Modelo metodológico para simplificar la comprensión de tres paradigmas de programación basándose en Aprendizaje Significativo*[1]

[Methodological proposal to simplify the concept under three computer programming paradigm using meaningful learning]

Omar Iván TREJOS BURITICÁ[2]

Recibo: 25.03.2015 – Aprobación: 26.09.2015

Resumen

Este artículo aborda una propuesta metodológica para que la comprensión general de los paradigmas de programación funcional, imperativa y orientada a objetos se simplifique de forma que los estudiantes de informática tengan un fundamento sólido a partir del cual puedan comprender sus diferencias y aplicaciones. La propuesta se basa en la cristalización de tres modelos teóricos asociados al modelo computacional que privilegia dentro del contexto de sus propias características, y se enmarca dentro de una investigación educativa de carácter cualitativo, articulada con la teoría que subyace a cada paradigma, la cual se ha socializado con estudiantes de Ingeniería de Sistemas, y se han recogido algunas opiniones al respecto de su facilidad para comprender los tres paradigmas. A partir de la utilización de este modelo se han evidenciado, vía resolución de problemas teóricos, una aplicación directa y sencilla del concepto de paradigma así como una mejor

* Modelo para la citación de este artículo:
TREJOS BURITICÁ, Omar Iván (2015). Modelo metodológico para simplificar la comprensión de tres paradigmas de programación basándose en Aprendizaje Significativo. En: Ventana Informática No. 33 (jul-dic). Manizales (Colombia): Facultad de Ciencias e Ingeniería, Universidad de Manizales. p. 125-142. ISSN: 0123-9678

1 Artículo de investigación científica y tecnológica proveniente de la Tesis Doctoral *Aprendizaje en Ingeniería: Un problema de comunicación*, culminada en *octubre de 2012* e inscrita en la Vicerrectoría de Investigaciones de la Universidad Tecnológica de Pereira.

2 Ingeniero de Sistemas, PhD. en Ciencias de la Educación. Docente de Planta, Facultad de Ingenierías, Ingeniería de Sistemas y Computación, Universidad Tecnológica de Pereira, (Pereira, Risaralda, Colombia), Correo electrónico: omartrejos@utp.edu.co

utilización de las herramientas que giran en torno a ellos. La comprensión de las diferencias, coincidencias y aplicaciones de los paradigmas ha generado elementos de alta motivación y aprendizaje autónomo en los estudiantes.

Palabras clave: *Aprendizaje significativo, paradigma de programación, programación de computadores, programación imperativa, programación funcional, programación orientada a objetos.*

Abstract

This article presents a methodological proposal to understand the basics of three computer programming paradigms (functional, imperative and object-oriented) to bring to informatics students a solid foundation to use their differences and their applications. We use three theorical models linked with de computational model associated to each paradigm and its characteristics. The methodology is allocated inside an educative and qualitative research accordingly the theory of each computer programming paradigm. The results of this methodology have been socialized with Systems Engineering students in the firsts and last semesters and we collected some opinions about it. From the use of this model and the use of problem based learning we evidence that the paradigm concept and its application is more direct and simple and, due this, the use of the tools around the concepts. Another effect of this methodology is a higher motivation and an autonomous learning in the students.

Keywords*: Computer programming, imperative programming, functional programming, meaningful learning, object-oriented programming, programming paradigm.*

Introducción

Los paradigmas de programación son la base para el desarrollo de soluciones que involucren tecnología computacional en programas como Ingeniería de Sistemas. Consideran Van Roy & Haridi (2004, 56) que el aprendizaje de los paradigmas de programación, así como del modelo computacional que subyace a cada uno, es un camino en el campo docente para que el estudiante pueda utilizar con mayor autonomía y destreza los lenguajes de programación. Por su parte, un paradigma implica un modelo matemático, un modelo conceptual y un modelo computacional que posibilita soluciones a un conjunto de situaciones

específicas que pueden ser resueltas con la tecnología computacional por la vía de la programación, señala Trejos (2012a, 39).

Estudiar los paradigmas, conocer el modelo matemático que subyace a cada uno así como el modelo conceptual y computacional son opciones que tiene un programador para aprovechar las posibilidades de un lenguaje en la solución de un problema. Se ha encontrado en la experiencia del autor de este artículo, que para los estudiantes de programación de computadores las diferencias y potencialidades de los paradigmas y la utilidad de su conocimiento es muy poco destacable y que, por el contrario, la atención se centra en la utilización y conocimiento profundo de los lenguajes de programación, de sus minucias sintácticas y técnicas, de sus aplicaciones y de las posibilidades que brindan al momento de programar en tiempos modernos, en coincidencia con Schildt (2010, 112).

La tecnología y sus expresiones modernas, lenguajes de programación, serán siempre formas de *vida breve*, es decir, formas de aprovechamiento de la tecnología computacional que siempre son reemplazados en breve término por nuevas expresiones de mayor alcance y que ofrecen mejores posibilidades, afirma Trejos (2012, 89). La fundamentación teórica que subyace a un lenguaje de programación es, precisamente, su paradigma asociado, lo cual constituye el sustento para que el lenguaje de programación sea la cristalización de lo que, desde una perspectiva teórica, las matemáticas pueden posibilitar.

