IX REUNION DE INTERCAMBIO DE EXPERIENCIAS EN ESTUDIOS SOBRE EDUCACION

NEWT, UNA HERRAMIENTA DE PROGRAMACION GRAFICA PARA LA ENSEÑANZA DEL PENSAMIENTO ALGORITMICO

Ing. Raúl V Ramírez Velarde
Departamento de Ciencias Computacionales
Campus Monterrey

Monterrey, N.L.

Agosto de 1991

Introducción

Es indudable que en el marco competitivo que se vive en la actualidad, se requiere que el profesionista tenga una adecuada formación que le permita utilizar los adelantos que ofrece la tecnología y poder encarar la competencia en el ejercicio de su profesión. Y dado que son las universidades las encargadas de proporcionar la mayor parte de dicha formación, son ellas las que tienen la mayor responsabilidad ante la sociedad de tomar la creación y asimilación de tecnología como un reto, y afrontarlo.

Un ejemplo muy claro de la forma en que las universidades están respondiendo, se encuentra en la enseñanza de computación, específicamente la programación. Para los alumnos de cualquier carrera profesional, por ejemplo en las ingenierías, el saber programar por lo menos en un lenguaje proporciona una herramienta indispensable para la aplicación de las diversas técnicas de análisis de sistemas, y diseño de equipo. Es por esto que todas las ingenierías contienen en sus primeros semestres materias curriculares de programación básica.

Sin embargo, muchos de los estudiantes que llegan a estos cursos no han tenido experiencia previa con las computadoras. Comprensiblemente, desconocen los procedimientos básicos para el manejo de las computadoras. Pero peor aún, se pueden sentir cohibidos por el miedo a quedar en ridículo o a no poder dominar la máquina, y pueden sentirse intimidados ante la simple perspectiva de descomponerla. Esto puede ser, para el alumno, una pastilla difícil de pasar.

Otro obstáculo que encuentran los estudiantes de programación es el énfasis que generalmente se da a la teoría, dejándose la práctica como una responsabilidad para el alumno. La resolución de problemas siempre es precedida por un proceso largo de instrucción formal en el salón donde al alumno se le enseña la sintaxis del lenguaje, la estructura de los programas y los comandos del compilador.

Así, cuando llega el momento de poner a prueba sus conocimientos, el programador novato tiene que enfrentar tres problemas al mismo tiempo: encontrar una solución al ejercicio, dominar la sintaxis del lenguaje y utilizar eficientemente el compilador, intérprete o ambiente de programación.

Es así como se pueden identificar dos problemas que tienen que afrontar los alumnos de computación: La falta de cultura computacional y el excesivo caudal de conocimientos que tienen que asimilar.

Este trabajo plantea una posible solución a ambos problemas a través de la utilización de programas educativos. Este tipo de programas pueden, de una forma amigable, introducir en los alumnos de lenguajes de programación básicos a los conceptos más simples sobre las computadoras y permiten desarrollar las estructuras mentales del pensamiento que necesitan para poder crear exitosamente programas.

Caracterización de los Problemas

Falta de Cultura Computacional

Esto es algo normal entre las personas que se enfrentan a un medio ambiente nuevo con problemas nuevos, como los alumnos de nuevo ingreso que entran de la preparatoria a hacer estudios profesionales. Este problema es quizá el más fácil de corregir, sin embargo, se requiere de una gran sensibilidad a las necesidades, temores y motivaciones de estos estudiantes.

Lo que se necesita es crear en los alumnos la suficiente confianza como para que se atrevan a explorar el ambiente computacional. Aquí se puede identificar la primera característica de los Programas Educativos: tienen que ser amables con el usuario y comunicativos. Es decir, que el programa educativo debe llevar un diálogo con el estudiante de tal forma que constantemente lo oriente en la utilización del sistema.

Además, los programas educativos deben crear la seguridad de que son herramientas y que deben utilizarse como tal. Esto es, que el programa debe ser confiable. Un programa educativo nunca debe incurrir en errores fatales o llevar a situaciones dónde no haya salida o, si esta realmente existe, es muy difícil de encontrar. El Programa Educativo aplicado al aprendizaje de las computadoras debe orientar.

La parte del Programa Educativo que se encarga de llevar este diálogo es lo que comúnmente se denomina como Interfase. La Interfase de Programa Educativo debe proveer un tipo de comunicación con el sistema que en todo momento oriente al estudiante sobre cuáles acciones puede tomar según las condiciones, orientarlo sobre esas condiciones, informar cuáles son los efectos de las acciones, evitar que se caiga en errores fatales y alentar la exploración.

