Una revisión de métodos pedagógicos innovadores para la enseñanza de la programación *

Mercedes Gómez Albarrán
Dpto. de Sistemas Informáticos y Programación
Universidad Complutense de Madrid
Ciudad Universitaria s/n, 28040 Madrid
e-mail: albarran@sip.ucm.es

Resumen
Este trabajo realiza una profunda revisión de las diferentes tendencias actuales en lo que se refiere a métodos pedagógicos de soporte al aprendizaje de la programación en general.

Se incluyen referencias a numerosos trabajos que ejemplifican cada una de las tendencias así como direcciones en el World Wide Web en las que conseguir un buen número de herramientas.

1. Introducción

Es notorio que las nuevas tecnologías de la información y la comunicación han provocado cambios en la manera en la que los profesores impartimos nuestras clases y también en la forma en la que nuestros alumnos estudian.

El World Wide Web es utilizado actualmente por los educadores como una plataforma que hace viable una fórmula complementaria a la docencia presencial. Dicha fórmula incluye desde la disponibilidad de una ingente cantidad de material educativo (apuntes, transparencias, resoluciones de ejercicios, etc.) hasta el desarrollo de herramientas educativas atractivas y útiles que pueden estar al alcance de todos.

En este trabajo se presentan diferentes tendencias actuales en lo que se refiere a métodos de soporte al aprendizaje de la programación en general. Numerosas muestras de dichos métodos pueden ser encontradas diseminadas a lo largo del World Wide Web.

Antes de revisar dichos métodos, no queremos pasar por alto un cambio en las tendencias de los enfoques llevados a la práctica.

Algunos trabajos iniciales de soporte al aprendizaje de la programación se centraron en el desarrollo de tutores inteligentes. Un ejemplo es PROUST [21], un tutor basado en conocimiento para la enseñanza del lenguaje de programación Pascal que identifica errores no sintácticos. PROUST forma parte de un sistema más amplio que asigna tareas a los estudiantes, analiza el trabajo que éstos hacen y, posteriormente, realiza sugerencias apropiadas. Evidentemente, para llevar a cabo su tarea PROUST debe comprender lo que el alumno está haciendo, para lo cual cuenta con una gran cantidad de conocimiento tanto acerca de las tareas a realizar por el alumno como acerca de errores típicos en los programas escritos en Pascal. Un trabajo en la misma línea, pero dedicado a la enseñanza del lenguaje de programación Lisp, es Lisp Tutor [29]. Un poco más allá va The Programmer's Apprentice [30], una herramienta basada en conocimiento que no sólo ayuda en la fase de implementación sino también en las fases de análisis y diseño.

Los trabajos iniciales en el campo de la enseñanza de la programación fueron intentos altamente sofisticados, con una elevada componente de conocimiento. Sin embargo, los resultados fueron algo decepcionantes. Por ejemplo, los diagnósticos automáticos de errores en los programas no son una tarea fácil. Esto es algo que se puso de manifiesto en el sistema PROUST. Cuando se abordaban programas de cierta complejidad surgían problemas para comprenderlos y la capacidad del sistema para identificar errores se reducía drásticamente.

Las tendencias actuales en los métodos de ayuda a la enseñanza de la programación se sitúan

* Este trabajo ha sido financiado por el Ministerio de Ciencia y Tecnología (TIC2002-01961).

al otro lado del espectro. Se trata de enfoques menos sofisticados e "inteligentes". Entre estos enfoques nos encontramos:

- Sistemas que incluyen un reducido entorno de desarrollo
- Entornos basados en ejemplos
- Entornos basados en visualización y animación
- Entornos de simulación

Estos enfoques se tratan, en el orden indicado, en las secciones posteriores.

2. Sistemas con entorno de desarrollo incorporado

Algunos trabajos actuales se centran en el desarrollo de sistemas de enseñanza que incluyen un reducido entorno de desarrollo en el que los alumnos pueden escribir código.