La ausencia de un modelo conceptual que permita establecer relaciones entre los paradigmas de programación, o su desconocimiento, ha llevado a que la enseñanza de la programación se convierta en una repetición de instrucciones de lenguajes por parte de los docentes y en el aprendizaje de una serie de usos y trucos que en últimas, aun haciendo gran gala de su utilización, no deja de adolecer de la fundamentación propia para que se aprovechen al máximo sus posibilidades.

Este artículo precisamente concibe la metodología propuesta como el más firme panorama que facilita una concepción de los paradigmas de programación y sus modelos asociados para, una vez asimilados en sus fundamentos, se puedan estudiar profundamente los lenguajes de programación. La figura 1 presenta el orden propuesto para el estudio de los paradigmas de programación y la lógica que subyace a su aplicación.

Aunque se admite que en la actualidad es posible desarrollar competencias de alto nivel en relación con la programación de

computadores acudiendo solo a la expresión tecnológica que representan los lenguajes de programación, sin tener que profundizar en los modelos que constituyen el paradigma que subyace a determinado lenguaje, como lo asegura Trejos (2012b, 77), la experiencia ha demostrado que el estudio de los paradigmas hace más fácil la comprensión de los lenguajes de programación y simplifica su utilización y aprovechamiento.

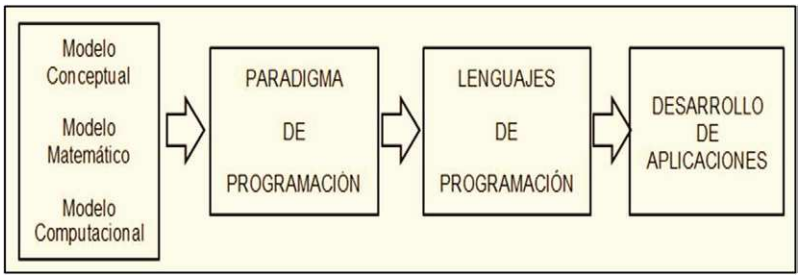

Figura 1. Relación de precedencia paradigmas – lenguajes de programación

No se descarta para investigaciones posteriores, la revisión detallada de la prioridad que se le concede a los paradigmas por encima de los lenguajes de programación. Desde esta perspectiva, la pregunta de investigación podría ser ¿cuáles son los conceptos y modelos que priman en el aprendizaje de un lenguaje de programación de manera que se puedan aprovechar y capitalizar al máximo los recursos que, tecnológicamente, estos ofrecen? La respuesta no se deja esperar y de allí se puede inferir la hipótesis asociada a la investigación que inspira este artículo: los paradigmas de programación contienen los modelos (conceptuales, matemáticos y computacionales) cuya comprensión facilita el aprovechamiento y capitalización de los recursos que posibilita un lenguaje de programación.

El marco de referencia de esta investigación es el aprendizaje tecnológico, se fundamenta en la teoría del aprendizaje significativo formulada por David Paul Ausubel y la teoría pedagógica socio-constructivista de Lev Vigotsky y su aplicación en el campo de la ingeniería de sistemas en el área de la programación de computadores. La posición epistemológica de estos enfoques alude a una concepción de naturaleza consciente y holística de manera que se vincule lo social con lo educativo y se entienda que el aprendizaje de lo fundamental propicia y facilita el aprendizaje de lo aplicativo de la mano de la motivación y la autonomía.

1. Fundamento teórico

La teoría de aprendizaje significativo fue formulada por David Paul Ausubel (1986, 72), cuyo planteamiento central establece que el significado constituye la esencia de un proceso de aprendizaje y la base para que el conocimiento se dinamice, cambien las estructuras cognitivas y modifique por otras que, a la luz del aprendiz, puedan ser más coherentes con sus propias experiencias. Como *significado* se entiende al sentido de utilidad y el para qué de un cúmulo de conocimientos con lo cual se está privilegiando la aplicación por encima de la teorización; que también puede tener una connotación teórica y la aplicación verse desde la perspectiva de la utilidad que la sociedad le conceda dentro del marco de sus necesidades, en relación con un área de conocimiento o con un conocimiento disciplinar específico, según Ausubel (1986, 102).

La teoría del aprendizaje significativo establece que un proceso de aprendizaje comprende tres elementos que constituyen las bases de todo su *corpus teórico*. Para Bruner (1963, 45), estos elementos son: el conocimiento previo, el nuevo conocimiento y la actitud del estudiante. El conocimiento previo está conformado por todo el conjunto de experiencias, vivencias, conceptos e interacciones que, de una u otra forma, han ido estructurando la visión que un aprendiz tiene del mundo. Esto implica que, indica Bruner (1991, 31), toda persona cuando comienza un proceso de aprendizaje ya tiene una serie de conceptos y formas de interpretar el mundo que le rodea que son producto de sus propias vivencias e interacciones tanto con el medio como con la sociedad.