La Carga Cognitiva

La Carga Cognitiva es la cantidad de conocimientos que se deben aprender para realizar una tarea. Los estudiantes de programación deben dominar la sintaxis del lenguaje, el manejo del editor y del compilador, y el manejo de la computadora, deben hacer un esfuerzo intelectual para diseñar un procedimiento que resuelva el problema que se le plantea y deben aprender a depurar su programa ¡ Todo al mismo tiempo! Esta claro que un estudiante de programación está simplemente cognitivamente sobrecargado. De aquí, otra forma en que un Programa Educativo puede ayudar al estudiante es disminuyendo la carga cognitiva.

El Pensamiento Algoritmico

No cabe duda que lo primero que debe aprender un alumno de lenguajes de programación es la lógica de programación. Es decir, el alumno debe aprender a pensar utilizando una técnica mental que le lleve a desarrollar un procedimiento de resolución de problemas aplicable a la utilización de computadoras. Esta técnica mental se denomina **pensamiento algoritmico**. Un algoritmo no es más

que una **sucesión de pasos relevantes que llevan a un fin**. Si el alumno no puede desarrollar las estructuras mentales que le permitan utilizar el pensamiento algoritmico, no importa el lenguaje que se intente enseñarle, jamás podrá crear programas.

¿Cómo se puede enseñar el pensamiento algoritmico? La herramienta que tradicionalente se ha venido utilizando en los cursos de computación para inculcar el pensamiento algoritmico en los estudiantes es el **diagrama de flujo**. El diagrama de flujo es una representación gráfica de un algoritmo. Un ejemplo típico se puede ver en la fig 1, que representa el diagrama de flujo del algoritmo que se utiliza para cambiar una llanta ponchada.

De esta figura se pueden identificar las dos características que hacen de los diagramas de flujo herramientas efectivas para la enseñanza del pensamiento algoritmico. Esta son: representación gráfica de las acciones de tal forma que es fácil identificar los diferentes tipos que se pueden llevar a cabo y representación gráfica del flujo de las acciones, que permite relacionar la ejecución de diferentes acciones dependiendo de la variación de las condiciones. Por ejemplo, si existe llanta de refacción se sustituye inmediatamente por la llanta descompuesta, si no existe, se tiene que vulcanizar la que se tiene. La figura 2 muestra un diagrama de flujo aplicado al algoritmo para encontrar las soluciones de una ecuación cuadrática.

Los Programas Educativos

Lo que este trabajo propone como solución para los problemas ya planteados es la utilización Programas educativos. Un Programa Educativo es un programa computacional que participa, como sustituto del maestro o como apoyo a aquél, en la tutoría de conocimientos y habilidades que se enseñen en alguna asignatura o disciplina del conocimiento.

Los objetivos que persigue un Programa educativo son los siguientes:

1. Reducir la carga cognitiva
2. Permitir la concentración en el aprendizaje de conceptos
3. Mostrar las relaciones entre estos conceptos
4. Presentar problemas realistas que sean retos intelectuales y ayudar en su solución
5. Propiciar la retención en el tiempo del conocimiento

Un Programa Educativo está constituido por dos elementos que ya se han mencionado:

1. La Instrucción Ayudada por Computadora
2. La Interfase Computacional

El Concepto de **Instrucción Ayudada por Computadora** (IAC) implica el diseño de material relacionado con un curso a impartir por computadora. Es decir, la IAC es la técnica de utilizar la Psicología, la Pedagogía, la Ingeniería de Software, la Inteligencia Artificial, etc. para producir la

instrucción. La **Interfase** es el medio a través del cual se imparte la instrucción, es decir, es el proceso a través del cual se lleva el diálogo Hombre-Máquina.

Al final, la IAC necesita de una Interfase correctamente diseñada para poder hacer llegar efectivamente el conocimiento a quien lo tenga que adquirir y asegurar el entendimiento y la retención. Se puede decir que la calidad de la IAC hace referencia a la **importancia** educacional del programa y la interfase afecta la **efectividad** y la **satisfacción** del estudiante.

Newt

Para ayudar a los alumnos a desarrollar el pensamiento algoritmico, se creó un sistema que facilita la resolución de problemas y la experimentación al mismo tiempo que introduce al alumno a la utilización de lenguajes de alto nivel. Este se sistema se llama Newt.

