Informatica Educativa Vol. 10 ,No. 1, 1997
UNIANDES - LIDIE, Colombia pp.57-73

DISEÑO DE SOFTWARE EDUCATIVO O DE SOFTWARE ESCOLAR ?

Enrique HINOSTROZA **Harvey MELLAR** **Lucio REHBEIN**
Pedro HEEP **Christina PRESTON**

RESUMEN

En este trabajo argumentamos sobre la necesidad de una cambio de perspectiva en la investigación en el área de diseño de software educativo. A través de una revisión de investigaciones previas en las áreas del diseño de software educativo llegamos a establecer la necesidad de conceptualizar este tipo de software no como una herramienta de aprendizaje, sino como una herramienta de apoyo a la enseñanza. Esta nueva conceptualización permite establecer la necesidad de conocer y considerar el contexto de uso de software educativo y de entender la perspectiva de los usuarios acerca de los roles y posibilidades de esta tecnología. En definitiva, argumentamos en favor de la necesidad de investigar en el concepto de software educativo desde una perspectiva situada (situated).

INTRODUCCIÓN

Enlaces es un proyecto nacional que forma parte del programa de Mejoramiento de la Calidad y Equidad de la Educación (MECE) del Ministerio de Educación de Chile. Este proyecto ha estado en funcionamiento desde 1993 en la Universidad de la Frontera (UFRO) en la región de La Araucanía en el sur de Chile. El Programa MECE busca tener un impacto de largo aliento en la calidad, equidad y descentralización de educación chilena. A través del Proyecto Enlaces, el programa MECE ha considerando la incorporación de tecnología computacional moderna en las escuelas más pobres del país.

Con este fin, entre los años 1993 y 1997 se ha instalado en forma progresiva una red computacional que incorpora a más de 1.000 escuelas primarias y secundarias, tanto urbanas como rurales. Para el año 2000 se ha planificado que la red contará con aproximadamente 5.000 escuelas. Además, en la Universidad de La Frontera, se creó el Instituto de Informática Educativa para coordinar el proyecto a nivel nacional, diseñar actividades pedagógicas, construir material para las escuelas y dar capacitación y asistencia técnica a una proporción de escuelas.

Entre los materiales entregados a las escuelas se encuentran productos de software educativo. Dicho software es, en buena medida, producido en el Instituto de Informática Educativa,

debido a la falta de productos en español y de productos que satisfagan las necesidades específicas de las escuelas chilenas. La experiencia en la introducción y uso de software educativo en las escuelas nos llevó a cuestionarnos no sólo el uso de software educativo (disponible en el mercado o producido localmente), sino también el concepto de software educativo y su rol en las escuelas.

El uso de software educativo en las escuelas es aún arena de debate y controversia. Por una parte, está la industria de software educativo con un mercado creciente de productos multimediales, evolucionando tan rápido como la tecnología de hardware (ABLA Learning Works, Broderbund, Microsoft Corp., Sanctuary Woods Corp, TAG Development, The Learning Company, Tom Snyder Productions, Unlimited, ZETA Multimedia, etc.)*. Por otra parte están los consumidores, en este caso hay evidencia de que el rol de esta tecnología es controversial [1], que su efecto no es concluyente [2] y que el software que se utiliza con mayor frecuencia es del tipo ensayo y error [3]. Por último están los grupos de investigación, produciendo un número creciente de informes focalizados en los procesos de enseñanza y aprendizaje que ocurren al utilizar productos de software específicos [4-9]. Esto tres grupos utilizan y ofrecen productos de software diferentes y tienen visiones muy distintas de cómo establecer su valor.

En este contexto, nuestra intención con este trabajo es contribuir a una mejor comprensión del concepto de software educativo a través del análisis del diseño de este tipo de software, incorporando una perspectiva distinta que presenta al software como una herramienta para el desempeño profesional del docente en el aula+ .

Para analizar el diseño del software, presentaremos las diferentes estrategias de clasificación de software educativo que se han utilizado, a través de estos diversos tipos de clasificación es posible visualizar los elementos de diseño imbuidos en el software. Basado en esta evidencia, proponemos una nueva forma de clasificar el software educativo que pone de manifiesto una área de problemas relacionados con la falta de comprensión del rol del computador en las escuelas y de las actividades que desempeña el profesor en el aula utilizando dicha tecnología.

DISEÑO TRADICIONAL DE SOFTWARE EDUCATIVO

---

* Los nombres de las empresas son marca registrada.

\+ En otro trabajo a publicar por los mismos autores se abordan tres perspectivas adicionales: la evaluación y desarrollo de software educativo junto a la innovación en educación. Estas otras perspectivas complementan los argumentos aqui presentados.

