INVESTIGACIÓN y CIENCIA
DE LA UNIVERSIDAD AUTÓNOMA DE AGUASCALIENTES
Número 48, (17-29) Enero-Abril 2010

Un estudio piloto sobre el efecto de los tutores cognitivos para la enseñanza de conceptos básicos de programación[1]

Carlos Argelio Arévalo Mercado[2]
Juan Manuel Gómez Reynoso [3]

RESUMEN

La enseñanza de la programación presenta problemas recurrentes en los alumnos del primer año de licenciatura, debido a la complejidad de su estructura de conocimiento. Diversos factores cognitivos han sido identificados en la literatura indicando un efecto en el rendimiento de los alumnos. Herramientas basadas en Tecnologías de la Información (TI) han sido desarrolladas para ayudar en el aprendizaje de la programación, contando con atributos de diseño y resultados diversos. El presente estudio reporta la aplicación de un estudio piloto con dos grupos de estudiantes de nivel medio del estado de Aguascalientes, usando prototipos de tutores cognitivos. Se aplicó una evaluación para medir el grado de retención así como pruebas estadísticas para analizar los datos. Los resultados preliminares no muestran diferencias estadísticamente significativas en el rendimiento de ambos grupos. Se argumenta que la ausencia de un instructor humano y la falta de una capacitación presencial previa, entre otras variables, pudo influir en los resultados obtenidos.

Palabras clave: Tutores cognitivos, programación, diseño instruccional, teoría de flexibilidad cognitiva, software para la enseñanza, e-learning.

***Key words**: Cognitive tutors, programming, instructional design, e-learning, cognitive flexibility theory, software for teaching.*

Recibido: 20 de junio de 2009, aceptado: 7 de septiembre de 2009

ABSTRACT

Learning to program is a recurring problem to first year undergraduate students, given its complex knowledge structure. Several cognitive factors have been identified in literature to have an effect in student performance. Tools based on Information Technology (IT) have been developed over time to aid apprentices in learning to program, with varying results and design attributes. This study reports the use of prototype cognitive tutors with two groups of high school students in the state of Aguascalientes, Mexico. An evaluation of performance was made and statistical tests were applied to collected data. Preliminary results show no statistical significant difference between the two groups. It is argued that the absence of a human instructor and the lack of face-to-face training could have influenced the results of the experimental group.

INTRODUCCIÓN

El problema de la enseñanza de la programación

La enseñanza de la programación, entendida como la habilidad para escribir programas de computadora, ha sido y es el tema de numerosas investigaciones a nivel mundial. La literatura reporta de manera recurrente (Dijkstra, 1989; Milne & Rowe, 2002) que los alumnos tienen dificultades para aprender a programar, hecho reflejado en los altos niveles de reprobación. Es claramente reconocido que la naturaleza del tema de la programación es compleja, debido, entre otros factores, a que su estructura tiende a ser jerárquica y no lineal, en tanto que los conceptos que

[1] El estudio se deriva de un proyecto de investigación financiado internamente por la Universidad Autónoma de Aguascalientes, denominado "Desarrollo de un objeto de aprendizaje para un dominio de conocimiento complejo".

[2] Departamento de Sistemas de Información, Centro de Ciencias Básicas, Universidad Autónoma de Aguascalientes, carevalo@correo.uaa.mx.

[3] Departamento de Sistemas Electrónicos, Centro de Ciencias Básicas, UAA, jmgr@correo.uaa.mx.

la conforman están fuertemente relacionados y resulta poco efectivo enseñarlos de manera aislada y secuencial (Du Boulay, 1989; Jenkins, 2002; Milne & Rowe, 2002). Dada la complejidad del problema, los investigadores han estudiado el tema desde diversas perspectivas y, a la fecha, han encontrado evidencia sobre algunos factores que influyen en la capacidad del estudiante para aprender a programar. Se menciona que la experiencia previa al primer curso de programación (Hagan & Markham, 2000; Holden & Weeden, 2005) afecta el desempeño del alumno durante los cursos introductorios. Los modelos mentales también influyen en el rendimiento del aprendiz de programador (Bayman, 1983; Fixx, 1993; Hegarty, 1993; Ma, 2007). Aspectos cognitivos como la llamada autoeficacia (Heggestad, 2005; Ramalingam, 2004; Wiedenbeck, LaBelle, & Kain, 2004), entendida como "lo bien que uno puede ejecutar cursos de acción requeridos para llevar a cabo situaciones prospectivas" (Bandura, 1982, pág. 122) y la ansiedad computacional (Brosnan, 1998) son factores sobre los cuales se tiene evidencia de un efecto en el rendimiento de los estudiantes. Finalmente, se habla de que la habilidad matemática (Hu, 2006; White, 2003) y los estilos de aprendizaje (Sadler-Smith & Smith, 2004; Thomas, Ratcliffe, Woodbury, & Jarman, 2002) también tienen influencia sobre la facilidad de aprendizaje del tema.

Por ello, estos factores que afectan el rendimiento del estudiante de programación hacen que sea sumamente difícil, desde el punto de vista pedagógico, el diseño de estrategias de enseñanza efectivas que vayan más allá de los métodos tradicionales. Por otro lado, es reconocido que el modo de enseñanza por medio de tutores humanos –el cual enseña a uno o dos alumnos– produce un efecto significativamente mejor (Bloom, 1984) que la enseñanza convencional dentro de un salón de clase (donde un profesor enseña a un grupo aproximado de 30 alumnos). De tal suerte que los métodos de enseñanza apoyados por TI buscan reproducir el efecto (du Boulay, 2000) que tiene esta modalidad de enseñanza personalizada.

Herramientas para la enseñanza de la programación basadas en Tecnologías de Información

A lo largo del tiempo los investigadores han desarrollado herramientas basadas en Tecnologías de Información (TI) buscando apoyar el proceso de enseñanza de la programación. La naturaleza de estas herramientas –y los resultados reportados– son sumamente diversos y una clasificación exhaustiva de ellos rebasa el alcance del presente artículo[4]. En general, las herramientas de apoyo a la enseñanza de la programación tienen en común los siguientes atributos:

- **Ayudas visuales.** Estas herramientas proporcionan apoyo al aprendiz, mediante animaciones del comportamiento de algoritmos (Hamer, 2004; Naps, 1998) o de conceptos complejos tales como la recursividad (Jehng, 1999).
- **Multimedia.** El uso de audio, video e interactividad también ha sido utilizado para ayudar en la comprensión de conceptos de programación (Chansilp, 2004; Cooper, 2003; McKeown, 2004).
- **Minilenguajes.** Este tipo de aplicaciones (Brusilovsky, 1998; McIver, 1999) buscan reducir la complejidad de los lenguajes de programación tradicionales, disminuyendo la cantidad de funciones disponibles y aumentando la usabilidad.
- **Uso de inteligencia artificial.** Los tutores inteligentes son una tecnología que intenta brindar retroalimentación dirigida al aprendiz de programador, al tiempo que rastrea patrones de uso (Brusilovsky, 1995; J. R. Anderson, Corbet, & K. R. Koedinger, 1995; Kumar, 2006).
- **Reutilización.** En este ámbito, los llamados objetos de aprendizaje (Wiley, 2000) prometen la reutilización de contenidos en diversos contextos por medio de estándares para su ensamble y búsqueda (Boyle, 2006; Kujansuu, 2006; Matthíasdóttir, 2006; Moisey, 2003; Neven, 2002).

