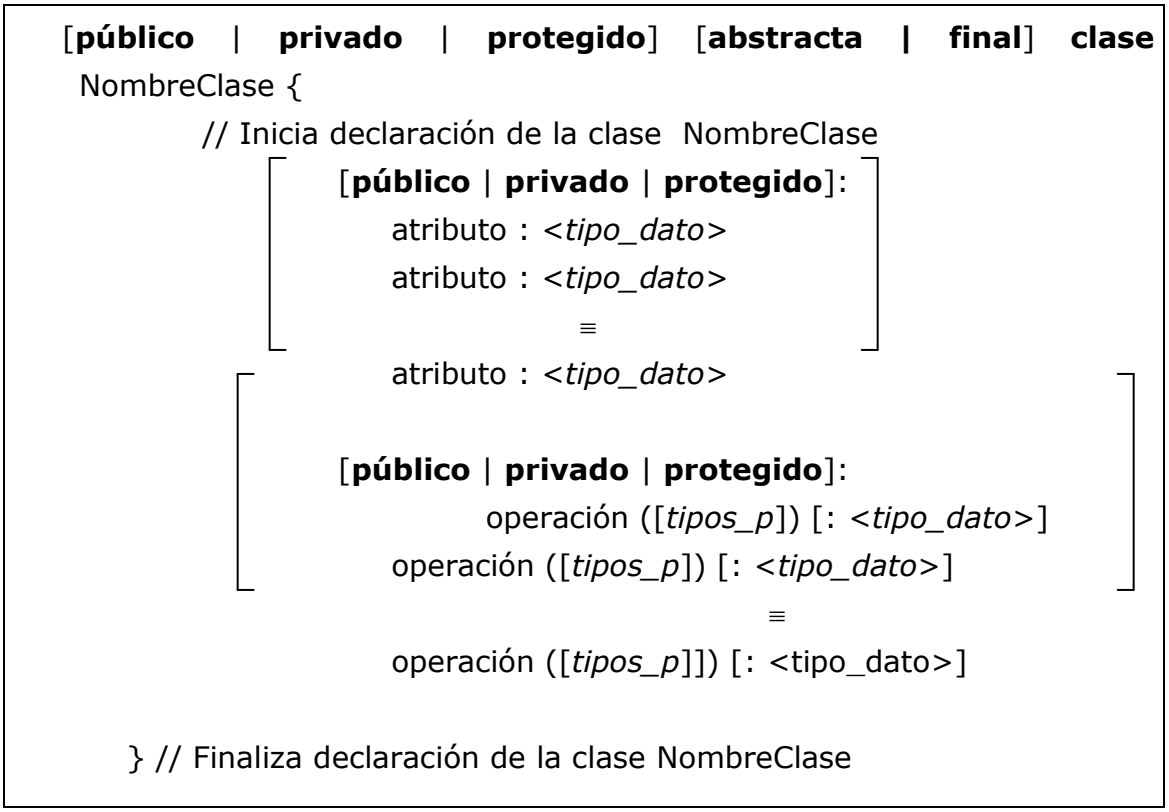

Figura 2. Definición de clase

En la figura 2 se indica la sintaxis para escribir una clase cuando se modela la solución de un problema mediante el paradigma orientado a objetos, independiente del lenguaje de programación en que va a ser implementado. Se propone al docente, como apoyo didáctico, la especificación del modelo

de datos mediante diagramas de clase UML.

De igual manera, se desarrolló una estructura para cada uno de los elementos planteados en la figura 1, elaborando así un lenguaje de modelado para la solución de algoritmos empleando el paradigma orientado a objetos e integrado con el paradigma estructurado.

Como parte de la didáctica que permite aplicar el lenguaje de modelado propuesto en la solución de algoritmos con el paradigma orientado a objetos, se propone en la figura 3 la estructura que debe llevar un algoritmo.

¡Error!

i) Enunciado del problema (a modo de comentario)

ii) Sentencias **importar**

iii) Declaración de clases

- Declaración de atributos y objetos agregados.
- Definición de métodos
- Seudo código de métodos en línea
- Seudo código de métodos fuera de línea

iv) **público clase** Proyecto {

público estático principal ( ){

// cuerpo del algoritmo principal

}

} // fin de Proyecto

Figura 3. Aspectos didácticos del método MIPSOO. Estructura general de un algoritmo Objetual

Para incluir el concepto de reutilización se sugiere al docente plantear un

esquema básico de clases que se pueden utilizar desde un comienzo como si ya existiese un FrameWork y permitir así utilizar clases que se consideran ya existentes y son de uso común. Sin embargo el docente y los estudiantes pueden elaborar algoritmos que permitan aumentar el Marco de Trabajo.