La tarea de diseñar software educativo puede ser analizada desde varios puntos de vista:

- desde la perspectiva metodológica, esto es, herramientas y técnicas para diseñar software (diseño orientado a objetos, diseño estructurado, diseño top-down, diseño bottom up, etc.),

- desde la perspectiva de diseño de los componentes de software (interfaz humano computador, contenidos, funcionalidad, etc.) y

- desde la perspectiva de las intenciones del autor del software, esto es, mirando el producto completo tal como fue concebido.

En este caso analizaremos el diseño de software educativo desde la perspectiva de las intenciones del autor, esto es, los principios de enseñanza y aprendizaje subyacentes en el software. Comprendiendo la dificultad conocer las intenciones reales del autor, nos focalizaremos en los elementos explícitos del diseño.

CLASIFICACIÓN DE SOFTWARE EDUCATIVO

Learning with Software [10] presenta una visión general de los diferentes intentos realizados para clasificar software. En este artículo utilizaremos una organización similar, incluyendo referencias adicionales (el texto citado de la fuente que se ha traducido se indica entre “ y ”).

I) Por tema:

> “Esencialmente este sistema provee una clasificación de software basado en las áreas curriculares escolares. Por ejemplo, todos los productos de software que se pueden utilizar, digamos en Ciencias Sociales. No es un buen sistema de clasificación cuando se discute sobre principios de informática educativa, pero es útil si sólo se desea describir recursos apropiados para áreas específicas enseñadas en las escuelas.”

II) Por tipo de software

Con las siguientes categorías:

- **Computador como tutor**

  Para funcionar como un tutor en un área temática específica, el computador debe ser programado por un experto para proveer un ‘profesor subordinado’ al usuario. En el contexto de uso, el computador (como si fuera un experto) presenta al estudiante los temas a tratar, junto con un conjunto de preguntas o directrices; el estudiante responde y el computador completa el ciclo de

aprendizaje evaluando la respuesta, de los resultados de dicha evaluación determina qué presentar a continuación.

- **Computador como herramienta**

  Para funcionar como herramienta, el computador debe tener software genérico, como procesadores de texto, planillas de cálculo, software de base de datos, etc.

- **Computador como aprendiz**

  Para funcionar como aprendiz el computador provee un ambiente en el cual el usuario puede 'enseñar' al computador expresando sus propias ideas y soluciones a problemas. Para enseñar al computador, el usuario debe aprender a programar, a comunicarse con el computador en un lenguaje que este comprenda.

Esta clasificación fue propuesta inicialmente por Taylor [11] y es mencionada por Squires y McDougall [12]. Una clasificación similar utilizando los criterios de 'tipo de software' fue utilizada por Laurillard [13], ella se basa en dos modelos de enseñanza-aprendizaje (i.e. modelos didáctico y de comunicación) y hace un análisis de software tutorial, de simulación y de tutores inteligentes, basado en el grado de control que el usuario (alumno) tiene sobre los siguientes componentes del diseño del software:

- estrategia de aprendizaje implementada en el software,
- la manipulación de los contenidos de aprendizaje y
- la descripción de los contenidos.

Otro enfoque para clasificar software que puede ser incluido en esta categoría fue propuesto por Chandler [14], el distingue las siguientes categorías:

- Tutorial
- Juego
- Juego de simulación
- Simulación experimental
- Herramientas libres de contenido
- Lenguajes de programación.

III) Por paradigma educacional

Esta clasificación incluye cuatro paradigmas:

- **Instruccional**, por ejemplo, software de ensayo y error, asociado a una perspectiva conductivista.
- **Revelatorio**, por ejemplo, simulaciones, asociado a un aprendizaje por descubrimiento o experimentación.
- **Conjetural**, por ejemplo, programación, asociado a la aplicación de constructivismo y otras visiones cognitivas de uso y desarrollo de software
- **Emancipatorio,** por ejemplo procesadores de texto, asociado a reducir la carga de trabajo, de tal forma que la enseñanza y aprendizaje pueda ocurrir sin incurrir en el consumo de tiempo del procesamiento de datos.

Esta clasificación fue propuesta inicialmente por Kemmis *et al.* [15]. También es mencionada por Squires y McDougall [12] y por Anderson *et al.* [16] quienes proponen una simplificación de esta propuesta basados en las ideas gemelas de interacción alrededor del computador e interacción con el computador. Crook [17] en una línea similar propone dos categorías: cerrado (bajo en control por parte del usuario) y abierto (alto en control por parte del usuario).