Newt es un programa que a través de una interfase gráfica permite la elaboración de un diagrama de flujo. Este diagrama de flujo se mapea a su correspondiente listado en un lenguaje de alto nivel. Sobre el diagrama de flujo, se pueden realizar operaciones de edición de tal forma que el alumno tiene la sensación de trabajar directamente sobre el algoritmo y no sobre un programa. Cuando el diagrama de flujo está listo, se puede ejecutar y observar qué instrucciones se están ejecutando.

El Nombre de Newt se debe a que la intención inicial en el diseño del programa era crear un sistema que a partir del diagrama de flujo se creara un listado en el lenguaje Pascal. De ahí que el programa debiera llevar el nombre de otro gran matemático Newton. Pero para nosotros es sólamente Newt.

Se diseñó el programa en forma de simulación interactiva. Esto es correspondiente a la parte instruccional del programa, donde se intenta llevar el conocimiento directamente a las estructuras más elevadas en la mente del estudiante. Aunque en su fase inicial no contiene ningún tutorial, contiene un complejo sistema de manejo de errores que proteje al usuario de cometer errores fatales al mismo tiempo que lo instruye en la forma de evitarlos.

Con respecto a la interfase, se diseñó Newt como un editor gráfico donde las opciones más comunes están al alcance de la mano constantemente y que alienta la experimentación. Los errores son detectados al instante y corregidos. Un programa realizado en Newt está siempre listo para correr, libre de errores.

La pantalla está organizada por medio de ventanas para que el alumno pueda ver la relación entre el diagrama de flujo, el listado que genera y la forma en que se ejecuta.

Por último debe está integrado con un rico menú de ejemplos que puedan ser cargados en línea y probados.

Metodología

Para poder desarrollar Newt, se llevó a cabo un profunda investigación sobre el diseño de Interfases gráficas y la Instrucción Ayudada por Computadora.

1. Instrucción Ayudada por Computadora

De esta investigación se encontró que el proceso de diseño de la IAC debe pasar por tres etapas:

1. Análisis
2. Desarrollo
3. Evaluación

Análisis

En la fase de análisis se deben realizar los siguientes:

Análisis de Necesidades. Se determinan los faltantes en desempeño, conocimiento o productividad. Una necesidad es la diferencia entre el estado actual (los conocimientos del estudiante antes del curso) y un estado deseado.

Análisis de Metas. Una meta es un desempeño o comportamiento específico. Las metas deben especificarse en función de variables y con valores específicos, por ejemplo, porcentaje de acierto, tiempo de retensión, tiempo de aprendizaje, etc.

Análisis de Tareas. Determina la secuencia de comportamientos que llevan a la terminación de una tarea y al cumplimiento de los objetivos.

Análisis de Contenido y Contexto. En el análisis de contenido se defienen los conceptos críticos de la instrucción, su organización jerárquica y sus relaciones. El análisis de contexto muestra la forma en que se utilizan los conceptos derivados del análisis de contenido.

Desarrollo

La fase de desarrollo se basa en un modelo creado por Robert Tennyson. Se muestra en al figura 3.

Evaluación

La evaluación debe realizarse junto con la de la Interfase puesto que no es posible ver los efectos de la instrucción sin ella.

2. Interfase

De igual manera, el desarrollo de la Interfase pasa por tres etapas:

1. Análisis
2. Desarrollo
3. Evaluación

Análisis

En la sección de análisis se identifican dos:

1. **Tareas**. Delimita la funcionabilidad del sistema y permite delinear las secuencias de instrucciones necesarias para accesar las funciones del sistema.
2. **Usuarios**. Permite definir las características del usuario y estructurar los comandos del programa de acuerdo a lo que se espera de él.

Desarrollo

Componentes del modelo de desarrollo de IAC	Metas Educativas				
	Adquisición del conocimienyo			Utilización del conocimiento	
Conocimiento	Declarativo	Procedural	Contextual	Complejidad Cognocitiva	Sistema Total
Objetivos del Aprendizaje	Información Verbal	Habilidad Intelectual	Habilidad Contextual	Estrategia Cognocitiva	Proceso Creativo
Tiempo de Instrucción	10%	20%	25%	30%	15%
Estrategia de Enseñanza	Exposición	Práctica	Resol. de Problemas	Complejidad y Dinámica	Autodirec-cionamiento

Fig. 3 Relación entre las estructuras mentales que componen el aprendizaje, el Modelo de Desarollo de IAC y las estrategias de enseñanza.