En este contexto, los objetivos del presente estudio piloto consistieron en medir el efecto de un tipo especial de tutor inteligente sobre el aprendizaje de los principios básicos de la programación en alumnos de nivel preparatoria. El modelo de investigación aplicado considera al desempeño/aprendizaje de la programación como variable dependiente y al método de estudio como variable independiente. Como objetivo secundario, se buscó detectar cuáles variables ambientales o cognitivas adicionales que no fueron controladas, pudieron haber tenido un efecto significativo en el proceso de transferencia de conocimiento. En este sentido, el estudio

4 Interesados en el tema pueden consultar Kelleher, 2003.

fue de tipo exploratorio. Se hipotetiza que el uso de una herramienta de enseñanza de la programación, como los tutores cognitivos, influyen positivamente en el aprendizaje de los conceptos básicos de programación.
Los tutores aquí descritos se encuentran en etapa de evaluación en forma de prototipos y los resultados obtenidos servirán para su posterior refinamiento.

MATERIALES Y MÉTODOS

Tutores cognitivos

La literatura sobre herramientas de enseñanza basadas en software reporta un tipo especial de aplicación conocida como tutor cognitivo (J. R. Anderson *et al.*, 1995), que cuenta con un historial positivo en la enseñanza de temas tales como el álgebra y la geometría (K. Koedinger & J. Anderson, 1997) y la enseñanza de lenguajes de programación de inteligencia artificial, tales como LISP (Corbett, 1993). La teoría detrás de estas herramientas es cognitiva conocida como ACT-R *(Adaptive Control of Tought - Rational,* J. Anderson, 2004; J.R. Anderson, 1996*)*, misma que busca comprender y simular el funcionamiento de la mente humana. Sin embargo, el costo de desarrollo de este tipo de aplicaciones es alto (K. Koedinger, V. Aleven, Hefferman, B. McLaren, & Hockenberry, 2004; Murray, 1999). Por ejemplo (K. Koedinger et al., 2004, pág. 8), mencionan que el esfuerzo estimado requerido para el desarrollo de un tutor inteligente se encuentra entre 100 y 1000 horas, por cada hora de instrucción. En vista de tal problemática, algunos investigadores (V. Aleven, B. McLaren, J. Sewall, & K. Koedinger, 2006; Vincent Aleven, Bruce McLaren, Jonathan Sewall, & Keneth Koedinger, 2006), han buscado maneras de disminuir el tiempo y costo de desarrollo de los tutores cognitivos, eliminando el requerimiento de contar con conocimientos de modelación cognitiva y hasta cierto punto, de conocimientos de programación.

En el presente estudio, se utilizó la metodología propuesta por (K. Koedinger *et al.,* 2004) en donde proponen las siguientes etapas para el desarrollo de tutores cognitivos, utilizando la herramienta CTAT[5] (Cognitive Tutors Authoring Tools) :

1.- Crear una interfaz gráfica, para ser utilizada por el estudiante.

Para esta investigación se utilizó el IDE Netbeans 5.0 para el desarrollo de la interfaz gráfica para el estudiante. Cabe mencionar que la herramienta CTAT incluye una serie de '*Widgets*' (objetos) que permiten lograr interactividad con el aprendiz y monitorear el comportamiento de sus respuestas (ver **Figura 1**).

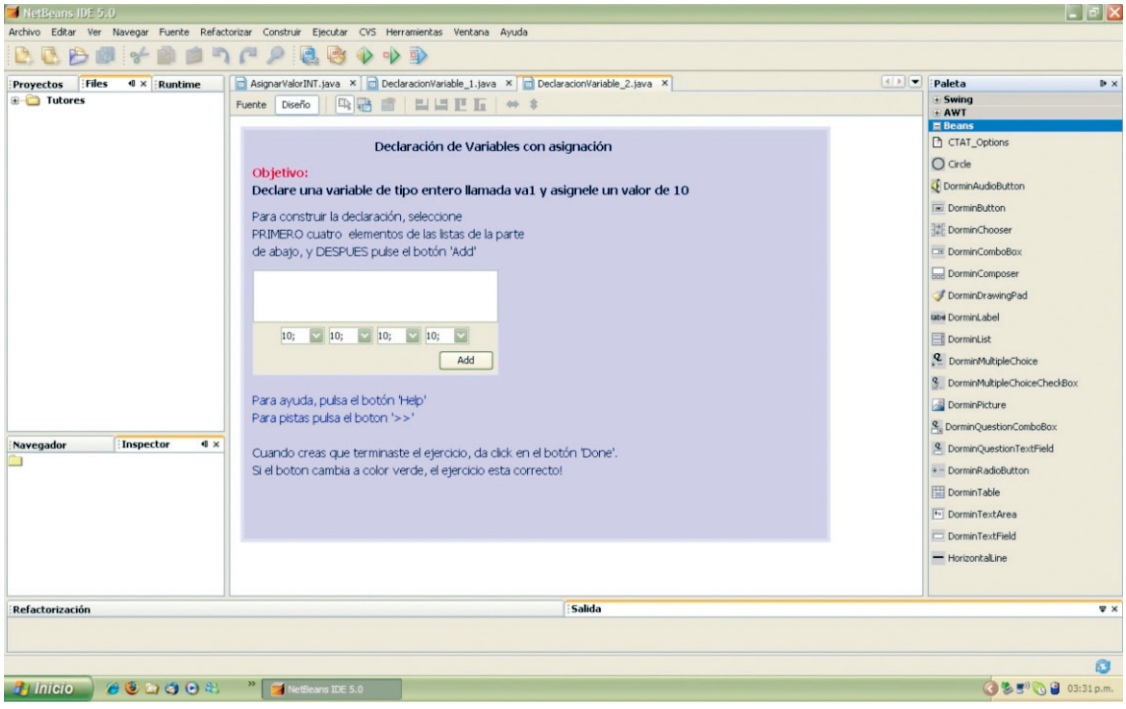


Figura 1. Desarrollo de interfaz gráfica.

5 http://ctat.pact.cs.cmu.edu/

INVESTIGACIÓN y CIENCIA
DE LA UNIVERSIDAD AUTÓNOMA DE AGUASCALIENTES
Número 48, (17-29) Enero-Abril 2010

2.- Demostrar soluciones alternativas correctas e incorrectas.

Este paso consiste en que una vez diseñada la interfaz de estudiante, se procede a crear 'por demostración' las posibles alternativas o secuencias de acciones que el estudiante puede potencialmente seleccionar en la interfase.

3.- Anotar los pasos de solución en un Árbol de comportamiento (Behavior graph).

La demostración del paso anterior, se registra en un árbol de comportamiento, anotando mensajes de ayuda, mensajes de retroalimentación y etiquetas para los conceptos y habilidades asociadas. Esta actividad es particularmente importante, ya que proporciona elementos de interactividad y retroalimentación al alumno (ver **Figura 2**). Los caminos correctos e incorrectos se señalan según corresponda y se anotan etiquetas a cada estado que corresponden a una habilidad concreta.