IV) Por uso

"Fatouros *et al.* [18] ofrecen una clasificación de CAL basada en el uso del software, con especial atención al rol del profesor de determinar cómo se puede utilizar el software para producir el aprendizaje, en particular, en niños jóvenes. Ellos dicen (p. 186):

Si el software es utilizado en un centro de aprendizaje para que los niños lo exploren libremente como una actividad planificada en el marco de una unidad de trabajo, las decisiones del profesor acerca de su implementación tienen un impacto significativo en el proceso de aprendizaje que se genera. Por ejemplo, idioma y habilidades sociales podrían ser desarrolladas a través del uso de un software típico de ensayo y error si la actividad se lleva a cabo de tal manera que promueva la discusión y negociación entre un par de niños usando el software, en lugar de tener a un niño aislado usando el software como una manera de comprobar el nivel de aprendizaje.

Estos autores ofrecen un sistema de la clasificación basado en los dominios relevantes o áreas de aprendizaje que los profesores planean que los niños exploren:

- imágenes

- sonidos
- texto
- cuentos e ideas
- hechos y figuras
- consecuencias

Esas áreas definen en forma general, dominios que extienden las áreas de desarrollo y del curriculum de la educación de la niñez temprana; y enfatizan la naturaleza integrada del aprendizaje de los niños."

Este tipo de clasificación también fue utilizada por Watson [19], quien se focalizó en la 'Actividad Educacional' que es apoyada por el software, el usa categorías diferentes pero el principio subyacente es similar. Clasifica el software en:

Actividad Educacional	**Contribución de la Tecnología**
Recolección de Información	Correo electrónico, Sistemas Expertos, Bases de Datos, CD-ROM
Análisis y Evaluación	Planillas de cálculo, Bases de Datos, Software de modelación
Presentación	Procesadores de texto, Publicadores, Software de presentaciones, Multimedios, Teletext, Gráficos

Self [20] también clasifica software basado en el rol que el computador debería tener en la escuela. El incluye las siguientes categorías:

- Provocando motivación
- Proveyendo nuevos estímulos
- Activando la respuesta del alumno
- Entregando información
- Estimulando la práctica
- Secuenciando el aprendizaje
- Proveyendo recursos

V) Por impulsos de aprender

Esta categoría se basa en una taxonomía propuesta por Bruce en el publicación interna 'Educational Technology: Tools for Inquiry, Communication, Construction, and Expression' [21] donde entrega una tipificación exhaustiva de géneros de recursos educativos (incluyendo software). Usa las maneras en que apoyan el aprendizaje integrado, basado en preguntas como código de clasificación. Define cuatro categorías amplias:

- Pregunta
- Comunicación
- Construcción
- Expresión

Resumiendo las diversas formas en que se ha intentado clasificar el software educativo, se puede establecer que dichas clasificaciones se han basado en:

- Los contenidos y temas (por tema)
- La funcionalidad implementada en el software (por tipo)
- El paradigma de aprendizaje imbuido en el software (por paradigma educacional)
- La estrategia enseñanza que puede ser provocada por el software o que ha sido imbuida en el diseño del software (por uso)
- La relación que un usuario puede establecer con el software o la necesidad educativa que el software intenta satisfacer (por impulsos de aprender).

Cada clasificación sirve para un propósito de análisis y comparación y se puede utilizar para uno u otro propósito específico. Por ejemplo, si el objetivo es construir una biblioteca de software que sea consultada por profesores, se puede usar la clasificación por área, si el objetivo es comparar los efectos del software en el rendimiento de los estudiantes, entonces se puede usar el paradigma educacional.

Ninguna de estas formas de clasificación ha sido ideada para abordar el tema del diseño de software educativo, por lo que, focalizándose en este aspecto, proponemos una clasificación alternativa que comprende tres grupos:

- El primer grupo incluye software educativo que se ha diseñado considerando que el usuario final será el alumno y que el software tendrá cierto efecto sobre dicho usuario, ya sea que lo use en forma individual o grupal. Ejemplos de este tipo de productos son Tutores Inteligentes, Aprendizaje Asistido por Computador, Resolución de Problemas, Software de Modelamiento, Ambientes de Autoría, etc. En general, llamaremos al

software incluido en este grupo 'herramientas cognitivas' o refiriéndonos al estilo del software, **diseño de software centrado en el aprendizaje**. Las características comunes de este grupo son que se asume que el software será usado por estudiantes en actividades ya sea individuales o grupales; y que el software tiene una teoría del aprendizaje imbuida, junto con un conjunto de propuestas acerca de cómo se debe usar para aprender, derivadas de dicha teoría.

- El segundo grupo corresponde al software que se ha diseñado focalizándose ya no en un usuario específico, sino en un ambiente de trabajo, en este caso, en el aula. Esto es, el software incluye un método de la enseñanza específico en su diseño y por eso se ha concebido como un 'organizador del aula' o refiriéndonos al estilo del software, **diseño de software centrado en la enseñanza**. Ejemplos de software de este grupo son organizadores de discusión, software que organizan y soportan actividades grupales, herramientas que organizan y estructuran presentaciones, etc. Este grupo tiene una propuesta didáctica imbuida en el software, que se utilizó como un paradigma para el diseño, en lugar de un conjunto de sugerencias para utilizar el software.

