Metodología de diseño, desarrollo y evaluación de software educativo

Tesis de Magíster en Informática. (versión resumida)
Facultad de Informática. UNLP

Ing. Zulma Cataldi

Directores: Dr. Ramón García-Martínez
Ing. Raúl Pessacq

ISBN 960-34-0204-2
2000

liema@mara.fi.uba.ar

Contenidos

Resumen

La presente tesis se orienta a realizar una contribución en el área de metodología para el diseño, desarrollo y evaluación de software.

En particular, la metodología que se propone es aplicable al proceso de desarrollo de software educativo, contemplándose en las distintas etapas metodológicas aspectos de naturaleza pedagógica que no son tenidos en cuenta en las metodologías convencionales

Debido a la diversidad y multiplicidad de las actividades que se requieren para elaborar el producto de software, la metodología da soporte a un desarrollo tecnológico interdisciplinario, que tiene como pilares a la ciencia informática y a las ciencias de la educación.

Abstract

The present thesis is guided to carry out a contribution in the area of methodologies for design, development and software evaluation.

In particular, the methodology proposed in this work is applicable to the process of educational software development. Pedagogical, aspects are contemplated in the different stages of the proposed methodology. This aspects are not kept in mind in the conventional methodologies.

Due to the diversity and multiplicity of the activities that are required to elaborate the software product, the proposed methodology gives support to an interdisciplinary technological development that has basis on the computer science and the education sciences.

Introducción

En el trabajo de tesis se plantea una metodología para el diseño, desarrollo y evaluación del software educativo. La misma se basa en la sinergia de dos campos del saber aparentemente disímiles: la ingeniería de software por un lado y las teorías de aprendizaje modernas por el otro, que convergen en la generación de un producto deseable: el software educativo.

El presente trabajo pretende contribuir a las crecientes investigaciones que en estos últimos años se vienen realizando, tratando de desarrollar un software que contemplase los objetivos educativos, sin desmedro de las pautas de calidad en software.

Por lo tanto, la elección de este tema de tesis reúne tres tipos de interés que todo trabajo de estas características debe comprender:

- Un interés pedagógico: ya que mediante el uso del software apropiado los alumnos adquirirán distintas capacidades a través de las estrategias de enseñanza utilizadas. Sin querer dejar de lado las líneas conductistas, los diseños en la actualidad se basan en las teorías de Bruner (1988), Ausubel y Novak (1983), Perkins (1995) y Gardner (1995), entre otros.
- Interés profesional: puesto que se enmarca en los lineamentos actuales de la ingeniería del software y los desarrollos realizados durante los últimos años en cuanto a normativa a utilizar en el diseño de los productos software.
- Un interés económico–social: ya que esta investigación pretende ser un aporte más al mejoramiento del nivel educativo del país que afectará todas las áreas productivas

Es un trabajo de relevancia en el ámbito educativo y tecnológico, con la derivación socio–económica consecuente. Es un desarrollo de base conceptual y se lo prevé como una herramienta metodológica aplicable tanto en el ámbito educativo como en el no educativo.

Es el deseo de la autora que este aporte sea además, una aproximación para la fijación de directrices aplicadas a las tareas concernientes al desarrollo de software para el área educativa y criterios de evaluación de los productos educativos.

El software educativo durante los últimos años, ha tenido un creciente desarrollo y gran parte del mismo ha sido realizado en forma desorganizada y poco documentada, y considerando el aumento exponencial que sufrirá en los próximos años, surge la necesidad de lograr una metodología disciplinada para su desarrollo, mediante los métodos, procedimientos y herramientas, que provee la ingeniería de software para construir programas educativos de calidad, siguiendo las pautas de las teorías del aprendizaje y de la comunicación subyacentes.

Las primeras ideas sobre desarrollo de software educativo aparecen en la década de los 60, tomando mayor auge después de la aparición de las microcomputadoras a fines de los 80.

Los programas se han desarrollado según tres líneas distintas. La primera corresponde a los lenguajes para el aprendizaje y de ella nace el Logo, como un lenguaje que fue utilizado en un sentido constructivista del aprendizaje. Es de decir, el alumno no descubre el conocimiento, sino que lo construye, sobre la base de su maduración, experiencia física y social (Bruner, 1988). Su evolución continua en la actualidad hacia otras formas de interacción llamadas micromundos. A partir de ahí se ha desarrollado infinidad de software de acuerdo a las diferentes teorías, tanto conductistas, constructivistas como cognitivistas (Gallego, 1997).

La segunda línea corresponde a la creación de lenguajes y herramientas que sirven para la generación del producto de software educativo. Ella, se inicia con la aparición de los lenguajes visuales, los orientados a objetos, la aplicación de los recursos multimediales (Nielsen, 1995) y las herramientas de autor, el campo del desarrollo del software se ha hecho muy complejo, razón por la cual se necesita de una metodología unificada para su desarrollo.

Por último, surgen los productos propiamente dichos que nacen con la enseñanza asistida por computadora (EAC) u ordenador (EAO) que dio la aparición del software educativo, y que a su vez se difundió según tres líneas de trabajo: como tutores (enseñanza asistida por computadoras), como aprendices y herramientas. (Schunk, 1997).

Se pueden enumerar algunos de los problemas detectados que aún subsisten, como la mistificación de las herramientas informáticas aplicadas por los técnicos, la falta de capacitación docente en el tema específico y que las reglas y los pasos metodológicos para la creación de software en general se modifican evolutivamente.

Es por ello, que se quiere presentar una solución informática para el diseño, desarrollo y evaluación tanto interna como externa, mediante la aplicación de las métricas correspondientes, para determinar los parámetros básicos del proyecto de software educativo, teniendo en cuenta los requerimientos particulares del mismo en cuanto a los aspectos pedagógicos. En este enfoque disciplinado para el desarrollo de dicho software, se pretende aplicar los métodos, procedimientos y herramientas de la ingeniería del software, los cuales ayudan a asegurar la calidad del mismo.

El software educativo, tiene características particulares en cuanto a la comunicación con el usuario (Gallego, 1997), las cuales no se pueden cuantificar mediante métricas porque están relacionadas con conductas de

aprendizajes o actos de significado, pero las reglas en la construcción de un programa son las mismas ya sea para el ámbito educativo, comercial, de investigación, u otros.
La eficacia del producto constituye a su vez un alto riesgo debido a que sólo puede ser medida después de finalizado y probado por los alumnos (Fainholc, 1994), por ello es fundamental la instancia de evaluación tanto interna como externa (Marquès, 1995; Sancho, 1994; Reeves, 1997; Meritxell, 1996), y la contextualizada para el logro del producto deseado.
Algunos autores como Marquès (1995) sostienen que las metodologías específicas a utilizar para el diseño del software educativo se pueden englobar bajo el nombre de ingeniería de software educativo.

Objetivos

Se han determinado una serie de objetivos que a continuación se detallan.

Objetivos pertinentes a la construcción del objeto de estudio

- Definir qué es software educativo
- Ofrecer un estudio crítico de la situación actual

Objetivo general

- Construir una metodología disciplinada para el desarrollo del software educativo, mediante la identificación de los métodos, los procedimientos, y las herramientas, que provee la ingeniería de software para el desarrollo de programas educativos de calidad, siguiendo las pautas de la teoría educativa subyacente.

Objetivos particulares

- Justificar un modelo apropiado para el ciclo de vida del software educativo.
- Desarrollar una metodología de evaluación interna y externa del software educativo a fin de lograr un producto de calidad.

Estructura de la tesis

La tesis se divide básicamente en seis grandes partes: Introducción, Estado del Arte, Descripción del Problema, Descripción de la Solución Propuesta, Parte Experimental y Conclusiones. Estas a su vez se subdividen en capítulos.
Introducción: Aquí se describe la presentación de la problemática contextualizada, los objetivos propuestos, y la estructura de la tesis.
Estado del Arte: En la **"Introducción"**, se describe como se efectuó el relevamiento realizado en cuanto a la evolución de los desarrollos de software educativo, sin base pedagógica en sus inicios y luego con la base de las teorías del aprendizaje que los sustentan. Se intenta dar un panorama de cómo evolucionaron hasta nuestros días tales desarrollos y qué aspectos deberían tomarse de la ingeniería de software para mejorar los diseños. Se pretende dar un panorama neutral de los trabajos más relevantes realizados en el área, presentando la mayor parte de las herramientas disponibles y sus fundamentos teóricos para considerarlas como un punto de partida para el desarrollo de la solución propuesta. En la sección 1, denominado **"Las teorías del aprendizaje y el diseño de software educativo"**, se presenta una síntesis diacrónica de las teorías del aprendizaje más conocidas y se las relaciona con la aparición del software educativo. Se plantea el panorama actual acerca de las problemáticas vigentes a causa de algunos aspectos divergentes en la construcción de los programas educativos. La sección 2, llamado **"El software educativo"**, es una síntesis de las tipologías y características principales de los programas educativos, los diferentes roles que pueden tener los profesores que los utilizan y los procesos de comprensión que se intenta desarrollar o incentivar en los alumnos. Se describen las interfaces de comunicación usuario–software y se considera el contexto de aplicación y uso de los programas mediante el empleo de una buena planificación didáctica.
En la sección 3, **"La ingeniería de software"** se describe desde la ingeniería de software, la gran variedad de modelos o paradigmas de ciclo de vida, para el desarrollo de los programas, ya sea para un gran proyecto de software, o un simple programa. Se describen las metodologías, los métodos, los procedimientos y las herramientas que utiliza la ingeniería de software. Se incluye un apartado especial acerca de las métricas utilizadas en la determinación de la calidad, como de la normativa vigente en cuanto a productos lógicos.
La sección 4 corresponde a **"Evaluación del software educativo"**, y en él se detallan todos los aspectos a tener en cuenta para una aplicación óptima a nivel áulico. Se resumen los trabajos considerados más relevantes en cuanto a evaluación que en general toman la forma de listas de control o valoración ponderativa de algunos criterios.
En **"Conclusiones del Estado del Arte"**: se quieren señalar simplemente, aquellos aspectos a tener en cuenta para los futuros diseños de los programas educativos, que han detectado otros investigadores, y las posibles soluciones a algunas de las problemáticas planteadas.
Descripción del Problema: En esta sección que consta de l capítulo 5; **"Presentación de la problemática"**, se describe el problema actual de los desarrollos de los programas educativos que se pretende resolver.
Descripción de la Solución Propuesta: Esta parte de la tesis se divide en dos capítulos. El primero, es la sección 6 **"Propuesta de metodología de diseño y desarrollo"**, y aquí se consideran las características de

los programas educativos, para definir un modelo de ciclo de vida y una teoría educativa apropiados. Se presentan las actividades a realizar en cada una de las etapas a fin de establecer los recursos materiales y humanos indispensables para cada una de ellas y se analiza la configuración del equipo de desarrollo y los roles cada uno de los participantes en los procesos.
La sección 7 es la **"Propuesta de evaluación",** sobre la base de la doble evaluación de los productos de software educativo, que debe considerar aspectos técnicos y pedagógicos, se evalúa un software desarrollado a partir de la propuesta de la sección anterior, y se detallan las evaluaciones interna y externa del producto. Se resumen los resultados de las evaluaciones realizadas, de los prototipos presentados y del producto final, llevados a cabo por los grupos de alumnos evaluadores. Se evaluaron progresivamente: la interface de comunicación en el primer caso, el funcionamiento de las bases de imágenes, vídeos y efectos, en el segundo y finalmente el producto final con, el agregado de la voz, los textos y la música. Se dedica un apartado a evaluar aquellos aspectos que se consideran importantes para desarrollar un software de calidad.
Parte Experimental: En la sección 8: **"Parte experimental"**: se realiza una evaluación contextualizada, contrastando un producto realizado con la metodología propuesta, con otro desarrollado sin una metodología explícita.
Finalmente se presentan las **Conclusiones** a esta propuesta y experiencia y se dejan bosquejadas algunas posibles líneas de investigación futuras.

Estado del Arte

1.Las teorías de aprendizaje y el diseño de software educativo.

1.1. Resumen

En esta sección se presenta la aparición y la evolución del software educativo a la luz de algunas de las teorías del aprendizaje más representativas. En este paralelismo, sólo se mencionan aquellas teorías que darán los marcos conceptuales para los desarrollos de los programas didácticos en función de las aplicaciones deseadas.

Partiendo del conductismo de Skinner, (sección 1.2) se pasa por el constructivismo (sección 1.3) y las diversas líneas de la psicología cognitiva (secciones 1.4, 1.5, 1.6) considerando también los aspectos metacognitivos (sección 1.7).

Se quieren destacar también, los cambios paradigmáticos producidos a partir del conductismo, el constructivismo y la psicología cognitiva y su repercusión en la construcción de software educativo (secciones 1.8 y 1.9). A partir de aquí es donde se concatena la noción de secuenciación de contenidos de Coll con la programación estructurada modular.

Pero, es con la irrupción de la computadoras personales a bajo costo, cuando se masiviza el uso de los programas educativos, y el uso de la computadora como tutor, herramienta o aprendiz, según Schunk, teniendo como sustrato la teorías de aprendizaje mencionadas. Por último, se da una síntesis de la problemática actual en el área (sección 1.10).

1.2. El condicionamiento operante: La instrucción programada.

A comienzos de la década de los 60 se pensó que una de las posibles soluciones a algunos a los problemas educativos de esa época, consistía en la aplicación de los avances tecnológicos a la enseñanza. Sin embargo, la introducción de los instrumentos tecnológicos no fue acompañada con una teoría acerca de la enseñanza y del aprendizaje.

Skinner (1958, 1963) formuló su teoría conductista del condicionamiento operante en los años treinta y, durante los primeros años de su carrera se interesó por la educación elaborando las *"máquinas de enseñanza"* y los *"sistemas de instrucción programada"*. El cambio conductual en el *"condicionamiento operante"* se da a través del refuerzo diferencial por aproximaciones sucesivas hacia la forma de comportamiento deseada, mediante el proceso de moldeamiento para modificar la conducta.

Durante los años sesenta aparecen una corriente de *"programadores"* (Deterline,1969), que empezaron a *"programar"* de una manera muy fácil, y, que careciendo de formación docente, tomaban un libro de texto, borraban alguna palabra de una frase elegida y la sustituían por una línea horizontal, para que el alumno anotara allí su respuesta. Repetían la frase varias veces por cada cuadro, pero borrando una palabra diferente cada vez.

En esta época es cuando comienzan los estudios referidos a la elaboración de lo que se considera una buena *"programación didáctica"*. La elaboración de una programación se iniciaba con el establecimiento de los objetivos generales en función del curriculum de los alumnos, se construía el programa, elaborando la serie de secuencias a seguir en "*cuadros*". Luego, se estudiaba el tipo de respuesta más adecuada y la clase de feedback[1] a lograr. El paso siguiente era la evaluación y revisión del programa sobre la base de las respuestas de los alumnos.

En este período, cobran interés los objetivos operacionales y conductuales a partir de un trabajo de Mager (1967), que se usó como un manual para los escritores de enseñanza programada. El objetivo debe describir una conducta observable y sus productos o logros.

Las décadas de los sesenta y setenta, destacan a una serie de autores dedicados a la definición, la elaboración y la redacción de objetivos conductuales tales como Gagné (1970), quien da una tipología de los aprendizajes, y para cada uno de ellos reconoce estadíos o fases, que son las condiciones psicológicas para un aprendizaje eficaz (Fernández Pérez, 1995). El aprendizaje ocurre así, a través de transformaciones de la información.

1.3. Los ambientes constructivistas de aprendizaje

Las primeras ideas sobre desarrollo de software educativo aparecen en la década de los 60, tomando mayor auge después de la aparición de las microcomputadoras a fines de los 80.

El uso de software educativo como material didáctico es relativamente nuevo, *los primeros pasos* fueron dados por el lenguaje Logo, que a partir de su desarrollo en el MIT (Instituto Tecnológico de Massachusetts) fue utilizado en numerosas escuelas y universidades.

[1] retroalimentación

Se desarrolla *una línea de software que corresponde a los lenguajes para el aprendizaje* y de ella nace el Logo, que fue utilizado en un sentido constructivista del aprendizaje.
Es decir, como sostiene Bruner: *"el punto crucial y definitorio del aprendizaje, del conocimiento de algo nuevo, radica en la posibilidad humana de abstraer en los objetos algunos pocos rasgos para construir criterios de agrupamiento de los objetos abstraídos",* a pesar de que con frecuencia acontece que los rasgos comunes son muchos menos y menores, que los rasgos que los diferencian como plantea Fernández Pérez (1995). En otras palabras, hace del proceso de formación de conceptos una instrumentalización cognitiva.
El alumno no descubre el conocimiento, sino que lo construye, en base a su maduración, experiencia física y social (Bruner 1988), es decir el contexto o medio ambiente.
Según Bruner, algunas de las habilidades a adquirir son: la capacidad de identificar la información relevante para un problema dado, de interpretarla, de clasificarla en forma útil, de buscar relaciones entre la información nueva y la adquirida previamente.
Hablar de ambientes de enseñanza constructivistas significa concebir el conocimiento desde la perspectiva de Piaget (1989) mediante desarrollos cognitivos basados en una fuerte interacción entre sujeto y objeto, donde el objeto trata de llegar al sujeto, mediante cierta perturbación de su equilibrio cognitivo, quien trata de acomodarse a esta nueva situación y producir la asimilación del objeto, con la consecuente adaptación a la nueva situación. En este esquema conceptual piagetiano, se parte de la acción, esencial, ya sea para la supervivencia, como para el desarrollo de la cognición. *"La postura constructivista psicogenética acepta la indisolubilidad del sujeto y del objeto en el proceso de conocimiento. Ambos se encuentran entrelazados, tanto el sujeto, que al actuar sobre el objeto, lo transforma y a la vez se estructura a sí mismo construyendo sus propios marcos y estructuras interpretativas"* (Castorina, 1989).
A partir de aquí, se ha desarrollado infinidad de software de acuerdo a las diferentes teorías, tanto conductuales, constructivistas y posteriormente cognitivistas (Gallego 1997).

1.4. El cognitivismo y los mapas conceptuales.

El cognitivismo tiene sus raíces en la ciencia cognitiva y en la teoría de procesamiento de la información. Howard Gardner psicólogo de Harvard, sostiene que el nacimiento de la psicología cognitiva es de 1956. Es a partir de esta fecha que se empieza a gestar el movimiento que algunos llaman revolución cognitiva y que a juicio de Lachman et al. (1979) *"constituyó un verdadero cambio de paradigmas en el sentido kuhniano*[2]*"*. (Hernández, 1997)
El cognitivismo es una teoría de aprendizaje donde la mente es un agente activo en el proceso de aprendizaje, construyendo y adaptando los esquemas mentales o sistemas de conocimiento. Bruner (1991) sostiene que la revolución cognitiva tenía como objetivo principal recuperar la mente, después de la época de la *"glaciación conductista"* (Hernández, 1998). En los inicios del modelo cognitivo, señala Bruner, había una firme intención en la realización de esfuerzos para indagar acerca de los procesos de construcción de los significados y producciones simbólicas, empleados para conocer la realidad circundante. Sin embargo, el papel creciente de la informática y las computadoras incorporó un planteamiento basado en la metáfora de las computadoras.
Dentro de la teoría cognitiva los psicólogos del procesamiento de la información usan la analogía de la computadora para explicar el aprendizaje humano, con el supuesto básico de que todo aprendizaje consiste en formar asociaciones entre estímulo y respuesta. Según Gardner (1987) y Rivière (1987): *"el paradigma del procesamiento de la información dentro de la psicología educativa, se inserta en la gran tradición racionalista de la filosofía, que otorga cierta preponderancia al sujeto en el acto del conocimiento"*.
Vigotzkii (1978), por otra parte desde su modelo sociocultural, destaca las actividades de aprendizaje con sentido social, atribuyendo gran importancia al entorno sociocomunicativo del sujeto para su desarrollo intelectual y personal. Sostiene que el cambio cognitivo, se da en la ZDP (zona de desarrollo próximo) o sea la distancia entre el nivel real de desarrollo y el nivel posible, mediante la resolución de problema mediado por un adulto o tutor, siendo a veces el aprendizaje repentino, en el sentido *gestáltico* [3] del *insight*. [4]
Entre las ideas de Vigotzkii, se deriva un concepto muy importante que es el que Bruner denomina *"andamiaje*" educativo que consiste en brindar apoyo, y en el caso de la computadora como herramienta, para permitir ampliar el alcance del sujeto y la realización de tareas que de otro modo serían imposibles y usarlos selectivamente cuando se necesitan.
Rogers (1984) habla de *"la facilitación del aprendizaje que aparece como una potencialidad natural de todo ser humano"*. Dice *que "el aprendizaje significativo"* tendrá lugar cuando el sujeto perciba al tema como importante para sus propios objetivos o satisfaciendo alguna de sus características o necesidades personales sociales. El término significativo también puede ser entendido siguiendo a Ausubel (1983), como un contenido que tiene una estructuración lógica interna y como aquel material que puede ser aprendido de manera significativa por el sujeto. Rogers afirma que "*el aprendizaje social más útil en el mundo es el aprendizaje*

[2] En referencia a Kuhn T. (1980) en *La estructura de las revoluciones científicas*. México. FCE.
[3] En referencia a la teoría de la forma
[4] En el sentido de visión repentina

del proceso de aprendizaje, que significa adquirir una actitud continua de apertura frente a las experiencias e incorporar a sí mismo el proceso de cambio".
"El conocimiento elaborado a través de conceptos teóricos de las diferentes disciplinas, requiere también desarrollos en la recepción en los alumnos para una comprensión significativa" (Ausubel, 1983). Esta denominación de *"comprensión significativa o aprendizaje significativo"* tiene para Ausubel un sentido muy particular: incorporar información nueva o conocimiento a un sistema organizado de conocimientos previos en el que existen elementos que tienen alguna relación con los nuevos.
El alumno que carece de tales esquemas desarrollados, no puede relacionar significativamente el nuevo conocimiento con sus incipientes esquemas de comprensión, por lo que, ante la exigencia escolar de aprendizaje de los contenidos disciplinares, no puede sino incorporarlos de manera arbitraria, memorística, superficial o fragmentaria. Este tipo de conocimiento es difícilmente aplicable en la práctica y, por ello, fácilmente olvidado.
El nuevo material de aprendizaje solamente provocará la transformación de las creencias y pensamientos del alumno cuando logre *"movilizar los esquemas ya existentes de su pensamiento".* Al alumno se le debe enseñar de tal manera, que pueda continuar aprendiendo en el futuro por sí solo. Ausubel y sus colaboradores, según expresa Coll (1994), *"concretan las intenciones educativas por la vía del acceso a los contenidos, lo cual exige tener un conocimiento profundo de los mismos para armar un esquema jerárquico y relacional".*
Según Novak y Ausubel, (1997) todos los alumnos pueden *"aprender significativamente un contenido, con la condición de que dispongan en su estructura cognoscitiva o cognitiva, de conceptos relevantes e inclusores".*
Cabe recordar la frase: *"El factor más importante que influye en el aprendizaje es lo que el alumno ya sabe. Averígüese esto y enséñese consecuentemente"*, (tal como Ausubel, Novak y Hanesian expresan en el prefacio de su *libro "Psicología Educativa. Un punto de vista cognoscitivo*"), esencial para construir herramientas o indicadores diagnósticos de la estructura cognitiva de los alumnos. El contenido del aprendizaje debe ordenarse de tal manera que los conceptos más generales e inclusivos se presenten al principio. Esto favorece la formación de conceptos inclusores en la estructura cognoscitiva de los alumnos que facilitan, posteriormente, el aprendizaje significativo de los otros elementos del contenido.
Para lograr una diferenciación progresiva del conocimiento del alumno y una *"reconciliación integradora"* posterior, las secuencias de aprendizaje tienen que ordenarse partiendo de los conceptos más generales y avanzando de forma progresiva hacia los conceptos más específicos.
"El aprendizaje significativo, es un aprendizaje globalizado en la medida en que el nuevo material de aprendizaje pueda relacionar de forma sustantiva y no arbitraria con lo que el alumno ya sabe", (Coll, 1994), con calidad de lo aprendido y duración del almacenamiento.
Los mapas conceptuales, adaptados de Novak (1984), surgen como una herramienta base para representar las relaciones significativas entre conceptos. Actualmente son el fundamento para la red semántica base para el desarrollo del software educativo cognitivista.
El mapa de base, es el punto de partida para el acuerdo entre los especialistas de las diferentes áreas que intervienen en dicho desarrollo. Esta base proveerá un camino de navegación libre de ambigüedades e incoherencias. Usando recursos hipermediales, se pueden construir documentos interrelacionados siguiendo una estructura jerárquica de modo que el alumno navegue pasando desde las informaciones más inclusivas a las más específicas.

1.5. La Teoría Uno y la Teoría de las inteligencias múltiples

David Perkins (1995), codirector del Proyecto Zero del Centro de Investigación para el Desarrollo Cognitivo, de Harvard, en su Teoría Uno afirma que *"la gente aprende más cuando tiene una oportunidad razonable y una motivación para hacerlo"*. Puede parecer imposible que este enunciado tan trivial, dice el autor, implique alguna mejora en la práctica educativa, pero basándose en el sentido común, se podrían señalar las siguientes condiciones:

- Información clara.
- Práctica reflexiva.
- Realimentación informativa.
- Fuerte motivación intrínseca y extrínseca.

La Teoría Uno intenta simplemente ser un punto de partida*: "Dada una tarea que se desea enseñar, si se suministra información clara sobre la misma mediante ejemplos y descripciones, si se ofrece a los alumnos tiempo para practicar dicha actividad y en pensar cómo encararla, si se provee de realimentación informativa con consejos claros y precisos para que el alumno mejore el rendimiento y se trabaja desde una plataforma de fuerte motivación intrínseca y extrínseca, es probable que se obtengan logros considerables en la enseñanza."* (Perkins, 1995).
La Teoría Uno no es un método de enseñanza, sino un conjunto de principios que todo método válido de enseñanza debe satisfacer. Sobre esto Perkins afirma en su obra: *"La Escuela Inteligente"*, que cualquier método válido de enseñanza encarna a la Teoría Uno y amplía sus principios para adaptarse a las necesidades

particulares del estudiante y del momento. *"Una buena enseñanza requiere métodos distintos para ocasiones distintas"*: la Teoría Uno debe subyacer a todos ellos. Mortimer Adler[5], en *"La Escuela Inteligente"* destaca tres modos de enseñar: la instrucción didáctica, el entrenamiento y la enseñanza socrática. (Perkins, 1995).
La Teoría Uno es compatible con el conductismo y con el constructivismo, pero no enfatiza la importancia de que el alumno elabore sus ideas con un alto grado de autonomía a fin de alcanzar la verdadera comprensión. Puede considerarse como un mojón que marca el primer hito hacia otras teorías más elaboradas y aún usando sólo sus dos versiones más simples: con la instrucción didáctica y el entrenamiento, se obtendrían resultados considerablemente mejores que los actuales.
En el caso de desarrollos del software educativo, se pueden incorporar, como sostiene Perkins, representaciones potentes, mediante imágenes mentales y utilizar modelos, de tal modo de estimular la motivación de los alumnos e intentar desarrollar actividades mentales como:
- evaluar y discriminar lo específico de lo particular,
- construir, crear,
- evaluar necesidades, procesos, resultados,
- investigar otras posibilidades de solución,
- resolver problemas inéditos,
- transferir conocimiento de y hacia otras áreas,
- sintetizar, globalizar, analizar, etc.

Perkins habla acerca de la conexión importante que existe entre la pedagogía de la comprensión (o el arte de enseñar a comprender) y las imágenes mentales, por lo que se puede decir que la relación es bilateral.
Esta relación recíproca existente puede ayudar al alumno a adquirir imágenes mentales, con lo cual desarrolla su capacidad de comprensión y al exigirles actividades de comprensión (como por ejemplo: predecir, explicar, resolver, ejemplificar, generalizar) se hará que construyan imágenes mentales, para lo que afirma Perkins que: *"se alimentan unas a otras como si fueran el Yin y el Yan de la comprensión"*. En cuanto a la transferencia, la idea es aprender en una situación determinada y luego aplicar lo aprendido en otra muy diferente.
Una enseñanza comprensiva para favorecer el desarrollo de los procesos reflexivos, es la mejor manera de generar la construcción del conocimiento no frágil.
Por otra parte, el psicólogo Howard Gardner (1993), quien enunció la "Teoría de las Inteligencias Múltiples" sostiene que la inteligencia humana posee siete dimensiones diferentes y a cada una de ellas corresponde un sistema simbólico diferente y un modo de representación: lógico-matemática, lingüística, musical, espacial, cinético-corporal, interpersonal e intrapersonal.
Gardner sostiene que la práctica educativa se centra fundamentalmente en las inteligencias matemática y lingüística y que dado el carácter múltiple de la inteligencia humana se debe ampliar el horizonte a fin de dar cabida a las diversas habilidades de las personas, proponiendo a los alumnos proyectos que admitan modos alternativos de expresión simbólica, creando proyectos grupales que inviten a los alumnos a trabajar con el lenguaje de los medios de comunicación y con sistemas simbólicos por los que sientan una mayor afinidad e induciendo una mayor diversidad de sistemas simbólicos en las diferentes áreas del saber.
La teoría de las inteligencias múltiples supera a la Teoría Uno en tanto que hace hincapié en la diversidad de la capacidad humana en la consecuente necesidad de diversificar las oportunidades y los caminos pedagógicos. (Perkins, 1995).

1.6. Las cogniciones repartidas o distribuidas.

Respecto de la relación persona-herramienta que interactúan para dar lugar al proceso cognitivo, Perkins (1985) dice que la cognición humana, siempre se produce de una manera física, social y simbólicamente repartida. Las personas piensan y recuerdan con la ayuda de toda clase de instrumentos físicos e incluso construyen otros nuevos con el fin de obtener ayuda. Las personas piensan y recuerdan por medio del intercambio con los otros, compartiendo información, puntos de vista y postulando ideas.
Libedinsky (1995) en el marco pedagógico de la utilización de tecnologías en el ámbito educativo, dice que uno de los principios clave que puede operar es el de las cogniciones repartidas. "*Cuando se examina la conducta humana en la resolución de problemas de la vida real y en entornos laborales, la gente parece pensar en asociación con otros y con la ayuda de herramientas provistas por la cultura, las cogniciones parecerían no ser independientes de las herramientas con las que se resuelve un problema. Las cogniciones parecerían distribuirse físicamente con nuestros útiles y herramientas, entre ellas la computadora, socialmente con quienes compartimos las tareas intelectuales y simbólicamente desde las palabras, gráficos y mapas conceptuales, entre otros, como medios de intercambio entre la gente. Los recursos físicos y sociales, participan en la cognición no sólo como fuente sino como vehículo del pensamiento".*(Libedinsky, 1995).

1.7. Aprender a aprender

[5] Citado por David Perkins (1995), en "*La Escuela Inteligente*". Gedisa.

La metacognición se refiere *al "conocimiento de los propios procesos cognitivos"*, es una forma de conocimiento que tiene como rasgo diferencial su referencia al sistema humano de procesamiento de información, es decir, conocer qué son, cómo se realizan, cómo se potencian o interfieren los procesos cognitivos como la percepción, la atención, la memorización, la lectura, etc.
Es el conocimiento que ha desarrollado el alumno acerca de sus experiencias almacenadas y de sus propios procesos cognoscitivos, así como de su conocimiento estratégico y la forma apropiada de uso. (Flavell,1993).
El conocimiento metacognitivo es de aparición relativamente tardía en casi todos los dominios del aprendizaje escolar. Básicamente, la metacognición tiene que ver con el conocimiento que cada uno tiene de sus propios procesos cognitivos, abarcando también, el control activo y la regulación de tales procesos, lo cual implica tener conciencia de las propias fortalezas y debilidades acerca del funcionamiento intelectual de cada uno.

1.8. Los desarrollos actuales de software

Una *segunda línea* en los desarrollos de software, corresponde a la creación de lenguajes y herramientas para la generación del producto de software educativo. Ella, se inicia con la aparición de los lenguajes visuales, los orientados a objetos, la aplicación de los recursos multimediales (Nielsen 1995) y las herramientas de autor, complejizando el campo del desarrollo del software, razón por la cual se necesita de una metodología unificada para su desarrollo.
Los lenguajes de programación han experimentado en los últimos años un notable auge. El por qué del crecimiento evolutivo, a partir de los lenguajes de máquina y ensambladores, debe buscarse en el intento por acercarse a los lenguajes naturales de las personas. Surgen así, los lenguajes de alto nivel o evolucionados, a partir del FORTRAN en 1955, desarrollado por IBM; el Cobol, se creó en 1960, como un intento del Comité CODASYL de lenguaje universal para aplicaciones comerciales, el PL/I, que surge en los sesenta para ser usado en los equipos de IBM 360.
El Basic surge en 1965, lenguaje ampliamente usado en el ámbito educativo y en 1970 aparece el Pascal, creado por el matemático Niklaus Wirth, basándose en el Algol de los sesenta. Este lenguaje en particular aporta los conceptos de programación estructurada, tipo de datos y diseño descendente. La evolución continúa hacia otros más modernos como el C, creado en 1972 por Denis Ritchie y el ADA, cuya estandarización se publicó en 1983 (Alcalde et al., 1988).
Los lenguajes se incorporaron rápidamente al ámbito educativo, porque se consideró que permitían ayudar a mejorar el pensamiento y acelerar el desarrollo cognitivo. Los estudios en este aspecto si bien sostienen que se pueden lograr habilidades cognitivas no indican que se facilite la transferencia hacia otras áreas del saber. (Liguori, 1995)

1.9. La aparición del software educativo

Por último aparecen *los productos propiamente* dichos de software educativo, con la difusión de las computadoras en la enseñanza, según tres líneas de trabajo, computadoras como tutores (enseñanza asistida por computadoras o EAC), como aprendices y como herramienta. (Schunk 1997).
La enseñanza asistida por computadora (EAC) o enseñanza basada en computadora (EBC) es un sistema que se utiliza sobre todo para efectuar ejercicios, cálculo, simulaciones y tutorías. Los programas de ejercicios son fáciles de realizar y los alumnos proceden a manejarlos en forma lineal en su repaso de información. Las tutorías presentan información y retroalimentación, de acuerdo a la respuesta de los estudiantes, que en este caso son programas ramificados.
Una aplicación interesante de las computadoras son las simulaciones por que permiten al alumno ponerse en contacto con una situación real que de otro modo nunca podría hacerlo, tal es el caso de los simuladores de vuelo o de una planta nuclear. Se presenta artificialmente una situación real y con gran uso de recursos gráficos e interactivos. El hecho de usar simulaciones por computadora, en la enseñanza tradicional ha logrado cambios positivos en los alumnos, en cuanto a la resolución de problemas, ya que brindan la posibilidad de acceso a la enseñanza de temas de difícil comprensión y demostración.
Como aprendices, sostiene Schunk (1997) que las computadoras permiten que los estudiantes aprendan a programar, facilitando el desarrollo de habilidades intelectuales tales como reflexión, razonamiento y resolución de problemas. Lepper (1985) sostiene que las computadoras pueden enseñar ciertas habilidades que no son posibles con los métodos tradicionales, y el aprender a programar ayuda a la resolución de problemas al modelado y división del problema en partes más pequeñas. También a la detección y corrección de errores.
Esta es la filosofía del Logo de Papert, al dar las órdenes en el Logo mediante conjunto de instrucciones que producen ciertas configuraciones, combinando comandos con procedimientos. Las investigaciones actuales destacan que la motivación es un aspecto clave que favorece el procesamiento profundo y no sólo el superficial. (Hopper y Hannafin, 1991).
La otra aplicación es la utilización de las computadoras como herramientas, mediante el uso de procesadores de textos, bases de datos, graficadores, planillas de cálculo y programas de comunicación, etc. Son herra-

mientas que ayudan a ordenar, procesar, almacenar, transmitir información, y que pueden mejorar el aprendizaje de acuerdo al uso que de ellas haga el docente.