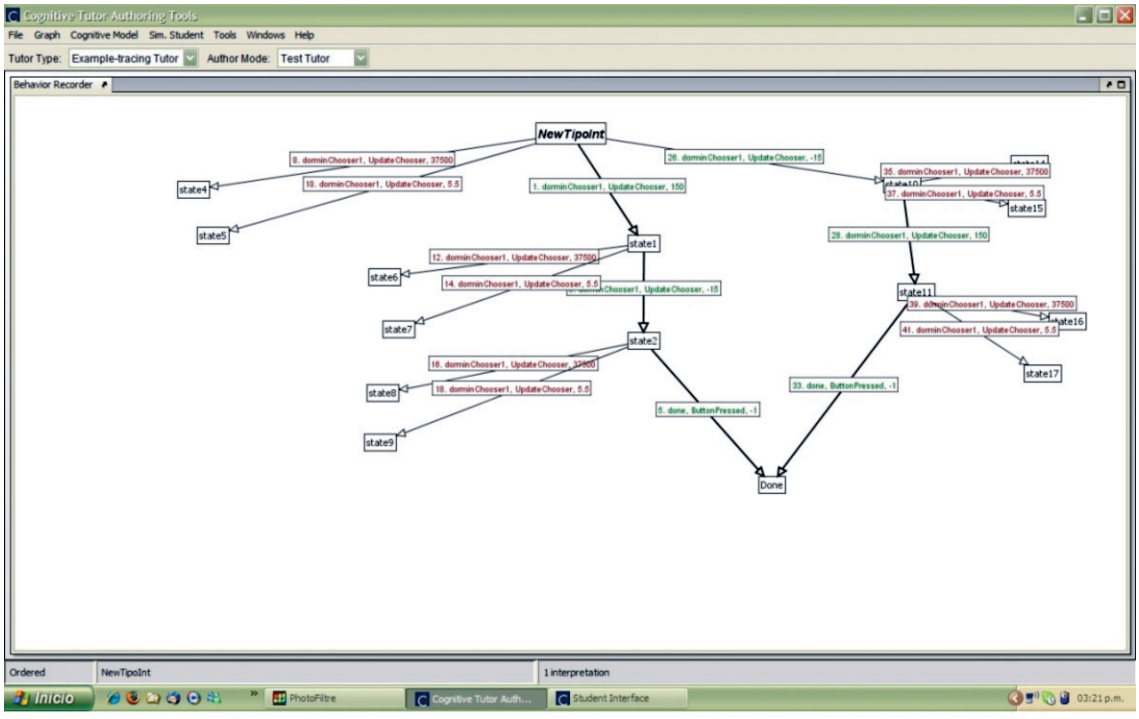


Figura 2. Árbol de comportamiento para una interfaz gráfica.

4.- Inspeccionar y revisar la matriz de habilidades.

Los tutores desarrollados con la herramienta CTAT pueden ser de dos tipos: Tutores por demostración (example-trace tutors) o tutores cognitivos. Los primeros, sirven de prototipos para los segundos y se desarrollan por demostración sin necesidad de contar con muchos conocimientos de programación y modelación cognitiva. Para muchos casos, el tutor por demostración puede ser suficiente y conviene empezar por ahí, antes de desarrollar los tutores cognitivos, que por su parte requieren mayor esfuerzo de diseño. Una parte central de este diseño es la matriz de habilidades, que a su vez, sirve de base para el desarrollo de las reglas de producción que darán un comportamiento inteligente y, más general, al tutor. Para este estudio, este paso no fue llevado a cabo, ya que los mini-tutores fueron desarrollados por demostración.

En el ámbito de la programación, los mini-tutores resultantes se enfocaron a la instrucción

de los conceptos de tipo de dato entero, declaración y asignación de valores enteros en una sola sentencia (ver **Figura 3**). Se puso especial cuidado en los mensajes de retroalimentación y de ayuda al alumno (ver **Figura 4**).

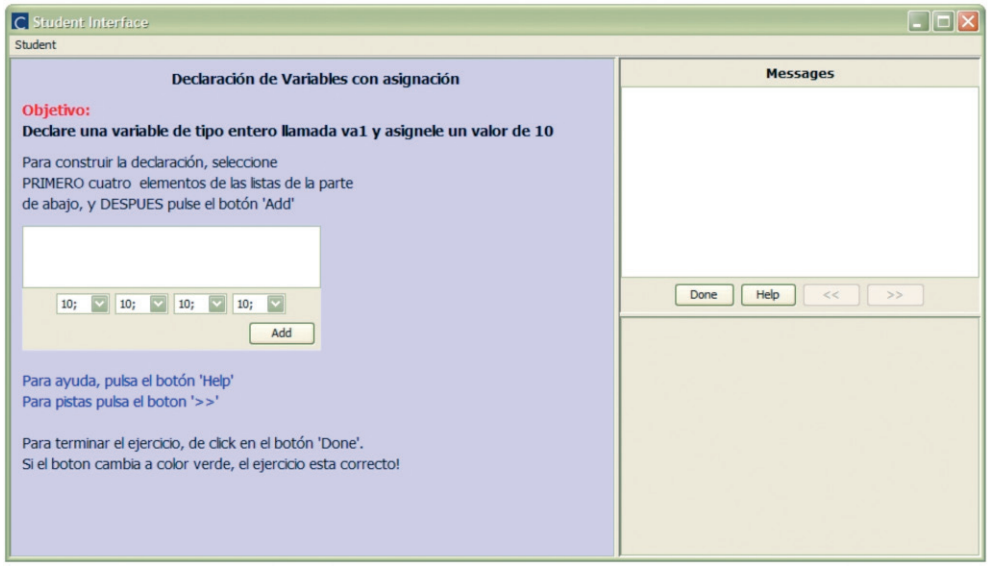


Figura 3. Ejemplo de mini-tutor para el concepto de 'declaración y asignación de valores a variables enteras'.

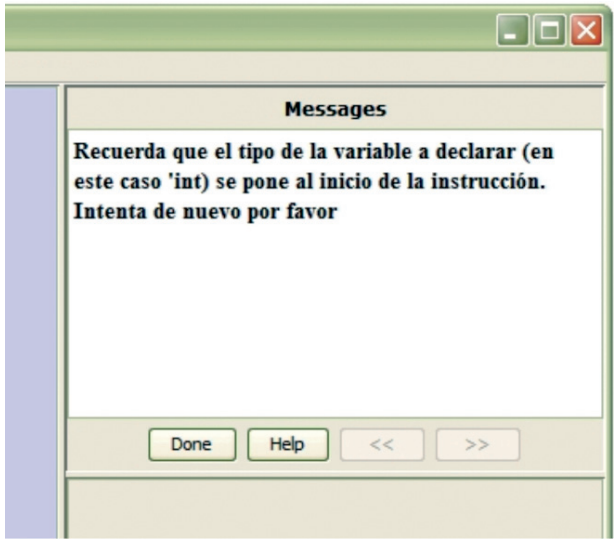


Figura 4. Ejemplo de mensaje de retroalimentación.

Para el despliegue de los mini-tutores se utilizó una plataforma LMS[6] (*Learning Management System*), en donde se diseñó un curso piloto (ver **Figura 5**), simulando el estudio de un tema, en este caso, los tipos de datos y variables enteras en lenguaje C. Se desarrollaron lecciones textuales a manera de explicación teórica, presentadas a los participantes en forma de páginas *web* estáticas. Los mini-tutores cognitivos, se pusieron a disposición de los alumnos participantes al final de cada lección.