Una idea común que hay detrás de estos trabajos es intentar conseguir entornos de desarrollo menos complejos que los comerciales. Estos últimos, en muchas ocasiones, resultan demasiado enrevesados para los alumnos que están aprendiendo a programar, quienes suelen utilizar un conjunto reducido de las facilidades que los entornos proporcionan.

Un ejemplo bastante reciente es AnimPascal [35], un entorno de enseñanza del lenguaje de programación Pascal que incluye capacidades de edición y compilación de código junto con visualización de la ejecución de los programas. Esta última capacidad consiste en que el alumno puede ver cómo afecta la ejecución de cada sentencia a los valores de las variables y la salida del programa. Esto no es algo novedoso pues los entornos de programación comerciales suelen ofrecer este tipo de facilidades.

Una característica que diferencia a AnimPascal de otros sistemas de enseñanza similares es que recoge el camino seguido por cada alumno en la solución de las cuestiones que aborda. Para cada cuestión se graban las diferentes versiones del programa solución elaborado por el alumno, junto con los resultados de las compilaciones sucesivas.

La información acerca del camino recorrido por el alumno para obtener una solución a un problema la consideramos de gran valor para poder sacar una idea de cómo concibe el alumno la programación y las técnicas de resolución de problemas. El uso concreto que hacen de dicha información en AnimPascal es que es procesada por los profesores posteriormente, quienes la utilizan para encontrar las "lagunas" de conocimiento y los problemas de aprendizaje de los alumnos.

Dentro de esta línea de trabajos se encuentra también el sistema BlueJ [2][24], una continuación del lenguaje y entorno Blue [22][23]. BlueJ es el resultado del trabajo conjunto de dos equipos de investigación: el de la Monash University, en Melbourne (Australia), y el del Mærsk Institute en la University of Southern Denmark.

BlueJ es un entorno de enseñanza para el lenguaje Java fácil de utilizar. Aparte de la facilidad de uso, también se hace énfasis en las técnicas de interacción, dando como resultado un entorno altamente interactivo que promueve la exploración y la experimentación.

Las versiones 1.2.2 y 1.3.0 beta1 de BlueJ, así como documentación acerca del entorno, se pueden obtener a partir de http://www.bluej.org.

3. Entornos basados en ejemplos

Sin lugar a dudas, los humanos utilizamos las soluciones a problemas previos para resolver nuevos problemas. Especialmente en el área de la programación, tanto los programadores con experiencia como los noveles a menudo se sirven de ejemplos de programas, ya hayan sido éstos desarrollados por ellos mismos o no. No en vano existe todo un área de investigación en Reutilización de software. Los profesores, conscientes de lo anterior, no dudamos en utilizar abundantes ejemplos de programas en nuestras clases.

No es de extrañar, por lo tanto, que como forma de ayuda al aprendizaje de la programación algunos trabajos giren en torno a la construcción de bases de ejemplos de programación y al desarrollo de mecanismos de acceso o selección (más o menos sofisticados) [10][18].

La falta de herramientas eficientes de selección de ejemplos es el cuello de botella de los entornos educativos basados en ejemplos. El problema crucial es que los alumnos noveles carecen del conocimiento y la experiencia necesarios para encontrar ejemplos relevantes

mediante herramientas como la selección a través de un menú [27] o la búsqueda mediante palabras clave de la especificación del problema o del propio código [17].

En [10] se analizan los problemas relacionados con la localización de ejemplos relevantes en este tipo de entornos, así como varios enfoques de selección de ejemplos:

- La selección conducida por el estudiante, para la que la tecnología hipermedia resulta altamente prometedora. Básicamente consiste en inspeccionar los ejemplos. La dificultad radica en cómo estructurar los ejemplos en un hiperespacio y qué tipo de enlaces proporcionar para la navegación
- La selección conducida por el sistema, en la que el sistema proporciona al alumno los ejemplos relevantes. Una posibilidad es que el sistema disponga de conocimiento acerca de qué ejemplos se adaptan a cada problema y, en base al problema que esté abordando el alumno, se sugiera lo que corresponda. Un enfoque más sofisticado es que el sistema disponga de conocimiento acerca del alumno y los problemas, y pueda usarlo para seleccionar "en vivo" el ejemplo relevante. En este tipo de enfoque, el alumno no suele poder pasar por alto la sugerencia del sistema.
- La selección colaborativa, en la que ambos agentes (sistema y alumno) intervienen: el primero sugiriendo ejemplos apropiados y el segundo realizando la selección definitiva.

