# PANORAMA DE LA INFORMÁTICA EDUCATIVA: DE LOS MÉTODOS CONDUCTISTAS A LAS TEORÍAS COGNITIVAS

*por* ALFREDO FERNÁNDEZ-VALMAYOR CRESPO, CARMEN FERNÁNDEZ CHAMIZO y ANTONIO VAQUERO SÁNCHEZ

*Universidad de Castilla-La Mancha, Universidad Complutense de Madrid*

## 1. *Introducción*

En este artículo se hace una revisión crítica de las diferentes formas de utilización del computador en la enseñanza durante estas últimas décadas, para después tratar de establecer cuáles pueden ser sus perspectivas de futuro.

En primer lugar se hace una reflexión sobre la historia de la Informática Educativa, poniendo de relieve que las expectativas que inicialmente despertó la enseñanza asistida por computador no se corresponden con su situación actual, quizá dominada por la incertidumbre y una falta clara de orientación.

También se analizan las distintas formas de utilización del computador en la enseñanza, centrándose en aquellas en las que el computador es utilizado como tutor. Se tratan después las diferentes aproximaciones al diseño de programas tutoriales, considerando dos metodologías principalmente. En primer lugar aquellas metodologías basadas en la ingeniería del *software* tradicional y el paradigma conductista y, en segundo lugar, aquellas basadas en las técnicas de programación de la Inteligencia Artificial y en el paradigma cognitivo.

Se concluye con una enumeración de las bases sobre las que se

asienta el trabajo de investigación que se está desarrollando en el Departamento de Informática y Automática de la Universidad Complutense. Con dicha investigación se pretende avanzar en la que es una de las principales dificultades de la aplicación de los computadores a la educación: la representación, en forma operativa, del conocimiento contenido en el lenguaje natural.

## 2. *Evolución histórica. De los grandes sistemas a las microcomputadoras*

Durante la última década se han publicado numerosos trabajos en los que se analiza el impacto que ha producido en el mundo educativo la paulatina introducción del computador en la enseñanza a partir de los años 60. Entre estos trabajos merecen especial atención los de Suppes (1979), O'Shea & Self (1985), Carnoy, Daley & Loop (1987), Kurland & Kurland (1987), Trowbridge (1987) y Bork (1989). Otros trabajos en los que se cubre un amplio período de la utilización de los computadores en la enseñanza, pero que se limitan a una visión del mismo desde el campo de la Inteligencia Artificial, son los de Sleeman & Brown (1982), Wenger (1987) y Mandl & Lesgold (1988). Una introducción a la historia de la Informática Educativa puede verse en Vaquero y Fernández Chamizo (1987).

Toda la información recogida en estos trabajos puede ser analizada y valorada desde muchos puntos de vista. Nosotros creemos que, a grandes rasgos, es posible distinguir, en estos casi 30 años de historia de aplicaciones de la Informática a la Educación, dos épocas claramente diferenciadas aunque en parte superpuestas en el tiempo, finales de los 70 y comienzo de los 80. Por otra parte, también es posible notar un cierto matiz, no siempre reconocido, de fracaso al haber quedado los resultados de la enseñanza asistida por computador muy por debajo de sus expectativas iniciales.

En la primera época el objetivo es una enseñanza más efectiva a un coste menor. Para alcanzar este objetivo se utilizan grandes computadores controlando un elevado número de terminales y se utiliza el paradigma conductista, con diversas matizaciones, como principio unificador de todas las estrategias pedagógicas empleadas en el diseño del «*courseware*» o *software* educativo. Los proyectos PLATO (Bitzer *et al.*, 1962) y TICCIT (Merrill, 1974) son seguramente los ejemplos más representativos de esta forma de concebir la enseñanza asistida por computadora.

La segunda época comienza antes de que la primera alcance su término, contribuyendo quizá de este modo a disimular el relativo fracaso de proyectos como los anteriormente citados.

Si la primera época puede caracterizarse por los grandes proyectos, la segunda podríamos caracterizarla por la descentralización del esfuerzo, que se hace posible tanto por la aparición de la computadora personal como por la diversificación de los paradigmas psico-pedagógicos, muchos de ellos claramente ligados a los planteamientos de la Psicología Cognitiva y de la Inteligencia Artificial (O'Shea & Self, 1985).

Si consideramos individualmente cada uno de los trabajos citados al comienzo de este apartado, veremos que la perspectiva que se nos ofrece en ellos es una visión interna, en la que investigadores y profesores analizan el impacto de la Informática en la educación desde dentro del propio sistema educativo, sin prestar quizá una atención suficiente a factores externos al mundo educativo, como son los factores sociales y tecnológicos, que en nuestra opinión han terminado por convertirse en las principales fuerzas conductoras de los acontecimientos.

Un ejemplo de esta situación nos la ofrece el trabajo publicado por Suppes en 1979. En él, la discusión sobre la influencia del CAI *(Computer Aided Instruction)* en la enseñanza se hace sobre la base de los resultados de casi doce proyectos, todos ellos funcionando en grandes computadores de tiempo compartido, mientras que en los años siguientes quedó patente que la introducción del microcomputador (el Apple II se comercializó a partir de 1977) estaba provocando un cambio radical en la forma de concebir la producción y utilización del CAI, cambio que pasaba completamente desapercibido en el estudio de Suppes.

Por otra parte, los puntos de vista y posiciones defendidas por investigadores desarrolladores de *software* y profesores, no suelen ser coincidentes, estando en algunos casos en franca contradicción (Holden, 1989). En muchos casos, el creciente número de computadores en los centros de enseñanza, en todos los niveles de la misma, se interpreta como una prueba de que nos movemos hacia un modelo de sistema educativo en el cual el computador juega un papel muy importante. En muchos de estos estudios se acepta, al menos de forma implícita, la hipótesis de que esta presencia creciente del computador en la educación se debe al éxito de las diferentes metodologías y proyectos que, a lo largo de estas tres últimas décadas, se han ido sucediendo con el fin de aumentar la calidad de la educación mediante un uso intensivo del computador.

En nuestra opinión, en estos estudios, no se considera suficientemente la hipótesis inversa, es decir, que la creciente presencia de computado-

res en los centros de enseñanza es sólo una consecuencia del éxito del computador en toda la sociedad y que precisamente es en el entorno educativo donde está siendo más difícil lograr la utilización de los computadores de forma efectiva, a pesar de que su utilización en este campo había comenzado antes que en ningún otro sector de la sociedad.

En uno de estos estudios, una amplia monografía preparada por la «School of Education» de la Universidad de Stanford bajo los auspicios de la UNESCO (Carnoy *et al.,* 1987), se cita la opinión de diversos investigadores para los que, al igual que sucedió antes con otras tecnologías, la utilización del computador en la enseñanza está llamada a ser un fracaso. Estos autores no niegan la potencialidad del computador para mejorar la calidad de la enseñanza pero diversas causas, entre las que ponen en primer lugar la poca calidad del *software,* conducirán en su opinión a este resultado. Esta opinión sobre la falta de calidad de casi todo el *software* educativo es sustentada por la gran mayoría de los autores.

No obstante, en la misma monografía se informa de los resultados de más de 200 estudios y evaluaciones sobre la efectividad de la enseñanza mediada por computador («Computer-Mediated Learning», CML) con un resultado claramente positivo, sobre todo para la enseñanza asistida por computador (CAI). Estos resultados pasan de ser positivos a muy positivos si la población de muestra se restringe a los estudiantes con peores calificaciones.

Con respecto al tipo de CAI, ejercicios («drill & practice») o programas tutoriales, los resultados son diversos, apareciendo en unos estudios como más efectivo el primero y en otros el segundo. Por otra parte, la escasez o el poco tiempo que llevan utilizándose los tutores inteligentes (*Intelligent Tutoring System* ITS, o, *Intelligent Computer Aided Instruction* ICAI) hace que los resultados obtenidos con ellos no sean representativos. Sin embargo, las evaluaciones puntuales realizadas con aquellos pocos ITS que, como el «Geometric Tutor» de Anderson, están disponibles en algunos centros de enseñanza no parecen diferir esencialmente de los resultados obtenidos con el CAI tradicional (Kafai, 1989).

Con respecto a la utilización del computador como herramienta en la enseñanza por descubrimiento, los resultados obtenidos en las evaluaciones no son tan concluyentes. Las evaluaciones llevadas a cabo con LOGO en Estados Unidos arrojan resultados desiguales, siendo quizá el aspecto más negativo de estos resultados el que, en los estudios realizados durante períodos de tiempo largos (dos años), no se haya

podido confirmar la hipótesis de S. Papert de que el uso continuado de un lenguaje de computador como LOGO crearía nuevas habilidades cognitivas en los niños.