6 Puede accederse en la dirreción: http://dsi.ccbas.uaa.mx/moodle.

INVESTIGACIÓN y CIENCIA
DE LA UNIVERSIDAD AUTÓNOMA DE AGUASCALIENTES
Número 48, (17-29) Enero-Abril 2010

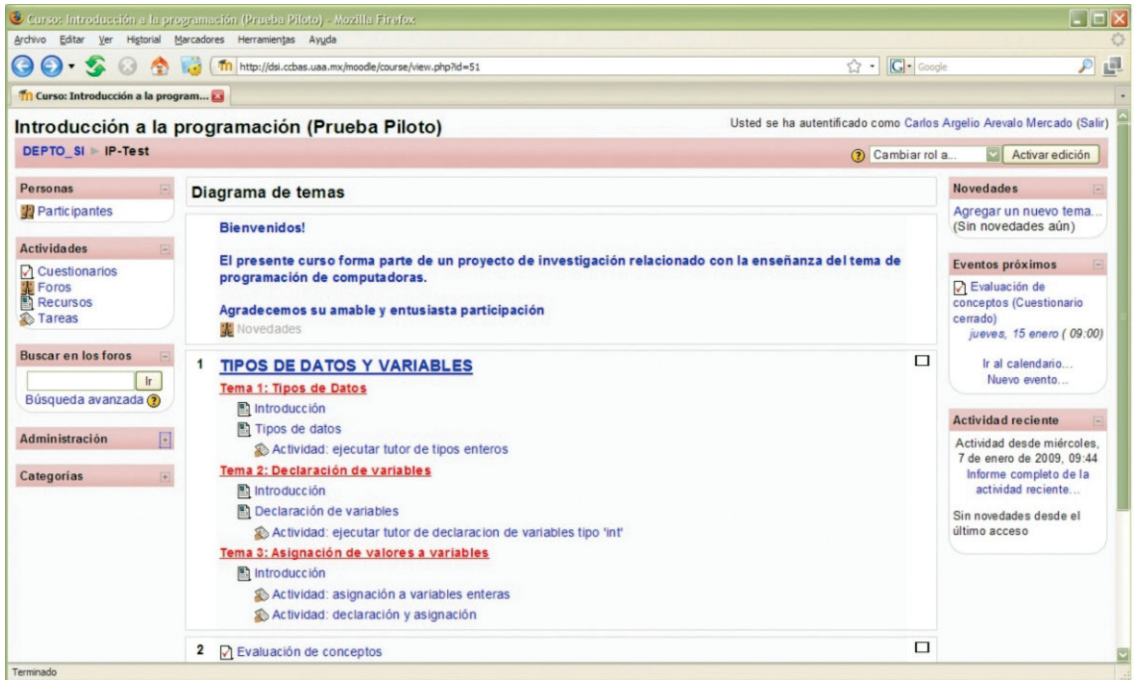


Figura 5. Curso piloto de introducción a la programación.

Diseño instruccional

El diseño instruccional se define como "la ciencia de crear especificaciones detalladas para el desarrollo, evaluación y mantenimiento de situaciones que facilitan el aprendizaje de unidades temáticas tanto grandes como pequeñas" (Richey, 1986, pág. 9).

Rothwell y Kazanas (1992), proponen un proceso de diseño instruccional (ver **Figura 6**), en el cual se busca enfocar el diseño del material didáctico (sea éste tradicional o basado en TI) hacia las necesidades específicas del aprendiz. Sin embargo, Spiro, *et. al*. 2003, argumentan que hay una base común para el fracaso de muchos sistemas instruccionales, la cual tiene que ver con problemas básicos en el diseño del propio material de apoyo. R.J. Spiro, *et al.,* señalan que los métodos de instrucción tradicionales que toman un enfoque lineal, no suelen tener problemas cuando el material está bien estructurado y es de naturaleza simple. En cambio "cuando el contenido aumenta en complejidad y poca estructuración, cantidades crecientes de información se pierden al usar los enfoques lineales y métodos de organización unidimensionales que tradicionalmente los acompañan" (R. Spiro *et al.*, 1991, pág. 21). Por lo tanto, para que la instrucción sea efectiva, diversos tópicos altamente entrelazados deben considerarse de manera simultánea y no lineal.

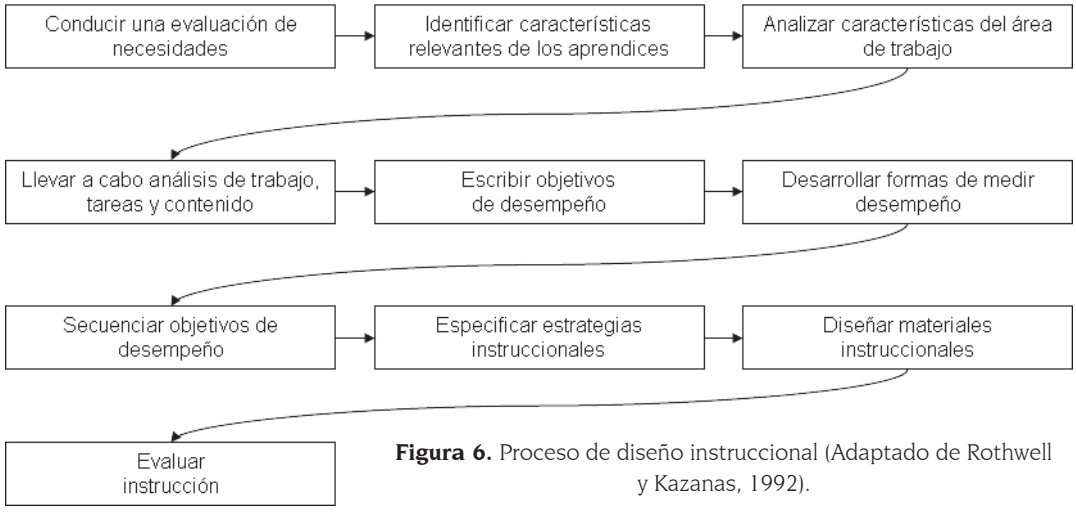


Figura 6. Proceso de diseño instruccional (Adaptado de Rothwell y Kazanas, 1992).

Para el presente estudio, las primeras tres etapas del proceso de diseño instruccional se llevaron a cabo mediante un estudio cualitativo, en la modalidad de sesión de Grupo Enfoque (*Focus Group*) a alumnos de tercer semestre[7] de la Licenciatura en Informática de la Universidad Autónoma de Aguascalientes, en el que, por medio de análisis hermenéutico, se detectó lo siguiente:

Tabla 1: Resultados de análisis hermenéutico, sesión de grupo de enfoque (*Focus Group*)

Tema relacionado	Ocurrencias
Interfase de usuario / usabilidad • Mensajes de error – retroalimentación • Idioma inglés • Exceso de características • Características didácticas de la herramienta • Visualización	9
Conocimiento cumulativo / conocimiento previo de la programación	7
Modelos mentales	5
Estilo de enseñanza del profesor – material didáctico proporcionado	4
Estilo de aprendizaje del alumno	2
Dificultad en el cambio de sintaxis de un lenguaje a otro	3
Aprendizaje por descubrimiento	3

Las restantes actividades del proceso de diseño instruccional tienen que ver con un análisis temático del dominio de conocimiento, en este caso, el aprendizaje de la programación. Un ejemplo simplificado de la descomposición temática del mismo puede visualizarse en el siguiente mapa conceptual (ver **Figura 7**).