Una idea más novedosa dentro del campo de la enseñanza basada en ejemplos es la que se presenta en WebEx [9], una herramienta basada en Web que permite explorar de manera interactiva ejemplos de programas autoexplicativos.

Lo novedoso de WebEx es el hecho de que los ejemplos sean autoexplicativos. En WebEx además se hace frente al problema de la heterogeneidad de los alumnos: alumnos con diferente nivel inicial de conocimiento y distintas capacidades de adquisición. La heterogeneidad hace que alumnos diferentes necesiten diferentes velocidades, ejemplos y nivel de detalle en las explicaciones de los ejemplos.

El núcleo de WebEx es una base de ejemplos de programación autoexplicativos: cada línea de código va acompañada de texto que aclara el papel de la misma en la solución global y su significado.

Los alumnos, en función de sus necesidades, pueden elegir su estrategia de exploración preferida. La diferencia entre las estrategias de exploración radica, básicamente, en la cantidad de comentarios asociados a los programas que está visible.

Para facilitar la navegación y la localización de ejemplos planean estructurar la base de ejemplos siguiendo un enfoque conceptual. Se trata de identificar los conceptos clave de los contenidos ejemplificados en los programas y mantener enlaces bidireccionales entre los conceptos y los ejemplos.

Otra característica de esta herramienta es que cada acción de un alumno en el entorno queda registrada, de forma que el profesor puede "monitorizar" la actividad de los alumnos y extraer conclusiones acerca de la forma en la que éstos trabajan con los ejemplos.

4. Entornos basados en visualización y animación

Sin duda alguna, uno de los métodos pedagógicos de enseñanza a la programación más explorado es el basado en la visualización y animación de algoritmos [37]. Se trata de presentar representaciones esquemáticas de programas y algoritmos, de manera que a los alumnos les sea más fácil entenderlos.

La animación y visualización de algoritmos se lleva utilizando desde hace más de dos décadas. Se puede decir que el comienzo de la animación y visualización de algoritmos es la cinta de vídeo *Sorting Out Sorting* presentada en 1981 en la conferencia ACM SIGGRAPH [1]. Desde entonces, numerosos sistemas y trabajos se han desarrollado centrándose en diferentes aspectos de la animación y la visualización.

En el World Wide Web podemos encontrar numerosos ejemplos de animaciones y visualizaciones predefinidas de algoritmos diseñadas por los instructores. Según los casos, estas animaciones y visualizaciones permiten un mayor o menor grado de intervención del alumno. En http://www.cs.hope.edu/~alganim/ccaa/index.html existe una enorme recopilación de enlaces a animaciones de algoritmos en el Web.

Aparte de las animaciones predefinidas, se han desarrollado numerosos sistemas de visualización y animación de algoritmos. La tendencia suele ser a conseguir independencia de la plataforma, facilidad de uso, soporte para inclusión del código fuente y descripciones textuales, etc.

A modo de ejemplo citaremos los siguientes:

- ANIMAL (A New Interactive Modeler for Animations in Lectures) [31][34], desarrollado en la Universidad de Siegen (Alemania). El sistema puede ser descargado a partir de http://www.animal.ahrgr.de/, donde también se pueden encontrar documentación, animaciones de ejemplo y publicaciones relativas al sistema.
- LEONARDO [13][16], desarrollado en la Universidad de Roma "La Sapienza" para la animación de programas escritos en C. Desde http://www.dis.uniroma1.it/~demetres/Leonardo/Leonardo.html se puede descargar el sistema (aún no disponible para Windows y Linux) junto con documentación adicional.
- XTANGO y POLKA, desarrollados en el Georgia Institute of Technology por el grupo de John Stasko –creador del sistema TANGO [36]– y accesibles desde http://www.cc.gatech.edu/gvu/softviz/algoanim/.
- JHAVÉ (Java-Hosted Algorithm Visualization Environment) [26], un entorno cliente-servidor que soporta la visualización de algoritmos en tres lenguajes de script diferentes: Samba, Animal y Gaigs. La versión disponible se puede conseguir desde http://csf11.acs.uwosh.edu/.

