Informática Educativa
Proyecto SIIE, Colombia

Vol . 9 , No. 1, 1996
pp. 9-32

UN MÉTODO DE DESARROLLO DE SOFTWARE EDUCATIVO

Enrique HINOSTROZA S.
Pedro HEPP K.
Pablo STRAUB B.

RESUMEN

El objetivo de este trabajo es describir un método para desarrollar software educativo. El método descrito se está aplicando y perfeccionando en el proyecto *Enlaces*, actualmente en ejecución en la Universidad de la Frontera y que es parte del programa MECE del Ministerio de Educación de Chile. El proyecto *Enlaces* está instalando una red de escuelas primarias y secundarias urbanas y rurales en Chile, entre los años 1993 y 2000.

Aunque el método se alimenta de la experiencia en Ingeniería de Software para desarrollar sistemas de información, está orientado específicamente a la producción de software educacional, para el trabajo entre niños y entre profesores y niños de educación general básica. Por esto, comprende aspectos no sólo de Ingeniería de Software, sino que también de Pedagogía, Psicología y Diseño Gráfico. Cada una de estas disciplinas tienen su aporte y sus propios mecanismos de evaluación que deben complementarse e integrarse [i].

Los principales aportes del modelo presentado son, la incorporación de evaluaciones tempranas del producto, la estructuración de las actividades a realizar para desarrollar software, separando claramente los roles de las diferentes profesiones que participan del desarrollo y las herramientas de documentación, control y organización que guían y apoyan el desarrollo.

INTRODUCCIÓN

Enlaces es un proyecto de informática educativa del Ministerio de Educación de Chile, cuyo objetivo es determinar los beneficios, contenidos, costos y replicabilidad de las iniciativas en torno a la informática educativa y redes educacionales. El proyecto incorpora mecanismos de evaluación de su impacto y busca preferentemente determinar los roles de la tecnología computacional y de telecomunicaciones en los liceos y las escuelas municipales y particular subvencionadas, focalizando en las de menores recursos. El Proyecto Enlaces es parte del Programa MECE del Ministerio de

Educación, que es un esfuerzo integral para mejorar la equidad y calidad de la educación en Chile [1].

El Proyecto *Enlaces* comenzó oficialmente en marzo de 1993 y a mayo de 1996 alcanzó la conexión de, aproximadamente, 200 establecimientos educacionales a través de la red INTERNET. Dichos establecimientos están localizados en 7 regiones de Chile, aunque principalmente en la IX región de la Araucanía. Adicionalmente, se estableció una sólida base técnica para permitir el crecimiento descentralizado a partir de 1996, en que se integrarán aproximadamente 300 nuevos establecimientos al Proyecto.

Cada uno de los establecimientos recibe equipamiento, software educativo, capacitación inicial y asistencia técnica periódica ya sea durante visitas, por medio de la misma red ya través de material de apoyo. El equipamiento es incorporado gradualmente, focalizando en el trabajo del profesor.

El proyecto *Enlaces* ha desarrollado el software “La Plaza", un programa con una interfaz humano-computador muy simple de aprender a usar, que permite acceder al computador, ya sea para utilizar *software* educativo y *software* de apoyo a la administración escolar o bien para aprovechar las comunicaciones, participando en proyectos educativos y entregando y obteniendo información a través de la red INTERNET.

Como se ha dicho, una parte del material que se les entrega a las escuelas de la red es *software* educativo. Este *software* es, en su mayoría, construido en el Centro de Informática Educativa, debido a la falencia en el mercado de productos en español que satisfagan las necesidades particulares de estas escuelas.

Este trabajo es una sistematización de la experiencia adquirida durante los años 1991 a 1995 en la construcción de al menos 30 productos de *software* multimedial. De estos, 10 han sido entregados para evaluación y uso en las escuelas de la red.

Primero, el trabajo presenta un resumen del método de desarrollo, luego contiene una descripción del modelo de desarrollo, que incluye una descripción del grupo de desarrollo, los principios subyacentes y una descripción de cada una de las etapas, las cuales son ilustradas utilizando un ejemplo concreto. Luego se presentan las conclusiones con un análisis del método, para finalizar con los agradecimientos y referencias.

---

[1] Para mayor información ver http://www.enlaces.ufro.cl/

RESUMEN DEL MÉTODO

El método para desarrollo de software descrito en este trabajo es el resultado de adaptar, modificar y ampliar modelos teóricos a necesidades reales [ii] [2]. Estas necesidades surgen del trabajo interdisciplinario, de la incorporación de multimedios y la focalización en educación.

El método propuesto consta de 4 actividades principales, algunas de las cuales se realizan en paralelo. El método se basa en el desarrollo incremental de un prototipo [iii, iv, v] en que en todas las etapas hay una evaluación crítica, modificaciones y redefiniciones, buscando converger hacia un producto evaluado en terreno. Las cuatro actividades principales son la Definición del Proyecto, el Diseño de la Aplicación, el Desarrollo de Prototipos y por último la Construcción del Producto. A continuación se resume cada una:

1. *Definición del Proyecto*, cuyo objetivo es hacer explícito, acordar y documentar:

- Objetivos y contenidos docentes del producto (qué y por qué),
- Caracterización de los usuarios, considerando a los alumnos y profesores (quién),
- Ambiente de uso, incluyendo lugar y circunstancia (dónde y cuándo),
- Recursos de desarrollo, incluyendo personas, equipos y tiempo (cómo),
- Recursos de uso, principalmente equipos (con qué).

2. *Diseño* (softectura) [3] *de la Aplicación* [vi], cuyo objetivo es definir la estructura externa o interfaz de la aplicación. Definimos la softectura usando bocetos de las diferentes partes de la interfaz, y diagramas que llamamos “maquetas” que muestran las relaciones entre los bocetos. La softectura de una aplicación nos ayuda a revisar sus objetivos.

---

[2] Colin POTTS, en su artículo “A Software Engineering Research Revisted”, plantea que hoy en día existe una tendencia emergente a utilizar la industria como laboratorio para descubrir nuevos métodos de desarrollo de software y que esta tendencia hace énfasis en la relevancia de la definición empírica de problemas, estudio de casos reales y aspectos contextuales. Este modelo es uno de esos casos.

[3] “Softectura” es una palabra acuñada por GILB (1988) que corresponde, por analogía con “arquitectura”, al diseño de software como lo ve el usuario. La softectura no incluye la estructura interna del software, sino sólo la estructura externa de la interfaz.

3. *Desarrollo de Prototipos*, cuyo objetivo es asegurarse que los objetivos del producto, recursos, softectura, etc. son alcanzables. Se desarrollan sucesivos prototipos del (partes de) producto final, revisando las ideas de las actividades 1 y 2. Estas revisiones contemplan la participación de futuros usuarios (alumnos y/o profesores).
4. *Construcción del Producto,* cuyo objetivo es desarrollar un producto terminado, bien Luego de converger evaluado y completo desde un punto de vista comercial (manuales, envoltorios, etc.). a un prototipo final, se comienza el proceso de definición de los elementos adicionales que requiere un producto para salir al mercado, es decir, el proceso de Marketing. Este involucra, manuales, publicidad, precios, distribución, soporte y actualizaciones.

