INGENIERIA DE SOFTWARE EDUCATIVO

Cataldi, Z.[1], Lage, F.[1], Pessacq, R.[2] y García Martínez, R.[3,4]

1. Laboratorio de Sistemas Operativos y Bases de Datos. Departamento de Computación. Facultad de Ingeniería UBA.
2. Facultad de Ingeniería. UNLP
3. Centro de Ingeniería del Software e Ingeniería del Conocimento (CAPIS) ITBA.
4. Laboratorio de Sistemas Inteligentes. Departamento de Computación. Facultad de Ingeniería UBA.

ferzul@mara.fi.uba.ar, rpessacq@ing.unlp.edu.ar, rgm@mara.fi.uba.ar

RESUMEN

Se sintetizan 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 principales paradigmas del ciclo de vida a la luz de la visión de Piattini [1996], desde la cascada tradicional hasta los actuales orientados a objetos.

A fin de seleccionar el ciclo de vida adecuado para cada desarrollo, se analizan las actividades de cada uno de los procesos del mismo. Se define calidad del software, la normativa vigente para un proyecto de software y se hace una revisión de las métricas de calidad comúnmente usadas.

Por último, se desea presentar una síntesis de los criterios de selección y de evaluación de los programas didácticos, a modo diacrónico en las últimas décadas y sincrónico en el fin de siglo, considerando las pautas fijadas en el ámbito didáctico para lograr una efectividad del producto. 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.

INTRODUCCIÓN

En esta segunda parte, se intenta responder a la pregunta: ¿Cuáles son los estándares de la ingeniería del software aplicables para los diseños de programas educativos? Se pretende este aporte sea además, una aproximación para la fijación de directrices para las tareas concernientes al desarrollo de software para el área educativa.

El software educativo es uno de los pilares del sistema de educativo a distancia y se perfila como la herramienta base de las próximas generaciones de educandos.

Las mismas técnicas que utiliza el software educativo se podrán aplicar al desarrollo de sistemas utilizables en otras áreas para el facilitar el manejo de terminales por usuarios no informáticos.

Debido al creciente desarrollo del software educativo durante los últimos años, 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 educativas y de la comunicación subyacentes.

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 [Gallego 1997] con el usuario, las cuales no se pueden cuantificar mediante métricas porque están relacionadas con conductas de aprendizajes. Pero, las reglas en la construcción de un programa son las mismas ya sea educativo, comercial, de investigación, etc.

Esta segunda parte, se pretenden considerar los aspectos técnicos de los diseños educativos a concatenar con los marcos teóricos educativos expuestos en la primera parte. Se seleccionan, fundamentalmente aquellas herramientas que facilitarán los diseños actuales.

1- LA INGENIERÍA DEL SOFTWARE

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 para aplicaciones generales.

El problema se agrava cuando el desarrollo corresponde al ámbito educativo debido a la total inexistencia de marcos teóricos interdisciplinarios entre las dos áreas de trabajo.

Aunque 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 etapas del desarrollo, de lo que constituye un material didáctico informatizado. Esta necesidad sigue vigente, ya que en la mayoría de los casos analizados, se trata de software hipermedial diseñados a partir de herramientas de autor.

Marquès [1995], es uno de los autores que plantea 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, otorgándoles el rol protagónico.

Es por este motivo, que en este artículo se sintetizan las metodologías, métodos, herramientas y procedimientos de la ingeniería de software, que deben ser utilizados para lograr un producto óptimo 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

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 al tipo 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 educativas 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 y col., 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 lo han tratado 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 Proccesses" (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].

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 y col. [1990]. En este modelo, el producto evoluciona a través de una secuencia de fases ordenadas en forma lineal y 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.

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]. En cada paso sucesivo se agregan al sistema nuevas funcionalidades o requisitos que permiten el refinado a partir de una versión previa.