Ausubel (1986, 7) establece que «*si me preguntaran qué es lo más importante en un proceso de aprendizaje yo diría que es lo que el estudiante ya sabe*» con lo cual privilegia la experiencia de aprendizaje, entendida como la vida misma, que tiene el estudiante y su corpus académico cuando se vincula a un proceso de aprendizaje. El conocimiento previo también se forma por la interacción con el medio, con la formación en familia, con el contacto con la sociedad y con sus propias experiencias que, a la postre, forman un cúmulo de conceptos y estructuras que son las que sirven de base y fundamento para el desarrollo de cualquier proceso de aprendizaje posterior, según Piaget (2001, 65).

Como nuevo conocimiento puede definirse todo el conjunto de experiencias, vivencias, conceptos, teorías y modelos que llegan por primera vez al estudiante o que, habiendo llegado alguna vez, ya se habían olvidado por completo posiblemente por falta de uso o, también,

por falta de significado. El nuevo conocimiento, como parte de un proceso de aprendizaje, es el encargado de impactar el conocimiento previo cuestionándolo, reforzándolo o simplemente modificándolo. La relación entre el conocimiento previo y el nuevo conocimiento es, de acuerdo con Bruner (1991, 60), lo que hace que el estudiante vaya modificando o reforzando sus estructuras cognitivas dentro del marco de una formación específica profesional.

El tercer componente de la teoría del aprendizaje significativo es la actitud del estudiante que está compuesta por dos partes: la motivación y la capacidad de relacionar el conocimiento previo con el nuevo conocimiento. La participación de un estudiante en un proceso de aprendizaje de la mano de la motivación permite que se incorporen conceptos como la autonomía y el autoaprendizaje pues más que lograr que el estudiante aprenda un *corpus* de conocimiento, afirma Hermann (2000, 29), el docente tiene la misión de lograr que el estudiante aprenda a aprender. En la motivación se incluye la voluntad libre y espontánea de que el estudiante quiera estar presente (física o virtualmente) en cada una de las actividades de un proceso de aprendizaje y, por lo tanto, en la puesta a disposición de unos ánimos que encuentren en el *corpus* académico de dicho proceso razones suficientes para continuarlo.

La motivación depende en alto grado tanto del docente como de las estrategias que adopte y manera como éste las combine con la temática propia que se aborde en el proceso de aprendizaje. Difícilmente, señala Ausubel (1989, 81), un proceso de aprendizaje logrará sus objetivos si de por medio no existe un factor motivacional suficientemente sólido que posibilite en los estudiantes el ánimo a continuar con dicho proceso. La figura 2 muestra la relación entre los componentes de la teoría del aprendizaje significativo.

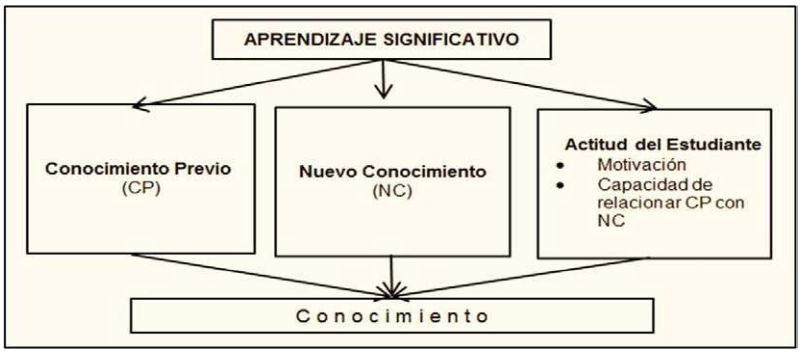

Figura 2. Relación componentes Aprendizaje Significativo (Trejos (2012a, 134)

En lo puramente técnico, este artículo intenta establecer una propuesta metodológica que simplifique la comprensión conceptual de tres paradigmas de programación: el paradigma imperativo, el paradigma funcional y el paradigma orientado a objetos. El primer paradigma que se aborda es el imperativo que se mueve dentro del marco de la programación estructurada y que, opuesto al paradigma funcional, establece los términos de la programación a partir del concepto de estado del programa y basándose en sentencias que son las encargadas de cambiar dicho estado según Deitel & Deitel (2010, 66). El núcleo de trabajo de la programación imperativa es el concepto de variable que no es más que un espacio de memoria en el cual se pueden almacenar datos. El conjunto de datos que contienen las variables en un momento dado, afirma Schildt (2010, 189), es lo que se conoce como estado del programa y las instrucciones de los programas imperativos se ocupan de cambiar dicho estado hasta llevarlo a un nivel tal que se satisfaga el objetivo propuesto como solución a un problema computacional. En un programa imperativo se consignan instrucciones que le indican al computador cómo debe realizarse la tarea o ejecutarse el algoritmo.