1.10. La problemática actual

Existen una serie de problemas detectados y que aún subsisten, en la construcción y uso de mediadores pedagógicos, quizás el más relevante sea el intento de desmistificación de las herramientas informáticas aplicadas por los técnicos, la falta de capacitación docente en el tema específico y el desarrollo tecnológico que se modifica rápida y evolutivamente, así como las reglas y los pasos metodológicos para la creación de software.

Es por ello, que se quiere presentar una propuesta informática para el diseño, desarrollo y evaluación tanto interna como externa, mediante la aplicación de las métricas correspondientes, para determinar los parámetros básicos del proyecto de software educativo, teniendo en cuenta los requerimientos particulares del mismo en cuanto a los aspectos pedagógicos.

En este enfoque disciplinado para el desarrollo de dicho software, se pretende aplicar los métodos, procedimientos y herramientas de la ingeniería del software, los cuales ayudan a asegurar la calidad del mismo.

Como la cantidad y la variedad de software educativo crece muy rápidamente, existe una necesidad de evaluarlo, cada vez mayor, para saber si es adecuado a los propósitos educativos. Los docentes necesitan saber cuándo y cómo un programa puede usarse para mejorar su enseñanza, y los alumnos necesitan saber cómo podrían mejorar sus aprendizajes, en este punto son los vendedores deberían asesorar de acuerdo a las necesidades de uso, y entre varios programas similares en el mercado cuál usar.

Los diseñadores de software educativo necesitan definir criterios a partir de los cuales puede evaluarse y posteriormente llevar a cabo una estrategia de evaluación práctica.

Por otra parte se debe diferenciar qué se pretende englobar con el término calidad: ya sea calidad del software desde el punto de vista técnico o calidad del producto desde el punto de vista educativo o ambas. Debe quedar claro que la calidad es un concepto *"multidimensional y polisémico"* porque es el resultado de una larga lista de factores que van desde la tecnología, los contenidos, el docentes, el currículum, etc.

La calidad del software educativo es cambiante desde la perspectiva de los objetivos. Asimismo, algunos investigadores (Campos 1996, Underwood 1990) mencionan que existe una gran controversia en lo que se refiere a determinar cuándo un software se considera "*educativo*", qué se debería evaluar en un software educativo y qué se considera como un *"software educativo de calidad"*.

Algunos sostienen que la calidad de un software educativo debería responder a un modelo de aprender, un modelo de alumno y el rol de la tecnología en el aula, más bien, a un modelo curricular (Flagg, 1990). Así, no es lo mismo evaluar un software que se utilizará como refuerzo de una clase magistral, que un software de apoyo al trabajo colaborativo de los alumnos.

El problema de la determinación de la calidad en medios de comunicación es un problema recurrente, para el cual numerosos investigadores intentaron definir criterios de calidad del software y compilar clasificaciones y catálogos de ellos. La idea era traducir estos catálogos en listas de verificación que pudieran ser de uso práctico para los docentes al juzgar los medios de comunicación educativos (Baumgartner y Payr, 1996).

Un programa educativo bien diseñado y utilizado ayuda a lograr los "*objetivos educativos",* entre los que se pueden mencionar: incrementar la calidad de la enseñanza que se ofrece a los estudiantes, reducir los costos de la misma, facilitar el acceso a la educación a mayor número de personas, etc.

Existe una diversidad de estudios que denotan la necesidad del uso de herramientas fáciles de usar y bien documentadas para evaluar tanto el software como sus interfaces (Norman y Drapper 1988; Norman 1988; Winograd 1996).

2. El software educativo

2.1. Resumen

En esta sección se presenta una definición de software educativo (sección 2.2), su tipología (sección 2.3) y su clasificación (sección 2.4). Se describen las principales funciones de los programas educativos (sección 2.5). Posteriormente, se analiza el rol del docente al aplicar los diferentes programas, de acuerdo al estilo docente y a la función de los mismos (sección 2.6). Desde el triángulo didáctico se aborda el problema del cambio del rol docente hacia los mediadores pedagógicos (sección 2.7). Luego, se consideran los objetivos educativos a lograr en las intervenciones didácticas (sección 2.8) y los procesos de pensamiento a desarrollar en los alumnos (sección 2.9), considerando aspectos tales como la motivación (sección 2.10), la organización de los contenidos (sección 2.11) y el diseño de las interfaces de comunicación (sección 2.12).

Finalmente, se exponen los puntos claves que debe tener en cuenta una buena planificación didáctica para el uso de los mediadores pedagógicos (sección 2.13).

2.2. Definiciones

Se define como software educativo a *"los programas de computación realizados con la finalidad de ser utilizados como facilitadores del proceso de enseñanza"* y consecuentemente de aprendizaje, con algunas características particulares tales como: la facilidad de uso, la interactividad y la posibilidad de personalización de la velocidad de los aprendizajes.

Marquès (1995) sostiene que se pueden usar como sinónimos *de "software educativo"* los términos *"programas didácticos"* y *"programas educativos"*, cen-trando su definición en *"aquellos programas que fueron creados con fines didácticos, en la cual excluye todo software del ámbito empresarial que se pueda aplicar a la educación aunque tengan una finalidad didáctica, pero que no fueron realizados específicamente para ello".*

Características	Descripción
Facilidad de uso	En lo posible autoexplicativos y con sistemas de ayuda
Capacidad de motivación	Mantener el interés de los alumnos
Relevancia curricular	Relacionados con las necesidades del docente
Versatilidad	Adaptables al recurso informático disponible
Enfoque pedagógico	Que sea actual: constructivista o cognitivista.
Orientación hacia los alumnos	Con control del contenido del aprendizaje
Evaluación	Incluirán módulos de evaluación y seguimiento.

Tabla 2.1: Características principales de los programas educativos, clasificación según Marquès (1998a).

En la Tabla 2.1 se pueden observar algunas de las características principales de los programas educativos. Se da por sentado que los programas deben usarse como recursos que incentiven los proceso de enseñanza y de aprendizaje, con características particulares respecto de otros materiales didácticos y con un uso intensivo de los recursos informáticos de que se dispone. (Marquès, 1998b).

2.3. Las Tipologías

Los programas educativos se pueden clasificar según diferentes tipologías. En la Tabla 2.2 se puede ver algunas de ellas de acuerdo a diferentes criterios.

Se debe considerar que un aspecto clave de todo buen diseño es tomar en cuenta las características de la interface de comunicación, la que deberá estar de acuerdo con la teoría comunicacional aplicada y con las diferentes estrategias para el desarrollo de determinados procesos mentales.

Por otra parte, cuando el software se desarrolla a partir de un lenguaje de programación, ya sea convencional, orientado a eventos u objetos, se tiene que considerar que se fundamenta en la estructura del algoritmo que lo soporta, cuyo diseño deberá reunir algunas características esenciales como la modularidad y el diseño descendente (como se verá en la sección 3).

Gran parte de los programas educativos pertenecen a un sub-grupo denominado hipermediales, y en ellos las bases de datos de imágenes fijas o en movimiento, vídeo clips y sonidos juegan un rol fundamental a la hora de diseñar el programa.

Tipologías según:		
Los contenidos	Temas, áreas curriculares	
Los destinatarios	Por niveles educativos, edad, conocimientos previos	
Su estructura	Tutorial, base de datos, simulador constructor, herramienta	
Sus bases de datos	Cerrados o abiertos	
Los medios que integra	Convencional hipermedia, realidad virtual	
Su inteligencia	Convencional, sistema experto	
Los objetivos educativos que pretende facilitar	Conceptuales, actitudinales, procedimentales	
Las procesos cognitivos que activa	Observación, identificación, construcción memorización, clasificación, análisis, síntesis, deducción, valoración, expresión, creación, etc.	
El tipo de interacción que propicia	Recognitiva, reconstructiva, intuitiva, constructiva	(Kemmis, 1970)
Su función en el	Instructivo, revelador, conjetural, emancipador[6]	

[6] Squires y Mc Dougall (1994) postulan estos cuatro paradigmas.

aprendizaje		
Su comportamiento	Tutor, herramienta, aprendiz	(Taylor, 1980)
El tratamiento de los errores	Tutorial, no tutorial	
Sus bases psicopedagógicas sobre el aprendizaje	Conductista, constructivista, cognitivista	(Gros Begoña, 1997)
Su función en la estrategia di-dáctica	Informar, motivar, orientar, ayudar, proveer recursos, facilitar prácticas, evaluar	
Su diseño	Centrado en el aprendizaje, centrado en la enseñanza, proveedor de recursos	

Tabla 2.2. Algunas tipologías, según Marquès (1998)

2.4. Clasificación de los programas didácticos

Una clasificación factible de los programas puede ser: tutoriales, simuladores, entornos de programación y herramientas de autor.

Los programas *tutoriales*, son programas que *dirigen* el aprendizaje de los alumnos mediante una teoría subyacente conductista de la enseñanza, guían los aprendizajes y comparan los resultados de los alumnos contra patrones, generando muchas veces nuevas ejercitaciones de refuerzo, si en la evaluación no se superaron los objetivos de aprendizaje.

En este grupo, se encuentran los programas derivados de la enseñanza programada, tendientes al desarrollo de habilidades, algunos de ellos son lineales y otros ramificados, pero en ambos casos de base conductual, siendo los ramificados del tipo interactivos.

Se han desarrollado modelos cognitivistas, donde se usa información parcial, y el alumno debe buscar el resto de la información para la resolución de un problema dado.

Dentro de esta categoría, están los sistemas tutoriales expertos o inteligentes, que son una guía para control del aprendizaje individual y brindan las explicaciones ante los errores, permitiendo su control y corrección.

Los programas *simuladores*, ejercitan los aprendizajes inductivo y deductivo de los alumnos mediante la toma de decisiones y adquisición de experiencia en situaciones imposibles de lograr desde la realidad, facilitando el aprendizaje por descubrimiento.

Los *entornos de programación*, tales como el Logo, permiten construir el conocimiento, paso a paso, facilitando al alumno la adquisición de nuevos conocimientos y el aprendizaje a partir de sus errores; y también conducen a los alumnos a la programación.

Las *herramientas de autor,* también llamadas *"lenguajes de autor"* permiten a los profesores construir programas del tipo tutoriales, especialmente a profesores que no disponen de grandes conocimientos de programación e informática, ya que usando muy pocas instrucciones, se pueden crear muy buenas aplicaciones hipermediales.

Algunos autores consideran que las *bases de datos para consulta*, son otro tipo de programas educativos, porque faciltan la exploración y la consulta selectiva, permitiendo extraer datos relevantes para resolver problemas, analizar y relacionar datos y extraer conclusiones. (Marquès, 1995).

Quedarían por analizar los programas usados como *herramientas de apoyo* tales como los procesadores de textos, planillas de cálculo, sistemas de gestión de bases de datos, graficadores, programas de comunicación, que no entran dentro de la clasificación de educativos, pero muchas veces son necesarios para la redacción final de trabajos, informes y monografías.

En la búsqueda permanente del mejoramiento de los procesos de enseñanza y de aprendizaje, se encuentra una herramienta poderosísima *en los sistemas hipermediales*, como un subconjunto del software educativo en general.

Se puede definir un sistema hipermedial como la combinación de hipertexto y multimedia.

Se entiende por *hipertexto al sistema de presentación de textos extensos con o sin imágenes donde se puede adicionar sonido, formando una red con nodos que son unidades de información, con enlaces y arcos dirigidos hacia otros nodos, la red no es más que un grafo orientado, que se aparta de la forma secuencial tradicional del libro. Multimedia es la presentación de la información con grandes volúmenes de texto, con imágenes fijas, dibujos con animación y vídeo digital. Por lo tanto la hipermedia es la combinación de hipertexto y multimedia.* (Nielsen,1995).

Algunos autores como García López (1997) sostienen que a pesar de que el multimedia interactivo es anterior a la aparición de las redes y a la realidad virtual y que el prefijo *"hiper"* engloba también a dichas fusiones interactivas**.**

2.5. Las funciones del software educativo

Las funciones del software educativo, están determinadas de acuerdo a la forma de uso de cada profesor. En la Tabla 2.3, se describen en forma sintética algunas de las funciones que pueden realizar los programas.

2.6. El rol docente y los usos del software.

El estilo docente ha cambiado a causa de la introducción de las computadoras en el aula, desde el tradicional suministrador de información, mediante clases magistrales a facilitadores, pudiendo de este modo realizar un análisis más preciso del proceso de aprendizaje de sus alumnos y una reflexión acerca de su propia práctica.

Función	Descripción
Informativa	Presentan contenidos que proporcionan una información estructurado-ra de la realidad. Representan la realidad y la ordenan. Son ejemplos, las bases de datos, los simuladores, los tutoriales.
Instructiva	Promueven actuaciones de lo estudiantes encaminadas a facilitar el logro de los objetivos educativos, el ejemplo son los programas tutoriales.
Motivadora	Suelen incluir elementos para captar en interés de los alumnos y enficarlo hacia los aspectos más importantes de las actividades.
Evaluadora	Al evaluar implícita o explícitamente, el trabajo de los alumnos.
Investigadora	Los más comunes son: las bases de datos, los simuladores y los entornos de programación.
Expresiva	Por la precisión en los lenguajes de programación, ya que el entorno informático, no permite ambigüedad expresiva.
Metalingüística	Al aprender lenguajes propios de la informática.
Lúdica	A veces, algunos programas refuerzan su uso, mediante la inclusión de elementos lúdicos.
Innovadora	Cuando utilizan la tecnología más reciente.

Tabla 2.3: funciones del software educativo según Marquès (1995)

Los *"mediadores pedagógicos"*, son el vínculo entre los estudiantes (sujetos) y los contenidos. La concepción tradicional de docente informante, ha cambiado hacia el facilitador o guía y tutor, y una nueva perspectiva es el uso de mediadores tales como los programas educativos, sean o no hipermediales, con toda la gama de posibles matices intermedios.

Cuando se desea aplicar un software educativo en un contexto áulico, se debe tener en cuenta, que para algunas asignaturas resulta más difícil incorporar el recurso informático al aula. Estas formas de incorporación están directamente relacionadas con las diferentes actitudes del docente, de acuerdo a su estilo, como se puede observar en la Tabla 3.4.

Magistral o de informante	El docente deja de ser la fuente principal de información de la clase.
Auxiliar	El docente conserva su función de informante, articulando diferentes medios.
Aplicativa	Se integra el rol del docente y se consolida el trabajo individual y grupal
Interactiva	Se favorece la comunicación, la construcción conjunta del conocimiento.

Tabla 3.4: el rol docente y el software educativo.

Los nuevos entornos de enseñanza y aprendizaje, exigen nuevos roles en profesores y alumnos, la perspectiva tradicional en todos los niveles educativos y especialmente en la educación superior del profesor como fuente única de información se ha transformado hacia un del profesor guía y consejero acerca del manejo de las fuentes apropiadas de información y desarrollador de destrezas y hábitos conducentes a la búsqueda, selección y tratamiento de la información.

Los estudiantes ya no son *receptores pasivos*, sino que se convierten en *alumnos activos* en la búsqueda, selección, procesamiento y asimilación de información.

La concepción tradicional ha cambiado hacia una *cultura del aprendizaje*, o sea una educación generalizada y una formación permanente, dentro de una avalancha constante de información. Es en esta cultura del aprendizaje, en la que el profesor debe encarar el rol de *gerenciador de los saberes y desarrollador de habilidades* que permitan a sus alumnos utilizar el análisis crítico y reflexivo.

2.7. Las funciones del profesor y los materiales didácticos

Los materiales didácticos, se pueden definir como *"el conjunto de medios materiales que intervienen en el acto didáctico, facilitando los procesos de enseñanza y de aprendizaje"*. Sus fines centrales persiguen facilitar la comunicación entre el docente y el estudiante para favorecer a través de la intuición y el razonamiento un acercamiento comprensivo de las ideas a través de los sentidos. (Eisner, 1992). Estos materiales didácticos constituyen la variable dependiente del proyecto pedagógico y del entorno de aprendizaje que se trate.
La utilización de software educativo como material didáctico, cambia la manera en la cual los profesores estimulan el aprendizaje en sus clases; cambia el tipo de interacción entre alumnos y docentes y por lo tanto cambia el rol y las funciones del profesor. En la Tabla 2.5 se presenta un resumen de dichas funciones:

Función	Características
Como proveedor de recursos	Muchas veces el profesor tiene que adaptar los materiales de un cierto paquete educativo a las características de la clase y a los fines que él plantea en ese momento.
Como Organizador	Cuando se usan computadoras, hay muchas formas de organizar su uso en el aula y variando de acuerdo a los diferentes estilos docentes. También se debe tener en cuenta la graduación del tiempo de interacción con las máquinas, ya que es en los diálogos en clase donde se produce gran parte del aprendizaje.
Como tutor	Hay profesores que usan un software para centrar las actividades. El profesor trabaja con un sólo alumno o un grupo pequeño, realizando actividades de tutoría como: razonar y buscar modelos o respuestas.
Como Investigador	A nivel áulico, el uso de software puede dar a los profesores ideas sobre los proceso de aprendizaje y de las dificultades de sus alumnos. En este papel de investigadores, los docentes, usan al software como una herramienta diagnóstica.
Como facilitador	Esta es la responsabilidad principal del docente, como facilitadores del aprendizaje de los estudiantes y la que no debe olvidarse, con la aparición de las demás funciones que surgen con la introducción del uso de las computadoras en el aula.

Tabla 2.5: Las funciones del profesor (Squires y Mc Dougall, 1994)

2.8. Los objetivos educativos[7]

Se entiende por objetivo *"algo"*[8] que se quiere lograr, o sea un estado al cual se quiere arribar (aunque no siempre hay que pensarlo desde el punto de vista conductual). A fin de enunciar correctamente un objetivo, de manera que sea tal y no la mera expresión de un deseo, es necesario que existan en él los tres elementos siguientes:

- *Intención:* Es el fin de todo objetivo. La intención debe ser clara y estar concretamente expresada en el enunciado del objetivo, debe enunciar con toda certeza y precisión qué se propone alcanzar. La intención debe ser no sólo concreta, sino también real. Un objetivo, al enunciar una intención debe proponer un fin concreto, de esta manera se podrá determinar con toda exactitud cuándo se logró alcanzar el fin propuesto. Si la intención no es concreta, nunca se podrá saber si el objetivo está cumplido.
- *Medida:* Es el elemento que vuelve al objetivo mensurable y esa cualidad de ser mensurable es la otorga la certeza de cumplimiento.
- *Plazo:* Es el período durante el cual debe lograrse el objetivo.

La *formulación de los objetivos sirve* para:

- *Fijar la situación actual:* El hecho de determinar un estado final a lograr, obliga, indefectiblemente, a fijar una situación actual. Si se quiere lograr algo en el futuro, se debe partir de una determinada situación en el presente. Aquí es donde se pone de manifiesto la importancia de la evaluación inicial o diagnóstica.[9]
- *Determinar el estado final a lograr:* Por medio de evaluaciones sumativas[10] o finales.
- *Determinar las estrategias a emplear:* Si se tiene una situación actual y un estado final, es evidente que se hace necesario un accionar que permita lograrlo. Las estrategias son opciones alternativas con un gran número de posibilidades diferentes. En la selección de las mismas se tienen en cuenta, además de su efi-

[7] Esta sección forma parte de un Trabajo presentado a la Dra. Beatriz Fainholc en el *Seminario de Sistemas Multimediales Aplicados a la Educación*. UTN. Buenos Aires. 1998.
[8] depende de qué sea este "*algo*", serán distintos los aspectos buscados.
[9] diseñada en base a una serie ejercicios específicos, para saber en qué etapas de su desarrollo se encuentra el alumno, especialmente si está en la etapa de desarrollo formal, según la visión de piagetiana.
[10] de producto final tal como se refieren Bork (1986) y Coll (1994).

ciencia: costo, tiempo de acción, sencillez, facilidades operativas, requerimiento de laboratorio, de biblioteca, sistemas informáticos, etc.
- *Medir los resultados:* Mediante evaluaciones formativas[11] (de procesos) y sumativas o finales. Puede ser parcial y sumativa, no necesariamente formativa.

2.9. Las actividades de comprensión a desarrollar por los alumnos

Entre las *actividades de comprensión* o *"procesos de pensamiento*" que los alumnos pueden desarrollar al interactuar con los programas educativos, se pueden mencionar:
- Explicar relaciones causa efecto.
- Formular conclusiones válidas.
- Describir limitaciones de los datos.
- Confrontar conocimientos nuevos con previos.
- Clasificar y seleccionar información.
- Producir, organizar y expresar ideas.
- Elaborar mapas conceptuales (teniendo en cuenta la reconciliación integradora y la diferenciación progresiva)
- Integrar el aprendizaje en diferentes áreas.
- Inferir correctamente.
- Evaluar el grado de adecuación de las ideas.
- Presentar argumentos pertinentes frente a fenómenos.
- Defender un punto de vista y fundamentar criterios.
- Resolver problemas elaborando estrategias metacognitivas.

La comprensión, implica el compromiso reflexivo del alumno con el contenido de enseñanza y la habilidad para articular significativamente el material comunicado por acciones de guía (Cediproe, 1998).
Entre *los objetivos de los programas educativos* se pueden mencionar:
- Crear expectativas en el estudiante y estimular la planificación de su aprendizaje.
- Dirigir la atención del estudiante y permitir que inicie su aprendizaje por diferentes caminos de acceso. (tiene gran importancia desde lo cognitivo).
- Asegurar situaciones de aprendizaje significativo.
- Aprovechar la posibilidad de usar imágenes, animaciones, simulaciones y sonidos.
- Desarrollar y hacer consciente el uso de diferentes estrategias:
  - de procesamiento de la información.
  - de producción y uso de la información.
  - De recreación de la información.
- Estimular la generalización y transferencia de lo aprendido.
- Ofrecer situaciones de resolución de problemas.
- Proveer retroalimentación constante e informar acerca de los progresos en el aprendizaje. (Zangara, 1998).

2.10. La motivación

Alessi y Trollip (1985), consideran que existe una motivación extrínseca independiente del programa utilizado, y una intrínseca inherente en la instrucción y recomiendan criterios para su promoción, como el uso de juegos, de exploración, de desafíos, incentivación de la curiosidad del estudiante, teniendo en cuenta un balance entre la motivación y el control del programa aplicado.
Las bases teóricas pueden ser provistas por alguna de las teorías de la motivación permitiendo crear desafíos, curiosidad, control y fantasía y con un diseño motivacional que mantenga la atención a través del mismo. Los estudiantes deben poder ver la utilidad de resolución de problemas.
Ausubel (1987) sostiene que el papel de la motivación en el aprendizaje es uno de los problemas más controvertidos de los teóricos de la psicología, y que aún las posiciones son muy encontradas. En la Tabla 2.6, se pueden ver la clasificación de los diferentes tipos de motivación.

Tipos	Características
Intrínseca	Es la que proviene del interior del sujeto por su compromiso con la tarea.
Relacionada con el yo	Se relaciona con la autoestima, con el no percibirse inferior que los demás
Centrada en la	Se relaciona con la satisfacción afectiva que produce la aceptación, aprobación

[11] de proceso o parcial, tal como se refieren Bork (1986) y Coll (1994).

valoración social	o aplauso por parte de personas consideradas superiores.
Extrínseca	Centrada en recompensas externas, se relaciona con premios y/o castigos

Tabla 2. 6: Tipos de motivación (Guiraudo, 1997)

La motivación intrínseca es superior a la extrínseca y para lograrla, quizás la manera más eficaz es mediante el entusiasmo propio del docente por lo que hace.
Para ello se debe considerar la creación de nuevos intereses en los alumnos como uno de los objetivos de la intervención pedagógica, teniendo en cuenta la escala motivacional de Maslow[12] con necesidades fisiológicas, de supervivencia, de seguridad, de amor, de pertenencia, de aceptación, de autoestima, de autorrealización.

2.11. La organización y presentación de los contenidos

La selección de los contenidos, es uno de los problemas recurrentes en educación que comienzan con el planteo del docente de qué enseñar, para qué enseñar y cómo enseñar.
En el análisis del *"qué enseñar"*, de acuerdo a los *"principios básicos",* ejes de todo el desarrollo, el docente que va a desarrollar software o que trabaja en un equipo de desarrollo, debe seleccionar la información a presentar y transmitir, determinando los contenidos y también su organización que dependerá de la subdivisión del eje temático principal en bloques de contenido y en sub-bloques.
La organización en bloques y sub-bloques se realizará de tal forma que permitan de navegación en sentido horizontal, vertical y transversal y deberán estar de acuerdo a las diferentes estrategias de búsqueda que se preparen desde alguna de las visiones de los diferentes paradigmas educativos.
Esta organización será acorde con el diseño de las pantallas más adecuado en cada caso, para la presentación de los contenidos.

2.12. La comunicación: Las interfaces humanas.

Gallego y Alonso (1997), ofrecen una guía metodológica para el diseño pedagógico de la interface de navegación, destacando la necesidad de un diseño adecuado tanto de la organización de los contenidos como de las estrategias de enseñanza y de aprendizaje. Esta interface es fundamental, ya que es el sistema de recursos mediante el cual el usuario interactúa con el sistema informático. Estos recursos implican tener en cuenta aspectos técnicos, de funcionamiento de la interface y también los cognitivos y emocionales resultantes de la interacción usuario-computadora.
El diálogo entre el usuario y el sistema informático debe ser lo más sencillo posible y debe proveerle los recursos necesarios para la navegación y obtención de la información buscada.
La interface es el elemento clave de comunicación o aspecto fundamental de diseño y presentación de los contenidos. Actualmente, se diseñan interfaces orientadas al usuario, lo más cercanas posible al lenguaje humano, incluyendo el modo de presentar la información en la pantalla y las funcionalidades brindadas al usuario para interactuar con el programa.
Según Gallego y Alonso (1997), las características principales de una interface orientada al usuario deben ser:

- Facilidad de manejo: la mejor interface de usuario es aquella que requiere el menor esfuerzo de aprendizaje.
- Originalidad: para promover la motivación y exploración.
- Homogeneidad: requiere de una interface con funciones claras para moverse de en el programa, incluyendo un mapa general.
- Versatilidad: que pueda incorporar nuevas funciones específicas.
- Adaptabilidad: deberá ofrecer modalidades de navegación de acuerdo al contenido, los destinatarios y el nivel de profundidad.
- Multimodalidad: con integración de modalidades de comunicación necesaria para cada concepto.
- Multidimensionalidad: para los diseños hipermediales.
- Agilidad: para que la interacción sea dinámica.
- Transparencia: cuanto más natural sea, será más fácil para el usuario acceder a los contenidos.
- Interactividad: darle al usuario un papel protagónico.
- Conectividad: para utilizar redes.

Respecto de las funciones, la interface debe tener una triple funcionalidad: utilidades, navegación e información.
En su artículo sobre los agentes de interface, Brenda Laurel (1990) señala como principales características de las mismas: son dar respuestas, actuar como agente, competencia y accesibilidad.

[12] Escala de Maslow A. H. (1943): "A theory of human motivation", Psychological Review, July, págs. 370-396.

La metáfora navegacional a aplicar estará condicionada por el tipo de contenido, las características de los destinatarios y el lenguaje o herramienta de autor usado para desarrollar el software. Las metáforas más utilizadas son las de los menús: cerrados, abiertos o mixtos y las de los iconos; en este caso su utilización es mucho más intuitiva. La metáfora espacial, es aquella que usa la realidad como modelo, con escenarios que simulan la realidad misma. Un modelo de interface espacial son los paisajes de información, este modelo incluye conjuntos de datos, documentos interactivos, recorridos guiados, películas y actividades.
Como no hay una metáfora ideal de menú principal de usuario, se trata de brindarle una combinación de todas ellas dando al mismo la posibilidad de realizar su elección.
Las metáforas navegacionales están asociadas a las diferentes estrategias de aprendizaje. Cuando se preparan programas totalmente interactivos, ramificados, con caminos de aprendizaje múltiples a elección del alumno, los estilos de aprendizaje pueden convertirse en un elemento más a tener en cuenta en el diseño didáctico (Alonso, 1992).
Las funciones de navegación permiten saber al usuario dónde está en cada momento, de dónde viene y a dónde puede ir. Los modelos de organización de la información para estructurar los contenidos de las aplicaciones educativas son muy diversos. Florín (1990) plantea una estructura multidimensional que permite al usuario acceder a la información sobre la base de distintos intereses.
La metodología recomendada por Gallego y Alonso (1997), para aplicar la interface al ámbito educativo y la formación, se basa en los siguiente principios:
- Ofrecer al usuario la posibilidad de que se sienta protagonista.
- Presentar los contenidos de forma atractiva y de fácil manejo.
- Combinar diferentes metáforas de navegación interactivas.
- Prever diversas funcionalidades de la interface de navegación en función del tipo de contenido, del destinatario y de los niveles de profundidad previstos.
- Considerar las normas de calidad en el diseño.

Las principales especificaciones de una interface de aprendizaje son:
- Facilidad de manejo.
- Ayudas alternativas.
- Sistema de seguimiento del alumno que permita el diagnóstico de progreso realizado en función del grado de logro de los objetivos.

En la Tabla 2.7 se pueden observar una clasificación de los diferentes tipos de pantallas a utilizar de acuerdo a los objetivos didácticos perseguidos.

Tipos de pantallas	Objetivos didácticos
Presentación del programa	– Captar la atención – Generar, dirigir, motivar y/o aumentar la motivación
Pantallas de antesala o de anticipación	– Anticipar los conceptos a aprender
Pantallas de presentación de información simple	– Presentar información: – Nueva y relevante – Relacionada con algún concepto posterior
Pantallas de presentación de información compleja: relación de información simple	– Integrar los conceptos en conceptos complejos
Pantallas de integración y síntesis de la Información	– Integrar los conceptos en categorías
Pantallas de actividades y resolución de problemas	– Autoevaluar gradualmente el aprendizaje – Reorganizar y aplicar la nueva información – Transferir el aprendizaje a situaciones nuevas
Pantallas de presentación de información de control	– Informar acerca de la marcha del aprendizaje
Interface de acceso a otras fuentes de Información	– Acceder a fuentes complementarias de información – Realizar consultas a tutores – Relacionarse virtualmente con otros compañeros de estudios.