Este modelo es útil cuando la definición de los requisitos es ambigua e imprecisa, porque permite el refinamiento, o sea se pueden ampliar los requisitos y las especificaciones derivadas de la etapa anterior.

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

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].

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:

- ✓ El modelo de agrupamiento o de clúster de Meyer [1990]

- ✓ El modelo fuente de Henderson-Sellers y Edwards [1990]
- ✓ El modelo de pinball de Amler [1994]

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."

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 etapas del proceso base de construcción de software. Este estándar determina el conjunto de actividades esenciales (no están ordenadas en el tiempo) que deben ser incorporadas dentro de un modelo de ciclo de vida del software y la documentación a considerar.

LA NECESIDAD DE UNA METODOLOGÍA DE DESARROLLO

Maddison [1983] define metodología como un conjunto de filosofías, etapas, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas de información.

Piattini [1996], 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". 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, conducentes a una mejor calidad.
- ✓ Un proceso de desarrollo controlado.
- ✓ Un proceso normalizado en una organización, no dependiente del personal.

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, pudiendo ser gráficos con textos.

CARACTERÍSTICAS Y CLASIFICACIÓN DE LAS METODOLOGÍAS

Se pueden enumerar una serie de características [Piattini, 1996] 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

EL CICLO DE VIDA Y LOS PROCESOS

Todo proyecto tiene asociado, por más pequeño que éste sea, una serie de pasos que se deben seguir tales como: planificación, estimación de recursos, seguimiento y control, y evaluación del mismo. 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 en 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. Para estimar el tamaño del producto o 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.

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 debajo:

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.

LA IDENTIFICACIÓN DE LA NECESIDAD

Un enunciado en términos concretos, es el punto de partida para la puesta en marcha de un proyecto y la evaluación de las posibles soluciones que darán la viabilidad del mismo. Por lo tanto, es deseable, confeccionar un informe de necesidades basados en las siguientes ítems de la tabla debajo:

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.

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 las salidas resultantes, la descomposición de los datos, el procesamiento de los mismos, las bases de datos y las interfaces de usuario.[J. Juzgado, 1996].(ver tabla debajo).

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 interface de usuario, de interface con otros software y con hardware. Requisitos de interface 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

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.

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

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 fisico 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.

Los módulos tienen una función específica y definida o sea cohesión máxima y mínima interacción con lo 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.

EL PROCESO DE IMPLEMENTACIÓN

Este proceso (ver tabla debajo), 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.

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

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.

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

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 nueva aplicación del ciclo de vida pero a un software existente en una iteración de desarrollo. El retiro es la baja de un sistema existente.

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

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.

Cuando ya existe código ejecutable, se pueden realizar las pruebas del mismo, como verificación y validación del software. Las pruebas consisten en ejecutar el software con determinados datos de entrada y producir resultados que luego serán comparados con los teóricos.

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

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. Es necesaria la documentación del sistema en un momento determinado, el establecimiento de una configuración inicial y un control de los cambios.

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.

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. La formación del usuario en el sistema es un aspecto fundamental, como también lo es la formación de los desarrolladores y del soporte técnico.

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

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 lograr, a partir de los requisitos del desarrollo especificados correctamente.

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 del 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

LA CALIDAD DEL SOFTWARE

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".**
>
> Crosby [1979] pone énfasis en la prevención y dice producto "**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. [Piattini, 1996]

LA CALIDAD EN INGENIERÍA DE SOFTWARE

El software es un producto que tiene características muy especiales, hay que tener en cuenta que es un producto que se desarrolla y se centra su el diseño, con una existencia lógica, de instrucciones sobre un soporte. Es 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.

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].

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.

MODELOS DE EVALUACIÓN DEL SOFTWARE

La revisión y las pruebas del software son procesos orientados a la detección de defectos en el producto.

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 F_C, según:

$$F_C = c_1 \times m_1 + c_2 \times m_2 + \ldots + c_n \times m_n$$

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. Mc Call 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 ogoal-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.

