Artículo original / Original article / Artigo original

Software tangible: metáforas, representaciones visuales y actividades de apoyo didáctico para la enseñanza en construcción de software*

Luis Carlos Díaz Chaparro, Miguel Eduardo Torres Moreno***, José Hernando Hurtado Rojas****, Germán Alberto Chavarro Flórez*****, Edgar Enrique Ruiz García********

Resumen

Introducción. Desde la experiencia en el proceso de enseñanza aprendizaje de los primeros niveles de análisis, diseño, construcción y ejecución de algoritmos se ha detectado que realmente es complicado ilustrar y comprender conceptos de carácter intangible. Resulta ser un reto para los profesores generar analogías u otro tipo de ayudas conceptuales que de alguna manera aporten a la imaginación o abstracción que deben fundar los estudiantes en el momento de entender los conceptos básicos de programación. **Objetivo**. Definir un conjunto de representaciones visuales y metáforas que permitan establecer una base para el desarrollo de materiales didácticos, aplicaciones de software y actividades pertinentes que brinden soporte a la enseñanza y aprendizaje en programación. **Materiales y m**étodos. Con base en algunas teorías del área de la pedagogía y la revisión de proyectos semejantes, se definieron varias propuestas de representación, metáforas y actividades de aprendizaje significativo, a partir de una primera selección de conceptos de programación. **Resultados**. Se compartieron experiencias docentes y se formalizaron varias actividades que fueron desarrolladas directamente en clase por parte de los profesores involucrados en la línea de trabajo denominada Software Tangible. Se generó y se utilizó un conjunto de instrumentos físicos y herramientas de software de apoyo didáctico relacionadas e integradas bajo una consistente definición de representaciones visuales comunes. **Conclusión**. El uso de las representaciones visuales, metáforas, instrumentos y actividades de aprendizaje significativo permite apoyar la generación de las abstracciones necesarias que contribuyen a comprender los conceptos intangibles propios de la programación y ejecución de algoritmos.

Palabras clave: educación en Ingeniería, didáctica, multimedia, metáforas, actividades de aprendizaje significativo.

* Articulo derivado del proyecto de investigación "Metáforas y actividades de apoyo didáctico al proceso de enseñanza de conceptos fundamentales de programación y ejecución de algoritmos". Primera etapa de la línea de trabajo denominada Software Tangible (SWT). Grupo de investigación ISTAR, Departamento de Ingeniería de Sistemas de la Pontificia Universidad Javeriana - Bogotá D. C., 2011.

** Magíster en Ingeniería de Sistemas y Computación, Universidad de los Andes. Profesor del Departamento de Ingeniería de Sistemas, Pontificia Universidad Javeriana. Bogotá, Colombia.

*** Magíster en Ciencias de la Computación, Mississippi State University (USA). Profesor del Departamento de Ingeniería de Sistemas, Pontificia Universidad Javeriana. Bogotá, Colombia.

**** Magíster en Educación con énfasis en cognición y creatividad, Pontificia Universidad Javeriana. Profesor del Departamento de Ingeniería de Sistemas, Pontificia Universidad Javeriana. Bogotá, Colombia.

***** Magíster en Ciencias de la Computación, Universidad Estatal de Nueva York (USA). Profesor del Departamento de Ingeniería de Sistemas, Pontificia Universidad Javeriana. Bogotá, Colombia.

****** Magíster en Ingeniería de Sistemas y Computación, Pontificia Universidad Javeriana. Profesor del Departamento de Ingeniería de Sistemas, Pontificia Universidad Javeriana. Bogotá, Colombia.

Correspondencia: Luis Carlos Díaz Chaparro, email: luisdiaz@javeriana.edu.co

Artículo recibido: 05/02/2012; Artículo aprobado: 01/11/2013

Tangible software: metaphors, visual representations and didactic support activities for teaching in software construction

Abstract

Introduction. From the experience in the teaching-learning process of the first levels of analysis, design, construction and execution of algorithms, the difficulty of illustrating and understanding intangible concepts has been detected. Generating analogies or other kinds of conceptual resources to contribute to the imagination or the abstraction students require to understand the basic elements of programming, turns out to be a challenge for teachers. **Objective**. Define a set of visual representations and metaphors to establish a base for the development of didactic materials, software applications and pertinent activities to provide support to teaching and learning in programming. **Materials and methods**. Based on some pedagogical theories and on the revision of similar projects, some representations, metaphors and proposals of meaningful learning activities were defined from a first selection of programming concepts. **Results**. Teaching experiences were shared and several activities to be developed in class were formalized. Those activities were carried out by the teachers involved in the work line called Tangible Software. A set of physical and software tools for didactic support was developed under a constant definition of common visual representations. **Conclusion**. The use of visual representations, metaphors, instruments and meaningful learning activities supports the generation of the necessary abstractions that contribute for understanding the intangible concepts of programming and of the execution of algorithms.

Key words: education on engineering, didactics, multimedia, metaphors, meaningful learning activities.

Software tangível: metáforas, representações visuais e atividades de apoio didático para o ensino em construção de software

Resumo

Introdução. Desde a experiência no processo de ensino aprendizagem dos primeiros níveis de análises, desenho, construção e execução de algoritmos se detectou que realmente é complicado ilustrar e compreender conceitos de caráter intangível. Resulta ser uma meta para os professores gerar analogias ou outro tipo de ajudas conceituais que de alguma maneira contribuam à imaginação ou abstração que devem fundar os estudantes no momento de entender os conceitos básicos de programação. **Objetivo**. Definir um conjunto de representações visuais e metáforas que permitam estabelecer uma base para o desenvolvimento de materiais didáticos, aplicações de software e atividades pertinentes que brindem suporte ao ensino e aprendizagem em programação. **Materiais e métodos**. Com base em algumas teorias da área da pedagogia e a revisão de projetos semelhantes, definiram-se várias propostas de representação, metáforas e atividades de aprendizagem significativa, a partir de uma primeira seleção de conceitos de programação. **Resultados**. Compartilharam-se experiências docentes e se formalizaram várias atividades que foram desenvolvidas diretamente em classe por parte dos professores envolvidos na linha de trabalho denominada Software Tangível. Gerou-se e se utilizou um conjunto de instrumentos físicos e ferramentas de software de apoio didático relacionadas e integradas sob uma consistente definição de representações visuais comuns. **Conclusão**. O uso das representações visuais, metáforas, instrumentos e atividades de aprendizagem significativa permite apoiar a geração das abstrações necessárias que contribuem a compreender os conceitos intangíveis próprios da programação e execução de algoritmos.

Palavras importantes: educação em Engenharia, didática, multimídia, metáforas, atividades de aprendizagem significativa.

Introducción

Aprender a programar es una actividad difícil, tanto para los estudiantes que se enfrentan por primera vez a entender y usar algoritmos, en el contexto de la construcción de software, como para los profesores que aceptan el desafío de enseñarla. Es difícil para los estudiantes, ya que deben, entre otras cosas, cambiar su forma de pensar para entender cada uno de los conceptos fundamentales en el tema, y para los docentes, por contar con pocas ayudas didácticas a la hora de darlos a entender, en especial, porque unos y otros deben apelar a la imaginación para lograr entender conceptos intangibles que se relacionan unos con otros a distintos niveles de abstracción.

Se genera entonces la necesidad de contar con representaciones, metáforas y analogías que contribuyan a la ilustración de los principales conceptos de programación y ejecución de algoritmos de una manera coherente y lúdica. La comprensión en instancias tempranas de

tales conceptos es vital para hacer más fácil el posterior análisis y diseño de algoritmos.

El presente documento recoge los principales aspectos desarrollados en un primer proyecto o fase que pertenece a una línea de trabajo denominada Software Tangible (en adelante SWT) del Departamento de Ingeniería de Sistemas de la Pontificia Universidad Javeriana. El trabajo se centra principalmente en el estudio, identificación y propuesta de un conjunto asociado de metáforas, representaciones visuales y modelos consistentes como base fundamental para la generación de actividades significativas de aprendizaje, desarrollo de materiales didácticos y aplicaciones de software que permitan contribuir al mejoramiento del proceso enseñanza aprendizaje en el área de programación.

A grandes rasgos se pretende desarrollar tres grandes fases: la primera, enfocada a realizar una exploración inicial sobre iniciativas similares en el contexto de software educativo y el estudio de algunas bases conceptuales sobre preferencias de pensamiento y los estilos de aprendizaje de las personas con el fin de proponer representaciones para algunos conceptos de programación seleccionados, la puesta en común de algunos talleres de clase por parte de varios profesores, el análisis y diseño de actividades de aprendizaje significativo y el desarrollo de material de apoyo didáctico y herramientas de software.

