UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS.

FACULTAD 9

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE

INGENIERO EN CIENCIAS INFORMÁTICAS.

MÉTODOS PARA EL DESARROLLO DE SOFTWARE EDUCATIVO.

UN ANÁLISIS COMPARATIVO.

AUTOR:

Yamilesy Maritza Bles Urquiza.

TUTOR:

M Sc. Ing. Febe Ángel Ciudad Ricardo.

CIUDAD DE LA HABANA, JULIO 2008

"AÑO 50 DE LA REVOLUCIÓN"

"Un científico no sólo tiene la obligación de investigar, sino que también tiene la responsabilidad ética sobre las consecuencias de aquello que produce su ingenio."

Peter Ustinov

Dedicatoria

A mi mamá por todo el esfuerzo que ha hecho por su entrega, amor y dedicación que ha tenido conmigo y con mi bebé.

A mi papá y a mi familia, por todo el apoyo que me dieron para que yo estuviera en esta universidad y me graduara.

A mi esposo por darme amor, cariño, comprensión y ayuda en todo este tiempo.

A mis amistades del Protocolo específicamente a las que trabajan en **"**La Casona".

A todos mis amigos que me han brindado su amor, cariño y ayuda para que salga adelante en todos los aspectos de la vida.

A todos aquellos que forman parte de este logro.

A mi tutor Febe Ángel Ciudad Ricardo por haber puesto tanto empeño y dedicación, además de que fue mi profesor en segundo y tercer año, por formar parte del claustro de profesores que contribuyó a mi formación profesional.

A mi mamá por todo el amor, esfuerzo y empeño que ha dedicado a mi vida porque yo sea una persona preparada y sea merecedora de un título.

A mis padres y familia, y a todas aquellas personas que han contribuido a lo largo de los años en mi formación profesional y a ser cada día mejor persona.

A la Profesora Zoraida que fue mi profesora guía en los años más difíciles de la universidad y colaboró no solo como profesora sino como si fuera mi mamá.

A todos los profesores que me dieron clase en todos los años que hicieron posible que yo me formara como una profesional de este país.

A Fidel Castro Ruz y a la Revolución queme dieron la oportunidad de estudiar en esta universidad para hacer realidad mi sueño de ser Ingeniera Informática.

A los amigos que me han acompañado en el transcurso de estos 5 años y que han contribuido en gran medida a que la estancia en esta universidad sea más placentera.

A Yoannia y Anay por ser mis compañeras de siempre en todo los momentos de mi vida buenos y malos.

Les Digo adiós a ustedes mis amigos, profesores y compañeros; siempre es preciso saber cuando se acaba una etapa de la vida recordar los bueno momentos que pase junto a ella. En esta etapa de mi vida que ha sido la más importante tengo mucho que agradecer no solo a los que hicieron posible el desarrollo de este trabajo de diploma sino a todos aquellos que me han ayudado y apoyado a lo largo de mi vida.

Se que para ustedes esto ha sido muy importante y que sienten un inmenso orgullo al verme graduada porque fue lo que mas han deseado. Aquí esta nuestro triunfo y éxito porque no es solo mío, es de ustedes también. Los Amo con la vida.

Gracias a todos por formar parte de mi vida y ser parte de este triunfo.

Declaración de Autoría

Por este medio declaro que soy la única autora de este trabajo y autorizo a la Universidad de las Ciencias Informáticas (UCI) para que haga el uso que estime pertinente del mismo.

Para que así conste firmamos la presente a los 9 días del mes de julio del 2008.

Yamileysy Maritza Bles Urquiza	M.Sc. Ing. Febe Ángel Ciudad Ricardo
Autor	Tutor

TUTOR: M.Sc. Ing. Febe Ángel Ciudad Ricardo

Graduado de: Ingeniero en Informática (2004).

Grado Científico: Máster en Informática Aplicada (2007)

Categoría Docente: Profesos Asistente

Centro de Trabajo: Universidad de las Ciencias Informáticas (UCI)

Cargo: Jefe del Departamento de Práctica Profesional & Ingeniería y Gestión de Software de la Facultad 9.

Actividad profesional: Profesor Asistente de Ingeniería de Software, Gestión de Software, Metodología de la Investigación Científica, Práctica Profesional 5 y Seminario de Tesis.