Tabla 2.7: Objetivos de los diferentes tipos de pantallas (Zangara, 1998)

2.13. La planificación didáctica.

Finalmente, una buena planificación didáctica para aplicación de un programa de computadora debe considerar aspectos tales como:

- *La inserción del programa en el currículum:* se deberá indicar para qué nivel educativo está dirigido el software y si está de acuerdo a un determinado currículum.
- *Los objetivos perseguidos:* constituyen el *"para qué"* de la propuesta educativa y la dirección de toda la acción educadora. César Coll (1994) dice que es la conducta esperable y que depende de la teoría del aprendizaje. Coll lo plantea como estrategias de pensamiento que se desea que el alumno realice, puntualizando las aspiraciones a corto y a largo plazo Ausubel (psicólogo cognitivo) habla de predisposición sin referirse a los procedimientos, usando estrategias cognitivas. Ampliando el esquema propuesto por Romiszowski (1981) en Coll (1994), quien estableció que a la concreción de las intenciones educativas puede accederse desde los contenidos, desde los resultados o desde las actividades, se debe agregar la posibilidad de acceder al conocimiento desde los medios, que atraviesan la realidad desde una visión tecnológica. Esta visión consiste en abordar la educación desde el paradigma *teleinformático*. Cuando se plantean los objetivos tanto para una asignatura, como en este caso de un software de un determinado tema en particular, el objetivo es el estado final logrado a partir de un estado inicial definido, este estado final real no siempre coincide con el valor teórico o probable a alcanzar en un tiempo definido. Existe un grado de apartamiento que es cuantificable y minimizar este apartamiento sería deseable.
- *Las características de los destinatarios:* hay que realizar una descripción en téminos de edad, prerrequisitos de contenidos y habilidades, nivel educativo formal o informal.
- *Los contenidos desarrollados:* los contenidos se pueden abordar de distintas maneras. Desde el punto de vista cognitivo los contenidos son casi más importantes que los objetivos, consiste en una delimitación de qué. Un ejemplo son las estructuras de mapas conceptuales como un representación gráfica de las relaciones de conceptos y el aprendizaje significativo. La estrategia de trabajo de Novak (1988) es el armado de mapas conceptuales para la toma de desiciones.
- *Metodología y actividades a desarrollar:* aquí el docente debe determinar de acuerdo a su metodología de aplicación del programa, cuáles son las actividades que va a desarrollar con sus alumnos, indicando si usará el software como material de apoyo, por ejemplo, si utilizará proyecciones como complementos y una sola computadora, o si los alumnos trabajarán en grupos o en forma individual. También debe quedar claro cuáles son los procesos de pensamiento que se pretende desarrollar en los alumnos a partir de la interacción como por ejemplo: comparar, discriminar, resumir, globalizar, analizar, concatenar, experimentar, construir, negociar, discutir, investigar, evaluar, etc.
- *Recursos necesarios, medios y tiempo de interacción:* en la planificación didáctica deben quedar especificados los recursos necesarios, los medios indispensables y el tiempo que durará la interacción con el software. En el caso particular de un software realizado por encargo y para apoyo del docente, no se puede cuantificar en forma precisa este tiempo, como para arribar a un resultado óptimo. Cuando se habla de software de apoyo, el tiempo de interacción del alumno con el programa mediado en términos absolutos no sirve, porque el programa fue diseñado para usarlo de soporte, con complementos por parte del docente, que no forman parte del programa mismo. Por este motivo, para un alumno principiante en el tema, el software se potencia con las explicaciones adicionales del docente, pero si luego queda a disposición de los alumnos que pueden usarlo y verlo cuantas veces deseen, hasta lograr dominio del tema la estimación del tiempo aquí, carece de sentido.
- *Evaluación de los aprendizajes:* la instancia de evaluación del proceso de enseñanza y aprendizaje, para este tipo de producto, es quizás la más difícil, ya que evaluar un software significa basarse en los resultados alcanzados por los alumnos en las pruebas diseñadas de acuerdo a la teoría educativa aplicada. Ya sea mediante acercamiento a los objetivos, o por desarrollo y estimulación de procesos mentales y significatividad de aprendizajes.

3. La ingeniería de software

3.1. Resumen

En esta sección se desea presentar los fundamentos en los que se basa el software educativo (sección 3.2): los métodos, las herramientas y los procedimientos que provee la ingeniería de software a fin de considerarlos para el desarrollo de los programas didácticos. Se describen y analizan los paradigmas principales del ciclo de vida (sección 3.2) a la luz de la visión de Mario Piattini, desde la cascada tradicional hasta los actuales orientados a objetos (sección 3.3).

Se destaca la necesidad de una metodología para el desarrollo de productos lógicos y se describen las más importantes (sección 3.4). A fin de seleccionar el ciclo de vida adecuado para cada desarrollo, se analizan las actividades de cada uno de los procesos del mismo (sección 3.5). Por último se define calidad del software y la normativa vigente (sección 3.6) para un proyecto de software y se hace una revisión de las métricas de calidad comúnmente usadas.

3.2. Fundamentos

Uno de los problemas más importantes con los que se enfrentan los ingenieros en software y los programadores en el momento de desarrollar un software de aplicación, es la falta de marcos teóricos comunes que puedan ser usados por todas las personas que participan en el desarrollo del proyecto informático.
El problema se agrava cuando el desarrollo corresponde al ámbito educativo debido a la inexistencia de marcos teóricos interdisciplinarios entre las áreas de trabajo.
Si bien, algunos autores como Galvis (1996) reconocen la necesidad de un marco de referencia, teniendo en cuenta que se debe lograr la satisfacción de los requisitos en las diversas fases del desarrollo, de lo que constituye un material didáctico informatizado; esta necesidad sigue vigente, aunque en la mayoría de los casos analizados, se trata de software hipermedial diseñado a partir de herramientas de autor.
Marquès (1995), es otro de los autores que plantean un ciclo de desarrollo para software educativo de programas en diez etapas, con una descripción detallada de las actividades y recursos necesarios para cada una de ellas. El inconveniente principal de esta metodología es que centra el eje de la construcción de los programas educativos en el equipo pedagógico, otorgándole el rol protagónico.
Es por este motivo, que en este capítulo se sintetizan las metodologías, métodos, herramientas y procedimientos de la ingeniería de software, que deben ser utilizados para lograr un producto de mejor calidad desde el punto de vista técnico. Su conocimiento y aplicación conjuntamente con las teorías: educativa, epistemológica y comunicacional permitirán el logro de un producto óptimo desde el punto de vista educativo.
Cabe recordar una de las primeras definiciones de ingeniería de software propuesta por Fritz Bauer en la primera conferencia importante dedicada al tema (Naur, 1969) como: *"El establecimiento y uso de principios de ingeniería robustos, orientados a obtener software económico y que funcione de manera eficiente sobre máquinas reales"*.
Posteriormente se han propuesto muchas definiciones destacando la importancia de base teórica ingenieril para el desarrollo del software.
"La ingeniería del software surge a partir de las ingenierías de sistemas y de hardware, y considera tres elementos clave: que son los métodos, las herramientas y los procedimientos que facilitan el control del proceso de desarrollo de software y brinda a los desarrolladores las bases de la calidad de una forma productiva". (Pressman, 1993).
La ingeniería de software está compuesta por una serie de modelos que abarcan los métodos, las herramientas y los procedimientos. Estos modelos se denominan frecuentemente *paradigmas de la ingeniería del software* y la elección de un paradigma se realiza básicamente de acuerdo a la naturaleza del proyecto y de la aplicación, los controles y las entregas a realizar.
Debido a las características particulares de los desarrollos educativos, ya que se deben tener en cuenta los aspectos pedagógicos y de la comunicación con el usuario, en cada caso en particular, la respuesta a la problemática debe basarse en una adaptación de los actuales paradigmas de desarrollo a las teorías de aprendizaje que permitan satisfacer una demanda en especial.
Para la construcción de un sistema de software, el proceso puede describirse sintéticamente como: la obtención de los requisitos del software, el diseño del sistema de software (diseño preliminar y diseño detallado), la implementación, las pruebas, la instalación, el mantenimiento y la ampliación o actualización del sistema.
El proceso de construcción está formado por etapas que son: la obtención de los requisitos, el diseño del sistema, la codificación y las pruebas del sistema. Desde la perspectiva del producto, se parte de una necesidad, se especifican los requisitos, se obtiene el diseño del mismo, el código respectivo y por último el sistema de software. Algunos autores sostienen que el nombre *ciclo de vida* ha sido relegado en los últimos años, utilizando en su lugar *proceso de software*, cambiando la perspectiva de producto a proceso. (J. Juzgado, 1996)
El software o producto, en su desarrollo pasa por una serie de etapas que se denominan ciclo de vida, siendo necesario, definir en todas las etapas del ciclo de vida del producto, los procesos, las actividades y las tareas a desarrollar.
Por lo tanto, se puede decir que: *"se denomina ciclo de vida a toda la vida del software, comenzando con su concepción y finalizando en el momento de la desinstalación del mismo*". (Sigwart et al., 1990), aunque a veces, se habla de ciclo de desarrollo, para denominar al subconjunto del ciclo de vida que empieza en el análisis y finaliza la entrega del producto.
Un ciclo de vida establece el orden de las etapas del proceso de software y los criterios a tener en cuenta para poder pasar de una etapa a la siguiente.
El tema del ciclo de vida ha sido tratado por algunas organizaciones profesionales y organismos internacionales como la IEEE (Institute of of Electrical and Electronics Engineers) y la ISO/IEC (International Standards Organization/International Electrochemical Commission), que han publicado normas tituladas *"Standard for Developing Software Life Cycle Processes"* (Estándar IEEE para el desarrollo de procesos del ciclo de vida del software) (IEEE, 1991) y *"Software life-cycle process"* (Proceso de ciclo de vida del software) (ISO, 1994).
Según la norma 1074 IEEE se define al ciclo de vida del software como *"una aproximación lógica a la adquisición, el suministro, el desarrollo, la explotación y el mantenimiento del software"* y la norma ISO 12207

define como modelo de ciclo de vida al *"marco de referencia, que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de requisitos hasta la finalización de su uso"*. Ambas consideran una actividad como un subconjunto de tareas y una tarea como una acción que transforma las entradas en salidas. (Piattini, 1996).

3.3. Los procesos del ciclo de vida del software

Según la norma ISO 12207-1, las actividades que se pueden realizar durante el ciclo de vida se pueden agrupar en cinco procesos principales, ocho de soporte y cuatro procesos generales de la organización, así como un proceso que permite adaptar el ciclo de vida a cada caso concreto.
En la Tabla 3.1 se describen los grupos de procesos citados.

Procesos Principales (Aquellos que resultan útiles a las personas que inician o realizan el desarrollo, explotación o mantenimiento durante el ciclo de vida).	Adquisición	Contiene las actividades y tareas que el usuario realiza para comprar un producto
	Suministro	Contiene las actividades y tareas que el suministrador realiza
	Desarrollo	Contiene las actividades de análisis de requisitos, diseño, codificación, integración, pruebas, instalación y aceptación.
	Explotación	También se denomina operación del software.
	Mantenimiento	Tiene como objetivo modificar el software manteniendo su consistencia.
Procesos de soporte (se aplican en cualquier punto del ciclo de vida)	Documentación	Registra la información producida en cada proceso o actividad del ciclo de vida
	Gestión de la configuración	Aplica procedimientos para controlar las modificaciones
	Aseguramiento de la calidad	Para asegurar que todo el software cumple con los requisitos especificados de calidad.
	Verificación	Para determinar si los requisitos están completos y son correctos.
	Validación	Para determinar si cumple con los requisitos previstos para su uso.
	Revisión	Para evaluar el estado del software en cada etapa del ciclo de vida
	Auditoría	Para determinar si se han cumplido los requisitos, planes y el contrato.
	Resolución de los problemas	Para asegurar el análisis y la eliminación de problemas encontrados durante el desarrollo.
Procesos de la Organización (Ayudan a la organización en general).	Gestión	Contiene actividades genéricas de la organización como planificación, seguimiento, control, revisión y evaluación.
	Mejora	Sirve para establecer, valorar, medir, controlar y mejorar los procesos del ciclo de vida del software.
	Infraestructura	Incluye la infraestructura necesaria: hardware, software, herramientas, técnicas, normas e instalaciones para el desarrollo, la explotación o el mantenimiento.
	Formación	Para mantener al personal formado: incluyendo el material de formación y el plan de formación.

Tabla 3.1: Los procesos del ciclo de vida del software según ISO 12207-1

3.3.1. El modelo en cascada

La versión original del modelo en cascada, fue presentada por Royce en 1970, aunque son más conocidos los refinamientos realizados por Boehm (1981), Sommerville (1985) y Sigwart et al. (1990). En este modelo, el producto evoluciona a través de una secuencia de fases ordenadas en forma lineal, permitiendo iteraciones al estado anterior. El número de etapas suele variar, pero en general suelen ser:

- Análisis de requisitos del sistema.
- Análisis de requisitos del software.
- Diseño preliminar.
- Diseño detallado.
- Codificación y pruebas.

– Explotación (u operación) y mantenimiento.

Las características de este modelo son:

– Cada fase empieza cuando se ha terminado la anterior.
– Para pasar a la fase posterior es necesario haber logrado los objetivos de la previa.
– Es útil como control de fechas de entregas.
– Al final de cada fase el personal técnico y los usuarios tienen la oportunidad de revisar el progreso del proyecto.

Mc Cracken y Jackson (1982) han realizado algunas críticas al modelo:

– Sostienen que los proyectos reales rara vez siguen una linealidad tal, y que casi siempre hay iteraciones que van más allá de la etapa anterior.
– Además, como el sistema no estará en funcionamiento hasta finalizar el proyecto, el usuario, recibe el primer producto al haber consumido casi la totalidad de los recursos.

Otra limitación que se argumenta es que el modelo supone que los requisitos pueden ser "congelados" antes de comenzar el diseño y esto significa un hardware asociado durante el tiempo que dure el proyecto.

3.3.2. El modelo incremental, de refinamiento sucesivo o mejora iterativa.

Las etapas son las mismas que en el ciclo de vida en cascada y su realización sigue el mismo orden, pero corrige la problemática de la linealidad del modelo en cascada. Este modelo incremental fue desarrollado por Lehman (1984), y en cada paso sucesivo agrega al sistema nuevas funcionalidades o requisitos que permiten el refinado a partir de una versión previa. El modelo es útil cuando la definición de los requisitos es ambigua y poco precisa, porque permite el refinamiento, o sea se pueden ampliar los requisitos y las especificaciones derivadas de la etapa anterior.

Uno de los problemas que puede presentar es detección de requisitos tardíamente, siendo su corrección tan costosa como en el caso de la cascada.

3.3.3. Prototipado evolutivo

El uso de prototipos se centra en la idea de ayudar a comprender los requisitos que plantea el usuario, sobre todo si este no tiene una idea muy acabada de lo que desea. También pueden utilizarse cuando el ingeniero de software tiene dudas acerca de la viabilidad de la solución pensada.Esta versión temprana de lo que será el producto, con una funcionalidad reducida, en principio, podrá incrementarse paulatinamente a través de refinamientos sucesivos de las especificaciones del sistema, evolucionando hasta llegar al sistema final.

Al usar prototipos, las etapas del ciclo de vida clásico quedan modificadas de la siguiente manera:

– Análisis de requisitos del sistema.
– Análisis de requisitos del software.
– Diseño, desarrollo e implementación del prototipo
– Prueba del prototipo.
– Refinamiento iterativo del prototipo.
– Refinamiento de las especificaciones del prototipo.
– Diseño e implementación del sistema final.
– Explotación (u operación) y mantenimiento.

Si bien el modelo de prototipos evolutivos, fácilmente modificables y ampliables es muy usado, en muchos casos pueden usarse prototipos descartables para esclarecer aquellos aspectos del sistema que no se comprenden bien. (J. Juzgado, 1996).

3.3.4. El modelo en espiral de Boehm

En 1988 Boehm propone el modelo en espiral, para superar algunas de las limitaciones del modelo en cascada. La espiral se forma a partir de una serie de ciclos de desarrollo y va evolucionando. Los ciclos internos de la espiral denotan análisis y prototipado y los externos el modelo clásico. En la dimensión radial están los costos acumulativos y la dimensión angular representa el progreso realizado en cada etapa.

En cada ciclo se empieza identificando los objetivos, las alternativas y las restricciones del mismo. Se deben evaluar las alternativas de solución respecto de los objetivos, considerando las restricciones en cada caso. Es en este momento en que se puede llevar a cabo el siguiente ciclo.

Una vez finalizado, comienza el planteo de un nuevo ciclo. Durante cada ciclo de la espiral, aparece el análisis de riesgos, identificando situaciones que pueden hacer fracasar el proyecto, demorarlo o incrementar su costo. El análisis de riesgo representa la misma cantidad de desplazamiento angular en cada etapa y el volumen barrido denota el incremento de los niveles de esfuerzo requeridos para el análisis de riesgo. (Boehm, 1988)

Pueden resumirse las siguientes ventajas respecto de los modelos anteriores:

– Se explicitan las diferentes alternativas posibles para lograr los objetivos.

- El modelo tiene en cuenta la identificación de los riesgos para cada alternativa y los modos de controlarlos.
- Este modelo es adaptable a desarrollos de todo tipo y no establece una diferencia entre desarrollo de software y mantenimiento del sistema

El modelo en espiral se adapta bien en la mayoría de los casos. En el caso de proyectos de riesgo, se hace necesaria la presencia de un experto en evaluación de riesgos para identificar y manejar las fuentes de riesgos potenciales del mismo.

3.3.5. Los modelos orientados al objeto

La tecnología de objetos permite acelerar el desarrollo de sistemas de manera iterativa e incremental, permitiendo la generalización de los componentes para que sean reutilizables. Piattini (1996) presenta algunos de los modelos propuestos desde esta perspectiva.

Los modelos a tener en cuenta son:

a. **El modelo de agrupamiento o de clúster:** según Meyer (1990), los modelos usuales de ciclo de vida se basan en una cultura del proyecto, mientras que los desarrollos orientados al objeto están basados en el producto, entendido como elementos software reutilizables, cuyo beneficio económico aparece a largo plazo.En el modelo de *agrupamiento* se tiene en cuenta esta nueva fase de generalización que aparece combinada con la fase de validación. El concepto clave de este modelo es el de agrupamiento, que es un conjunto de clases relacionadas con un objetivo común. Se crean así diferentes sub–ciclos de vida que se pueden solapar en el tiempo y cada agrupamiento depende de los desarrollados con anterioridad, ya que se utiliza un enfoque ascendente: se empieza por las clases más básicas que pueden estar incluidas en bibliotecas.
b. **El modelo fuente:** Fue desarrollado por Henderson-Sellers y Edwards (1990), y representa gráficamente el alto grado de iteración y solapamiento que hace posible la tecnología de objetos. En la base está el análisis de requisitos, a partir del cual va creciendo el ciclo de vida, cayendo sólo para el mantenimiento necesario a la piscina, que sería el repositorio de clases.
c. **El modelo de remolino:** Rumbaugh (1992), analiza las cuestiones que afectan al ciclo de vida en el desarrollo orientado al objeto. Rumbaugh considera que el modelo en cascada supone una sola dimensión de iteración, consistente en la fase del proceso.

Dimensiones	Descripción
Amplitud	Tamaño del desarrollo
Profundidad	Nivel de abstracción o detalle
Madurez	Grado de completitud, corrección y elegancia
Alternativas	Diferentes soluciones de un problema
Alcance	En cuanto a cambio en los requisitos

Tabla 3.2: Las diferentes dimensiones según Rumbaugh (1990).

Debido a la identificación de otras dimensiones como amplitud, profundidad, madurez, alternativas y alcance, este proceso sería un desarrollo multicíclico, fractal más que lineal, en forma de remolino (ver Tabla 3.2).

d. Modelo pinball[13]: es un modelo propuesto por Amler (1994), quien señala que el pinball es el que refleja realmente la forma en la que se desarrolla el software. En este modelo el proyecto completo o un subproyecto está representado por la pelota y el jugador es el equipo de desarrollo. En forma iterativa se procede a encontrar clases, atributos, métodos e interrelaciones (en la fase de análisis) y definir colaboraciones, herencia, agregación y subsistema (que se incluyen en el diseño). Un último paso es la programación, prueba e implementación, y como en el pinball, los pasos se pueden tomar en cualquier orden y de forma simultánea. Amler destaca que se puede jugar a lo seguro, con tecnologías y métodos probados, o al límite, con mayor riesgo, pero con probabilidades de conseguir buenos beneficios. Destaca que la habilidad y la experiencia son los factores más importantes.

Llorca et al. (1991) sostienen que estos modelos se caracterizan por el desarrollo orientado al objeto, ya que:

- Eliminan los límites entre fases, tornándose cada vez más difusos debido a la naturaleza interactiva del desarrollo orientado al objeto.
- Permiten una nueva forma de concebir los lenguajes de programación y su uso e incorporan bibliotecas de clases y otros componentes reutilizables.
- La forma de trabajo es muy dinámica, debido al alto grado de iteración y solapamiento.

[13] En relación al juego.

Los expertos en tecnologías de objetos, proponen un desarrollo interactivo e incremental, existiendo un ciclo evolutivo del sistema en el sentido análisis-diseño-instrumentación–análisis, que se lleva a cabo en forma iterativa. Algunas metodologías hablan de diseños o metodologías recursivos pero como incrementales. (Piattini,1996)
Goldberg (1993), dice que *"la idea de la integración incremental es la diferencia clave de cómo debe ser gestionado un proyecto que utiliza tecnología orientada al objeto.*" Así las actividades de validación, verificación y aseguramiento de la calidad se pueden realizar para cada iteración de cada fase de cada incremento en el desarrollo del sistema, o sea en forma continuada.
Existen otros modelos de ciclo de vida, que no se han detallado en esta selección ya que aunque presenten ciertas potencialidades, no están muy extendidos. (J. Juzgado, 1996).
En el estándar IEEE 1074-1991 (IEEE, 1991) se detallan las fases del proceso base de construcción de software. Este estándar determina el *"conjunto de actividades esenciales que deben ser incorporadas dentro de un modelo de ciclo de vida del software y la documentación involucrada"*, pero estas actividades no están ordenadas en el tiempo.

3.4. La necesidad de una metodología de desarrollo

Para desarrollar un proyecto de software es necesario establecer un enfoque disciplinado y sistemático. Las metodologías de desarrollo influyen directamente en el proceso de construcción y se elaboran a partir del marco definido por uno o más ciclos de vida. (Piattini, 1996)
Según Piattini (1996), no hay un consenso entre los autores sobre el concepto de metodología, y por lo tanto no existe una definición universalmente aceptada. Sí hay un acuerdo en considerar a la metodología como *"un conjunto de pasos y procedimientos que deben seguirse para el desarrollo del software"*.
Maddison (1983) define metodología como un conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas de información. Por lo tanto, una metodología es un conjunto de componentes que especifican:
– Cómo se debe dividir un proyecto en etapas.
– Qué tareas se llevan a cabo en cada etapa.
– Qué salidas se producen y cuándo se deben producir.
– Qué restricciones se aplican.
– Qué herramientas se van a utilizar.
– Cómo se gestiona y controla un proyecto.

Generalizando, Piattini llega a la definición de metodología de desarrollo como *"un conjunto de procedimientos, técnicas, herramientas, y un soporte documental que ayuda a los desarrolladores a realizar nuevo software"*. Normalmente consistirá en fases o etapas descompuestas en subfases, módulos, etapas, pasos, etc. Esta descomposición ayuda a los desarrolladores en la elección de las técnicas a utilizar en cada estado del proyecto, facilitando la planificación, gestión, control y evaluación de los proyectos.
Sintetizando lo anterior, el autor dice que: *"una metodología representa el camino para desarrollar software de una manera sistemática"*.
Las metodologías persiguen tres necesidades principales:
– Mejores aplicaciones, tendientes a una mejor calidad, aunque a veces no es suficiente.
– Un proceso de desarrollo controlado, que asegure uso de recursos apropiados y costo adecuado.
– Un proceso estándar en la organización, que no sienta los cambios del personal.

Las metodologías a veces tienen diferentes objetivos, pero los más representativos pueden ser:
– Brindar un método sistemático, de modo de controlar el progreso del desarrollo.
– Especificar los requerimientos de un software en forma apropiada.
– Construir productos bien documentados y de fácil mantenimiento.
– Ayudar a identificar las necesidades de cambio lo más pronto posible.
– Proporcionar un sistema ágil que satisfaga a todas las personas involucradas.

Los procesos se descomponen hasta el nivel de tareas o actividades elementales, donde cada tarea está identificada por un procedimiento que define la forma de llevarla a cabo. Para aplicar un procedimiento se pueden usar una o más técnicas. Estas pueden ser gráficas con apoyos textuales, formales y determinan el formato de los productos resultantes en la tarea.
Para llevar a cabo las tareas se pueden usar herramientas software que automatizan la aplicación en determinado grado.

3.4.1. Evolución de las metodologías de desarrollo

Los primeros desarrollos no tuvieron una metodología definida, fueron totalmente artesanales y se los llamó desarrollos convencionales. Se caracterizaron por el aspecto monolítico de los programas y por una falta de control de lo que sucede en un proyecto. Estas limitaciones condujeron a una serie de problemas y por lo tanto a una búsqueda más sistemática en el desarrollo.

Como una posibilidad de *"nuevo orden"*, surge el desarrollo estructurado, sobre la base de la programación estructurada, los métodos de análisis y el diseño estructurado. Esta nueva etapa es la piedra fundamental para la construcción de programas con métodos ingenieriles.
La programación estructurada, aparece en los sesenta en el ámbito científico y en los setenta pasa al ámbito empresarial. Tiene como punto de partida el establecimiento y uso de normas para la aplicación de estructuras de datos y control.
En los setenta, el enfoque estructurado, se extiende de la *fase de análisis* a la *fase de diseño* y las técnicas estructuradas se dirigen tanto a los aspectos técnicos como los relacionados con la gestión en la construcción de software.
Myers (1975), Yourdon y Constantine (1975) y Page-Jones (1980), en sus publicaciones, definen al módulo del programa como el componente básico de la construcción software, pasando luego a la normalización de la estructura los módulos de programa y al refinamiento posterior. Es este período comienzan a aplicarse medidas de calidad de los programas.
Según Gane y Sarsons (1977), y DeMarco (1979), consideraron que uno de los problemas de los programas desarrollados monolíticamente era que se necesitaba una total comprensión total de las especificaciones, por parte de los analistas, que en muchos casos eran ambiguas, y en otros eran obsoletas al llegar al final del proyecto.
La base de la programación y el diseño estructurados, es un análisis del problema usando el diseño *top-down o descendente*, con énfasis en las especificaciones funcionales. Se compone de diagramas, con textos de referencia de los mismos, y con independencia para que se puedan leer en forma parcial, con una redundancia mínima, de modo que los cambios no lo afecten en forma notable.
También, se puede destacar, que ha habido una evolución en cuanto al modelado de sistemas en tiempo real, modelado de datos y estudio de eventos. (Piattini, 1996).
El *paradigma orientado objetos*, que aparece más tarde, trata a los procesos y los datos en forma conjunta, modularizando la información y el procesamiento.
Aparece el Smalltalk, con énfasis en la abstracción de datos, considerando a los problemas como un conjunto de objetos de datos a los que se les adicionaba un conjunto de operaciones, pero se fundamenta en la abstracción, la modularidad y el ocultamiento de la información, derivados del diseño estructurado.
En los ochenta, aparecen el C++ y el object C y más tarde el Lenguaje ADA, del Departamento de Defensa (DoD) de los Estados Unidos para la definición y mejora de tecnologías basada en este lenguaje.
En general para los desarrollos de las metodologías orientadas al objeto se tomaron de los conceptos de técnicas estructuradas.

3.4.2. Características y clasificación de las metodologías

Se pueden enumerar una serie de características que debe tener la metodología y que influirán en el entorno de desarrollo:
- Reglas predefinidas.
- Determinación de los pasos del ciclo de vida.
- Verificaciones en cada etapa.
- Planificación y control.
- Comunicación efectiva entre desarrolladores y usuarios.
- Flexibilidad: aplicación en un amplio espectro de casos.
- De fácil comprensión.
- Soporte de herramientas automatizadas.
- Que permita definir mediciones que indiquen mejoras.
- Que permita modificaciones.
- Que soporte reusabilidad del software.

Las metodologías se pueden clasificar considerando tres dimensiones de acuerdo a la Tabla 3.3.

3.4.2.1. Metodologías estructuradas

Estas metodologías definen los modelos del sistema que representan los procesos, los flujos y la estructura de datos de un modo descendente, pasando de la una visión general del problema a un nivel de abstracción más sencillo, pudiendo centrarse en las funciones o procesos del sistema, en la estructura de datos o en ambas, dando lugar a los tipos de metodologías indicados en la Tabla 3.3.
Las metodologías orientadas a procesos como las de Gane y Sarsons (1979), DeMarco (1979) y Yourdon (1989), tienen como base la utilización de un método descendente para la descomposición funcional del problema y se apoyan en técnicas gráficas de especificación estructurada.

Enfoque	Tipo de sistema	Formalidad
Estructuradas	Gestión	No formal

– Orientadas a procesos – Orientadas a datos – Jerárquicos – No jerárquicos – Mixtas		
Orientadas a objetos	Tiempo real	Formal

Tabla 3.3: Clasificación de las metodologías (Piattini, 1996)

Estos modelos gráficos, jerárquicos, descendentes y particionados son los diagramas de flujo de datos (DFD), los diccionarios de datos (DD) donde se definen los datos y los detalles de especificaciones de los procesos.
En la bibliografía, se pueden consultar los detalles de las metodologías de DeMarco (1979) y Gane y Sarsons (1977), las cuales se fueron refinando a través del tiempo y se expandieron a las fases de diseño e implementación. En la Tabla 3.4 se presenta las diferencias entre las tres metodologías citadas:

Fases del análisis estructurado		
Método de Gane y Sarsons	**Método de DeMarco**	**Método de Jourdon**
– Construir un modelo lógico actual – Construir un modelo lógico del nuevo sistema – Seleccionar un modelo lógico – Crear un nuevo modelo físico del sistema – Empaquetar la especificación	– Construir un modelo físico actual – Construir un modelo lógico actual – Crear un conjunto de modelos físicos alternativos – Examinar los costos y tiempos de cada opción – Empaquetar la especificación	– Realizar los diagramas d flujo del sistema – Realizar el diagrama de estructuras evaluar el diseño, midiendo la calidad cohesión y el acoplamiento. – Preparar el diseño para la implantación

Tabla 3.4: Diferencias entre las metodologías de Gane y Sarsons, DeMarco y Yourdon (Piattini, 1996)

La metodología de orientación a objetos, cambia el modo de ver al sistema, como un modelado de objetos que interactúan entre sí y no desde el punto de vista de la funcionalidad y la descomposición en tareas y módulos, pasando de las funciones de los programas y datos almacenados a un enfoque integrador y unificado.

Las metodologías orientadas al objeto, se pueden clasificar en: puras, que cambian radicalmente la perspectiva estructurada como sostiene Booch (1991) y evolutivas como la de Rumbaugh (1991) y Martin y Odell (1997), que toman al diseño estructurado como base para el desarrollo orientado al objeto.
El proceso y la notación de diseño de Booch (1991) y los escenarios de Jacobson (J. Juzgado, 1996), están siendo utilizados por otras metodologías más recientes y sistemáticas, conjuntamente con las métricas y los modelos de mejora del software como el CMM[14] (Modelo de Calidad y Madurez) o SPICE de ISO (Konrad y Paulk, 1995).

3.5. El ciclo de vida y los procesos

Todo proyecto tiene asociado, por más pequeño que éste sea, pasos que se deben seguir tales como: planificación, estimación de recursos, seguimiento y control, y evaluación del proyecto. La selección de un modelo de ciclo de vida está asociada a un orden en la realización de las actividades a desarrollar.
La red de actividades, es la que permitirá establecer a partir de la matriz de precedencia el camino crítico, como la secuencia de tareas más larga de principio al fin.
El diagrama de Gantt, o los diagramas calendario permitirán establecer el estado del proyecto en un determinado momento a partir de su inicio, en cuanto a recursos se refiere.
Para estimar el tamaño del producto o del programa a desarrollar, definido como la cantidad de código fuente, especificaciones, casos de prueba, documentación del usuario y otros productos, que han de ser desarrollados, se debe recurrir a datos estadísticos propios o no. La estimación consiste en la predicción del personal, el esfuerzo y el costo asociado para llevar a cabo todas las actividades del mismo.

3.5.1. La planificación de la gestión proyecto

Se la puede describir en términos de, las actividades a realizar, los documentos de salida y de las técnicas a utilizar como se observa en la Tabla 3.5.

[14] CMM: Capability Madurity Model

3.5.2. La identificación de la necesidad

La identificación de una necesidad, enunciada en términos concretos, es el punto de partida para la puesta en marcha de un proyecto y la evaluación de las posibles soluciones darán la viabilidad del mismo.

Planificación de la gestión del proyecto	
Actividades a realizar	Confeccionar el mapa de actividades para el modelo elegido del ciclo de vida, asignar de los recursos, definir el proyecto, planificar la gestión.
Documentos de salida	Plan de gestión, plan de retiro.
Técnicas a utilizar	CPM, PERT, diagrama de Gantt, estadísticas, simulación (Montecarlo), puntos funcionales, Modelos de estimación (COCOMO), Técnicas de descomposición para estimación.

Tabla 3.5: Planificación de la gestión del proyecto.

Por lo tanto, es deseable, confeccionar un informe de necesidades basados en los ítems de la Tabla 3.6:

Identificación de la necesidad	
Actividades a realizar	Identificar necesidades, formular posibles soluciones y estudiar su viabilidad.
Documentos de salida	Informe de necesidades. Alternativas de solución. Soluciones factibles.
Técnicas a usar	De adquisición de conocimientos, análisis costo-beneficio, modelización, diagramas de flujos de datos, prototipado.

Tabla 3.6: Identificación de la necesidad

3.5.3. El proceso de especificación de los requisitos

Consiste en establecer de un modo conciso, claro y preciso el conjunto de requisitos que deben ser satisfechos por el software a desarrollar. El objetivo es determinar en forma total y consistente los requisitos de software. El análisis se realiza sobre la salida resultante, la descomposición de los datos, el procesamiento de los mismos, las bases de datos y las interfaces de usuario. (J. Juzgado, 1996). (ver Tabla 3.7).
Se debe considerar que un requisito es una condición o característica que debe tener el programa para satisfacer un documento formal. Estos requisitos pueden ser funcionales, de rendimiento o de interfaces. Los primeros especifican la función que el programa debe realizar, los segundos especifican una característica numérica y los últimos determinan las características de las interfaces, usuario-software, software-hardware y software-software. (Juzgado, 1996).