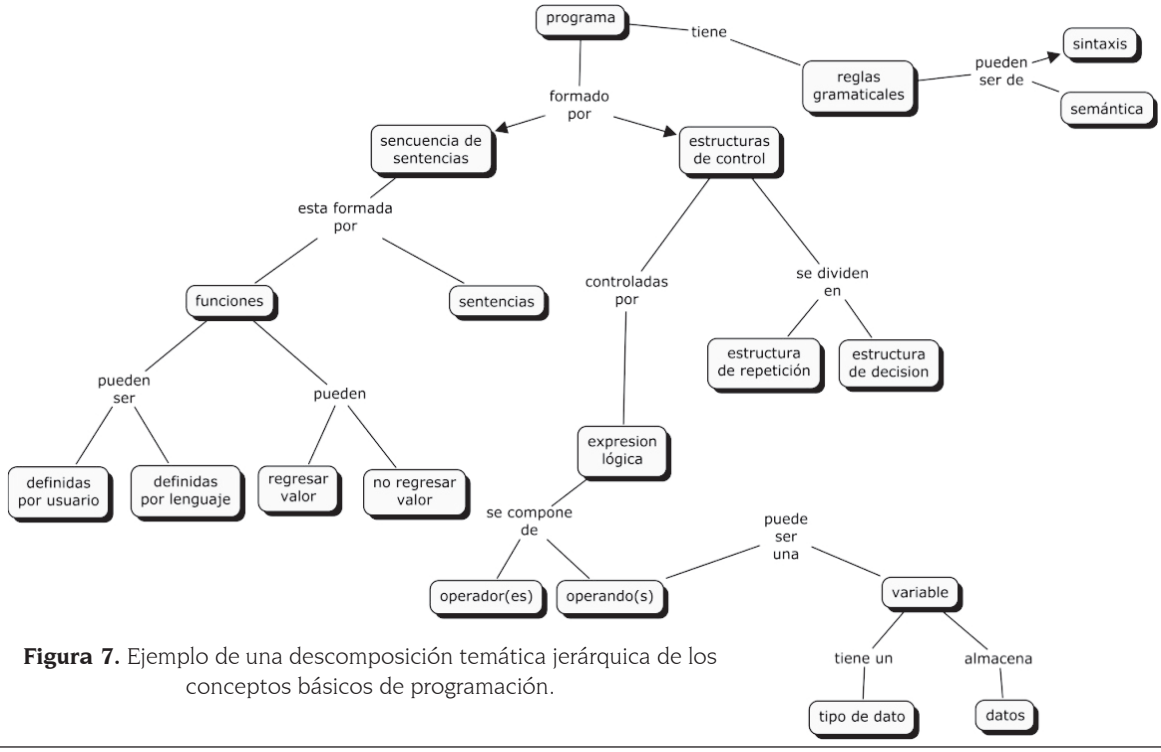


Figura 7. Ejemplo de una descomposición temática jerárquica de los conceptos básicos de programación.

[7] Notar que esta sesión de enfoque fue previa al estudio experimental realizado con alumnos de preparatoria. Estos datos cualitativos se recolectaron con el fin de entender cuáles situaciones habían sido más problemáticas para los alumnos de los primeros semestres de Licenciatura en Informática de la UAA.

INVESTIGACIÓN y CIENCIA
DE LA UNIVERSIDAD AUTÓNOMA DE AGUASCALIENTES
Número 48, (17-29) Enero-Abril 2010

Si se toma en cuenta la interrelación que existe entre los conceptos del dominio (siguiendo las ideas indicadas por Rand J. Spiro & Collins, 2007), enfocándonos, por ejemplo, en el concepto de 'variable', podemos observar la siguiente situación de no-linealidad (ver **Figura 8**), en donde resalta la naturaleza compleja de los conceptos de programación, donde el concepto 'variable' se relaciona con otros conceptos, tales como funciones, expresiones lógicas, estructuras de control y parámetros, entre otros posibles.

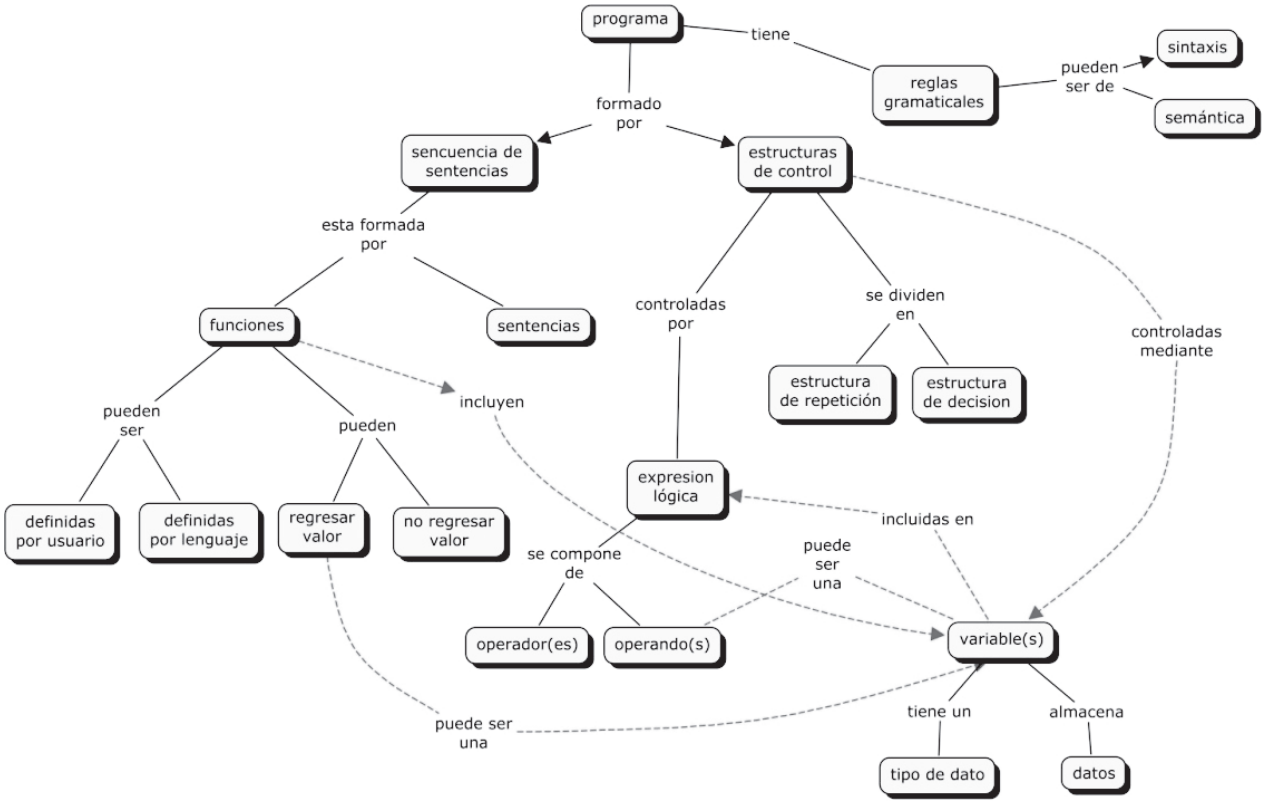


Figura 8. Ejemplo de una descomposición temática y jerárquica de los conceptos básicos de programación, incluyendo temas relacionados con el concepto 'variable'.