Otra línea de trabajos dentro de esta tendencia son las denominadas "teaching machines". En este caso el objetivo no es mostrar cómo se comporta un algoritmo concreto para facilitar la comprensión de la idea que hay detrás, sino mostrar el efecto que tiene la ejecución de las sentencias básicas de un cierto lenguaje de programación sobre los elementos de la computadora. Es decir, actúan a modo de "intérpretes" de las sentencias a través de la visualización del estado de la memoria y otros recursos. Se han construido herramientas de este tipo para enseñar lenguajes como C++ [8][38], un subconjunto de Java [3][25], e incluso una especie de máquina de Turing que ha sido usada para introducir la programación formal a los alumnos [14].

Recientemente, se están llevando a cabo proyectos en los que se recopilan tanto texto como animaciones y visualizaciones de algoritmos en los llamados "hypertextbooks" [7], como solución a la existencia actual de numerosas visualizaciones independientes que existen en el World Wide Web.

Para finalizar este apartado, señalar que existen trabajos en los que se ha evaluado la efectividad de los entornos basados en visualización y animación. Por ejemplo, en [32] se relata una experiencia de uso en grupos numerosos de alumnos y los resultados son esperanzadores.

Ahora bien, aún queda mucho camino por andar y los sistemas de visualización de algoritmos deben afrontar una serie de aspectos pedagógicos para que su uso sea más efectivo. Tal y como se desprende de trabajos como [19][33], sería conveniente que los entornos satisficieran, entre otros, requisitos pedagógicos como:

- Ser sistemas de propósito general, de forma que pudiesen integrarse a lo largo de todo un curso y no aplicarse sólo a aspectos muy concretos del temario.
- Permitir la introducción de datos con los que pueda trabajar el algoritmo, teniendo siempre cuidado con no sobrecargar al alumno excesivamente con la entrada de datos.
- Disponer de capacidad de "vuelta atrás", de manera que cuando un alumno se encuentre perdido o confundido por la representación esquemática proporcionada pueda retroceder.
- Disponer de predicción interactiva, es decir, interrumpir con preguntas en los puntos interesantes de la ejecución del algoritmo de forma que el alumno tenga que predecir qué va a ocurrir a continuación. Esto puede ayudar a despertar el interés de los alumnos.

  En un trabajo reciente [20] se pone en tela de juicio el valor de la predicción interactiva si los alumnos no se toman en serio las preguntas sino que la consideran un mero juego de adivinación. Para ello se sugiere satisfacer el siguiente punto.
- Recoger la interacción de los alumnos (las respuestas a las preguntas) en una base de datos, de manera que pueda ser utilizada por los profesores para detectar puntos en los que

los alumnos tienen dificultad e incluso como parte de la evaluación que se haga de los alumnos.

5. Entornos basados en simulación

Para finalizar, revisamos aquellos trabajos que se centran en el desarrollo de entornos de simulación. El objetivo de estos entornos es ayudar a las alumnos a comprender cómo funcionan los programas. Pero no se trata de visualizar la ejecución de los mismos viendo cómo afectan a los recursos de la computadora sino de ayudar a entender el efecto de las instrucciones de los lenguajes mediante algún tipo de simulación. Los alumnos observan un "mundo imaginario" en el que habitan seres cuyo comportamiento viene dictado por la ejecución de las instrucciones del programa. La ejecución de cada instrucción tiene definida de antemano una acción concreta que los seres llevan a cabo en ese mundo.

Indudablemente uno de los entornos de este tipo que ha tenido más éxito ha sido Karel, The Robot [28], utilizado para la presentación y enseñanza de los principios y estructuras básicos de la programación estructurada.