La esencia de los programas imperativos recae en las variables y las sentencias que modifican los contenidos de estas variables en donde cada paso corresponde a una instrucción y solo así, una a una, se ejecutan teniendo en cuenta una regla trivial: cada instrucción se realiza después de la anterior y antes de la siguiente según Trejos (2000, 33). Entre las dificultades que se presentan con este paradigma está la necesidad de mejorar sus mecanismos de depuración y la implementación de procesos paralelos de forma que el trabajo se pueda dividir en hilos que se ejecuten simultáneamente.

El paradigma funcional, que se enmarca en la programación declarativa, corresponde al fundamento de los lenguajes llamados funcionales que se pueden definir como esos lenguajes en los cuales las variables no tienen estado, principal característica del paradigma imperativo, lo cual significa que no experimentan cambios en su contenido a lo largo del tiempo lo cual las lleva a ser inmutables pues no se pueden cambiar los valores a lo largo de una misma ejecución. Los programas se escriban a partir de la definición de funciones que se componen, a su vez, de expresiones. El núcleo de trabajo principal de la programación funcional es el concepto de función, sus características, sus elementos constitutivos y sus argumentos.

Trejos (2000, 121) considera que la función es un pequeño programa que cumple un pequeño objetivo. Es la unidad de trabajo de la lógica funcional y de su paradigma asociado y permite que un problema cuya solución exija un objetivo complejo pueda ser dividido en pequeñas partes, cada una problémica, de manera que resolverlas de manera independiente y enlazarlas origina la solución al gran problema original. Esta técnica se conoce como *divide and conquer* (divide y vencerás) y permite visualizar la solución de un problema más en sus elementos constitutivos que en el todo del problema.

La utilización de funciones posibilita hallar con mucha facilidad errores tanto lógicos como sintácticos. Muñoz (2014, 4) señala que como lenguajes funcionales se cuentan *Scheme*, *Haskell*, *OCaml* y *Lisp*, entre otros. En un lenguaje funcional, por sus características, los procesos cíclicos se resuelven con el concepto de recursividad según Van Roy (2008, 53) que, en lo puramente técnico, es la capacidad que tiene una función de llamarse a sí misma y que en lo matemático es la posibilidad de llamar a una función y que parte de su desempeño sea su misma definición. Los fundamentos matemáticos de los lenguajes funcionales corresponden al sistema formal que diseñó Alonzo Church hacia la primera mitad del siglo XX con lo cual definió cómputos y estudio las aplicaciones a partir de las funciones de Cálculo Lambda.

El tercer paradigma es el de programación orientada a objetos según el cual, señalan Jeandrock et al. (2010, 29), la esencia de su lógica es el concepto de objeto que es la conformación de unidades lógicas que se componen de métodos y atributos. Los métodos son las funciones que le dan uso al objeto como tal y los atributos son las características que lo distinguen de otro objeto[3].

El conjunto de atributos y métodos es lo que conforma el objeto en sí y lo diferencia de otros objetos, lo que implica que dos objetos son iguales cuando dicho conjunto es exactamente igual en ambos. El conjunto completo de atributos y de métodos podría ser infinito si se analiza que las características y los usos de un objeto pueden ser tantos como la imaginación, la cultura y la creatividad así lo permitan. La definición de los atributos y los métodos de un objeto es lo que se conoce como una clase y un ejemplo de la clase se conoce como una instancia.

3 A manera de ejemplo, se plantean dos objetos, con sus atributos y métodos:
Objeto 1: Avión. Atributos: Tamaño, capacidad, horas de vuelo, combustible, peso, ventanas, tripulación. Métodos: Volar, aterrizar, comunicarse, cambiar de ruta, accidentarse, solicitar pista.
Objeto 2: Lápiz. Atributos: Tamaño, peso, textura, color, sabor, olor, material. Métodos: Escribir, dibujar, señalar, apuntar, disparar, rascar, trancar, medir.

Para Schildt (2009, 55), en la programación orientada a objetos los datos (atributos) y las funciones (métodos) siempre están en relación y se busca ir más allá de un simple procesamiento que transforma una entrada en una salida. En la programación orientada a objetos se busca solucionar problemas computacionales intentando interpretar el mundo, es decir, en donde coexisten objetos de diferentes tipos con características y usos diferentes. La POO (programación orientada a objetos) tiene conceptos propios como clase, herencia, polimorfismo, método, atributo, etc.

La conformación y estructuración de un objeto es, sin lugar a dudas, un acto de abstracción de la mente humana al crear una imitación del mundo real a partir de sus características y sus usos que, técnicamente, se han llamado atributos y métodos respectivamente y que posibilitan conceptos como encapsulamiento, modularidad, ocultación, herencia, polimorfismo y otras. Diferentes lenguajes orientados a objetos han cobrado gran importancia en el mundo moderno, entre ellos se cuentan *Java*, *Object C*, *C#*, *Delphi*, *Visual Basic*, *COOL* y *Fortran 95*. La avanzada y alta penetración de la programación orientada a objetos y su paradigma asociado se debe posiblemente a que este paradigma interpreta el mundo de una forma más exacta a como se ve sin mayores esfuerzos para ajustar la visión del mundo a las restricciones que un determinado modelo o paradigma permite como sucede en los paradigmas funcional e imperativo.