Cada etapa está compuesta por una serie de sub-etapas que se explican en la siguiente sección a través de diagramas.

EL MODELO DE DESARROLLO

En esta sección se explican tres dimensiones del modelo de desarrollo, la primera es el grupo humano que desarrollará el producto (Grupo de Desarrollo), la segunda corresponde a un conjunto de principios que inspiran y enmarcan el diseño e implementación del software. La tercera dimensión es la secuencia de actividades o pasos que comprende el modelo.

EL GRUPO DE DESARROLLO

El grupo de personas que desarrollará la aplicación como un proyecto, debería participar desde el inicio de las reuniones, esto permitirá generar y mantener una "cultura" del grupo de desarrollo. Este grupo idealmente está constituido por un *jefe de proyecto*, responsable de la administración y coordinación de las tareas del grupo de desarrollo. Un *educador* y/o *psicólogo educacional*, a cargo de las fuentes de contenidos, las características del usuario, los aspectos motivacionales, la calidad educativa de los contenidos, el proceso de evaluación de la aplicación, etc. Un *ingeniero de software*, a cargo del diseño lógico y de la programación de la aplicación como constructo de ingeniería. Un *diseñador gráfico*, responsable de los recursos gráficos que se utilizarán en la interfaz humano-computador de la aplicación, como la diagramación de la interfaz, la simbología de la iconografía, el uso de colores, etc. Un *experto* en el tema, responsable de los contenidos y el método pedagógico utilizado en

la práctica. Puede ser el mismo educador u otra persona que sabe y ha enseñado el tema a tratar.

Los roles específicos de cada persona van cambiando durante el desarrollo del producto. En las primeras actividades el aporte especializado de cada profesión es menos notorio, ya que se trabaja preferentemente en base a reuniones y discusiones grupales. A medida que se avanza en la definición del proyecto, la especialización se hace necesaria, ya que se requieren criterios especializados que aporten y definan las decisiones. Avanzado el diseño, cada profesión toma un rol claramente definido y cada persona tiene un ámbito de acción concreto. Estos roles se hacen patentes al revisar las actividades del modelo.

La experiencia de este grupo de desarrollo de software será determinante en la estimación de recursos necesarios para el desarrollo.

PRINCIPIOS SUBYACENTES

La experiencia acumulada ha permitido definir un marco de referencia para el diseño e implementación de los proyectos de software. El marco de referencia se traduce en un conjunto de principios que deberán estar presentes en cada uno de los desarrollos. Estos se pueden clasificar en tres áreas: lo *perceptivo*, que se refiere a lo que el usuario podrá percibir al usar el software, lo *metodológico*, es decir, los principios que soportan el diseño de la utilización de la aplicación en el aula y lo *funcional*, es decir, qué podrá hacer el usuario con la aplicación.

Desde el punto de vista de la **percepción**:

- *Sugerente*: se refiere a que el *software* propone al usuario actividades interesantes para él y que puede desarrollar fuera del computador (ej.: construir un artefacto, cuyo diseño y principios se explican en el software). Esto nace de la concepción de las salas de clase de un computador (one computer classroom) en que se desprotagoniza al computador como fin instruccional [vii].
- *Atractivo*: el usuario se siente motivado a utilizar el software, la apariencia de éste es llamativa y conocida. Se presentan metáforas y elementos conocidos para el usuario.
- *Invitante al uso y a la exploración* (no intimidante). Esto incluye sugerencias e invitaciones para observar mejor el medio ambiente del usuario (su casa, escuela, personas que le rodean), en áreas relacionadas con la aplicación. El usuario se siente motivado a utilizar el software y puede hacer la transferencia de contenidos desde el software a su vida (Teoría de Campo de Kurt Lewin en [viii]; [ix]).

- *Relevante:* el usuario encuentra contenidos que le interesan puesto que le son útiles para su vida diaria, tienen que ver con su persona, sus intereses y expectativas. Este principio se sustenta en el concepto de educación pertinente [7].

Desde el punto de vista **metodológico**:

- *Colaborativo:* para trabajo grupal, en grupos geográficamente dispersos (en red) o en el aula. Este principio es coherente con la situación actual de aislación geográfica y social de la educación básica chilena [x]. Además, es coherente con las tendencias de trabajo grupal [xi] y los principios de la zona de desarrollo próximo de Vygotsky [xii].
- *Complementario:* el software complementa algún proyecto o trabajo organizado por el profesor o los alumnos, es decir, no pretende reemplazar al profesor, sino que servirle de apoyo. Este principio apoya la concepción del computador como una herramienta al servicio del profesor, facilitando su integración a la práctica pedagógica [7].

Desde el punto de vista **funcional**:

- *Interactivo:* el usuario tiene el control del software la mayor parte del tiempo, como principio básico del diseño de software de este tipo [xiii, xiv].
- *Entrega un resultado*: el usuario puede llevarse consigo algo producido por el software (Ej.: un hoja impresa con un diseño o un texto o el resultado de su trabajo). Es un aspecto de retroalimentación que se basa en la motivación al logro del usuario, es importante desde la perspectiva sicológica [xv].
- Es una *herramienta*: el usuario puede hacer, construir o producir algo personal con el software, no sólo "consumir" información. Potencia aún más la retroalimentación que el usuario tendrá al utilizar el software [14] .

ACTIVIDADES DEL MODELO

En esta parte se detallan las actividades o pasos que comprende el modelo.

Definición del Proyecto

Esta primera actividad supone que la idea esencial del proyecto ya está concebida, es decir, que existe una necesidad y que alguien (típicamente un profesor) está muy interesado en utilizar la informática para satisfacer, al menos en parte, esa necesidad. Esta actividad pretende descubrir la forma de satisfacer esa necesidad a través de un proyecto de software. Por ejemplo:

Supongamos que en una escuela primaria existe un profesor de Artes Plásticas que siente la necesidad de que sus alumnos conozcan algo de arte y, lo más importante, puedan desarrollar sus habilidades artísticas.

Una vez explicitada esta necesidad en el marco del grupo de desarrollo, en esta etapa inicial del proyecto se utiliza una técnica de reunión denominada "lluvia de ideas" (Brainstorming), para establecer el mayor número posible de escenarios o alternativas de solución de esta necesidad (pensamiento divergente). Una vez que las ideas nuevas no aportan significativamente al conjunto existente, se comienza el proceso de convergencia hacia un consenso del grupo de desarrollo en torno a la aplicación, el tema, el usuario y los recursos.

Los puntos concretos que este grupo de desarrollo deberá comenzar a definir en esta actividad son: la Aplicación en sí, el Usuario y su Medio Ambiente, los Recursos Requeridos y las Fuentes de Información. A continuación se describen en detalle estos puntos.