Especificación de requisitos	
Actividades a realizar	Definir y desarrollar los requisitos del software y de las interfaces.
Documentos de salida	Especificación de los requisitos del software, requisitos de interfaces de usuario, de interfaces con otro software y con hardware. Requisitos de interfaces con el medio.
Técnicas a usar	Técnicas orientadas a los procesos: Análisis estructurado: diagramas de flujo de datos (DFD), diccionario de datos (DD), especificación de procesos. Diagramas de actividades. Técnicas orientadas a los datos: Diagramas entidad relación y diagramas de datos. Técnicas orientadas a los objetos. Diagramas de clases/objetos Jerarquía de clases/objetos Técnicas formales de especificación: Técnicas relacionales: ecuaciones implícitas, relaciones recurrentes, axiomas algebraicos. Técnicas orientadas al estado: tablas de decisión, de eventos, de transición, mecanismos de estado finitos, etc. Técnicas de prototipación

Tabla 3.7: La especificación de los requisitos

3.5.4. El proceso de diseño

El proceso de diseño es la piedra angular para la obtención de un producto coherente que satisfaga los requisitos de software. El diseño desde el punto de vista técnico comprende cuatro tipos de actividades: diseño de datos, arquitectónico, procedimental y diseño de interfaces y desde el punto de vista del proyecto evoluciona desde un diseño preliminar al diseño detallado.
El diseño de datos, modela las estructuras de datos necesarias para el desarrollo, el arquitectónico define las relaciones entre las estructuras del programa, considerando el desarrollo de módulos que se relacionan, mezcla la estructura de programas y de datos, y define las interfaces. El diseño procedimental transforma estructuras en descripción procedimental del software y por último el diseño de interface establece los mecanismos de interacción humano-computadora.
Desde el punto de vista del proyecto, el diseño preliminar se centra en las funciones y estructuras de los componentes que forman el sistema y el detallado se ocupa de refinar el anterior en algoritmos, para cada módulo.
Una actividad importante a realizar es el diseño conceptual, lógico y físico de la base de datos, si la hubiera. Este proceso de diseño, es la correcta traducción de los requisitos de software en un producto. (ver Tabla 3.8).
Se deben aplicar algunos principios conducentes a un software de calidad, tales como:
- Abstracción.
- Refinamiento sucesivo.
- Modularidad (consiste en la división en forma lógica de elementos en funciones y subfunciones).
- Estructura jerárquica en módulos con control entre componentes.
- Estructura de los datos.
- Procedimientos por capas funcionales.
- Ocultamiento de la información, etc., aplicación de métodos sistemáticos y una revisión constante.

Para evaluar la calidad de un diseño se deben tener en cuenta criterios tales como:
- División en módulos con funciones independientes.
- Organización jerárquica de los módulos.
- Representaciones de datos y procedimientos distintas.
- Minimización de la complejidad de las conexiones entre las interfaces.
- Reproducibilidad del método de diseño con los datos de los requisitos.

Los diseños modulares, reducen la problemática de los cambios, permitiendo desarrollos en paralelo. Para la definición de los módulos se usan conceptos tales como la abstracción y el ocultamiento de la información derivados de la independencia funcional de los mismos.
Los módulos tienen una función específica y definida o sea cohesión máxima y mínima interacción con los otros módulos o acoplamiento mínimo. La cohesión es una medida de la fortaleza funcional del módulo y la el acoplamiento es una medida de interdependencia de los módulos de un programa.
Existen herramientas de tipo CASE (Computer Aided Software Engineerig) que permiten automatizar el proceso traducción a código.

3.5.5. El proceso de implementación

Proceso de diseño	
Actividades a realizar	Realizar el diseño arquitectónico, analizar el flujo de información, diseñar la base de datos, diseñar las interfaces, desarrollar los algoritmos, realizar el diseño detallado.
Documentos de salida	Descripción del diseño del software de la arquitectura del software, del flujo de información, descripción de la base de datos, de las interfaces, de los algoritmos.
Técnicas a usar	Técnicas orientadas a los procesos: diseño estructurado, diálogo de las interfaces, diseño lógico, HIPO (Hierarchy Input Process Output). Técnicas orientadas a los datos. Modelo lógico y físico de datos. Jackson, etc. Técnicas orientada a los objetos: Modelo clase/objeto, diagrama de módulos. Técnicas de bajo nivel: Programación estructurada: diagramas de árbol Programación orientada a objetos: diagrama de procesos Técnicas de prototipación, Técnicas de refinamiento, Jackson, etc.

Tabla 3.8: El proceso de diseño

Proceso de implementación	
Actividades a realizar	Crear los datos de prueba, crear código fuente, generar el código fuente, crear la documentación, planificar y realizar la integración de módulos.
Documentos de salida	Datos de prueba, documentación del sistema y del usuario. Plan de integración.
Técnicas a usar	Lenguajes de programación. Jackson

Tabla 3.9: El proceso de implementación.

Este proceso (ver Tabla 3.9), produce código fuente, código de la base de datos y documentación, de base de acuerdo a los estándares utilizados. La salida de este proceso conduce a las pruebas de validación y verificación.

3.5.6. El proceso de instalación

Este proceso se centra en la verificación de la implementación adecuada del software y en la conformidad del cliente, previa prueba de aceptación (Tabla 3.10).

Proceso de instalación	
Actividades a realizar	Planificar la instalación, instalar el software, cargar la base de datos, realizar las prueba de aceptación.
Documentos de salida	Plan de instalación del software e informe de instalación

Tabla 3.10: El proceso de instalación

3.5.7. Los procesos de mantenimiento y retiro

El proceso de mantenimiento se centra en el cambio asociado a los errores detectados, fallas, mejoras solicitadas y cambios. Se lo considera como una vuelta a la aplicación del ciclo de vida pero con un software existente como iteraciones de desarrollo.
Los tipos de mantenimiento pueden ser: correctivos, ante defectos encontrados, adaptativos, o sea, cambios del software de acuerdo al cambio en el entorno y de mejoras, con agregado de funciones adicionales.

Procesos de:	Mantenimiento	Retiro
Actividades a realizar	Reaplicar el ciclo de vida.	Notificar al usuario, realizar las operaciones en paralelo y retirar el sistema.
Documentos de salida	Orden de mantenimiento Y recomendaciones de mantenimiento.	Plan de retiro

Tabla 3.11: Los procesos de mantenimiento y retiro.

El retiro es la baja de un sistema existente. Muchas veces, se lo reemplaza por una nueva versión, y otras por un nuevo sistema, siendo otras veces reemplazado por un nuevo sistema que opera temporalmente en paralelo durante un cierto tiempo de preparación del nuevo sistema. (ver Tabla 3.11).

3.5.8. El proceso de verificación y validación

Las tareas que abarca son las siguientes: pruebas de verificación, revisiones y auditoría e incluye las tareas de validación y pruebas de validación que se realizan durante el ciclo de vida del software para asegurar la satisfacción con los requisitos.
Para la verificación y validación del software, cuando ya exista código ejecutable, se pueden realizar las pruebas del mismo que consisten en ejecutar el software con determinados datos de entrada y producir resultados que luego serán comparados con los teóricos.
Un proceso asociado a las pruebas, es la depuración que consiste en tratar de deducir dónde están localizados los defectos en el software que hacen que este no funcione correctamente. (ver Tabla 3.12).

Procesos de verificación y de validación	
Actividades a realizar	Planificar y ejecutar las tareas de verificación y validación. Recoger y analizar los datos de las métricas, planificar las pruebas, desarrollar las especificaciones de las pruebas y ejecutarlas.
Documentos de salida	Plan de verificación y validación. Informes de evaluación. Plan de pruebas. Especificación de las pruebas. Resultados de las pruebas.

Técnicas a usar	Técnicas de prueba de caja blanca: Técnicas de prueba de caja negra: Revisiones formales. Auditorías.

Tabla 3.12: Los procesos de verificación y validación.

3.5.9. El proceso de la gestión de la configuración

El proceso llamado gestión de la configuración, involucra la gestión de los cambios durante el ciclo de vida que a partir de la configuración del sistema en un dado momento, tiene como objetivo un control de los cambios producidos y la coherencia del mismo.

Proceso de gestión de la configuración	
Actividades a realizar	Planificar la gestión de la configuración, identificar la configuración, realizar el control de la configuración y la información de estado de la misma.
Documentos de salida	Plan de gestión de la configuración, orden de cambio, cambio de estado, informe de estado.

Tabla 3.13: El proceso de gestión de la configuración.

Para ello es necesario la documentación del sistema en un momento determinado, el establecimiento de una configuración inicial y control de los cambios. (ver Tabla 3.13).

3.5.10. Los procesos de desarrollo de la documentación y de formación

Este proceso permite planificar, diseñar, implementar, editar, producir, distribuir y mantener los documentos para los desarrolladores y los usuarios.
Para una utilización efectiva del sistema se debe proporcionar al usuario las instrucciones y guías necesarias acerca del uso del software y de sus limitaciones. Es un punto fundamental la formación del usuario en el sistema. También es importante la formación de los desarrolladores y soporte técnico. (ver Tabla 3.14).

Proceso de desarrollo de:	**La documentación**	**Formación**
Actividades a realizar	Planificar e implementar la documentación, producir y distribuir la documentación.	Planificar el programa de formación. Desarrollar materiales de formación. Validar e implementar el programa.
Documentos de salida	Plan de documentación.	Plan de formación.

Tabla 3.14: Los procesos de documentación y de formación.

3.5.11. La selección de un ciclo de vida

La elección de un ciclo de vida adecuado para cada desarrollo está relacionada con las características del producto a obtener, a partir de los requisitos del desarrollo especificados.
De acuerdo al tipo de desarrollo, es conveniente realizar una adaptación del proceso descripto anteriormente en forma general y realizar, la matriz de actividades, partiendo del mapa de actividades−etapa del ciclo de vida elegido, la que se confecciona como una tabla de doble entrada, colocándose una cruz en la actividad a realizar en cada etapa. El mapa completo se denomina entonces, matriz de actividades para un determinado ciclo de vida elegido.
A partir de esta matriz se puede pasar a una estimación del tiempo y costo de cada actividad y para el proyecto global. También se pueden estimar los recursos necesarios por actividad. Algunas actividades se realizan por única vez y otras se repiten en cada etapa

3.6. El concepto de la calidad

Habría que comenzar con una revisión de algunas definiciones acerca de lo que significa calidad.
Deming (1982) propuso la idea de la calidad como conformidad con requisitos y confiabilidad en el funcionamiento. Juran (1995) dice brevemente*: "Quality is fitness for use"*, o sea es la adecuación del producto al uso, suponiendo un producto libre de deficiencias, cuyas características permiten la satisfacción del usuario.
Crosby (1979) pone énfasis en la prevención y dice *"con defectos cero"*. La norma ISO 8402 define la calidad como:
"Totalidad de características de un producto o servicio que le confieren su aptitud para satisfacer unas necesidades expresadas o implícitas".
Estas necesidades especificadas, bien pueden estar en un contrato o se deben definir explícitamente.
El logro de la calidad puede tener tres orígenes: *calidad realizada*, *calidad programada* y *calidad necesaria*. La primera es la que es capaz de obtener la persona que realiza el trabajo, la segunda es la que ha pretendido

obtener y la tercera la que exige el cliente y que le gustaría recibir. La gestión de la calidad pretenderá que estas coincidan.

3.6.1. La calidad en ingeniería de software

El software es un producto con características muy especiales, hay que tener en cuenta que es un producto que se desarrolla y se centra su diseño, con una existencia lógica de instrucciones sobre un soporte, siendo un producto que no se gasta con el uso como otros y repararlo no significa restaurarlo al estado original, sino corregir algún defecto de origen lo que significa que el producto entregado posee defectos, que podrán ser solucionados en la etapa de mantenimiento (Piattini, 1996).
El diccionario IEEE estándar de ingeniería del software (IEEE, 1990) dice que son software: *"los programas de ordenador, los procedimientos y, posiblemente la documentación asociada y los datos relativos a la operación del sistema informática"*, no limitándose al código.
El estándar IEEE 6.10 -1990 (IEEE, 1990) da la definición de calidad como **"***el grado con el que un sistema, componente o proceso cumple con los requisitos especificados y las necesidades o expectativas del cliente o usuario".*
Pressman (1993) la define como *"concordancia del software con los requisitos explícitamente establecidos, con los estándares de desarrollo expresamente fijados y con los requisitos implícitos, no establecidos formalmente que desea el usuario".*
La aplicación de estándares de desarrollo y de normas para el software permitirá lograr calidad técnica del mismo. La calidad del software se puede ver a nivel empresa como implantación de un sistema de calidad y a nivel de proyecto aplicando las técnicas de evaluación y control de la calidad del software a lo largo del ciclo de vida.
El interrogante que surge es: *¿Por dónde empezar el proyecto de mejora de la calidad de los programas en general?*. Como respuesta a ello, se analizarán brevemente en la sección siguiente los aportes de algunas de las instituciones están trabajando al respecto y que elaboraron algunos estándares como: IEEE, ISO y SEI.[15]
Los requisitos explícitos, ya sean funcionales, de seguridad, de rendimiento, de interface, son la culminación de la etapa de análisis y quedan establecidos en el documento de especificación de requisitos del software y es en la etapa de análisis donde muchos de los requisitos implícitos no expresados formalmente por el usuario quedarán declarados en el documento de especificación.

3.6.2. La calidad desde el aspecto organizacional. La familia ISO 9000

La familia ISO 9000, es un conjunto de normas en las que se apoya el sistema de calidad de una empresa.
La norma ISO 9000 define al sistema de calidad desde la perspectiva de la organización como: **"***La estructura de organización, de responsabilidades, de actividades, de recursos y de procedimientos que se establecen para llevar a cabo la gestión de calidad"* (ISO-9001: 1994). En la Tabla 3.15 se define la terminología básica empleada.
Desde esta posición, se debe fijar la estructura organizativa ligada al sistema de gestión de la calidad a través de líneas jerárquicas y de comunicación.
Entre las normas con la que las organizaciones cuentan para el logro del aseguramiento de la calidad de los productos software, están las de la familia ISO 9000. (ISO 9000, 1991). Si bien esta familia en un principio se la diseñó para aplicaciones industriales en general, existe una versión la 9000-3, específica para productos lógicos. (ISO 9000-3, 1991).
Bajo la denominación de ISO SPICE (Konrad, Paulk y Graydon, 1995) se desarrollaron para ISO e IEC[16] un conjunto de estándares en un intento de armonizar los diferentes esfuerzos en el mundo para conducir a la obtención de un proceso de software confiable.

Término empleado	Definición
Gestión de la calidad	Es un aspecto de la función general de la gestión que determina y aplica la política de la calidad (objetivos y directrices generales de calidad) y normalmente se aplica a nivel empresa. (Aenor).
Aseguramiento de la Calidad	"Es el conjunto de actividades planificadas y sistemáticas necesarias para aportar la confianza de que el producto (software) satisfará los requisitos dados de calidad" (Aenor, 1992). El estándar IEEE (1990) dice que se puede referir al "conjunto de actividades para evaluar el proceso mediante el cual se desarrolla el producto"
Control de la	Son las técnicas y actividades de tipo operativo destinadas a controlar un proceso y elimi-

[15] IEEE. Institute of Electrican and Electronics Engineering.
ISO: International Organization for standarization.
SEI: Software Engineering Institute.
[16] IEC: International Electrotechnical Comission.

calidad del software	nar las causas de defectos durante las etapas del ciclo de vida. (Aenor, 1992).
Verificación y Validación	La IEEE (1990) dice que es una actividad ligada al control de la calidad en el ámbito del software. La **verificación** trata de comprobar si el producto construido en una etapa del ciclo de vida satisface los requisitos establecidos en la etapa anterior, respondiendo a la pregunta: ¿está el producto construido correctamente? La **validación**: consiste en comprobar si el software construido satisface los requisitos del usuario, respondiendo a la pregunta. ¿El producto es el correcto?

Tabla 3.15: Terminología empleada.

Uno de los productos del proyecto es una guía para buenas prácticas de dirección e ingeniería de software, similar a CMM[17] del SEI y al Trillium de Northern Telecom.
La idea central es crear un modo para medir la capacidad mientras se permite una aproximación a la mejora como los niveles de madurez del CMM, siendo estas aproximaciones para medir la implementación e institucionalización de procesos específicos.
El estándar IEEE 1074-1991 define el conjunto de actividades requeridas para llevar a cabo el desarrollo y el mantenimiento de un producto software, cuya calidad sea confiable. La definición de los procesos y de las actividades para cada una de las etapas del ciclo de vida elegido permiten obtener un producto que se ajuste a los requerimientos. Desde esta perspectiva, cada actividad se la define a partir de una descripción de la misma y de las respectivas entradas y salidas.
Por otra parte, cuantificar la calidad significa tener que medirla. Para cuantificarla se requiere de mediciones indirectas de algunas manifestaciones de la misma, que se denominan métricas.
El estándar IEEE 1061-1992, establece el propósito de las métricas para calidad del software, provee un sistema de métricas y provee de una metodología para el establecimiento de los requerimientos de calidad. No prescribe métricas específicas, pero si ejemplos de aplicación del mismo.
Existen algunos modelos como el CMM (Capability Madurity Model) desarrollado por Victor Basili en el SEI (Software Engineering Institute) de la Universidad Carnegie Mellon (CMU), pensado para el Departamento de Defensa (DoD) de los Estados Unidos, el cual se extendió rápidamente al ámbito empresarial.
Este modelo considera el *grado de madurez del producto*, teniendo en cuenta cinco etapas bien diferenciadas. Estas etapas van desde un proceso inmaduro e improvisado, con falta de rigurosidad metodológica, hasta llegar a un proceso totalmente maduro, que evidencia rigurosidad y consistencia en la administración de los recursos y adecuación de los requerimientos con el proceso.
Quizás la clave consiste en la mejora evolutiva, a partir de establecimiento y claridad de los puntos de partida y de arribo en cada una de las etapas.
La evolución a través de los cinco niveles: inicial, repetible, definido, administrado y optimizado, permiten establecer los controles requeridos para gestionar los proyectos: planificar, administrar, controlar, supervisar (nivel 2), para permitir (en el nivel 3) integrar los aspectos organizacionales con el proyecto en si. Recién, en el nivel 4, se establece una administración cuantitativa del proceso y de la calidad del producto software y el último, define los aspectos a tener en cuenta para la prevención de los defectos, implementación de mejoras y cambios.
Es un modelo aplicable a organizaciones que desarrollan proyectos, y que permite luego comparar resultados, pero su aplicación no garantiza el éxito del proyecto, tornando a la empresa en una estructura más rígida.
Este modelo de mejora gradual del proceso de software y calidad del producto, no es el único y su aplicabilidad depende de las necesidades y características de la organización.
Si bien a través de estos estándares (ISO 9000-3 y CMM) se intenta obtener una mejora continua, se centran exclusivamente en los procesos y no en los procesos de pruebas, quedando fuera de su alcance el plan de pruebas y la selección de las mismas, aunque en la ISO se menciona la documentación de las pruebas y no el proceso de prueba específicamente, habría que ver también el estándar 820 de la IEEE (IEEE, 1991).
Bender (1996) sugiere agregar al CMM, una *"área clave de proceso"* (KPA, Key Process Area), para la prueba y evaluación del software. Burnstein, et al. (1996) desarrollan un modelo que denominan TMM (Testing Madurity Model) a modo de complemento al CMM, como un indicador de la madurez del proceso de prueba, a fin de implementar las mejora pertinentes. Se basa en niveles de madurez y metas para cada uno, con cuestionarios de valoración de cumplimiento en cada nivel, como también entrenamiento del personal de prueba y valoración mediante de cuestionarios y entrevistas.

3.6.3. El concepto de calidad del software

[17] CMM: Capability Madurity Model.

Boehm (1978) y McCall (1977) descomponen el concepto de calidad en propiedades más sencillas de medir y de evaluar. El modelo de McCall se basa en la descomposición del concepto de calidad en tres usos importantes de un producto de software desde el punto de vista del usuario:

- Características de operación.
- Capacidad para soportar cambios (ser modificado).
- Adaptabilidad a nuevos entornos.

Cada capacidad se descompone en una serie de factores a saber: facilidad de uso, integridad, fiabilidad, corrección, flexibilidad, facilidad de prueba, facilidad de mantenimiento, transportabilidad, reusabilidad y interoperabilidad.

Cada factor se descompone en criterios o propiedades internas del software que determinan su calidad: facilidad de operación, facilidad de comunicación, facilidad de formación o aprendizaje, control de accesos, facilidad de auditoría, eficiencia de ejecución, eficiencia de almacenamiento, exactitud o precisión, consistencia, tolerancia a fallas, modularidad, simplicidad, completitud, facilidad de traza, autodescripción, capacidad de expansión, generalidad, instrumentación independencia entre sistema y software, independencia del hardware, compatibilidad de comunicaciones y compatibilidad de datos.

Mc Call define el factor de calidad como FC, según:

$FC = c_1 \, x \, m_1 + c_2 \, x \, m_2 + ... + c_n \, x \, m_n$

Donde los c_i son los coeficientes de regresión y los m_i son las métricas que afectan al factor de la calidad.

Estos criterios pueden ser evaluados mediante un conjunto de métricas, las que se pueden calcular observando directamente el software. Para cada criterio McCall propuso una serie de métricas, aunque, muchas de ellas sólo pueden ser medidas en forma subjetiva. Las métricas pueden estar en forma de listas de comprobaciones, para obtener el grado de los atributos específicos del software. McCall propuso un esquema de graduación mediante una escala que va de cero (bajo) a 10 (alto) y utiliza como métricas los criterios o propiedades internas del software citados anteriormente.

La norma IEEE 1061 propone un modelo de medición muy parecido al de McCall y la norma ISO 9126 (ISO, 1991) establece un modelo propio, similar al de McCall.

En la década del ochenta, se comenzó a usar modelos particulares de evaluación para cada empresa o proyecto, implantándose el concepto de calidad relativa. Gilb (1988) propone la creación de una especificación de requisitos de calidad a redactar conjuntamente el usuario y los analistas, determinando así la lista de características que definan la calidad de cada aplicación. Este enfoque se ha asociado a la filosofía QFD (Quality Function Deployment), o el despliegue de la función de la calidad que se aplica al ámbito de la gestión de la calidad industrial y en el que se han basado modelos posteriores. Otros modelos son los de Basili y Rombach (1988) proponen el paradigma GQM, (objetivo–pregunta–métrica o goal-question-metric) para evaluar la calidad de cada proyecto.

Grady y Caswell (1987) presentan un enfoque de medición inspirado en el control estadístico de procesos aplicado a la industria convencional de fabricación, considerando a la calidad como la ausencia de defectos, que en este caso pueden ser fallas, defectos o errores.

3.6.4. Métricas de calidad del software

Para la evaluación de la calidad es más habitual referirse a medidas del producto que en medidas del proceso. Una métrica (Fenton, 1997) es *"una asignación de un valor a un atributo de una entidad de software, ya sea un producto o un proceso"*. En todos los casos las métricas representan medidas indirectas de la calidad, ya que sólo se miden las manifestaciones de ella.

Se pueden tener métricas basadas en el texto del código y métricas basadas en la estructura de control del código.

Métricas basadas en el texto del código: En general, se pueden tomar la cantidad de líneas de código, como un indicador de tamaño, el número de líneas de comentarios como un indicador de la documentación interna, el número de instrucciones, el porcentaje de líneas de código o densidad de documentación, etc. Halstead (1975), propone sus métricas dentro de este tipo, denominadas: *"Ciencia del software"*.

Métricas basadas en la estructura de control del código: Pueden tomarse dos tipos de medidas: unas relacionadas con el control intramodular, basada en el grafo de control y otras relacionadas con la arquitectura en módulos, basada en el grafo de llamadas o en el diagrama de estructuras. Las métricas de McCabe (McCabe, 1976) son del primer tipo y constituyen un indicador del número de caminos independientes linealmente basándose en conceptos matemáticos que existen en un grafo.

Yin y Winchester (1978) definen el *fan-in* y el *fan-out* de cada módulo. Henry y Kafura (1984) definen la complejidad del módulo. Piattini (1996) sostiene que los resultados parecen indicar que mejores valores de métricas implican un menor mantenimiento posterior debido a un menor número de defectos.

3.6.5. Las diferentes aproximaciones

Siguiendo a Fenton (1997), la medición es un proceso por el cual, se debe asignar números o símbolos a atributos y entidades en el mundo real, de tal modo de describirlas de acuerdo a reglas definidas claramente.
Recordando a DeMarco (1982) *"no se puede controlar lo que no se puede medir"*, esto daría una idea acaba de cuál es la necesidad primordial de efectuar mediciones sobre un proceso, en un proyecto. Cada acción de medición de algo, debe estar motivada por un objetivo particular o necesidad definida claramente. Si este razonamiento, se suma al principio de Gilb (1988), citado por Fenton: *"los proyectos que no tienen objetivos claros, no arriban a metas claras",* queda explícita la necesidad de una metodología y la adecuación de lo que se deba medir al tipo de software a desarrollar y su uso particular.
¿Para qué medir? Bien, para ayudarnos a entender qué está sucediendo durante un desarrollo, analizando los desvíos respecto de las líneas de base, o para controlar lo que está sucediendo en el proyecto o programa respecto de una línea de base, o, por último, para mejorar los procesos.
Si bien en los proyectos software empresarial, se miden esfuerzo, costo, productividad, (COCOMO y COCOMO II),[18] capacidad y madurez, calidad (Mc Call) confiabilidad, entre otras métricas, no todos los modelos desarrollados para este tipo de software, serían adecuados para los proyectos de programas educativos.
Ya se describió sintéticamente el modelo de tres niveles de Mc Call (1977), llamado comúnmente FCM (Factor Criteria Metric). Cada factor de calidad está compuesto por un criterio, que es lo que realmente se mide, ya que son más fáciles d entender. En un tercer nivel se describe el grado de pertinencia de las relaciones entre factores, aclarando que hay que .... *"dividir para conquistar"* .
De acuerdo a Boehm (1978) y a Mc Call (1977) ambos definen atributos externos: confiablidad, usabilidad (utilidad) y mantenibilidad y eficiencia y testeabilidad como internos. Dentro de los internos, también estarían la estructurabilidad y la modularidad y estos se reflejan sobre los externos. A cada uno se le asignan criterios y métricas, posteriormente.
La duda que surge es: *¿Utilizar un modelo predefinido o definir un modelo propio?*. Ya que los anteriores son típicos para modelos fijos, esta podría ser una salida al problema, como sostiene Gilb (1977) o el COQUAMO (Constructive Quality Model) de Kitchenham y Walter (1989).
La ISO 9126 (1991) define calidad en términos de atributos de interés para el usuario del producto de software, algunos internos y otros externos al mismo, define seis factores y atributos. Se lo puede considerar la piedra angular en la definición de un proceso para evaluación de la calidad del software.
Otros investigadores, miden portabilidad, integridad, densidad de defectos, no habiendo consenso de qué es el defecto que se mide, por lo cual habría que definirlo en cada caso.
Llegando al concepto de *"utilidad (usability) del software"* como la extensión para la cual, el producto es conveniente y práctico, de un modo intuitivo se la podría considerar como amigabilidad teniendo en cuenta su practicidad o valor práctico.[19]
Fenton dice que se podría definir en términos de *"la probabilidad de que un operador de un sistema no experimente un problema en la interface de usuario durante un período dado de operación en condiciones normales de operación o cómo el usuario interactúa con la interface"*.
Por ahí, sería conveniente remitirse a características internas más simples que conduzcan a una buena utilidad, como:
- Buen uso de menú y gráficos.
- Mensajes de error e informativos.
- Funciones de ayuda.
- Manuales bien estructurados.

Este podría ser el *concepto clave,* buscado, especialmente para los programas educativos y en él se hará hincapié.

3.6.6. La verificación y la validación del software

Otros aspectos a tener en cuenta, son las revisiones y las pruebas del software como parte del ciclo de vida, que se utilizan para detectar fallas en los requisitos, en el diseño y en la implantación y son procesos orientados a la detección de defectos en el producto, dándole mucha importancia a las revisiones
La verificación y la validación del software (VyV) incluyen un conjunto de procedimientos, actividades, técnicas y herramientas que se utilizan paralelamente al desarrollo del software, para asegurar que el producto resuelve correctamente el problema para el que fuera diseñado. El objetivo es prevenir las fallas desde los requerimientos hasta su implementación.
La VyV actúa sobre los productos intermedios intentando detectar y corregir cuanto antes sus defectos y desviaciones del objetivo primigenio, si las hubiera.

[18] COCOMO: COst COnstructive MOdel
[19] Simon & Schuster´s. International Dictionary. McMillan, USA.

Respecto de las pruebas a realizar en el software, ellas pueden ser dinámicas o estáticas, de acuerdo a si se realizan o no sobre el código. Entre las pruebas dinámicas, están las llamadas de caja blanca y las de caja negra, *"testeando"* los procedimientos estructurales o las salidas. Y entre las estáticas están las revisiones, las verificaciones, a fin de detectar problemas durante el proceso de desarrollo.

3.6.7. Las revisiones del software

El estándar IEEE (1989), indica que para conseguir los objetivos de aseguramiento de la calidad se disponen de los siguientes métodos como se puede ver en la Tabla 3.18.
Las revisiones y auditorías se pueden utilizar para revisar procedimientos e gestión y productos. Un tipo de revisiones a considerar son las *revisiones técnicas*, cuyo objetivo es evaluar un producto intermedio de desarrollo para comprobar que el producto se ajuste a las especificaciones y que se está llevando a cabo de acuerdo a los planes y estándares.

Objetivos de calidad	Método principal
Evaluación	Revisión de gestión, revisión técnica
Verificación	Inspección, walkthrough
Validación	Pruebas
Confirmación de cumplimiento	Auditoría

Tabla 3.18: Métodos principales para conseguir objetivos del aseguramiento de la calidad.

Las *inspecciones* tienen por objetivo detectar y registrar los defectos del producto intermedio, en forma rigurosa y formal, buscando que el producto satisfaga las especificaciones y estándares. Están pensadas como una medida de ayuda al gestor.
Los *walkthroughs* se realizan para evaluar un producto en busca de defectos u omisiones, como una medida de ayuda al desarrollador, considerando las posibles alternativas de solución cuyo objetivo es comprender bien el objeto.
Los métodos citados anteriormente constituyen el análisis estático ya que no se necesita ejecutar el software. Una mención especial dentro de los análisis dinámicos del software corresponde a las pueblas modulares y de integración, según se detalla en el estándar IEEE-1012 (1986) y las pruebas pueden ser funcionales o estructurales. En la figura 3.1 se puede ver, la relación entre de algunos procesos de aseguramiento de la calidad sobre productos y proyectos:
Las auditorías se realizan para determinar en forma objetiva que los productos desarrollados se ajustan a los estándares. En el estándar IEEE (1984), se recomienda realizar una cantidad determinada de revisiones y auditorías sobre todo en software críticos.

4. La evaluación de software educativo.

4.1. Resumen

En esta sección se definen los tipos de evaluación a las que se deben someter los programas educativos (sección 4.2). Se describen las evaluaciones interna (sección 4.3) y externa (sección 4.4), y se analizan los instrumentos de evaluación utilizados (sección 4.5), presentándose una síntesis de los criterios de selección y de evaluación de los programas didácticos, a modo diacrónico durante las últimas décadas y sincrónico en el fin de siglo, considerando las pautas fijadas en el ámbito didáctico para lograr una efectividad del producto (sección 4.6).
Para este relevamiento y análisis documental, se han tenido en cuenta las líneas didácticas de investigadores de diferentes escuelas como la norteamericana y la española, y también algunos otros trabajos relevantes en evaluación de programas educativos, aunque consideren casi exclusivamente el aspecto técnico y muy poco el pedagógico.

4.2. La evaluación

La evaluación de los programas educativos es un proceso que consiste en la determinación del grado de adecuación de dichos programas al contexto educativo. Cuando el programa llega al docente, es de suponer que ha sido analizado y evaluado tanto en sus aspectos pedagógicos y didácticos, como en los técnicos que hacen a la calidad del producto desarrollado según ciertas pautas de garantía de calidad.
Básicamente, se realizan las evaluaciones interna y externa del software, a fin de detectar los problemas que generarán cambios en el producto, lo antes posible, a fin de reducir costos y esfuerzos posteriores. Estas evaluaciones consideran las eventuales modificaciones sugeridas por el equipo de desarrollo y por los usuarios finales, teniéndose en cuenta a docentes y alumnos en el contexto de aprendizaje.

Cuando un producto del tipo comercial educativo, llega al docente, significa que ha superado las etapas de evaluaciones interna y externa y además se espera obtener el grado de eficacia y de eficiencia del producto para lo que se deberá realizar una evaluación en el contexto de uso particular.
Es preciso definir ciertos "*criterios*" para seleccionar un programa que esté *"de acuerdo a las necesidades del docente".* Además se debe considerar el uso de los vocablos "*evaluación*" y "*valoración*"[20], que en muchos de los trabajos consultados se usan indistintamente para determinar si un programa dado cumple con los objetivos tanto técnicos como pedagógicos y didácticos para lo que fue pensado.

4.3. La evaluación interna

De acuerdo a lo expuesto anteriormente, se deberá llevar a cabo una evaluación interna del software, (Marquès, 1995) que estará a cargo de los miembros del equipo de desarrollo y otra evaluación, la externa en la que participan profesores y alumnos destinatarios del programa, cuando se haya terminado el mismo, o esté casi listo.
La versión preliminar a evaluar poseerá todas las funcionalidades del programa, pero algunos aspectos como los mensajes, las imágenes y los gráficos serán provisorios e incompletos y las bases de datos muchas veces estarán cargadas parcialmente.
Luego de la evaluación realizada por el equipo de desarrollo, se confeccionará una lista con los cambios necesarios y las modificaciones, las mejoras estructurales y todos aquellos aspectos que se crean convenientes antes de utilizar el programa o de comercializarlo, buscando adecuación a las especificaciones de requerimientos y un aseguramiento de los aspectos funcionales y pedagógico–didácticos.
Algunos autores como Marquès (1995) consideran que se pueden contemplar tres aspectos fundamentales en la evaluación en general: aspectos técnicos, pedagógicos y funcionales.
Los primeros permitirán asegurar la calidad del producto desde el punto de vista técnico específicamente, pudiéndose realizar un análisis estructural de elementos tales como el diseño de pantallas y la interface de comunicación.
Los aspectos pedagógicos, son aquellos que se refieren al fin con el que el software será utilizado. Por ello hay que analizar elementos como: los objetivos educativos, los contenidos y los caminos pedagógicos, que se deben considerar en toda buena programación didáctica. Respecto de los funcionales, habría que considerar cuáles son las ventajas que le da al profesor como material didáctico, cómo facilita los aprendizajes de los alumnos y cuáles de las funciones del pensamiento favorece.
Bork (1986), denomina a esta evaluación interna como formativa, o sea la evaluación del proceso, como aquella realizada generalmente por los desarrolladores.
Para realizar las evaluaciones generalmente se utilizan listas de control o *checklists*, mediante planillas o plantillas de *checklists* y casillas de verificación, incluyendo no sólo preguntas cerradas, sino preguntas abiertas sobre diversos aspectos del programa. Estos resultados son los que necesita el equipo desarrollador para hacer todos los cambios necesarios y convenientes.
Luego de producidos los cambios, es cuando se agregan los efectos faltantes (como sonido, animaciones, imágenes y gráficos) y se carga de la base de datos final (si la hubiere), para proceder a emitir una versión de prueba externa (Marquès, 1995).
La documentación, además de ser un proceso que se realiza paralelamente durante todo el desarrollo del programa, será también evaluada externamente, junto con el programa. Este punto es muy importante, ya que la mayor parte de los programas carecen de ambas documentaciones: interna y externa, y en otros casos esta es escasa como para conocer su estructura interna.

