Herramientas para apoyar el descubrimiento de vocaciones en Ciencias de la Computación

Sonia V. Rueda Andrea Cohen Telma Delladio
Sebastian Gottifredi Luciano H. Tamargo

Departamento de Ciencias e Ingeniería de la Computación - Universidad Nacional del Sur
Avenida Alem 1253, (B8000BCP), Bahía Blanca, Argentina
Tel: (0291) 459-5135 / Fax: (0291) 459-5136
{svr, ac, td, sg, lt}@cs.uns.edu.ar

Resumen Este trabajo describe las herramientas evaluadas por el Departamento de Ciencias e Ingeniería de la Computación de la Universidad Nacional del Sur, para mejorar la difusión de sus carreras y despertar el interés de los estudiantes de nivel medio en las Ciencias de la Computación. Presenta además algunas de las alternativas de aplicación diseñadas con estos objetivos y delinea el trabajo futuro en base a estas propuestas.

Palabras claves: descubrimiento de vocaciones tempranas, difusión oferta académica, construcción de mundos virtuales, entornos visuales de programación, desarrollo del pensamiento computacional.

1 Introducción

Desde hace más de 10 años el Departamento de Ciencias e Ingeniería de la Computación (DCIC) de la Universidad Nacional del Sur (UNS) participa intensamente en actividades de articulación con el nivel medio. Uno de los objetivos es brindar información a los alumnos del último año del nivel secundario y a los de primer año del nivel superior, para favorecer el tránsito entre el ámbito de la escuela y el universitario, ayudando al proceso de adaptación. Otro de los objetivos es mejorar la difusión de las carreras ofrecidas por esta unidad académica y despertar el interés de los estudiantes de nivel medio en temas vinculados a las Ciencias de la Computación.

A partir de este segundo objetivo se han diseñado e implementado varios proyectos. En ellos participan docentes del nivel medio y superior, estudiantes de las escuelas y tutores del DCIC. En el marco de estos proyectos, el DCIC brinda charlas informativas en establecimientos educativos, organiza conferencias a cargo de profesionales de la industria de software para difundir la oferta laboral, coordina visitas de estudiantes a Empresas de Software y ofrece un Taller de Entrenamiento para Competencias de Programación y Cursos de Resolución de Problemas. Esta última actividad está orientada específicamente a alumnos del último año de nivel medio. El DCIC, como todas las unidades académicas de la UNS, participa además en la Muestra Anual de carreras, evento que brinda la posibilidad de difundir las carreras e interactuar con estudiantes de la zona.

Algunos de los proyectos se enmarcan en convenios específicos celebrados entre el DCIC y Escuelas de Nivel Secundario. Se trata de establecimientos que incluyen una modalidad orientada a Informática, y la intención es estimular a sus alumnos para que elijan carreras de esta disciplina. Estas escuelas incluyen en sus planes de estudio materias de programación. La propuesta es incorporar nuevas herramientas que permitan generar actividades motivadoras para los alumnos y significativas para el aprendizaje.

El DCIC organiza también actividades con escuelas que no ofrecen una modalidad vinculada a la Informática. En la mayoría de los casos los planes de estudios en estas escuelas incluyen contenidos de Computación, enfocados exclusivamente a capacitar a los estudiantes en el uso de utilitarios. Los jóvenes no realizan ninguna práctica en programación, por lo general no conocen las incumbencias de las carreras de Ciencias de la Computación y con frecuencia sólo tienen información referida a algunas de las oportunidades laborales que ofrecen las mismas. La intención del DCIC, en tal caso, es brindar información acerca de la actividad profesional de los graduados en Ciencias de la Computación y generar espacios para que los alumnos puedan tener un contacto con la programación.

Cualquiera sea la modalidad, la inclusión de actividades de programación permite desarrollar el *pensamiento computacional*. Esta capacidad favorece la resolución de problemas, no sólo en el marco de las Ciencias de la Computación, sino en cualquier disciplina vinculada a la ciencia y la tecnología.