Un primer análisis de todos estos resultados nos lleva a la conclusión de que la creciente utilización del computador en la enseñanza es un fenómeno complejo y de largo alcance, cuyos efectos deben ser considerados a largo plazo y en el marco de la revolución tecnológica que está sufriendo toda la sociedad. No es de extrañar, por tanto, que no exista una dirección clara de progreso y que tanto los resultados de experiencias concretas como los de las evaluaciones globales, realizadas por distintos grupos de investigadores, sean en muchos casos totalmente dispares.

Un hecho característico es que prácticamente en ninguno de los estudios mencionados se duda de la potencialidad del computador como un instrumento «singular» para favorecer el aprendizaje. Las discrepancias surgen con respecto al modelo de enseñanza y al papel que en ella deba asumir el computador. En 1968 Patrik Suppes escribía que la introducción del computador en la enseñanza estaba llamada a provocar una revolución en la educación, comparable a la que supuso la generalización de los libros en las escuelas del siglo XVIII (Suppes *et al.*, 1968). En años posteriores, esta metáfora u otras similares continuaron utilizándose para describir los efectos que el computador iba a producir en la educación (Bork, 1979; Sanders, 1981), pero es evidente que nada de esto ha sucedido.

Desde nuestro punto de vista la reflexión que hace Suppes es muy acertada, pero creemos que, para conseguir que el computador tenga un papel en el proceso educativo equiparable al que tienen en la actualidad los libros y el lenguaje escrito, debemos dotar a los computadores de programas capaces de adquirir y transmitir el conocimiento contenido en un texto. Dicho de otro modo, el esfuerzo informático en el campo de la enseñanza asistida por computadora debe medirse en términos del conocimiento que los programas son capaces de representar, adquirir y transmitir.

### 3. *Tipos de* software *educativo y modelos de aprendizaje*

Sin ánimo de ser exhaustivos, podemos considerar que los programas utilizados en educación están dentro de una de estas tres categorías:

1. Programas CAI, escritos utilizando las técnicas de programa-

ción y de la ingeniería del *software* tradicionales. En estos programas la actividad del alumno está controlada, en mayor o menor medida, por el computador y la estrategia pedagógica utilizada es del tipo «ejercicios» o tutorial. Su base científica está constituida principalmente por el paradigma conductista, aunque en la actualidad se puede apreciar en estos programas la utilización de ideas procedentes de la Psicología Evolutiva y Cognitiva (Bork, 1986).

2. Simulaciones y micromundos. Las simulaciones didácticas de procesos físicos y biológicos y el lenguaje de programación LOGO son los ejemplos más característicos de este tipo de programas. En ellos el computador se utiliza para crear un entorno simulado, un micromundo, sometido a sus propias leyes, que el alumno debe descubrir o aprender a utilizar, mediante la exploración y la experimentación dentro de ese entorno. En estos programas el computador no controla la actividad del alumno. Su función es la de ser una herramienta que potencie el desarrollo de las habilidades cognitivas del alumno.

Para los investigadores que defienden esta aproximación, la revolución no está basada tanto en la capacidad del computador para procesar conocimiento como en los cambios que la utilización del computador puede provocar en la mente del alumno. Este enfoque, basado fundamentalmente en las teorías de J. Piaget y en la Psicología Cognitiva, es defendido brillantemente por S. Papert (1980) y muchos de sus colegas del laboratorio de Inteligencia Artificial del MIT.

3. Programas ICAI o ITS, basados en técnicas de Inteligencia Artificial. Suelen tener la forma de tutoriales en los que el alumno puede tomar la iniciativa. Además, la comunicación con el computador suele realizarse en un subconjunto más o menos amplio del lenguaje natural. Las diferencias más profundas, con respecto a los programas mencionados en el primer grupo, son debidas a la forma en que se concibe su diseño. Mientras que un programa CAI trata de inducir en el alumno la respuesta correcta, mediante una serie de estímulos que han sido cuidadosamente planificados, los procesos de un ITS pretenden ser capaces de simular alguna de las capacidades cognitivas del alumno y utilizar los resultados de esta simulación como base de las decisiones pedagógicas a tomar.

En la práctica un programa de enseñanza puede no pertenecer a una sola de estas tres categorías sino ser el resultado de utilizar varias de ellas. Por ejemplo, muchos autores consideran que una buena estrategia pedagógica es que un programa tutorial incluya algunas simulaciones de los procesos que se quieren explicar al alumno. También existe actualmente la tendencia a incluir secuencias dirigidas en los

micromundos, bien directamente o bien como secuencias de ayuda, para hacerlos más eficientes en cuanto al tiempo empleado y tipo de alumnos que puedan sacar partido del sistema (Mandl & Lesgold, 1988).

En este artículo no vamos a tratar en detalle de los programas incluidos dentro de la segunda categoría, micromundos y simulaciones. La razón de hacerlo así es que el conocimiento necesario para implementar estos programas es siempre un conocimiento de marcado carácter procedimental y específico, diferente del que nosotros consideramos que es el problema básico de la informática educativa: la búsqueda de formalismos que nos permitan representar en la máquina de forma computacionalmente tratable, los conocimientos que normalmente representamos y transmitimos mediante el lenguaje natural. Por ello, vamos a centrar nuestro estudio en los programas tutoriales, programas en los que esta forma de representación del conocimiento, tanto del conocimiento pedagógico como del conocimiento sobre un dominio específico, ocupa una posición predominante.

## *4. Programas tutoriales: el paradigma conductista*

Una gran parte del *software* educativo desarrollado hasta ahora corresponde al tipo denominado «tutorial». Este tipo de programas intenta reproducir la forma de enseñanza que está basada en el diálogo con un tutor. En estos diálogos «socráticos», el tutor, a base de preguntas, va provocando la reflexión en el alumno y haciendo que éste construya, por sí mismo, las respuestas correctas y, en definitiva, que aprenda los conceptos objeto de estudio.

De momento vamos a dejar a un lado las técnicas de Inteligencia Artificial utilizadas para desarrollar «tutores inteligentes» y vamos a centrarnos en los programas tutoriales desarrollados dentro del CAI tradicional.

Estos programas están basados, en mayor o menor medida, en el paradigma conductista y en este marco ocupa un lugar relevante la hipótesis de que en el individuo es posible identificar dimensiones de variación intelectual y diseñar tests específicos para medir su habilidad a lo largo de estas dimensiones. De este modo, los tests constituyen la forma de detectar el grado de conocimiento que tiene el alumno sobre una materia, utilizándose esta información como base para decidir cuál será la estrategia de enseñanza más adecuada a seguir.

Un esquema de este tipo es fundamentalmente un esquema determinista, con secuencias, entradas y salidas claramente establecidas, por lo que para su producción y diseño se adaptan perfectamente las técnicas de programación e ingeniería del *software* tradicionales (Kurtz & Bork, 1981; Bork, 1986).

Se puede considerar que este tipo de *software* tiene su origen a finales de la década de los 50 en las universidades americanas, cuando los investigadores comienzan a poder disponer de computadores y el conductismo y su aplicación pedagógica, la enseñanza programada, se encuentran en pleno auge. No es por tanto de extrañar que sea el conductismo el fundamento científico de los primeros programas de enseñanza por computador, que en principio no serían más que la versión computarizada de las máquinas de enseñanza programada (Coulson, 1962).

No obstante, diferentes investigadores, como Bitzer (1962), Crowder (1962) y Suppes (1960), entre otros, se percatan de que el computador permite implementar esquemas de aprendizaje mucho más complejos y mejor adaptados a la situación escolar que los esquemas de condicionamiento operante de las máquinas de enseñanza programada (Skinner, 1970), como son el «aprendizaje verbal significativo» (Ausubel, 1963, 1968) y el «Mastery Learning» (Bloom, 1976), que desde entonces han sido el principal soporte pedagógico y psicológico de la enseñanza asistida por computador (CAI).

Aunque dentro del CAI existen importantes diferencias en cuanto a las estrategias de producción de programas y en cuanto al contexto en que éstos deben utilizarse, la mayoría de los investigadores coinciden en que el computador es el instrumento ideal para alcanzar una serie de objetivos pedagógicos fundamentales como son:

— Tomar en cuenta las diferencias individuales entre alumnos, a un nivel más profundo de lo que hasta ahora había sido posible.

— Ofrecer atención inmediata a las respuestas de los alumnos.

— Sustituir al profesor en las tareas más rutinarias, permitiéndole dedicar más tiempo al alumno individual.

También desde un principio, aunque con menos énfasis, se consideraron los inconvenientes que puede plantear la introducción del computador en las escuelas. Por ejemplo en el informe del Stanford's Aritmetic Program (Suppes *et al.,* 1968) se señalaban los siguientes:

— Falta de fiabilidad de las máquinas.

— Que los programas educativos constituyan en sí mismos un curriculum sin interés, trivial y excesivamente simplificado.