4.4. La evaluación externa

La evaluación externa permite obtener las sugerencias de los alumnos potenciales, quienes serán en definitiva los usuarios del software y de los docentes que lo utilizarán como material didáctico.
Durante este tipo de pruebas, se encuentran a menudo errores imprevistos no detectados y se verifica el cumplimiento de los programas con los objetivos educativos que se han considerado en el diseño.
Alfred Bork (1986) la denomina evaluación sumativa y es la evaluación del producto final que generalmente la realizan equipos distintos a los desarrolladores. La información se recoge mediante checklists y preguntas cerradas y abiertas a contestar luego de interactuar con el programa, durante un tiempo predeterminado.

---

[20] *Valorar:* en un sentido amplio, según Squires y Mc Dougall (1994), tiene en cuenta la evaluación, la revisión y la selección. Tanto la revisión (resumen de las características para permitir la selección) y la selección (es la valoración de los profesores antes del uso en el aula) se las puede considerar como parte de la valoración misma en un sentido más restringido.
Evaluar: es determinar el grado de adecuación, ya sea durante el desarrollo para hacer las modificaciones (formativa) o posterior a él mediante experiencias de uso (sumativa).

En casi todas las investigaciones consideradas se denota la falta de herramientas de evaluación sencillas y de documentación de los programas educativos.
Como resultado de ambas evaluaciones, se obtiene la llamada primera versión del programa con su respectivo manual de usuario, que contiene todos los aspectos que se consideren indispensables para el uso docente, con detalles técnicos, y del entorno pedagógico y didáctico para el que se desarrolló el programa.
A la etapa de adquisición de un programa le sobreviene, una etapa fundamental: la de mantenimiento y de actualización, en la cual la documentación técnica tanto externa como interna juega un papel importantísimo, ya que es la base para cambios posteriores, y que en la mayoría de los casos no se desarrolla o es casi inexistente.
Tanto la etapa de actualización como la de mantenimiento, a la hora de hablar de programas educativos, no se tienen en cuenta al adquirir el producto y éste puede quedar fuera de mercado, en muy poco tiempo, debido a los avances tecnológicos, sin posibilidad de agregarle nuevas funcionalidades.

4.5. Los instrumentos de evaluación

En general, los instrumentos más usados, son los cuestionarios de valoración, donde las respuestas a estos cuestionarios son valoradas entre 0 y 5, por ejemplo, siendo el resultado el grado de conformidad del usuario con las afirmaciones propuestas.
Los instrumentos de evaluación, en forma de planillas se deben confeccionar con inclusión no sólo de preguntas del tipo cerradas, sino también de preguntas abiertas, y casillas de verificación, permitiendo al usuario final la descripción de aspectos problemáticos y particulares del programa que no hayan sido tenidos en cuenta durante la confección del instrumento.
Se deberá tener en cuenta al redactar los cuestionarios, la utilización de un vocabulario adecuado, sin ambigüedades y claro para los destinatarios previstos en cada caso en particular.
En la mayor parte de los cuestionarios relevados se consideran algunos aspectos claves o sobresalientes: como el logro de los objetivos, los aspectos técnicos, el desarrollo de contenidos, actividades y la documentación. Estos aspectos se categorizan en ítems, según cada propuesta.
Como cada propuesta de evaluación está relacionada a un software particular o a paquetes de programas que se relacionan, se deben analizar con cuidado los diferentes criterios para evaluación de estos medios didácticos, debidos a las particularidades del software educativo, considerándolas sólo como una "*guía*" que luego se deberá "*readaptar*" a cada contexto educativo particular. Es esta adaptación a cada contexto en particular la que nos permite afirmar que no hay un instrumento de evaluación único, sino que el mismo será función del contexto de aplicación.

4.6. Las propuestas de selección y evaluación de software educativo

En las últimas décadas se han elaborado muchas propuestas con listas de criterios para seleccionar y evaluar el software educativo, algunas forma individual y otras en el ámbito institucional. Si bien varían en cuanto a su contenido y estilo, todas ellas tienen un objetivo común: ayudar al docente a elegir y valorar un programa adecuado para sus necesidades.
Cabero (1993) sostiene que las propuestas para la evaluación de los programas informáticos han sido muy variadas. Sancho (1994) después de revisar distintas escalas propone una como representativa de las dimensiones de análisis, que se describe a continuación.
Pretendía recoger información acerca de: contenidos (puntos 1 y 25), aspecto técnico del programa (3, 6,17, y 22), motivación para el alumno (4), valoración didáctica general del programa (5 y 11), claridad del programa (7, 18, 23 y 2), duración del programa (9), facilidad de manejo (10 y 12), adecuación a los receptores (14, 15, 20 y 21) y objetivos (13,16 y 24) en unas 25 preguntas. Cada ítem debía ser valorado de 1 a 5 de acuerdo con la siguiente escala (5) muy adecuado (4) bastante adecuado (3) adecuado (2) poco adecuado (1) nada adecuado, pudiéndose utilizar la contestación NA: no aplicable. Como se puede ver en los números dentro de los paréntesis, no había un agrupamiento a priori en categorías, sino que las preguntas relacionadas a un mismo ítem estaban intercaladas con otras.
Marquès (1995), investigador de la Universidad Autónoma de Barcelona propone una ficha para catalogación y evaluación de programas didácticos. Considera rasgos generales del programa, objetivos que se persiguen, tipología, contenidos que se tratan, valoración técnica, valoración pedagógica, aspectos negativos, etc., concientizado de que al evaluar un programa, hay que considerar sus características y su adecuación al contexto en el que se quiere utilizar.
En su ficha separa la catalogación, cuyo objetivo es proporcionar una idea de las prestaciones que ofrece, de la evaluación que recoge la información del profesor acerca de diferentes aspectos del programa.
Los aspectos técnicos a considerar son: las pantallas, el algoritmo principal, el entorno de comunicación y las bases de datos; los aspectos pedagógicos: los objetivos educativos, los contenidos, las actividades interactivas, la integración curricular, la documentación del programa, los aspectos funcionales: utilidad del programa en cuanto a motivación y facilitación de aprendizajes.

Considera a la evaluación contextual de un programa como a la forma en que ha sido utilizado en clase un determinado programa independientemente de su calidad técnica y pedagógica. Esta evaluación tiene en cuenta el grado de logro de los objetivos educativos respecto de los planificados. Insiste en que la metodología utilizada por el profesor constituye el principal elemento determinante del éxito de la intervención didáctica, por lo tanto debe tenerse en cuenta la motivación previa que ha realizado el profesor antes de la sesión, la distribución de los alumnos en clase, la autonomía para interactuar con el programa. Aquí juega un rol importante las características de los alumnos, el grado de motivación, los estilos cognitivos, los intereses, el conocimiento previo y las capacidades.

Osuna, Bermejo y Berroso (1997), del Departamento de Ciencias de la Educación de la Universidad de Extremadura, proponen una escala de evaluación para software educativo. Sostienen la necesidad de una evaluación sistemática debida a la creciente cantidad de productos informáticos generados por la por la industria informática. La misma debería facilitar la toma de decisiones de los profesores y administradores educativos para su adquisición y uso. La escala de evaluación que articulan contiene: identificación del programa, valoración de elementos y valoración de relaciones contexto-entrada-proceso. Valoran los elementos en muy adecuados, adecuados, poco adecuados y nada adecuados.

Cabero Almenara (1992), de la Universidad de Sevilla, sostiene que uno de los grandes problemas para el uso de la informática en el terreno educativo radica en la existencia y calidad del software. La calidad del software educativo es un concepto complejo y muy difícil de precisar, ya que es el resultado de una serie de interacciones: el contenido, el docente, la tecnología que condicionarán los resultados que de él se obtengan. Un problema recurrente de los medios de enseñanza, consiste en determinar de qué manera pueden diseñarse para que la comunicación de los mensajes sea más eficaz y la interacción con el usuario sea lo más útil posible. En síntesis, para que facilite el aprendizaje y el recuerdo de la información por ellos transmitida y propicie entorno de aprendizaje más variados.

Los principios de diseño variarán de acuerdo a la función que tengan y al papel que desempeñe en el proceso de enseñanza y aprendizaje, bien sea transmisión de la información, evaluación de los estudiantes, presentación de ejemplos, motivación, simulación de fenómenos, etc.

La evaluación del software educativo puede ser realizada desde diversas perspectivas y por diversas personas y especialistas; respecto de este aspecto Olivares et al. (1990) llaman la atención en que éstos pueden ser especialistas de comunicación informática, especialistas en comunicación audiovisual, evaluadores externos, profesores, alumnos. Cada grupo tendrá preocupaciones diferentes, que llevan a observar aspectos desiguales.

En su investigación Cabero encuentra que tanto alumnos como docentes, sugieren que la forma de diseñar el software educativo cae dentro de la concepción de libro interactivo, forma que se destaca por el uso de organizadores previos, resúmenes, división de pantallas, buen uso de animación, aparición progresiva de la información, utilización de parpadeo, ejercicios prácticos para los alumnos. Enfatiza respecto de la cautela a tener en cuenta en los resultados ya que psicológicamente y actitudinalmente podrían estar influenciados por el factor novedad y la difícil ruptura del paradigma tradicional del libro.

Otras observaciones hacen referencia a la rigidez en cuanto que no deja pasar de la segunda a la tercera etapa, por ejemplo hasta que no se aciertan muchas respuestas bien, por lo que el programa se hace monótono.

Como conclusiones: el programa debería tener un ítem NS/NC (no sabe/no contesta), haría falta un ítem que contemple el uso del hipermedia, que puede ser la clave para la rigidez encontrada. También se encontró que la referencia explícita a valores, temas transversales, contenidos actitudinales, apenas se reflejan o se dan en forma poco explícita Bunes et al. (1993) y Bolívar (1995), pero aunque sujeto a modificaciones, lo puede considerar como válido como una primera aproximación.

Por último, se ha dejado a Bartolomé Pina (1998), de Departamento de Didáctica de la Universidad de Barcelona, quien considera que la larga lista de preguntas en los instrumentos de evaluación, tiene dos objeciones principales: una que proviene de la relevancia de los parámetros observables y la otra de la relatividad de estos parámetros.

Normalmente los parámetros relacionados con la adecuación para la realización de un aprendizaje concreto, capacidad de estímulo resultan difícilmente observables, y su medida suele adolecer de una gran subjetividad. Cabe señalar que hay algunos parámetros observables que pueden ser relevantes, como la explicitación de los objetivos, pero a veces, un programa puede no explicitarlos ya que es parte de un diseño curricular modular y sus objetivos estarán definidos en ese marco. El autor señala que inclusive aquellos parámetros relevantes que hacen referencia a los beneficios en términos de aprendizaje, se relacionan al diseño curricular y al modo de uso de los medios por el docente.

Quizás, sostiene Pina (1998), debería considerarse *el "uso didáctico de los medios"*, ya que este es el aspecto clave. Por consiguiente, habría que evaluar el software en función del uso que se hiciere de él. De este modo siempre habría una forma original para aplicar un programa en los aprendizajes. Sería deseable, definir criterios que ayuden a los docentes a la selección, ya que no existe a información acerca de la evaluación median-

te el uso controlado de un programa determinado. En estos casos, cabe recurrir a la experiencia o a la consulta de los grandes distribuidores cono Anaya Interactiva[21] y Zeta Multimedia[22].

Conclusiones del Estado del Arte

En estos primeras cuatro secciones, se presentó la evolución de los programas educativos de computadora, como un estudio diacrónico, paralelamente a los desarrollos de las teorías de aprendizaje y líneas más difundidas desde los años sesenta.

Se observa, la base conductista de Skinner[23] que permeó los primeros desarrollos y se puede decir que, en sus inicios, toda una época hasta aproximadamente los setenta, fue educada de acuerdo a los principios de la instrucción programada, aunque no exclusivamente sobre la base de software.

Estos primeros diseños, están muy lejos de los actuales hipermedias didácticos, desarrollados mediante aplicación de estrategias cognitivas y metacognitivas con una fundamentación muy fuerte basada en estudios psicología cognitiva.

Mediante este gran paso hacia el conocimiento de los procesos mentales, la psicología cognitiva, intenta la aplicación de estrategias específicas de aprendizaje y del propio conocimiento. Este conocimiento de los propios aprendizajes, es el que permite hacer hincapié en determinadas técnicas específicas para desarrollar ciertas funciones del pensamiento. La interpretación de estas técnicas por parte de los programadores, es de fundamental importancia, como así, la superación de la etapa de construcción de programas monolíticos de software y poco documentados.

La aparición de la programación estructurada y la diversidad de lenguajes que se suceden a partir de los primeros: Basic, Fortran, Cobol, conducen al Smalltalk, C, C+, Visual Basic, etc., que diversifican el panorama actual, pero a la vez permiten la aplicación específica y el uso de la programación orientada a objetos y a eventos, para facilitar los desarrollos de los programas actuales con una gran gama de recursos disponibles, impensables, diez años atrás, por ejemplo.

Uno de los factores, a tener en cuenta, es el vertiginoso desarrollo tecnológico de los últimos años, que facilita las comunicaciones. Las NTIC (Nuevas Tecnología de Comunicación e Información) han sufrido durante los últimos años ritmo de avance muy notorio, cada vez más, se hace necesaria una habilidad mayor en el manejo y procesamiento de volúmenes de información cada vez más grandes. Esto repercute sobre la población estudiantil, ya que se necesitan estudiantes con nuevas habilidades, no sólo de gestión, sino de selección y de acceso a la información que permitan generar nuevo conocimiento. Los programas educativos deben cambiarse y actualizarse en función de estas necesidades, pero es el docente el que debe considerar cuál es la magnitud de dicha necesidad y cuál será su grado de incorporación al aula.

Siempre se vuelve al triángulo irreductible: docente-alumno-contenido, como el punto de partida de toda práctica educativa. Si bien, algunos plantean que los aprendizajes mediados, son en sí, el uso de los medios didácticos, siendo éstos los que determinan en definitiva la eficiencia del proceso de aprendizaje. Aquí se debe señalar la necesidad de hacer buen uso de tales medios y para ello habría que redefinir qué es ser un buen docente, como aquel que hace buen uso didáctico de los medios y no sólo un buen uso de los medios didácticos.

Habría que señalar entonces que hay una relación directa entre el estilo docente y la eficiencia del medio: *¿Un docente en cuestión o un docente en rutina?* Como señala Fernández Pérez (1995), esta es una de las variables más importantes, a tener en cuenta en el acto didáctico, ya que es el estilo docente el que condiciona el uso didáctico de los medios.

Es por ello, que se debe remarcar que existe una gran cantidad de variables a tener en cuenta en las prácticas educativas y que difícilmente se puedan comparar experiencias, a menos que estas se realicen de un modo muy controlado. Pero, trabajar en un ámbito de *"laboratorio",* diseñando experiencias que sean "*comparables*", muchas veces implica cambiar el ambiente natural donde se desarrolla el aprendizaje. Otras veces, significa dejar variables de lado ponderando a priori que su influencia en los resultados es insignificante.

Quedaría por señalar que las largas listas de criterios a desarrollar para evaluar los programas educativos, son datos relativos a la hora de hacer uso del recurso educativo. El rol docente, es el que condiciona, el uso de los programas, siendo la creatividad y la originalidad de las propuestas las que permiten incrementar el valor de los medios y no el medio mismo.

De acuerdo a las necesidades en cada caso, y a la teoría educativa y/o el curriculum, se deberán adaptar algunos de los paradigmas del ciclo de vida, discriminando en cada etapa las actividades a realizar con la documentación, las técnicas y herramientas a utilizar. Este ciclo de vida elegido, será acorde al tipo de lenguaje de programación, particulares como los orientados a objetos, de acuerdo al tipo de proyecto o programa, se realizarán las estimaciones de tiempo, personal y costo, de algún modo convencional.

[21] www.anayainteractiva.com
[22] www.zetamultimedia.es
[23] Citado en Fernández Pérez (1995).

En la mayor parte de los diseños realizados, los usuarios, en general solicitan uno o más prototipos, para ver como evoluciona el desarrollo durante el tiempo previsto y si éstos coinciden con sus expectativas. Aunque se han descripto varios de los modelos de ciclo de vida más usados, es recurrente inclusive en orientación a objetos utilizar modelos con prototipos evolutivos, aunque no se excluyen otras posibilidades.
Se prevé plantear de este modo una posible solución a la problemática de los desarrollos de los programas educativos actuales, mediante la definición de los procesos de construcción de los programas educativos y las tareas involucradas en cada uno de ellos, considerando las herramientas y técnicas a usar en cada una de ellas. Esto podría traducirse en la adopción de una gran matriz de procesos, actividades y tareas, asociadas a un ciclo de vida.
Quedaría por considerar la buena consistencia en la simbiosis de la programación estructurada y la teoría del aprendizaje significativo de Ausubel (1997) y los mapas conceptuales de Novak (1988), de acuerdo a la línea de la Escuela de Harvard de David Perkins (1995) y Howard Gardner (1997). También habría que considerar los desarrollos de Coll (1994) y de Rogers (1984).
El cuanto a las metodologías de desarrollo de programas educativos: se ha detectado que gran parte de los docentes que intentan hacer un uso de las computadoras lo hacen como una necesidad de estar acorde a los avances tecnológicos, sin hacer un uso racional del recurso. No se detectó el uso de una metodología para el desarrollo de los programas, sino la necesidad de usar programas que faciliten la visualización e interpretación de algunos temas, para los cuales se realizan "programas" sin base metodológica.
Por otra parte, existe también una gran cantidad de programadores, sin base didáctica, que intenta aplicar su propia racionalidad y criterio al diseño de los programas educativos. Muchos de ellos son egresados de institutos terciarios, que sólo conocen fundamentos de programación en algún lenguaje, y están muy lejos de las normativas vigentes y de los criterios de modularización, programación estructurada y más aún de la documentación interna y externa de los programas.
De más está decir, que la metodología en la investigación tecnológica, ya sea esta tecnología dura o blanda, es una de las asignaturas ausentes en la mayoría de las carreras de grado, esta ausencia es la que condiciona y limita la confección de buenas propuestas para el desarrollo de aplicaciones tecnológicas. Habría que pensar además, que a esto aún le falta el ingrediente educativo: la enseñanza orientada hacia la pedagogía de la diversidad y la educación reflexiva, dinámica e informada, sobre la que nos hace reflexionar David Perkins (1995) en sus trabajos.
Quizás también, habría que considerar que gran parte de los desarrollos de software se realizan en forma intuitiva incursionando en las dos áreas y el resultado, a la hora de las aplicaciones es bueno, y hasta se logran los objetivos educativos: el alumno aprende. Quizás, llegado a este punto habría que señalar, que: para un especialista novel, viéndolo desde la posición de etnógrafo, le sería más fácil aplicar una metodología con sólidos fundamentos teóricos sumados a la base de conocimientos adquirida sólo por prueba y error y mucho esfuerzo, pero adquirir estos conocimientos requiere de mucho tiempo.
La base epistemológica es fundamental en ambos campos del saber: el infomático y el educativo, y más aún si los mismos se potencian y las metodologías se suman, a la luz de las semejanzas y diferencias, como dice Morín (1985) desde el paradigma de la complejidad en este fin de siglo y Ander Egg (1995) recrea en sus obras.
Se ha presentado el marco conceptual de los desarrollos de software actuales, para poder incorporarlos los diseños de programas educativos, siendo a partir aquí que se marca la necesidad de una trabajo interdisciplinario muy fuerte.
Los pilares de las aplicaciones tecnológicas están condicionados, es época de trabajo interdisciplinario, en este caso concibiendo la diversidad y multidimensionalidad de las actividades que se requieren hoy día y su interacción. El desarrollo tecnológico y la práctica docente, deben avanzar juntos; aunque en la realidad no sea así, al menos hasta ahora.
Quizás, habría que capacitar a los profesionales de ambas áreas insistiendo en un nuevo modo de ver las cosas, dejando los reduccionismos y la unidimensionalidad de la realidad por una visión poliocular. (Morín, 1995).
En síntesis: habría que reiterar como solución a las deficiencias detectadas que a la programación estructurada puede sumarse la teoría del aprendizaje significativo con todas sus variantes para el desarrollo de los programas. En cuanto a la evaluación deben considerarse con mucho detenimiento las variables involucradas a la hora de hacer comparaciones.

Descripción de la Problemática

5. Presentación de la problemática

5.1.Resumen

Se desea presentar la problemática que responde a la siguiente pregunta: ¿Qué normativas provee la ingeniería de software y cuáles son las teorías de aprendizaje a aplicar para el diseño del software educativo? (sección 5.2).
Se desglosa la problemática, ya que gran parte de los programas educativos desarrollados en la actualidad no consideran las metodologías provistas por la ingeniería de software para su diseño (sección 5.3). A este defecto se le agrega que en otros casos se desconocen los principios de las teorías de aprendizaje que les dan marco, no se evalúan lo suficiente (sección 5.4) y no se aplican las pautas de calidad (sección 5.5).

5.2. La problemática

Se ha detectado una serie de problemas relacionados con el diseño y el desarrollo del software educativo. Por una parte *existe una falta de capacitación de los docentes de áreas no informáticas, que conduce a intentos de construcción de programas educativos sin fundamentos metodológicos y una falta de capacitación en el* área educativa de los programadores que intentan desarrollar programas educativos y que en la mayoría de las veces caen en desarrollos meramente conductistas. Esto no significa que la base conductista no sea apropiada en determinados contextos y favorecedora de determinadas situaciones de aprendizaje.
Por estos motivos, realizar un producto con características tan particulares, requiere de una sólida base en ingeniería de software sumada a un conocimiento de las teorías de aprendizaje y a los ámbitos de aplicación, es decir a los entornos socio-educativos actuales.
El problema se puede dividir, para su mejor estudio, en dos subproblemas relacionados: por un lado el problema de cómo diseñar y desarrollar programas educativos de calidad y por otro lado cómo evaluar este tipo de desarrollos.

5.3. El diseño y desarrollo del software educativo

Los programas educativos constituyen un producto que posee algunas características, como las técnicas, que pueden ser evaluadas mediante el uso de métricas adecuadas y otras que están relacionadas con la significatividad de los aprendizajes, en un ámbito donde la cantidad de variables a tener en cuenta son muchas: el estilo docente, el modo de uso de la herramienta informática por el docente, el estilo de aprendizaje de los alumnos, el contexto áulico, el estilo institucional, el tipo de curriculum, etc.
Algunas otras características las define el usuario final, es decir, son los alumnos quienes en última instancia deciden si el producto es eficiente o no, después de haberse finalizado el programa y realizado la evaluación contextualizada.
El uso de metodologías adecuadas de desarrollo en principio, puede solucionar el problema de la calidad, para obtener un programa educativo, libre de errores, o con una tasa de errores no encontrados ni detectados, tolerable, esto requiere a su vez de un equipo de desarrollo con experiencia. Pero, como en cada producto desarrollado debe desencadenar un aprendizaje de cierto tipo o desarrollar una habilidad o capacidad en particular, sumada a la calidad técnica se necesita una evaluación desde el punto de vista de los aprendizajes que se pretenden lograr.

5.4. La evaluación del software educativo

La evaluación es para los programas educativos, la etapa más importante del todo el proceso de construcción, evaluando desde el diseño del producto y la producción del mismo, hasta el modo de uso, el tiempo y el momento de uso. La evaluación es una tarea constante a lo largo de todo el desarrollo y aún después, en el contexto de aplicación, ya que requiere también de evaluación de las estrategias cognitivas propuestas.
Como la cantidad de software educativo ha crecido muy rápidamente, el docente se encuentra con la necesidad cada vez mayor de evaluarlo para determinar el grado de adecuación de un programa su propio entorno. Ellos necesitan saber cómo utilizar un programa determinado y cuándo deberían utilizarlo para mejorar su enseñanza. Por otra parte los alumnos también deben saber cómo mediante tal o cual programa podrían mejorar sus aprendizajes.
Es cierto, que confeccionar un instrumento de evaluación y realizar la evaluación de un programa en particular con un grupo de alumnos específico no brindará resultados generalizables a todos los ámbitos de aplicación, pero esta puede ser una guía como punto de partida de selección del programa para el docente.
Los proveedores de programas educativos deberían informar y aconsejar al docente acerca de la conveniencia de usar tal o cual programa adecuándose a las necesidades de éste, pero son muy pocas las empresas que realmente brindan el asesoramiento pedagógico.

En la actualidad son muy pocos los programas que tienen documentación interna y aún externa. En muchos casos los manuales de usuario se remiten a especificaciones técnicas y requerimientos del programa.
Las herramientas de evaluación que se han relevado son largas y tediosas listas de preguntas o listas de verificación que en la mayoría de los casos no pueden ser aplicadas más que a una situación de aprendizaje en particular: un programa educativo, con un docente con un estilo propio, con alumnos en una situación áulica, en una realidad diferente a las demás. Por este motivo, una lista de evaluación significa personalizarla a las condiciones de trabajo en particular. De este modo sólo se pueden usar sus resultados como orientativos y no cómo comparativos entre dos situaciones de aprendizaje diferentes.
Es en este momento donde se debe recordar que aprender es el fin y las computadoras es el medio mediante el cual esos aprendizajes se pueden facilitar y que no es el *"único medio*": si no con sólo recordar que *"Sócrates no tenía computadoras para enseñar",* bastaría para darse cuenta que si bien esta herramienta nos permite gestionar los grandes volúmenes de información que hoy día se manejan, es tan sólo uno de los medios a través de los cuales se puede aprender. Un muy buen medio mal empleado puede ser más nocivo que el más discreto de los medios empleado en el momento preciso, en el lugar adecuado y en la cantidad necesaria.
Cuando se evalúan los programas educativos se consideran largas listas de criterios aplicadas en situaciones que distan mucho del contexto áulico. Estas listas carecen en algunos casos de sentido práctico por su falta de dinámica y de adaptación al cambio tecnológico constante.
Algunas evaluaciones experimentales, utilizan grupos de alumnos (de igual edad, distribución de género, conocimientos previos) con comparación de resultados. Pensar en comparar los resultados obtenidos, mediante pruebas tradicionales entre un grupo que utiliza la herramienta computacional y el software educativo con otro que utiliza medios convencionales como libros, significa caer en el reduccionismo de pensar que aprender es reproducir contenidos, repetir o memorizar contenidos.
Este es uno de los problemas que a menudo se presenta en las evaluaciones del software educativo: realizar una evaluación mediante una lista de comprobación, es útil, pero faltan los alumnos y evaluar logros de dos grupos comparando resultados, significa considerar al alumno sin la influencia del entorno social, la motivación inherente de esta contextualización y perder de vista la expectativa social que lo conduce a adquirir ese aprendizaje.

5.5. La calidad en el software educativo

El problema de la calidad se debe situar en el contexto de diseño y desarrollo como un producto de software más y en el contexto de calidad educativa como respuesta a un modelo de aprender o un modelo curricular, según las características de un programa que puede ser de refuerzo a una clase explicativa o de apoyo para trabajo colaborativo.
El problema de la determinación de la calidad en el uso de medios de comunicación es uno de los problemas educativos recurrentes. Si bien se pueden utilizar los catálogos de listas de verificación, pero la idea es dar a estas listas un marco de uso práctico.
Cuando se habla de calidad de la enseñanza que se ofrece a los estudiantes como uno de los objetivos educativos a lograr mediante aplicación de programas bien diseñados, se piensa en el logro de los estudiantes con un perfil lo más cercano posible al ideal.
La calidad educativa de los programas se puede ver como la potenciación de habilidades cognitivas y de adquisición de conocimientos, mediante el uso de programas específicos que desencadenan las funciones superiores del pensamiento.
Por otra parte se quiere estudiantes, capaces de controlar el contenido de sus aprendizajes y de su trabajo, promoviendo el autoaprendizaje[24], y el aprender a aprender, mediante el uso y conocimiento de estrategias metacognitivas.

Por estas razones, en las secciones siguientes se planteará, una de las posibles soluciones a esta problemática. Para ello, se tendrán en cuenta las herramientas y recursos a utilizar descriptos en el Estado del Arte, debiéndose destacar que existe además una necesidad imperiosa de rediseño de las prácticas educativas[25] y una concientización de un perfeccionamiento docente constante a lo largo de la trayectoria del mismo.
Esto permitirá considerar, analizar, incorporar y por último poner en práctica no sólo el uso de los programas educativos, sino aspectos concernientes a su diseño, desarrollo y evaluación, utilizando los criterios científicos y tecnológicos apropiados y acordes a las necesidades particulares.

[24] o aprendizaje autónomo.
[25] en referencia a que: *"La cultura del aprendizaje dirigida a la reproducción de saberes previamente establecidos, debe dar paso a una cultura de la comprensión, del análisis crítico, sobre lo que hacemos y creemos. .. "* (Pozo Municio, 1998).

Solución Propuesta

6. Propuesta de metodología de diseño y desarrollo

6.1. Resumen

Se propone y se justifica el ciclo de vida elegido para el diseño del software educativo (sección 6.1). Se consideran las etapas del ciclo y se define la matriz de actividades (sección 6.4). A partir de ella se describen las herramientas y las técnicas que se utilizarán en cada uno de los procesos considerados.

En la propuesta se ha elegido un ciclo de vida de prototipos evolutivos con refinamientos sucesivos como punto de partida. En la etapa metodológica de diseño se integra a los instrumentos de representación clásicos los que provee el enfoque cognitivista–contructivista.

La propuesta metodológica considera la construcción de programas educativos desde un aspecto integral, teniendo en cuenta los aspectos pedagógicos en el ciclo de vida.

Se pone un interés especial en la configuración de los perfiles de los diferentes profesionales del equipo de desarrollo. (sección 6.4), en el diseño del programa (secciones 6.5 y 6.6) y en la confección de la documentación de los procesos (sección 6.7).

6.2. La elección del ciclo de vida

El modelo de ciclo de vida elegido es el de *prototipos evolutivos con refinamientos sucesivos,* por varios motivos a saber:

- Cuando el software a desarrollar es por encargo, es interesante tener una idea de cómo será el programa lo antes posible, y a fin de disminuir las expectativas del cliente o usuario, se le irán entregando prototipos con funcionalidades en forma incremental, para que se los pruebe durante un período de tiempo a convenir y haga las sugerencias y los cambios en etapas lo más tempranas posibles del ciclo de vida.
- Por otra parte, es importante cuando se desea que el usuario sepa cuanto antes si el producto tal cómo se lo interpretó está de acuerdo a sus necesidades y consideraciones.
- En muchos casos, el usuario no puede dar una idea detallada de lo que desea, y debido a ello, el desarrollador no termina de saber qué es lo que éste quiere exactamente, por lo que cada prototipo realizado, significa una revisión de los requerimientos y un refinamiento de dichos requerimientos a fin de acercarse al producto final.

  En el ciclo de vida de prototipo incremental se definen las siguientes etapas:

  1. *Factibilidad (FAC)*
  2. *Definición de requisitos del sistema (RES)*
  3. *Especificación de los requisitos del prototipo (REP)*
  4. *Diseño del prototipo (DPR)*
  5. *Diseño detallado el prototipo (DDP)*
  6. *Desarrollo del prototipo (codificación) (DEP)*
  7. *Implementación y prueba del prototipo (IPP)*
  8. *Refinamiento iterativo de las especificaciones del prototipo (aumentando el objetivo y/o el alcance). Luego, se puede volver a la etapa 2 o continuar si se logró el objetivo y alcance deseados. (RIT)*
  9. *Diseño del sistema final (DSF)*
  10. *Implementación del sistema final (ISF)*
  11. *Operación y mantenimiento (OPM)*
  12. *Retiro (si corresponde) (RET)*

A continuación se describen cada una de las etapas del ciclo de vida elegido que formarán parte de la matriz de actividades[26]:

Factibilidad (FAC): En esta etapa se define el producto software y se determina su factibilidad en el ciclo de vida desde la perspectiva de la relación costo –beneficio, como así las ventajas y desventajas respecto de otros productos.

Requisitos del sistema (RES): En esta etapa se deben definir las funcionalidades requeridas para el desarrollo del sistema (o programa), las interfaces y el tipo de diseño.

Especificación de requisitos del prototipo (REP): Consiste en especificar las funciones requeridas, las interfaces y el rendimiento para el prototipo. Aquí se considerarán incrementos en porcentajes de la funcionalidad total del sistema.

Diseño del prototipo (DPR): Es poner en ejecución del plan del prototipo, ya que una vez fijadas las restricciones con el usuario, hay que mostrar el mismo funcionando, aunque sean sólo algunas fun-

[26] La matriz de actividades base para el diseño de software, se adaptó de Juristo Juzgado N. (1996): *Proceso de Construcción del software y ciclos de vida en módulos de Proyectos de Software.* Universidad Politécnica de Madrid.

cionalidades restringidas. Aquí, hay que hacer un análisis de cómo se va a trabajar, qué módulos se van a hacer, con qué lógica y qué funciones se van a usar.