- El tercer grupo corresponde al software que se ha diseñado como una herramienta general o recurso que se puede usar de diferentes maneras. Este género de software no tiene supuestos pedagógicos explícitos imbuidos, más bien se ha concebido como un 'material de enseñanza/proveedor de recursos'. Ejemplos de software de este grupo son procesadores de texto, enciclopedias, CD-ROMs de contenidos, hojas de cálculo, software de correo electrónico, etc.

A pesar de que esta clasificación no es exhaustiva y por lo tanto tiene exepciones, será útil a nuestro propósito de análisis. Estos tres grupos representan diferentes tendencias de conceptualizar el rol de software educativo durante su etapa de diseño en el marco de la actividad de enseñanza y aprendizaje.

El análisis a continuación profundizará algunas de las ideas respecto a cada uno de los grupos presentados, aludiendo a los aspectos más relevantes tratados en la literatura.

HACIA UN NUEVO DISEÑO DE SOFTWARE EDUCATIVO

En esta sección proponemos una nueva forma de clasificar el software educativo que implica una nueva perspectiva de diseño de este tipo de productos. La clasificación propuesta se basa en los principios ya sea de aprendizaje o enseñanza que los autores hacen explícitos en el producto de software en desarrollo.

Herramientas cognitivas o Diseño de software centrado en el aprendizaje

En este grupo el diseño de software es visto como una actividad cuya meta es producir una herramienta que, se espera, tenga un efecto o impacto, a nivel cognitivo. En otras palabras, se basa en una teoría del aprendizaje que proporciona el marco conceptual del diseño del software. Estas teorías incluyen conductivismo, constructivismo y otras, todas ellas tienen un elemento claro de auto-aprendizaje y por eso presentan supuestos similares en el diseño. Este grupo también incluye software que incorpora algunas estrategias de enseñanza, pero generalmente están imbuidas en un marco de alguna teoría de aprendizaje que está explícito en el software. Desde la perspectiva del diseñador el aprendizaje ocurre a través de la interacción del usuario con el computador. Este grupo de software se puede representar a través de la siguiente figura:

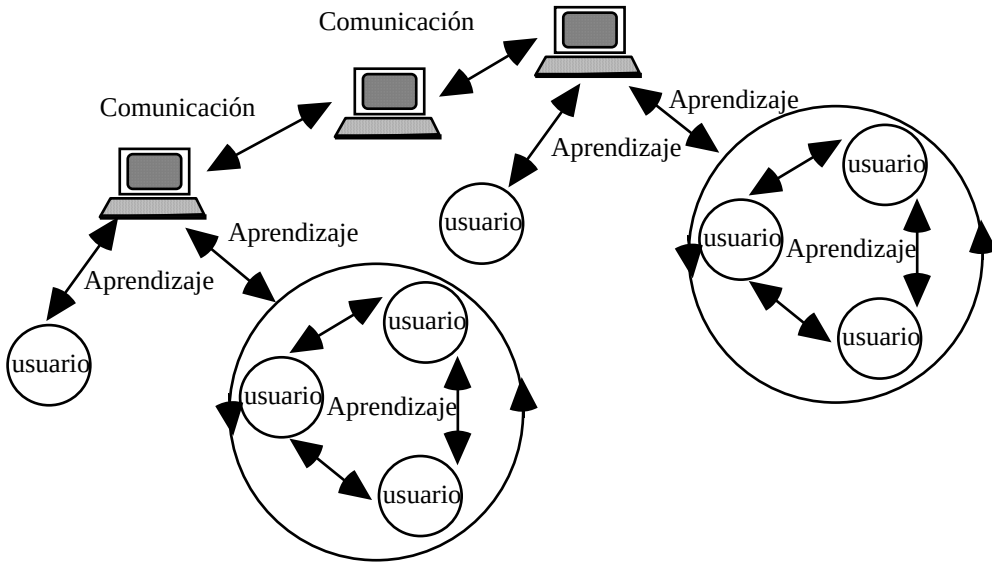

Figura 1. En este esquema los alumnos interactúan con el software y mientras esta interacción ocurre ellos 'aprenden'. El foco de aprendizaje está en la interacción con y/o alrededor del computador.