— Privación de estímulo. ¿Puede el computador producir un ambiente suficientemente rico en estímulos? Para Suppes, en 1968, este podría ser un problema importante en el futuro, cuando la cantidad de tiempo al día que pasara el estudiante frente al terminal fuese una fracción importante del tiempo total que dedica al aprendizaje.

— El coste.

Es interesante hacer notar que más de 20 años después, objetivos y problemas continúan siendo básicamente los mismos, exceptuando el problema de la fiabilidad de las máquinas, que hoy en día está claramente resuelto y el problema del coste, que en principio se consideraba como algo fundamentalmente ligado al precio de las máquinas y hoy como algo debido principalmente al diseño e implementación de los programas.

El «Mastery Learning» (ML) es quizá el marco teórico más elaborado dentro del enfoque tradicional de la enseñanza asistida por computador (CAI). El ML se apoya en la idea de que, bajo las condiciones adecuadas, la mayoría de los estudiantes pueden aprender bien lo que los profesores quieren que aprendan y que, de este modo, la educación puede ser mejorada sustancialmente.

El ML está más relacionado con el conductismo que con las teorías cognitivas, aunque ha sido influido por éstas. Su punto de partida es la materia que hay que «grabar» en la mente del estudiante. Esta materia se divide en componentes atómicos, tales como ejercicios y colecciones lógicas de los mismos, que se agrupan en unidades.

Los alumnos, solos o en grupos, deben trabajar a lo largo de estas unidades de una forma organizada, a su propio ritmo, y deben dominar perfectamente («master») un determinado porcentaje de cada unidad (típicamente el 80 por ciento) antes de continuar con las unidades siguientes de la secuencia.

Generalmente se acepta que Benjamín S. Bloom es el teórico más importante del ML. A la idea conductista del refuerzo de las respuestas deseadas, Bloom añadió el «feedback» correctivo en el caso de las respuestas no deseadas. Además añade el concepto de «tiempo» necesario para que el estudiante llegue a dominar el objetivo, que puede variar notablemente en función del estudiante individual considerado.

El núcleo de ML es el diagnóstico y la corrección de errores en las fases de test y re-enseñanza, lo que impide que los errores de aprendizaje se acumulen. Estos procedimientos también dan más tiempo y

oportunidad de aprender a los más lentos. El profesor está obligado a enseñar y ayudar al alumno hasta que éste sea capaz de dominar un porcentaje elevado de la materia.

### *4.1. Fases en el desarrollo de un programa tutorial tradicional y herramientas de ayuda*

Aunque existen diversas metodologías a la hora de desarrollar programas tutoriales, generalmente se distinguen tres etapas:

— Diseño.

— Producción.

— Evaluación y perfeccionamiento.

La etapa de diseño corresponde al proceso de planificación del material didáctico y es, sin duda, la etapa más crítica. Esta fase está formada en realidad por varias subetapas. Algunos autores distinguen entre el «diseño pedagógico» y el «diseño de pantalla», mientras que otros consideran ambos diseños simultáneamente. Lo que sí es bastante general en todas las metodologías es la existencia de dos pasos: diseño pedagógico inicial, o diseño conceptual, y diseño pedagógico detallado.

La etapa de diseño, normalmente, es realizada por un equipo de profesionales, entre los que suele haber, al menos, un profesor con gran experiencia en la materia a impartir, un experto en investigación relacionada con dicha materia, un experto en enseñanza con nuevas teconologías y un experto en Informática. En general, se recomienda un número de 3 a 5 personas, aunque algunos autores proponen un grupo más numeroso (5-10 personas) para el diseño pedagógico inicial.

El diseño conceptual requiere más experiencia en el área de la materia a impartir, en teoría del aprendizaje y en estrategias pedagógicas y motivación, mientras que el diseño detallado requiere un mayor conocimiento de Informática y de las posibilidades específicas de las computadoras que se van a utilizar.

En el diseño inicial se establecen las características globales del *software* educativo que se va a desarrollar, los puntos que debe cubrir, los alumnos a los que va dirigido, los enfoques que se van a utilizar para introducir los temas, etc. El diseño inicial es, por tanto, equivalente a la propuesta inicial en un proyecto de investigación y constituye la base en la que se apoya el diseño detallado.

En la fase de diseño detallado, se establecen las características de todas las secuencias de interacción entre el programa y el alumno. En esta etapa se fijan con precisión los objetivos y subobjetivos de cada

parte del programa educativo, los prerrequisitos, la información que va a ser presentada al alumno, las preguntas que se le van a formular, las respuestas correctas, las respuestas incorrectas esperadas y la acción que el programa debe realizar para cada una de ellas, etc. Además, si no existe una fase separada de diseño de pantalla, en esta fase se especificarán todos los detalles relativos a la forma de presentación de información al alumno en la pantalla.

Durante la fase de diseño se suelen utilizar formularios o diagramas para representar las decisiones adoptadas. El formalismo de representación depende de la metodología utilizada, pero, usualmente, el producto final de la etapa de diseño es un «guión» que servirá como base para la etapa de producción.

La etapa de producción es una etapa puramente informática, consistente en el desarrollo de un producto *software.* Como tal incluirá todas las fases típicas del ciclo de vida del *software:* análisis, diseño, codificación y prueba. El guión producido por la etapa de diseño pedagógico servirá aquí como documento de especificación de requerimientos.

Finalmente, tenemos la etapa de evaluación y perfeccionamiento. Hay que resaltar aquí que se trata de una evaluación del material desde el punto de vista pedagógico. La parte de prueba puramente informática ya se habrá realizado en la fase de producción.

Algunos autores distinguen dos tipos de evaluaciones: formativa y sumativa. La evaluación formativa se realiza cuando el material no está aún terminado. El *software* educativo desarrollado se prueba con muestras de alumnos, almacenando sus respuestas, realizando exámenes previos y posteriores a la utilización del material, registrando en vídeo algunas sesiones de utilización del programa, etc. Con los resultados de todas estas pruebas el *software* se va perfeccionando hasta que finalmente se da por terminada esta fase y se distribuyen los programas educativos para su utilización en los centros de enseñanza.

Por el contrario, la evaluación sumativa se hace cuando un programa educativo lleva ya varios años en funcionamiento en varios centros educativos y se quieren conocer sus efectos sobre el aprendizaje de la materia en cuestión.

Todas estas fases del desarrollo de un programa tutorial no son, en realidad, fases totalmente separadas. Al final de cada fase, se raliza una revisión, cuyos resultados pueden provocar la vuelta a la etapa o etapas anteriores. De esta forma, se produce una realimentación continua entre las distintas fases.

Se han desarrollado diversas herramientas informáticas de ayuda al desarrollo de *software* educativo. Ahora bien, la práctica totalidad de estas herramientas corresponde a la etapa de producción, por lo que las etapas de diseño y evaluación suelen realizarse a mano. Vamos a ver, a continuación, las herramientas de ayuda a la producción que se utilizan actualmente: los lenguajes de programación de propósito general, los lenguajes de autor y los sistemas de autor.

Cualquier lenguaje de programación es válido, en principio, para desarrollar *software* educativo. Entre los lenguajes de programación de propósito general que han sido utilizados en este área podemos citar FORTRAN, APL, BASIC y Pascal. Ahora bien, la utilización de un lenguaje de programación exige un profundo conocimiento del mismo, así como ciertos conocimientos de Informática. Por esta razón, o bien los profesores del equipo de diseño se ven obligados a aprender un lenguaje de programación, o bien es necesario recurrir a un equipo de programadores.

En el primer caso, los profesores-programadores tienden a centrarse más en los problemas informáticos que en los objetivos pedagógicos que persiguen. Además, al no disponer de una gran experiencia informática, puede que no obtengan el máximo rendimiento de la computadora y que sus programas sean lentos o utilicen gran cantidad de memoria.

En el segundo caso, los profesores no necesitan aprender a programar pero, si la comunicación con el equipo de programadores no es buena, los programas desarrollados no cumplirán los objetivos previstos por los profesores. También en este caso la falta de conocimientos informáticos de los profesores puede ser contraproducente, ya que puede conducir a una sobreestimación o subestimación de las posibilidades de la computadora.

En cualquier caso, el tiempo de desarrollo del *software* educativo utilizando un lenguaje de programación de propósito general es considerablemente alto. Esto tiene claros efectos negativos, sobre todo en el diseño de la interfaz de alumno. Los profesores necesitan disponer de un prototipo rápido para comprobar la legibilidad del texto en pantalla, la necesidad de utilizar gráficos complementarios, etc., y todas estas funciones exigen herramientas más potentes que los meros lenguajes de programación de propósito general.