En la figura 4, se muestra un paquete con clases de uso común:

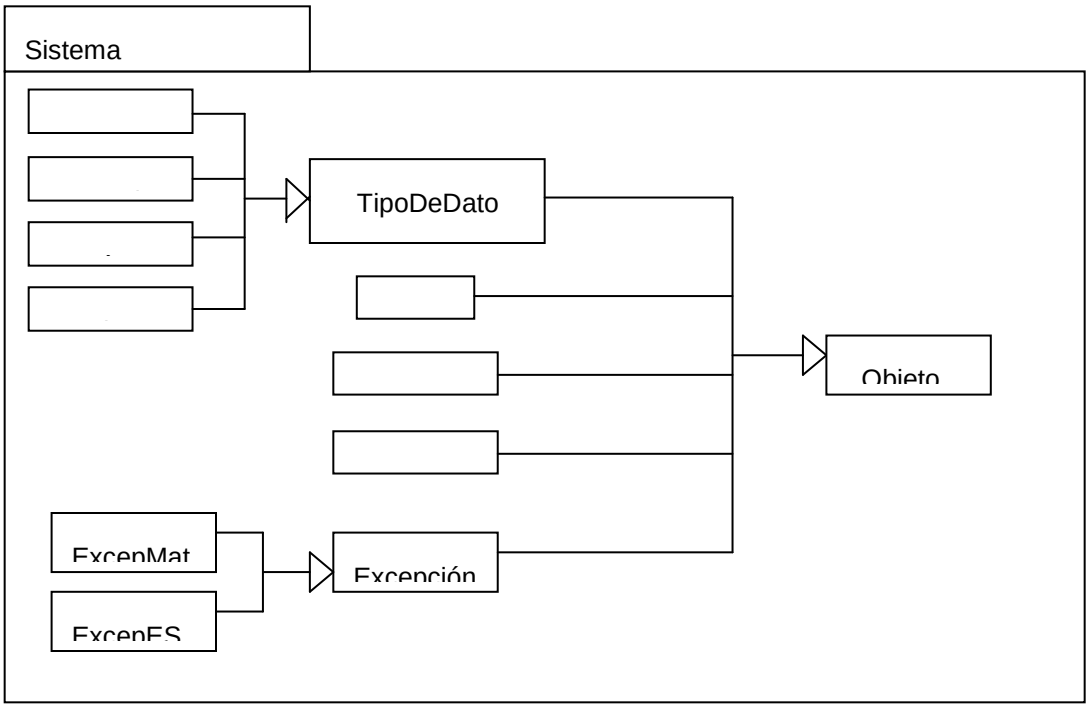

Figura 4. Aspectos didácticos del método MIPSOO. Clases de Uso Común.

5. Microcurrículo: propuesta para un curso de lógica de programación con orientación a objetos

5.1. Justificación

El aprendizaje en cursos tipo CS debe ligarse a las tendencias para desarrollo de software imperantes en el mercado y a los paradigmas de programación asociados a los lenguajes más utilizados en las aplicaciones

interconectadas de alto rendimiento.

Debido a que la orientación a objetos constituye el paradigma de programación subyacente en lenguajes como Java, Visual Basic.NET y C#, entre otros, herramientas *ad hoc* para desarrollo de software en la organización moderna, se justifica el tratado de los elementos conceptuales asociados al enfoque de objetos, que absorbe al paradigma imperativo y lo complementa con otros conceptos propios de la realidad circundante, como objeto, herencia y polimorfismo.

La orientación a objetos surge en Noruega en 1967 con el lenguaje Simula 67, que introdujo por primera vez los conceptos de clase, subrutina y subclase (conceptos muy similares a los tratados en los lenguajes orientados a objetos modernos).

Small Talk fue el primer lenguaje puro orientados a objetos, creado en la década de los años 70 por los científicos de Palo Alto Xerox. Small Talk utiliza sólo clases y objetos. Java, lenguaje de actualidad considerado puro orientado a objetos, usa tipos de datos primitivos y Wrappers, que son clases utilizadas para encapsular los tipos de datos primitivos.

En los años 80 (siglo pasado) Bjarne Stroustrup de AT&T Labs amplió el lenguaje C para crear C++, que soporta la programación orientada a objetos. En esta misma década se desarrollaron otros lenguajes orientados a objetos como Objective C, Common Lisp Object System (CIOS), Object Pascal y Ada, entre otros.