Reusser [22] tiene software de este grupo en mente cuando describe software educativo como:

"ambientes computacionales deben ser vistos como expandidores de mentes o herramientas catalizadoras para aprendices inteligentes y resolvedores de problemas cuasi autónomos. Deben proveer estructuras estimulantes y facilitadoras para potenciar actividades de construcción de significados, tales como planificación, representación y reflexión" (p. 146)

Otro ejemplo de este tipo de diseño es dado por Laurillard [13]. Ella describe dos modelos de enseñanza y aprendizaje: el modelo didáctico y el de comunicación. En el primero habla sobre un 'conocimiento de preceptos' que es transmitido por el profesor al estudiante. En el segundo describe conocimiento como un 'bien (commodity) negociable' entre profesor y alumno.

Basándose en este modelo, ella especifica los siguientes requisitos de software:

- el estudiante debe tener acceso directo al domino del objeto
- el software debe tener conocimiento operacional del domino
- el software debe poder dar retroalimentación intrínseca
- el software debe hacer las metas del ejercicio explícitas.

Como en el caso previo, esta definición excluye al profesor en el proceso de aprendizaje, que no es el caso de todo el software incluido en este grupo. Otras definiciones corresponden a las categorías mencionadas antes e incluyen la clasificación del computador como herramienta, como aprendiz y a través de algunos de los paradigmas educativos (instruccional, revelatorio y conjetural).

Desde nuestra perspectiva el problema principal de este tipo de software es que los supuestos que se hicieron durante el diseño se basan en teorías de aprendizaje por lo que se asume que el computador se usará de una manera específica para producir el efecto diseñado. Este supuesto obliga al profesor a actuar de una manera particular, de tal forma de crear la situación en la cual el alumno pueda interactuar con el software tal y como el diseñador lo especificó y con esto ocurra el aprendizaje deseado.

Ver nota *

---

* Es posible tener un segundo nivel de clasificación dentro de esta categoría. Dicho nivel sería definido utilizando las diferentes teorías de aprendizaje, instrucción y cognición. Por ejemplo:

- Teorías de aprendizaje:
  - ⇒ Constructivismo
  - ⇒ Conductivismo
- Teorías cognitivas
  - ⇒ Derivadas de Vygotsky
  - ⇒ Derivadas de Piaget
- Teorías instruccionales
  - ⇒ Bruner
  - ⇒ Gagne
  - ⇒ Derivadas de Vygotsky
- Modelos comunicacionales
- Control del aprendizaje por parte de alumnos

Wilson & Cole [23] presentan a una clasificación similar de modelos cognitivos de aprendizaje.

Organizadores de aula o diseño de software centrado en la enseñanza

El diseño del software de este grupo tiene su orígenes en métodos de enseñanza particulares; esto es, se ha concebido como un apoyo a la organización de las actividades del profesor en el aula. La diferencia esencial entre este grupo de software y el anterior, es que el software diseñado tiene uno (o varios) supuestos explícitos (s) de cómo usar el computador en el aula.

Esta forma de diseñar software educativo integra el computador en una estrategia de enseñanza, dándole al profesor un rol especial en las actividades. Dicho rol se hace explícito en el diseño del software. En este caso el aprendizaje ocurre durante la actividad en el aula, no necesariamente, en la interacción con el computador; de hecho, a menudo el software se diseña de tal forma que el alumno no necesita usar el software. El software de este grupo se puede representar a través de la siguiente figura :

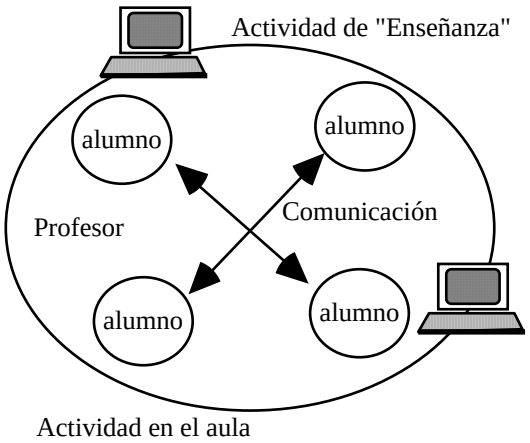

Figura 2. En este esquema los alumnos son parte de una actividad de aprendizaje y los computadores toman el rol de apoyar dicha actividad. El aprendizaje ocurre a través de las actividades en el aula.

Fraser *et al.* [24] describe los diferentes roles que el profesor, alumnos o computador adoptan cuando utilizan software en una situación de aula. Los diferentes roles descritos son (p. 212):

- Gerente (táctico), corrector, evaluador, operador del computador
- Planificador de la tarea, preguntador, ejemplificador, estratega
- Explicador, demostrador, escenificador, constructor de la imagen, focalizador, imitador, reglamentador, entrenador

- Juez, consejero, auxiliador, abogado del diablo, dador de coraje, estimulador, oyente/soporte, observador, receptor, diagnosticador, resolvedor de problemas.
- Alumno compañero, aplicador de reglar, hipotetizador, resolvedor de problemas
- Recurso, sistema a explorar, dador de información.