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.

LA VERIFICACIÓN Y LA VALIDACIÓN DEL SOFTWARE

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 el problema para el que fuera diseñado.

La VyV actúa sobre los productos intermedios intentando detectar y corregir cuanto antes sus defectos y desviaciones del objetivo si las hubiera.

2- LA EVALUACIÓN DE SOFTWARE EDUCATIVO.

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. 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.

LA EVALUACIÓN NTERNA

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 (que se realiza al obtener la versión alfa) y otra evaluación externa en la que participan profesores y alumnos destinatarios del programa, cuando se haya terminado el mismo, o esté casi listo.

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 cuales son las ventajas que da al profesor como material didáctico y cómo facilita los aprendizajes de los alumnos.

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 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 hacer todos los cambios necesarios y convenientes. Luego de producidos los cambios, se agregarán los efectos faltantes (como sonido, animaciones, imágenes y gráficos) y la carga de la base de datos, para proceder a emitir una versión denominada beta. [Marquès, 1995]

La documentación es un proceso que se realiza paralelamente durante todo el desarrollo del programa, y también deberá ser evaluada externamente, junto con el programa.

LA EVALUACIÓN EXTERNA

La evaluación externa permite obtener sugerencias de los alumnos, quiénes serán en definitiva los usuarios del software y de los docentes que lo utilizarán como material didáctico. Durante este tipo de pruebas, se encuentran 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.

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 obtendrá la primera versión del programa con su respectivo manual de usuario, conteniendo todos los aspectos que se consideren indispensables para el uso docente, con detalles técnicos, y del entorno pedagógico y didáctico en el que se desarrolló el programa.

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 instrumento de evaluación, en forma de planillas se deben confeccionar con inclusión de preguntas del tipo cerradas, 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 de software es particular, se deben analizar con cuidado las diferentes propuestas de evaluación de medios didácticos y en particular de software educativo, teniéndoselas sólo como una "guía" que luego se deberá "readaptar" a cada contexto educativo particular.

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 a nivel individual y otras a nivel institucional. Si bien varían en cuanto a contenido y estilo, todas ellas tienen un objetivo común, que es ayudar al docente a elegir y valorar un programa adecuado.

En cuanto a las propuestas de evaluación se puede citar el formulario para la evaluación de materiales informáticos de MicroSIFT (Microcomputer Software Information For Teachers) [1982] del Northwest Regional Educational Laboratory, de Oregon en Estados Unidos, [OCDE, 1989]. Esta institución fue la primera que se dedicó a informar acerca del software educativo. Diseñó un instrumento que sirviera como base para el proceso de evaluación.

Salvas y Thomas [1964], elaboraron una lista de control para el Education Departament of Victoria, Australia. Coburn y Kelman [1985] intentaron dar orientaciones para la selección de software educativo como conjuntos de preguntas referidas a contenido de programa, pedagogía, manejo del programa y resultados de los estudiantes.

Preece y Jones [1985] utilizaron un hoja de criterios de selección elaborada para uso en un curso de la Open University, para reforzar los conocimientos de los profesores en la selección del software. Templeton en 1985 publica un libro con criterios de selección para padres, sugiriendo los criterios para adquisición de los programas y Reay [1985] detecta la necesidad de aplicación de un enfoque de evaluación más general, para cubrir la gran variedad de programas, pero lo suficientemente específico para poder tomar decisiones acerca de la conveniencia o no de su uso.