Condiciones del estudio

Se llevó a cabo un estudio cuasi-experimental con alumnos de preparatorias públicas sin experiencia previa en materias de programación (población objetivo), en donde se buscó medir el efecto en el desempeño de los mini-tutores en dos temas básicos de programación (los tipos de datos y la asignación de valores a variables). La selección de las preparatorias no se realizó de manera aleatoria, sino por disponibilidad de los participantes. La población de ambas preparatorias es similar debido a que provienen del sector público y por no contar con experiencia previa en la programación. La selección de los participantes fue por invitación. Se crearon dos grupos con las siguientes características:

Grupo 1
Alumnos de 5º semestre de la preparatoria pública Lic. Jesús Reyes Heroles (Aguascalientes), sin antecedentes de materias de programación. La cantidad de participantes en este grupo fue de 27 alumnos (n=27). Las condiciones ambientales consistieron en que el estudio se llevó a cabo en un laboratorio de cómputo con conexión a internet. La participación del instructor consistió en indicar a los participantes dónde se encontraban las ligas al material didáctico, en este caso, páginas web estáticas. Este grupo no utilizó los mini-tutores, sirviendo como grupo de control.

Grupo 2
Alumnos entre 1º y 5º semestre, de la Preparatoria de la Universidad Autónoma de Aguascalientes, sin experiencia previa de programación.

Los alumnos participaron por invitación directa. Utilizaron computadoras y conexión a internet disponibles en sus domicilios. Para este grupo, la participación del instructor no fue presencial (a diferencia del grupo de control), pero se brindó asistencia en línea (*messenger*) a los participantes que lo requirieron. Las instrucciones generales de acceso al curso piloto se les proporcionaron mediante un texto enviado por correo electrónico. Es importante mencionar que se registró el uso de los mini-turores por parte de los participantes de este grupo por medio del historial de actividades de la plataforma *Moodle*, en la cual se registran fechas y horas de entrada a cada recurso, corroborando así su uso. No se incluyeron instrucciones de uso sobre los mini-tutores. El tamaño de la muestra fue de 19 (n=19).

Instrumento de evaluación.

Se utilizó un cuestionario de ocho preguntas de opción múltiple, con un tiempo límite para ambos grupos de 15 minutos, el cual evaluaba los conceptos de:

- Tipo de dato 'int' (entero)
- Declaración de variables int
- Tipos de datos char (carácter)
- Declaración de variables char

El diseño del instrumento se realizó tomando como referencia los conceptos explicados en el propio material didáctico en línea. El instrumento se aplicó igualmente en línea a ambos grupos (en forma de cuestionario de opción múltiple), por medio de la plataforma *Moodle*. Esto permitió registrar la duración en el llenado del cuestionario de cada participante y realizar análisis de correlación. En el caso del Grupo 1, el llenado fue al final de la sesión de laboratorio y en el caso del Grupo 2, al finalizar todas las sesiones y después de haber usado los mini-tutores. El instrumento no fue previamente validado para verificar el comportamiento normal del mismo, lo que puede considerarse como una limitación del estudio.

RESULTADOS

Se recolectaron 46 observaciones entre los dos grupos participantes. Se descartaron cuatro observaciones por contener datos inconsistentes[8]. Los resultados generales, indicados por la estadística descriptiva de las observaciones recolectadas para ambos grupos se muestran en la **Tabla 2.**

Tabla 2. Estadística descriptiva

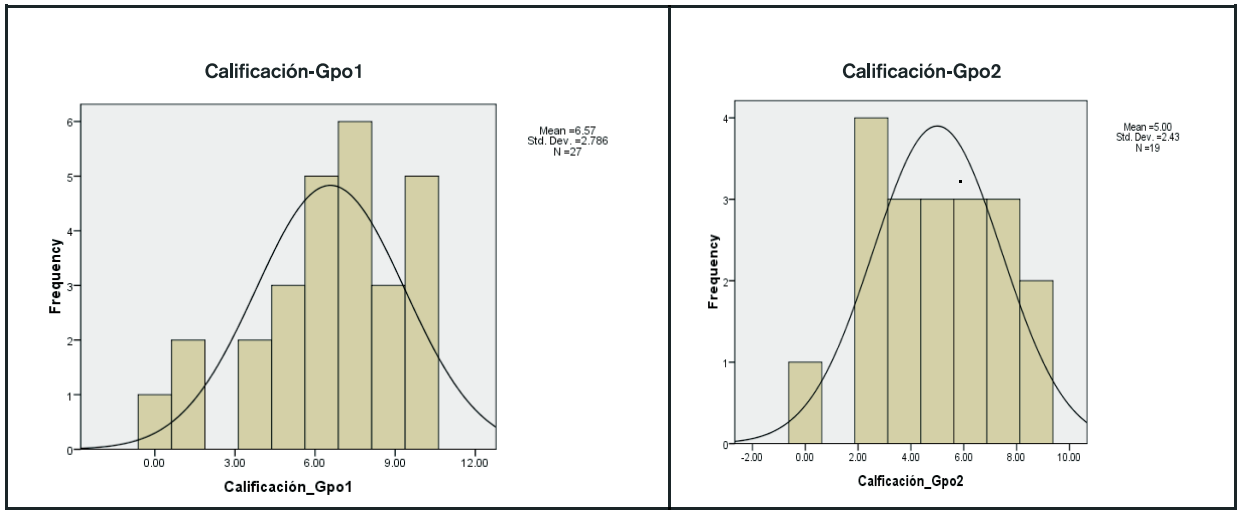
		Tiempo_Gpo1	Calificación_Gpo1	Tiempo_Gpo2	Calificación_Gpo2
N	Válidos	27	27	19	19
	Faltantes	0	0	8	8
	Media	3.6841	6.5741	3.6500	5.0000
	Mediana	3.0000	7.5000	3.0000	5.0000
	Moda	3.00	7.50	1.00[a]	2.50
	Desviación Std.	2.15519	2.78631	3.51185	2.42956
	Varianza	4.645	7.764	12.333	5.903
	Mínimo	.47	.00	.35	.00
	Máximo	9.00	10.00	15.00	8.75

a. Existen múltiples modas. Se muestra el valor más pequeño.

Se observa que la media de la calificación del Grupo 1 es mayor (6.57) a la del Grupo 2 (5.0). Es decir, que en promedio, la calificación del grupo experimental fue menor a la del grupo de control. En ambos casos, la desviación estándar es muy similar, lo que indica poca variabilidad y dispersión de los datos. La distribución de frecuencias de las calificaciones obtenidas por ambos grupos (ver **Tabla 3**), muestra que el Grupo 1 (grupo de control sin uso de tutores cognitivos) presenta el mayor porcentaje de frecuencias en el rango de 7.5 de calificación. Muestra, además, cinco observaciones con calificación de 10. El Grupo 2 (grupo experimental usando tutores cognitivos a distancia) presenta una distribución de porcentajes uniforme (15.8%) en los rangos de calificaciones de 3.75, 5, 6.25 y 7.5. Este grupo no tuvo calificaciones iguales a 10.

[8] Cuestionarios que no fueron terminados y que registraron una calificación de '0'.

