Compilador para traducir ubicaciones geográficas en instrucciones atómicas en una aplicación de aprendizaje ubicuo de programación

Denis Acosta[1], Margarita Alvarez[1], Elena Durán[1]

[1] Instituto de Investigaciones en Informática y Sistemas de Información
Facultad de Ciencias Exactas y Tecnologías – Universidad Nacional de Santiago del Estero
denislionelacosta@gmail.com, {alvarez,eduran}@unse.edu.ar

Resumen. El aprendizaje de los conceptos básicos de programación, como las estructuras de control, son consideradas difíciles debido a su complejidad y al nivel de abstracción requerido por el estudiante. Ante esta problemática, se ha considerado desafiante el desarrollo de una aplicación de aprendizaje ubicuo que asista a los estudiantes en el aprendizaje de estos conceptos. La aplicación, por medio de técnicas de realidad aumentada le muestra un objeto al estudiante, quien se debe desplazar en busca del mismo y como resultado la aplicación le genera un programa con las acciones realizadas. En este artículo se presenta la arquitectura de la aplicación de aprendizaje ubicuo, y en particular se desarrolla el módulo de compilación que permite traducir las coordenadas geográficas obtenidas por el celular del estudiante en instrucciones de un lenguaje de programación. Se muestran además las pruebas realizadas al módulo de compilación, que evidencian la viabilidad de realizar esta traducción.

Palabras Claves: Aprendizaje ubicuo, Arquitectura, Compiladores, Estructuras de control, Geolocalización.

1 Introducción

Por aprendizaje ubicuo (AU) puede entenderse, aquel que no se circunscribe a un contexto determinado ni viene condicionado por un espacio concreto, sino que la adquisición de nuevos conocimientos puede producirse en cualquier situación en la que se encuentre el sujeto, independientemente del momento y lugar [1]. El AU permite a los estudiantes acceder a todo tipo de información a partir de la interacción con los objetos que lo circundan, ya sean físicos o virtuales [2]. Favorece la colaboración, conecta los espacios formales, no formales e informales. También, resitúa la ubicación del aprendizaje, tanto dentro como fuera del aula y adapta los contenidos, presentaciones y actividades a las características de los estudiantes, y su contexto.

Por otra parte, las actividades de aprendizaje asociadas a la programación han sido reconocidas con alto grado de dificultad. Varios estudios han determinado que las causas que generan dicha problemática se relacionan con determinadas características que suceden dentro del aula y con ciertas habilidades cognitivas que son relevantes al

momento del aprendizaje de los fundamentos de programación. Entre ellas, están la capacidad de abstracción, una buena aptitud lógico-matemática y la facilidad para la resolución de problemas de orden algorítmico [3].

Ante la problemática planteada y las ventajas que proporciona el AU, se ha considerado desafiante el desarrollo de una aplicación que apoye el AU de los conceptos básicos de programación. La misma serviría para iniciar a los estudiantes en el aprendizaje de los conceptos básicos, mostrándole los algoritmos que realiza cuando se desplaza en un determinado ambiente, utilizando para ello el enfoque del microaprendizaje que propone aprendizajes con esfuerzos relativamente cortos y que consumen poco tiempo [4]. Así, la aplicación le muestra un objeto, empleando técnicas de realidad aumentada, que el estudiante debe obtener; el estudiante camina hasta el objeto y como resultado la aplicación le genera un programa con las acciones realizadas. Para ello, la arquitectura de la aplicación prevé un módulo de compilación que toma como entrada los datos de los sensores del celular del estudiante (GPS, Acelerómetro y Giroscopio) y los traduce a un programa escrito en el lenguaje diseñado a tal fin. Esto representó un importante desafío, ya que no se cuenta actualmente con una herramienta que haga posible esta traducción. Además, la interacción con la aplicación le provee una respuesta inmediata, favoreciendo al alumno la experimentación en el mundo real y permitiéndole formar rápidamente un modelo mental gracias a las respuestas que arroja la herramienta.

En este artículo se presenta la arquitectura de la aplicación de AU y el desarrollo del módulo de compilación, con las pruebas realizadas.

En las siguientes secciones se citan antecedentes de trabajos relacionados, se presenta la arquitectura de la aplicación y, el diseño y construcción del compilador. Finalmente, se muestran las pruebas realizadas y se expresan algunas conclusiones respecto al trabajo realizado, y las acciones a futuro.