Los lenguajes de autor son lenguajes de programación específicamente concebidos para el desarrollo de aplicaciones pedagógicas de las computadoras. Un lenguaje de autor es en realidad un lenguaje de

programación de alto nivel, ligeramente simplificado, al que se han añadido algunas funciones útiles para la creación de *software* educativo. En general, cada expresión de un lenguaje de autor suele representar una síntesis de varias expresiones de un lenguaje de programación clásico.

Los primeros lenguajes de autor aparecieron en los años sesenta. Entre otros cabe mencionar el lenguaje COURSEWRITER desarrollado para sistemas IBM y el lenguaje TUTOR para el sistema PLATO. Además de estos dos lenguajes, que fueron desarrollados para sistemas específicos, se desarrollaron otros lenguajes de autor independientes del sistema como PLANIT, NATAL y PILOT. Podemos citar también el lenguaje de autor PEPA MACA, desarrollado en el Centro de Cálculo de la Universidad Politécnica de Cataluña.

La ventaja que puede representar la utilización de un lenguaje de autor frente a un lenguaje de propósito general es que algunas acciones habituales en el desarrollo de *software* educativo pueden expresarse mediante una única instrucción, lo que podría requerir muchas instrucciones en otro tipo de lenguajes ordinarios de programación.

Como contrapartida, se han encontrado varios inconvenientes en la utilización de los lenguajes de autor, lo cual ha frenado su implantación. La crítica más importante que hacen los usuarios de este tipo de lenguajes es que no evitan la programación y que, además, estos lenguajes son tan complejos como los demás (salvo algunas excepciones como PILOT) y requieren el mismo tiempo de aprendizaje que un lenguaje de programación clásico.

Los sistemas de autor son esencialmente programas que permiten crear *software* educativo sin necesidad de programar. Los autores especifican los contenidos de la materia que quieren enseñar y la estrategia de control del aprendizaje (acciones a realizar según la respuesta del alumno) que quieren utilizar. Con ello el sistema de autor genera automáticamente el programa correspondiente a las especificaciones dadas por los profesores. Este programa será utilizado por los alumnos en las sesiones de EAC (Enseñanza Asistida por Computadora).

La computadora con la cual los autores crean sus cursos no tiene por qué ser la misma que utiliza el alumno en las sesiones de EAC. Usualmente la computadora utilizada por los autores será más potente y con mayor capacidad de almacenamiento que la del alumno.

Los programas didácticos generados por un sistema de autor pueden estar escritos en un lenguaje de programación de propósito general o bien en un lenguaje de autor como los mencionados en el apartado

anterior. Esto dependerá del sistema de autor concreto y de la filosofía con que haya sido construido.

Hay tres razones principales que impulsaron el desarrollo de los sistemas de autor. La razón fundamental fue permitir la creación de *software* educativo a profesores sin conocimientos informáticos. De esta manera, los profesores pueden experimentar directamente con prototipos rápidos desarrollados por ellos mismos, adaptándolos y modificándolos de forma sencilla.

El segundo objetivo que se persigue al diseñar sistemas de autor es reducir el tiempo de desarrollo del *software* educativo. Estos sistemas permiten reducir considerablemente el tiempo de codificación de los programas educativos y evitan prácticamente la fase de depuración (corrección de errores). Ahora bien, lo que no pueden acortar estos sistemas es la fase del diseño pedagógico que es previa a la fase de informatización.

El tercer factor que motivó el desarrollo de los sistemas de autor fue conseguir una mayor transportabilidad del *software* educativo. Teniendo en cuenta los grandes costes de desarrollo del material educativo informatizado, si un mismo material puede utilizarse en distintas computadoras y distintos centros de enseñanza, su rentabilidad será mucho mayor y su precio de venta podrá ser más bajo. Sin embargo, no hay una estandarización ni en los lenguajes de programación de propósito general ni en los lenguajes de autor usados por los diversos centros e instituciones. Como un sistema de autor genera programas automáticamente, en principio es posible que estos programas sean generados en divesos lenguajes. El autor introduce el contenido de un curso y el sistema de autor produce distintos programas, correspondientes a ese contenido, pero escritos en diferentes lenguajes de programación para diferentes computadoras.

Los primeros sistemas de autor aparecieron en los años setenta. Entre ellos se encuentran VAULT, TICCIT y varios sistemas de autor desarrollados para PLATO. Posteriormente han aparecido numerosos sistemas de autor, especialmente diseñados para microcomputadoras, como TENCORE, MENTOR, MICROTEXT, SAM, TOPCLASS, SIETE (Sistema de Autor en español desarrollado en el Departamento de Informática y Automática de la Universidad Complutense), etc. (Strawford, 1988; Vaquero *et al.,* 1986). Su principal inconveniente reside en que el material que puede crearse con ellos tiene una estructura prefijada que suele limitar la creatividad de los autores.

### *4.2. Perspectivas de futuro*

Por lo que se refiere al futuro inmediato, la tendencia generalizada consiste en potenciar los sistemas de autor actuales, añadiéndoles nuevas funciones que aumenten sus posibilidades de aplicación. Esta tendencia se centra, en primer lugar, en potenciar la utilización de distintos medios audiovisuales para presentar información.

También se pretende desarrollar herramientas de ayuda para la fase de diseño pedagógico, que permitan estructurar los conocimientos y las secuencias de diálogos de una forma sencilla. Otra tendencia, que ya es una realidad en muchos casos, es la utilización de sistemas de hipertexto e hipermedia (Smith & Weiss, 1988) para la presentación de información al alumno.

Hipertexto es la escritura no secuencial. Un hipertexto es un conjunto de trozos de texto, u otros tipos de información, que están enlazados entre sí de una forma no secuencial. En esencia, un hipertexto es un grafo dirigido en el que cada nodo contiene un trozo de texto o información de otro tipo. En los documentos impresos habituales, la única información no secuencial está constituida por las notas a pie de página. Por esta razón, a veces se denomina al hipertexto «nota a pie de página generalizada». En hipermedia los nodos pueden contener cualquier tipo de información: Texto, gráficos, imágenes, sonidos, etc. ... Además en sistemas avanzados pueden combinarse nodos «pasivos» con otros «activos», como bases de datos, hoja electrónica o nodos ejecutables.

La utilización de sistemas de hipertexto en la enseñanza se está generalizando y todo parece indicar que tendrá un fuerte auge en los próximos años. Ahora bien, habrá que dotar a estos sistemas de métodos de «navegación» (Nielsen, 1990), para que el alumno no se sienta perdido, así como de formas de control de las acciones del alumno, para que el sistema pueda intervenir, con finalidad didáctica o de ayuda, en función de dichas acciones.

## *5. El paradigma cognitivo y la Inteligencia Artificial*

Podemos decir que la principal limitación del paradigma conductista es que sólo proporciona una descripción cuantitativa de la conducta. En ningún momento se pretende generar una descripción del estado interno en que se encuentra el individuo, ni descubrir los procesos internos responsables del aprendizaje.

Por el contrario, el enfoque cognitivo, formulado por Newell, Shaw y Simon en 1960, difiere de la aproximación conductista en que su objetivo es una descripción cualitativa de los procesos involucrados en la conducta cognitiva del individuo. Esta descripción se materializa en unos programas de computador que simulan uno o varios aspectos de la conducta del sujeto (en este caso referentes a la resolución de problemas), especificándose en estas simulaciones tanto las estructuras de datos como los algoritmos con los que se quieren reproducir los procesos cognitivos del ser humano.

Un enfoque diferente, pero también en la línea de la simulación cognitiva, es el propuesto por Quillian (1967), quien desarrolló las redes semánticas como un modelo de los procesos asociativos de la memoria humana. Tomando como base los trabajos de Quillian, Jaime Carbonell (1970) desarrolló SCHOLAR, que se suele considerar como el primer programa de enseñanza asistida por computadora que utiliza técnicas de Inteligencia Artificial (Wenger, 1987).

Aunque no se pueda decir que el paradigma cognitivo haya permitido superar el actual «impasse» en que parece encontrarse la informática educativa, no cabe duda de que sus efectos en la investigación informática y psicológica han sido muy importantes, permitiendo ensayar nuevas aproximaciones al problema del aprendizaje y de la representación del conocimiento en el hombre y en las máquinas, ayudando de esta manera a poner de manifiesto que en la aplicación del computador a la enseñanza existen, por debajo del problema pedagógico, una serie de problemas de carácter fundamental y de un gran interés informático (Pea & Soloway, 1988).