Diseño detallado del prototipo (DDP): Esta etapa es una especificación verificada de la estructura de control, la estructura de los datos, las relaciones de interfaces, el tamaño, los algoritmos básicos y las suposiciones de cada componente del programa. En esta etapa no sólo se definen y sino que se documentan los algoritmos que llevarán a cabo la función a realizar por cada uno de los módulos. El diseño de software, es un proceso que se centra en cuatro atributos distintos del programa: la estructura de datos, la arquitectura del software, el detalle procedimental y la caracterización de la interface. En este proceso deben traducirse los requisitos a una representación del software que pueda ser establecida de forma que se obtenga la calidad requerida antes de que comience la codificación.

Desarrollo del prototipo (codificación) (DEP): Consiste en realizar la codificación o diseño detallado, en forma legible para la máquina.

Implementación y prueba del prototipo (IPP): Consiste en lograr un funcionamiento adecuado del producto software en el sistema informático, funcionando operacionalmente, incluyendo objetivos tales como la conversión del programa y datos (si la hubiere), la instalación y el entrenamiento. La prueba debe asegurar que se han probado toas las sentencias del mismo, y que en las funciones externas se han realizado pruebas que aseguren que la entrada definida produce los resultados que se esperan realmente.

Refinamiento iterativo de las especificaciones del prototipo (RIT): Es un aumento de la funcionalidad del sistema, para luego volver REP a fin de aumentar la funcionalidad del prototipo o continuar, si se logró el objetivo y alcance deseados.

Diseño del sistema final (DSF): Consiste en ajustar las restricciones o condiciones finales e integrar los últimos módulos.

Implementación del sistema final (ISF): Es el sistema de informático funcionando operativamente, incluyendo tales objetivos como conversión del programa y datos, (si la hubiere), la instalación y La capacitación del personal..

Operación y mantenimiento (OPM): Es la puesta en funcionamiento del sistema informático, objetivo que se repite para cada actualización.

Retiro (RET): Es una transición adecuada de las funciones realizadas para el producto y sus sucesores

Luego, se definen los procesos básicos para este ciclo de vida, y las actividades para cada uno de ellos. Los procesos incluyen aquellos concernientes al desarrollo de software y los específicos teniendo en cuenta los aspectos educativos, aunque en la propuesta, no se define una teoría educativa en particular, sino que las actividades permiten ver un enfoque cognitivista-constructivista.

En la Tabla 6.1, se puede ver la matriz de actividades, con el desglose de actividades para cada uno de los procesos para el ciclo de vida elegido.

En la matriz de actividades, se puede observar en color gris, las etapas del ciclo de vida involucradas al incrementar las funcionalidades a fin de obtener los diferentes prototipos, que han de desarrollar. En arial *normal* se destacan las actividades relativas al desarrollo de software propiamente dicho y en *arial cursiva* a aquellas actividades concernientes a definir y considerar los aspectos educativos didáctico-pedagógicos para desarrollar el software.

Una vez definidas cada una de las actividades, quedaría por considerar qué herramientas y técnicas se utilizarán para cada una de ellas, teniendo en cuenta el modelo de ciclo de vida elegido (prototipado evolutivo) y la teoría educativa que sustente al desarrollo.

6.3. La matriz de actividades (Tabla 6.1)

ACTIVIDADES DE LOS PROCESOS	FAC	RES	REP	DPR	DDP	DEP	IPP	RIT	DSF	ISF	OPM	RET
Proceso de identificación de la necesidad educativa												
Identificar necesidad del programa educativo	✓											
Identificar o seleccionar la teoría educativa a utilizar para el desarrollo de acuerdo a la necesidad	✓											
Proceso de selección del modelo de ciclo de vida												
Identificar de los posibles modelos ciclos de vida el que más se adapta a las necesidades.	✓											
Seleccionar un modelo para el programa de acuerdo a la teoría educativa elegida.	✓											
Proceso de iniciación, planificación y estimación del proyecto												
Establecer la matriz de actividades para el ciclo de vida considerando la teoría educativa a usar	✓											
Asignar los recursos del proyecto/programa	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Definir el entorno del proyecto/programa	✓	✓	✓		✓			✓				
Planificar la gestión del proyecto/programa	✓	✓	✓					✓				
Proceso de seguimiento y control del proyecto												
Analizar los riesgos	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Realizar la planificación de contingencias		✓	✓	✓	✓	✓	✓	✓	✓	✓		
Gestionar el proyecto/programa	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Implementar el sistema/programa	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Archivar registros			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Proceso de gestión de calidad del software												
Planificar la garantía de calidad del software		✓	✓	✓				✓				
Desarrollar métricas de calidad		✓	✓	✓	✓	✓		✓				
Gestionar la calidad del software	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Identificar a los responsables de cada tarea	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Implementar el sistema de informes de problemas			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Archivar registros			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Proceso de exploración de conceptos												

Identificar las necesidades educativas	✓											
Formular las posibles soluciones potenciales	✓	✓	✓									
Identificar las necesidades del soporte lógico	✓	✓	✓						✓			
Formular soluciones potenciales compatibles	✓	✓	✓									
Realizar los estudios de viabilidad	✓	✓							✓			
Refinar y concretar la idea o necesidad		✓	✓						✓			
Proceso de asignación del sistema												
Analizar las funcionalidades del sistema/programa		✓	✓	✓					✓			
Definir las funcionalidades del programa		✓	✓	✓								
Desarrollar la arquitectura del sistema/programa basándose en la teoría educativa elegida.		✓	✓	✓								
Descomponer los requisitos del sistema/programa		✓	✓									
Proceso de análisis de requisitos educativos												
Definir los objetivos educativos		✓	✓	✓								
Definir las características del grupo destinatario		✓	✓	✓								
Definir los contenidos y el recorte de contenidos		✓	✓	✓								
Definir las estrategias didácticas		✓	✓	✓								
Definir las actividades mentales a desarrollar		✓	✓	✓								
Definir el nivel de integración curricular		✓	✓	✓								
Definir el tipo de uso del programa y nivel de interactividad		✓	✓	✓								
Definir los efectos motivantes		✓	✓	✓								
Definir los posibles caminos pedagógicos		✓	✓	✓								
Definir el tiempo y modo de uso		✓	✓	✓								
Definir el hardware necesario		✓	✓	✓								
Proceso de análisis de requisitos del software												
Definir el tipo de programa a desarrollar		✓	✓	✓								
Definir los requerimientos de las interfaces		✓	✓	✓					✓			
Definir el tipo de interactividad		✓	✓	✓								
Definir y desarrollar los requisitos del software		✓	✓	✓					✓			
Priorizar e integrar los requisitos educativos con los del software		✓	✓	✓					✓			
Proceso de diseño												
Definir la organización de los menús				✓	✓							
Definir el tipo de íconos a usar				✓	✓							
Seleccionar los efectos a usar: sonidos, música, animaciones, vídeos, etc.				✓								

Seleccionar los textos a usar				✓								
Asegurar la facilidad de lectura				✓								
Realizar el diseño de las pantallas				✓	✓							
Realizar el diseño de los menús				✓								
Realizar los storyboards (si corresponde)				✓	✓							
Realizar el diseño preliminar				✓								
Analizar el flujo de información				✓	✓							
Diseñar la base de datos (si la hubiere)				✓	✓							
Diseñar las interfaces				✓	✓							
Definir los criterios de navegación				✓								
Definir las actividades: Información, preguntas, búsqueda, resolución de ejercicios, etc.				✓								
Definir los tipos de módulos a usar: de evaluación, de problemas, de preguntas, etc.				✓								
Definir los tipos de ayudas didácticas: errores, mensajes de ayuda, etc.				✓								
Desarrollar los algoritmos		✓	✓	✓	✓							
Realizar el diseño detallado												
Confeccionar la documentación				✓	✓							
Proceso de implementación e integración de módulos												
Crear los datos para las pruebas				✓	✓	✓			✓			
Crear el código fuente				✓	✓	✓			✓			
Generar el código objeto				✓	✓	✓			✓			
Crear la documentación de operación				✓		✓			✓			
Planificar la integración de los módulos				✓	✓	✓			✓			
Proceso de instalación y aceptación												
Planificar la instalación							✓		✓			
Distribuir el software							✓			✓		
Instalar el software							✓			✓		
Cargar la base de datos si la hubiere							✓			✓		

ACTIVIDADES DE LOS PROCESOS	FAC	RES	REP	DPR	DDP	DEP	IPP	RIT	DSF	ISF	OPM	RET
Aceptar el software en el entorno de operación							✓			✓		
Realizar las actualizaciones pertinentes							✓				✓	
Proceso de operación y soporte												
Operar el sistema/programa											✓	

Proveer asistencia técnica y consultas on line.											✓	
Mantener el historial de pedidos de soporte											✓	
Proceso de mantenimiento												
Realizar el mantenimiento correctivo											✓	
Reaplicar el ciclo de vida del software											✓	
Proceso de retiro												
Notificar al usuario											✓	✓
Conducir operaciones en paralelo												✓
Retirar el sistema												✓
Proceso de verificación y validación												
Planificar la verificación y validación del software		✓	✓	✓								
Ejecutar las tareas de verificación y validación		✓	✓	✓	✓							
Recoger y analizar los datos de las métricas		✓	✓	✓	✓	✓	✓		✓	✓	✓	✓
Planificar las pruebas de V y V					✓	✓	✓		✓	✓	✓	✓
Desarrollar las especificaciones de las pruebas					✓	✓						
Ejecutar las pruebas							✓		✓			
Archivar los resultados							✓		✓	✓		
Proceso de evaluación de los prototipos del software												
Confeccionar el instrumento de evaluación							✓					
Evaluar los prototipos del programa							✓					
Elaborar los resultados							✓					
Identificar los cambios y ajustes a realizar							✓					
Llevar a cabo las modificaciones pertinentes							✓					
Archivar los resultados							✓					
Proceso de evaluación interna y externa del software												
Confeccionar los instrumentos de evaluación interna y externa									✓			
Realizar las evaluaciones interna y externa.									✓			
Elaborar los resultados									✓			
Llevar a cabo las modificaciones pertinentes									✓			
Obtener la versión final del programa.									✓			
Archivar los resultados									✓			
Proceso de evaluación contextualizada												
Diseñar la evaluación: definir grupo/s (caso de control y experimen-									✓			

tal), tiempo, docente, materiales a usar y modo.												
Aplicar de la prueba									✓			
Identificar posibles problemas.									✓			
Realizar las modificaciones y ajustes a la versión									✓			
Archivar los resultados									✓			
Proceso de configuración												
Planificar la gestión de la configuración		✓	✓	✓								
Realizar la gestión de la configuración		✓	✓	✓	✓	✓	✓		✓			
Realizar el control de la configuración				✓	✓	✓	✓		✓	✓	✓	✓
Realizar la información del estado de la configuración				✓	✓	✓	✓		✓	✓	✓	✓
Proceso de documentación técnica												
Planificar la documentación interna			✓	✓	✓							
Planificar la documentación externa			✓	✓	✓							
Implementar las documentaciones						✓	✓					
Incluir los resultados de las evaluaciones							✓		✓			
Elaborar el manual de usuario con información técnica.			✓	✓	✓	✓	✓	✓	✓			
Producir y distribuir la documentación			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Proceso de documentación didáctica												
Planificar la documentación didáctica			✓	✓	✓							
Elaborar una guía didáctica, con ejemplos de uso.								✓	✓			
Adjuntar la información didáctica pertinente, los caminos pedagógicos, las teorías de aprendizaje y la programación didáctica.									✓			
Producir la documentación y adjuntarla al programa.						✓	✓		✓	✓		
Proceso de formación del personal												
Planificar el programa de formación		✓	✓	✓								
Desarrollar los materiales de formación			✓	✓	✓	✓	✓		✓			
Validar el programa de formación						✓	✓		✓	✓		
Implementar el programa de formación							✓			✓		

Tabla 6.1: Matriz de actividades básica para un programa educativo, con ciclo de vida de prototipado evolutivo.

En la Tabla 6.2, se pueden observar los métodos, las técnicas y/o herramientas a emplear en cada uno de los diferentes procesos

Procesos		Documento de salida	Métodos/Técnicas/ Herramientas a emplear
Proceso de identificación de la necesidad educativa		Definición del marco educativo y comunicacional.	Encuesta, entrevista
Proceso de selección del modelo de ciclo de vida		Ciclo de vida adoptado	
Proceso de iniciación, planificación y estimación del proyecto		Plan de gestión del proyecto	Diagrama de Gantt o Pert (CPM). Modelos empíricos de estimación
Proceso de seguimiento y control del proyecto (programa),		Análisis de riesgos y plan de contingencias. Registro histórico del proyecto	Modelizado. Prototipado. Revisiones. Auditorías. Análisis CPM.
Proceso de gestión de calidad del software		Plan de garantía de calidad. Recomendaciones de mejora de calidad.	Técnicas de planificación. Métricas de calidad del software
Proceso de exploración de conceptos		Informe de necesidades. Posibles soluciones factibles	Análisis Costo Beneficio. DFD. Prototipado
Proceso de asignación del programa (sistema).		Especificación de requisitos funcionales de hardware y software. Especificación de interfaces del sistema o programa. Descripción funcional. Arquitectura.	DFD Módulos
Proceso de análisis de requisitos educativos		Especificación de los objetivos y estructuración de conceptos. Selección de contenidos y pertinencia.	Enfoques cognitivistas. Enfoques constructivistas. Estrategias cognitivas.
Proceso de análisis de requisitos del software		Especificación de requisitos del software, de interfaces de usuario y otros software. Interface de hardware y con el sistema físico.	Análisis estructurado. DFD. DD. Diagramas E/R. Técnicas de Prototipación.
Proceso de diseño	de los contenidos	Identificación de los procesos mentales a estimular. Definición de las actividades a realizar por los alumnos. Jerarquización de los conceptos.	Uso de estrategias cognitivas. Teoría de Ausubel y Novak.[27] Uso de mapas conceptuales.
	del software	Descripción del diseño del software y de la arquitectura. Descripción del flujo de información, bases, interfaces y algoritmos.	Programación estructurada. Programación Orientada a objetos. Técnicas de prototipado.
Proceso de implementación e integración de módulos		Datos para las pruebas. Documentación del sistema o programa y del usuario. Plan de integración.	Lenguajes de Programación
Proceso de instalación		Plan de instalación. Informe de Instalación.	Lenguajes de Programación
Proceso de operación y soporte		Histórico de pedidos de soporte	Análisis estadístico.
Proceso de mantenimiento		Recomendaciones de mantenimiento	Reaplicar el ciclo de vida
Proceso de retiro		Plan de retiro	
Proceso de verificación y validación		Plan de verificación y validación Plan de pruebas. Especificación y resumen de la prueba. Software probado.	Pruebas de caja negra y pruebas de caja blanca
Proceso de evaluación de los prototi-		Diseño del instrumento de evaluación.	Cuestionario estructurado, semi y

[27]Se ha señalado esta teoría como una guía simplemente, a modo de referencia, puede utilizarse otra, de acuerdo a las necesidades y/o tipo de propuesta educativa. Ver referencias: Ausubel (1983) y Novak (1988).

pos del software	Resumen de la prueba. Selección de la muestra.	abierto.	
Proceso de evaluación interna y externa del software	Diseño del instrumento de evaluación. Resumen de la prueba. Selección de la muestra.	Cuestionario estructurado, semi y abierto	
Proceso de evaluación contextualizada	Diseño de la experiencia. Definición de los grupos de control y experimental.	Técnicas de análisis pre-post. Test de Raven. Prueba de Wilcoxon.	
Proceso de configuración	Plan de gestión de la configuración	Base de datos	Diagramas Pert y Gantt
Proceso de documentación técnica	Plan de documentación técnica.		
Proceso de documentación didáctica	Plan de confección de la documentación didáctica.		
Proceso de formación y capacitación del personal	Plan de formación y capacitación		

Tabla 6.2: Definición de los métodos, técnicas y/o herramientas a utilizar en cada proceso

6.4. El equipo de trabajo

La creación de los programas educativos, es una tarea que compete a diferentes áreas del saber, y, en este tipo de proyectos educativos es fundamental la formación y la conformación de los equipos de desarrollo.
Para ello, es necesario recurrir a especialistas en desarrollos de software, planificadores eficaces y docentes conocedores del área de experticia en que se desarrollará el programa ya que en cada caso habrá que generar un modelo pedagógico de acuerdo a las necesidades.
Si bien, la conformación de los equipos de desarrollo para software comercial, es una instancia que requiere de la habilidad del líder del proyecto y de la buenas relaciones entre los integrantes, en este caso en particular, la situación se vuelve mucho más crítica, debido a interdisciplinariedad del equipo, centrado en la tarea de integrar y coordinar profesionales de campos disciplinares diferentes. Básicamente; se requieren de profesionales con los perfiles que se señalan en la Tabla 6.3:

Profesionales del área en la que se quiere desarrollar el software	Profesores y especialistas en pedagogía para determinar los contenidos a incluir y expertos en el área de desarrollo
Profesionales desarrolladores de software	Analistas y programadores. Para el análisis del proyecto y la codificación.
Coordinador del proyecto	Como en todo proyecto soportado por una ingeniería de base, recaerá en el ingeniero de software.
Personal técnico de apoyo (diseño gráfico y sonido)	De acuerdo a las dimensiones del desarrollo habrá operadores, diseñadores gráficos, especialistas en sonido, vídeo.

Tabla 6.3: Profesionales que se requieren para construir software educativo.

Algunos autores como Marquès (1995) consideran al proceso de desarrollo del software en un eje centrado en el equipo pedagógico. Se deberá tener en cuenta que el programa a desarrollar tiene dos aspectos: uno que es el logro de la calidad técnica que es responsabilidad del equipo técnico y ése debe ser el eje central, ya que diseñar y desarrollar software es responsabilidad de profesionales del área. La determinación de las tareas a asignar de cada uno integrantes del equipo de desarrollo, en cada una de las etapas juega un rol crucial a la hora de optimizar los tiempos de desarrollo.
Cabe señalar que en algunos casos que así lo requieran se recurrirá a asesores en tecnologías informáticas muy específicas y de redes aplicadas a la educación, y psicopedagogos que orientarán al equipo cuando se requiera un producto con alguna característica particular.
Uno de los aspectos a tener en cuenta en todo equipo de trabajo, es una perfecta definición de las actividades de cada uno de los miembros. El *"quién hace qué"*, facilita la identificación de los responsables en cada una de las etapas de trabajo, y permite un seguimiento evolutivo de cada uno de los componentes del software a lo largo de su desarrollo.
Por otra parte, de acuerdo al modelo de desarrollo propuesto, habrá evaluaciones del producto, donde tendrán que intervenir todos los integrantes del equipo.

6.5. El primer diseño del programa

Para diseñar el entorno de comunicación, cuando el equipo de desarrollo tiene escaso conocimiento en informática, comúnmente se utiliza la técnica de guiones o de *"storyboard"*. Estas series de bosquejos, se los debe considerar como una puesta en común de las necesidades, y se llevan a cabo a partir del acuerdo de las partes del equipo.
Los desarrolladores de software expertos y especialistas en el diseño de interfaces humanas elaborarán algunos prototipos de las pantallas a tal efecto. Es por ello, que se descartará el uso de los guiones en este sentido, no así desde la perspectiva de un punto de partida común entre docentes y programadores para interpretar los requerimientos de trabajo.

6.6. Acerca del diseño

Al análisis de requisitos educativos, le sobreviene el análisis de requisitos de software. Luego, la etapa de diseño del software educativo es una de las más importantes en el ciclo de vida de este producto, donde deben interactuar los especialistas en pedagogía y en contenidos con los de diseño de software desde la perspectiva computacional.
Se puede dividir la etapa de diseño, en dos grandes sub–etapas: una es el proceso de diseño de los contenidos y la otra el proceso de diseño del software.
Posteriormente, el diseño de los contenidos se integrará con el del software, siendo este un punto crucial, ya que aquí es donde hay que poner a prueba las estrategias cognitivas, para que el alumno tenga la posibilidad de acceder al conocimiento a través de las diferentes actividades propuestas.
Aquí, resulta fundamental la buena estructuración de los conceptos, considerando en definir correctamente el tema, los objetivos propuestos y las actividades para que el alumno pueda acceder al conocimiento.
Quizás, se podría afirmar que hay que definir las tareas y las actividades, siendo éstas las diferentes situaciones problemáticas referidas a la temática en cuestión que se le presenten al alumno.
En algunos casos, las tareas a ejecutar por los estudiantes pueden estar definidas en niveles de complejidad y de dificultad incrementales. Aunque este puede ser un punto cuestionable, ya que si el software pretende ser de soporte al docente, las tareas y actividades las podría proponer el mismo enseñante, a través de consignas claras y precisas.
Algunos autores, plantean la realización de un análisis subjetivo o sea, un panorama total de la tarea, desde la perspectiva de la tarea misma y del sujeto que la resuelve, de su nivel de dificultad, de la estructura, y de la forma como se organiza y articula con los objetivos y posibles modos de solución. (Valencia, 1998). Este autor plantea el análisis desde la descripción objetiva de la consigna y el análisis desde los factores que desde el punto de vista cognitivo problematizan y/o hacen significativo el conocimiento.

6.7. La documentación

Un aspecto a considerar que es de especial relevancia es el desarrollo de la documentación técnica a lo largo del ciclo de vida del producto: ya sea esta documentación interna y/o externa. La primera considera a todos aquellos comentarios del programa que serán útiles a la hora de realizar modificaciones posteriores, ya sea por el mismo equipo de desarrollo o por otro. Aunque también hay que prever las ayudas on-line a los usuarios, para lo cual se deberá desarrollar un módulo especial a tal efecto, si se la piensa implementar.
La documentación externa, es la que se refiere a todo el material confeccionado desde la etapa inicial de análisis, con los diagramas de entidades y relaciones, estructuras de datos, diagramas de flujos de procesos, diseño modular descendente, etc., y toda aquella que se considere pertinente para interpretar el desarrollo del programa.
Por último hay que señalar la necesidad de un manual del usuario claro y didáctico, para que el docente pueda recurrir a él como elemento de ayuda. En este manual, se considerarán todos los aspectos técnicos requeridos para el funcionamiento del programa y se podría incluir una guía de soluciones, para aquellos problemas más frecuentes.
Se puede considerar también la confección de una *"guía o manual didáctico",* para brindarle al docente todo aquello que se considere necesario para su aplicación. En esta guía o manual de aplicaciones didácticas se debe dar referencia acerca de: objetivos, contenidos, destinatarios, actividades propuestas, y sería interesante incluir qué teoría o teorías de aprendizaje sustentan el desarrollo y cuál es el tratamiento de los errores de los estudiantes en sus procesos de aprendizajes, que permite el programa.
También se deberían almacenar los resultados de las evaluaciones realizadas, considerando los aspectos técnicos, hayan sido éstos señalados como positivos o negativos y las funcionalidades, detallando los resultados estadísticos y teniendo en cuenta el tipo de instrumento elaborado para la toma de dichos resultados y la selección y el tipo de las muestras, de quiénes evaluaron el producto.

6.8. Otras cuestiones

Hay que señalar que tanto para proyectos grandes o pequeños, se podrá realizar la estimación del esfuerzo, del tiempo y los recursos mediante un método convencional, del tipo COCOMO, puntos funcionales u otras.

De este modo se obtendrá el tiempo estimado de duración del proyecto, el personal PSF (personal software full–time) requerido, y el costo del proyecto. Hay que señalar además, que se debe tener en cuenta la estimación de las LDC (líneas de código) para realizar el cálculo y esto presupone alguna experiencia en proyectos similares.
Recordando el principio de Gilb (1988) de los objetivos difusos: *"los proyectos que no tienen objetivos claros, no lograrán sus metas claramente"*, lo que significa que para poder llevar a cabo un proyecto o simplemente un programa, es necesario una buena planificación. Una buena planificación es una distribución eficiente de los recursos en el tiempo. Para ello se plantean metas y submetas, las que se deberán cumplir, de acuerdo a los diagramas de Gantt, por ejemplo.
Para una buena distribución de los recursos es necesario, partir de la definición clara de los requerimientos, y así continuar a lo largo del ciclo de vida del software. Esto conlleva a un conocimiento acabado de todas las actividades involucradas con previsión de los recursos necesarios en cada una de ellas. Se debe además considerar la existencia del camino crítico y su repercusión sobre la duración total de proyecto (o programa) y los posibles retardos, si los hubiere.
Desde el punto de vista metodológico, definir cada uno de los pasos a seguir, con las herramientas involucradas en cada uno de ellos, significa disponer de una manera clara y precisa de una *"hoja de ruta"* para continuar con el proyecto. En la sección 3, ya se ha visto acerca de la necesidad y la importancia de las metodologías, para los desarrollos de software, tal como lo señala Piattini (1996). La realización de un trabajo interdisciplinario hace que tal necesidad sea mucho más importante, ya que se deben realizar tareas en conjunto y en paralelo.
Otra cuestión, es la definición del ciclo de vida más apropiado para cada necesidad. La solución propuesta, es sólo una de tantas posibles, pero las tareas involucradas, a la hora de realizar la matriz de actividades, son las mismas.

7. Propuesta de Evaluación

7.1. Resumen

En el presente capítulo, se desea considerar una propuesta de evaluación de un software que ha sido desarrollado con la metodología expuesta en la sección anterior.
Se destacan aspectos del desarrollo del software educativo (sección 7.2). Se presentan y discuten los resultados de una evaluación incremental realizada a prototipo v1 (sección 7.2.1.), prototipo v2 (sección 7.2.2.) y prototipo vfinal (sección 7.2.3.), adecuando las sucesivas versiones a la evaluación formulada por los potenciales usuarios.
Se describe el desarrollo de la evaluación interna formulada por el equipo de desarrollo (sección 7.3) y la evaluación externa (sección 7.4) realizada por los docentes de los alumnos (potenciales usuarios).
Se presenta finalmente una propuesta de criterios tabulados para medir la calidad del software educativo (sección 7.5).

7.2. Desarrollo

Se tomó un caso: un programa que fuera solicitado por los docentes y alumnos de una carrera de postgrado universitaria no Informática.
Se solicitó para la asignatura "Computación" un software donde pudieran apreciar las partes de la computadora y en especial el funcionamiento interno de la misma, ya que se consideraba que una clase expositiva no era suficiente para las expectativas de los alumnos. Se consideró oportuno realizar un programa en Delphi 3, y para la evaluación del mismo desarrollar varios prototipos con incrementos en las funcionalidades. Para ello, se le solicitó a un programador realizar el programa de acuerdo a la metodología propuesta en la sección anterior, con la ayuda de un especialista en contenidos del área temática. De este modo, se partió de un mapa de conceptos de este tema, como los desarrollados por Novak (1988), en orden conceptual jerárquico o del tipo árbol.
Se presentó la propuesta de evaluación de cada prototipo a los alumnos de una carrera de postgrado en Tecnología, y luego fue seguida de un plan de incorporación de las modificaciones sugeridas.
La preguntas consideraban aspectos de la interface de comunicación y de los contenidos desarrollados, debiendo ser valoradas con una escala de calificaciones entre 1 y a 5 (siendo 5: excelente 4: muy bueno 3: bueno 2: regular 1:malo o 5: muy adecuado 4: bastante 3: poco 2: muy poco 1: nada, de acuerdo al tipo de pregunta), pudiéndose obtener un valor promedio de la calificación. Este valor permite obtener una puntuación de los aspectos tenidos en cuenta, para poder reformular o modificar aquellos que hayan tenido una puntuación menor que 2.5.
En todos los casos de evaluación había un espacio abierto para las sugerencias al cambio o reflexión acerca del programa o de la situación de interacción.
Se evaluaron los dos prototipos (v1 y v2) y el prototipo final (vfinal), el que también se evaluó en forma interna, externa y contextualizada.

7.2.1. Prototipo V1 (versión 1)

Para la evaluación del mismo, se tomó un grupo de 20 alumnos, recogiéndose los resultados cuantitativos en la Tabla que se adjunta en el Anexo I.
Para el primer prototipo, se consideró pertinente presentar un pre-diseño de las pantallas, el menú desplegable y el árbol de contenidos. Las imágenes y los vídeos y el sonido no estaban cargados aún. Se pensó en un diseño de interface del tipo Windows estándar, pero el programa en sí mismo mostraba muy poco Las preguntas realizadas se pueden ver en la Tabla 7.1.

1. ¿Considera adecuado el diseño general de la pantalla?
2. ¿Considera adecuado el uso de las
   Ventanas
   Botones
   Colores
   Tipos de letras?
3. ¿Considera que el programa es interactivo?
4. ¿Considera la interface como amigable?
5. ¿Le da buena información acerca del recorrido?
6. ¿Considera criteriosa la secuenciación de las pantallas?
7. ¿Es de fácil manejo?
8. ¿Considera que el uso de los íconos es correcto?
9. ¿Le resulta útil el uso de teclas rápidas?
10. ¿Ha despertado interés en usted?

Sugerencias de cambio Si- No

Tabla 7.1: Esquema de Evaluación del prototipo v1.

De acuerdo a los resultados, que se pueden resumir cualitativamente: el diseño de la pantalla pareció adecuado, como las ventanas y los botones, pero no así con los colores utilizados y los tipos de letras. La interface pareció fácil de navegar y la secuenciación de las pantallas en general fue considerada como muy buena y de fácil manejo.
No hubo problemas en cuanto a la interactividad y despertó el interés y curiosidad en saber como sería el segundo prototipo del programa, ya con más funcionalidades incorporadas. Otra cuestión a señalar, fue que muchos desconocían la existencia de las teclas rápidas, lo que realmente no les interesaba.
Hubo una pregunta no ponderada y abierta, donde los alumnos que lo creían conveniente debían realizar sugerencias de cambio antes de pasar a una etapa posterior del desarrollo.
Las sugerencias fueron básicamente :

- Usar un tamaño de letra más grande de modo que fuera bien legible en una notebook.
- Cambiar los colores para que hubiera más contraste.
- Cambiar el puntero del mouse cuando se activaba un objeto de la pantalla.

7.2.2. Prototipo V2 (versión 2)

De acuerdo a las preguntas ponderadas y las sugerencias realizadas se pasó al siguiente prototipo incremental con las mejoras pertinentes. Se cargó el glosario, las imágenes, algunos vídeos y la información acerca de cada una de las partes de la computadora. Ahora se podría tener una idea mucho más cercana a lo que sería el programa finalizado.
En este caso las preguntas acerca de la interface comunicación fueron pocas, se precisaba en aspectos relacionados a los contenidos y su pertinencia, se hacía mucho hincapié en la presentación de los mismos, la estructuración y la adecuación a las necesidades del grupo.

1. ¿Considera adecuada la selección de los contenidos?
2. ¿Consideraría adecuado el uso del programa terminado en otros niveles?
3. ¿Los cambios realizados fueron pertinentes?
4. ¿Quisiera que el programa fuera un tutorial?
5. ¿Le facilita la comprensión acerca del tema?
6. ¿Quisiera sonido en los vídeos?

   Sugerencias de cambio Si- No

Tabla 7.2: Esquema de Evaluación del prototipo v2.

En el Anexo II se adjuntan los resultados obtenidos con las ponderaciones y las sugerencias de los alumnos. Cabe destacar que a la mayor parte de los alumnos no les interesó en demasía considerar la realización de un programa del tipo tutorial, por lo que se aprecia que no le interesaba al grupo reemplazar totalmente al docente en sus explicaciones, sino usar el software como material de apoyo al docente y orientado a la ejercitación. Respecto de las sugerencias, quizás la más relevante es que el programa finalizado les permitirá *"ver cosas que no hubieran imaginado"*.

7.2.3. Evaluación del prototipo versión final (vfinal)

Se confeccionó una planilla con preguntas pertinentes a diferentes criterios, tomando como base la utilidad, aspectos pedagógicos y didácticos y técnicos. Las preguntas que se observan en la Tabla 7.2, se deben ponderar como en los casos anteriores.

Utilidad[28]	1.Facilidad de Uso
	2.Grado de adaptación a otros niveles de usuarios.
Pedagógicos y didácticos	3.Claridad de contenidos
	4.Nivel de actualización
	5.Interface de navegación
	6.Nivel de Motivación
	7.¿Es adecuado para la comprensión del tema?
	8.¿Es adecuado para el aprendizaje del tema?
Técnicos	9. ¿Hay documentación y ayudas?
	10.¿Son adecuados los recursos que necesita?

Tabla 7.3: Esquema de Evaluación del Producto Final

Por último, se solicitaron algunas sugerencias a los usuarios, mediante un ítem abierto, sean éstas para el uso del programa o para realizar algún cambio que se considere pertinente. Los resultados obtenidos se adjuntan en el Anexo III. En general, no se solicitaron cambios, y a partir de la evaluación del programa, se deriva que hubo aceptación y acuerdo respecto de los cambios producidos en las etapas anteriores.

7.3. Evaluación interna

El grupo que trabajó en el desarrollo del programa, estuvo de acuerdo con los cambios propuestos por los alumnos. Además consideró la pertinencia de las sugerencias. Es importante destacar que un programa de esta naturaleza debe ser actualizado permanentemente, lo que implica un gran tiempo insumido en actualizar los contenidos.
Además se consideró la propuesta de los alumnos, de usarlo paralelamente a las explicaciones del docente o de usarlo como apoyo a las clases de práctica y entrenamiento.

7.4. Evaluación externa

Se presentó el programa a docentes de una carrera de Ciencias no Informáticas, quienes lo consideraron como una herramienta interesante a la hora de tener que profundizar los conocimientos acerca del tema. Se les proporcionó una planilla similar a las anteriores, con preguntas cerradas y abiertas y se los proveyó del producto terminado.
Los resultados cuantitativos, se pueden ver en el Anexo IV.
Cualitativamente consideraron interesante la propuesta y remarcaron que a veces el grado de dificultad que tienen los usuarios no informáticos para entender cómo funciona la máquina, es muy grande. Otra de las consideraciones realizadas es que desde la escuela primaria se debería alfabetizar en informática, y que habría que pensarlo en lo sucesivo.