2 Antecedentes

En esta sección se presentan antecedentes de aplicaciones de software para la enseñanza de los conceptos básicos de programación basada en la teoría de microaprendizaje y de realidad aumentada, y también algunos antecedentes de trabajos referidos al aprendizaje de cuestiones informáticas y computacionales mediante AU.

En [5] proporcionaron un medio flexible de AU con una aplicación móvil para que los estudiantes accedan al material. Los autores adoptaron el pensamiento computacional (PC) para ayudar a los estudiantes a desarrollar habilidades informáticas prácticas. Se eligieron tres clases de estudiantes de primer año para el estudio empírico. Se dividieron en tres grupos: dos grupos experimentales (grupo AU&PC y grupo PC) y un grupo de control. Según los resultados de este estudio, los estudiantes que recibieron el tratamiento de AU podrían tener habilidades informáticas significativamente mejores en el uso de PowerPoint y Word que los que no. Sin embargo, el tratamiento de la PC no resultó en un mejor desarrollo de las habilidades informáticas de los estudiantes en esta investigación.

Para alentar a los estudiantes universitarios a tomar cursos de programación de computadoras, en [6] se propone un sistema de apoyo al AU personalizado basado en múltiples fuentes de información. Este incluye una nueva técnica que integra problemas de aprendizaje de los estudiantes, nivel de conocimiento y estilo de aprendizaje para personalizar caminos de aprendizaje que le ofrecen al estudiante para su experiencia de aprendizaje de conceptos de programación.

En [7] se identifican los requisitos necesarios para apoyar el AU en las herramientas de software existentes para enseñar y aprender programación de computadoras a niños de entre 4 y 10 años de edad. Veintidós herramientas fueron analizadas y contrastadas con cinco características conocidas del AU: permanencia, accesibilidad, inmediatez, interactividad y consciencia del contexto. En una nota final, recomiendan agregar características de AU en las herramientas de programación para niños.

Boonbrahm et al. [8] desarrollaron una herramienta para aprender el flujo primario de comandos y las estructuras de control, incluyen la secuencia, las estructuras de selección e iteración. Con el empleo de la herramienta, los estudiantes construyen un diagrama de flujo del programa mediante el uso de marcadores de realidad aumentada. El software desarrollado captura la imagen del diagrama de flujo que el alumno ha construido, procesa el programa y muestra el resultado de la ejecución del comando. La herramienta identifica el comando, el valor de la variable y el operador booleano. Luego, el software simula el resultado de cada comando y así el alumno puede comprobar si la lógica del programa es correcta o no.

Skalka y Drlík [9] presentan un modelo conceptual de un sistema para mejorar las habilidades de programación basadas en la teoría del microaprendizaje en un entorno de aprendizaje móvil. El framework se basa en la combinación adecuada de objetos de contenido e interactivos. El objeto de contenido representa una unidad elemental de información. El objeto interactivo sigue cada unidad de contenido y verifican la comprensión de los estudiantes sobre el contenido educativo presentado.

En la revisión de antecedentes realizada no se encontraron trabajos que presenten experiencias de aplicación del AU en conceptos de programación, ni aplicaciones de software con esta finalidad.

Por otra parte, si bien existen aplicaciones que traducen coordenadas geográficas, no se han encontrado antecedentes de conversión de tales unidades a un lenguaje de programación como el que se propone en este trabajo.

3 Arquitectura de la Aplicación de Aprendizaje Ubicuo

3.1 Descripción de la Aplicación

La finalidad de la aplicación es que apoye al docente en la introducción de los conceptos de programación referidos a las estructuras de control básicas: secuencial, de selección y de iteración y, que le permita al estudiante su transición a la elaboración de algoritmos más complejos.

Para el diseño de la aplicación se ha considerado el enfoque del microaprendizaje. A tal fin, se ha dividido la enseñanza de las estructuras básicas de control en tres

niveles, de forma tal de introducir en cada nivel una estructura básica de control, incrementando el nivel de complejidad de los ejercicios que puede realizar el estudiante. En el nivel 1 se realizan ejercicios sencillos con el objetivo de aprender el concepto de instrucción y de secuencia de instrucciones. En el nivel 2 se incluyen ejercicios donde aparece la estructura condicional, dando lugar a instrucciones para girar a la derecha y a la izquierda de forma tal de evitar obstáculos. En el nivel 3 los ejercicios introducen el concepto de repetición.