Estos roles reflejan al comportamiento del profesor y/ o alumnos mientras se usa un software durante una clase en el aula y podrían ser un buen punto de partida para pensar en el diseño de software en este grupo.

Un ejemplo de este tipo de enfoque es dado por Dockterman [25] en una descripción de software que fue diseñado para ser utilizado en una situación de una sala de clases con un solo computador.

Mercer [26, pp. 31-32], describiendo las implicaciones del contexto en el aprendizaje, escribe:

"1 Implica que el proceso de aprendizaje acerca de, o a través de, computadores, esencialmente no tiene que ver con la relación entre un aprendiz/usuario y la máquina -la interfaz- ni siquiera con el software que se esté usando. En cambio, tiene mucho más relación con el marco contextual en el cual el aprendiz/usuario hace cosas con el computador ...

2 Implica eso que lo que es aprendido por algunos niños en particular a través del uso de computadores sólo puede ser comprendido en términos de la relación enseñanza-aprendizaje en la cual dicho aprendizaje ocurrió ..."

Con esta definición, Mercer cambia el foco del diseño de software de la interacción del alumno-máquina al contexto de uso. Pero todavía asume una cierta cantidad de interacción entre el alumno y el computador, que no es requerida en nuestra descripción de software correspondiente a este grupo.

Apoyo adicional a la idea de que el software educativo tiene un rol en contexto del aula, en lugar de en la interacción individuo-computador se encuentra en el trabajo de Olson [27] quien halló dos maneras diferentes en que los profesores utilizan el computador:

- Caballo de Troya : El computador se usa como una ayuda para innovar en la estrategia de enseñanza.
- Herramienta de expresión: El computador es un instrumento para expresar cómo quieren ser vistos como profesores.

Estos hallazgos muestran dos papeles diferentes del computador como un recurso de enseñanza en el sentido que provee al profesor un apoyo para el ejercicio de su trabajo. En este

caso el apoyo no está directamente relacionado a la actividad específica de enseñanza, pero sí al desempeño profesional del profesor.

Material de enseñanza/proveedor de recursos

En este grupo de software incluimos aquellos productos que se pueden ver como software multipropósito, que sirven básicamente como un recurso para llevar a cabo una tarea específica (similar a un diccionario, una enciclopedia, una novela, un lápiz, una regla o un compás). Este tipo de software no incluye una estrategia de aprendizaje o enseñanza explícita, pero ayuda a ejecutar procesos de aprendizaje y/o enseñanza.

El aprendizaje podría ocurrir tanto a través de la interacción del alumno con el software como a través de la actividad organizada por el profesor. El computador es conceptualizado como una herramienta especial para ejecutar alguna actividad o como un poderoso recurso similar a un libro.

Hay muchas experiencias que reportan el uso de software 'tradicional' en escuelas, tales como procesadores de palabras, bases de datos, hojas de cálculo, enciclopedias, etc. Software que, generalmente, ha sido diseñado para ser utilizado en otros contextos de trabajo (industria, administración, bibliotecas, hogar, etc.) y ha sido 'introducido' al contexto de las salas de clases. Profesores que utilizan este tipo de software a menudo apelan a razones de tipo 'vocacional', argumentando que los estudiantes necesitan prepararse para usar este tipo de software cuando entren al mercado laboral.

DISCUSIÓN

Dejando de lado el uso de software como material para la enseñanza/proveedor de recursos, en las secciones anteriores hemos argumentado que el diseño de software educativo sigue una de dos tendencias:

- Diseño de software centrado en el aprendizaje: Aquí los autores tratan de construir software que implemente alguna teoría de aprendizaje, cognitiva o instruccional. Al hacerlo, confieren al computador un alto grado de responsabilidad de los resultados del aprendizaje.
- Diseño de software centrado en la enseñanza: Aquí los autores tratan de hallar maneras en que el computador puede ser utilizado como una herramienta que apoye del proceso de enseñanza. Asumen una mirada más sistémica del proceso de enseñanza-aprendizaje, en lugar de una concepción particular de aprendizaje con o alrededor del computador.

Con respecto a la primera tendencia, independientemente de las intenciones del diseñador, algunas investigaciones muestran que el software que se utiliza con mayor frecuencia en las

escuela se basa en actividades tipo ensayo y error [3]. Con respecto a la segunda tendencia la pregunta clave que no ha sido resulta es: ¿cuál es el rol del computador en la enseñanza ?

A pesar de las altas expectativas acerca del uso de computadores en educación, algunas investigaciones han mostrado que en este campo el papel de la tecnología es polémico [1] y su efecto no es concluyente [2]. Al tratar de explicar esta situación hay dos corrientes principales de argumentos, la primera afirma que el profesor debe estar más y mejor instruido en el uso de la tecnología para así poder dominarla, y la segunda corriente se refiere a la falta de comprensión de los diseñadores de software acerca del proceso de enseñanza/aprendizaje.