A principios los años 90 se consolida la orientación a objetos como uno de los mejores paradigmas de programación para resolver problemas en una extensa gama de dominios; además, el paradigma fortalece la generación rápida de prototipos (RAD: Rapid Aplication Developments), que permite el desarrollo de aplicaciones aun cuando los requerimientos iniciales no estén totalmente especificados.

En 1995, James Golsling y su grupo desarrollador de Sun Microsystems

lanzan Java, lenguaje concebido inicialmente para la programación de equipos electrónicos, y luego utilizado en ambientes corporativos y entornos Web por su facilidad en la reutilización de software existente.

En 1997-98 se desarrollan herramientas CASE (Computer Aided System Engineering) orientadas a objetos.

Desde el año 1998 a la fecha, se desarrollan las arquitecturas de objetos distribuidos RMI, Corba, COM y DCOM.

Con la orientación a objetos se mejora el diseño, desarrollo y mantenimiento del software, ofreciendo una solución a largo plazo a los problemas y preocupaciones que han existido desde comienzos del historial del software: falta de portabilidad del código, reusabilidad y modificación (antes difíciles de lograr), ciclos largos de desarrollo y técnicas de programación no intuitivas.

Actualmente la orientación a objetos parece ser el mejor paradigma. Sin embargo, no es la panacea. Trata de eliminar la crisis del software.

Entre los creadores de metodologías orientadas a objetos se encuentran Grady Booch, James Rumbaught, Ivar Jacobson, James Martin y Peter Cheng.

5.2. Objetivo general

Compenetrar al estudiante en la solución de problemas mediante su modelación dirigida a los objetos (ocultamiento de la información, sobrecarga de métodos, herencia, polimorfismo, etc.), y el desarrollo de métodos con algoritmos estructurados, independiente de cualquier lenguaje de programación.

5.3. Objetivos específicos

- Comprender el concepto de paradigma de programación y las principales diferencias conceptuales entre los paradigmas imperativo,

funcional, heurístico y orientado a objetos.

- Formular y aplicar una metodología para la solución de problemas en el computador.
- Definir, caracterizar y aplicar elementos subyacentes en el paradigma orientado a objetos, a fin de presentar la solución a problemas en términos de dicho paradigma.
- Desarrollar algoritmos mediante el uso de las estructuras secuencia, selección y ciclo, para incorporarlos como métodos a las soluciones de problemas basados en objetos.
- Identificar los objetos y a partir de ellos las clases requeridas para el consecuente diagrama en UML.
- Desarrollar la habilidad de pensamiento recursivo.
- Generar Tipos de Datos Abstractos y sus funciones asociadas para el análisis y solución de un problema.

5.4. Competencias

Distinguir los elementos conceptuales básicos de la orientación a objetos y asociarlos de forma significativa.

Definir de forma acertada, para todo problema planteado, la entrada y salida requerida, las clases que intervienen y el proceso general que conduzca a la solución más óptima.

Realizar abstracciones (objetos, clases, asociaciones) ajustadas a un problema específico, es decir, enfocadas tanto al ámbito del problema como al ámbito de la solución.

Desarrollar los algoritmos (operaciones o métodos) aplicando la teoría de objetos.

5.5. Metodología

Clases magistrales donde el profesor aplicará el modelo pedagógico constructivista, teniendo en cuenta que se debe inmiscuir al estudiante en procesos de aprendizaje por descubrimiento, activo, basado en problemas (situado) y significativo.

5.6. Contenido

- Paradigmas de programación
- Paradigmas de programación y sus tipos
- Paradigma imperativo
- Paradigma funcional
- Paradigma heurístico
- Paradigma orientado a objetos
- Introducción a la programación orientada a objetos
- El "cuento" como ambientación al trabajo con objetos

    Paso 1: Redacción del cuento (o enunciado de la situación problemática)

    Paso 2: Tabla de personajes

    Paso 3: Tabla de actividades

    Paso 4: Modelo estático

    Paso 5: Modelo dinámico

- Clases y objetos

    Introducción

- Características de los objetos
- Identidad del Objeto
- Clasificación
- Encapsulación y ocultación de datos
- Mantenibilidad
- Reusabilidad
- Polimorfismo
- Herencia
- Definición de una clase
    - Objetos como instancias de clase
    - El paquete como agrupación de clases
- Estado de los objetos
    - Tipos de datos primitivos
    - Atributos
    - Operadores y expresiones
- Comportamiento de los objetos
    - Variables vs. Atributos
    - Constantes
    - Sentencias de control
- Secuencia