El robot habita en un mundo bidimensional muy simple compuesto por avenidas que van de norte a sur y calles que van de este a oeste. Las calles y las avenidas están numeradas de forma similar a cómo lo está el primer cuadrante del plano cartesiano, y existen muros impenetrables que hacen las veces de ejes de dicho cuadrante. Los robots sólo pueden estar en intersecciones de calles y avenidas, y sólo pueden estar mirando en una dirección (norte, sur, este u oeste). También puede haber muros impenetrables entre intersecciones adyacentes. Aparte de los robots, en el mundo existen otro tipo de objetos: timbres , los cuales sólo pueden estar situados en intersecciones.

El robot cuenta con una serie de periféricos: tres cámaras de vídeo que le permiten mirar de frente, a izquierda y a derecha; una brújula que indica la dirección en la que mira; un micrófono que le permite escuchar el sonido de los timbres; un brazo que le permite coger timbres, introducirlos y sacarlos de recipientes, y colocarlos de nuevo en el suelo; y una bolsa para guardar timbres.

Las seis acciones básicas que puede realizar el robot son: encenderse; moverse hacia delante; girar a la izquierda; coger un timbre e introducirlo en la bolsa; sacar un timbre de la bolsa y colocarlo en el suelo; y apagarse. Aparte de este reducido lenguaje del robot (correspondiente al igualmente reducido juego de instrucciones básicas del lenguaje de programación que se puede emplear para escribir los programas), los alumnos pueden definir sus propias funciones (implementadas en términos del lenguaje básico del robot), y el robot también puede comprobar condiciones para poder elegir entre alternativas o repetir acciones. De lo que no se dispone es de componentes básicas como la creación y manipulación algebraica de datos así como operaciones de entrada/salida.

El lenguaje utilizado por Karel es un lenguaje tipo Pascal. Cuando se ejecutan los programas, los alumnos pueden ver el mundo de Karel y observar los cambios que se producen en él a medida que avanza la ejecución.

Se puede obtener una versión de Karel a partir de http://www.sourceforge.net.

Tal ha sido el éxito de Karel que Joseph Bergin ha realizado una versión para la enseñanza de la Programación orientada a objetos utilizando una sintaxis similar a la de C++ y Java: Karel++ [5]. Los dos primeros capítulos de [5] pueden leerse on-line desde la página de Karel++: http://www.csis.pace.edu/~bergin/karel.html. Una versión más reciente con sintaxis Java pura está disponible desde hace poco también gracias a Joseph Bergin: Karel J. Robot [4][6]. Finalmente, JKarelRobot [11] es otra extensión del inicial que soporta estilos de programación tipo Pascal, Java y Lisp.

Otro ejemplo de este tipo de entornos de enseñanza es Alice [15], un entorno de animación en 3-D que permite crear mundos virtuales que reflejan el estado del programa y van cambiando a medida que cambia dicho estado. Alice ha sido desarrollado por el grupo de investigación Stage3 de la Universidad de Carnegie Mellon y está disponible desde http://www.alice.org/. Está siendo utilizado actualmente para abordar una estrategia *objects-first* en asignaturas de introducción a la programación [12].

6. Conclusión

Esta ponencia ha realizado un repaso en anchura y en profundidad por diferentes métodos pedagógicos que están siendo empleados en la actualidad en el área de la enseñanza de la programación.

Se han revisado cuatro métodos pedagógicos: los sistemas que incluyen un "limitado" entorno de programación, los entornos basados en ejemplos, los entornos basados en visualización y animación, y los entornos basados en simulación.

Todos los métodos son prometedores y están siendo aplicados con aparente éxito en diferentes centros. Sin embargo, los estudios de su efectividad son en su mayoría exploratorios. En este sentido, convendría realizar estudios empíricos que constatasen las aparentes mejoras que introducen en el proceso de aprendizaje de los alumnos.