Los conceptos de programación seleccionados inicialmente para la primera fase exploratoria tienen que ver con tipos primitivos de datos, vectores y matrices, estructuras de programación (asignaciones, ciclos y condicionales), el concepto de función, programación orientada a objetos (objetos, mensajes, métodos, atributos), procesos relacionados con la solución de problemas y algunas generalidades sobre el área de ingeniería de software.

Posteriormente, en una segunda y tercera fase, se pretende perfeccionar y formalizar las propuestas visuales, las analogías, las metáforas y las actividades de aprendizaje significativo con el fin de crear un repositorio o banco de talleres y aplicaciones de apoyo. El banco estaría enriquecido con la generación de aplicaciones como juegos, simuladores, mundos virtuales y ambientes de aprendizaje en general que permitan brindar un mejor acercamiento a la forma de funcionamiento de los algoritmos y el entendimiento de sus conceptos asociados.

En la sección de Materiales y Métodos se presentan algunos elementos de soporte conceptual, tanto para las propuestas de representación como para las actividades didácticas, y la revisión a algunos proyectos de referencia sobre software educativo. Igualmente, se describen las propuestas visuales y metáforas para los principales conceptos de programación seleccionados, las actividades significativas de aprendizaje y las herramientas didácticas que se han generado en el proyecto. En la sección de Resultados se hace un recuento de las principales consecuencias y observaciones sobre la aplicación de las actividades propuestas en el salón de clase.

Materiales y métodos

El proceso que se ha seguido en el desarrollo de la primera fase exploratoria se ha centrado en el estudio de aspectos didácticos, la revisión de trabajos relacionados en el área de software educativo, la elaboración de pequeños ejercicios pilotos y experiencias reales en clase, la elaboración de material de apoyo didáctico y el uso de aplicaciones de software.

En el ámbito de la didáctica y la pedagogía se han revisado varios trabajos como soporte al desarrollo de las representaciones propuestas y las actividades de aprendizaje significativo. Sin embargo, para los propósitos de la primera fase del proyecto, se hizo mayor énfasis en el uso del modelo de pensamiento integral elaborado por Ned Herrmann (1996), los estilos de aprendizaje, referente a la manera en la cual una persona se apropia del conocimiento (Allison & Hayes, 1996; Rena & Pratt, 2003), la taxonomía de Bloom (Anderson & Krathwohl, 2001; Anderson & all, 2001) que divide en dominios jerárquicos la forma en que las personas aprenden, así como en la noción de nivel de abstracción.

El modelo de pensamiento integral y los estilos de aprendizaje

El modelo de pensamiento integral de Ned Herrmann (1996) se basa en la forma de funcio-

namiento del cerebro a partir del cual el autor define una estructura de cuatro cuadrantes en donde cada uno está asociado con ciertas cualidades de las personas. El modelo (figura 1) está acompañado de un instrumento de pruebas desarrollado por Herrmann con el que se puede establecer un perfil para determinar las preferencias de pensamiento de un individuo. El instrumento lleva perfeccionándose más de 40 años, desde los años 70, y aplicándose a más de un millón de personas alrededor del mundo.

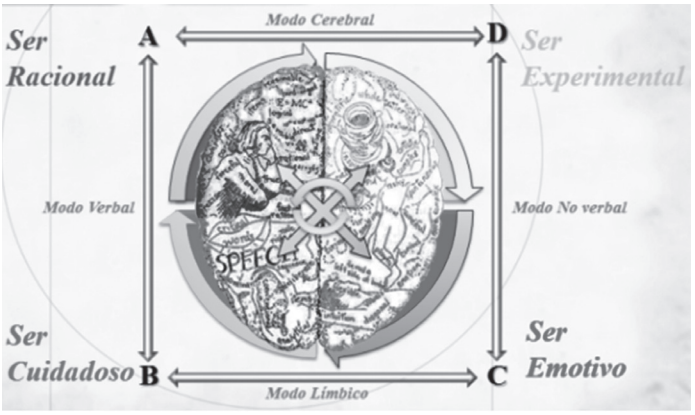


Figura 1. Cuadrantes del Modelo Integral de Pensamiento de Ned Herrmann (Herrmann, 1996). Los estilos de aprendizaje de las personas propuestos por Rena & Pratt (2003), tienen que ver con la preferencia visual-verbal, visual-espacial, auditiva, kinestésico o táctil, relacional-intrapersonal, relacional-interpersonal y lógico-matemático que tienen las personas a la hora de aprender un determinado tema. Estos estilos se resumen en figura 2.

Los estilos de aprendizaje de las personas propuestos por Rena & Pratt (2003), tienen que ver con la preferencia visual-verbal, visual-espacial, auditiva, kinestésico o táctil, relacional-intrapersonal, relacional-interpersonal y lógico-matemático que tienen las personas a la hora de aprender un determinado tema. Estos estilos se resumen en figura 2.

Finalmente, en la figura 3 se presenta uno de los resultados del trabajo realizado por Barros, Sánchez y Rojas (2007), en el cual se muestra la integración entre las preferencias de pensamiento del modelo integral y los estilos de aprendizaje, que permite seleccionar cierto tipo de actividades didácticas dependiendo del tema y el interés del profesor. El símbolo de la mano con el pulgar arriba significa que el estilo de aprendizaje es favorable para ese cuadrante de preferencia de pensamiento; el símbolo que muestra la palma de la mano significa que debe usarse con precaución, y el símbolo de la mano con el pulgar abajo significa que es preferible no utilizarlo.

La utilización del modelo de Ned Herrmann, los estilos de aprendizaje y la taxonomía de Bloom han permitido, en gran medida, soportar las propuestas desarrolladas y enriquecer los criterios de selección de las representaciones o metáforas propuestas, así como el desarrollo de las actividades didácticas en cada caso.

Otras aproximaciones de software educativo asociadas con el proyecto

Se hizo una revisión de varios proyectos en el área de software educativo que tuvieran relación con SWT. La mayoría de tales proyectos tienen que ver con la visualización de algoritmos en tiempo de ejecución, ambientes virtuales de aprendizaje, sistemas e-learning y sistemas multimedia, entre otros, que de una u otra manera aportan o complementan el desarrollo del proyecto propuesto.

En general, se pueden encontrar muchos otros desarrollos. Sin embargo, en ocasiones se trata de aplicaciones que simplemente hacen

un cambio de formato entre el contenido de un libro o un medio tradicional, a un medio de soporte digital que se puede ver desde el computador. También existen otras aplicaciones relacionadas con juegos, muchos de los cuales en sí mismos se alejan finalmente de lo que se quiere aprender o enseñar, y algunos otros, que se acercan a la propuesta del presente trabajo (en cuanto al uso de metáforas), como aquellos basados en piezas de Lego (Hood & Hood, 2005) pero cuyas representaciones están más enfocadas a la solución de pequeños problemas que a los conceptos de programación y ejecución de algoritmos propiamente dichos.

De otro lado, varios de los proyectos de visualización de algoritmos, como los mostrados en Hosking (1996), Masterson & Masterson (2001), Carlisle, Wilson, Humphies & Hadfield, 2005) (De Pauw, Lorenz, Vlissides & Wegman), resultan ser demasiado técnicos y se centran más en herramientas de desarrollo de software propiamente dicho o en la depuración de programas que en un apoyo didáctico para la comprensión de los temas sobre los cuales se ocupan.