Cada uno de estos tres paradigmas (funcional, imperativo y orientado a objetos) constituye todo un conjunto de conceptos, teorías y ejemplos que se salen del espíritu de este artículo. Sin embargo se han querido plantear los conceptos fundamentales que permiten realizar una primera aproximación a dichos paradigmas con el ánimo de tener criterios que posibiliten el entendimiento y, posteriormente, el aprendizaje y aplicación de los lenguajes a los cuales les subyace cada uno de estos paradigmas. Debe advertirse que se encuentra con gran frecuencia los llamados lenguajes híbridos que no son ejemplo exclusivo de un solo paradigma sino que aceptan la utilización de los recursos de dos, e incluso los tres paradigmas, parabién de las necesidades de los programadores.

2. Metodología

¿Cómo entender la concepción primaria y fundamental de cada paradigma? ¿Cómo aprovechar el concepto de cada uno para,

posteriormente, capitalizar los recursos y facilidades que un lenguaje de programación puede brindar? Esas son las razones que inspiran este artículo y por lo tanto la metodología que se ha propuesto es la de simplificar los conceptos a un punto que se pueda ver, con mayor claridad, cuál es el perfil de cada paradigma así como sus coincidencias y divergencias con los demás paradigmas.

La tabla 1 presenta una caracterización esquemática simplificada de cada paradigma. El paradigma funcional privilegia el concepto de función, sus características y sus relaciones en la construcción de soluciones a problemas computacionales. La esencia del paradigma funcional está en la función cuyo cuerpo normalmente está constituido por expresiones que permiten lograr el objetivo establecido. Dentro de las funciones, siempre existirá una función principal aunque el paradigma funcional no caracteriza la función principal de ninguna manera, simplemente la lógica propia del programa permite analizar cuál es la primera función que debe llamarse para que todas las demás cobren vida. La recursión constituye uno de las herramientas conceptuales más fuertes y efectivas que tiene la programación funcional y en cualquier momento se le puede hacer tanto seguimiento a todo un programa como a una sola función para validar sus resultados.

En el caso del paradigma imperativo se privilegia el concepto de estructura según Trejos (2012a, 36), sus características y sus relaciones en la construcción de soluciones a problemas computacionales. La esencia del paradigma imperativo radica en el concepto de variable, su contenido (estado) y en las estructuras básicas (secuencias, condicionales y ciclos). Aunque se pueden construir programas imperativos basados en funciones, no son éstas la esencia de este paradigma pues podría concebirse una solución con una sola función que contenga todo el algoritmo. Cuando se trabaja con funciones, en el paradigma imperativo, siempre existe una función que es la principal y que, normalmente, está caracterizada en los lenguajes de programación que se enmarcan dentro de este paradigma. El objetivo de la función principal es el de coordinar el llamado a las demás funciones. Aunque se maneja la recursión no es uno de los grandes pilares de este tipo de programación. Se pueden ubicar puntos de seguimiento o ruptura en cualquier parte del programa para verificar los estados temporales de las variables.

Tabla 1. Caracterización esquemática de cada paradigma

Tópico	Paradigmas		
	Funcional	**Imperativo**	**Orientado a Objetos**
Nociones Básicas	Función constructora definida Aplicación de una función Independencia de funciones	Variable con estado Las instrucciones cambian el contenido de las variables Existencia y dependencia de procedimientos	Estados asociados a las instancias de las clases Concepto de clase e instancia de clase Dependencia e interdependencia de clases Herencia, polimorfismo, encapsulamiento
Procedimientos	Se llaman funciones Cada llamado a una función representa el valor que produce	Los procedimientos tienen nombre y parámetros Los procedimientos retornan valores	Se trabajan con objetos que son instancias de clase Un objeto es la unión entre un conjunto de atributos (variables) y un conjunto de métodos (funciones)
Estructuras de datos	Suma directa y productos cartesianos	Arrays Registros Estructuras dinámicas	Productos cartesianos Arrays Registros Estructuras dinámicas
Programa	Declaración de funciones + expresión principal	Declaraciones de procedimientos + rutina principal	Declaración de clases + clase principal
Ejecución	El evaluador de expresiones fija el orden de los cálculos y almacenamientos temporales	Seguimiento a las instrucciones una a una	El orden de ejecución lo establece la clase principal y la relación entre los diferentes métodos de cada clase
Modelo de cálculo	Reescritura controlada por patrones	Máquina de Turing	Máquina virtual
Ciclos	Basados en recursión	Basados en estructuras	Basados en estructuras y también tiene ciclos recursivos
Seguimiento	Procesos debugging	Puntos de ruptura (breakpoint) Procesos debugging	Puntos de ruptura (breakpoint) Procesos debugging
Notación	Prefijo, posfijo	Infijo	Infijo
Arquitectura	Medianamente dependiente de la arquitectura	Totalmente dependiente de la arquitectura	Totalmente independiente de la arquitectura

En lo que corresponde al paradigma orientado a objetos la esencia radica en el concepto de Objeto que es una instancia de una clase, en sus características y en sus relaciones en la construcción de soluciones a

problemas computacionales. Los programas OO se construyen partiendo de la relación que se puedan establecer entre ellos. Al igual que en el paradigma imperativo, en un programa OO siempre existe un objeto que podría considerarse como principal y que tiene alta similitud con la función principal de los programas imperativos. Su dinámica es un poco diferente a la de los llamados a funciones dado que existen algunas relaciones adicionales que hacen la diferencia. El rastreo o seguimiento de los objetos, de sus atributos y de sus métodos es una opción bastante útil para conocer el estado parcial de funcionamiento de los programas OO.