Existen numerosas herramientas y diferentes estrategias que pueden utilizarse tanto para introducir el pensamiento computacional en talleres o cursos cortos, como para organizar asignaturas completas. En general no son alternativas excluyentes sino que, por el contrario, se complementan y pueden aplicarse en secuencia, comenzando con las más simples y continuando con otras más versátiles, pero que demandan mayor capacitación.

La sección que sigue caracteriza el pensamiento computacional, luego describe las herramientas evaluadas por el DCIC para contribuir en el desarrollo de esta capacidad y mejorar la difusión de sus carreras. Se presentan también algunas de las actividades diseñadas con este objetivo. El trabajo concluye con algunos lineamientos para el trabajo futuro.

2. El pensamiento computacional

El pensamiento computacional es un concepto presentado en [5] por Jeanette M. Wing. Esta noción busca caracterizar la solución de problemas de cualquier índole utilizando herramientas de las Ciencias de la Computación. Al enfrentarse a un problema uno puede preguntarse, entre otras cosas: ¿Cuán difícil es resolver el problema? ¿Cuál es la mejor forma de resolverlo? Las Ciencias de la Computación brindan sólidos fundamentos teóricos para responder a estas preguntas en forma precisa. Por ejemplo, en relación a la dificultad del problema, debemos considerar el poder expresivo de la arquitectura subyacente, *i.e.,* el dispositivo computacional en el que se ejecutará la solución encontrada. En este sentido, deberemos considerar el

repertorio de instrucciones de la máquina en cuestión, los recursos que ofrece (memoria, disco, etc.) y su ambiente operativo.

En función de lo expuesto, el pensamiento computacional consiste en pensar recursivamente, pensar paralelamente, interpretar el código como datos y los datos como código, establecer chequeos, entender las virtudes y peligros del uso de cada recurso, comprender el costo y el poder de cada mecanismo. Más aún, el pensamiento computacional implica realizar abstracciones y descomposiciones cuando se trata de resolver una tarea compleja o diseñar un sistema grande, tener la confianza de que es posible usar y modificar un sistema complejo sin la necesidad de entender cada detalle. Asimismo, el pensamiento computacional conlleva a no solamente juzgar un diseño o solución por su correctitud y eficiencia, sino también por su estética, simpleza de diseño y elegancia; es utilizar razonamiento heurístico, aprendizaje y planificación para encontrar soluciones en escenarios con incertidumbre.

Todos los elementos arriba mencionados pueden ser empleados en la resolución de problemas de cualquier índole, y las Ciencias de la Computación proveen métodos y guías para utilizarlos adecuadamente. Para ilustrar esto, consideremos los siguientes ejemplos, correspondientes a situaciones de la vida cotidiana. Cuando una niña tiene que ir a la escuela por la mañana, guarda en su mochila todas las cosas que necesitará durante el día; esto es cacheo y *prefetching*. De manera similar, cuando un niño pierde las llaves de su casa, uno le sugiere que vuelva mentalmente sobre sus pasos hasta recordar el punto en que las vio por última vez; esto es *backtracking*. Cuando un joven decide en qué cola del supermercado ubicarse, está realizando un análisis de performance sobre un sistema multi-servidor.

En la actualidad es posible evidenciar la influencia del pensamiento computacional en diversas disciplinas. A modo de ejemplo, en el análisis estadístico, la técnica de *machine learning* resulta una herramienta esencial: el *aprendizaje estadístico* (en inglés, *statistical learning*) es utilizado en problemas de gran escala en términos del tamaño y dimensión de los datos empleados. Esto era inimaginable hace algunos años. Otro ejemplo es la influencia del pensamiento computacional en la biología, no sólo reflejada en la exploración de grandes secuencias de datos en búsqueda de patrones, sino también en la utilización de algoritmos y estructuras de datos para representar la organización y comportamiento de diversos agentes biológicos como las proteínas. Así, el pensamiento computacional ha influenciado la forma de pensar de los biólogos. De manera similar, las teorías de juegos computacionales influyen en cómo piensan los economistas, la nano-computación en cómo piensan los químicos, y la computación cuántica en cómo piensan los físicos.