Algunos retos que se pueden plantear a los diferentes métodos, y que en algunos trabajos concretos ya se afrontan, son:

- La inclusión de mecanismos que permitan el seguimiento del alumno. La motivación es doble: por un lado, ayudar a los instructores en la localización de lagunas cognitivas; por otro lado, poder utilizar los resultados como parte de la valoración general del conocimiento de los alumnos, a la vez que éstos utilizan de forma más responsable las herramientas.
- Afrontar la heterogeneidad (tanto de capacidad como de predisposición) de los alumnos. De esta forma, las herramientas resultan atractivas para un porcentaje mayor del alumnado.
- La introducción de mecanismos que permitan la colaboración y cooperación entre alumnos, soportando así el trabajo en equipo para la resolución de problemas en programación.

Referencias

[1] Baecker, R., 1981: "Sorting Out Sorting", *Procs. of SIGGRAPH*.

[2] Barnes, D.J., and Kölling, M., 2003: *Objects First with Java – A Practical Introduction using BlueJ*, Prentice Hall.

[3] Ben-Ari, M., Myller, N., Sutinen, E., and Tarhio, J., 2002: "Perspectives on program animation with Jeliot", *Procs. of Software Visualization: International Seminar, Lecture Notes in Computer Science 2269*.

[4] Bergin, J., 2000: "Introducing Objects with Karel J. Robot", *Procs. of the Workshop "Tools and Environments for Understanding Object-Oriented Concepts", European Conference on Object-Oriented Programming*.

[5] Bergin, J., Stehlik, M., Roberts, J., and Pattis, R., 1997: *Karel++ - A Gentle Introduction to the Art of Object-Oriented Programming*, John Wiley & Sons.

[6] Bergin, J., Stehlik, M., Roberts, J., and Pattis, R., 2003: *Karel J. Robot - A Gentle Introduction to the Art of Object-Oriented Programming in Java* (manuscrito sin publicar, disponible en http://csis.pace.edu/~bergin/KarelJava/Karel++JavaEdition.html)

[7] Boroni, C.M., Goosey, F.W., Grinder, M.T., and Ross, R.J., 2001: "Engaging Students with Active Learning Resources: Hypertextbooks for the Web", *Procs. of the 32nd SIGCSE Technical Symposium on Computer Science Education.*

[8] Bruce-Lockhart, M.P., and Norwell, T.S., 2000: "Lifting the Hood of the Computer: Program Animation with the Teaching Machine", *Procs. of the Canadian Electrical and Computer Engineering Conference*.

[9] Brusilovsky, P., 2001: "WebEx: Learning from Examples in a Programming Course", *Procs. of the World Conference of the WWW and Internet*.

[10] Brusilovsky, P., and Weber, G., 1996: "Collaborative example selection in an intelligent example-based programming environment", *Procs. of the International Conference on Learning Sciences*.

[11] Buck, D., and Stucki, D.J., 2001: "JKarelRobot: A Case Study in Supporting Levels of Cognitive Development in the Computer Science Curriculum", *Procs. of the 32nd SIGCSE Technical Symposium on Computer Science Education.*

[12] Cooper, S., Dann, W., and Pausch, R., 2003: "Teaching Objects-First in Introductory Computer Science", aceptado para *34th SIGCSE Technical Symposium on Computer*

Science Education (disponible desde http://db.grinnell.edu/sigcse/sigcse2003/programAtaGlance.asp)

[13] Crescenzi, P., Demetrescu, C., Finocchi, I. and Petreschi, R., 2000: "Reversible execution and visualization of programs with Leonardo", *Journal of Visual Languages and Computing*, 11(2).

[14] Dagdilelis, v., and Satratzemi, M., 2001: "Post's Machine: A Didactic Microworld as an Introduction to Formal Programming", *Education and Information Technologies*, 6(2).

[15] Dann, W., Cooper, S., and Pausch, R., 2000: "Making the Connection: Programming with Animated Small World", *Procs. of the Annual Conference on Innovation and Technology in Computer Science Education*.