La tabla 2 presenta el modelo simplificado en el cual se comparan los tres paradigmas, donde se especifican las características fundamentales que hacen la diferencia en cada uno de los paradigmas de forma que su estudio y profundización posibilite y permita el aprendizaje y entendimiento pleno del paradigma como tal.

Tabla 2. Simplificación de los tres paradigmas de programación

Tópico	Paradigmas		
	Funcional	**Imperativo**	**Orientado a Objetos**
Esencia	El concepto de función	El concepto de variable	El concepto de clase (atributos + métodos) que equivale a (variables + funciones)
Modelo computacional	Máquina de expresiones	Máquina de Turing	Máquina virtual
Recurso	Recursividad	Estructuras básicas	Relación entre objetos

En la programación funcional el concepto de función corresponde a un pequeño conjunto de instrucciones que pueden lograr, a su vez, un pequeño objetivo, que tiene un nombre identificador, que puede recibir argumentos y que puede retornar valores. La máquina de expresiones es el concepto computacional que permite resolver el recurso más íntimo y propio que tiene este tipo de programación y la recursividad es la capacidad que posibilita la programación funcional de que una función se llame así misma incorporando una dinámica que establece diferencias significativas en la solución de los problemas.

En el caso del paradigma imperativo el concepto de variable es el que gobierna la construcción de los programas, de hecho, un programa imperativo no es más que un conjunto de instrucciones que permiten cambiar progresivamente los contenidos de las variables de manera que, al final, se pueda lograr el objetivo deseado. La

máquina de Turing es precisamente el modelo computacional a través del cual se han diseñado los computadores en el mundo y que permite que un conjunto de instrucciones logren determinados objetivos en un computador. Las estructuras básicas corresponden a los conceptos de secuencias, condicionales y ciclos sobre las cuales se fundamentan las soluciones a los problemas computacionales desde la perspectiva de la programación imperativa.

La programación orientada a objetos se resume también de manera muy fácil: el concepto de objeto como una instancia de una clase en donde ésta no es más que la descripción técnica y sintáctica de un conjunto de atributos y métodos. Los atributos pueden asociarse con el concepto de variable pero embebido dentro de una determinada clase. Los métodos pueden asociarse con el concepto de función, como se concibe en la programación funcional y puede ser utilizado en la programación imperativa, sin constituirse en su esencia pero facilitando la comprensión, depuración, modificación y actualización de una solución imperativa. La máquina virtual es el modelo técnicamente ideal que permite a un programa realizado en un lenguaje orientado a objetos, en su gran mayoría, ejecutarse de manera totalmente independiente de la arquitectura en donde vaya a trabajar.

El recurso más importante que tiene el paradigma orientado a objetos es precisamente las relaciones que se pueden establecer entre los objetos, concebidos como instancias de clases, y más exactamente las relaciones que se pueden establecer entre los atributos de un objeto y los atributos de otro objeto o, también, los métodos de una clase y los métodos de otra clase.

A la pregunta ¿para qué sirve esta simplificación de los tres paradigmas en sus conceptos más elementales?, se puede responder: - para comprender la esencia de cada uno, lo que permite comprender su aplicación a través de los lenguajes de programación, - para establecer una relación entre los tres paradigmas que no son conceptos ni formas de pensar aisladas sino que involucran, dentro de un marco evolutivo, conceptos comunes, y - para que, en unión con los conceptos expuestos, pueda definirse la orientación de los esfuerzos docentes en las asignaturas, la eficiencia en el uso del tiempo por parte del estudiante y el diseño de los contenidos.

3. Resultados y discusión

3.1 Descripción de resultados

Este modelo simplificado se ha socializado con estudiantes y profesores de las asignaturas respectivas dentro de un programa de Ingeniería de Sistemas y Computación. La tabla 3 muestra algunas de las respuestas obtenidas cuando este modelo de simplificación de los paradigmas, se ha compartido con estudiantes de primeros semestres, con estudiantes de semestres intermedios y con estudiantes de semestres avanzados así como con docentes del área de las asignaturas respectivas.

Para efectos de la confiabilidad en las respuestas se han omitido los nombres propios de las personas que manifestaron, de manera directa, su opinión al respecto de esta simplificación. Como puede observarse el perfil de las respuestas es significativamente favorable pues se valora el esfuerzo de llegar a lo esencial en cada paradigma para que, a partir de su profundización, se pueda llegar a las últimas fronteras aplicativas del lenguaje.