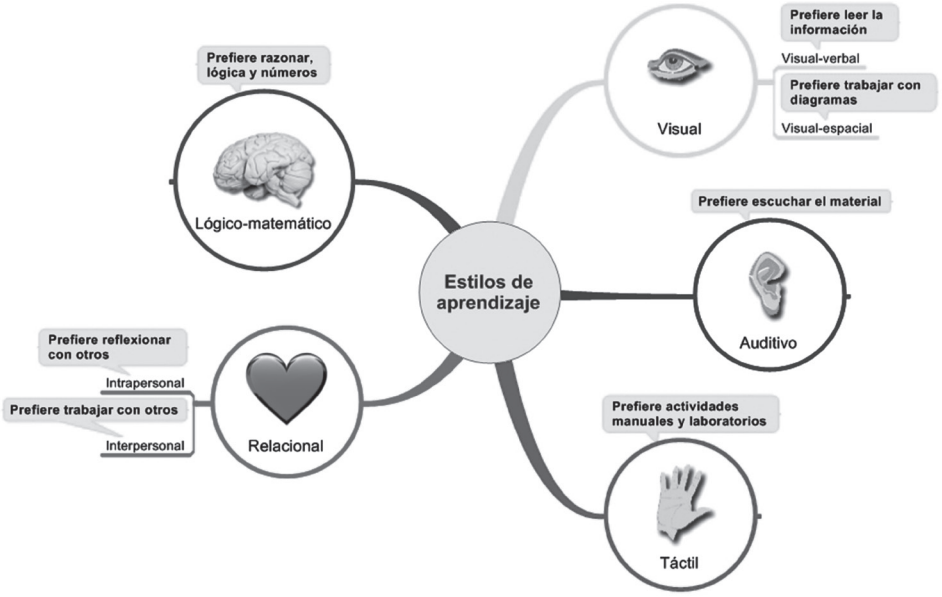
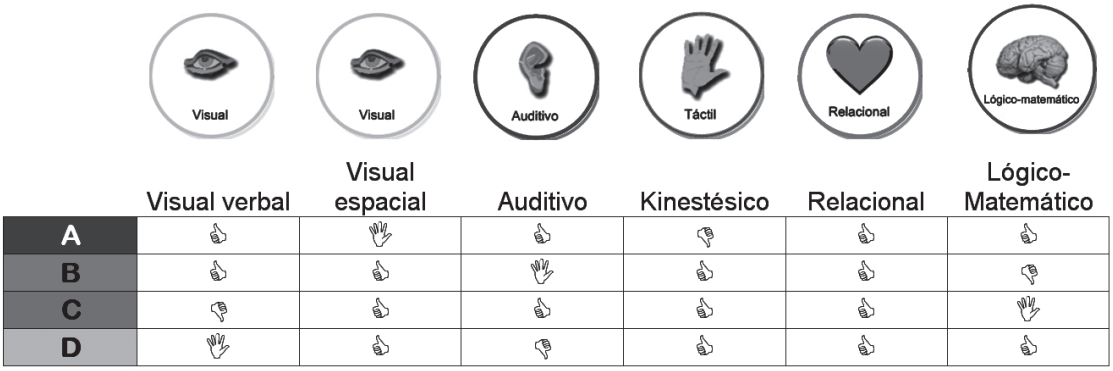


Figura 2. Mapa mental de los estilos de aprendizaje propuesto por Rena & Pratt (2003)



	Visual verbal	Visual espacial	Auditivo	Kinestésico	Relacional	Lógico-Matemático
A	👍	✋	👍	👎	👍	👍
B	👍	👍	✋	👍	👍	👎
C	👎	👍	👍	👍	👍	✋
D	✋	👍	👎	👍	👍	👍

Figura 3: *Relación preferencias de pensamiento vs estilos de aprendizaje* propuesta por Barros, Sánchez y Rojas (2007)

En el caso particular de herramientas para la enseñanza de la programación se han creado diversas aplicaciones, en diferentes grupos de investigación y universidades, con el fin de apoyar los procesos de aprendizaje de algún lenguaje de programación particular, reforzar conocimientos relacionados con el proceso de definición, prueba e implementación de algoritmos, así como los simplemente orientados a acercar a los estudiantes al funcionamiento interno del computador. Más recientemente Powers et al. han hecho la siguiente clasificación:

- *Herramientas narrativas*: soportan narrativas para contar historias, i. e. Alice
- *Herramientas visuales*: permiten la construcción de programas por medio del uso de interfaces drag and drop, i. e. Alice, Scratch.
- *Herramientas de flujo de datos*: que por medio de representaciones gráficas permiten representar flujos y orden de cómputo, i. e. PSeInt.
- *Construcciones de salida específica*: donde los estudiantes pueden construir físicamente elementos que no solo interactúan por los medios usuales pantalla y teclado, sino que usan multimedia o movimiento de partes (robots), i.e. Lego MindStorms.
- *Herramientas de lenguajes específicos*: donde los usuarios novatos pueden usar versiones reducidas de un lenguaje particular a medida que su experticia se incrementa, i.e. RoboLab.
-

En el contexto de SWT se han revisado de manera más detallada las herramientas Alice, Scratch y Lego Mindstorms, dado que el enfoque de estas herramientas se relaciona directamente con la propuesta de nuestro trabajo.

Alice: Alice (Alice.org) es un entorno de animación en 3D que permite a los usuarios construir entornos virtuales, bien sea como historias o, bien, como juegos interactivos construido en la Universidad Carnegie Mellon. En sus inicios, Alice fue construido con el fin de apoyar la construcción de entornos 3D y modelar el comportamiento de objetos ubicados en dichos entornos por medio del uso de drag and drop de objetos 3D, que permite modificar sus atributos o comportamiento durante el tiempo para construir una animación o un juego.

En la actualidad, Alice es usado como herramienta base en cursos introductorios de programación estructurada, y orientada a objetos para presentar a los estudiantes universitarios y de últimos años de Educación Secundaria los conceptos básicos de programación, solución de problemas y elementos de la programación y análisis orientado a objetos, por medio de la programación visual de comportamientos y acciones de los elementos 3D. Además, cuenta con herramientas de apoyo para docentes (Alice Teaches Object Oriented Computer Programming To Kids).

Algunas ventajas del uso de Alice en cursos introductorios de programación de nivel universitario, reportadas por Cooper & Pausch (2000) son:

- Alto sentido del diseño: se ve que al final del curso los estudiantes prefieren ir al papel para expresar su proceso de análisis y diseño de los problemas.
- Contextualización de los conceptos de orientación a objetos (clases, objetos, atributos, comportamientos, encapsulamiento, herencia, uso adecuado de tipos de datos, etc.).
- Aprendizaje, comprensión y uso de la construcción iterativa e incremental.

Como única debilidad se expresa la dificultad que tienen los estudiantes para pasar a lenguajes formales como son C++ o Java en particular en lo que tiene que ver con la sintaxis, pues con Alice no se tiene la oportunidad de vivir y corregir dichos tipos de errores.

Scratch: Scratch es una herramienta diseñada por el MIT Media Lab, para el aprendizaje y enseñanza de habilidades para la solución de problemas de manera sistemática y colaborativa para niños entre los 8 y 16 años (Resnick et al., 2009), que permite construir programas a partir de instrucciones que se asemejan a rompecabezas. Además, tiene una numerosa base de datos de proyectos creados por la comunidad, lo cual permite la reutilización de proyectos. Posee, también, una gran cantidad de material para docentes y usuarios.

En la actualidad Scratch está ganando popularidad en el ámbito de la enseñanza básica secundaria y su uso se está extendiendo, gracias

a que está integrada al PC (Pi, 2012), es un computador de bajo costo y capacidad reducida diseñado en Inglaterra por una organización sin ánimo de lucro, con el fin de apoyar y motivar el uso y aprendizaje de conceptos de programación de computadores en países pobres.

Lego MindStorms: Lego MindStorms es una plataforma de construcción de robots y programación, basada en las piezas proveídas por LEGO, de amplio uso en entornos universitarios, incluso con la propuesta de un currículo creado para apoyar la solución de problemas cotidianos y aplicación práctica de conceptos de programación, análisis lógico y sistemático de problemas (Klassner & Anderson, 2003).

Otro tipo de trabajos relacionados con aspectos educativos en ingeniería, que pueden contribuir al desarrollo del proyecto, tienen que ver con la propuesta del uso de juegos, desde otra perspectiva a la mencionada anteriormente o, como el proyecto Cupi2 de la Universidad de los Andes (Villalobos et al., 2006), con el estudio de la problemática de aprender a programar, vista desde distintas perspectivas.

Nivel de abstracción, metáforas y ayudas didácticas

Un aspecto importante para la definición de representaciones y metáforas es la noción de "*nivel de abstracción*", concepto que es utilizado en numerosos ámbitos del conocimiento y que, para el caso del área de construcción de software, es crucial en el sentido de tratar de esconder ciertos detalles o información no relevante en un momento o contexto específico a la hora de explicar un concepto particular.

No hacer clara la noción de nivel de abstracción en el momento de explicar un concepto puede terminar por confundir a la persona que aprende. El profesor debe explicar sus bondades y hacer explícito el contexto de trabajo desde el cual imparte la información que quiere enseñar. Este aspecto adquiere mayor relevancia, dada la idea de proponer representaciones visuales o analogías para la mayoría de los conceptos intangibles que caracterizan la construcción de software. Hacer claridad sobre el nivel de abstracción requerido antes de explicar los distintos conceptos que sean del caso, a partir de las representaciones visuales definidas, es también responsabilidad de quien enseña.