Así, el pensamiento computacional puede verse como una capacidad fundamental que todo individuo, además de los estudiantes y graduados en Ciencias de la Computación, debería desarrollar. A partir de esta premisa es muy importante que los estudiantes de todos los niveles desarrollen el pensamiento computacional [10]. Más aún, diversos estudios muestran que la influencia del pensamiento computacional en cursos con estudiantes de secundaria y de nivel superior ha incrementado sus capacidades de resolución de problemas [6,7], así como también su interés por las Ciencias de la Computación [9].

Una de las aproximaciones más exitosas para acercar a los estudiantes de nivel medio al pensamiento computacional es el diseño y construcción de mundos virtuales o juegos interactivos [8]. Mientras que la simulación del mundo real debe respetar las

normas de la Naturaleza y los principios de la Física, en la creación de mundos virtuales todo es posible. Los jóvenes pueden crear escenarios y modelar el comportamiento de sus personajes con el único límite de su propia imaginación [14].

En la siguiente sección se describe una serie de herramientas que brindan un ambiente adecuado para el desarrollo de este tipo de aplicaciones, ayudando a su vez a los alumnos a desarrollar competencias en el pensamiento computacional.

3. Herramientas

El análisis de las herramientas que se describen en esta sección se enfocó a la generación de actividades orientadas al descubrimiento de vocaciones tempranas en Ciencias de Computación. Todas ellas permiten desarrollar el pensamiento computacional de los jóvenes, con la convicción de que esta capacidad favorece la resolución de problemas, el diseño de proyectos y la comunicación de ideas en distintas áreas. Así, la incorporación de estas herramientas en el aula brinda un aporte significativo, aún para aquellos que no van a continuar sus estudios en carreras relacionadas con la Informática.

Todas las herramientas que se describen en este trabajo son gratuitas y permiten la enseñanza de conceptos de programación y el diseño de actividades de exploración y experimentación para la resolución de problemas. Pueden ser utilizadas a partir de los 12 años, algunas inclusive un poco antes, y brindan repositorios para compartir las producciones en línea favoreciendo el trabajo colaborativo. Aunque todas pueden ser utilizadas con poco entrenamiento, algunas tienen una interface más simple que otras. Cada una tiene diferentes requerimientos de sistemas.

3.1 Scratch

Scratch es un entorno de programación que permite crear animaciones para contar historias o producir videojuegos [1]. Cada animación se construye definiendo el comportamiento de uno o más objetos que actúan sobre un escenario, a partir de un repertorio de instrucciones basado en Logo. En la interface cada instrucción está representada por un *bloque* y el repertorio completo está clasificado en: movimiento, apariencia, control, datos, operadores, sensores, eventos, lápiz y sonido. Los bloques se van encastrando en el orden que van a ejecutarse.

Aunque la construcción de cada programa se realiza a partir de un repertorio de instrucciones provisto por un lenguaje, las primeras etapas del aprendizaje se simplifican porque no es necesario aprender la sintaxis de la notación y el resultado de la ejecución es inmediato.

En Scratch programar consiste en encastrar bloques. Cada secuencia de bloques encastrados puede recibir un nombre e incorporarse al conjunto de bloques del repertorio. Así, cada programa en Scratch puede ser importado como un comando y combinado con otros para crear nuevas animaciones. Es justamente esta característica la que le dio el nombre a la aplicación, que se deriva de la técnica de "scratching" usada por los Disc Jockeys para pasar música y reutilizar piezas.

El proyecto que le dio origen se inició en 2003 en el Laboratorio de Medios del MIT con el lema *Imagina, Crea y Comparte*. Ha recibido el apoyo de National Science Foundation, Fundación Intel, Microsoft, Fundación MacArthur, Fundación LEGO, Fundación Code-to-Learn, Google, Dell, entre otros. El entorno puede instalarse para Windows, Mac y Linux.