### *5.1. Representación del conocimiento*

En nuestra opinión la representación del conocimiento es la cuestión fundamental que subyace en la utilización del computador en la enseñanza (Fernández-Valmayor & Fernández Chamizo, 1988). Desde este punto de vista, la principal limitación del CAI no es su identificación con el paradigma conductista, ya que últimamente en el CAI se utilizan planteamientos pedagógicos basados en los principios de la Psicología Evolutiva y Cognitiva (Trowbridge & Bork, 1981; Trowbridge *et al.*, 1987), sino su dependencia de la informática tradicional. En un programa tradicional el conocimiento está representado implícitamente en la estructura del programa y en las secuencias de instrucciones. Estas técnicas de programación, basadas en los procedimientos y en el análisis descendente de una tarea compleja, son suficientes para modelar el condicionamiento operante o esquemas de enseñanza como el ML, pero

carecen de los mecanismos de representación adecuados para simular procesos cognitivos más elaborados.

La simulación cognitiva asume siempre, explícita o implícitamente, un marco teórico. El primero y más básico de los pilares de este marco se refiere a la forma de representar el conocimiento.

> «Representación del conocimiento es una combinación de estructuras de datos y de los procedimientos que las interpretan de modo que, si éstos se utilizan de forma correcta en un programa, resulte una conducta inteligente, una conducta basada en lo que el sistema sabe» (Barr & Feigenbaum, 1981).

En la actualidad, la práctica totalidad de los sistemas ITS-ICAI asumen lo que Newell (1980) denomina «hipótesis de un sistema de símbolos físicos», según la cual el conocimiento puede representarse mediante símbolos, agrupándose éstos para formar fórmulas. El esquema de representación mediante fórmulas puede traducirse a otros esquemas simbólicos como, por ejemplo, las redes semánticas (Rich, 1983). Sin embargo, estos sistemas contrastan radicalmente con el paradigma conexionista y la investigación en redes neuronales, basados en la hipótesis sub-simbólica (Anderson & Hinton, 1981), según la cual el significado se representa mediante configuraciones de nodos en una red, cuyas conexiones tienen un peso que puede ajustarse en sucesivos ciclos. En esta exposición y en nuestros propios trabajos nosotros hemos seguido, la hipótesis simbólica, pero no trataremos de justificar esta postura más allá de constatar que esta hipótesis ha probado su utilidad en el modelado de muchos procesos cognitivos (Langley *et al.*, 1989).

La búsqueda de formalismos para representar el conocimiento en un computador no es sólo una cuestión meramente técnica sino que, dependiendo de los formalismos que utilicemos, básicamente esquemas declarativos frente a esquemas basados en procedimientos, algunos procesos cognitivos podrán ser programados o modelados fácilmente, mientras que otros serán muy difíciles de modelar de forma adecuada.

Desde el punto de vista de Winograd (1975), las técnicas de programación en Inteligencia Artificial (IA) pueden verse como un progresivo esfuerzo por lograr lo que él llama «modularidad global», es decir, conseguir que el flujo del programa no tenga que determinarlo el programador, sino unos pocos procedimientos generales y las estructuras de datos que constituyen el conocimiento representado en el programa. Para Winograd las técnicas de programación estructurada de la informática tradicional representan el esfuerzo por lograr la «modularidad local», es decir, la técnica mediante la que el programador

trata de disminuir la complejidad de las interacciones entre procedimientos haciéndolas explícitas mediante el paso de parámetros y evitando los efectos laterales.

Conseguir que el flujo del programa sea independiente de las previsiones del programador implica, de forma implícita, disponer de algo tan fundamental como un esquema sencillo para simular el aprendizaje. Hoy en día quizá no tenga mucho sentido calificar de inteligente un sistema que no es capaz de aprender de su propia experiencia (Schank, 1987), y este requisito es todavía más necesario para un sistema que pretende simular nuestra capacidad de comprensión del lenguaje, puesto que esta tarea implica un proceso de aprendizaje continuo.

### *5.2. Representación del conocimiento y aprendizaje máquina*

Podemos decir que en informática tenemos desde formalismos de representación del conocimiento basados en procedimientos, que implican mecanismos de aprendizaje complejos y esquemas de interacción entre procedimientos previamente planificados por el programador, hasta esquemas de representación puramente declarativos, en los que la interacción entre proposiciones es mínima y aprender puede consistir simplemente en borrar o añadir proposiciones a la base de conocimientos.

Entre estos dos extremos tendríamos los sistemas basados en reglas de producción, en los que la representación del conocimiento se realiza mediante procedimientos (consecuentes de las reglas de producción), al mismo tiempo que se minimiza la interacción entre ellos al ser ésta sólo posible a través de la memoria de trabajo. También tendríamos los sistemas basados en esquemas o «frames», en los que una estructura declarativa, una jerarquía de «frames», se complementa con una serie de procedimientos «procedure attachment» que saben cómo realizar tareas específicas «to-fill», «when-filled», «if-needed», etc. Dentro de este esquema, las redes semánticas, sistema de representación ampliamente utilizado en los ITS, podrían ser consideradas como una primera etapa en la evolución de los sistemas basados en «frames» (Charniak *et al.*, 1987).

Los dos sistemas citados más arriba como posturas de síntesis en la controversia declarativo/procedimental, los sistemas basados en producciones y los sistemas basados en «frames», constituyen la base en que se apoya la simulación cognitiva tanto en IA como en el diseño de la mayoría de los ITS. Sin embargo, algunos ITS como DEBUGGY (Brown & Burton, 1978) han utilizado una red de procedimientos para repre-

sentar el conocimiento relativo a su dominio, pero este planteamiento ha tropezado con las dificultades propias de todo esquema procedural. Por ejemplo en DEBUGGY, cuyo dominio son los algoritmos de sustracción de números de más de un dígito, el algoritmo correcto se representaba mediante una red de procedimientos que tenía asociadas en algunos de sus nodos versiones erróneas («buggy») del procedimiento correcto. El principal inconveniente de este planteamiento es que hay que realizar previamente un extenso análisis empírico de los errores cometidos por los alumnos, para luego incluirlos en una red de procedimientos (este proceso nos recuerda, aunque con algunas diferencias a los métodos de diseño del CAI tradicional). Además, si después se quería añadir o modificar algún procedimiento de la red, esto podía dar lugar a extensas modificaciones de la misma.

De la dificultad de modelar de esta forma el conocimiento humano puede dar una idea el que, para un proceso tan sencillo como la sustracción, VanLehn (1982) informe de más de 100 errores de procedimiento cometidos por los alumnos. Para otros investigadores, como Ridgway (1988), es sencillamente imposible que un modelo como DEBUGGY, y al margen de otras críticas a su planteamiento pedagógico, pueda alcanzar una competencia siquiera comparable a la de un profesor humano, incluso en un dominio tan restringido como el de la resta, y ello debido al propio modelo cognitivo que emplea, la red de procedimientos, que carece de un mecanismo apropiado para ir actualizando su base de conocimientos como consecuencia de su interacción con alumnos y profesores.

No obstante, estas críticas a los modelos basados en redes de procedimientos deben entenderse como críticas a su validez como modelos explicativos de los procesos cognitivos, no a su utilidad, e incluso idoneidad, para modelar determinados aspectos del razonamiento de los expertos, que son difícilmente expresables mediante reglas (Dhar & Pople, 1987).

Las reglas o producciones son quizá el formalismo de representación del conocimiento que más ha caracterizado a los ITS y ello es quizá porque, aunque posterior a la utilización de redes semánticas (Carbonell, 1970), este formalismo sugiere un mecanismo de aprendizaje más inmediato que las redes semánticas, lo que en tareas de enseñanza constituye una indudable ventaja. Para algunos investigadores (Larkin *et al.*, 1980), los pares condición-acción que forman las producciones son la más sofisticada contrapartida a los pares estímulo-respuesta de la psicología conductista. Newell y Simon (1972) proponen los sistemas de producción como un modelo del conocimiento humano. Sin embargo, el

término «sistemas de producción» no siempre tiene unos límites claramente definidos:

> «El concepto de sistema de producción es difuso, y es difícil determinar dónde están sus límites y dónde comienzan otros formalismos utilizados en Informática y en Psicología. En Informática se utiliza una categoría más general llamada sistemas dirigidos por patrones, que incluye a los sistemas deductivos como MYCIN y otros tipos de arquitecturas» (Anderson, 1983).

Un sistema de producción (con encadenamiento hacia adelante) consiste básicamente en un conjunto de reglas condición-acción y una memoria dinámica o memoria de trabajo. El sistema de producción opera en ciclos. En cada ciclo las condiciones de cada producción se comparan con el estado actual de la memoria de trabajo. De entre las reglas que comparan con éxito, se seleccionan una o varias, que activan los procedimientos asociados con ellas. Los procedimientos, cuando se ejecutan, afectan al estado de la memoria de trabajo, haciendo que nuevas producciones puedan ser seleccionadas en el ciclo siguiente. El proceso continúa hasta que no se pueden seleccionar más producciones, o encontramos una orden de detener el proceso (Langley *et al.,* 1989).