Otro aspecto fundamental y condición importante dentro de los ámbitos de representación es que se requiere una clara relación entre las metáforas definidas y el concepto adyacente que se quiere presentar. Es evidente que se requiere asegurar un hilo conductor asociado con las relaciones existentes entre las distintas representaciones y su contraparte con los distintos detalles del concepto en cuestión.

La relación entre el concepto y la representación es aún más trascendental cuando se quiere pasar a un nivel de abstracción diferente o cuando se quiere explicar ciertos conceptos que dependen de otros ya establecidos; es importante, también, tener en cuenta una coherencia o hilo conductor entre niveles o dimensiones de los conceptos y sus representaciones.

Propuestas de representación

Con el resultado del trabajo de Barros Sánchez, & Rojas (2007), el estudio de las referencias de algunos trabajos en el área de software educativo y teniendo en cuenta los aspectos asociados con los niveles de abstracción, el uso de metáforas y la puesta en común de algunas actividades docentes, realizadas en las clases de los docentes que participan en la primera fase de SWT, se establecieron dos frentes de trabajo para apalancar el entendimiento de los conceptos de programación y ejecución de algoritmos seleccionados:

- Desarrollar un conjunto de instrumentos físicos y herramientas de apoyo didáctico, relacionadas e integradas bajo una consistente definición de representaciones visuales comunes.
- Proponer un conjunto de actividades de aprendizaje significativo, basado en las representaciones, metáforas y analogías definidas que usen potencialmente dichos instrumentos y herramientas.

Representación de la memoria y los tipos primitivos de datos. Se trata de la utilización de elementos geométricos elementales y el uso de colores para representar los tipos primitivos de datos presentes en la mayoría de los

lenguajes de programación. Esta primera propuesta de representación visual, y sus posibles variantes, se constituye en la base fundamental para el diseño o adaptación de elementos físicos y la construcción de herramientas de software que brinden soporte a las actividades de apoyo didáctico para la enseñanza y el aprendizaje de los conceptos asociados con el almacenamiento de datos en la memoria principal del computador y el uso de los valores almacenados en ella como variables dentro de un programa.

La propuesta de representación involucra los detalles relacionados con la representación visual de la memoria del computador y los tipos primitivos de datos presentes en la mayoría de los lenguajes de programación: entero, carácter, booleano, flotante y los apuntadores relacionados con estos tipos de datos.

Es suficiente una revisión a unos cuantos libros, manuales y ayudas, relacionados con la programación y ejecución de algoritmos, para advertir que un gran número de ellos presentan el concepto de memoria del computador como un inmenso espacio que contiene una gran colección de "casillas" a manera de un estante o armario en el cual cada una de ellas puede almacenar datos. Por lo general, se presentan gráficas o imágenes de matrices o tiras de tales casillas dentro de las cuales se pueden observar los valores almacenados. En resumen, la figura 4 muestra una propuesta de representación visual a partir de la cual el estudiante puede tener una abstracción para entender inicialmente el concepto de memoria.

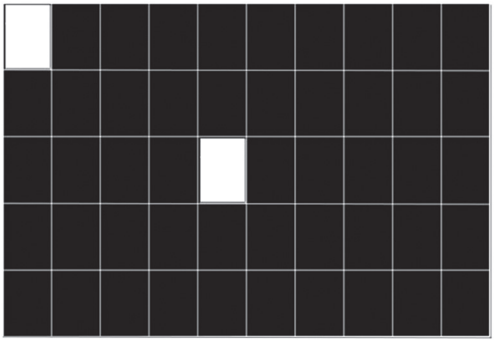


Figura 4. Propuesta de representación básica de la memoria

Es clave tener presente el alcance de la representación y el nivel de abstracción que se desea definir con cada una de las representaciones propuestas. Es claro que el concepto de memoria se podría explicar desde varios puntos de vista y niveles de abstracción, niveles que se inician en el área física del hardware y la electrónica relacionada, pasando por la –nuevamente– "representación" binaria de unos y ceros de tales valores, así como el uso de palabras o grupos de valores binarios, como el byte, para representar datos de más alto nivel. Para el caso particular de la representación de la memoria del computador definida en la figura 4, el nivel de abstracción elegido omite intencionalmente los detalles electrónicos y físicos de cómo se almacenan o se interpretan los datos para verlos desde un punto de vista más elevado. La representación en este nivel de abstracción permite, sin pensar en los detalles, tener presente que es relevante establecer qué tipo de dato se está almacenando en la memoria del computador, el lugar o la dirección de memoria asociada y su tamaño en términos del número de bytes utilizados; información de suma relevancia a la hora de interpretar y realizar operaciones con ellos.

Además de la representación de la memoria como tal, es importante contar con una representación propia para cada uno de los posibles tipos de datos que pueden ocuparla. Se propo-

ne entonces un conjunto de formas geométricas y colores para cada uno de los tipos básicos de datos:

Entero: se define como un rectángulo de color azul. Una posible variación de la representación es el ancho del rectángulo, si se tiene en cuenta el número de bytes requerido para el almacenamiento de un dato entero en la memoria. En principio, y dada la idea de ocultar algunos detalles de bajo nivel, para el ancho de esta figura solamente se tendrá en cuenta su relación con el tipo de dato flotante que también se representa a través de un rectángulo.

Carácter: se define como un círculo de color verde. Para las dimensiones de la figura podría ser discutible nuevamente el número de bytes que ocupa en la memoria. Sin embargo, para el propósito y el nivel de abstracción que se requiere usar para introducir el tema inicial de los tipos primitivos de datos lo más relevante es la diferenciación entre ellos en términos de lo que almacenan y cómo lo almacenan.

Booleano: se define como un rombo de colores blanco y gris.

Flotante: su representación visual es la de un rectángulo de color morado. El ancho podría variar, pero como se mencionó anteriormente, lo que sí debe ser claro es su diferenciación con el tipo de dato entero. El ancho del rectángulo para el tipo de dato flotante será el doble de ancho que el tipo entero.

Apuntador: un tipo de dato interesante es el apuntador. Su representación visual es la de un triángulo de color rojo, acompañado de una flecha que señalará la posición hacia el dato que apunta. La figura 5 muestra cada una de las representaciones descritas anteriormente.

Representación de arreglos: dentro del marco de SWT y como iniciativa de un estudiante (Anderson & Krathwohl, 2001), se propone una representación para el tema de vectores y matrices centrada en una analogía con las cubetas utilizadas para almacenar huevos. En la analogía se definieron diferentes colores de huevos que representaban los distintos tipos de datos que podían almacenar los arreglos (vectores o matrices), etiquetas para representar las posiciones en el arreglo y la cubeta de huevos como el contenedor de los datos relacionados. La figura 6 ilustra un ejemplo de un vector de cuatro posiciones. Lógicamente, existen las representaciones de los demás tipos de datos básicos en las cuales se mantiene el uso del color definido en la representación visual de los tipos de datos básicos presentados anteriormente.

Representación del concepto de objeto. Otra aproximación que adicionalmente busca integrar representaciones asociadas (como la de los tipos primitivos de datos) tiene que ver con el concepto de objeto en el contexto de la programación orientada a objetos. La figura 7 muestra la representación propuesta; la idea está inspirada en algunas ilustraciones de una guía sobre el tema de la tecnología orientada a objetos, realizada por Taylor (1991). A partir de esta representación se pueden explicar más fácilmente todos los conceptos clave de la programación orientada a objetos.