En la primera corriente de argumentos, algunos autores se quejan de la capacidad de los profesores de entender y/o adaptar productos de software a sus actividades en el aula. Beynon y Mackay [28] escriben que el ser instruido en tecnología demanda más que la capacidad de poder usar tecnología (*hands-on*), también demanda la habilidad de 'leerla'. Este concepto de leer la tecnología es muy complejo y no tiene respuesta simple, especialmente en educación. También en esta corriente, Handler [29] y Winship [30] se quejan acerca de la falta de instrucción apropiada y apoyo para profesores que quieren usar esta tecnología.

Otros problemas con software informados por Winship [30] que reflejan una postura más bien intermedia son:

- a los profesores les parece muy difícil identificar software que crean que les será útil en su propia enseñanza.
- muchos productos de software existes son difíciles de integrar a la enseñanza ya sea porque es demasiado fácil, demasiado difícil o toma demasiado tiempo antes de poder apreciar resultados útiles.
- a menudo los profesores deben invertir mucho tiempo de preparación antes de que software se pueda utilizar en el aula.

En general parece haber una disociación entre lo que se ofrece hoy día como buen diseño de software educativo, lo qué los profesores verdaderamente hacen con software en las escuelas y las expectativas de los profesores de lo que se puede hacer con software en las escuelas.

Así, una crítica general al diseño de software educativo a la fecha es que existe un falta de comprensión de lo que ocurre en el aula y del discurso del profesor y su realidad [22, 27, 31-34]. Mercer y Scrimshaw [33] y Olson [27] argumentan que sabemos demasiado poco acerca de las actividades en el aula con los computadores. Crook [31] y Koedinger y Anderson [32] argumentan que debemos entender el discurso de enseñanza y el contexto instruccional. Winograd y Flores [34] hablan acerca de la necesidad de comprender el dominio de acción

del usuario, en este caso el profesor y el alumno. Reusser [22] habla sobre la filosofía pedagógica y didáctica que el diseño de software debe incorporar y de la importancia de las actividades de enseñanza y aprendizaje que ocurren en el escenario de acción (behavioural setting) de la instrucción.

El software escolar (y no educativo) se debe diseñar para ser usado en la escuela, con propósitos y para necesidades que estén presentes en la escuela. La implicación de esta definición es que para diseñar un producto de software escolar, primero es necesario conocer las necesidades de la escuela y de éste punto de partida, diseñar un producto de software que ayude a satisfacer dicha necesidad, en otras palabras, comprender el papel que el software puede jugar a en esta solución dentro de este contexto. Implica no imponer diseños de software preconcebidos para mejorar las actividades profesionales de los profesores, si no diseñar, basado en sus prácticas actuales, productos de software que les permita hacer lo qué hacen hoy en forma más efectiva y eficiente.

AGRADECIMIENTOS

Este trabajo ha sido realizado gracias al apoyo de Fondecyt, proyecto número 1960854, del Ministerio de Educación a través del Programa MECE en el proyecto Enlaces y al Consejo Británico, a través del vínculo académico 'Telematics Partnership Between Chile and UK'. 3

REFERENCIAS

1 LOWTHER, D. Y H.J. SULLIVAN. (1994). *Teacher and technologist believes about educational technology*. Educational Technology Research and Development, **42**(4): p. 73-87.

2 JOHNSON, D.C., M.J. COX, Y D.M. WATSON. (1994). *Evaluating the impact of IT on pupils' achievements.* Journal of Computer Assisted Learning, (10): p. 138-156.

3 EVANS-ANDRIS, M.. (1995). *An examination of computing styles among teachers in elementary schools.* Educational Technology Research and Development, **43**(2): p. 15-31.

4. DISESSA, A.A., C. HOYLES, Y R. NOSS. (1995). ed. *Computers and exploratory learning*. NATO ASI Series F Subseries Advanced Educational Technology, NATO. Springer: London.

5. LABORDE, J.-M.. (1995). ed. *Intelligent Environments: The case of geometry*. NATO ASI Series F Subseries Advanced Educational Technology, NATO. Springer: London.

6. MELLAR, H.. (1994). *et al.*, ed. *Learning with artificial worlds: computer based modelling in the curriculum*. The Falmer Press: London.

7. SCHWARTZ, J.L.(1996). *Motion toys for eye and mind.* Communication of the ACM, 1996. **39**(8): p. 94-96.

8. SCHWARTZ, J.L., M. YERUSHALMY, Y B. WILSON. (1993). ed. *The geometric supposer: what is the case of?* Technology in Education, R.S. NICKERSON, Erlbaum: London.