En la fase de desarrollo se identifican tres elementos importantes:

1. **Especificación Funcional**. Explica la estructura global del programa y su operación en términos de entradas, procesos y elementos de las salidas.
2. **Prototipos**. Proveen de una explicación detallada de los aspectos dinámicos e interactivos del programa, implementando en forma visual las funciones críticas del sistema con un mínimo esfuerzo.
3. **Pruebas Piloto**. Son pruebas con usuarios reales en las que se verifica de algunas funciones completas del programa estén correctas.

Evaluación

Se puede llevar de tres formas:

1. **Pruebas de Aceptación Mínimas**. Se verifica el cumplimiento mínimo de las especificaciones derivadas de la sección de análisis.
2. **Encuestas**. Es la forma más sencilla de evaluar y más detallada. Tienen la ventaja de que pueden abarcar un gran número de usuarios.
3. **Entrevistas y Discusiones**. El contacto directo con usuarios lleva a encontrar sugerencias muy particulares y muy constructivas. Las discusiones en grupo deja en claro que comentarios son parte del sentir general y requieren mayor atención.

Conclusión

Después de haber hecho una profunda investigación sobre el desarrollo de Instrucción Ayudada por Computadora e Interfases Gráficas se puede decir que se tiene suficiente teoría para desarrollar Programas Educativos con la suficiente capacidad para tomar un papel significativo en la educación. Es posible desarrollar, siguiendo una metodología adecuada, sistemas que pueden funcionar como cursos bases de la instrucción o como auxiliares en ella. Los Programas Educativos diseñados como curso base tendrán un papel proponderante en la instrucción sustituyendo totalmente o casi totalmente al maestro. Los Programas Educativos que sean diseñados como auxiliares, pueden tener un papel muy importante al demostrar conceptos que son difíciles de entender en el pizarrón utilizando las capacidades de simulación, graficación y animación de las computadoras modernas. También pueden fortalecer secciones de la instrucción que no pueden ser cubiertas extensivamente en el salón de clase por falta de tiempo o por fallas u omisiones en los conocimientos que se asume el alumno debe tener al iniciar el curso.

Bibliografía

Alessi S. M. "Computer Based Instruction". 1989.

Blaschke C. L. "Integrated Learning Systems/Instructional Networks: Current Uses and Trends". En Educational Technology. Noviembre 1990.

Dennis B. y Tsai C. "HyperCard in Educational Research: And Introduction and Case Study". En Educational Research. Febrero 1990.

Brown J. R. y Cunningham S. "Visualization in Higher Education". En Academic Computing. Marzo 1990.

Chabay R. W. y Sherwood B. A. "Computer Assisted Instruction and Intelligent Tutoring Systems". Hillsdale, NJ: Earlbaum. 1989.

Cunningham S. y Zimmerman W. "Visualization in Teaching and Learning Mathematics". Sufragado por Mathematical Association of America. 1990.

Ellson R. "Visualization at Work". En Academic Computing. Marzo 1990.

Foley J. D. y Wallace V. L. "The Art of Natural Graphic Man-Machine Conversation". En Proceeding of the IEEE. Abril 1974.

Forsythe, A. I. y Keenan, T. A. "Lenguajes de Diagramas de Flujo". Editorial Limusa, México. 1974

Hansen E. "The Role Of Interactive Video Technology in Higher Education: Case Study and Proposed Framework". En Educational Technology. Septiembre 1990.

Hunka S. "Designing Guidelines for CAI Authoring Systems". En Educational Technology. Noviembre 1989.

"A Right to Die? The Case of Dax Cowart". IBM Applications Brief. 1990.

Kearsley G. y Halley R. "Designing Interactive Software". Park Row Press. La Jolla, California. 1985.

Mitzel H. E. "Computer Based Education". En Encyclopedia of Educational Research. Volumen 1. The Free Press, N. Y. 1979.

Okey J. R. "Tools of Analisis in Instructional Development". En Educational Technology. Junio 1990.

Perez R. S. y Seidel R. J. "Using Artificial Intelligence in Education: Computer-Based Tools for Instructional Development". En Educational Technology. Marzo 1990.

Rubin T. "User Interfase Design for Computer Systems". Halsted Press. N.Y. 1988.

Schaefermeyer S. "Standards for Instructional Computing Software Design and Development". En Educational Technology. Junio 1990.

Shneiderman B. "Designing the User Interfase". Addison-Wesley Company Publishing Inc. 1987.

Smith P. E. "Some Learning and Instructional Theory Considerations for the Development of Computer Related Instructional Materials". En Educational Technology. Noviembre 1989.