3.2 Blockly

Blockly es un entorno de programación visual diseñado con una fuerte influencia de Scratch, pero con algunas características distintivas. Como en Scratch, no es necesario aprender un lenguaje, las instrucciones se seleccionan de un repertorio de bloques, pero no se instala, sino que se ejecuta sobre un navegador web. Cada programa puede exportarse a varios lenguajes, entre ellos JavaScript y Python. Es de código abierto y muy escalable. Se utiliza para distintos tipos de proyectos, además de usarse como herramienta para enseñar a programar, en particular en aplicaciones de robótica.

Bajo el lema "Teach the Hour of Code in your classroom" Code.org ofrece un sitio web en el que difunde varios tutoriales con desafíos para resolver usando Blockly, una base de datos con información referida a dónde aprender a programar y un video con testimonios de personajes relevantes como Bill Gates o Mark Zuckerberg [2].

El primer tutorial está diseñado para ser resuelto en una hora, por una persona sin conocimientos de programación. Cada uno de los que siguen, previstos también para realizar en una hora, asumen que se completaron las actividades anteriores. Aunque cada problema se enuncia verbalmente, la interface permite visualizar el desafío a través de una imagen.

Algunas soluciones requieren indicarle a un personaje el camino que debe recorrer y las acciones que debe realizar para alcanzar un objetivo. Los bloques de movimiento permiten avanzar y girar, los de acción le indican al personaje que debe llenar un pozo o quitar una pila de tierra y los bloques de control permiten repetir acciones o ejecutarlas en forma condicional. Otros problemas plantean dibujar figuras geométricas a partir de herramientas para avanzar, girar, repetir, ejecutar en forma condicional, establecer el color, etc. En todos los casos es posible visualizar el código que se genera en JavaScript a partir de los bloques encastrados.

Code.org es una organización sin fines de lucro que lanzó en 2013 una campaña para estimular la enseñanza de programación en las escuelas primarias y secundarias. La expectativa es desarrollar el pensamiento computacional de los estudiantes y aumentar el número de interesados en estudiar Ciencias de Computación. La premisa es que, con una proyección basada en el número de alumnos actual, entre 2013 y 2020, sólo en Estados Unidos quedarán sin cubrir cientos de miles de puestos de trabajo en esta disciplina.

3.3 Alice

Alice es un entorno de programación integrado que ofrece una interface amigable para crear animaciones 3D [13]. Permite construir simulaciones y videojuegos, arrastrando objetos a un escenario y modificando sus propiedades. Mientras se construye un mundo virtual, el entorno muestra las instrucciones generadas en un

lenguaje orientado a objetos permitiendo visualizar la relación entre el comportamiento de los objetos y el código.

Alice fue concebido para ser utilizado en el ámbito educativo y brinda recursos para que los docentes diseñen y adapten actividades para diferentes niveles y en diferentes asignaturas. El sitio oficial ofrece también documentos y videos de entrenamiento desde un primer nivel exploratorio que permite configurar escenarios e insertar objetos, hasta otros mucho más avanzados a través de los cuales es posible controlar eventos y programar juegos interactivos.

En Argentina la Fundación Sadosky promueve el uso de Alice en las escuelas de nivel medio en el marco del proyecto *Vocaciones en TIC*, orientado a despertar el interés de los adolescentes en la programación. El objetivo es introducir conceptos de programación orientada a objetos y contribuir al desarrollo del pensamiento computacional. Aunque la construcción de animaciones es más simple que usando un lenguaje de programación convencional, el entrenamiento de docentes y alumnos demanda un esfuerzo considerable. La fundación organiza seminarios para docentes con una carga horaria aproximada de 40 horas.

Alice puede ser instalado sobre distintas plataformas (Windows, Mac y Linux).