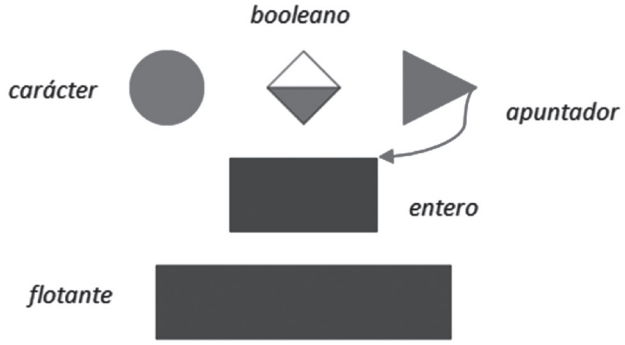


Figura 5. Representación de tipos primitivos de datos

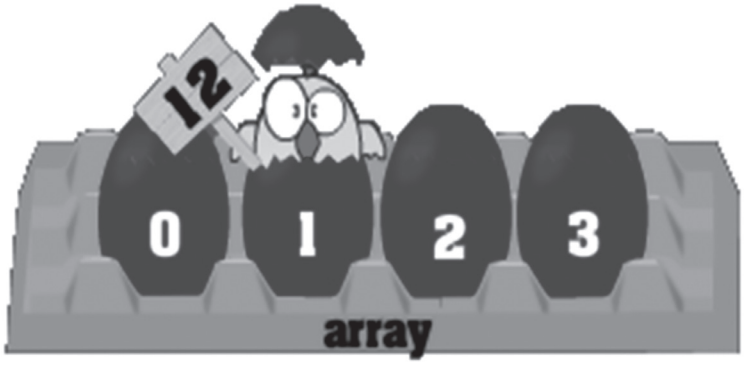


Figura 6. Representación de un vector de enteros (color azul) en el tutor de VMCreative (Montenegro, 2009)

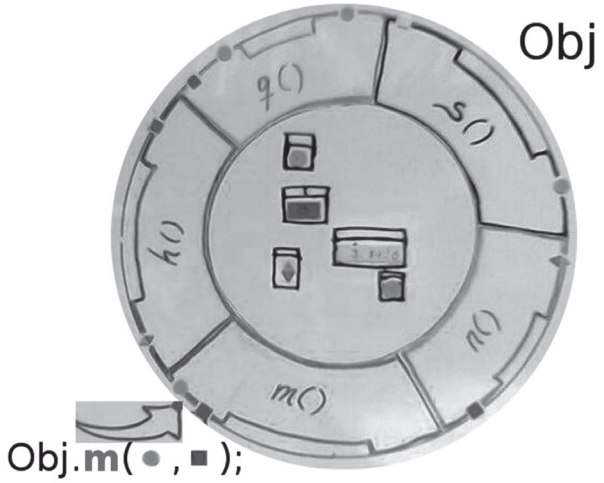


Figura 7. Propuesta a mano alzada de la representación del concepto de Objeto

En la propuesta de representación del concepto de objeto se tienen en cuenta varios detalles:

- La representación del objeto contempla otras representaciones importantes: la representación de una función o método y la representación de sus atributos.
- Los atributos corresponden primordialmente a los tipos básicos de datos cuya representación ya fue explicada. Los atributos más complejos pertenecen a clases que se pueden representar recursivamente como otros objetos.
- Los métodos se representan como “cajas negras” que tienen entradas y salidas. Dichas entradas y salidas están en términos de tipos de datos, para lo cual se utiliza nuevamente la representación correspondiente (figuras geometricas y colores).
- El objeto como un todo se representa a través de un gran círculo que tiene una capa exterior compuesta por la representación de cada método, y una capa interna que contiene las representaciones de sus atributos.
- Los llamados a los métodos del objeto se representan con una flecha que es acompañada por el mensaje respectivo (nombre del objeto, nombre del método y sus parámetros).

El pseudocódigo como representación de algoritmos. Las representaciones clásicas de

las instrucciones de programación siempre han estado dadas por el uso de diagramas de flujo y el pseudocódigo. En este ámbito, la línea de trabajo del grupo se ha centrado en buscar la mejor manera de usar estas representaciones de tal forma que se puedan garantizar y unificar los conceptos básicos para desarrollar algoritmos antes que el estudiante se involucre con un lenguaje de programación de alto nivel.

Como uno de los estilos de aprendizaje más fuertes y preferidos por las personas es el visual, se hace necesario, además de la idea de unificar conceptos de programación, sin importar el lenguaje de alto nivel, utilizar componentes gráficos asociados con los principales bloques de instrucciones en un programa. Como ejemplo, la figura 8 muestra un algoritmo que permite conocer si un número entero ingresado por teclado es primo o no; se utilizan en la figura componentes gráficos definidos en una herramienta denominada Psicoder (Vásquez & Páez, 2008).

Metáforas sobre tareas y procesos. Se han desarrollado varias metáforas asociadas con tareas técnicas de programación y procesos de nivel superior para poder hacer analogías con el fin de contextualizar un tema, identificar problemáticas, entender conceptos y establecer conclusiones.

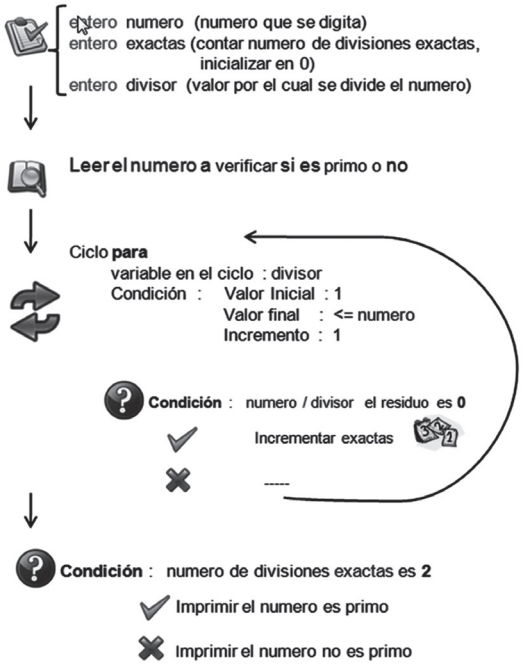


Figura 8. Ejemplo del uso de los componentes visuales para describir un algoritmo

Algunos ejemplos tienen que ver con la construcción de artefactos o la solución de problemas en equipo, como aquellos relacionados con labores de robots, en temas de estrategias de solución de problemas como “divide y vencerás”, la integración de funciones en un programa, el proceso de ingeniería de software, el establecimiento de requerimientos, manejo de recursos, tiempo y el empleo de roles de cada uno de los participantes.

Actividades de aprendizaje significativo, generación de instrumentos didácticos y aplicaciones de software

Como ilustración del uso de las representaciones propuestas y la puesta en práctica de varios principios del pensamiento integral, los estilos de aprendizaje y la taxonomía de Bloom se presenta a continuación un conjunto de actividades y ejemplos, así como el desarro-

llo y uso de instrumentos de apoyo didáctico y herramientas de software que han permitido enriquecer el proceso de enseñanza aprendizaje en el área de construcción de software. No es del alcance de este documento mostrar los detalles de todas y cada una de estas experiencias.

En general, a partir de las representaciones propuestas, se pueden crear, adaptar o desarrollar elementos físicos para utilizar en el tablero de clase, ingeniarse, por ejemplo, actividades de clase que permitan a los estudiantes jugar con tales elementos, o construir distintos escenarios o aplicaciones de software derivadas, acordes con la estrategia que desee realizar el profesor, quien al final debe formalizar el concepto o los conceptos que quiere dejar en sus estudiantes.

Sobre la memoria y los tipos primitivos de datos. El primer ejemplo tiene que ver con la generación de talleres lúdicos en clase en los cuales se utilizan bosquejos bastante bien elaborados sobre las representaciones definidas: cartulinas que contienen una matriz con celdas numeradas (la memoria) y uso de pequeñas cajas que pueden pegarse a las celdas de dicha matriz (para representar las potenciales variables que se pueden generar en un programa); las cajas permiten además ilustrar el nombre de la variable y el valor del dato que contiene. Con estos elementos, los estudiantes deben seguir algunas expresiones en algún lenguaje de programación para realizar una simulación del estado de la memoria en cuanto al almacenamiento de los datos y el uso de variables, acorde con cada paso de ejecución de un pequeño algoritmo.

Un segundo ejemplo tiene que ver con el uso del tablero para discutir una prueba de escritorio y las circunstancias específicas de un determinado ejemplo del uso de variables, tipos primitivos de datos y operaciones dentro de un algoritmo usando las representaciones geométricas definidas.

Se han propuesto aplicaciones de software dentro del marco de SWT para hacer más sencillas las explicaciones, las actividades lúdicas y todas aquellas ventajas que puede brindar un programa para ilustrar desde el computador escenarios de práctica y juego. La figura 9 muestra la interfaz de una aplicación de software denominada “TAVA: Herramienta didáctica para apoyar el proceso de enseñanza-aprendizaje en los conceptos de variables, tipos de datos y apuntadores” (Barón & Ronderos, 2009).