3.2 Arquitectura de la Aplicación

La arquitectura de la aplicación (Fig. 1) describe los módulos y las interrelaciones entre ellos.

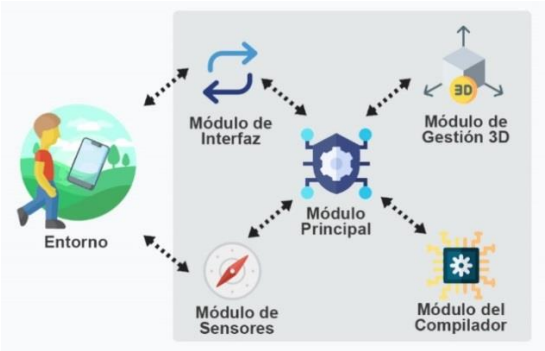


Fig. 1. Arquitectura de la aplicación

El módulo de interfaz se encarga de la comunicación del estudiante con la aplicación. Determina el nivel de complejidad para ofrecerle las actividades de aprendizaje de acuerdo con el nivel, y por último, muestra al estudiante el programa resultante.

El módulo de sensores, es el encargado de sensar el entorno del dispositivo y coordinar con el Módulo Principal. Cuando el estudiante encuentra un objeto virtual, interviene el Módulo de Gestión 3D enviando el objeto listo para ser procesado por la interfaz.

Una vez finalizada la interacción, el Módulo Principal envía todos los datos recolectados al compilador para que pueda traducirlos y otorgar como resultado el algoritmo (programa) obtenido en el camino hasta alcanzar el objeto. Por último, el módulo de interfaz muestra el programa resultante usando símbolos representativos.

Así, la aplicación determina el nivel de acuerdo con la interacción del estudiante con la aplicación y el aprendizaje alcanzado, y mediante el uso de la cámara del teléfono, captura el lugar y le inserta objetos en realidad aumentada.

El objetivo es que el estudiante llegue caminando hasta el lugar donde se encuentra el objeto virtual. La aplicación le devuelve al estudiante un programa formado por un conjunto de instrucciones atómicas con la secuencia de pasos que realizó hasta encontrar el objeto.

4 Diseño y construcción del módulo compilador

Para el diseño y desarrollo del compilador se siguió la metodología que se indica a continuación. Diseño y construcción de las fases de análisis: análisis léxico, análisis sintáctico y análisis semántico. Luego, se desarrolló la fase de síntesis: el generador de código.

La construcción se ha realizado con la herramienta ANTLR (ANother Tool for Language Recognition), un generador de analizadores para leer, procesar, ejecutar o traducir texto estructurado.

4.1 Lenguaje fuente y lenguaje objeto

Debido a que un compilador es un programa que lee un programa escrito en un lenguaje fuente y lo traduce a un programa equivalente en un lenguaje objeto, se define en este apartado ambos lenguajes.

Para garantizar un diseño completamente modular e independiente de la aplicación, se ha diseñado el lenguaje fuente con una sintaxis lo más simplificada posible, de forma tal que, el resto de los módulos, no requieran de una interfaz compleja para comunicarse con el compilador.

Los datos de entrada al compilador son cada una de las ubicaciones geográficas (latitudes y longitudes) obtenidas por los diferentes dispositivos capaces de sensar el contexto geográfico (GPS, Acelerómetro y Giroscopio). Por lo tanto, el lenguaje fuente está compuesto del nivel del usuario y el conjunto de puntos geográficos.

El programa resultante de la compilación está escrito en el lenguaje objeto. Se han tenido en cuenta las siguientes características para la selección de las instrucciones o sentencias del lenguaje objeto, considerando principalmente el aprendizaje de los estudiantes:

1- El lenguaje debe ser visible de forma natural, la mayoría de las operaciones realizadas por el actor son producto de cambios visibles en el micro mundo real [10].

2- El lenguaje debe ser atractivo y significativo para la categoría de estudiantes prevista [10].

3- El lenguaje debe tener la capacidad de añadir nuevas características (instrucciones) sin tener que ser reestructurado nuevamente (extensibilidad).