Tabla 3. Distribuciones de frecuencias de las calificaciones obtenidas por ambos grupos



Calificación grupo 1

		Frec.	Porc.	Porcentaje Válido	Porcentaje Acumulado
Válidos	0	1	3.7%	3.7%	3.7%
	1.25	2	7.4%	7.4%	11.1%
	3.75	2	7.4%	7.4%	18.5%
	5	3	11.1%	11.1%	29.6%
	6.25	5	18.5%	18.5%	48.1%
	7.5	6	22.2%	22.2%	70.4%
	8.75	3	11.1%	11.1%	81.5%
	10	5	18.5%	18.5%	100.0%
	Total	27	100.0%	100.0%	

Calificación grupo 2

		Frec.	Porc.	Porcentaje Válido	Porcentaje Acumulado
Válidos	0	1	5.3%	5.3%	5.3%
	2.5	4	21.1%	21.1%	26.3%
	3.75	3	15.8%	15.8%	42.1%
	5	3	15.8%	15.8%	57.9%
	6.25	3	15.8%	15.8%	73.7%
	7.5	3	15.8%	15.8%	89.5%
	8.75	2	10.5%	10.5%	100.0%
	Total	19	100.0%	100.0%	

De acuerdo a los resultados anteriores, podría interpretarse que el Grupo 1 tuvo un mejor desempeño que el Grupo 2. Para verificar si existió una diferencia estadísticamente significativa en las calificaciones de ambos grupos, se corrió una prueba de análisis de varianza (ver **Tabla 4**), resultando un valor de 0.053, que proporciona indicios de que no existe diferencia en el desempeño de ambos grupos.

Tabla 4. Prueba ANOVA

Calificación					
	Suma de cuadrados	**df**	**Cuadrado de la media**	**F**	**Sig.**
Entre grupos	27.632	1	27.632	3.946	0.053
Dentro de los grupos	308.102	44	7.002		
Total	335.734	45			

Las pruebas de correlación no arrojan un valor significativo (.031) entre tiempo y calificación (ver **Tabla 5**). Es decir, no hay una correlación entre el desempeño de los participantes y el tiempo tomado en contestar la evaluación.

Tabla 5. Pruebas de correlación

		Tiempo	Calificación
Tiempo	Correlación Pearson	1.000	.031
	Sig. (2-tailed)		.838
	N	46	46
Calificación	Correlación Pearson	.031	1.000
	Sig. (2-tailed)	.838	
	N	46	46

DISCUSIÓN

A primera vista, los resultados obtenidos en cuanto al efecto en el aprendizaje pueden parecer desalentadores, pero creemos que estos son preliminares y es necesario realizar más estudios experimentales, considerando los resultados de otros estudios (Aleven *et al*., 2006; J. R. Anderson *et al*., 1995; Kumar, 2006), en donde se reportan efectos positivos. Además, puede agregarse que la experiencia obtenida sobre las variables experimentales resultó útil y será tomada en cuenta para futuros desarrollos.

CONCLUSIONES

Dados los resultados de las pruebas estadísticas realizadas, se considera que no permiten concluir aún si los tutores cognitivos para la enseñanza de la programación, desarrollados con las características permitidas por la herramienta CTAT utilizada, tienen un efecto positivo en el aprendizaje de estudiantes de nivel medio, debido a variaciones no previstas en las condiciones de las muestras.

Por otro lado, se considera que los objetivos secundarios del estudio si fueron alcanzados, al detectarse la naturaleza de dichas variaciones e identificar la manera en que éstas pueden influir en el aprendizaje. Se observó que el instructor –que no estuvo presente en el grupo experimental–, pudo haber influido en la confianza y facilidad de uso del material didáctico de los participantes del grupo de control. Es decir, que a la hora de asistir en las dudas sobre el uso del material (aún cuando este consistió sólo de páginas web estáticas) los participantes del grupo de control pudieron recurrir a un instructor humano que estaba disponible en el laboratorio donde se condujo el estudio. Esto mostró indicios de que el diseño del tutor cognitivo debe incluir suficientes características de 'ayuda' para el alumno. Asimismo, puede argumentarse que los participantes del grupo experimental pudieron haberse beneficiado al recibir la capacitación previa, ya sea presencial o a distancia. Es decir, aún cuando el diseño de la interfaz de usuario de los mini-tutores cognitivos es muy sencilla, posiblemente las instrucciones enviadas por escrito no fueron suficientes. En este sentido, los resultados sugieren que la aplicación de pruebas de usabilidad pueden brindar retroalimentación a los desarrolladores (e instructores) sobre el diseño de la interfaz de los mini-tutores cognitivos. Finalmente, algunas de las limitantes técnicas (por ejemplo, la necesidad de instalar componentes de software en las máquinas de algunos participantes) o demográficas encontradas (por ejemplo, la diferencia de semestres en una parte de los participantes del Grupo 2), creemos que han aportado una experiencia valiosa para la ejecución de subsecuentes experimentos.

Actualmente se discute la utilidad de los tutores cognitivos en la enseñanza de la programación.

INVESTIGACIÓN y CIENCIA
DE LA UNIVERSIDAD AUTÓNOMA DE AGUASCALIENTES
Número 48, (17-29) Enero-Abril 2010

REFERENCIAS

- ALEVEN, V.,*et al.,* Rapid authoring of intelligent tutors for real-world and experimental use. En: *8th International Conference on Intelligent Tutoring Systems (ITS 2006),* Berlín, 61-70, 2006.
- ALEVEN, V., *et al*., The Cognitive Tutor Authoring Tools (CTAT): Preliminary evaluation of efficiency gains. En: *7th Annual Intelligent Tutoring Systems Conference*, Maceio, Brazil, 2006.
- ANDERSON, J., An Integrated Theory of the mind. *Psychological Review*, 111(4), 1036-1060, 2004.
- ANDERSON, J. R., CORBET, A. y KOEDINGER, K. R., Cognitive Tutors, Lessons Learned. *The Journal of Learning Sciences*, 4(2), 167-207, 1995.
- ANDERSON, J., ACT, A simple Theory of Cognition. *American Psychologist*, 51(4), 355-365, 1996.
- BANDURA, A., Self-Efficacy Mechanism in Human Agency. *American Psychologist*, 37(2), 122-147, 1982.
- BAYMAN, P. A diagnosis of beginning programmers' misconceptions of BASIC programming statements. *Communications of the ACM* , 26(9), 677 - 679, 1983.
- BLOOM, B. S., The 2 sigma problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring. *Educational Researcher*, *13*(6), 4-16, 1984.
- BOYLE, T., The design and development of second generation learning objects, En: *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2006,* Chesapeake, VA., 2-12, 2006.
- BROSNAN, M., The impact of computer anxiety and self-efficacy upon performance. *Journal of Computer Assisted Learning*, 14, 223-234, 1998.
- BRUSILOVSKY, P., Mini-languages: A Way to Learn Programming Principles. *Education and Information Technologies*, 2(1), 65-83, 1998.
- BRUSILOVSKY, P. Intelligent learning environments for programming: The case for integration and adaptation. En: *Proceedings of AI-ED'95, 7th World Conference on Artificial Intelligence in Education,* Washington, DC., 1995.
- CHANSILP, K., Student's responses to the use of an interactive multimedia tool for learning computer programming. En: *World Conference on Educational Multimedia, Hypermedia & Telecommunications*, Proceedings of Ed-Media 2004, Norfolk, USA: L. Cantoni & C. McLoughlin (Eds)., 1739-1746, 2004.
- COOPER, S., Using animated 3D graphics to prepare novices for CS1. *Computer science education*, 13(1), 3-30. 2003.
- CORBETT, A., The predictive validity of student modeling in the ACT Programming Tutor. *Artificial Intelligence and Education: The Proceedings of AI-ED 93*. Charlottesville, VA: AACe.: In P. Brna, S. Ohlsson, & H. Pain (eds.), 1993.
- DIJKSTRA, E., On the Cruelty of Really Teaching Computer Science, *Communications of the ACM*, *32*(12), 1398-1404, 1989.
- DU BOULAY, B., Can we learn from ITSs? En *Intelligent Tutoring Systems*, Lecture notes in computer science Vol. 1839, Springer Berlin / Heidelberg, 9-17, 2000.
- DU BOULAY, J., Some difficulties of learning to program, *Lawrence Erlbaum Associates, Hillsdale*: 1989.
- FIXX, V. , Mental representations of programs by novices and experts. En: *Conference on Human Factors in Computing Systems*, *Proceedings of the SIGCHI conference on Human factors in computing systems,* Amsterdam, The Netherlands : ACM Press, NY, 74-79, 1993.
- HAGAN, D., & MARKHAM, S. Does it help to have some programming experience before beginning of a computer degree program? En: *Annual Joint Conference Integrating Technology into Computer Science Education* /Proceedings of the 5th annual SIGCSE/SIGCUE ITiCSEconference on Innovation and technology in computer science education, Helsinki, Finland, (25 - 28), 2000.
- HAMER, J., A lightweight visualizer for Java, En *Proceedings of the Third Program Visualization Workshop,* Warwick, UK, 54-61, 2004.
- HEGARTY, M., Constructing mental models from text and diagrams, *J. Mem. Lang.*, 32, 717–742, 1993.
- HEGGESTAD, E. The Predictive Validity of Self-Efficacy in Training Performance: Little More Than Past Performance. *Journal of Experimental Psychology,* 11(2), 84-97, 2005.
- HOLDEN, E., y WEEDEN, E., Prior Experience and New IT Students, *Issues in Informing Science and Information Technology*, 2, 189, 2005.
- HU, C., It's Mathematical, after all -the nature of learning computer programming. *Educational Information Technology*, 11, 83-92. doi: 10.1007/s10639-005-5714-4, 2006.
- JEHNG, J., A visualisation approach to learning the concept of recursion. *Journal of Computer Assisted Learning*, *15*, 279-290, 1999.
- JENKINS, T., On the difficulty of learning to program: En: *3rd annual LTSN-ICS Conference*. Loughborough University: LTSN Centre of information and computer sciences, 2002.