En teoría, este modelo de programación es completamente modular y ha sido utilizado para modelar procesos de aprendizaje en niños, como el aprendizaje de la resta (Young & O'Shea, 1981), y las etapas de desarrollo de los niños descritas por la psicología evolutiva (Young, 1976; Klahr & Siegler, 1978). Ahora bien, el verdadero aprendizaje, la adquisición automática de reglas por parte del sistema, es un problema para el que todavía no existen paradigmas generales sino más bien técnicas concretas para casos específicos (Cuena, 1986; Wilkins, Clancey & Buchanan, 1988).

Estos modelos simulan los errores cometidos por los niños y su mayor o menor grado de desarrollo, añadiendo y borrando reglas de la base de conocimiento. Para que estos modelos sean verdaderamente modulares y así cumplir sus objetivos, los investigadores deben minimizar la interacción entre reglas, proceso que resulta ser mucho más complicado de lo inicialmente previsto, especialmente cuando el número de reglas es considerable, lo que por otra parte es completamente necesario para simular situaciones reales.

A veces, los ITS basados en reglas han estado basados en el éxito de un sistema experto anterior, como en el caso de GUIDON, que utiliza una base de conocimientos ya existente (la base de conocimientos de MYCIN), para enseñar a los alumnos a razonar sobre el mismo dominio

que el sistema experto. Los resultados del proyecto GUIDON (Clancey, 1987) demuestran que un conjunto de reglas de producción sin estructurar no es una base suficiente para un programa tutorial, ya que éste necesita hacer explícitas las estrategias y las relaciones en que se apoyaron los autores de la base de conocimientos. Con respecto al conocimiento estrictamente tutorial, también representado en este sistema mediante reglas situación-acción («t-rules»), se pone de manifiesto la necesidad de encontrar un marco teórico, basado en una teoría de la memoria y del aprendizaje, capaz de justificar los principios implícitos en estas reglas.

En otras ocasions, los ITS están basados en una teoría del aprendizaje, como por ejemplo los tutores construidos por Anderson y su grupo en Carnegie-Mellon (Anderson, 1983). Estos tutores están basados en ACT* que pretende ser una teoría comprensiva del conocimiento humano. Este no es el único aspecto singular de los tutores («LISP» y «Geometry») construidos por Anderson, ya que éstos, al contrario de la mayoría de los ITS, no son prototipos experimentales construidos para ensayar una aproximación determinada al problema de la enseñanza, sino que son sistemas completos actualmente utilizados por estudiantes reales (Kafai, 1989).

### *5.3. Sistemas basados en reglas frente a sistemas basados en esquemas*

Si pensamos que las posibilidades revolucionarias del computador en la educación están basadas en sus posibilidades para simular los procesos cognitivos y en especial la comprensión del lenguaje, surgen diversas cuestiones como, ¿cuáles son los modelos posibles?, ¿cuál es el modelo correcto? y ¿cuáles son los modelos que actualmente se están utilizando?

En nuestra opinión, tanto en la IA en general como en la construcción de ITS en particular, hoy día se utilizan solamente dos modelos básicos. El más conocido es, como ya citamos anteriormente, el modelo basado en reglas o producciones (quizá como reflejo del éxito alcanzado en la industria de los sistemas expertos basados en reglas). El otro modelo es el que Anderson (1983) llama «arquitecturas basadas en esquemas» y que en la literatura encontramos especializado bajo diferentes nombres como «frame systems», «slot and filler data bases» y más recientemente CBR, «Case-Based Reasoning systems» (Riesbeck & Schank, 1989).

Ambos sistemas, los de producción y los basados en esquemas, son marcos generales en los que se aplican una variedad de teorías específi-

cas, como pueden ser la ya citada ACT* de Anderson, utilizada en el marco de los sistemas de producción, y la «Dependencia Conceptual» de Schank, utilizada en el marco de los sistemas basados en esquemas.

Al tratar de comparar ambos enfoques, la primera dificultad está en su diferencia en cuanto al grado de detalle en que estos sistemas han sido especificados por sus propios creadores. Los sistemas de producción y las teorías como la ya citada ACT*, están mucho más claramente formalizados que los sistemas basados en esquemas.

El funcionamiento de un sistema de producción ya lo hemos descrito, en líneas generales, en párrafos anteriores. También hemos hablado de sus ventajas, como su modularidad (es fácil añadir y borrar reglas de su base de conocimiento), y hemos citado las limitaciones de este proceso, especialmente cuando las reglas contenidas en la base son muchas y no es posible controlar la interacción entre las mismas.

Sin embargo, los sistemas basados en reglas tienen problemas más básicos que el de las interacciones no deseadas entre reglas. El primero es que, por la propia naturaleza de las reglas, en un sistema de este tipo es fácil captar el conocimiento que corresponde a situaciones como «¿qué hacer en tal caso?», pero es difícil captar el conocimiento que podríamos caracterizar por las preguntas «¿por qué funciona?», o «¿qué significa?», denominado a veces «conocimiento profundo del dominio» (McCarthy, 1987).

Otro problema, que afecta a la propia esencia de la arquitectura de los sistemas basados en reglas, es que el conocimiento queda disperso en la base en cientos de reglas (completamente independientes unas de otras en el caso ideal), lo cual constituye un modelo muy poco plausible de la forma en que los seres humanos tienen organizado el conocimiento en su mente. Muchos experimentos han demostrado (ver por ejemplo Nisbett & Ross, 1980) que los seres humanos utilizan extructuras bien organizadas de conocimiento, de forma que conocimientos y experiencias relacionadas están «cerca» unas de otras. Insistiendo más todavía en este aspecto, quizá el resultado empírico más notable es que un sistema experto con más reglas en su base de conocimientos es más lento que el mismo sistema con menos reglas, mientras que en los expertos humanos se observa lo contrario. Las personas con gran experiencia en un campo del saber, que de ser cierto este modelo tendrían una cantidad de reglas específicas en su mente mucho mayor que las personas inexpertas, son mucho más rápidas dando una respuesta o una solución a una situación determinada.

El último problema que citaremos para estos sistemas es la dificul-

tad de representar mediante reglas experiencias concretas. Las reglas son un formalismo poco adecuado para representar las secuencias de sucesos, las situaciones prototípicas y los hechos concretos que componen una gran parte de nuestra experiencia (Riesbeck & Schank, 1989).

Todas estas dificultades contribuyen a hacer más grave el problema del cuello de botella en la adquisición del conocimiento en los sistemas basados en reglas, tanto en el aprendizaje o actualización dinámica de la base de conocimientos, mediante su interacción con los usuarios, como para su mantenimiento, un problema que es capital para poder utilizar con éxito la informática en la educación.

Por otra parte, los sistemas basados en esquemas no tienen, como ya hemos dicho anteriormente, un marco tan claramente definido como los basados en reglas y las diferentes implementaciones de sistemas concretos pueden diferir bastante unas de otras. No obstante, podemos decir que en todas ellas el ciclo básico de control consiste en comparar, «matching», un esquema (que representa un episodio o un nuevo *item* de información) con los esquemas contenidos en su memoria, que en algunos casos puede estar subdividida en memoria a largo plazo, con todos los esquemas que contiene el sistema, y memoria de trabajo con un pequeño número de esquemas activos en ese instante.

En cualquier caso, asociar el nuevo esquema a los esquemas que ya están en memoria es el concepto central. Además este «matching» o emparejamiento del nuevo esquema con uno o varios de los esquemas ya en memoria puede ser parcial, con lo que se pone en marcha uno de los recursos principales de este tipo de sistemas, que consiste en completar la información recibida con la información que el sistema supone por defecto. Por ejemplo, al ver la parte frontal de una figura geométrica conocida, completamos esta información con la que tenemos por defecto para la parte posterior (Winston, 1984).

En algunos de estos sistemas, un nuevo esquema se guarda en memoria como una especialización a utilizar en futuros procesos de «matching». En otros, la estrategia puede consistir en modificar un esquema anterior, para así tener un esquema más general o que represente mejor el caso prototípico. La organización de la base de conocimientos es jerárquica y en muchos casos, aunque no siempre, esta jerarquía se implementa mediante relaciones del tipo «es-un» o «parte-de».

El mayor problema de los sistemas basados en esquemas es que no hacen distinción entre el conocimiento declarativo y procedimental (Anderson, 1983). Fundamentalmente no ofrecen ningún mecanismo

con el que simular la automatización de una conducta, una conducta que al principio realizamos basados en nuestro conocimiento declarativo y que, después de un cierto período, automatizamos en un proceso más cercano al esquema de los sistemas basados en producciones (un procedimiento que se activa automáticamente al reconocer una situación determinada).