En este caso, el lenguaje objeto está formado por las siguientes instrucciones atómicas: avanzar (representa un paso hacia adelante realizado por el estudiante en su caminata hasta llegar al objeto), retroceder (representa un paso hacia atrás que el alumno realiza para cumplir el objetivo del ejercicio), girar izquierda y girar derecha (permiten desviarse del camino lineal), si hay objeto entonces (representa la bifurcación cuando se presenta o no un obstáculo) y repetir (con esta instrucción el estudiante visualiza una sentencia de control repetitiva).

4.2 Construcción de las fases del compilador

Para la construcción del analizador léxico se han definido los componentes léxicos y los patrones o expresiones regulares que generan dichos componentes. Los componentes léxicos definidos son: símbolos de puntuación y constante numérica.

Para la fase del Analizador Sintáctico se define la gramática libre de contexto mediante la notación BNF que es la entrada al generador ANTLR.

```
start:
  LLAVE_A
    CORCHETE_A nivel CORCHETE_C
    CORCHETE_A puntos CORCHETE_C
  LLAVE_C;
nivel: NUMERO;
puntos: tupla (COMA tupla)*;
tupla: PARENTESIS_A latlong COMA latlong PARENTESIS_C;

latlong: NUMERO PUNTO NUMERO;
```

En la fase de análisis semántico la verificación de tipos se realiza a las constantes numéricas para determinar si son de tipo latitud o de tipo longitud.

4.3 Generación del programa objeto

Para la generación del programa objeto se realiza el procedimiento descripto en Fig. 2 que aplica tres algoritmos: el de relación, de clasificación y el algoritmo rankeador.

Los algoritmos de relación y de clasificación, se han desarrollado para determinar la relación entre dos pares de puntos geográficos de forma tal de determinar si el estudiante ha realizado un paso hacia adelante, hacia atrás o a girado en alguna dirección.

Cada vez que se almacenan dos elementos consecutivos (proceso que ejecuta el analizador sintáctico cuando recorre el árbol de sintaxis abstracta), se realiza una comparación para determinar cuál es la relación existente entre ellos (vector) y se clasifica dicha relación en su equivalente a las instrucciones atómicas.

Dado que un vector en el plano es un par ordenado (a, b) de números reales perteneciente al espacio R^2 [11], se considera para este caso a "a" como el conjunto de latitudes y el componente "b" como el conjunto de longitudes. Las latitudes y longitudes son la entrada al Algoritmo de Relación y los resultados indican los distintos sentidos que puede tener un vector. A partir de los vectores obtenidos, se aplica el algoritmo de clasificación que establece cuál es la relación existente entre dichos vectores. Los resultados que pueden presentarse luego de clasificar un par de vectores se dividen en dos categorías: paralelos y ortogonales.

De acuerdo con [11] para determinar cuándo un par de vectores pertenece a un conjunto u otro se considera:

Paralelismo de vectores: sean u y v vectores no nulos de R^2, el vector u es paralelo al vector v si y solo si existe un escalar no nulo c tal que:

$$u = c.v. \quad (1)$$

Fig. 2. Proceso para la obtención del programa

Ortogonalidad: sean u y v vectores no nulo de R^2, se dice que u es ortogonal a v si y solo si el producto escalar de u y v es igual a cero,

$$c.v = 0 \quad (2)$$

Ángulos entre vectores: sean u y v vectores no nulos de R^2 y sea . el producto escalar, entonces existe y es único un α $[0, \pi]$ tal que:

$$\cos(\alpha) = \frac{u.v}{\|u\|\|v\|} \quad (3)$$

Para la clasificación de los vectores se aplican las siguientes reglas:

- Si ambos vectores calculados son paralelos se asume que el usuario avanzó en línea recta y para determinar si avanzó o retrocedió se considera el sentido de los vectores.
- Si los vectores son perpendiculares, se calcula la intersección entre ellos de forma tal de obtener el ángulo que los une. Si el ángulo es cercano a 90°, se asume que se trata de un cambio de dirección.
  - ✓ Si el vector superior tiene un sentido a la izquierda, se dirá que el usuario giro a la izquierda.
  - ✓ Si el vector superior tiene un sentido a la derecha, se dirá que el usuario giro a la derecha.