3.4 Pilas Engine

Pilas Engine [12] es un entorno de programación de videojuegos basado en Python. Esta herramienta busca brindar una interface simplificada de desarrollo para jóvenes que comienzan a programar, sin necesidad de contar con amplios conocimientos previos en este área. Está inspirada en motores modernos para el desarrollo profesional de videojuegos como Cocos2d o Pygame y herramientas para el aprendizaje como Logo. A diferencia de las propuestas de Scratch, Blockly o Alice, en Pilas Engine la programación se realiza mediante instrucciones Python, aunque el motor provee primitivas simplificadas que permiten al usuario abstraerse de los detalles de dicho lenguaje. De esta forma, la herramienta ofrece instrucciones para agregar actores, modificarlos, añadirles comportamiento o hacerlos interactivos. El entorno brinda además la posibilidad de visualizar inmediatamente el comportamiento generado por las instrucciones escritas.

Esta herramienta es libre y gratuita bajo la licencia LGPL, y puede ser utilizada en sistemas operativos GNU/Linux, Windows y Mac OS X. A diferencia de las otras alternativas estudiadas en esta sección, el software y la documentación de Pilas Engine se encuentran completamente en idioma español.

Pilas Engine forma parte de las herramientas promocionadas para el aprendizaje de programación en el marco de la iniciativa *program.ar* [11]. Tal iniciativa fue impulsada por del Estado de la Nación Argentina para fomentar el acercamiento de los jóvenes a las Ciencias de la Computación, así como también para concientizar al público en general de la importancia de esta disciplina en el mundo moderno.

3.5 Algodoo

Algodoo es un simulador 2D de física de mundo abierto en el cual es posible crear escenas de simulación que permiten experimentar con diferentes componentes y visualizar cómo estos siguen las leyes de la física. Una escena de simulación se

construye utilizando herramientas intuitivas que permiten dibujar componentes como círculos, polígonos, engranajes, cuerdas, fluidos, motores, ejes y punteros láser, entre otros. Algodoo permite al usuario interactuar fácilmente con estos componentes mediante herramientas clásicas de dibujado, así como también configurar sus propiedades físicas como velocidad, atracción, masa y refracción. Al ejecutar una escena de simulación se visualiza efectivamente cómo los componentes interactúan entre sí, regidos por las leyes de la física. Además, mientras se ejecuta la simulación, el usuario puede interactuar dinámicamente con los componentes involucrados, aplicando fuerzas, moviéndolos, o cortándolos. Esto brinda la posibilidad de diseñar juegos y desafíos que deben ser resueltos por el usuario que interactúa con la simulación.

De manera similar a las herramientas presentadas en las secciones anteriores, para utilizar Algodoo durante las primeras etapas de aprendizaje, el usuario no necesita poseer conocimientos formales acerca de la física subyacente. Esto se debe a que, como se puede observar en las primeras lecciones y tutoriales de la herramienta, es posible crear escenas de simulación sencillas guiándose sólo por nociones intuitivas o de sentido común. Aun así, la herramienta provee los medios para crear escenarios más complejos y brinda un lenguaje de scripting llamado *Thyme* que permite programar comportamiento especializado para los componentes de una escena.

Algodoo surgió en el año 2008 a partir de la Tesis de Maestría de Emil Ernerfeldt [3] y basa su motor físico subyacente en el modelo de resolución de restricciones lineal SPOOK [4]. Actualmente, Algodoo se encuentra bajo la dirección de *Algoryx Simulation AB*, una empresa que desarrolla motores de simulación para la industria. La herramienta se puede utilizar en Windows y Mac OS X, y posee soporte para tablets. Algodoo cuenta con una gran comunidad para compartir escenas de simulación creadas por los usuarios, contando en la actualidad con más de 50.000 de estas. Por otra parte, a diferencia de las herramientas anteriores, el foco de esta herramienta se encuentra en el aprendizaje de física. Sin embargo, dado su poder expresivo, es ampliamente utilizada para ayudar al desarrollo de competencias en matemática y resolución de problemas.

4. Alternativas de aplicación

Como se mencionó en la introducción, el análisis presentado en este trabajo está enfocado al diseño de actividades que contribuyan a desarrollar el pensamiento computacional en alumnos del nivel medio. Claramente, el desarrollo de esta capacidad será diferente en una modalidad orientada a la Informática, con respecto a otra modalidad que no incluye contenidos de Informática o los concentra exclusivamente en el uso de utilitarios.