Otros problemas importantes tienen un carácter más técnico. Por ejemplo, los esquemas tienden a ser estructuras complejas difíciles de manipular en el computador. El proceso de «matching», especialmente el «matching» parcial entre estructuras complejas, es una operación computacionalmente difícil para la que no siempre hay una solución satisfactoria (Charniak *et al.,* 1987). También, en esta misma línea de dificultades técnicas, estarían los problemas de aprendizaje, que en este caso equivalen a la adquisición de nuevos esquemas complejos. En este último problema, el de las estrategias de adquisición de esquemas complejos, en el que estamos trabajando en nuestro grupo, por considerar que ésta es una cuestión fundamental para lograr nuestro objetivo: Un sistema informático en el que se pueda representar en forma operativa el conocimiento contenido en el lenguaje.

Finalmente, añadiremos que, como ocurría en el caso de la clasificación que anteriormente hicimos del *software* educativo, los sistemas realmente implementados incluyen en muchas ocasiones características de ambos enfoques, reglas de producción de una parte y jerarquías de «frames» de otra.

## 6. *Conclusiones. Un nuevo enfoque de la Informática Educativa*

El análisis que hemos realizado de los distintos modos de utilización del computador en la enseñanza nos muestra que tanto en los sistemas CAI como en los ITS, el computador se concibe como una máquina de enseñar. Esto implica que la transmisión tutorial del conocimiento, subyace como objetivo principal a todo el planteamiento de tales sistemas.

Algunos investigadores, como Anderson, consideran que el desarrollo de programas tutoriales es un excelente campo de prueba para las teorías cognitivas y para encontrar nuevos problemas en los que centrar la investigación en ciencia cognitiva y se muestran bastante escépticos con respecto a los resultados prácticos que el desarrollo de los ITS pueda tener en el sistema educativo.

> «El sistema educativo es extraordinariamente conservador y puede ocurrir que los avances de la tecnología sean sencillamente ignorados por éste. Por muchas razones, la bondad de un método educativo tiene muy poca repercusión sobre la política educativa. La justificación real para desarrollar "Tutores Inteligentes" debe tener como objetivo mejorar nuestra comprensión sobre cómo aprenden los seres humanos. Si estos "Tutores Inteligentes" tienen, además, un efecto social beneficioso, mucho mejor.» (Anderson, 1987.)

Sin embargo, nosotros pensamos que no es sólo una cuestión de sensibilidad social o de políticas educativas adecuadas. El objetivo, tantas veces enunciado, de conseguir que el computador sea la causa de una mejora significativa en la educación, podría lograrse, si el computador llegase a ser una máquina verdaderamente capaz de procesar información. Para ello, la cuestión esencial es disponer de métodos que nos permitan representar, internamente en el computador, el mismo conocimiento que representamos en un texto y que el computador sea capaz de simular los procesos cognitivos que permiten resumir y relacionar este conocimiento con otros previamente adquiridos. Además, todo ello debe ser un proceso dinámico, es decir, no importa tanto la calidad de un primer resultado como la capacidad del sistema para mejorar como consecuencia de su interacción con los usuarios.

Éste es el objetivo principal del trabajo que estamos desarrollando en el Departamento de Informática y Automática de la Universidad Complutense (Fernández-Valmayor & Fernández Chamizo, 1990; Fernández-Valmayor, 1990). Para avanzar en este camino estamos trabajando en el desarrollo de un prototipo basado en esquemas o «frames». Dentro de este tipo de sistemas, hemos seguido una línea orientada a facilitar el aprendizaje mediante la construcción de una memoria dinámica capaz de crear nuevos esquemas a partir de la información suministrada por los usuarios. Nuestro sistema se apoya en gran medida en los trabajos realizados por R. Schank (1982) y sus colaboradores, estando constituido el núcleo del sistema por una base de conocimientos dinámica, que nosotros hemos concebido como un modelo cognitivo de la memoria humana.

Es claro que la utilidad de una base de conocimientos de este tipo en el entorno educativo no está ligada, «a priori», a ninguna estrategia pedagógica concreta y que quizá su mayor utilidad pueda ser la de ayudar a un investigador, o a un alumno, a procesar el enorme volumen de información textual hoy día disponible. En nuestra concepción de la utilización del computador en la enseñanza, los programas tutoriales (que pueden ser diseñados independientemente) son un tipo especial de interfaz con el sistema. La información que un programa tutorial

entrega a la base de conocimientos estaría basada en las respuestas de los alumnos. El sistema, operando sobre esta información, con los mismos procedimientos que utiliza para organizar y elaborar la información suministrada por los profesores, podrá ir refinando unas estructuras de datos que representarán lo que el sistema «sabe» sobre los alumnos.

Finalmente, añadir que en nuestro trabajo estamos experimentando con dominios de conocimiento concretos, en los que hemos conseguido poner de manifiesto que esta forma de representar y tratar la información resulta particularmente apropiada para modelar los procesos cognitivos.

**Dirección de los autores:** Alfredo Fernández-Valmayor Crespo, Carmen Fernández Chamizo y Antonio Vaquero Sánchez, Departamento de Informática y Automática, Facultad de Ciencias Físicas, Universidad Complutense de Madrid, Ciudad Universitaria, s/n., 28040 Madrid.

*Fecha de recepción de la versión definitiva de este artículo:* 20.XII.1990.

## BIBLIOGRAFÍA

ANDERSON, J. R. (1987) Methodologies for studying knowledge, *Behavioral and Brain Sciences,* vol. 10, pp. 467-505.

— (1983) *The Architecture of cognition* (Cambridge, MA, Harvard University Press).

ANDERSON, J. A. & Hinton, G. E. (1981) Models of information processing in the brain, en G. E. HINTON & J. A. ANDERSON (Eds.) *Parallel models of associative memory* (Hillsdale, N.J., Lawrence Erlbaum Associates).

AUSUBEL D. P. (1963) *The Psychology of Meaningful Verbal Learning* (New York, Grune and Stratton).

— (1968) *Educational Psychology: A cognitive View* (New York, Holt, Rinehart and Winston).

BITZER, D. L.; BRAUNFELD, P. G. and LINCHTENBERGER, W. W. (1962) PLATO II: A multiple-student, computer-controlled, automatic teaching device, en COULSON, J. E. (ed.) *Programmed Learning and Computer-Based Instruction* (New York, John Wiley and Sons).

BARR, A. & FEINGENBAUM, E. A. (1981) *The Handbook of Artificial Intelligence* (CA., William Kauffmann).

BLOOM, B. S. (1976) *Human Characteristics and School Learning* (New York, McGraw-Hill).

BORK, A. (1979) Interactive Learning, *American Journal of Physics,* 47, pp. 5-10.

— (1986) *El Ordenador en la Enseñanza* (Barcelona, Ed. Gustavo Gili).

— (1989) The History of Technology and Education. Informe técnico, Educational Technology Center, *University of California, Irvine.*

BROWN, J. S. & BURTON, R. R. (1978) Diagnostic models for procedural bugs in basic mathematical skills, *Cognitive Science,* 2, pp. 155-192.

CARBONELL, J. R. (1970) AI in CAI: an artificial intelligence approach to computer-assisted instruction, *IEEE Trans. on Man -Machine Systems,* 11, pp. 190-202.

CARNOY, M.; DALEY, H. & LOOP, L. (1987) Education and Computers: Vision and Reality. *ED/87/WS/37 Monografía editada por la UNESCO.* School of Education. Standord University, Standford, California y Division of Educational Sciences, Contents and Methods of Education (UNESCO, Paris).

CHARNIAK, E.; RIESBECK, C.; MCDERMONTT, D.; MEEHAN, J. (1987) *Artificial Intelligence Programming* (N.J., Lawrence Erlbaum Associates).

CLANCEY, W. J. 1 (1987) *Knowledge-based Tutoring: The GUIDON Program* (Cambridge, Massachusetts, MIT Press).

COULSON, J. E. (EDITOR) (1962) *Programmed Learning and Computer-Based Instruction* (New York, John Wiley and Sons).

CUENA, J. (1986) Adquisición del conocimiento y aprendizaje en sistemas basados en reglas, en CUENA J.; FERNÁNDEZ, G.; LÓPEZ DE MÁNTARAS, R.; VERDEJO, M. F. *Inteligencia Artificial: Sistemas Expertos* (Madrid, Alianza Editorial).

DHAR, V. and POPLE, H. E. (1987) Rule-Based versus Structure-Based Models for Explaining and Generating Expert Behavior, *Communications of the ACM,* vol. 30, n. 6, pp. 542-555.

FERNÁNDEZ-VALMAYOR, A. & FERNÁNDEZ CHAMIZO, C. (1988) Representación del Conocimiento en los Sistemas de Enseñanza Basados en Computadora, *Revista de la Real Academia de Ciencias Exactas, Físicas y Naturales de Madrid,* 82:2, pp. 319-321.