Finalmente, se mapea cada una de las reglas con la instrucción atómica equivalente (Tabla 1).

Tabla 1. Mapeo de las relaciones a instrucciones atómicas.

Relación	Sentido	Instrucción Atómica
Paralelos	Hacia arriba	AVANZAR
	Hacia abajo	RETROCEDER
Ortogonales	Izquierda	GIRAR IZQUIERDA
	Derecha	GIRAR DERECHA

Como última etapa, el compilador tendrá la tarea de adaptar el conjunto de instrucciones atómicas en base al nivel que posea el estudiante. Para ello, se aplica el

Algoritmo Rankeador, que toma como datos de entrada el conjunto de instrucciones atómicas y retorna como respuesta un conjunto adaptado en base a su nivel. Ésta adaptación produce una reducción de instrucciones o una agregación de nuevas, tales como: repetir y si hay obstáculo entonces. Para la sentencia condicional, el compilador al detectar la tupla (0,0) infiere que hay un obstáculo. Por ejemplo, si se tiene como entrada tres puntos: (-64.25139158964157,-27.80154927681387), (0,0), (-64.25133660435677,-27.801606219132715) la aplicación deduce que entre el punto 1 y el punto 3 existe un obstáculo.

5 Pruebas

Se realizaron pruebas para todos los niveles de complejidad. Se describe a continuación una prueba para cada tipo.

En la tabla 2 se puede observar una prueba realizada para el nivel de complejidad 1. Se observan el conjunto de latitudes y longitudes capturadas por la aplicación cuando el estudiante realiza el recorrido hacia el objeto (Fig. 4.a). También, se muestra la salida del módulo de compilación, programa resultante, y por último la interfaz que se le presenta al estudiante.

En la tabla 3 se puede observar una prueba realizada para el nivel de complejidad 2. En el recorrido que realiza el estudiante se puede encontrar con un obstáculo (Fig.4.b), por lo que el programa resultante introduce el concepto de la estructura condicional.

En la tabla 4 se puede observar una prueba realizada para el nivel de complejidad 3. El estudiante al realizar varias veces la misma operación, por ejemplo, avanzar(Fig. 4.c), el programa resultante contiene la sentencia repetir, introduciendo así, el concepto de estructura de control repetitiva.

Tabla 2. Prueba realizada para el nivel 1 de complejidad.

Entrada	Salida	Interfaz

{
[1]
[(-64.25157129764557,-27.801613336920475),
(-64.25148211419582,-27.80157715482788),
(-64.25138555467129,-27.8015510562618),
(-64.25134532153606,-27.801586645213977),
(-64.25128899514675,-27.80155639460536),
(-64.25122663378716,-27.80151843304538),
(-64.25117500126362,- 27.80154868366457)]
}
	{
	[AVANZAR,
	AVANZAR,
	GIRAR_DERECHA,
	AVANZAR,
	AVANZAR,
	GIRAR_DERECHA]
	}
		

Tabla 3. Prueba realizada para el nivel 2 de complejidad.

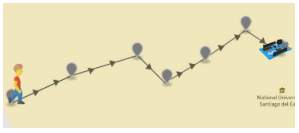
Entrada	Salida	Interfaz

{
[2]
[(-64.25157129764557,-27.80161570951628),
(-64.25148278474808,-27.801581306871892),
(-64.25139158964157,-27.80154927681387),
(0,0),
(-64.25133660435677,-27.801606219132715),
(-64.25126284360884,-27.801596728748322),
(-64.2511984705925,-27.801581306871892)]
}
	{
	[AVANZAR,
	AVANZAR,
	SI HAY OBJETO
	ENTONCES,
	GIRAR DERECHA,
	AVANZAR,
	GIRAR IZQUIERDA]
	}
		

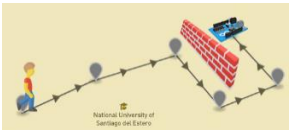
Tabla 4. Prueba realizada para el nivel 3 de complejidad.