Para este último caso, el DCIC está ofreciendo actualmente talleres para alumnos que brindan la oportunidad de programar durante una hora utilizando Blockly, en el marco de las actividades propuestas por code.org. Estos talleres se realizan en la escuela o en los laboratorios del DCIC, según resulte más adecuado. Aquellos que se sientan motivados por la propuesta, pueden continuar utilizando la herramienta de

manera autónoma o participar en el Taller de Entrenamiento para Competencias de Programación que ofrece el DCIC desde hace tres años.

Para las escuelas con modalidad de Informática, el DCIC está organizando Talleres de Capacitación Docente que permitan la actualización curricular de las asignaturas de Programación. Los talleres son dictados por docentes del DCIC y sus destinatarios son docentes del nivel medio que dictan asignaturas de programación. El impacto es en este caso mayor porque cada docente que participa propaga las nuevas herramientas entre sus alumnos.

Cada taller se estructura en dos etapas, la primera se enfoca a la presentación de una herramienta específica y actividades concretas para implementar en el aula. La segunda etapa demanda una participación más activa de los docentes del nivel medio porque la expectativa es socializar sus experiencias.

En este contexto, es posible optar por un taller de Pilas Engine o uno de Algodoo. A este último, también están invitados a participar docentes de Matemática y Física. Ambos entornos pueden introducirse de manera incremental. Docentes y alumnos pueden comenzar a utilizarlo partiendo de escenarios creados previamente y aprender a usar las primitivas modificando los objetos. A medida que se adquiere experiencia es posible plantear desafíos más ambiciosos.

Ambos parten de la base de que la complejidad de la programación está ligada al lenguaje y la simplicidad del ambiente. En las primeras etapas del aprendizaje de la programación, efectivamente la rigurosidad sintáctica y semántica del lenguaje suelen ser un obstáculo. La manipulación de objetos en un ambiente interactivo, facilita el proceso de construcción.

Sin embargo, a medida que se plantean desafíos más complejos, la incidencia de la sintaxis en la dificultad del problema es menor y el programar de manera interactiva, sin un diseño previo, estimula la creatividad pero también genera hábitos que pueden conducir a una metodología algo caótica.

Es importante notar que las estrategias de resolución que resultan adecuadas para construir un mundo virtual, probablemente con un objetivo flexible, pueden no serlo para resolver otro tipo de problemas, con requerimientos especificados con precisión. Cuando la propuesta didáctica está orientada no sólo a mostrar qué es programar, sino también qué es el desarrollo de software, es necesario considerar que no basta con estimular la creatividad; es también importante proponer criterios de calidad y una metodología que guíe el proceso de construcción considerando estos criterios. Estos aspectos requieren la incorporación de temas de ingeniería de software y forman parte del trabajo futuro que se describe a continuación.

Por el momento, no se han organizado actividades utilizando Alice. Se resolvió analizar la herramienta dado que es una de las elegidas por la Fundación Sadosky en su Programa *Vocaciones en TIC*. Sin embargo, se seleccionaron los demás recursos descriptos dada la simplicidad de los entornos, la posibilidad de diseñar actividades que permitan presentarlos en secuencia y en cada uno proponer desafíos incrementando gradualmente la complejidad.

5. Trabajo Futuro

Las experiencias realizadas hasta el momento han sido muy positivas y alentadoras. El trabajo con escuelas con modalidad Informática crece cada año y se enriquece con nuevas actividades. Sin embargo, es necesario avanzar aun más, estableciendo vínculos con instituciones de la zona de influencia en Bahía Blanca. Otro aspecto importante desde la gestión, es consolidar proyectos que permitan otorgar créditos a los docentes que asisten a los talleres. También es fundamental realizar un seguimiento de los alumnos que ingresan en la UNS, luego de haber participado en actividades de articulación y utilizado las herramientas descriptas en las secciones previas.