En cuanto a los instrumentos de evaluación, cabría acotar que Heller [1991] cita dos listas de control para evaluación: la de Shall, Leake y Whitacker [1986] y la de EDUCOM [1989]. Es interesante, el planteo respecto de la duda de la validez de las listas de control presentado por Johnston [1987], quien sostiene que no se puede prever cómo se utilizará un determinado software en el aula, ya que depende del enfoque docente, tipo de curriculum, estrategia usada y de los usuarios finales. Muchas veces profesores y alumnos pueden descubrir aplicaciones de la tecnología que se pasarían desapercibidos al diseñador. Esta es quizás una de las apreciaciones más significativas, ya que tiene en cuenta la multidimensionalidad de los procesos involucrados y los diferentes estilos de los actores en los procesos, como también los aspectos institucionales al considerar el tipo de curriculum.

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. La propuesta de María Ester del Moral [1998] consiste en una ficha de evaluación de aspectos creativos y de elementos de evaluación.

Carlos Dorado [1998] presenta una propuesta de evaluación: denominada guía de requerimientos y funcionalidades didácticas en el diseño y creación de sistemas interactivos multimedia. Esta es una de las listas más completas en cuanto a la consideración de los aspectos pedagógicos. Es importante destacar la consideración y el señalamiento de las actividades de comprensión que promueve: observar, retener, inferir, transferir, comparar, ordenar, clasificar, recuperar, presentar, interpretar, evaluar, etc.

Meritxell Estebanell [1996] propone un ficha de evaluación de programas educativos, que se caracteriza por categorizar en: identificación del programa, requerimientos técnicos, descripción, análisis didáctico, ubicación, problemas y propuesta de integración curricular. Hace un gran hincapié en el análisis didáctico y los aprendizajes que posibilita.

Marquès [1995], de la Universidad Autónoma de Barcelona propone una ficha para catalogación y evaluación de programas didácticos, concientizado de que al evaluar un programa, hay que considerar sus caracteristicas y su adecuación al contexto en el que se quiere utilizar. Considera la evaluación contextual de los programas como 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.

Un trabajo para tener en cuenta es el de Reeves [1993, 1997] con su matriz tridimensional en la que propone como dimensiones de análisis: la pedagógica, la matriz de evaluación y las dimensiones de interface de usuario, desglosando cada una en un lista de preguntas. Como la evaluación del software educativo puede ser realizada desde diversas perspectivas y por diversas personas y especialistas; respecto de este aspecto Olivares y col. [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 una investigación de Truett en 1984, encuentra que para la evaluación de productos comerciales de software educativo se utilizaba dos procedimientos: valoración por parte de los profesores y estudios experimentales, dos estrategias que utiliza Cabero Almenara para su investigación. Resume las investigaciones de Preece y Jones [1985], Bork [1986], Wellington [1985], Gayan y Segarra [1985], Garrido [1991], Murrilo y Fernádez [1992] y Requena y Romero [1983].

Merece especial mención la evaluación distribuida de software educativo de Sánchez y Alonso [1997] de la Universidad de Chile. Presentan una propuesta para mejorar y enriquecer la evaluación del software educativo que consiste en un cuestionario, que considera tres aspectos fundamentales de la producción del software educativo: el diseño de interfaces de usuario, la informática educativa y la comunicación visual. Presentan una forma de acceso innovadora mediante el acceso vía Internet, a fin de hacer más simple el proceso de evaluación del software educativo. Señalan que algunos autores consideran que debería responder a un modelo curricular (por ejemplo, no es lo mismo un programa para reforzamiento que y uno de apoyo a aprendizaje colaborativo) centrado en el profesor o en el alumno y que este sea constructivista o conductista y que diversos autores consideran que todo software debe considerar en su construcción un proceso de evaluación formativa y sumativa.

Komosky y col. [1995] analizan siete etapas para la selección de software educativo, Clarke y Peté [1996] plantean la metáfora de evaluación del software educativo en forma de mapa conceptual hipertextual en la Web.