Entrada	Salida	Interfaz

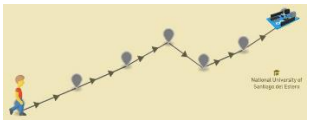
{
[3]
[(-64.25157062709332,-27.801612150622535),
(-64.2514868080616,-27.801581306871892),
(-64.25141505897044,-27.801553428858966),
(-64.25135537981987,-27.8015231782411),
(-64.25130508840084,-27.801555801456097),
(-64.25124809145927,-27.801533854930724),
(-64.25118573009968,-27.801497079662074)]
}
	{
	[REPETIR 3,
	AVANZAR,
	GIRAR DERECHA,
	REPETIR 1,
	AVANZAR]
	}
		



(a) Recorrido para el nivel 1.



(b) Recorrido para el nivel 2.



(c) Recorrido para el nivel 3.

Fig. 4. Recorrido del estudiante de las pruebas realizadas.

6 Conclusiones

En este trabajo se ha presentado la arquitectura de una aplicación de AU para la enseñanza de los conceptos básicos de programación en un intento por superar los problemas de aprendizaje de los conceptos que incluyen las estructuras de control.

También, hemos descripto el módulo de compilación que permite traducir las ubicaciones geográficas que se registran de las actividades que el estudiante realiza en

el entorno ubicuo y las pruebas realizadas al compilador. Estas pruebas ponen de manifiesto que es factible contar en una aplicación de AU que convierta acertadamente las coordenadas en instrucciones de un lenguaje, que luego los estudiantes podrán usar para programar. Este módulo de compilación resulta útil, novedoso y de amplia aplicación, ya que no existen antecedentes de este tipo de traducción de ubicación geográfica a instrucción de programa, que además se puede visualizar gráficamente, y que podrá ampliarse para generar instrucciones más complejas a posteriori.

Los trabajos futuros se orientan al desarrollo de los otros módulos que conforman la aplicación y a su evaluación en cursos de enseñanza de la programación.

De esta forma, tanto la geolocalización como el AU, pueden resultar en poderosas herramientas que posibiliten la generación de espacios de aprendizaje más ricos y motivadores.

Referencias

1. Burbules, N.C.: Ubiquitous Learning and the Future of Teaching. Encounters, 13, 3--14 (2012)
2. Villalustre Martínez, L., del Moral Pérez, M. E.: Geolocalización y realidad aumentada para un aprendizaje ubicuo en la formación inicial del profesorado. @tic revista d'innovació educativa, 21, 40--48 (2018)
3. Insuasti, J.: Problemas de enseñanza y aprendizaje de los fundamentos de programación. Revista educación y desarrollo social*, 10,* 234--246 (2016)
4. Salinas, J., y Marín, V. I.: Pasado, presente y futuro del microlearning como estrategia para el desarrollo profesional. In: Campus Virtuales, Vol. III, Num. 2, pp. 46--61. Huelva, España (2014)
5. Chia-Wen Tsai, Pei-Di Shen, Meng-Chuan Tsai & Wen-Yu Chen: Exploring the effects of web-mediated computational thinking on developing students' computing skills in a ubiquitous learning environment. Interactive Learning Environments, 25:6, 762--777 (2017)
6. Chookaew S., Wanichsan D., Hwang G-J., Panjaburee P.: Effects of a personalised ubiquitous learning support system on university students' learning performance and attitudes in computer-programming courses. In: International Journal of Mobile Learning and Organisation (2015)
7. Hosanee Y., Panchoo S.: The Analysis and the Need of Ubiquitous Learning to Engage Children in Coding. In: Fleming P., Lacquet B., Sanei S., Deb K., Jakobsson A. (eds) Smart and Sustainable Engineering for Next Generation Applications. ELECOM 2018. Lecture Notes in Electrical Engineering, vol 561. Springer, Cham (2019)
8. Boonbrahm, S., Boonbrahm, P., Kaewrat, C., Pengkaew, P., Khachorncharoenkul, P.: Teaching Fundamental Programming Using Augmented Reality. International Journal of Interactive Mobile Technologies (iJIM) (2019)
9. Skalka J., Drlík, M.: Conceptual Framework of Microlearning-based Training Mobile Application for Improving Programming Skills Interactive Mobile Communication Technologies and Learning. IMCL 2017. Advances in Intelligent Systems and Computing, vol 725. Springer, Cham (2017)
10. Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A., and Miller, P.: Mini-languages: A Way to Learn Programming Principles. Education and Information Technologies 2, pp. 65--83 (1997)
11. Kolman, B., Hill, D.: Algebra Lineal 8va Edición. Pearson Educación 216--224. (2006)