Tabla 3. Algunas opiniones textuales sobre el modelo de simplificación

Actores	Respuesta 1	Respuesta 2	Respuesta 3	Respuesta 4
Docentes	Me parece una concepción muy sencilla y efectiva ya que permite establecer relaciones entre los paradigmas	Creo que simplifica demasiado los tres paradigmas pero lo pone a disposición de todos los estudiantes	Aunque las simplificaciones omiten elementos importantes, creo que esta versión más fácil aprender a programar	Me parece que es importante ir hacia lo esencial para poder enseñar las aplicaciones con mayor profundidad
Estudiantes Primeros Semestres	Esto le abre a uno un panorama de expectativas porque recién estamos comenzando	Muy bueno saber para dónde vamos así no sepamos en donde estamos ni de donde partimos	Creo que el profesor tiene muy claro el mundo de la programación y así se lo hace ver a uno	El propósito de esta simplificación lo anima a uno a estudiar con más ganas
Estudiantes Semestres Intermedios	Si me hubieran enseñado esto no se me habría hecho tan difícil las programaciones	Es importante que todos los docentes de progra (sic) sepan esto así como el profe (sic)	Para uno es importante, así sea tardío, tener clara la relación entre los paradigmas	Nunca es tarde para entender por fin cómo es eso de los objetos y la funcional
Estudiantes Últimos semestres	Ahora veo la utilidad de saber esto, ha debido ser más oportuno	Uno al final se casa con un paradigma pero con esta simplificación todos quedan muy fáciles	Se tiene gran posibilidad de avanzar más en la medida en que lo fundamental se entiende	Aprender a programar es aprender a llegar a lo esencial en los paradigmas de programación

En la Figura 3 se presenta la información estadística comparativa de las respuestas recibidas, mostrando que las opiniones tanto favorables como desfavorables[4] aportan a la discusión y todo pareciera indicar que con o sin experiencia, estudiantes y docente, coinciden en que el concepto de función, el concepto de variable y la amalgama entre ambas constituyen un excelente fundamento para abordar el estudio de la programación en sus diferentes paradigmas.

3.2 Discusión de resultados

Como se puede observar las opiniones favorables a esta propuesta de simplificación de las bases conceptuales de los paradigmas han sido mayores que las desfavorables. Entre las opiniones desfavorables se encuentran algunas como las que priorizan la profundización en cada paradigma y que sea el estudiante el que llegue a esta conclusión (sin descalificar la conclusión), las que establecen poca importancia a la simplificación de la esencia de los fundamentos de cada paradigma y consideran que es mucho más importante la expresión tecnológica y las habilidades que se deriven del conocimiento profundo del lenguaje y una más[5] que propone como esencia de los paradigmas los siguientes: para el paradigma funcional, la recursividad; para el imperativo, las estructuras básicas; y la herencia para la programación orientada a objetos.

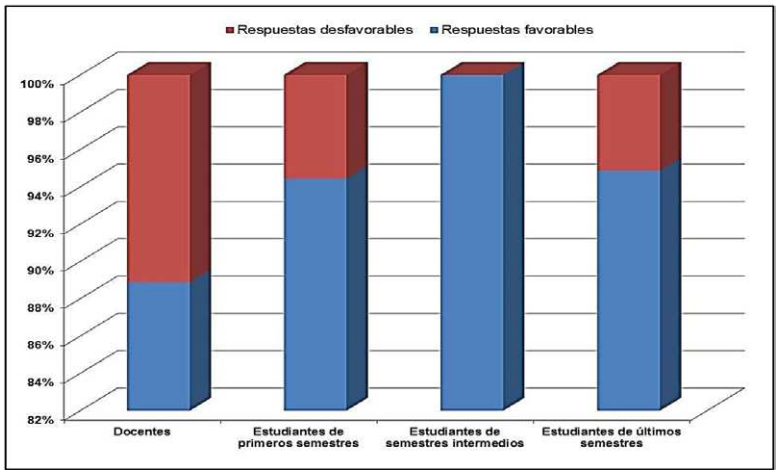

Figura 3. Estadísticas comparativas de las respuestas

4 Debe aclararse que como opinión favorable se entiende aquella que coincide y comparte los propósitos y la propuesta metodológica que contiene; como opinión desfavorable se entiende aquella que, en algún sentido, no comparte plenamente el modelo propuesto.

5 Debe admitirse que los argumentos esbozados para la última propuesta tienen un interesante sentido aunque no se puede negar que los conceptos no tienen una relación tan íntima como la simplificación que se propone en este artículo.

La preocupación mayor estaba en llegar a una simplificación verdaderamente útil y que acudiera a lo puramente esencial en cada uno de los paradigmas. Si bien al respecto pueden existir diferentes opiniones, ha llamado la atención que docentes y estudiantes de últimos semestres coinciden en pensar que lo expuesto efectivamente constituye la esencia de cada paradigma y que la relación entre ellos obedece a los conceptos mínimos que los conectan. Debe admitirse que las respuestas de los docentes fueron enriquecedoras y podrían nutrirse con opiniones de egresados no docentes y, mejor aún, recibir opiniones de programadores empíricos que tengan los fundamentos para evaluar los paradigmas desde una óptica menos instrumental y más conceptual.