- KELLEHER, C., *Lowering the Barriers to Programming: A survey of programming environments and languages for novice programmers* 03-137, 2003.
- KOEDINGER, K., Opening the Door to Non-Programmers: Authoring Intelligent Tutor Behavior by Demonstration. En *Proceedings of 7th Annual Intelligent Tutoring Systems Conference*. Maceio, Brazil, 2004.
- KOEDINGER, K., y ANDERSON, J., Intelligent Tutoring Goes To School in the Big City. *International Journal of Artificial Intelligence in Education*, *8*, 30-43, 1997.
- KUJANSUU, E., Using program visualisation learning objects with non-major students with different study background. En: *Methods, Materials and Tools for Programming Education,* Tampere, Finland, 21-26, 2006.
- KUMAR, A. N., The Effect of Using Problem-Solving Tutors on the Self-Confidence of Students, *Presented at the 18th Workshop of the Psychology of Programming Interest Group*, University of Sussex, 275 - 283, 2006.
- MA, L., Investigating the viability of mental models held by novice programmers. *ACM SIGCSE Bulletin*, 39(1), 499-503, 2007.
- MATTHÍASDÓTTIR, Á., Usefulness of learning objects in computer science learning. En: *Methods, Materials and Tools for Programming Education,* Tampere, Finland: TAMPERE POLYTECHNIC - University of Applied Sciences Publications, 27-31, 2006.
- MCIVER, L., GRAIL: A Zeroth Programming Language. En: *International conference of computing in education (ICCE),* Chiba, Japan, 43-50, 1999.
- MCKEOWN, J., The use of a multimedia lesson to increase novice programmers' understanding of programming array concepts. *Journal of Computing Sciences in Colleges*, 19(4), 39 - 50, 2004.
- MILNE, I., & ROWE, G., Difficulties in Learning and Teaching Programming—Views of Students and Tutors. *Education and Information Technologies*, *7*(1), 55-66, 2002.
- MOISEY, S., Fulfilling the promise of learning objects. En: *In Handbook of Distance Education*, 1, 323, 2003.
- MURRAY, T., Authoring Intelligent Tutoring Systems: An Analysis of the State of the Art. *International Journal of Artificial Intelligence in Education*,10, 98-129, 1999.
- NAPS, T., A Java visualizer class: incorporating algorithm visualizations into students' programs. En: *Annual Joint Conference Integrating Technology into Computer Science Education , Proceedings of the 6th annual conference on the teaching of computing and the 3rd annual conference on Integrating technology into computer science education: Changing the delivery of computer science education.* Dublin City Univ., Ireland : ACM Press New York, NY, 181-184, 1998.
- NEVEN, F., Reusable learning objects: a survey of LOM-based repositories. En: *International Multimedia Conference, Proceedings of the tenth ACM international conference on Multimedia*, Juan-les-Pins, France, 291 - 294, 2002.
- RAMALINGAM, V., Self-Efficacy and mental models in learning to program. En: *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education,* Leeds, United Kingdom: ACM Press New York, NY, 171 - 175, 2004.
- RICHEY, R., *The Theoretical and Conceptual Bases of Instructional Design*. New York: Nichols, 1986.
- ROTHWELL, W. J., & KAZANAS, H., *Mastering the instructional design process: a systematic approach*. San Francisco: Jossey-Bass Publishers.
- SADLER-SMITH, E., & SMITH, P. J., Strategies for accommodating individuals' styles and preferences in flexible learning programmes. *British Journal of Educational Technology*, 35(4), 395-412, 2004.
- SPIRO, R., *et al.,* Cognitive Flexibility Theory: Hypermedia for Complex Learning, Adaptive Knowledge Application, and Experience Acceleration. *Educational Technology*, 43(5), 5-10, 2003.
- SPIRO, R., *et al.,* Cognitive Flexibility, Constructivism, and Hypertext: Random Access Instruction for Advanced Knowledge Acquisition in Ill-Structured Domains. *Educational Technology*, 24-33, 1991.
- SPIRO, R. J., y B. P. COLLINS, Reflections on a Post-Gutenberg Epistemology for Video Use in Ill-Structured Domains: Fostering Complex Learning and Cognitive Flexibility. En: *Video research in the learning sciences*, Lawrence Erlbaum Associates, 2007.
- THOMAS, L., *et al.,* Learning Styles and Performance in the introductory programming sequence. *ACM SIGCSE Bulletin*, 34(1), 33 - 37, 2002.
- WHITE, G., Standarized mathematics scores as a prerequisite for a first programming course. *Mathematics and Computer Education*, 37(1), 2003.
- WIEDENBECK, S., LABELLE, D. y KAIN, V. Factors affecting course outcomes in introductory programming. En *16th Workshop of the psychology of programming interest group*. Institute of Technology, Carlow, Ireland, 2004.
- WILEY, D., *The Instructional Use of Learning Objects*. Bloomington: Association for Educational Communications and Technology, 2000.
- Fotografías proporcionadas por el autor.