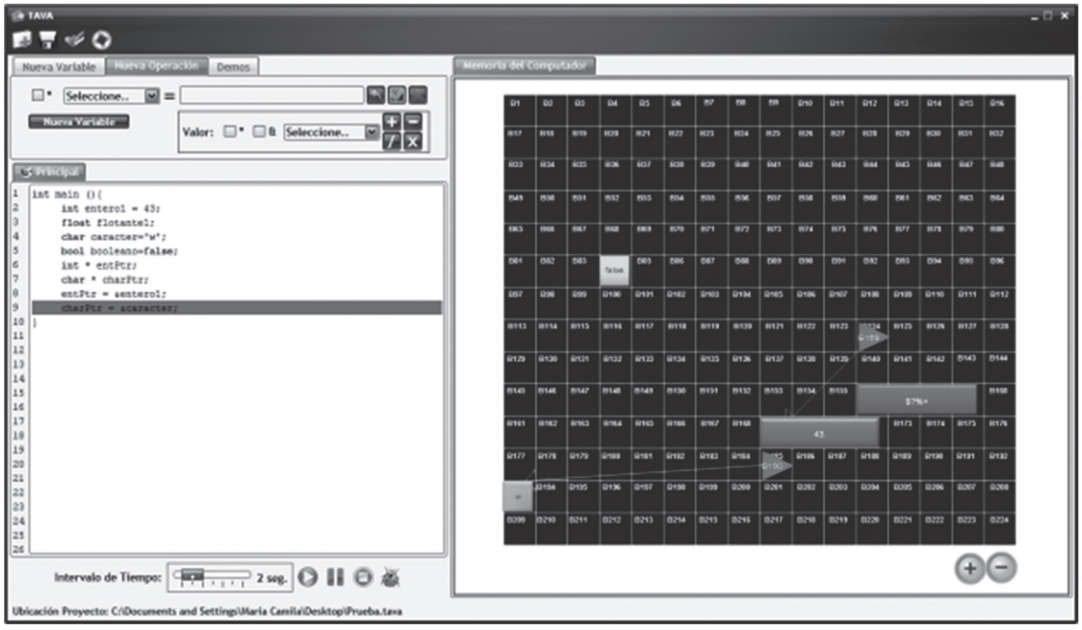


Figura 9. Interfaz de la herramienta TAVA (Barón & Ronderos, 2009)

Con TAVA los estudiantes pueden seguir pequeños fragmentos de código y ver una animación de lo que sucede en la memoria durante la fase de declaración de variables, y en la ejecución de cada uno de los pasos de un programa, utilizando las representaciones ya señaladas. El uso de herramientas de software permite contar con una mejor apreciación de los conceptos de una manera más didáctica. A largo plazo, se pueden utilizar los mismos principios de representación no solo para explicar los conceptos como tales, sino con el objetivo de desarrollar y depurar código en un lenguaje de programación de alto nivel.

Sobre los arreglos de vectores y matrices. La representación definida en este caso permite realizar un sinnúmero de actividades lúdicas y ejemplos durante la clase en el momento de explicar y diferenciar los conceptos centrales del uso de arreglos: el contenedor de los datos (la cubeta de huevos), la homogeneidad del tipo de dato para todo el arreglo (todos los huevos del mismo color), el tamaño del arreglo (dimensiones de la cubeta), la diferencia entre la posición de una dato (un lugar para un huevo en la cubeta) y el contenido del mismo (el dato en el interior del huevo). También es posible utilizar la propuesta para recrear situaciones sobre el tema de recorridos y orden dentro un arreglo.

Como ayuda didáctica se propuso la realización del prototipo de una herramienta de software que permitiera visualizar todos los conceptos y los ejemplos que fueran posibles haciendo uso de la representación propuesta. El resultado fue el prototipo de una aplicación denominada VMCreative: "Herramienta didáctica de apoyo al proceso de enseñanza y aprendizaje de los conceptos básicos de vectores y matrices" (Montenegro, 2009).

Sobre el concepto de objeto. Como ejemplos de la representación propuesta para un objeto se puede mencionar la realización de talleres en clase en los cuales los estudiantes deben tratar de dibujar objetos particulares a partir de la especificación de las clases correspondientes en UML y algunas sentencias específicas de código. Su uso ha sido parte clave e integral de la explicación de los conceptos de clase y objeto, encapsulamiento, paso de mensajes, estado de un objeto, etc. en los cursos que abordan el tema.

De igual forma, en este tema se propuso el desarrollo de una herramienta piloto para contar con un apoyo didáctico que permitiera, en el mismo contexto o ambiente de trabajo, mostrar la definición de clases en UML, definir las instrucciones para crear objetos y visualizar los objetos creados bajo la representación definida. La figura 10 muestra la interfaz de una aplicación de software que cumple con estas exigencias y que se denominada "POOInteractiva: Prototipo de una herramienta didáctica para apoyar el proceso de enseñanza-aprendizaje de conceptos fundamentales de programación orientada a objetos" (Rubio, 2008).

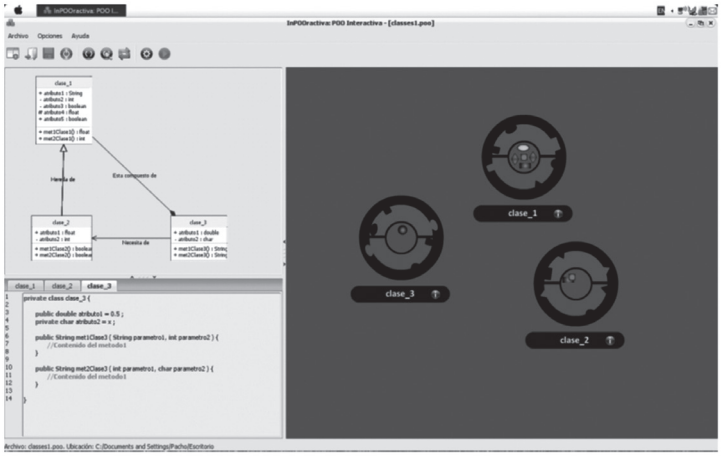


Figura 10. ***Interfaz de los espacios de trabajo de la herramienta POOInteractiva*** **(Rubio, 2008)**

El prototipo de la herramienta está compuesta de tres espacios de trabajo: el primero relacionado con el modelo de clases UML que define un proyecto de trabajo, el segundo un editor de código para generar las instrucciones de creación de objetos, a partir del modelo UML definido y, finalmente, un espacio para visualizar el estado de la aplicación, acorde con la ejecución de cada una de las sentencias previamente escritas. Con POOInteractiva el estudiante puede contar con un apoyo muy interesante para hacer sus abstracciones alrededor del concepto de objetos.

Para nadie es un secreto que la enseñanza del paradigma de programación orientada a objetos presenta dificultades para los estudiantes que deben alcanzar un alto nivel de abstracción. Una parte clave es que el estudiante debe pensar y modelar en términos de clases (un mundo estático), al tiempo que debe imaginarse cómo interactuarán los objetos derivados en el contexto de ejecución de sus programas (un mundo dinámico), aspecto nada trivial para quien se inicia en el tema. Poder apreciar los objetos derivados (utilizando la representación definida) a partir del modelo UML ayuda a cerrar la brecha.

Sobre el uso de pseudocódigo. Usualmente las primeras actividades de programación básica se hacen utilizando pseudocódigo. Pero normalmente la organización de dichas actividades requiere acuerdos entre los profesores o los instructores para definir de manera más precisa la forma de usar ese conjunto de instrucciones que adicionalmente deben "funcionar en el papel". Se trata de una buena opción, como sucede con los diagramas de flujo; sin embargo, tiene esa dificultad para hacer ver a los estudiantes que esas instrucciones –que no están en un lenguaje de programación de alto nivel– tienen sentido y podrían ejecutarse para resolver un problema.

La única forma de aprovechar al máximo la representación de los algoritmos a través del pseudocódigo era contar con el apoyo de la tecnología y lograr que el pseudocódigo no se quedara en el papel, sino que, por el contrario, se pudiera editar en el computador, ejecutar las instrucciones generadas, hacer pruebas y contar con los acuerdos de uso de la representación de manera tácita. La respuesta es una herramienta denominada Psicoder (Vásquez & Páez, 2008) que además de cumplir con los requisitos anteriormente planteados posee un componente visual que permite al estudiante definir los datos que va a necesitar en la solución a un problema, así como las acciones macro a desarrollar. Actualmente, en todas las actividades de aprendizaje significativo hechas en clase, el estudiante puede plantear la solución gráfica de un problema como en un diagrama de flujo; la herramienta permite realizar procesos de compilación, ejecución de instrucciones y puede, incluso, generar código en lenguaje de programación de alto nivel.