La Aplicación

Interesa definir y delimitar la aplicación. Esto se logra definiendo el contenido principal y los temas secundarios, su uso, sus limitaciones, la profundidad de los contenidos y los medios posibles a incorporar. Conviene que el grupo acuerde, en una oración, el objetivo de la aplicación. Por ejemplo: "... este software tiene como objetivo que el alumno conozca diferentes ramas artísticas y sus técnicas básicas". La definición de la aplicación se logra paulatinamente, una vez que se han analizado los siguientes puntos:

- La utilidad (pedagógica) de la aplicación. Se debe explicitar su rol en el contexto pedagógico, es decir su relación con los planes y programas oficiales de educación básica. Por ejemplo:

La aplicación podrá ser utilizada para:

(i) que el alumno aprenda contenidos generales sobre las ramas artísticas más comunes (literatura, pintura, música, teatro y escultura). Entre los contenidos generales están los principales representantes chilenos y algunas de sus obras.

(ii) que el alumno conozca y sepa aplicar las técnicas básicas y simples de cada una de ellas. Es necesario hacer notar que la descripción puede cruzar más de un objetivo de determinado nivel.

- La necesidad de utilizar software. Se debe justificar la utilidad de un producto de software. Por ejemplo:

A través del uso de software (multimedios) será posible mostrar a los alumnos ejemplos reales de las diferentes ramas artísticas. Además, será posible describir las diferentes técnicas utilizadas.

- Las oportunidades de uso. Se deben explicitar las diferentes áreas en las que la aplicación podría ser utilizada y los objetivos de tales usos. Por ejemplo:

El profesor de Artes Plásticas tendrá a su disposición una colección de representantes de diferentes ramas artísticas. El podrá organizar actividades de análisis grupales de las obras, de las diferentes corrientes artísticas, etc. Además, a través de las técnicas, el profesor podrá organizar concursos de esculturas, pinturas, etc.

El profesor de Música podrá utilizarlo para que los alumnos analicen obras famosas (óperas, misas, etc.) y puedan aprender a crear una.

El profesor de Castellano podrá utilizar la rama de literatura y teatro para dar a conocer a los principales representantes y organizar representaciones, talleres de poesía, etc.

En este punto es necesario considerar la descripción del medio en que será utilizado el software, ya que es necesario considerar los métodos y técnicas pedagógicas predominantes en las escuelas para definir la forma de uso del software.

- Las limitaciones de la aplicación. Se debe explicitar qué problemas o necesidades no resuelve. Por ejemplo:

El software no reemplaza los contenidos específicos de las asignaturas que apoya (Artes Plásticas, Música, Historia, Castellano, etc.). Por ejemplo, no contiene a todos los representantes del arte que se estudian, no pretende enseñar gramática o conjugaciones verbales, a pesar de que los alumnos la ejercitarán.

La selección y definición del tema debe justificarse, respondiendo satisfactoriamente las siguientes preguntas:

- ¿Porqué abordar el tema? Esta pregunta debe responderse desde la perspectiva pedagógica, argumentando claramente la relevancia y pertinencia dela aplicación en términos de contenidos y actividades propuestas. Un ejemplo de una justificación es:

El arte es un tema pertinente a la educación primaria, ya que en esta etapa de desarrollo el alumno comienza a descubrir y potenciar sus habilidades artísticas.

La disponibilidad de ejemplos de estas obras en localidades alejadas de grandes centros urbanos es muy limitada, por lo que la posibilidad de tenerlos disponibles en una aplicación es de gran ayuda para el profesor y muy motivante para el alumno.

- ¿ Qué alternativas existen (libro, folleto, vídeo, comprar un software similar, lograr los objetivos sin utilizar computadores, etc.)? Desarrollar software es caro, consume recursos humanos, de computador e insumos. Conviene cuantificar estos factores de modo de analizar con suficientes antecedentes, los costos Vs. los beneficios (evaluación social de proyectos). En muchas ocasiones, existen alternativas mejores al uso de software computacional para desarrollar proyectos educativos en las

escuelas o en la casa. En otras ocasiones, el software podría limitarse a complementar, a ser una parte, de un proyecto educativo. Por ejemplo:

En lugar de construir un software que contenga las diferentes ramas artísticas, se puede organizar un proyecto que incluya visitas a museos, plazas y bibliotecas locales y de otras ciudades. Además de trabajar en algún momento un aspecto puntual del proyecto, en el computador, como podría ser escribir una composición de lo visto y aprendido en los paseos.

En este punto, alguien del grupo actúa de "abogado del diablo". Idealmente se invita a una persona ajena al proyecto, de reconocida capacidad crítica y potencial usuario del sistema a opinar al respecto.

La definición del tema o contenido de la aplicación se expresa en términos de una malla (no necesariamente jerárquica) que refleje el espacio global de los temas a tratar. Esta estructura generalmente es un grafo en el cual cada nodo representa un tema o sub-tema y las uniones entre los temas expresan relaciones de tipo: explicación, profundización, componente o secuencia lógica de revisión.

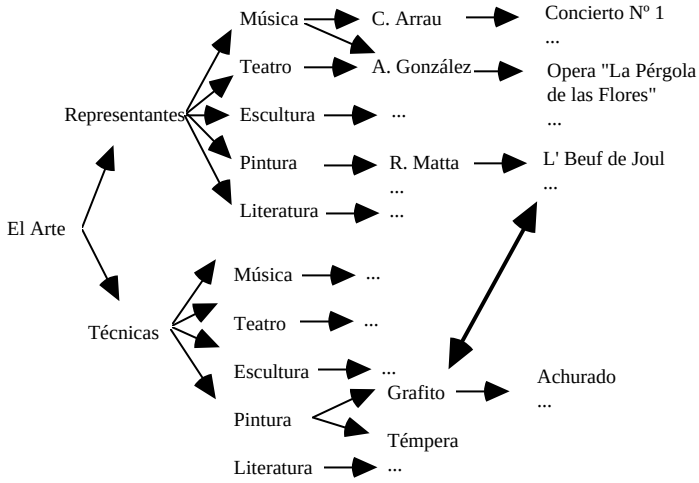

Figura 1. Ejemplo de una malla de temas y sub-temas en proceso.

Tanto el tema principal como cada sub-tema, debe ser explicado, indicando aspectos como con qué profundidad se tratarán los contenidos, con qué medios se cuenta (texto, gráfica, sonido, vídeo, animación).

En esta parte también se discuten los primeros criterios específicos de evaluación del software. Hay criterios generales, aplicables a todo software (por ejemplo: normas ISO/IEC 9126:1991) y hay criterios específicos que deben aplicarse al software educativo [xvi, xvii, xviii].

El Usuario y su Medio Ambiente

La definición del usuario de la aplicación y el medio ambiente en el que vive y la utilizará es determinante para el trabajo de los temas, sub-temas, los recursos gráficos y lingüísticos, aspectos motivacionales y la tarea de recolección de información.