7.5. La calidad de los programas de software: un problema interdisciplinario

7.5.1. La propuesta: ¿qué medir en el software educativo?

Considerando que el producto software educativo, de acuerdo a las necesidades de aplicación y a los objetivos educativos perseguidos, requiere de características especiales, y además que éstas sean apropiadas en cuanto a calidad y pertinencia, se elaboró la Tabla 7.4, partiendo del criterio de usabilidad (en el sentido de amigabilidad) y analizando algunos subcriterios. A éstos se los puede calificar de acuerdo a tres niveles pro-

[28] En sentido práctico.

puestos: muy bueno, bueno o malo. Cada nivel tiene una puntuación. Al final de la evaluación el puntaje obtenido, estará entre los que se pueden observar en la Tabla 7.5, donde obviamente, se ve que un programas con una puntuación entre 21 y 30 puntos estará dentro de un nivel de calidad aceptable.
Se ha evaluado el programa de acuerdo a la Tabla 7.4, obteniéndose un promedio de 22.1 para 20 docentes evaluadores como se aprecia en el Anexo V.

7.5.2. La calidad desde la perspectiva pedagógica

Rivera Quijano (1999) considera que el estudiante es el centro del nuevo paradigma educativo denominado "Learner Centered Aproach" (LCA) o "aprendizaje centrado en el estudiante", en muchos casos considerando a éste como un consumidor o cliente inclusive, por duro que parezca.
Como Tecnólogo Educativo, afirma que "la calidad de los proyectos tecnológicos se mide en términos del comportamiento observado al final de la formación". Señala que "el programa de formación debe denotar atributos mensurables y observables en el estudiante, de los contrario, es imposible determinar si el programa logra los objetivos o no".
Considera que para la "Dimarche Scientifique" o método científico en educación, es esencial la humildad y el espíritu de equipo, siendo la humildad para cuestionar nuestras prácticas y nuestras experiencias. (Romiszowski, 1981), proposiciones a las que adhiero.

Criterio: Utilidad (amigabilidad)	**Subcriterio**	**Calificación**	**Puntaje**	**Puntaje obtenido**
Utilidad Externa	Velocidad a de aprendizaje (learnability)	Muy buena	3	
		Buena	2	
		Mala	1	
	Facilidad de uso (operabilidad)	Muy buena	3	
		Buena	2	
		Mala	1	
	Nivel de adicción	Muy buena	3	
		Buena	2	
		Mala	1	
Utilidad Interna	Nivel de legibilidad (Lecturibility)	Muy buena	3	
		Buena	2	
		Mala	1	
	Grado de comprensión	Muy bueno	3	
		Bueno	2	
		Malo	1	
	Estructuras de los manuales	Muy buena	3	
		Buena	2	
		Mala	1	
	Uso de menús, gráficos e imágenes	Muy bueno	3	
		Bueno	2	
		Malo	1	
	Mensajes de errores e información	Muy bueno	3	
		Bueno	2	
		Malo	1	
	Ayudas on–line	Muy buena	3	
		Buena	2	
		Mala	1	
	Definición de adecuación de la interface	Muy buena	3	
		Buena	2	
		Mala	1	
	Puntaje obtenido			

Tabla 7.4: Criterios y subcriterios

Puntaje	**Evaluación de la propuesta**	**Calidad**
1-10	Mala	Inaceptable
11-20	Regular	Dudosa
21-30	Buena	Aceptable

Tabla 7.5: Tabla de puntuación.

7.5.3. Algo más acerca de la evaluación de los programas educativos

La evaluación de los programas educativos es un proceso que consiste en la determinación del grado de adecuación de dichos programas al contexto educativo. Cuando el programa llega al docente, se supone que ha sido analizado y evaluado tanto en sus aspectos pedagógicos y didácticos, como en los técnicos que hacen a la calidad del producto desarrollado según ciertas pautas de garantía de calidad.

Básicamente, se realizan las evaluaciones interna y externa del software, a fin de detectar los problemas que generarán cambios en el producto, lo antes posible, a fin de reducir costos y esfuerzos posteriores. Estas evaluaciones consideran las eventuales modificaciones sugeridas por el equipo de desarrollo y por los usuarios finales, teniéndose en cuenta a docentes y alumnos en el contexto de aprendizaje.

Cuando un producto del tipo comercial educativo, llega al docente, significa que ha superado las etapas de evaluaciones interna y externa. Además para obtener el grado de eficacia y de eficiencia del producto se deberá realizar una evaluación en el contexto de uso.

Es preciso definir ciertos "criterios" para seleccionar un programa como "de acuerdo a las necesidades del docente", y se debe considerar el uso de los vocablos evaluación y valoración que en muchos de los trabajos consultados se usan indistintamente para determinar si un programa dado cumple con los objetivos tanto técnicos como pedagógicos y didácticos para lo que fue pensado

7.5.4. La integración de perspectivas

Rivera Quijano (1999) considera que el punto de partida de las propuestas debe ser el análisis de necesidades entre la situación ideal a la cual arribar y la actual, definida en términos claros y precisos, poniendo énfasis en el diseño pedagógico de las actividades de aprendizaje o sea, partiendo del proyecto mismo y no de las necesidades, para su elaboración.

Considera que el punto más importante de la planificación y el diseño de los productos educativos está en la definición correcta de los objetivos generales y particulares, como la determinación del tipo de los contenidos, la metodología a usar y la evaluación a hacer.

Mediante un diseño didáctico, o programación didáctica, rigurosa y detallada, se puede decidir, cuál es la mejor tecnología que se adapta a cada situación problemática particular. pero los modelos de creación pedagógica, son múltiples, y dependen de las características de cada institución.

"La evaluación es aquella cuyos logros pueden ser observados y medidos. En la medida que un nuevo proyecto permite la incorporación de conocimientos nuevos o de habilidades por parte de los estudiantes, y esos conocimientos pueden ser observados y medidos, esto da la seguridad que el proyecto responde a la necesidad. Este diseño pedagógico es el que permite saber si lo que se está haciendo es educativo o no. (Rivera Quijano, 1999)

Por último, Guitert (1999) afirma que "La existencia de prácticas innovadoras pueden resultar amplificadas por la utilización de tecnologías, pero no suelen ser provocadas por la tecnología misma, ya que no están tan relacionadas con su introducción en el aula, como con la concepción previa que el profesor tenga sobre su propia práctica pedagógica".

Las evaluaciones realizadas dejan entrever varias cuestiones a saber:

- A los alumnos les interesa una interface estandarizada como las comerciales.
- Los productos donde se pueda navegar sin perderse brindan más confianza, la de saber dónde está parados en cada momento.
- La estructuración de los contenidos es un punto fundamental para saber hacia dónde van en los aprendizajes.
- Los alumnos se sienten motivados por participar de los cambios en las etapas de construcción del programa.
- La evaluación de los prototipos sucesivos, permite hacer cambios rápidos, de acuerdo a las sugerencias del usuario o potencial usuario.
- Es importante la buena documentación del programa a fin de hacer dichos cambios y las posteriores actualizaciones lo más eficientemente posible.
- Los usuarios no informáticos necesitan adicionalmente al software, un docente o guía, que les ayude a interpretar los procesos, que vistos en los vídeos e imágenes, y no quieren perder el diálogo fluido, que no tendrían con la máquina, aunque el programa fuera muy interactivo.
- Las evaluaciones son una guía para tener en cuenta, considerando un determinado conjunto de variables.
- Cuando se diseñan aplicaciones tecnológicas, la metodología de desarrollo es uno de los pilares que permitirán lograr acercarse a los objetivos propuestos.

Por otra parte, la calidad en los programas educativos, sigue siendo muy difícil de cuantificar, y las medidas indirectas son resultados parciales y puntuales, pero se pueden tomar como un *"referente parcial",* a partir del cual elaborar la propuesta particular.

Es por ello, que considero de poca utilidad las largas listas de preguntas acerca de una propuesta en particular, para un docente particular, un alumno y un contexto determinado.
Lo que sí es interesante y práctico es la ponderación de algunos indicadores que permitan establecer una calidad "aceptable" a partir de criterios como el de utilidad, vista como "*amigabilidad*" y tomada desde un aspecto interno y externo al programa mismo.
Actualmente, hay una gran necesidad de optimizar los procesos, y particularmente el educativo viendo al alumno como el "*cliente*", y desde este aspecto no sólo se debe lograr la satisfacción, sino un acercamiento a un perfil ideal que el estudiante deberá tener luego de un cierto "*entrenamiento*".
La experiencia dice, que se puede realizar el programa educativo, con las herramientas más novedosas y mejores, pero, en realidad, el indicador más preciso se obtiene cuando el estudiante finaliza el ciclo o período y adquiere el conjunto de habilidades y conocimientos previstos, o cuando se incorpora al sistema productivo. Estos son finalmente los indicadores válidos que permitirán dar una idea de la calidad o bondad de al propuesta.

8. Evaluación contextualizada

8.1. Resumen

Se presenta la evaluación de un software educativo en un contexto similar a aquel para el cual fuera creado el programa. Los resultados de este tipo de evaluación se consideran como los más representativos ya que dan cuenta de las reacciones de los potenciales usuarios ante el programa y dan cuenta de la eficacia del producto. (Fainholc, 1994).
Para ello se tienen en cuenta las variables involucradas en el proceso de enseñanza y de aprendizaje tales como el docente y estilo docente, tipo de alumnos destinatarios, el tiempo y modo de uso del software, el currículum, entre otras.
Se formulan y se describen las etapas preparatorias (sección 8.2) y, posteriormente se presentan las experiencias realizadas a fin de establecer las diferencias en cuanto a logro de aprendizajes significativos entre un software desarrollado con una metodología extendida para cautelar los aspectos pedagógicos, partiendo de un paradigma clásico de la ingeniería de software tal como se describe en la sección 6 y uno de idéntica funcionalidad desarrollado con una metodología estándar (sección 8.3).
Para ello, se formaron dos grupos equilibrados[29] mediante la definición de pares homólogos: uno de control, llamado A y otro experimental ó B. Para definir grupos equilibrados, se partió de la aplicación del test de matrices progresivas de Raven[30] a los sujetos.
Ambos grupos, en conjunto recibieron la misma instrucción acerca de los aspectos teóricos, mediante clases expositivas, siendo el tema desarrollado: el funcionamiento interno de una computadora personal. Luego, al grupo de control "A" se le mostró aspectos inherentes a la lógica de funcionamiento mediante un software desarrollado con una metodología que no contempla los aspectos pedagógicos y al grupo experimental "B", mediante un software desarrollado con la metodología propuesta extendida. Los software señalados fueron desarrollados por equipos de implementación diferentes.
Una vez realizadas las experiencias, se verificó el rendimiento de los alumnos mediante la misma evaluación para los dos grupos, se aplicó un test estadístico de comparación para muestras pequeñas de Wilcoxon, obteniéndose las conclusiones que se enuncian.

8.2. Formulación de la tesis y etapas preparatorias

La tesis a demostrar experimentalmente es:

> *"El software educativo desarrollado con una metodología que contempla aspectos psicopedagógicos en el modelo de ciclo de vida del software permite un mejor aprendizaje de los conceptos que un software que ha sido desarrollado con una metodología que no los contempla".*

Para operacionalizar esta tesis se desarrollaron las siguientes etapas preparatorias:

ETAPA I: Se tomó un curso de Computación de un postgrado, no informático, donde a los alumnos se les debía instruir acerca del funcionamiento de una computadora personal. Para la experiencia, se tomó el tema específico de la estructura de una computadora, sus partes principales y el funcionamiento las unidades componentes.

[29] Se parte del supuesto que los pares homólogos requeridos para el aprareamiento de Wilcoxon, tendrán una respuesta similar ante los nuevos aprendizajes. Se poría haber usado en lugar de la puntuación del Test de una prueba de evaluación diagñóstica diseñada específicamente.
[30] Raven J. C. (1970), *Test de Matrices Progresivas*. Escala General. Vol.3. Paidós.

ETAPA II: Mediante la aplicación del Test de Raven de Matrices Progresivas, se formaron pares de homólogos con igual puntuación en dicho test[31], como se observa en la Tabla 8.1. Se formaron dos grupos: uno de control "A" y otro experimental "B".

ETAPA III: A ambos grupos en conjunto se les explicó el tema en sus aspectos teóricos, de modo tradicional, mediante una clase expositiva. Luego, el grupo A se ejercitó con un software desarrollado sin un metodología de que cautelara los aspectos pedagógicos y el grupo B utilizó un software desarrollado con la metodología propuesta. Las actividades desarrolladas por los grupos se resumen en la Tabla 8.2, resaltándose las diferencias.

Grupo A		**Grupo B**	
Alumno	**Puntuación (%)**	**Alumno**	**Puntuación (%)**
Paloma	9.83	Enrique	9.83
Javier	9.79	Carlos	9.79
Susana	9.72	Silvia	9.72
Carola	9.66	Cristina	9.66
Miguel	9.61	Guillermo	9.61
Mónica	9.58	Luis	9.58
Favio	9.5	Gustavo	9.5
Alejandro	9.33	Marcos	9.33
José	9	Ada	9

Tabla 8.1: Pares homólogos formados de acuerdo al Test de Raven

Actividades	**Grupo A**	**Grupo B**
Aspectos Teóricos Clase Tradicional Magistral Expositiva	Explicación del funcionamiento de una PC y de sus partes	
	Se usaron dibujos en el pizarrón	
	Se usaron transparencias ilustrativas.	
Aspectos Prácticos Ejercitación	Explicación del funcionamiento de una PC y de sus partes	
	Se usó un software multimedia del tema, desarrollado con metodología que no contemplaba los aspectos pedagógicos.	Se usó un programa confeccionado con la metodología propuesta extendida.
	Ambos programas tenían el mismo conjunto de imágenes y vídeos. Partían del mismo árbol de conceptos, pero diferían en el diseño y secuenciación.	
Las clases fueron dictadas por el mismo docente		
Los dos grupos fueron evaluados con el mismo conjunto de preguntas.		

Tabla 8.2: Actividades de los dos grupos.

Con estas tres etapas concluidas se estuvo en condiciones de operacionalizar la tesis inicial reformulándola de la siguiente manera:

"Siendo los valores de las variables independientes los mismos para ambos grupos de alumnos, salvo la variable: 'programa utilizado para la ejercitación'; el grupo de alumnos que trabaje con el programa desarrollado con la metodología propuesta (grupo B), debe tener un mejor rendimiento que el grupo de alumnos que utilice el programa desarrollado con la metodología convencional (grupo A)"

8.3. Desarrollo de la experiencia y valuación estadística

Al finalizar la ejercitación ambos grupos fueron sometidos a la misma prueba, confeccionada específicamente, siendo los resultados obtenidos los que se presentan en la Tabla 8.3:

Grupo A		**Grupo B**	
Alumno	**Nota**	**Alumno**	**Nota**
Paloma	6	Enrique	10
Javier	7	Carlos	9
Susana	6	Silvia	10
Carola	8	Cristina	7

[31] De la totalidad de los alumnos del curso se tomaron los nueve pares homólogos, que se presentan en la tabla 8.1 seleccionados con igual puntuación.

Miguel	7	Guillermo	8
Mónica	7	Luis	7
Favio	8	Gustavo	8
Alejandro	7	Marcos	8
José	8	Ada	10

Tabla 8.3: Tabla comparativa del rendimiento obtenido en la prueba.

Aplicado el test de Wilcoxon (Ledesma, 1980), a los grupos A y B, habiendo tomado al A como grupo de control, *se espera que el grupo B tenga un mejor rendimiento. Como la única diferencia, sería el software con el cual se desarrolló la ejercitación, el hecho de estar trabajando con pares homólogos, permitiría concluir que de existir una diferencia esta se deba al software, en particular a la metodología aplicada para su desarrollo*.

Grupo A		**Grupo B**		$\mathbf{D_{A-B}}$
Alumno	**Nota**	**Alumno**	**Nota**	
Paloma	6	Enrique	10	-4
Javier	7	Carlos	9	-2
Susana	6	Silvia	10	-4
Carola	8	Cristina	7	1
Miguel	7	Guillermo	8	-1
Mónica	7	Luis	7	0
Favio	8	Gustavo	8	0
Alejandro	7	Marcos	8	-1
José	8	Ada	10	-2

Tabla 8.4: cálculo de las diferencias $D_{A-B.}$

El primer paso del test de Wilcoxon (Ledesma, 1980), consiste en realizar la diferencia de calificaciones entre ambos grupos: En la Tabla 8.4 se puede observar en la última columna la diferencia $D_{A-B.}$ Como indica el método de Wilcoxon se procede al ordenamiento por valor absoluto de las diferencias como se ve en la Tabla 8.5.

1
-1
-1
-2
-2
-4
-4

Tabla 8.5: Ordenamiento de las diferencias.

Luego, se le asignan los números de orden a cada valor y en el caso de valores con valor absoluto igual se promedian las posiciones, tal como se observa en la Tabla 8.6.

1	1	2
-1	2	2
-1	3	2
-2	4	4.5
-2	5	4.5
-4	6	6.5
-4	7	6.5

Tabla 8.6: Obtención de los números de orden.

Finalmente, se suman los números de orden de las diferencias negativas tal como se aprecia en la Tabla 8.7.

-1	2
-1	2
-2	4.5
-2	4.5
-4	6.5

-4	6.5
SUMA	***26***

Tabla 8.7: Suma de las diferencias negativas.

Según la Tabla 10 (ver Tabla 8.9) del apéndice del libro de Ledesma de Estadística Médica (1980) y el Manual de al Universidad de Málaga de Bioestadística (1999) para los niveles significación de 5% donde $\mathbf{2\,\alpha \leq 0{,}05}$ (siendo α la probabilidad de error de primer orden) y para un número de muestras **n = 7** (en este caso el número de pares homólogos cuyas diferencias $\mathbf{D_{A-B}}$ sean diferentes a cero) se puede observar que:

n: Número de pares	$\mathbf{2\,\alpha \leq 0{,}05}$
6	0-21
7	3-26

Tabla 8.9: de Wilcoxon para muestras continuas de pares homólogos

la suma de los números de orden de las seis observaciones negativas cae fuera de los límites tabulados. Y, como si: *"o bien coincide con uno de los límites del intervalo de significatividad o está fuera de dichos límites, la diferencia es significativa",* (descartándose entonces la hipótesis nula de contraste), se puede decir que ***la diferencia*** entre el método aplicado al grupo B y al grupo A ***es significativa a favor de B*****,** con lo que experimentalmente se confirma la tesis:

"Los alumnos que trabajaron con el programa desarrollado con la metodología extendida (grupo B), tienen un mejor rendimiento que los alumnos que utilizaron el programa desarrollado con la metodología que no considera los aspectos pedagógicos en su diseño (grupo A)".

Desde esta perspectiva queda demostrada experimentalmente la tesis central:

"El software educativo desarrollado con una metodología que contempla aspectos psicopedagógicos en el modelo de ciclo de vida permite un mejor aprendizaje de los conceptos que un software que ha sido desarrollado con una metodología que no los contempla".

Los resultados de la experiencia apoyan la metodología propuesta en esta tesis, con independencia de los resultados satisfactorios creemos que la misma está sujeta a replicaciones posteriores, debido a que fue restringida.

Desde la perspectiva tecnológica, se ha insistido en el hecho, de que el nivel de estructuración y el grado de adecuación de los contenidos es función directa del rendimiento esperado por los alumnos, la comprensión del material educativo depende fundamentalmente de la organización y estructuración de los contenidos del mismo. (Pozo Municio, 1998; Fainholc, 1998)

Esta coherencia interna, se logra mediante un desarrollo metódico, que permite realizar las conexiones lógicas y conceptuales entre los elementos. Esta información organizada, dice Pozo Municio (1998), se parece a un árbol de conocimientos, en el que se pueden establecer relaciones diversas entre ellos y recorrer diferentes rutas para recuperar el conocimiento y mediante la comprensión de la misma se podrá *"reconstruir"* o *"traducir el material"* a las palabras propias del aprendiz.

Por este motivo, la producción de programas educativos, no es una tarea sencilla, sino que se debe aplicar la secuencia metodológica que provee básicamente la ingeniería de software, sea en la elección de un ciclo de vida para el desarrollo de los productos, por un lado o en el uso de las técnicas, y herramientas por el otro, pero deben estar articuladas dinámicamente con una teoría acerca de los aprendizajes y de los niveles de representación de los alumnos cuando estos necesiten interpretar los fenómenos físicos.

Desde la perspectiva pedagógica, a fin de mejorar las prácticas educativas, los docentes investigadores (Gutiérrez, 1997) han basado sus estudios en las teorías del aprendizaje, pero en muchos casos las teorías solamente, si bien dan una respuesta válida para llegar a una eventual solución, no conducen al cierre definitivo del problema.

Es por ello, que se ha tomado como eje central para el análisis de la experiencia, algunos de los campos disciplinares[32] de la *"Ciencia Cognitiva"*[33].

[32] Los campos disciplinares que confluyen y contribuyen son: la psicología cognitiva, la neurociencia, la lingüística, la ciencia de la computación y la filosofía.

[33] Gardner (1988) dice: "*defino la ciencia cognitiva como un empeño contemporáneo de base empírica por responder a interrogantes epistemológicos de antigua data, en particular los vinculados a la naturaleza del conocimiento, sus elementos componentes, sus fuentes, evolución y difusión. Aunque a veces la expresión ciencia cognitiva se hace ex-*

A fin de acotar el problema se ha tomado la teoría de Jonhson-Laird (1998) dentro de la Psicología Cognitiva, para fundamentar los resultados experimentales, desde la perspectiva de las representaciones mentales.
Para los alumnos es importante entender los *"fenómenos",* saber qué los causa y sus consecuencias, cómo replicarlos y darles fin; esto significa tener un *"modelo de funcionamiento"* o *"de trabajo"*.
Para intentar explicar los altos rendimientos, en general de los alumnos de ambos grupos, habría que considerar que *"han podido formar modelos mentales"*, sea proposicionales o por analogía[34], y sus imágenes mentales fueron las que les permitieron, interpretar los procesos o fenómenos. Los alumnos de estos grupos, no sólo entienden claramente los conceptos, y los pueden transferir de y hacia otras asignaturas, aunque algunos sólo utilicen las fórmulas y/o las definiciones y otros adapten los modelos del material de estudio. El rendimiento del grupo que *"forma imágenes mentales"*, es superior a aquel que no lo hace.
Por otra parte, Perkins (1995) habla acerca de la conexión importante que existe entre la *pedagogía de la comprensión* (o el arte de enseñar a comprender) *y las imágenes mentales,* por lo que se puede decir que la relación es bilateral.
Esta relación recíproca existente es la que puede ayudar al alumno a adquirir y elaborar imágenes mentales, con lo cual desarrolla su capacidad de comprensión y al exigirles actividades de comprensión (como por ejemplo: predecir, explicar, resolver, ejemplificar, generalizar) se hará que construyan imágenes mentales, para lo que afirma Perkins que: *"se alimentan unas a otras como si fueran el Yin y el Yan de la comprensión"*.
La detección de las *"representaciones mentales"* de los alumnos, no es el tema central de este estudio, pero la construcción de "*modelos*" implica un aprendizaje significativo de conceptos. Esta construcción se vería ampliamente favorecida con el uso de materiales de estudio mutimediales, especialmente con vídeos y con una secuenciación adecuada y lógica, como se deriva de los resultados obtenidos por el grupo experimental.

tensiva a todas las formas del conocimiento, (...) yo la aplicaré principalmente a los esfuerzos por explicar el conocimiento humano".

[34] Jonhson-Laird (1998) plantea que existen al menos tres formas en la que se puede codificar mentalmente para representar información; las representaciones proposicionales, los modelos mentales y las imágenes, sean estas auditivas, visuales o táctiles.

Conclusiones

Como conclusiones finales del trabajo realizado, se puede puntualizar, que el software educativo, es uno de los pilares en que se sostiene, del sistema educativo a distancia y, como material de aprendizaje, su comprensión depende fundamentalmente de la organización y estructuración de los contenidos del mismo.
Esta coherencia interna, se logra mediante un desarrollo metódico, que permite realizar las conexiones lógicas y conceptuales entre los elementos. Esta información organizada, dice Pozo Municio (1998), se parece a un árbol de conocimientos, en el que se pueden establecer relaciones diversas entre ellos y recorrer diferentes rutas para recuperar el conocimiento y mediante la comprensión de la misma se podrá *"reconstruir"* o *"traducir el material"* a las palabras propias del aprendiz.
Recordando a Freire (1997), en *"Pedagogía de la autonomía"*, quien dice que *"No temo decir que carece de validez la enseñanza que no resulta en un aprendizaje, en que el aprendiz no se volvió capaz de recrear o de rehacer lo enseñado, en que lo enseñado que no fue aprehendido no pudo ser realmente aprendido por el aprendiz",* se puede pensar en el desarrollo de los materiales para formar sujetos autónomos, más que autodidactas.

1. Aportes del presente trabajo

1. Se ha presentado un estudio crítico del estado de los desarrollos de los programas de software educativos de modo diacrónico, en paralelo con las diferentes teorías y líneas educativas, desde su aparición hasta la actualidad.
2. Se ha visto, que la situación actual se complejiza, en tanto existe una gran cantidad de lenguajes de programación que posibilitan diferentes alternativas de desarrollo, como así también, los avances en cuanto a la tecnología informática, que permiten utilizar recursos impensados una década atrás.
3. Se han detectado las problemáticas concernientes al diseño, desarrollo y evaluación de los programas educativos y se ha propuesto una de las posibles metodologías, como solución a dichas necesidades.
4. Se ha tomado un software desarrollado con las características propuestas y se mostrado experimentalmente, que los alumnos que lo usaron, obtiene un rendimiento notable respecto de otros productos que existen en el mercado.
5. Se ha tenido en cuenta las opiniones del grupo de alumnos "evaluadores" del programa, que fueron los que probaron los prototipos e hicieron las sugerencias que como se ve en las Tablas que se adjuntan en los Anexos I a IV.
6. Durante el proceso de investigación se detectaron líneas de investigación derivadas del problema central tratado que serán enumeradas en el epígrafe siguiente.

2. Líneas de trabajo futuras.

A partir de las problemáticas detectadas y de los resultados obtenidos se pueden derivar las siguientes líneas de trabajo:

1. Desarrollo de otras metodologías basadas en diferentes combinaciones de ciclos de vida del software y de teorías de aprendizaje, estableciendo así comparaciones con la solución propuesta en este trabajo y con ponderación de resultados.
2. Es en este punto, en el que se ha pensado, partiendo de la necesidad detectada en alumnos y docentes, en desarrollar un buscador de Internet para este sector particular, que es el EGB, y por cierto, en castellano y para necesidades particulares.
3. Otra de las posibles líneas de trabajo, que se deriva al respecto del los programas educativos, es la adaptación de los programas de mercado y/o desarrollo de nuevos programas a las necesidades específicas de algunos grupos de estudiantes, como los hipoacúsicos, en sus distintos niveles, ya que se ha detectado, que el software desarrollado para estos grupos con necesidades especiales, es escaso y caro.
4. Un línea de trabajo de fondo en los desarrollos con base informática sería, el nivel de infoalbabetización de los docentes que desean aplicar y/o diseñar programas didácticos, como así también el caso inverso de los programadores que deseen hacerlos.
5. Por último, se ha dejado un tema central, de las investigaciones didácticas actuales, respecto de la influencia del estilo docente en los procesos educativos y, en este caso en particular en que deben aplicar la herramienta informática y los productos lógicos.

Anexos

Anexo I: Planilla de evaluación de la interface de comunicación. Prototipo v1

Calificación de 1 a 5 (5: excelente 4: muy bueno 3: bueno 2: regular 1: malo o 5: muy adecuado 4: bastante 3: poco 2: muy poco 1: nada)																						
Número de orden		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	Prom
1. Considera adecuado el diseño general de la pantalla?		4	4	3	3	3	3	3	3	3	3	4	4	3	4	4	3	3	4	4	3	3.4
2. Considera adecuado el uso de las	Ventanas	5	4	4	4	4	4	4	4	4	4	4	4	4	4	4	3	4	3	4	4	3.95
	Botones	4	4	3	4	4	4	3	4						3	3	4	3	4	4	4	2.33
	Colores	4	1	2	2	2	2	2	2	2	3	3	2	2	3	3	2	2	2	2	2	2.25
	Tipos de letras?	3	4	2	2	4	4	3	4	3	2	3	3	3	3	3	3	2	2	2	2	2.65
3. Considera que el programa es interactivo?		5	4	4	4	2	3	2	3	4	3	4	4	4	3	3	3	4	4	4	4	3.55
4. Considera la interface como amigable?		4	4	4	3	4	4	4	4	4	4	4	4	4	4	4	3	4	3	3	3	3.6
5. Le da buena información acerca del recorrido?		5	5	4	5	4	4	4	5	4	4	4	4	4	4	4	4	4	5	5	5	4.15
6. Considera criteriosa la secuenciación de las pantallas?		4	4	4	3	4	4	4	4	5	4	5	5	5	4	4	4	3	3	3	4	4.0
7. Es de fácil manejo?		4	5	5	4	5	5	5	4	5	4	5	5	4	5	4	5	5	5	5	5	4.7
8. Considera que el uso de los íconos es correcto?		4	5	4	5	5	5	5	4	4	3	4	5	4	4	4	4	4	4	4	4	4.25
9. Le resulta útil el uso de teclas rápidas?		3	3	3	5	-	-	-	4	-	-	-	-	-	-	-	-	3	-	4	4	3.62
10. Ha despertado interés en usted?		4	4	3	5	3	4	3	4	4	4	4	4	4	5	5	4	4	3	4	4	3.95
Sugerencias de cambio Si- No		N	S	S	S	S	S	S	N			S	S		S	S	S	S	S	S	S	-

Número de orden	**Sugerencias de cambio**
1	Cambiar los colores en pantalla para que resulte más atractivo
2	Para evaluar mejor tendría que estar más completo, sugiero mayor colorido y que el indicador sea distinto. Es decir no flecha sino mano.
3	Cuando se posiciona en algo que se expande que aparezca otro apuntador como la mano, y que a la vez haya cambio de relieve. Ponerle sonido al momento de activar algo. Más vistoso, más atractivo.
4	Cambio de colores - más contraste
5	Interconectar los elementos por medio de dibujos que representen cables en la pantalla de presentación. Tipo diagrama de flujo
6	Cambiar la flecha de indicación por la manito
10	Tamaño de letra más grande para usar en la notebook.
15	
16	Cambiar los colores de pantalla para que resulte más contraste.
	Hace falta un icono de retorno en la pantalla. Tamaño y tipo de letra
18	Sería bueno agregar algún tapiz de fondo por que parece que los elementos en la pantalla están flotando?
	Las letras de los botones son muy débiles, les falta fuerza.
19	Colocar el icono de “volver” fijo en la pantalla para retornar a ventanas anteriores.
20	Mejor color de pantalla, letra más gruesa, crear subventanas u opciones de algunos temas

Anexo II: Evaluación de contenidos y pertinencia. Prototipo v2

Calificación de 1 a 5 (5: excelente 4: muy bueno 3: bueno 2: regular 1: malo o 5: muy adecuado 4: bastante 3: poco 2: muy poco 1: nada)																					
Número de orden	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	Prom
1.¿Considera adecuada la selección de los contenidos?	4	5	3	5	4	5	4	3	5	3	4	4	4	5	4	5	3	4	4	4	4.1
2.¿Consideraría adecuado el uso del programa terminado en otros niveles?	4		5	5	5	5	4	4	4	4	4	5	5	4	4	4	4	4	5	5	4.42
3.¿Los cambios desde realizados fueron pertinentes?	4	4	4	4	4	4	4	5	5	5	5	5	4	4	4	3	3	4	4	4	4.15
4.¿Quisiera que el programa fuera un tutorial?	2	3	4	3	3	3	2	2	2	3	3	3	3	3	3	3	3	3	4	3	2.9
5. ¿Le facilita la comprensión acerca del tema?	4	4	4	4	5	5	4	4	5	5	4	4	4	4	4	4	4	5	5	5	4.35
6. ¿Quisiera sonido en los vídeos?	3	3	3	4	4	3	3	3	3	4	4	4	3	3	3	3	3	3	3	3	3.75
Sugerencias de cambio Si- No	-	S	-	S	S	-	S	-	S	-	S	-	S	-	-	-	-	-	-	-	-

Número de orden	**Sugerencias de cambio**
1	Poner las direcciones web y la bibliografía en las pantallas explicativas.
2	Ahora puedo evaluar mejor la capacidad del programa y me lo imagino terminado
3	Sugiero que no se le ponga audio, ya que me gusta tener al docente que me explique lo que pasa en el video y así puedo hacerle preguntas. Sólo lo usaría para los eventos.
6 10 16	Considero que los cambios estuvieron bien y que respeten en los programas el formato de Windows
	Los colores siguen siendo muy pálidos
	El problema es que hay que actualizarlo constantemente.
19	Me parece bueno el programa porque me permite ver cosas que no me hubiera imaginado y que desconocía

Anexo III: Evaluación Prototipo versión final (vfinal)

Calificación de 1 a 5 (5: excelente 4: muy bueno/a 3: bueno/a 2: regular 1: malo/a o 5: muy adecuado 4: bastante 3: poco 2: muy poco 1: nada)																						
Aspecto	**Número de orden**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	Prom
Utilidad	1. Facilidad de Uso	5	5	5	5	4	5	4	5	5	5	5	4	4	4	4	4	4	4	4	4	4.75
	2. Grado de adaptación a otros niveles de usuarios.	4	5	4	5	4	5	5	5	5	5	4	5	5	4	4	4	5	5	5	5	4.65
Pedagógicos y didácticos	3. Claridad de contenidos	4	4	4	3	3	3	4	4	3	4	3	3	3	3	3	4	4	4	4	4	3.55
	4. Nivel de actualización	5	5	5	5	5	5	4	4	5	5	5	4	4	4	4	4	4	4	4	4	4.45
	5. Interface de navegación	4	4	4	4	3	3	3	3	4	4	3	3	3	3	3	3	3	3	3	3	3.3
	6. Nivel de Motivación	4	4	4	3	3	3	3	4	4	4	4	4	4	4	4	4	3	3	3	3	3.8
	7. ¿Es adecuado para la comprensión del tema?	3	5	5	5	4	3	3	3	4	5	5	5	3	4	4	3	3	3	3	4	3.85
	8. ¿Es adecuado para el aprendizaje del tema?	3	5	5	5	3	3	3	4	4	4	3	5	5	5	4	4	4	3	3	3	3.9
Técnicos	9. Documentación y ayudas	4	4	4	4	5	5	5	4	4	3	5	5	5	4	4	4	3	3	3	3	4.05
	10. ¿Son adecuados los recursos que necesita?	5	5	5	5	5	3	3	4	5	5	5	5	5	5	5	3	3	3	3	4	4.1
Sugerencias																						

Anexo IV: Evaluación Externa de Producto Final

Calificación de 1 a 5 (5: excelente 4: muy bueno 3: bueno 2: regular 1: malo o 5: muy adecuado 4: bastante 3: poco 2: muy poco 1: nada)																						
Número de orden		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	Prom
1. ¿Considera adecuado el diseño general de la pantalla?		4	4	4	4	4	3	4	4	5	5	5	5	5	5	5	5	5	5	5	5	4.55
2. ¿Considera adecuado el uso de	Ventanas	3	3	4	4	4	4	4	4	5	5	5	5	5	4	5	4	5	4	5	4	4.3
	Botones	4	4	4	4	4	4	4	5	5	5	5	5	5	4	4	4	4	4	4	4	4.3
	Colores	3	3	3	3	4	4	4	4	4	3	3	3	3	4	4	4	4	4	4	4	3.6
	Tipos de letras?	4	4	4	4	4	3	3	3	4	3	3	4	4	4	4	4	4	4	4	4	3.75
3. ¿Considera que el programa es interactivo?		4	4	4	4	4	4	4	4	5	5	4	4	4	4	4	4	4	4	4	4	4.1
4. ¿Considera la interface como amigable?		4	4	4	4	4	4	5	5	5	4	4	5	5	4	4	4	4	4	4	4	4.25
5. ¿Le da buena información acerca del recorrido?		5	5	5	5	4	4	4	4	5	5	5	5	5	5	5	5	5	5	5	5	4.8
6. ¿Considera criteriosa la secuenciación de las pantallas?		4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
7. ¿Es de fácil manejo?		5	5	5	5	4	4	5	5	5	4	4	4	4	4	5	5	5	5	5	5	4.25
8. ¿Considera que el uso de los íconos es correcto?		5	5	5	5	5	4	4	4	4	4	4	5	5	5	5	5	5	5	5	5	4.7
9. ¿Le resulta útil el uso de teclas rápidas?		3	3	3	3	2			2	2	2				2	2		2	2	2		2.3
10. ¿Considera adecuada la selección de los contenidos?		4	4	4	4	4	4	5	5	5	4	4	4	4	4	5	4	5	4	5	4	4.05
11. ¿Consideraría adecuado el uso del programa terminado en otros niveles?		4	4	4	4	4	4	4	4	4	4	4	5	5	4	5	4	5	5	4	4	4
12. ¿Quisiera que el programa fuera un tutorial?		4	4	4	-	-	-	3	3	3	-	-	-	-	-	3	3	3	-	-	-	3.33
13. ¿Le facilita la comprensión acerca del tema?		4	4	4	4	3	3	3	3	5	5	5	5	5	5	5	5	5	5	5	5	4.15
14. ¿Quisiera sonido en los vídeos?		5	2	3	4	4	3	3	2	2	2	2	3	3	-	-	3	-	-	3	3	2.37
15. ¿Ha despertado interés en usted?		5	5	5	5	5	4	4	4	4	3	3	3	-	5	5	5	-	5	-	5	4.41
Sugerencias de cambio Si- No		S	-	S	-	-	-	S	-	-	-	S	-	-	S	-	-	S	-	-	S	-

Número de orden	**Sugerencias de Cambio**	**Número de orden**	**Sugerencias de Cambio**
1	Los colores más fuertes.	12	No me interesan las teclas rápidas.
5	Las letras más grandes	14	Habría que poner más ventanas.
10	Habría que considerar usarlo en otros cursos	15	Me vino muy bien por que yo no sabía nada
		16	Quizás sería bueno, proyectarlo mientras el docente explica

Anexo V: Aplicación de Criterios y Subcriterios de Calidad (ver Tabla 7.5)

Número de orden	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	Prom.
Puntaje obtenido	20	21	23	20	23	20	24	25	20	21	23	22	22	22	23	21	17	24	26	25	22.1

Referencias Bibliográficas

AACE. Society for Information Technology and Teacher Education. www.AACE.org

Aenor (1992): *Normas para la gestión y el aseguramiento de la calidad*. Madrid.