Sobre tareas y procesos. En cuanto a las actividades más cercanas con la elaboración de procesos, se han establecido metáforas, casos o enunciados que involucran las tareas, artefactos y conceptos clave que se pretenden enseñar. Como ejemplo de contextualización del desarrollo de actividades lúdicas y de apoyo didáctico de los temas tratados se mencionan, de manera general, un par de metáforas o casos de una colección mayor que se seguirán formalizando aún más, pensando en la segunda fase de SWT.

"*El equipo o empresa de programación*" es una actividad para trabajar el tema de funciones. Uno de los mayores retos de programar con el uso de funciones es la integración de varias de ellas para construir una solución completa. Con el objetivo de que los estudiantes experimenten tal situación se realiza un taller donde un problema de complejidad media es dividido en piezas más pequeñas usando la estrategia de "divide y vencerás". Las diferentes piezas del problema son asignadas a grupos de estudiantes de una clase para que se desarrollen las funciones que brinden la solución correspondiente a cada pieza. La parte más importante del taller consiste en que los grupos se integran en equipos más grandes para construir la solución final a partir de las funciones desarrolladas por los grupos iniciales.

"*Análisis de caso*": el objetivo es enfrentar al estudiante a problemas de mediana complejidad, aún antes de conocer los conceptos de la programación. Se desea que identifique los elementos clave del análisis y efectúe un dise-

ño básico de la solución. Como apoyo se ha desarrollado a manera de guía una plantilla que los estudiantes deben seguir y diligenciar, trabajando en grupos de dos personas para fomentar el trabajo de programación en parejas. La actividad, como varias otras, se ha desarrollado por varios semestres.

Resultados

El primer resultado general es poder contar con un espacio para compartir experiencias docentes en el Departamento de Ingeniería de Sistemas en la Pontificia Universidad Javeriana, espacio en el cual los propios estudiantes han participado y contribuido, entre otras cosas, en desarrollar aplicaciones de software para apoyar el proceso de enseñanza aprendizaje en el área de programación.

El segundo resultado es dar inicio a la formalización de iniciativas y propuestas alrededor del uso de representaciones visuales y metáforas, así como de actividades de aprendizaje significativo para la enseñanza de los conceptos de programación y ejecución de algoritmos.

Dentro de la organización de las actividades de aprendizaje significativo se realizaron pruebas de usabilidad a todas las herramientas mencionadas anteriormente acordes con el número de estudiantes de los cursos en los cuales se podían utilizar, particularmente en un curso de primer semestre de programación denominado “Pensamiento Algorítmico”. El curso cuenta cada semestre con cerca de 20 secciones y 350 estudiantes aproximadamente. De la misma manera, se realizaron algunas encuestas asociadas con los aspectos didácticos cuyos resultados han sido más que gratificantes.

Como ejemplo, en el proyecto TAVA se realizaron pruebas a 52 estudiantes pertenecientes a los cursos de Pensamiento Algorítmico y de Programación de Computadores (curso de segundo semestre de programación) y a ocho estudiantes de Ingeniería de Sistemas de noveno y décimo semestre. Como resultado, los estudiantes consideraron en un 85 % que era una ayuda efectiva para aprender los conceptos, y un 92 % opinó que las representaciones visuales eran claras y lograban identificar, gracias a ellas, los conceptos básicos del tema. Los detalles de los resultados de las pruebas de usabilidad y las encuestas se pueden encontrar en las referencias de cada uno de los trabajos mencionados.

Con el uso de la aplicación de software denominada Psicoder los estudiantes han logrado pensar más en la solución de los problemas que se les plantean en clase que en los detalles de la codificación de algoritmos al usar la herramienta que es completamente visual; adicionalmente, pueden mostrar una solución algorítmica cercana a un lenguaje de programación de alto nivel. Se destaca una mayor motivación por parte de los estudiantes al momento de afrontar la resolución de problemas usando el computador, ya que desde las primeras clases del semestre comienza a ser utilizada la aplicación y los estudiantes pueden ver sus resultados de manera inmediata. Una desventaja que se encuentra es que los estudiantes cuando ya han avanzado en el curso prefieren seguir con la metáfora inicial de las gráficas y la herramienta de Psicoder a tener que utilizar una herramienta de programación de alto nivel.

En términos generales, Psicoder ha tenido un gran éxito y una favorable aceptación por parte de los estudiantes; de hecho, es utilizada libremente en varias universidades colombianas y del exterior. La aplicación empezó a ser trabajada en la asignatura de Pensamiento Algorítmico desde el año 2009 por los estudiantes de la Facultad de Ingeniería de la Pontifica Universidad Javeriana; actualmente la población que se cubre semestralmente con la ayuda didáctica de la herramienta es la misma que se mencionó anteriormente (350 estudiantes en 20 sesiones aproximadamente).

En cuanto al uso de metáforas y casos asociados con procesos y tareas, las actividades piloto de aprendizaje significativo realizadas en los cursos de Pensamiento Algorítmico se han evaluado a través de encuestas durante tres semestres para recoger la percepción de los estudiantes con respecto al curso. Los resultados de la experiencia se describen a continuación.

En los talleres sobre el análisis de casos y problemas no triviales los estudiantes manifiestan

que las actividades les permiten abordar la solución de problemas desde una perspectiva más analítica, haciéndoles entender que el énfasis está en entender lo que se hace, en el análisis y el diseño, y no en la sintaxis de la herramienta usada para expresar la solución. Encuentran que es aplicable a otras asignaturas y a la vida real. Consideran importante trabajar con problemas que no sean tan sencillos desde el comienzo. Los casos proveen una plataforma de inicio adecuada para entender la programación y su contexto de trabajo. Por otra parte, han encontrado de gran utilidad las labores en parejas para aclarar dudas y poder discutir puntos de vista. Las actividades se han extendido para cubrir todos los grupos de la asignatura Pensamiento Algorítmico desde el segundo semestre de 2011.

Las actividades asociadas con la metáfora del equipo de programación se han venido desarrollando en grupos seleccionados de la asignatura de Pensamiento Algorítmico, desde el primer semestre del año 2007 con profesores de planta. Lo más destacable es el entendimiento por parte de los estudiantes del concepto de división en sub-problemas y la identificación de los elementos clave que afectan el trabajo a partir de esta aproximación; los estudiantes identifican que el tema de funciones es uno de los más complicados del curso. Desde el primer semestre de 2009, ampliado a algunas otras secciones del curso, los estudiantes manifiestan que la actividad les permite entender los problemas que se presentan al integrar funciones y trabajar su solución de manera conjunta con otros compañeros. Se ven con claridad el paso de parámetros, el uso de variables locales y el mecanismo de retorno. Los estudiantes identifican que participar en talleres prácticos que simulen desarrollos reales en equipos de integración de software es clave y necesario.

Otro resultado es empezar a lograr una integración de varias iniciativas de profesores y estudiantes a partir de puntos comunes en cuanto a la formalización de ciertas representaciones de los conceptos clave de programación y ejecución de algoritmos. Finalmente, la formalización de las representaciones, herramientas y actividades permitirá concretar una base importante y una visión diferente, centrada en tales representaciones, para el desarrollo de alternativas lúdicas de apoyo didáctico y aprendizaje activo en los temas de programación.

Con el propósito de formalizar otras aproximaciones y propuestas de actividades significativas de aprendizaje, como las mencionadas en este documento, se iniciará una segunda etapa centrada en analizar otras alternativas, diseñar otras propuestas y ampliar la evaluación de su uso. De igual manera, se propone la organización en un sistema de información para el mantenimiento de los datos, las actividades propiamente dichas, el material asociado y los resultados de su aplicación en el tiempo con base en un formato que, en principio, contempla la siguiente información importante, la cual se utilizó de manera no exhaustiva en la actual fase de exploración.

- Ficha pedagógica de la actividad (incluye objetivos, participantes y descripción general)
- Instrumentos a utilizar (herramientas de software o instrumentos físicos)
- Preparación de la actividad
- Guía de ejecución
- Actividades de reflexión y cierre (evaluación)

El sistema de información permitirá, posteriormente, la generación de un banco de actividades que puedan ser publicadas para el uso de los profesores y estudiantes. El objetivo final del banco es poner a disposición todo un conjunto de apoyos didácticos que contribuyan a mejorar el proceso de enseñanza aprendizaje de la programación.

Conclusiones

Los nuevos escenarios de la educación en ingeniería y los avances de la tecnología exigen el desarrollo de nuevas propuestas didácticas que contribuyan al proceso de enseñanza aprendizaje de los conceptos de programación y ejecución de algoritmos. Como consecuencia, el uso de representaciones visuales, metáforas, aplicaciones de software y actividades de aprendizaje significativo es clave.