Para apoyar este proceso, se construye una **ficha del usuario tipo** que incluye:

- Descripción sicológica del usuario. Esta descripción incluye aspectos internos (niveles de desarrollo, aspectos cognitivos, capacidades de abstracción, motivaciones, etc.). Estas características determinarán, en parte, la interfaz humano-computador, el estilo de tratamiento de los temas, etc. [12, xix, xx]. Por ejemplo:

> Alumnos y alumnas de 4º a 8º básico, con capacidad de lectura y escritura. En una etapa de desarrollo operacional concreta, motivación de aceptación social, etc.

- Descripción del medio ambiente social y cultural del usuario: casa, escuela, trabajo, familia, tradiciones (por ejemplo, Moffat, [xxi]). Esta descripción aporta en términos del conocimiento que se tendrá respecto a la pertinencia de los contenidos y actividades propuestas. Por Ejemplo:

> Escuelas rurales, nivel socio-económico bajo, con actividades laborales a temprana edad, familias numerosas (con tíos, abuelos, padres y amigos en una misma casa), tradición oral, poca lectura, etc. [xxii].

- Descripción del medio ambiente en el cual se usará la aplicación: escuela, biblioteca, sala de clases, etc. (esto se amplía más en el diseño que en la definición ya que en el diseño se incluyen aspectos metodológicos). El contexto en que es utilizado el software determinará el estilo motivacional, las metáforas a utilizar, las actividades a realizar, etc. [xxiii]. Por ejemplo:

> El estilo de la escuela es más bien tradicional, con organizaciones de salas y bibliotecas en forma de auditorio. El estilo de trabajo de los profesores es expositivo, sin diálogo, por lo que hay dos posibilidades:
>
> La aplicación será usada por grupos de alumnos en los computadores ubicados en la biblioteca o sala de "computación" de la escuela, en horas de clase. Los grupos podrán rotar en el uso del computador durante el período de la clase. O bien: El software será utilizado por el profesor durante la clase de Castellano, Ciencias Sociales y/o Artes Plásticas, con un computador y un proyector de pantalla.

En este punto es útil revisar si existen ciertos patrones que permitan describir el estilo docente predominante en la(s) escuelas que se distribuirá la aplicación. Esto permitirá adecuar de mejor forma las oportunidades de uso [xxiv].

Una vez construida la ficha básica del usuario, se conjeturan las *expectativas* que el usuario podría tener al usar el software. Estas se construyen a partir de la respuesta hipotética a las siguientes interrogantes:

- ¿ qué sabe ya del tema ? Se trata de estimar el contexto que el usuario tendrá al comenzar a recorrer la aplicación. Por ejemplo:

Un niño de una zona rural, conoce probablemente menos sobre el arte que un niño de una zona urbana. Probablemente ambos han escuchado hablar de los representantes más típicos como Neruda, Mistral, Arrau, etc. de las áreas de literatura y música, pero es poco probable que conozcan sobre escultura, teatro y ópera. Además, dada la situación de aislamiento, probablemente no han tenido acceso a ver expresiones artísticas en persona (cuadros, ópera, teatro, etc.). Es probable que desconozcan, en general, las técnicas asociadas.

- ¿ qué le gustaría ver o encontrar ? Se trata de explicitar el espíritu que tendrá este software. Por ejemplo:

Al usuario le gustaría ver piezas de arte que pueda apreciar en detalle, no sólo una descripción de éstas. Le asignará valor a los contenidos más realistas.

En cuanto a las técnicas, esperará ver descripciones que le permitan llevar a cabo tareas, sin necesidad de adquirir elementos especiales. Es decir, esperará poder hacer algo con los elementos de su entorno geográfico y cultural.

El profesor espera encontrar actividades útiles a su práctica pedagógica, que le sea fácil transferir lo propuesto a sus necesidades actuales de enseñanza.

- ¿ qué necesita saber ? Se trata de definir el nivel de conocimiento que se requerirá para utilizar el software. Por ejemplo:

El usuario necesita saber “navegar” utilizando el ratón. No se necesitan conocimientos conceptuales previos.

Las respuestas a estas interrogantes permitirán establecer cuáles son las expectativas que el grupo de desarrollo tiene de la aplicación. Para obtener las respuestas se recurre a instrumentos de evaluación y entrevistas.

Recursos Requeridos

Para determinar los recursos requeridos, se lleva a cabo una primera estimación de los siguientes puntos:

Qué equipos se requieren para el *desarrollo*:

- Computador y monitor
- Periféricos como impresoras, módem, etc.
- Equipos Digitalizadores.
- Programas de diseño gráfico, animaciones y edición de digitalizaciones.

- Programas para integrar los diferentes medios y producir el software.
- Espacio en disco.
- Memoria RAM.
- Velocidad estimada de proceso.
- Repositorios de respaldo (servidores, discos, etc.)

Qué equipos se requieren para **utilizarlo**:

- Computador y monitor
- Periféricos como impresoras, módem, etc.
- Programas para desplegar el software.
- El espacio en disco

Tiempo estimado de desarrollo:

- Definición, incluyendo: tema, usuario, medio, recursos y recolección de información.
- Diseño, considerando cada iteración.
- Desarrollo de prototipos. En general, depende de la factibilidad y celeridad de las pruebas con usuarios.
- Construcción de un producto. Esta estimación es muy relativa, depende de la redefinición -diseño y prototipos- y las necesidades de ampliación y cambio.

Estas estimaciones aportan criterios de evaluación de la factibilidad técnica del desarrollo, pero dependerán de la experiencia del grupo de diseño/desarrollo, la disponibilidad de información y recursos para el desarrollo[xxv].

Fuentes de Información

Se trata de recolectar la mayor cantidad de información posible relacionada con el tema, incluyendo otros proyectos similares. Esta recolección permitirá definir mejor el contenido de la aplicación, dándole un criterio de realismo y en algunos casos será un criterio para descartar el proyecto, por ejemplo, por falta de información disponible. Además, permitirá estimar mejor los tiempos de desarrollo.

Es importante mantener un registro exacto de las fuentes y medios de los que se ha recolectado información ya que ésta deberá ser utilizada durante las siguientes etapas y es necesario saber cómo localizarla. Además, es frecuente que en esta etapa los distintos miembros del equipo se encuentren con información repetida por lo que es necesario cotejar y eliminar la redundancia. Una buena forma de hacerlo es

mantener una tabla con las fuentes, medios y destino de la información en el software, por ejemplo:

Medio	Texto	Foto	Texto	Texto	Vídeo/Sonid	Foto/Sonido
Artistas	Biografía	Autor	Obra 1	Obra 2	Obra 1	Obra 2
Pintura/ R. Matta	Vida y Obra de Matta	Sí, Biografía de Matta	Vida y Obra de Matta	Vida y Obra de Matta	Galería de Bellas Artes	Vida y Obra de Matta
Pintura/	...					
Música/ Arrau	Texto personal	Artículo revista	Descripción enciclopedia	IX Sinfonía,	Concierto , CD.	Concierto V,

Tabla 1. Ejemplo de una tabla de registro de información

Esta matriz permite mantener un control detallado del avance en el desarrollo de la aplicación. Además, se puede utilizar como una estimación de los recursos de desarrollo que faltan.