Tennyson R. D. "A Proposed Cognitive Paradigm of Learning for Educational Technology. En Educational Technology. Junio 1990.

Tennyson R. D. "Integrated Instructionnal Design Theory: Advancements from Cognitive Science and Instructional Technology". En Educational Technology. Julio 1990.

Thornburg M. S. "Knowledge-Based Tutoring of Special Education Classification Concepts". En Educational Technology. Marzo 1990.

Whitney R. E. y Urquhart N. S. "Microcomputers in the Mathematical Science: Effects on Courses, Students, and Instructors". En Academic Computing. Marzo 1990.

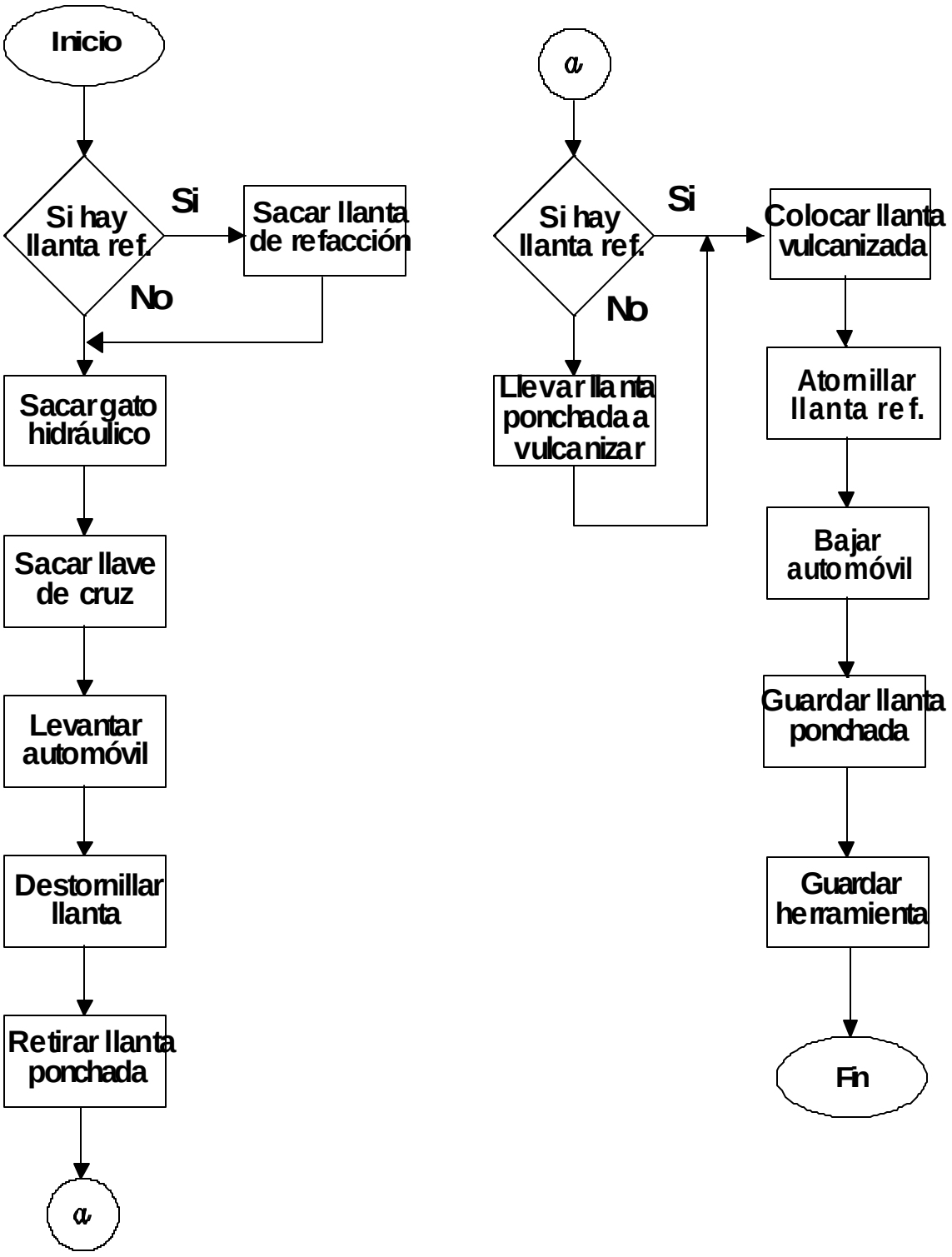

Figura 1 Diagrama de Flujo que representa el algoritmo para cambiar una llanta ponchada