— (1990) An Educational Application of Memory Organization Models, *Proceedings of the ARCE International Conference on Advanced Research on Computers in Education,* IFIP & IPSJ, Tokyo.

FERNÁNDEZ-VALMAYOR, A. (1990) *Diseño de una base de conocimientos dinámica y su aplicación en un entorno educativo,* tesis doctoral. Departamento de Informática y Automática, Facultad de Ciencias Físicas (Madrid, Universidad Complutense).

HOLDEN, C. (1989) Computers Make Slow Progress in Class, *Science,* vol. 244, pp. 906-909.

KAFAI, Y. (1989) What happens if you introduce an Intelligent Tutoring System in the classroom: A case study of the Geometry Tutor, *Proceedings of the NECC 89,* en W. C. RYAN (Ed.) International Council on Computers for Education, University of Oregon, pp. 46-53.

KLAHRL, D. and SIEGLER, R. (1978) The Representation of Children's Knowledge, *Advances in Child Development,* vol. 12, en H. W. REESE and L. P. LIPSETT, eds. (New York Academic Press).

KURTZ, B. L. & BORK, A. (1981) An SADT Model for the Production of Computer Based Learning Material, en R. LEWIS & D. TAGG *Computers in Education,* (Ed.), North-Holland Publishing Company, IFIP.

KURLAND, D. M. & KURLAND, L. C. (1987) Computers Applications in Education: A Historical Overview, *Ann. Rev. Comput. Sci.,* 2, pp. 317-358.

LANGLEY, P.; WOGULIS, J.; OHLSSONHL, S. (1989) Rules and principles in Cognitive Diagnosis, Technical Report 89-02, ICS University of California, Irvine, en N. FREDERICKSEN (Ed.) *Diagnostic Monitoring of Skill and Knowledge Acquisition* (Lawrence Erlbaum Associates).

LARKIN, J. L.; MCDERMOTT, J.; SIMON, D. P. and SIMON, H. A. (1980) Expert and Novice Performance in Solving Physics Problems, *Science,* vol. 208, pp. 1.335-1.342.

MCCARTHY, J. (1987) Generality in AI, *Communications of the ACM,* 30, 12, pp. 25-37.

MANDL, H. & LESGOLD, A. (Editors) (1988) *Learning Issues for Intelligent Tutoring Systems* (New York, Spring-Verlag).

MERRIL, M. D. (1974) Premises, propositions and research underlying the design of a learner controlled computer assisted instruction system: a summary for the TICCIT system, *Working Paper No. 44, Div. Inst. Services,* Birgham Young University.

NEWELL, A. (1980) Reasoning, problem solving, and decision processes: The problem space hypothesis, en NICKERSON R. (Ed.) *Attention and performance* VIII (Hillsdale, N.J., Erlbaum).

NEWELL, A. & SIMON, H. (1972) *Human Problem Solving,* (N.J., Englewood Cliffs, Prentice-Hall).

NEWELL, A.; SHAW, J. & SIMON, H. A. (1960) Report on a general problem-solving program for a computer, *Proceedings of the International Conference on Information Processing, UNESCO,* Paris.

NIELSEN, J. (1990) The Art of Navigating through Hypertext, *Com. of the ACM,* March, 33:3.

NISBETT, R. & ROSS, L. (1980) *Human Inference: Strategies and Shortcomings of Social Judgments* (Englewood Cliffs N.J., Prentice-Hall).

O'SHEA, T. & SELF, J. (1985) *Enseñanza y aprendizaje con ordenadores* (Madrid, Ediciones Anaya Multimedia).

PAPER, S. (1980) *Mindstorms: Children, Computers and Powerful Ideas* (New York, Basic Books).

QUILLIAN, M. R. (1967) *Word Concepts: A Theory and Simulation of Some Basic Semantic Capabilities,* en BRACHMAN, R. J. y LEVESQUE, H. J. (Eds.) *Reading in Knowledge Representation* (CA, Morgan Kaufmann Publishers Inc., 1985).

RICH, E. (1983) *Artificial Intelligence* (New York, McGraw-Hill).

RIDGWAY, J. (1988) *Of course CAI is impossible ... worse though, it might be seditious,* en J. SELF (Ed.) *Artificial Intelligence and Human Learning* (London, Chapman and Hall Computing).

RIESBECK, C. K. & SCHANK, R. C. (1989) *Inside Case-Based Reasoning* (Hillsdale, N.J., Lawrence Erlbaum Associates).

SANDERS, D. H. (1981) *Computers in Society* (New York, McGraw-Hill).

SCHANK, R. C. (1987) What is AI, Anyway?, *AI Magazine,* Winter.

— (1982) *Dynamic memory. A theory of reminding and learning in computers and people* (Cambridge, MA, Cambridge University Press).

SKINNER, B. F. (1970) *Tecnología de la enseñanza* (Barcelona, Labor).

SLEEMAN, D. & BROWN, J. S., Ed. (1982) *Intelligent Tutoring systems* (London, Academic Press).

SMITH, J. & WEISS, S. (1988) An Overview of Hipertext, *Comm. of the ACM,* July 1988, vol. 31-7.

STRAWFORD, G. (1988) *Authoring Packages, a comparative report.* The National Interactive Video Centre.

SUPPES, P. & ATKINSON, R. C. (1960) *Markov learning models for multiperson interactions* (Stanford, California, Stanford University Press).

SUPPES, P.; JERMAN, M. & BRIAN, D. (1968) *Computer-Assisted Instruction: Stanford's 1965-66 Arithmetic Program,* Institute for Mathematical Studies in the Social Sciences (Stanford University, Academic Press, New York).

SUPPES, P. (1979) Current Trends on Computer-Assisted Instruction, *Advances in Computers,* 18, pp. 173-229 (Academic Press).

TROWBRIDGE, D. (1987) A Look at Educational Computing in 1987, Technical report submitted to the BSCS study of elementary school science, *Center for Design of Educational Computing, Carnegie Mellon University.*

TROWBRIDGE, D. & BORK, A. (1981) Computer Based Modules for Early Adolescence, en LEWIS, R. & TAGG, D. (Ed.) *Computers in Education* (North-Holland Publishing Company, IFIP).

VANLEHN, K. (1982) Bugs are not enough: Empirical studies of bugs, impasses, and repairs in procedural skills, *Journal of Mathematical Behavior,* 3, pp. 3-72.

VAQUERO, A.; FERNÁNDEZ CHAMIZO, C. (1987) *La Informática aplicada a la Enseñanza* (Madrid, Eudema).

VAQUERO, A.; FERNÁNDEZ CHAMIZO, C.; SÁNCHEZ, J. M.; TROYA, J. M.; HERNÁNDEZ, L. (1986) SIETE: Sistema Informatizado en Español para el desarrollo de Temas de Enseñanza, *Revista de la Real Academia de Ciencias Exactas Físicas y Naturales,* tomo LXXX, pp. 473-476.

WENGER, E. (1987) *Artificial Intelligence and Tutoring Systems* (CA, Morgan Kaufmann Publishers).

WILKENS, D. C.; CLANCEY, W. J. & BUCHANAN, B. G. (1988) Using and Evaluating Differential Modeling in Intelligent Tutoring and Apprentice Learning Systems, en PSOTKA, J.; DAN MASSEY, L.; MUTTER, S. A. (Editors) *Intelligent Tutoring Systems. Lessons Learned* (N.J., Lawrence Erlbaum Associates).

WINOGRAD, T. (1975) Frame Representations and the Declarative/Procedural Controversy, BRACHMAN, R. J. y LEVESQUE, H. J. (Eds.) *Readings in Knowledge Representation* (CA, Morgan Kaufmann Publishers Inc., 1985).

WINSTON, P. H. (1984) *Artificial Intelligence* (New York, Addison-Wesley).

YOUNG, R. M. (1976) *Seriation by Children: An Artificial Intelligence Analysis of a Piagetian Task* (Birkhauser, Basel).

YOUNG, R. M. & O'SHEA, T. (1981) Errors in Children's Substration, *Cognitive Science,* vol. 5, pp. 153-177.

SUMMARY: COMPUTERS IN EDUCATION: FROM BEHAVIORAL TO COGNITIVE THEORIES.

It this paper we present an analysis of how computers have been used in the educational environment during the last decades. Two different methodologies for designing tut rial programs are brought into focus: methodologies based in behavioral theories and classical software engineering are compared with those based on cognitive theories and Artificial Intelligence programming techniques. Finally, we present our work in the «Departamento de Informática y Automática de la Universidad Complutense». The core of the system we are working on, is a dynamic knowledge base in which we represent domain and student knowledge. In our approach, tutorial programs are considered as interfaces with that knowledge base.

KEY WORDS: Computer Assisted Instruction. Intelligent Tutoring Systems. Artificial Intelligence. Behavioral theories. Cognitive Science.