Como resultado de la actividad de definición del proyecto se tiene un *documento de consenso* o *especificación*, con los principales acuerdos para el desarrollo. Dicho documento contendrá una descripción clara de la aplicación y abordará cada uno de los puntos antes descritos.

En la actividad recién descrita el grupo actúa como una unidad, sin hacer diferencias estrictas en función de las especialidades de cada uno. Sin perjuicio de lo anterior, se aprovecha la experiencia de cada integrante para la recolección y análisis de la información especializada, por ejemplo, el ingeniero puede ser más apropiado para evaluar la funcionalidad de otros software, el sicólogo o educador para el análisis del usuario y el diseñador para evaluar la apariencia de materiales alternativos, etc.

DISEÑO DE LA APLICACIÓN

La actividad de diseño de la aplicación, a su vez, se divide en tres sub-actividades, estas son la de Definición de la Estructura General, la de Convergencia hacia una Interfaz, el Contenido y la Estructura y por último, la Definición del Prototipo. A continuación se detalla cada una:

DEFINICIÓN DE LA ESTRUCTURA GENERAL

Luego de la definición de la aplicación en la primera actividad, esta parte de la segunda actividad busca:

- Recolectar el máximo de información posible.
- Definir la estructura de la aplicación: cómo organizar y presentar los temas y sub-temas, cómo navegar por la aplicación, etc.
- Construir "maquetas" o bocetos con la estructura para visualizar la aplicación, sus componentes, los mecanismos de ubicación y navegación del usuario, etc.
- Validar la estructura o maqueta con usuarios reales.

Antes de explicar esta actividad, es necesario aclarar que para ilustrar las diferentes sub-actividades se utilizan diagramas de procesos (Figuras 2, 4, 6 y 7). En éstas, se identifican las actividades concretas que se realizan (rectángulos) y los arcos que las unen representan el flujo de los productos intermedios.

Los productos intermedios que considera este modelo son los siguientes (la “j” indica la versión):

D_j Documento de consenso o especificación (etapa de definición).

M_j Maqueta o Boceto. Esta es una representación en papel de la estructura de la aplicación, incluye, entre otros aspectos, bocetos de la interfaz, del contenido y de la navegación a utilizar. A través de esta maqueta es posible tener una visión más acabada y concreta de la estructura general de la aplicación y sus componentes. Además, al ser una representación en papel es de bajo costo y sencilla de modificar.

I_j Información. Representa el contenido recolectado para la aplicación, puede ser material escrito, videos, sonidos, etc. El control de ésta se lleva a través de la Tabla de Registro de Información (Tabla 1), creada en la actividad de Definición.

P_j Prototipo computacional. Es una aplicación utilizable a nivel básico por un usuario poco entrenado, puede presentar deficiencias funcionales y estéticas según lo que se desee evaluar. En este modelo se utiliza como punto de convergencia de dos o más áreas involucradas en el desarrollo.

RP Resultados de pruebas o evaluaciones que se le hacen al producto intermedio.

S Versión final de la aplicación. Es el programa listo para la etapa de producción.

A continuación se muestra la primera parte de la actividad de diseño:

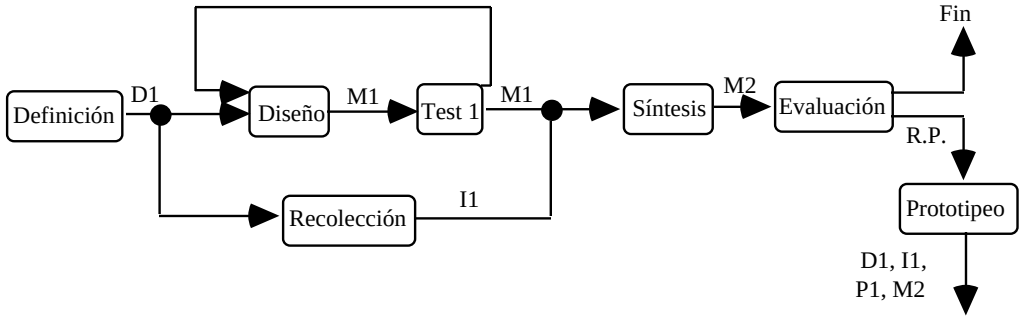

Figura 2. Representación gráfica de la primera parte de la actividad de diseño.

En base al documento de consenso o especificación (D1) que se obtiene de la actividad de Definición, en la actividad de Diseño se construye M1, que es la primera representación de la estructura de la aplicación. La maqueta es analizada (Test 1) al interior del grupo y se discuten los puntos acordados en el documento de consenso, este proceso se repite hasta llegar a un diseño consensual de M1. En paralelo, se recolecta información sobre los temas y sub-temas de la aplicación (I1). La Síntesis del Diseño y la Recolección dan origen a la segunda maqueta M2.

M2 es una concepción más realista de la estructura de la aplicación, ya que incorpora información más concreta de los contenidos. Esta es evaluada para determinar si se sigue o no con la aplicación. La Evaluación se realiza teniendo en cuenta la definición de usuario realizada, criterios de la disponibilidad de información, la complejidad de la estructura y los costos de implementación. Si la evaluación es positiva, se construye un esqueleto computacional de la aplicación (Prototipeo), resultando el prototipo P1. Si no es positiva, se puede volver a la etapa anterior, redefiniendo la aplicación o desechar el proyecto.

El resultado de esta primera parte del diseño es una maqueta de la aplicación y un esqueleto computacional. La maqueta contiene, principalmente, la estructura general del software (figura 3) y el esqueleto computacional es la implementación de esa maqueta en el computador. Cada uno de los documentos generados en esta etapa servirán de base para la próxima, por esto la importancia de probar y evaluar exhaustivamente cada material generado y/o recolectado.

Informática Educativa, **9** (1), 1996

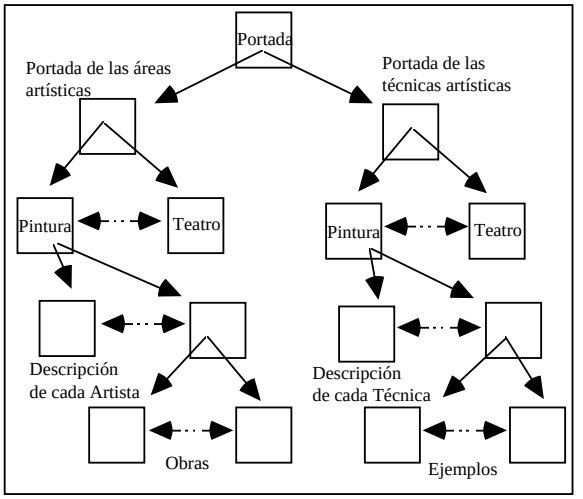

Figura 3. Ejemplo de una maqueta.