9. SOLOWAY, E. Y A. PRYOR. (1996). *Using computational media to facilitate learning.* Communications of the ACM. **39**(8): p. 83-109.

10. LEARNING, *Learning with Software*1995, Open Learning Technology Corporation Limited: www: http://gwis2.circ.gwu.edu:80/~kearsley/.
11. TAYLOR, R.P., ed. *The Computer in the School: Tutor, Tool, Tutee*. 1980, Teacher College Press: New York.
12. SQUIRES, D. Y A. MCDOUGALL. (1994). *Choosing and using educational software: a teachers' guide*. London: The Falmer Press.
13. LAURILLARD, D., *Computers and the emancipation of students: giving control to the learner,* in *Computers and learning,* O. BOYD-BARRET Y E. SCANLON, (editores, 1990), Addison Wesley & The Open University: Wolinkham. p. 64-80.
14. CHANDLER, D., *Young learners and the microcomputer*. 1984, London: Milton Keynes, Open University.
15. KEMMIS, S., R. ATKIN, Y E. WRIGHT, *How do students learn?*1977, Centre for Applied Research in Education, University of East Anglia, UK:
16. ANDERSON, A. (1993). *et al., Software style and interaction around the microcomputer.* Computers and Education. **20**(3): p. 235-250.
17. CROOK, C., *Computers in the classroom: defining a social context,* in *Computers, cognition and development,* J. RUTKOWSKA Y C. CROOK, (editores, 1987). John Wiley & Sons: Chichester. p. 35-53.
18. FATOUROS, C., T. DOWNES, Y S. BLACKWELL. (1994). *In control: Young children learning with computers*. Wentworth Falls: NSW: Social Science Press.
19. WATSON, L., *Appropriate tools ? IT in the primarand classroom,* in *Computers into classroom: more questions than answers,* J. BEYNON Y H. MACKAY, (editores, 1993), The Falmer Press: London. p. 78-91.
20. SELF, J. (1985). *Microcomputers in education: a critical appraisal of educational software.* Brighton: The Harvest Press. Ltd.
21. BRUCE, B.C. (1996). *Educational technology: tools for inquiry, communication, construction, and expression.* www: http://www.ed.uiuc.edu/EdPsy-387/Ed-Tech-Taxonomy.html.
22. REUSSER, K., *Tutoring systems and pedagogical theory: representation tools for understanding, planning and reflection in problem solving,* in *Computers as cognitive tools,* S.P. LAJOIE Y S.J. DERRY, (editores, 1993), Lawrence Erlbaum: Hillsdale. p. 143-177.
23. WILSON, B. Y P. COLE. (1991). *A review of cognitive teaching models.* Educational Technology Research and Development. **39**(4): p. .47-64.
24. FRASER, R., *et al., Learning activities and classroom roles with and without the microcomputer,* in *Computers and learning,* O. BOYD-BARRET Y E. SCANLON, (editores, 1991), Addison Wesley & Open University: Wokingham.
25. DOCKTERMAN, D.A.(1991). ed. *Great teaching in the one computer classroom*. Tom Snyder Productions: .
26. MERCER, N., *Computer-based activities in classroom contexts,* in *Language, classrooms and computers,* P. SCRIMSHAW, (editor, 1993). Routledge: London. p. 27-39.
27. OLSON, J. (1988). *Schoolworlds/microworlds: computers and the culture of the classroom.* Oxford: Pergamon Press.
28. BEYNON, J. Y H. MACKAY. (1993). ed. *Computers into classroom: more questions than answers*. The Falmer Press: London.

29. HANDLER, M. (1993). Preparing new teachers to use computer technology: perceptions and suggestions for teacher educators. Computers & Education. **20**(2): p. 147-156.
30. WINSHIP, J.A. (1989). Information technologand in education: the quest for quality software. Paris: Organisation for Economic Co-operation and development.
31. CROOK, C. (1991). Computers in the zone of proximal development: implications for evaluation. Computers in Education. **17**(1): p. 81-91.
32. KOEDINGER, K.R. Y J.R. ANDERSON, Reifying implicit planning in geometry: guidelines for model-based intelligent tutoring systems design, in Computers as cognitive tools, S.P. LAJOIE Y S.J. DERRY, (editores, 1993), Lawrence Erlbaum: Hillsdale. p. 15-45.
33. MERCER, N. Y P. SCRIMSHAW, Researching the electronic classroom, in Language, classrooms and computers, P. SCRIMSHAW, (editor, 1993). Routledge: London. p. 184-191.
34. WINOGRAD, T. Y F. FLORES. (1986). Understanding computers and cognition: a new foundation for design. Addison-Wesley Pub. Co.