Al acudir a este modelo de simplificación se ha podido encontrar en el concepto de función la verdadera esencia de los tres, un tema que se sale del espíritu del presente documento. Empero, vale la pena aclarar que la programación funcional se estudia con énfasis en dicho concepto, la programación imperativa se soporta en él para la construcción modular de sus soluciones y se construyen las clases teniendo en cuenta la importancia de los métodos (que son símiles de las funciones), pudiéndose llegar a niveles insospechados en el conocimiento, no solo de la expresión instrumental de la programación sino de la esencia conceptual de los paradigmas que subyacen a ellos.

4. Conclusiones

Es posible avanzar a pasos agigantados en el conocimiento de un lenguaje de programación si se conocen los fundamentos del paradigma que subyace a él y además si, de esta forma, se presenta a los estudiantes para que la programación no se reduzca a una buena utilización tecnológica sino a una buena conceptualización teórica dado que lo tecnológico es mucho más efímero que lo conceptual.

Es posible llegar a los elementos conceptuales que constituyen la esencia de un paradigma de programación y a partir de ellos encontrar en un lenguaje de programación la instrumentalización necesaria para demostrar lo que la teoría provee.

Es posible estudiar la programación de computadores a partir del estudio de los paradigmas sin que tengan que estar asociados a ningún lenguaje de programación y de esta forma entregarle a los alumnos la posibilidad de poder enfrentarse a cualquier lenguaje de programación siempre y

cuando se haga de la manera adecuada. También es posible, y se deja como una provocación investigativa, llegar a inferir los fundamentos de un paradigma a partir de conocer profundamente las características de un lenguaje de programación.

Si se profundiza en el análisis de los paradigmas de programación y su aplicación a través de los lenguajes de programación, es posible encontrar temas comunes que permitan simplificar la comprensión tanto de los paradigmas como de las aplicaciones que de ellos se deriven. Para ello es necesario que los docentes de las áreas de programación conozcan no solo la instrumentalización de los paradigmas a través de los lenguajes de programación sino la fundamentación que aquellas le proveen a estos.

5. Agradecimientos

Se agradece a la Universidad Tecnológica de Pereira toda la colaboración al respecto de las gestiones alrededor tanto del Doctorado como de la finalización de la Tesis Doctoral.

6. Referencias bibliográficas

AUSUBEL, D. (1986). Sicología Educativa: Un punto de vista cognoscitivo. Ciudad de México (México): Trillas. 562 p. ISBN: 968-24-1334-6

BRUNER, J. S. (1963). El proceso de la Educación. Ciudad de México (País): Editorial Hispanoamericana. 149 p. ISBN: 84-7112-179-4

BRUNER, J. S. (1991). Actos de significado. Madrid (España): Alianza Editorial. 168 p. ISBN: 9788420648125

DEITEL, P. & DEITEL, H. (2010). C How to program. New Jersey (NJ, USA): Pearson Education. 912 p. ISBN: 978-0132990448

HERMANN, W. (2000). The whole brain. New York (NY, USA): McGraw Hill. 334 p. ISBN: 978-0070284623

JEANDROCK, E.; EVANS, I.; GOLLAPUDI, D. & HAASE K. (2010). The Java EE 6 tutorial Basic Concepts. Boston: Pearson Education. 559 p. ISBN: 978-0137081851

MUÑOZ GUERRERO, L.E. (2014). Programación Funcional con Scheme. Pereira (Colombia): Calameo. E-Book.

PIAGET, J. (2001). Sicología y Pedagogía. México (México): Crítica. 176 p. ISBN: 9788484322030

SCHILDT, H. (2009). The complete reference Java. 7 ed. New York (NY, USA): McGraw Hill. 1274 p. ISBN: 978-0071808552

SCHILDT, H. (2010). The complete reference C. 4 ed. Berkeyley (USA): McGraw Hill. 805 p. ISBN: 978-0072121247

TREJOS BURITICÁ, O. (2000). La Esencia de la Lógica de Programación. Pereira (Colombia): Papiro. 336 p. ISBN: 958-33-1125-1

TREJOS BURITICÁ, O.I. (2012a). Aprendizaje en Ingeniería: un problema de comunicación. Pereira (Colombia): Tesis Doctoral (Doctor en Ciencias de la Educación)- Pereira (Colombia): Universidad Tecnológica de Pereira. 442 p.

TREJOS BURITICÁ, O.I. (2012b). Fundamentos de Programación. Pereira (Colombia): Editorial Papiro. 153 p. ISBN: 958823610

VAN ROY, P. (2006). Technique au Programming [e-book pre version]. Estocolmo (Suecia): Université Catholique de Louvain. 800 p. e-Book. Pre Version.
VAN ROY, P. & HARIDI, S. (2004). Concepts, Techniques and Models of Computer Programming. Cambridge (MA, USA): MIT Press. 936 p. ISBN: 978-0262220699