Fundamentan el usao del hipertexto basándose en la taxonomía de Bloom [1956][1]. Akahori [1995] en Japón, remarca la laboriosidad del proceso de producción de software

Baumgartner y Payr [1996] hablan de la evaluación de los programas que se realiza mediante listas de control o grupos de aprendizaje, cotejando sus diferencias de resultados. Ambos métodos son reduccionistas en si mismos, ya que o no consideran a los estudiantes o se los piensa como receptáculos de conocimientos. Reemplazan el concepto de calidad absoluta por valores relativos. Garrido y Geisser [1996], proponen partir de las características de los alumnos tanto psicológicas, psicosociales y cognitivas y del contexto, la experiencia del docente en el área, los objetivos curriculares y el entorno administrativo, habiéndose de definir cómo el software será usado en la clase y si es apropiado para alcanzar los objetivos.

Hammond N. y col [1996], plantean cuándo y cómo el docente puede usar los programas educativos para mejorar los proceso de enseñanza y aprendizaje. Netiva Caftori y col. [1997] resumen los posibles efectos del uso de software no supervisado y sugiere entre otras cosas la evaluación de los programas, teniendo relevancia los aspectos sociales y los objetivos educacionales, basados en la idea de construcción del docente alrededor del software.

En 1997, Pessacq y col. desarrollaron un método de evaluación de software educativo en el contexto de la Universidad Nacional de La Plata, agrupando las preguntas en tres categorías: contenidos, metodología de enseñanza y características de diseño del programa. Reunían en un cuarto grupo consideraciones generales. La intención era medir la utilidad de software en los procesos de enseñanza y aprendizaje. Hacían especial hincapié en el diseño del programa o amigabilidad y en la filosofía del programa. Hablan del concepto de calidad verdadera como difícil de cuantificar, y polisémico pensado como satisfacción del usuario, y que como no es medible cuantitativamente, se puede estimar como satisfacción de usuario. Consideran que los resultados de la evaluación pueden permitir mejorar las versiones del programa.

Por último, se ha dejado a Bartolomé Pina [1998], 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 usos que se hicieren 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 mediante el uso controlado de un programa determinado. En estos caso, cabe recurrir a la experiencia o a la consulta de los grandes distribuidores cono Anaya Interactiva[2] y Zeta Multimedia[3].

CONCLUSIONES

De acuerdo a las necesidades en cada caso, y a la teoría educativa y/o el curriculum, se deberá 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.

De acuerdo a los diseños que hemos realizados, los usuarios en general solicitan uno o más prototipos, para ver como evoluciona el desarrollo durante el tiempo previsto y si 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.

Quedaría por señalar una 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].

Se prevé plantear de este modo una posible solución a la problemática de los desarrollos de los programas educativos,

Por último habría que 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, 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.

---

[1] Conocimiento, comprensión, aplicación, análisis, síntesis y evaluación.
[2] www.anayainteractiva.com
[3] www.zetamultimedia.es

REFERENCIAS

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

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

Amler (1994): citado por Piattini (1996)

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

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 aprooach to the evaluation of interactive media.** Educational Multimedia and Hypermedia. AACE, peter.baumgartner@uni-klu.ac.at, sabine.payr@uni.klu.ac.at

Baumgartner y Payr (1996): **Educational multimedia**, Association for el Advancement of Computing Education, Charlottesville, V.A.

Bloom B, y col. (1956): **Taxonomy of educational objetives. The classification of educational goals**. David McKay. N. Y.

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

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

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

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

Cabero Almenara (1992): **Diseño de Software informático**. Universidad de Sevilla. Bordón, 44,4 383-391 ISSN: 0210-5934 cabero@cica.es

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. Asociation for el Advancement of Computing Education, Charlottesville, V.A. caftori@neiu.edu, paprzycki_m@utpb.edu

Clarke P. Y Peté M. (1996): **The KwaZulu concept burger: A hypertext concept map of educational software evaluation.**

Coburn P., Kelman P. y col. (1985): **Practical guide to computers in Education**, Reading Massachusetts. Addison Wesley, citado en Squires y Mc Dougall (1994).