En la figura 3, cada cuadrado representa una unidad conceptual del software (relacionado con los temas o sub-temas) o un punto de elección de navegación, las flechas representan la navegación en profundidad que podrá realizar el usuario. Las líneas horizontales indican la existencia de múltiples unidades con posibilidad de navegación horizontal a ese nivel. En esta etapa no se incluyen otras navegaciones para mantener la discusión centrada sólo en la estructura y no en los aspectos de navegación.

A través de esta representación en papel, es posible visualizar la estructura completa del software a construir y tener una idea más clara del todo [14].También es una buena forma de conocer y validar las relaciones semánticas que un usuario puede establecer [xxvi]. Además, es posible hacer las modificaciones, simplemente, cambiando de lugar los papeles.

CONVERGENCIA DE INTERFAZ, CONTENIDOS Y ESTRUCTURA

En general, en esta parte de la actividad de diseño, se construyen bocetos con más contenido y expresividad, llegando a un mayor nivel de detalle e integración de los diferentes elementos (información, maquetas y prototipos). Además, se incluyen nuevas pruebas con usuarios y evaluaciones del proyecto. Este parte de la etapa se muestra en la figura 4

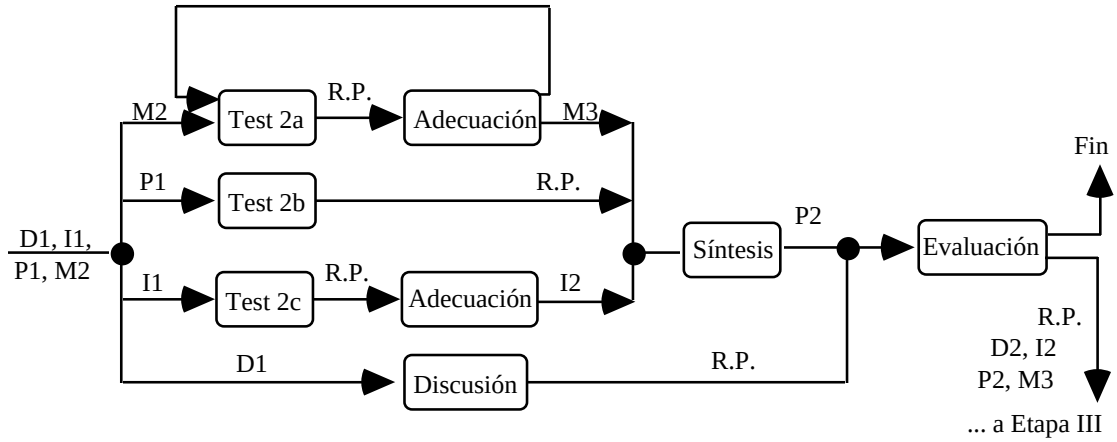

Figura 4. Representación gráfica de la segunda etapa del diseño.

En base a los documentos obtenidos en la etapa anterior, comienza un trabajo en paralelo de los diferentes integrantes del equipo de desarrollo. El Diseñador [4], Pedagogo e Ingeniero harán pruebas (Test 2a) de la maqueta M2 en términos de la estructura para determinar la navegación que se utilizará. Las pruebas se realizan invitando a personas no involucradas directamente en el proyecto a analizar la maqueta, este proceso se realiza un par de veces adecuando la maqueta según los resultados intermedios.

Cuando los cambios resultantes de los Tests no son significativos, se consolida la versión mejorada de la maqueta (M3). Esta incluye la estructura de la aplicación, una propuesta de la diagramación y estilo de la interfaz y los distintos tipos de navegación (lineal, hipermedial, por el índice, etc.). Un ejemplo de una versión simplificada de esta maqueta se aprecia en la siguiente figura:

[4] El subrayado indica el que dirige la prueba.

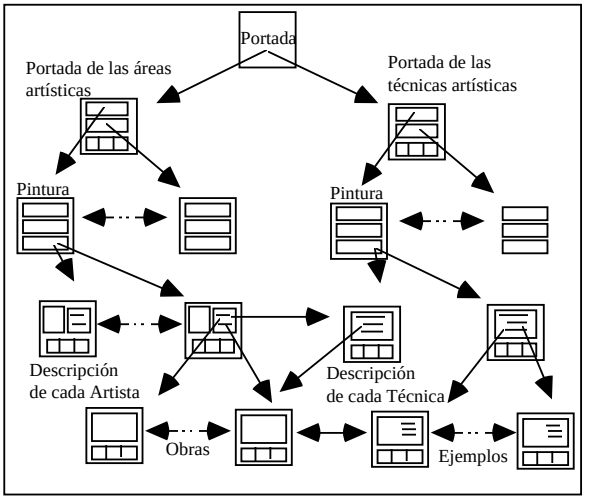

Figura 5. Ejemplo de una versión más avanzada de una maqueta.

El Sicólogo y el Pedagogo harán pruebas del prototipo computacional (Test 2b) para determinar el nivel de abstracción necesario para navegar por la aplicación, los enlaces hipermediales y multimediales [xxvii], etc. Estas pruebas se realizan con potenciales usuarios de la aplicación, se les hace una prueba guiada pidiéndoles que expliquen la estructura después de recorrerla.

En tanto, se sigue recolectando información sobre el tema a tratar. El Pedagogo y el Sicólogo harán pruebas (Test 2c) con los contenidos recolectados para determinar el estilo de redacción, el nivel de profundidad a incluir, etc., resultando I2. Estas pruebas se llevan a cabo con potenciales usuarios a los que se les solicita leer o ver la información, luego se evalúa el nivel de comprensión alcanzado y la motivación observada.

La síntesis de los resultados de los tres tests se traduce a un nuevo prototipo computacional que es evaluado entre todos (P2). La evaluación se hace en términos de una comparación con el documento de consenso, chequeando todos los aspectos definidos. Por ejemplo, cuán cercana o alejada está la concepción original dela actual en términos de la usabilidad del software por parte del tipo de usuario definido [xxviii] y de una evaluación de costos de implementación y desarrollo. Si de ese análisis resulta conveniente y factible seguir adelante, se reformula el documento de consenso y se sigue, si no, se puede volver a la etapa anterior y redefinir el proyecto.