Akahori Kenji (1985): *Evaluation of Educational Computers Software in Japan (I y II): Methods and results*. PLET, vol. 25, 46-66.

Alcalde E., García M. y Peñuelas S. (1988): *Informática Básica*. Mc Graw Hill.

Alessi S. M. y Trollip S. R. (1985): *Computer-based instruction. Methods and Development.* Prentice Hall. Nueva Jersey.

Alonso, C. M. (1992): *Estilos de aprendizaje y tecnologías de la información*. Conferencia europea sobre tecnología de la Información. Barcelona. 3-6 noviembre.

Amler (1994): citado por Piattini M.(1996): *Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión.* Rama. Madrid.

Ander Egg, Ezequiel (1986): *Acerca del pensar científico. Humanitas*. Buenos Aires.

Aspillaga M. (1991): *Para un diseño efectivo de presentación de la información en la computadora,* Revista de Tecnología Educativa, XI, 4, 307-323.

Ausubel D., Novak J. y Hanesian H.(1978): *Psicología educativa. Un punto de vista cognitivo.* Trillas. Ediciones 1978, 1997.

Basili V. y Rombach H. (1988): *The TAME project: towards improvement -orientated software enviroments.* IEEE Transactions on Software Engineering, vol. 14, número 6, págs. 758-773.

Baumgartner P. y Payr S. (1996): *Learning as action: A social science approach to the evaluation of interactive media*. Universities of Innsbruk and Klagenfurt. Educational Multimedia and Hipermedia, AACE, Charlottesville, V.A. www.webcom.com/journal/baumgart.html

Bender, Richard, (1996): *Proposed software evaluation and test KPA*. Bender and Associated Inc. Position Papers, White Paper, April.

Benett (1996): *Computers as tutors: Solving a crisis in education.* www.cris.com/~faben1/

Blease (1986): *Evaluating Educational Software*. Londres. Croom Helm, citado en Squires y Mc Dougall (1994).

Bloom B, et al. (1956): *Taxonomy of educational objectives. The classification of educational goals.* David McKay. N. Y.

Boehm B. (1978): *Characteristics of Software Quality*. Nueva York. North Holland.

Boehm B. (1981): *Software Engineering Economics,* Englewood Clifs, Nueva Jersey.

Boehm B. (1988): *A spiral model of software development and enhancement.* Computer 1988 IEEE págs. 61-72.

Bolívar A. (1995): *La evaluación de valores y actitudes*. Madrid, Anaya, citado en Castillo Segurado (1997)

Booch G. (1991): *Object Oriented Design with applications*. Redwood City. Benjamin Cummings Publishing

Bork A. (1986): *El ordenador en la enseñanza. Análisis y perspectivas de futuro,* Barcelona, Gustavo Gili.

Bruner J. (1988): *Desarrollo cognitivo y educación*. Morata. Madrid.

Bruner J. (1991): *Actos de significado. Más allá de la revolución cognitiva*: Madrid. Alianza.

Bunes y otros (1993): *Los valores del LOGSE*. Un análisis de documentos a través de la metodología de Hall-Tonna, Bilbao. ICE, Universidad de Deusto, citado en Castillo Segurado (1997)

Burnstein I. et al. (1996): *Developing a Testing Madurity Model. Part I,* Crosstalk, STSC, Hill Air Force Base, Utah, Agosto.

Cabero J. (1993): citado en Sancho J. (coord.) (1994): *Para una Tecnología Educativa.* Editorial Horsori. Barcelona. España. pág. 255.

Cabero Almenara J. (1992): *Diseño de Software Informático.* Universidad de Sevilla. Bordón, 44,4 383-391 ISSN: 0210-5934

Caftori N. y Paprzycki M. (1997): *The design, evaluation and usage of educational software* en Price J. D., Rosa K, Mc Neil S. Y Willis J. Editores (1997): Technology and Teacher Education Annual. Association for el Advancement of Computing Education, Charlottesville, V.A. Campos F. et al. (1996): *Dez etapas o desenvolvimiento de software educacional do tipo hipermidia:* Memorias del Tercer Congreso Iberoamericano de Informática Educativa de Barranquilla, Colombia citado en Sanchez J. y Alonso O.

Castillo Segurado (1997): *Un ejemplo de evaluación de software educativo multimedia.* Edutec 97. Comunicaciones: Formación y recursos.

Castorina J. A. (1989): *La posición del objeto en el desarrollo del conocimiento,* en Castorina *et al*. *Problemas de la psicología genética***.** Buenos Aires. Miño y Dávila Eds.

Cediproe, (1998): *Actividades para el logro de la Comprensión*. Material de trabajo.

Clarke P. y Peté M. (1996): *The KwaZulu concept burger: A hypertext concept map of educational software evaluation. AACE Site´97. Consultado el 20/06/99 a las 22 horas. www.AACE.org/pubs/elec_pub/th_char.htm*

Coburn P., Kelman P. et al. (1985): *Practical guide to computers in Education*, Reading Massachusetts. Addison Wesley, citado en Squires y Mc Dougall (1994).
Coll César (1994): *Psicología y Curriculum*. Paidós.
Crosby P. (1979): *La calidad no cuesta*. Mc Graw Hill. México
Cruz Feliú, Jaime (1986): *Teorías del Aprendizaje y Tecnología de la Enseñanza,* Trillas.
Del Moral M. E. (1998): citada por Marquès (1998b): *Programas didácticos: diseño y evaluación*. Universidad Autónoma de Barcelona. Consultado en octubre de 1998. www.doe.d5.ub.es/te
DeMarco T. (1979): *Structured analysis and systems specifications*. Prentice Hall.
Deming W. E. (1982): *Out of the crisis*. Cambridge University Press.
Deterline W. A. (1969): *Introducción a la Enseñanza Programada*, Buenos Aires Troquel.
Doll C. A. (1987): *Evaluating Educational software*. Chicago-London: American Library Association.
Dorado Carlos (1998): Citado por Marquès (1998) y comunicación vía e-mail del 26 mayo de 1999, cdorado@pie.xtec.es
EDUCOM (1989): *Software snapshots: Where are you in the picture?,* Washington D. C. EDUCOM, citado en Squires y Mc Dougall (1994).
Eisner E. (1992): *Procesos cognitivos y curriculum***.** Ed. Martínez Roca. Barcelona.
England E. (1988): *Case study: iterative screen design-errors as the basic of learning.* Educational & Training Technology International, 26,2, 149-155, citado en Cabero Almenara (1992).
Fainholc Beatriz: (1994): *La tecnología educativa propia y apropiada.* Humanitas. Bs. As.
Feire Paulo (1997): *Pedagogía de la Autonomía*. Ediciones.Siglo XXI
Fenton N. (1991): Software Metrics. A rigorous and practical approach. PWS Publishing Company. Boston.
Fernández Pérez M. (1995): *Las tareas de la Profesión de Enseñar*. Siglo veintiuno Editores.
Flagg B. (1990): *Formative evaluation for educational technologies*. Hillsdale, New Jersey: Lawrence Erlbaum Associates, Publishers.
Flavell J. H. (1993): *El desarrollo cognitivo*. Madrid. Ed. Visor.
Florín F. (1990): *Information Landscape*, Ambroa S y Hooper K.: *Learning with interactive multimedia.* Nueva York: Apple Computer, Inc. And Microsoft Press.
Gagné R. M. (1970): *The conditions of Learning*. Nueva York, Holt Rinehart & Winston.
Gallego D, Alonso C.: (1997): *Los Sistemas Multimediales desde una Perspectiva Pedagógica en Multimedia***,** UNED, Madrid.
Gallego D. y Alonso C. (1997): *Multimedia.* UNED. España.
Gallego M. J.: (1997): *La tecnología Educativa en acción.* Granada, Force. Universidad de Granada, p. 191-208.
Galvis A. (1996): *Software educativo multimidia aspectos críticos no seu ciclo de vida.* Revista Brasileira de Informática no Educaçao. Sociedad Brasileira de Computaçao. Consultado el 22/06/99 a las 23 horas.
www.janus.ufse.br:1085/revista/nr1/galvis_p.htm
Gane C. y Sarsons T. (1977): *Análisis Estructurado de Sistemas*. Quinta reimpresión. El Ateneo.
García López M. y Ruiz del Olmo F. (1997): *Nuevas Tecnologías, "El relato hipermedia"*. Universidad de Málaga, 1997.
Gardner H. (1995): *La mente no escolarizada*. Paidós
Gardner H. (1987-8): *La nueva ciencia de la mente: Historia de la psicología cognitiva.* Barcelona. Paidós.
Gardner H. (1993): *Las Inteligencias Múltiples: La teoría en la práctica.* Barcelona. Paidós.
Gardner H. (1997): *Inteligencias Múltiples.* Paidós
Garrido M. (1991): *Diseño y creación de software educativo*, Infodidac, 14-15, 31-34.
Garrido P. y Geisser C. (1996): *A metodology for software evaluation*.
Gayan J. y Segarra D. (1985): *Propuesta de evaluación de programas de enseñanza asistida por ordenador***,** en Pfeiffer, A. y Galván, J. (ed): *Informática y escuela,* Madrid, FUNDESCO, 379-382.
Gilb T.(1988): *Principles of software project management*. Nueva York. Addison Wesley.
Goldberg Mark F. (1991): *Portrait de Seymour Papert*, vol. 48. Nº 7 (1990-1991): Pág. 68-70, citado *en Seymour Papert* 1965-1996 por Paula Holder. (1996)
Goldberg Mark F. (1993): *Wishful Thinking.* Object Magazine, vol. 3, número 1, mayo-junio, págs. 87-88, citado en Piattini (1996)
Grady R. y Caswell (1987): *Software metrics establishing a company wide program.* Nueva Jersey. Prentice Hall.
Gros Begoña (1997): *Diseños y programas educativos*. Barcelona Ariel, citado en Marquès Graells Pedro: (1999): *Programas didácticos: diseño y evaluación.* Consultado el 22/05/99 www.xtec.es/~marquès/edusoft
Guiraudo, María Teresa (1997): *Seminario de Psicosociología de los Aprendizajes*. UTN-FRBA.

Guitert Catasús, Montserrat (1999): *Principios a tener en cuanta para una buena práctica pedagógica en Tecnología educativa y en educación a distancia.* III Curso Internacional de Tecnología Educativa Apropiada. 8 y 9 de mayo.

Gutiérrez, M. C. (1997) *Transferencia de masa; un problema a resolver.* Seminario de Psicosociología de los Aprendizajes. Maestría en Docencia Universitaria. UTN-FRBA.

Halstead M. (1975): *Elements of software science*. Nueva York. Elsevier.

Hammond N., Trapp A. et al. (1996): *Evaluating educational software: a suitables case for analyisis. AACE.* www.york.ac.uk/inst/ctipsych/ewb/CTI/WebCip/Hammond.html

Hartley J. (1972): *Strategies for Programmed Instruction: An Educational Technology.* Londres, Butterworths, citado por Cruz Feliú, Jaime (1986) en *Teorías del Aprendizaje y Tecnología de la Enseñanza,* Trillas.

Hativa W. y Reingold A. (1987): *Effects of audiovisual stimuli on learning through microcomputer-based class presentation,* Instructional Science, 16, 287-306, citado en Cabero Almenara (1992).

Heller R. (1991): *Evaluating Software: A review of the options*, Computers and education, 17, (4) págs. 285-291.

Henderson-Sellers B. y Edwards J. M. (1990*): Book Two of Object-Oriented Knowledge:* The Working Object; Prentice Hall.

Henry S. y Kafura D. (1984): *The evaluation of software systems: software practice an experience.* Vol. 14, número 6, págs. 561-573.

Hernández Rojas G. (1998): *Paradigmas en psicología de la educación*. Paidós Educador.

Holder Paula (1996): Seymour Papert 1965-1996. Consultado el 24/05/99 a las 22:40. www.ezinfo.ucs.indiana.edu/~pjholder/page3.html

Hopper y Hannafin, (1991) *Psychological Perspectives on emerging instructional Technology: A Critical Analysis.* Educational Psychologist, 26, 69-95, citado en Schunk Dale H.: (1997): *Teorías de la Educación*, Prentice Hall.

IEEE (1984a): IEEE 730. *Standard for software quality assurance Plans*. N. Y.

IEEE (1986): a. Standard 1008. *Standard for Software Unit Testing*. N. Y. B. Standard 1012. *Standard for software verification and validation Plans.*

IEEE (1989) Normas para el *Aseguramiento de la Calidad*

IEEE (1990): Standard 610, *Computer Dictionary*. Nueva York .

IEEE (1991): *Standard for Developing Software Life Cycle Process*. IEEE Std. 1074-1991 Nueva York. IEEE Computer Society.

IEEE (1991b): *IEEE Standard for software Test and Documentation*. Std. 820-1983.

IEEE (1992): *Standard for a software quality metrics methodology*. Std.1061

ISO (1991): *Information Technology Software Quality Evaluation Characteristics.* ISO 9126. Ginebra, Suiza.

ISO (1994): ISO/IEC 12701-1, *Software Life-cycle Process.*

ISO (1995) 12207-1: Information Technology-Software Life Cycle Processes. International Standard Organization. Suiza.

ISO 8402 (1994): *Gestión de la calidad y aseguramiento de la calidad*. Vocabulario.

ISO 9000 (1994): *Normas para la gestión de la calidad y el aseguramiento de la calidad.* Guía para su selección y uso.

ISO 9001 (1994): *Sistemas de la calidad.* Modelo para el aseguramiento de la calidad en el diseño, suministro y mantenimiento de soportes lógicos.

J. Juzgado, N. (1996): *Procesos de construcción del software y ciclos de vida.* Universidad Politécnica de Madrid.

Jackson M. A. (1975): *Principles of Program Design*. Nueva York. Academic Press.

Johnson-Laird, P.N. (1998): *El ordenador y la mente: introducción a la ciencia cognitiva*. Paidós.

Johnston V. M. (1987a): *Attitudes towards microcomputers in learning: 2. Teachers and software for language development,* Educational Research, 29, 2, 137-145, citado en Cabero Almenara (1992).

Johnston V. M. (1987b): *The evaluation of Microcomputer Programas: An area of debate,* Journal of Computer Assisted Learning, 3, (1): págs. 40-50.

Juran J. M. (1995): *Análisis y Planeación de la calidad*. Mc Graw Hill

Kemmis S. (1976): *The educational Potential of Computer Assisted Learning: Qualitative Evidence About Student Learning.*U. K. University of East Anglia.

Kemmis S., Atkin R. y Wright E. (1973-1975): *How do students learn?*: Working papers on CAL. Documento de trabajo número 5. Centre for applied Research in education. University of East Anglia. Gran Bretaña, citado en Squires y Mc Dougall (1994).

Komosky P. et al. (1995): *Seven steps to responsible software selection*. Eric Digest. Clearinghouse on information and Technology, Syracuse. N. Y.

Konrad M., Paulk M. y Graydon A. (1995*): An overview of Spice´s model for Process Management.* SEI. Proceedings of the Fifth International Conference on Software Quality, Austin, TX. 1995 Págs. 291-301

Ktcheman y Walter (1989): Citado en Fenton (1991): Software Metrics. A rigurous and practical approach. PWS Publishing Company. Boston.

Lachman R. et al. (1979): *Cognitive psychology and information processing: An introduction.* Hillsdale, N. J. Erlbaum.

Laurel B. (1990): *The art of human computer interface design*. Nueva York. Addison Wesley.

Ledesma D. A. (1980): *Estadística Médica*. Eudeba

Lehman M. (1984): *A Further Model of Coherent Programming Processes. Workshop,* Egham, UK, febrero, págs. 27-33, citado en Piattini (1996).

Lepper (1985): *Microcomputer in education: Motivational and social Issues*. American Psychologist, 40, 1-18, citado en Schunk Dale H.: (1997): *Teorías de la Educación* , Prentice Hall.

Libedinsky, M. (1995): *La utilización del correo electrónico en la escuela*, en Litwin (1995): *Tecnología educativa. Políticas, historias, propuestas*, Paidós.

Liguori, L. (1995): *Las nuevas tecnologías de información y comunicación,* en Litwin (1995): *Tecnología educativa. Políticas, historias, propuestas,* Paidós.

Llorca J. et al. (1991): *Desarrollo de software dirigido a objetos.* (DDO): En Novática, vol. XVIII, número 47, citado en Piattini (1996).

Logo, (1994) *Logo: Educational applications of.* (1994): *In the International Enciclopedia of Education,* vol. 15, pág. 3508-3512, citado *en Seymour Papert* 1965-1996 por Paula Holder (1996)

MacDonald B., Atkin R., Jenkins D.y Kemmis S. (1977): *Computed Assisted Learning: its educational potential,* en Hooper R. (Ed.): Final Report of the Director National Development program in Computer Assisted Learning. Londres. Council for Educational Technology, citado en Squires y Mc Dougall (1994).

Maddison R. N. (1983): *Information System methodologies*. Wiley Henden.

Mager R. F. (1967): *Formulación operativa de objetivos didácticos*, Madrid, Marova.

Manual de la Universidad de Málaga. Bioestadística: Métodos y Aplicaciones. ISBN 847496-653-1. Facultad de Medicina. consultado el 28/9/99 a las 10 hs. www.ftp.medprev.uma.es/libro/node148.htm

Markle S. M. (1967): *Empirical Testing of Programs en P. C. Lange Ed. Programmed Instruction,* Chicago University of Chicago Press, págs. 104-138, citado por Cruz Feliú, Jaime (1986) *en Teorías del Aprendizaje y Tecnología de la Enseñanza,* Trillas.

Marquès P. (1995): *Metodología para la elaboración de software educativo en Software Educativo. Guía de uso y metodología de diseño*. Barcelona Estel. Consultado el 24/05/99 a las 23 horas. www.xtec.es/~pmarques, www.doe.d5.ub.es

Marquès, Pere: (1998a): *La evaluación de programas didácticos*. Comunicación y Pedagogía, Nº 149, p. 53-58. Barcelona.

Marquès, Pere: (1998b): *Programas didácticos: diseño y evaluación*. Universidad Autónoma de Barcelona. Consultado el 15/10/98 a las 22:30 horas www.doe.d5.ub.es/te

Martin J. y Odell J. (1997): *Métodos orientados a objetos*. Prentice Hall.

Mc Cracken D. y Jackson A. (1982): *Lifecycle concepts considered harmful*: ACM, Sigsoft Software Engineering Notes, vol. 7, número 2, abril, págs. 29-32, citado en Piattini (1996).

McCabe T. (1976): *A complexity measure.* IEEE Transactions on software Engineering, vol.2, número 4, págs. 308-320.

McCall J. (1977): *Factors in software quality,* vols. I, II y III. NTIS; Roma, citado en Piattini (1996)

Meritxell Estebanell (1996): *Ficha de Evaluación de Programas Educativos*, Universidad de Girona.

Meyer B. (1990): *La nueva cultura del desarrollo de software*. En System, setiembre, págs. 12-13, citado en Piattini (1996).

MicroSIFT (1982): *Evaluation guide for Microcomputer-Based Instructional Packages. Microcomputer Software Information for Teachers.* (MicroSIFT): Northwest Regional Laboratory, Oregon, citado en Squires y Mc Dougall (1994).

Morín Edgard (1995): *Ciencia con conciencia*. Anthropos.

Murrilo F.J. y Fernádez M. J. (1992): *Software educativo. Algunos criterios para su evaluación,* Infodidac, 18, 8-12.

Myers G. (1975): *Reliable Systems through Composite design,* 1º Ed. Petrocelli Charter., citado en Piattini (1996).

Naur P. Y Randell B. (1969): Editores. Software engineering: A report on a Conference sponsored by the NATO Science Committee, citado en Pressman (1993).

Newell y Simon (1975): *Procesamiento de la información en la computadora y en el hombre,* en Crosson F. J. (comp.): *Inteligencia humana e Inteligencia Artificial.* Fondo de Cultura Económica: México.

Nielsen Jacob: (1995): *Multimedia and Hypertext, The Internet and Beyond*, - AP Professional.
Norman D. (1988): *The psychology of everyday things*. New York. Basic Books.
Norman D. y Drapper S. (1988): *User centered system design*. Hillsdale. N.J: Lawrence Erlbaum.
Novak J. y Gowin D. B. (1988): *Aprendiendo a aprender,* Barcelona. Martínez Roca.
OCDE (Organización para la Cooperación y el Desarrollo Económico) (1989): *Information Technologies in Education: The Quest for Quality Software,* Paris, Organisation for the Economic Cooperation and Development.
Olivares M. A. et al. (1990): *A proposal to answer the necessity to evaluate computer software,* en McDougall, A. y Dowling, G. (editors): Computers in Education, Elsevier Science Publishers, North-Holland, 171-174,
Osuna J., Bermejo J. L. y Berroso J. (1997): *Evaluación de medios informáticos: una escala de evaluación para software educativo.* EDUTEC 97. Comunicaciones: Formación y recursos.
OTA (Office Technology Assessment de E.E.U.U.) (1988): *Power on! New tools for Teaching and Learning.* Washington D. C, U.S. Government Printing Office, citado en Squires y Mc Dougall (1994).
Page-Jones M. (1980): *The Practical Guide to Structured Systems Design*. 1º Ed. Yourdon Press.
Papert Seymour: (1981): *Desafío a la mente,* Ediciones Galápagos.
Pelgrum J. y Plomp T. (1991): *The use of computers Worlwide*. Oxford, Pergamon Press, citado en Squires y Mc Dougall (1994).
Perkins D. (1995): *La Escuela Inteligente*. Gedisa
Pessacq R., Iglesias O. et al. (1997): *Evaluation of University Educational Software*. John Wiley & Sons. Apl. Eng. Educ. 5: 181.185.
Piaget J.(1989): *La construcción de lo real en el niño*. Crítica. Grijalbo.
Piattini M. (1996): *Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión.* Rama. Madrid.
Pina, Bartolomé (1998): *Sistemas multimedias en educación.* Consulta on line www.doe.d5.es.te/WEBNTES/t, 20 de abril de 1999 a las 21 horas.
Pozo Municio, I: (1998): *Aprendices y Maestros*. Alianza.
Preece J. y Jones A. (1985): *Training teachers to select educational software: results of a formative evaluation of an Open University pack,* British Journal of Educational Technology, 16, 1, 9-20, citado en Cabero Almenara (1992).
Prendres Espinosa M. P. (1996): *El multimedio en entornos educativos, en II Jornadas sobre medios de comunicación, recursos y materiales para la mejora educativa,* Sevilla, Centro Municipal de Investigación y Dinamización Educativa y Secretariado de Recursos Audiovisuales y Nuevas Tecnologías, citado en Castillo Segurado (1997).
Pressman R.(1993): *Ingeniería de Software. Un enfoque práctico*. Mc Graw Hill.
Raven J. C. (1979): *Test de Matrices Progresivas. Escala General*. Vol. 3b. Paidós. Buenos Aires.
Raven J. C. (1979): *Test de Matrices Progresivas. Manual para la Aplicación*. Paidós. Buenos Aires (con notas de Jaime Bernstein).
Reay D. G. (1985): *Evaluating Educational software for the classroom,* en Reid I, y Rushton J. (Eds.): *Teachers, computers and the classroom*. Manchester. Manchester University Press, págs. 79-87, citado en Squires y Mc Dougall (1994).
Reeves T. C. (1993): *Evaluating technology based learning*, in Piskurich. ASTD. Handbook of Information Technology, citado en Reeves T. C. (1997).
Reeves T. C. (1997): *Evaluation tools*, consulta on line en diciembre de 1998. www.mime1.marc.gatech.edu/MM_tools/evaluation.html
Requena A. y Romero F. (1983): *¿Cómo seleccionar el software educativo?, El ordenador personal,* 13, 47-51, citado en Cabero Almenara (1992).
Rivera Quijano M. (1999): *Nuevos caminos para evaluar proyectos y materiales educativos tecnológicos y para educación a distancia*. III Curso Internacional de Tecnología Educativa Apropiada. 8 y 9 de mayo de 1999.
Rivière A. (1987): *El sujeto de la psicología cognitiva*: Madrid. Alianza.
Rogers C. (1984): *Libertad y creatividad en la educación*. Paidós
Romiszowski (1981): Universidad de Syracuse. E.E. U: U: *Designing Instructional System.* London: Nichols Kogan Page.
Romiszowski A. D. (1981): citado en *Psicología y Curriculum* por César Coll (1994): Paidós.
Rowntree D. (1982): *Educational Technology in curriculum development*. Londres. Harper and Row, citado en Squires y Mc Dougall (1994).
Royce W. (1970): *Managing the development of Large software systems: Concepts and techniques*. Proceedings, Wescon, agosto, 1970, citado en Piattini (1996).
Rumbaugh J. (1991): *Object Oriented modeling and design*: Prentice Hall, Englewood Cliffs. Nueva Jersey.
Rumbaugh J. (1992): *Over waterfall and into the whirpool*. En JOOP, mayo, págs. 23-26, citado en Piattini (1996).

Salvas A. D. y Thomas G. J. (1964): *Evaluation of software,* Melbourne, Department of Victoria, citado en Squires y Mc Dougall (1994).
Sánchez J. y Alonso O. (1997-8): *Evaluación distribuida de software educativo a través de Web.* www.dcc.uchile.cl/~oalonso/educacion/, consultado el 23/11/98 a las 23:30 horas.
Sancho J. (1994): *Para una Tecnología Educativa,* Editorial Horsori. Barcelona. España.
Schunk Dale H. (1997): *Teorías de la Educación,* Prentice Hall.
Self J. (1985): *Microcomputers in Education: a critical appraisal of educational software*. Brighton, Harvester Press, citado en Squires y Mc Dougall (1994).
Shall W. E., Leake L. y Whitacker W. (1986): *Computer education: Literacy and beyond:* Monterrey, California. Brooks-Cole, citado en Squires y Mc Dougall (1994).
Sigwart C. et al. (1990): *Software Engineering: a project oriented approach*. Franklin, Beedle y Associates, Inc., Irvine, California, citado en Piattini (1996).
Skinner B. F., (1958, 1963): Teaching Machines, Science, publicado en 1958; Reflection on a decade of teaching Machines, publicado en 1963, citados por Cruz Feliú, Jaime (1986) en *Teorías del Aprendizaje y Tecnología de la Enseñanza,* Trillas.
Smith D. y Keep R. (1986): *Children's opinions of educational software*, Educational Research, 28, 2, 83-88, citado en Cabero Almenara (1992).
Solomon, C. (1987): *Entornos de aprendizaje con ordenadores. Una reflexión sobre las teorías del aprendizaje y la educación*. Temas de Educación. Paidós. M.E.C.
Sommerville Y. (1985): *Software Engineering.* Addison Wesley.
Squires D. y Mc Dougall A. (1994): *Cómo elegir y utilizar software educativo*. Morata. Barcelona.
Stufflebeam D. y Shinkfield (1987): *Evaluación Sistemática*. Paidós.
Taylor R. P. (1980): *The computer in the School: tutor, tool, tutee*. Nueva York. Teachers College Press, citado en Squires y Mc Dougall (1994).
Templeton R. (1985): *Be careful but Don´t worry: a guide to buying educational software,* en Tagg W, (Ed.): *A Parent´s Guide to Educational Software,* Londres, Telegraphs Publications, págs. 54-64, citado en Squires y Mc Dougall (1994).
Truett A. (1984): *Field testing educational software: are publishers making the effort?*, Educational Technology, mayo, 7-12, citado en Cabero Almenara (1992).
Underwood J. D. y Underwood G. (1990): *Computers and Learning*, Oxford, Basil Blackwell, citado en Cabero Almenara (1992).
Valencia M. E., Toro I. y Donneys C. (1998): *Desarrollo de aplicaciones hipemedia: propuesta para el diseño educativo.* TISE´98. Consultado el 28/9/99 a las 10 hs. en www.sofia,univalle.edu.co/gidse
Vigotzkii L. (1978): Mind in Society. *The development of higher psychological process.* Cambridge. M. A. Harvard University Press.
Villar, M.; Mínguez, E. (1998): *Guía de evaluación de software educativo*. Grupo ORIXE. Euskadi.
Wellington J. J. (1985): *Children, computers and the curriculum,* Cambridge, Harper & Row, Publishers, citado en Cabero Almenara (1992).
Winograd T. (1996): *Bringing design to software*. New York. ACM Press.
Wishart J. (1989): *Cognitive factors related to the user involvement with computers and their effects upon learning an educational computer game.* Paper read at the Cal'89, Conference University of Surrey, citado en Cabero Almenara (1992).
Yin B. y Winchester J. (1978): *The establishment and use of measures to evaluate the quality of software design.* Performance Evaluation Review, vol. 7, número 3-4, págs 45-52, citado en Piattini (1996).
Yourdon E. y Constantine L. (1975): *Structured design,* 2º Ed. Englewood Cliffs, Prentice Hall.
Zangara A. (1998): *Seminario de Sistemas Multimediales Aplicados a la Educación.* UTN.