El uso de recursos basados en aspectos visuales y táctiles permite llegar de manera po-

sitiva a un mayor número de estudiantes que encuentran tales estilos de aprendizaje acordes con sus preferencias de pensamiento. Por ende, el uso de las representaciones visuales, metáforas e instrumentos físicos o de software permite apoyar la generación de las abstracciones necesarias que contribuyen a comprender los conceptos intangibles propios de la programación y ejecución de algoritmos.

De igual manera, la formalización de las experiencias significativas de aprendizaje apoyadas en instrumentos didácticos de soporte permite contar con mecanismos más apropiados para los escenarios exigidos por el aprendizaje activo y los nuevos espacios de trabajo para los estudiantes que demandan las nuevas iniciativas y modelos de enseñanza, como es el caso de la propuesta CDIO por citar una de ellas.

La formalización de representaciones y de actividades de aprendizaje significativo facilitará el diseño y ejecución de las mismas, contribuirá en la concepción de mejores criterios para su utilización y pertinencia en ciertos contextos y evitará, en gran medida, la creación de aplicaciones tecnológicas aisladas y muchas veces incoherentes con los conceptos que se desean enseñar o aprender.

Es vital señalar algunas apreciaciones sobre el uso de las representaciones visuales, las metáforas y las didácticas que deben tenerse en cuenta en la continuación del presente trabajo o para emprender otros proyectos similares:

- Con base en la teoría sobre el pensamiento integral, los estilos de aprendizaje y los resultados del trabajo de Barros, Sánchez, & Rojas (2007), la cuestión no es que exista una única opción de representación o que una didáctica sea buena o mala en sí misma, sino que la didáctica es efectiva o no, dependiendo del tipo de pensamiento y el estilo de aprendizaje del estudiante.
- Las propuestas presentadas o la generación de algunas otras requieren de una estrategia integrada en su diseño, acorde con los niveles de abstracción involucrados para que sean coherentes durante su utilización. Acorde con el punto anterior, su uso también debe ser integrado con relación a utilizar o adecuar dos o más didácticas complementarias en una actividad significativa de aprendizaje. Las didácticas finalmente seleccionadas deben estar enfocadas por el profesor de acuerdo con las características propias del tema, del curso y sus estudiantes.
- En conjunto, todas las potenciales representaciones definidas, las herramientas disponibles, los casos y las actividades significativas de aprendizaje propuestas son, fundamentalmente, una ayuda que puede ser implementada de diferentes maneras pero ello no quita la responsabilidad del profesor, quien finalmente es el encargado de formalizar el concepto en cuestión.
- El profesor debe estar en la capacidad de usar la representación o metáfora solamente hasta un momento indicado. Es responsabilidad del profesor saber cuándo abandonar la representación o la metáfora y asegurarse de precisar el formalismo de cada uno de los conceptos que quiere enseñar.

Finalmente, como se indicó en la primera sección, el trabajo futuro tiene que ver con seguir formalizando las propuestas de representación, el desarrollo de herramientas y las actividades de aprendizaje activo relacionadas, pensando en la generación de un banco de herramientas didácticas coherentes con los temas propuestos, así como la evaluación más formal y detallada de la efectividad y el impacto de su utilización. Se espera poder publicar de manera libre otros instrumentos, las guías relacionadas y las experiencias de aprendizaje significativo que se sigan emprendiendo.

Referencias bibliográficas

- Alice Teaches Object Oriented Computer Programming To Kids. (2011). [Software de computación]. Recuperado de http://www.makeuseof.com/tag/alice-teaches-object-oriented-computer-programming-to-kids/
- Allinson, C. & Hayes, J. (1996). The cognitive Style Index: A measure of intuition analysis for organizational research. *Journal of Management Studies*. 33 (1), 119–135.
- Anderson, I. W., & Krathwohl, D. (2001). *A Taxonomy for Learning, Teaching and Assessing:*

a Revision of Bloom's Taxonomy of Educational Objectives. New York: Longman.

- Anderson, L. (2001). *A Taxonomy for Learning, Teaching, and assessing.* Addison Wesley Logman.
- Barón, l. & Ronderos, M. (2009). *TAVA: Herramienta didáctica para apoyar el proceso de enseñanza-aprendizaje en los conceptos de variables, tipos de datos y apuntadores* (Trabajo de grado). Bogotá D.C.: Pontificia Universidad Javeriana.
- Barros, R. Sánchez, l. M., & RojaS, J. A. (2007). Diseño de instrumentos didácticos para aprendizaje activo basado en teoría de colores. *XXVII Reunión Nacional de ACOFI, Cartagena de Indias*.
- Carlisle, m., Wilson, F., Humphies, J. & Hadfield, S. (2005). RAPTOR: A Visual Programming Environment for Teaching Algorithmic Problem Solving. USA: Department of Computer Science, United States Air Force Academia.
- CDIO Initiative: Conceiving, D. I. (2011). *CDIO Initiative: Conceiving, Designing, Implementing, Operating»,* [Software de computación]. Recuperado de http://www.cdio.org .
- Cooper, S., Dann, W. & Pausch, R. (2000). Alice: a 3-D tool for introductory programming concept. *Proceedings of the fifth annual CCSC northeastern conference on The journal of computing in small colleges,* 107–116.
- De Pauw, W., Lorenz, D., Vlissides, J., & Wegman, M. (s. f.). E*xecution Patterns in Object-Oriented Visualization*. IBM T.J. New York, USA: Watson Research Center.
- Herrmann, N. (1996). *The whole brain business book.* New York: McGraw-Hill.
- Hood, C. & Hood, D. (2005). Teaching Programming and Language Concepts using Lego. *Illinois Institute of Technology, ACM*.
- Hosking, J. (1996). *Visualization of Object Oriented Program Execution*. *N*ew Zealand: Department of Computer Science University of Auckland.
- Klassner, F. & Anderson, S. (2003). LEGO MindStorms: not just for K-12 anymore. *IEEE Robotics & Automation Magazine, 10 (2)*, 12-18.
- Lego Mindstorms. (2011). *LEGO.com MINDSTORMS : Home.* [Software de computación]. Recuperado de http://mindstorms.lego.com/en-us/Default.aspx.
- Masterson, T. & Masterson, M. (2001). *Sivil: A true visual programming language for students*. Buffalo, NY: Computer Science Department, Canisius Collage.
- Montenegro, O. (2009). *VMCreative: herramienta didáctica de apoyo al proceso de enseñanza y aprendizaje de los conceptos básicos de vectores y matrices* (Trabajo de Grado). Bogotá, D. C.: Pontificia Universidad Javeriana.
- Pi, R. (2012). *Raspberry Pi.* [Software de computación]. Recuperado de http://www.raspberrypi.org/.
- Powers, et al. (2011). *Robotica en la escuela.* Recuperado de http://www.roboticaenlaescuela.es/blog/wpcontent/uploads/2010/05/10.1.1.107.466.pdf.
- *PSeInt. (2011). PIPEH PSeudo Intérprete.* [Software de computación]. Recuperado de http://pseint.sourceforge.net/.
- Rena, P. & Pratt, K. (2003). *The Virtual Student.* San Francisco: Jossey Bass higher and adult education series.
- Resnick et al. (2009). Scratch: programming for all. *Commun. ACM,* 52(11), 60–67.
- Robolab. (2011). *ROBOLAB @ TUFTS - HOME.* [Software de computación]. Recuperado de http://www.ceeo.tufts.edu/robolabatceeo/.
- Rubio, C. (2008). *POOInteractiva: Prototipo de una herramienta didáctica para apoyar el proceso de enseñanza-aprendizaje de conceptos fundamentales de programación orientada a objetos (*Trabajo de Grado). Bogotá, D. C.: Pontificia Universidad Javeriana.
- Scratch. (2011). *«Scratch | Home | imagine, program, share»,* [Software de computación]. Recuperado de http://scratch.mit.edu/.
- Taylor, D. A. (1991). *Object Oriented Technology: A Manager Guide.* Canada: Adisson-Wesley.
- Vásquez, J. & Páez, L. (2008). *Psicoder: Software Educativo para facilitar el proceso de enseñanza-aprendizaje en un curso básico de programación (*Trabajo de Grado). Bogotá, D. C.: Pontificia Universidad Javeriana.
- VILLALOBOS et al. (2006). *Proyecto Cupi2. Bogotá: D.C.: Universidad de los Andes.* Recuperado de Internet: http://cupi2.uniandes.edu.co.