DEFINICIÓN DEL PROTOTIPO

Este es el último paso de la actividad de diseño, se muestra en la figura 6:

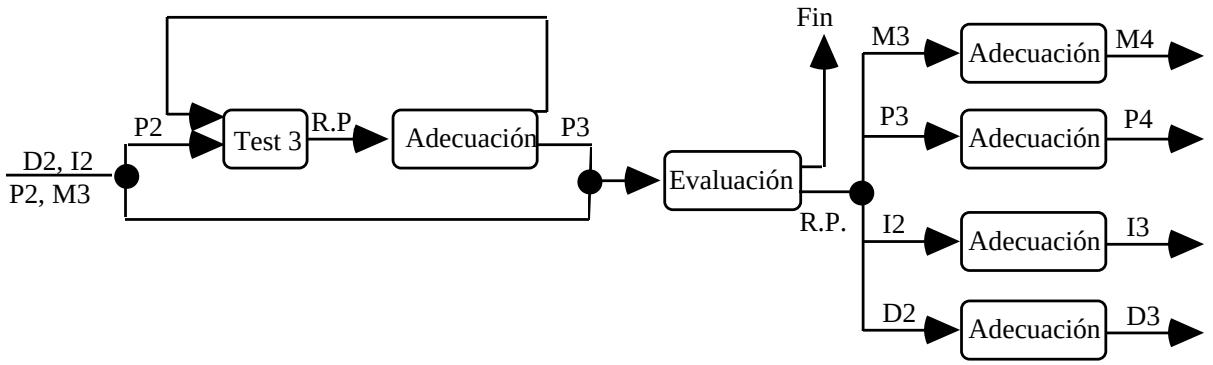

Figura 6. Representación gráfica de la etapa de desarrollo

El trabajo en esta etapa se realiza en conjunto, se trata de evaluar con personas externas (Test 3) el prototipo P2 del software. Los evaluadores son profesores y alumnos de las escuelas en las que se utilizará el software. Estos lo recorren libremente y se les asiste en las dudas que surjan durante la exploración. Luego se les pide su opinión en términos de la facilidad de uso, el aporte pedagógico y metodológico, etc. Este prototipo incluye cierta cantidad de contenidos para que un usuario externo se pueda formar una imagen más real y completa delo que será el software. Este proceso de prueba se repite hasta que no hayan modificaciones sustantivas a realizar al prototipo, o las modificaciones involucren un cambio mayor.

En base al resultado de estas pruebas se lleva acabo una evaluación crítica del proyecto. Si el resultado es positivo, los diferentes documentos se modifican en los aspectos relevantes, resultando M4, P4, I3 y D3. Los documentos modificados corresponden a la especificación formal del diseño de la aplicación. A partir de ahora, no deberían haber cambios substanciales en la concepción y diseño del proyecto. Si esto ocurre, es necesario retroceder un paso y llevar acabo las modificaciones correspondientes.

DESARROLLO DE PROTOTIPOS

La evolución que sigue el proyecto es básicamente en torno al desarrollo del prototipo que se incrementa en contenido, gráfica, navegación y funcionalidad sin

hacer cambios substanciales en las estructuras, estilos gráficos, la redacción y capacidades de la aplicación.

Cada prototipo es evaluado con los usuarios finales, profesores y alumnos, realizando pruebas en terreno y en el laboratorio. La primera prueba es la llamada "a-Test", las pruebas que se realizan después, son llamados "ß-Test" (figura 7).

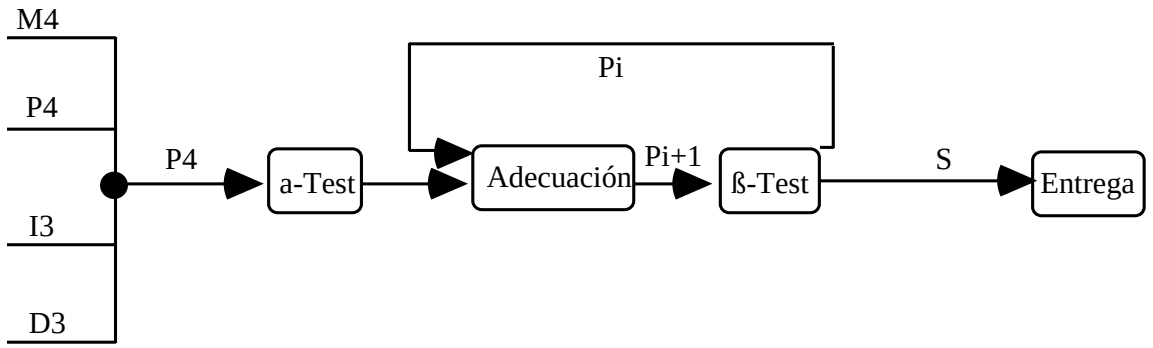

Figura 7. Representación gráfica del desarrollo de prototipos.

En cada evaluación realizada es importante documentar los logros y deficiencias encontradas. En general estas no son atribuibles a una sola disciplina que interviene en el desarrollo, más bien serán producto de su interacción. Las modificaciones y correcciones a realizar se deben definir a través de una reunión en la que participen representantes de todas las disciplinas y deberán surgir de común acuerdo. Estos criterios constituirán las medidas de logro para la próxima iteración [27].

Es importante hacer notar que en la etapa anterior, la estructura básica de las diferentes dimensiones del proyecto está definida (gráfica, arquitectura, contenidos, funcionalidad, etc.) y que la evolución ocurre sólo en términos de contenidos. Por ejemplo, se incorporan textos más o menos extensos (sin eliminar el texto), se cambian los estilos delos trazos (sin sacar el trazo), se implementan funcionalidades de forma más o menos complejas (sin cambiar la funcionalidad), etc.

El número de iteraciones en esta etapa estará relacionado con el tamaño de la aplicación y la cantidad de medios que incorpore. En un trabajo futuro, se espera poder establecer una función que permita estimar este aspecto.

El siguiente paso es llegar a tener un producto comercializable, esto implica la construcción de una serie de elementos anexos que se verán en la próxima etapa.

CONSTRUCCIÓN DE UN PRODUCTO

El resultado de la iteración prototipo-evaluación en terreno, debe finalizar con una aplicación que deberá ser "envasada" y distribuida. Para llevar a cabo este proceso es necesario definir y construir los siguientes puntos:

- Manual de usuario. Explica cómo usar el software.
- Guía metodológica. Explica cómo se pensó utilizar este software en términos pedagógicos, incluye una copia de las guías y materiales que se pueden imprimir desde el software.
- Pruebas intensivas y normalizadas del software.
- Estrategias de marketing asociadas (empaque, distribución, difusión, mercados, precio).

Esta etapa del desarrollo se analizará en versiones futuras de este trabajo.

CONCLUSIONES

En este trabajo se muestra el método de desarrollo utilizado para llevar a cabo la definición, diseño y construcción de programas hipermediales para educación. Desde la perspectiva de Ingeniería de Software este trabajo es una modelación del desarrollo de múltiples proyectos de software realizados, es decir, tiene una fuerte base empírica.

Los principales aportes del método están en los siguientes aspectos.