[16] Demetrescu, C., and Finocchi, I., 2000: "Smooth animation of algorithms in a declarative framework", *Journal of Visual Languages and Computing*, 12(3).

[17] Faries, J.M., and Reiser, B.J., 1988: "Access and use of previous solutions in a problem solving situation", *Procs. of the 10th Annual Conference of the Cognitive Science Society*.

[18] Guzdial, M., 1995: "Software-realized scaffolding to facilitate programming for science learning", *Interactive Learning Environments*, 4(1).

[19] Hundhausen, C., Douglas, S., and Stasko, J., 2002: "A Meta-Study of Algorithm Visualization Effectiveness", *Journal of Visual Languages and Computing*, 13(3).

[20] Jarc, D., Feldman, M.b., and Heller, R.S., 2000: "Assessing the Benefits of Interactive Prediction Using Web-based Algorithm Animation Courseware", *Procs. of 31st SIGCSE Technical Symposium on Computer Science Education*.

[21] Johnson, L.W., and Soloway, E., 1985: "PROUST: Knowledge-Based Program Understanding", *IEEE Transactions on Software Engineering*, 11(3).

[22] Kölling, M., 1999: "Teaching Object Orientation with the Blue Environment", *Journal of Object-Oriented Programming*, 12(2).

[23] Kölling, M. and Rosenberg, J., 1996: "Blue - A Language for Teaching Object-Oriented Programming", *Procs. of 27th SIGCSE Technical Symposium on Computer Science Education*.

[24] Kölling, M., and Rosenberg, J., 2001: "Guidelines for Teaching Object Orientation with Java", *Procs. of the Annual Conference on Innovation and Technology in Computer Science Education*.

[25] Levy, R., Ben-Ari, M., Uronen, P., 2003: "The Jeliot 2000 Program Animation System", *Computer & Education*, 40(1).

[26] Naps, T., Eagan, J., and Norton, L., 2000: "JHAVÉ – An Environment to Actively Engage Students in Web-based Algorithm Visualization", *Procs. of the 31st SIGCSE Technical Symposium on Computer Science Education.*

[27] Neal, L.R., 1989: "A system for example-based programming", *Procs. of Human Factors in Computing Systems*.

[28] Pattis, R., 1981: *Karel the Robot*, John Wiley & Sons.

[29] Reiser, B.J., Anderson, J.R., and Farrell, R.G., 1985: "Dynamic Student Modelling in an Intelligent Tutor for Lisp Programming", *Procs. of the 9th International Joint Conference on Artificial Intelligence*.

[30] Rich, C., and Waters, R.C., 1988: "The Programmer's Apprentice: A Research Overview", *Computer*, 21(11).

[31] Röβling, G., and Freisleben, B., 2001: "The Extensible Visualization Framework ANIMAL", *Procs. of the 32nd SIGCSE Technical Symposium on Computer Science Education.*

[32] Röβling, G., and Freisleben, B., 2000: "Experiences in Using animations in Introductory Computer Science Lectures", *Procs. of the 31st SIGCSE Technical Symposium on Computer Science Education.*

[33] Röβling, G., and Naps, T., 2002: "A Testbed for Pedagogical Requirements in Algorithm Visualizations", *Procs. of the Annual Conference on Innovation and Technology in Computer Science Education*.

[34] Röβling, G., Schüler, M., and Freisleben, B., 2000: "The ANIMAL Algorithm Animation Tool", *Procs. of the Annual Conference on Innovation and Technology in Computer Science Education*.

[35] Satratzemi, M., Daglilelis, V., and Evagelidis, G., 2001: "A system for program

visualization and problem-solving path assessment of novice programmers", *Procs. of the Annual Conference on Innovation and Technology in Computer Science Education*.

[36] Stasko, J., 1990: "TANGO: A Framework and System for Algorithm Animation", *Computer*, 23(9).

[37] Stasko, J., Domingue, J., Price, B.A., Brown, M.H., 1998: *Software Visualization: Programming as a Multimedia Experience*, MIT Press.

[38] The Teaching Machine of C++: http://www.engr.mun.ca/~theo/TM.