Crosby P. (1979): **La calidad no cuesta**. Mc Graw Hill. México

Del Moral M. E. (1998): citada por Marquès [1998], emoral@pinon.ccu.uniovi.es

Deming W. E. (1982): **Out of the crisis**. Cambridge University Press.

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).

Fenton (1997): Software Metrics. A rigurous and practical approach. PWS Publishing Company.

Galvis A. (1996**): Software educativo multimidia aspectos críticos no seu ciclo de vida.** Revista Brasileira de Informática no Educaçao. Sociedad Brasileira de Computaçao, 1996. www.janus.ufse.br:1085/revista/nr1/galvis_p.htm

Gardner H. (1993): **Las inteligencias múltiples. La teoría en la práctica.** Barcelona. Paidós

Gallego D. y Alonso C. (1997): **Multimedia**. UNED. España.

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 (1993). **Wishful Thinking**. Object Magazine, vol. 3, número1, 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.

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

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

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 sofware systems, software practise an experience.** Vol. 14, número 6, págs. 561-573.

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

IEEE (1991): **Standard for Developing Software Life Cycle Proccess**. IEEE std. 1074-1991 Nueva York. IEEE computer Society.

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**.

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.

Johnston V. M. (1987): **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

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

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

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

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

Marquès P. (1995): **Metodología para la elaboración de software educativo** en **Software Educativo. Guía de uso y metodología de diseño**. Barcelona Estel. www.xtec.es/~pmarques, www.doe.d5.ub.es

Marquès, Pere: (1998): **Programas didácticos: diseño y evaluación**. Universidad Autónoma de Barcelona. Consultado en octubre de 1998. www.doe.d5.ub.es/te

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. mem@fce.udg.es

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).

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

Novak J. y Gowin D. B. (1988): **Aprendiendo a aprender**, Barcelona. Martínez Roca.

OCDE (1989): **Information Technologies in Education**: **The Quest for Quality Software**, París, Organisation for the Economic Cooperation and Development.

Olivares M. A. y col. (1990): **A proposal to answer the necessity to evaluate computer software**, en McDougall, A. y Dowling, G. (editors): Computers in Education, Elsevier Science Pubishers, 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 (1988): **Power on! New tools for Teaching and Learning**. Washington D. C., U.S. Government Printing Office, citado en Squires y Mc Dougall (1994).

Perkins D. (1995): **La Escuela Inteligente**. Gedisa

Pessacq R., Iglesias O. (1997): **Evaluation of University Educational Software**. John Wiley y Sons. Apl. Eng. Educ. 5: 181.185.

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.

Preece J. y Jones A. (1985). **Traigning 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).

Pressman R.(1993): **Ingeniería de Software. Un enfoque práctico**. Mc Graw Hill.

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, pégs. 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): Evalaution 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).

Royce W. (1970): **Managing the development of Large software systems: concepts and Techniques**. Proceedings, Wescon, agosto, 1970, citado en Piattini (1996).

Salvas A. D. y Thomas G. J. (1964): **Evaluation of software**, Melbourne, Departament of Victoria, citado en Squires y Mc Dougall (1994).

Sanchez J. y Alonso O. (1997): **Evaluación distribuida de software educativo a través de la WEB**. http://www.dcc.uchile.cl/~oalonso/educacion/

Sancho (1994): **Para una Tecnología Educativa**, Editorial Horsori. Barcelona. España.

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. y col. (1990): **Software Engineering: a project oriented approach**. Franklin, Beedle y Assocites, Inc., Irvine, California, citado en Piattini (1996).

Sommerville Y. (1985): **Software Engineering**. Addison Wesley.

Templeton R. (1985): **Be careful but Don´t worry: a gue 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).

Wellington J. J. (1985): **Children, computers and the curriculum,** Cambridge, Harper & Row, Publishers, citado en Cabero Almenara (1992).

Yin B. y Winchester J. (1978): **The establishment an 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).