- En cada una de las etapas de desarrollo se incorporan las múltiples dimensiones del desarrollo (gráfica, arquitectura, contenidos, funcionalidad, etc.), cada una de ellas se centra esencialmente en el usuario, llevando a cabo distintos tipos de prueba (i.e. de contenidos, de interfaz, de estructura, etc.).
- Se incorporan evaluaciones tempranas del proyecto para decidir su conveniencia y factibilidad.
- Se utilizan técnicas de representación que permiten tener una idea clara del proyecto sin incurrir en desarrollos costosos (maquetas, bocetos y prototipos).
- Se hace una proposición para la organización del trabajo de los grupos de desarrollo multidisciplinario, en términos de trabajo paralelo, e hitos de convergencia claramente definidos.

El método presentado no se ha tratado de generalizar para el desarrollo de programas de otro tipo que no sean multimediales y educativos, ya que no es su objetivo. Además, las suposiciones en que se basa (existencia de contenidos, etapas de desarrollo cognitivo diferenciadas, etc.), hacen preveer que su generalización no es factible ni deseable.

Actualmente se han construido diez productos de software utilizando el método expuesto, cuatro de éstos llevan más de un año de uso y han sido utilizados por más de mil usuarios distintos (alumnos de escuelas). Estos productos tienen incorporado un mecanismo automático que registra la navegación que el usuario hace. A través de esta información se han podido lograr importantes mejoras para futuras versiones [xxix] Misanchuk y Schwier, 1992). Los seis restantes están en etapa de desarrollo de prototipos y ya han sido probados por más de veinte usuarios distintos.

Una parte del trabajo futuro estará dirigido allegar a una mejor y más completa definición de los roles de cada uno de los integrantes del equipo durante las diferentes etapas de desarrollo, la incorporación de algunas funciones o criterios de estimación de esfuerzo en la etapa de definición y ciertamente, la depuración de las etapas mismas.

Otra parte se está llevando a cabo a través de la formalización del proceso de evaluación del software en las distintas etapas de desarrollo, definiendo claramente los objetivos, participantes y métodos a utilizar en cada una de ellas.

Por último, se está avanzando en la estructuración de la información recolectada a través de los mecanismos de rastreo de usuarios para futuras publicaciones.

AGRADECIMIENTOS

Este trabajo ha sido realizado gracias al apoyo de FONDECYT, proyectos número 1930611 y número 1930569 y del Ministerio de Educación, a través del Programa MECE en el proyecto *Enlaces*.

REFERENCIAS

i GILLETE, J. E. The New Language of Images: Observations on Computer Multimedia Form*. Multimedia Review*. 3 (2), Summer 1992, pp.20-29.

ii POTTS, C. A Software Engineering Research Revisted. *IEEE Software*, September 1993, pp. 19-28.

iii BOEHM, B. W. *Software Engineering Economics*. Prentice-Hall, Inc. 1981.

iv DeGRACE, P. HULET STAHL L. *Wicked Problems, Righteous Solutions: A Catalogue of Modern Software Engineering Paradigms.* Yourdon Press, Prentice Hall, Inc. Englewood Cliffs, New Jersey 07632, 1990.

v SOMMERVILLE, I. *Software Engineering*. Addison Wesley, 3a Edición, 1989

vi GILB, T. *Principles of Software Engineering Management*. Addison Wesley, 1988.

vii DOCKTERMAN, D. A. (Ed.) *Great Teaching in the One Computer Classroom* Tom Snyder Productions, 1991.

viii NUTTIN, J. *Teoría de la Motivación Humana*, Buenos Aires, PAIDOS. 1982.

ix SORRENTINO, R. y HIGGINS, T. *Handbook of Motivation and Cognition*. New York: The Guilford Press. 1986.

x MINEDUC, *Primary Education Improvement Project, Technical Report*. Chilean Ministry of Education, June 1991.

xi CROOK, C. *Computers and the collaborative experience of learning*. London: Routledge. 1994.

xii WERTSCH, J. V. *Vygotsky and the social formation of mind*. London, Cambridge: Harvard University Press. 1985.

xiii THIMBLEBY, H. *User Interface Design*. Addison Wesley Pub. Co., New York, 1990.

xiv AMBRON, S. HOOPER, K. (Eds.) *Learning with Interactive Multimedia: Developing and Using Multimedia Tools in Education*. Microsoft Press, 1990.

xv DAVIDOFF, L. *Introducción a la Psicología*, Buenos Aires, PAIDOS.1984.

xvi REISER, R. A, DICK, W. Evaluating Instructional Software. *Educational Technology Research and Development*. 38 (3), 1990, pp.43-50. ISSN 1042-1629.

xvii TOLHURST, D. A Checklist for Evaluating Content-Based Hypertext Computer Software. *Educational Technology*. March 1992, pp.17-21.

xviii ZAHNER, J. E., REISER, R. A, DICK, W., GILL, B. Evaluating Instructional Software: A Simplified Model. *Educational Technology Research and Development*. 40 (3), 1990, pp.55-62.

xix AKINS, A. S. Human/Computer Interface Issues in Educational Computing. *Proceedings of National Educational Computing Conference - NECC*. 1993. pp. 47-51.

xx KEELER, M. A. DENNING, S. M. The challenge of interface design for communication theory: from interaction metaphor to contexts of discovery *Interacting with Computers*. 3 (3). Butterworth-Heinemann Ltd., 1991, pp.283-301.

xxi MOFFAT, A. *Psicoterapia del Oprimido*. Editorial ECRO. Buenos Aires, 1972.

xxii BRAVO, C. MONTENEGRO, H. *Educación, Niñez y Pobreza*. Editorial Nueva Universidad, Santiago de Chile, 1977.

xxiii SOMEKH, B. The Human Interface: Hidden Issues in CMC Affecting use in Schools. En MASON, R., KAYE A. (Eds.) *Mindweave: Communications, Computers and Distance Education*. Pergamon Press, 1989, pp. 242-246.

xxiv WATSON, L. Appropriate tools ? IT in the primary classroom. En J. BEYNON & H. MACKAY (Eds.), *Computers into classroom: more questions than answers* (pp. 78-91). London: The Falmer Press.1993.

xxv CLEBORNE, D. M. User Developed Computer-Assisted Instruction: Alternatives in Authoring Software. *Educational Technology*. April 1992, pp.7-1.

xxvi HARRIS, J. B, GRANDGENETT, N. F. A Developmental Hypermedia Materials. *Journal of Educational Multimedia and Hypermedia*. 2 (1),1993, pp. 83-101.Sequence of Children's Semantic Relationships: Implications for the Design of Interactive.

xxvii NELSON, W. A. PALUMBO, D. B. Learning, Instruction and Hypermedia. *Journal of Educational Multimedia and Hypermedia*. 1 (3),1992, pp. 287-299.

xxviii NIELSEN, J. The usability Engineering Life Cicle. *IEEE Computer*, March, 1992, pp. 12-22.

xxix MISANCHUK E. R. SCHWIER R. A. Representing Interactive Multimedia and Hypermedia Audit Trails *Journal of Educational Multimedia and Hypermedia*. 1 (3), 1992, pp. 355-372.