Por último, desde el punto de vista académico, como hemos mencionado antes, los entornos propuestos en este trabajo simplifican la interacción y son muy adecuados para enseñar a programar a través de la construcción de mundos virtuales. Es importante avanzar en el análisis del proceso de aprendizaje, posterior al primer contacto, para evaluar el desarrollo de la capacidad de abstracción y el razonamiento lógico. Por otra parte, dado que las Ciencias de la Computación no se circunscriben a la programación, es necesario considerar la incorporación de otros contenidos que favorezcan el pensamiento computacional. En particular, incluir temas de ingeniería de software a medida que los problemas planteados aumenten de escala.

Con respecto a las demás modalidades, que claramente incluyen a la mayoría de los alumnos de nivel medio, la propuesta de ofrecer una hora de contacto con la programación ha sido hasta el momento sumamente satisfactoria. Los jóvenes se entusiasman al ir superando los desafíos, que están organizados con una dificultad creciente, de modo que progresivamente requieren mayor esfuerzo y concentración. Sin embargo, se trata sólo de un primer contacto y, aunque permite mostrar qué habilidades demanda la programación, es difícil descubrir una vocación luego de una actividad de una hora. La expectativa es que esta hora motive a los alumnos para realizar alguna otra actividad ofrecida por el DCIC para reforzar el acercamiento a las Ciencias de la Computación.

La concreción de cada proyecto demanda un esfuerzo considerable de gestión, tanto por parte de las escuelas como del DCIC. Cualquiera sea la modalidad, una tarea importante es evaluar el impacto de cada una de estas iniciativas, a partir de metas concretas y para hacerlo es necesario que se logre una continuidad en el tiempo. Probablemente, lograr esta continuidad sea el mayor desafío.

Referencias

[1] Scratch: Programming for All Resnick M., Maloney J, Monroy-Hernandez A., Rusk N., Eastmond E., Brennan K, Millner A., Rosenbaum E., Silver J., Silverman B., Kafai Y. Communications of the ACM (Nov 2009), Vol. 52, No. 11

[2] http://www.code.org

[3] Emil Ernerfeldt Master Thesis 2008, Dept. Computing Science, Umeå University, Sweden

[4] Claude Lacoursière's PhD Thesis 2007, Dept. Computing Science, Umeå University, Sweden

[5] Computational Thinking Wing J. M. Communications of the ACM (Mar 2006) Vol. 49, No. 3

[6] Computational thinking for youth in practice. Lee I., Martin F., Denner J., Coulter B., Allan W., Erickson J., Malyn-Smith J., Werner L. ACM Inroads (Mar 2011) Vol. 2 Issue 1, pp 32 -37.

[7] Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? Barr V., Stephenson C. ACM Inroads (Mar 2011) Vol. 2 Issue 1, pp 48 -54.

[8] Scalable game design and the development of a checklist for getting computational thinking into public schools. Repenning A., Webb D., Ioannidou A. SIGCSE '10 Proceedings of the 41st ACM technical symposium on Computer science education, pp. 265- 269.

[9] A multidisciplinary approach towards computational thinking for science majors. Hambrusch S., Hoffmann C., Korb J.T., Haugan M., Hosking A.L. SIGCSE '09 Proceedings of the 40th ACM technical symposium on Computer science education, pp. 183-187

[10] Computational thinking and thinking about computing. Wing J. M. Phil. Trans. R. Soc. A 28 October 2008 vol. 366 no. 1881, pp. 3717-3725

[11] http://program.ar

[12] http://pilas-engine.com.ar

[13] Alice, Greenfoot, and Scratch -- A Discussion. Utting I., Cooper S. Kölling M., Maloney J., Resnick M. ACM Transactions on Computing Education (Nov 2010). Vol. 10 Issue 4, Article No. 17.

[14] "El rol de la simulación simbólica en la Teoría Constructivista" Sonia Rueda Ateneo de Profesores, IV Congreso Argentina de Ciencias de la Computación. Neuquén-Argentina. (Octubre 1998)