- Selector
- Ciclo
- Métodos
- Estructura general de un método
- Paso de parámetros
- Constructores y destructores
- Sobrecarga de métodos
- Excepciones
- Relaciones entre objetos

    Generalización
- Clase derivada
- Clase abstracta
- Ligadura dinámica (polimorfismo)

    Herencia múltiple e interfaces

    Composición
- Estructuras de datos fundamentales

    La clase Vector

    La clase Matriz

6. Conclusiones y recomendaciones

El trabajo investigativo conduce a las siguientes conclusiones:

- Los cursos de iniciación a las ciencias de la computación se deben impartir según los lineamientos establecidos por las tendencias de los nuevos lenguajes de programación imperantes en el mercado del software, es decir de acuerdo con el paradigma que lleve la teoría subyacente para el desarrollo de aplicaciones de alto desempeño.

- En este sentido, el paradigma orientado a objetos es el más conveniente en la actualidad.

- Los componentes pedagógicos, didácticos, cognoscitivos, son fundamentales al momento de iniciar estudios de iniciación a la programación. El constructivismo y la didáctica con elementos del aprendizaje por descubrimiento, significativo y basado en problemas, pueden garantizar un proceso cognitivo exitoso.

- Es recomendable el desarrollo de una herramienta software fundamentada en el MIPSOO, que permita la solución de problemas de programación con orientación a objetos.

- Finalizado este trabajo de construcción teórica, se proyecta socializar el método con docentes vinculados a instituciones de educación superior nacionales y extranjeras, compartir la experiencia con grupos de investigación interesados en la temática y otras asociadas a la ingeniería de software.

7. Bibliografía

1. *Computing Curricula 2005. The Overview Report .including The Guide to Undergraduate Degree Programs in Computing for undergraduate degree programs in Computer Engineering, Computer Science, Information Systems, Information Technology, Software Engineering. A volume of the Computing Curricula Series. The Joint Task Force for Computing Curricula* 2005.

A cooperative project of The Association for Computing (ACM), The Association for Information Systems (AIS), The Computer Society (IEEE-CS). http://www.acm.org/education/curricula.html Fecha de acceso: Junio de 2005

2. ALVARADO, Graciela Elena y Ana María Ferraro de Velo. Mejoramiento de la enseñanza de la programación orientada a objetos, http://www.utn.edu.ar/scyt/proyectos/index.htm Fecha de acceso: Junio de 2003.

3. GÓMEZ CASTRO Ricardo A., Álvaro H. Galvis Panqueva y Olga Mariño Drews, Ingeniería de software educativo con modelaje orientado a objetos: un medio para desarrollar micromundos interactivos, http://www.minerva.uevora.pt/simposio/comunicacoes/rigomezmarino.html Fecha de acceso: Junio de 2003.

4. WOLF, Heidi. Programación orientada a objetos (¿más sencillo o más

complicado?), ComputerWorld España, Nº620, Mayo de 1995.

5. DEVIS BOTELLA, Ricardo. Gestión de proyectos orientados a objetos: una incursión cruenta, http://www.well.com/user/ritchie/oopm-uic.html. Fecha de acceso: Julio de 2004.

6. ZAPATA, Juan Diego. "El Cuento" y su papel en la enseñanza de la orientación por objetos, http://conexred.eafit.edu.co/~jzapata. Fecha de acceso: Julio de 2003.

7. GAYO AVELLO, Daniel et al. Reflexiones y experiencias sobre la enseñanza de la POO como único paradigma. Actas de las IX Jornadas de Enseñanza Universitaria de Informática. Cádiz. Julio de 2003. P. 405 - 412.

8. RIOS CASTRILLÓN, Fabián. Apuntes de diagramación. Facultad de Ingeniería, Universidad de Antioquia, 1982.

9. BOTERO TABARES, Ricardo. Metodología para la algoritmia orientada a objetos. Facultad de Educación, Universidad de Antioquia, 2003.

10. CASTRO CASTRO, Carlos Arturo. Javier Cardona Palacio. Edgar Eusebio López. Cesar Jaramillo. Hacia una Nueva Forma de Enseñar la Lógica de Programación. Acierto Revista Académica y Científica de la Corporación Universitaria Rémington. UniRemington ISSN 1657-5954 Octubre de 2001. Medellín.

11. KEREKI, INÉS. Enseñando y aprendiendo programación orientada a objetos en los primeros cursos de programación, http://www.cusoft.org.uy/docs97/ensenanzainicial.doc. . Fecha de acceso: Julio de 2004.