Dirección: Universidad de las Ciencias Informáticas (UCI), Edificio: 40, Apto: 40104. Teléfono: +53 – 7 – 8372557

E-mail: fciudad@uci.cu

Resumen

El avance de la Ciencia y la Tecnología en la actualidad, ha traído consigo numerosos cambios en disímiles sectores y esferas de la sociedad a través de la introducción de las Tecnologías de la Información y las Comunicaciones (TIC). La Universidad de las Ciencias Informáticas (UCI) enmarcada en este desarrollo se ha fusionado a este amplio y competitivo mercado. Esta tiene como fuerte línea de producción, el software educativo, como vía de integración de estos procesos de enseñanza-aprendizaje con las nuevas tecnologías. Los estudios que han sido realizados, sobre el proceso que se lleva a cabo para la elaboración de estos productos, han arrojado un significativo número de problemas que atentan contra el correcto desarrollo de los proyectos de producción de software educativo.

En toda la industria de software a nivel mundial se utilizan métodos para la modelación de los programas de cómputo, estos están compuestos por actividades. Existen diferentes métodos para la construcción de software a nivel nacional entre estos se pueden mencionar las fases definidas por Pressman, que se utilizan para el desarrollo del software a nivel mundial. Teniendo en cuenta las necesidades y perspectivas de la educación en el país, en la UCI se creó un Sistema Metodológico para el Desarrollo del Software Educativo con el objetivo de obtener una guía para el proceso de desarrollo de este tipo de software, que cumpliera de forma rápida y eficiente con las especificidades requeridas por los usuarios. Estos métodos establecen metas con el fin de obtener una mejor elaboración de los productos; los mismos se rigen por un grupo de actividades que favorecen una mejor construcción del software pues le indican la organización, los puntos a tener en cuenta para su desarrollo.

Este trabajo está estructurado en dos capítulos. El Capítulo 1, contempla la fundamentación teórica de esta investigación, en la cual son expuestos los principales conceptos y argumentos que esclarecen el objeto de estudio. En el Capítulo 2 se realizó una comparación entre los métodos utilizados en el flujo productivo de la UCI y se describe cuál es el método más eficiente para su desarrollo por el orden de prioridad que tiene.

Palabras clave:

Software Educativo, Métodos, Procesos, Modelos.

Índice de Tablas y Figuras

Índice

Introducción

"Las empresas que desarrollan software no pueden ignorar que su negocio es un negocio de software, y que el modelo que cada una adopte para las actividades de desarrollo y mantenimiento tiene implicaciones relevantes en la eficiencia general del negocio. El problema que pueden encontrar quienes deciden implantar métodos más eficientes, es caer en la desorientación ante el abanico de modelos de calidad, de procesos y de técnicas de trabajo desplegado en la última década; o abrazar al primero que se presenta en la puerta de la organización como "solución" de eficiencia y calidad". (PALACIO, 2005)

En la actualidad se reconoce una gran variedad de modelos y metodologías, que se han ido desarrollando a nivel internacional y adaptándose a las condiciones de la empresa en la que se desarrolla así como a las características del producto en particular. Por la importancia que se le confiere a los métodos relacionados con el proceso de desarrollo de software, muchos autores dedican su tiempo a la organización, explicación y mejora de los mismos. Hoy día existe un significativo número de clasificaciones para los modelos de desarrollo de software y las metodologías que se conocen.

"La cada vez más creciente necesidad de integración entre los procesos de enseñanza-aprendizaje y las tecnologías de la Información y las Comunicaciones ha generado una alta demanda de productos de software educativo a nivel mundial. Ante este fenómeno un amplio y competitivo mercado de desarrollo de software educativo incrementa la presión sobre los proyectos de esta rama en materia de eficiencia y calidad de las soluciones en todos los aspectos del desarrollo: tecnológicos, financieros, contractuales, etc." (MARTÍNEZ, y otros, 1993) Para su creación no se ha podido contar con un conjunto amplio de metodologías que permitan la guía de estos procesos productivos pues no existe un estándar internacional que regule la gestión de los mismos.

Precisamente, retomando una idea anterior, sí existe un significativo número de modelos, metodologías y procesos para guiar el desarrollo de software, pero la inserción del software educativo a esta gran industria ha promovido el replanteamiento de algunos métodos dirigidos básicamente a la solución de determinadas características que hacen valorar nuevas alternativas para un software de nuevo tipo.

Cuba ha identificado desde muy temprano la conveniencia y necesidad de dominar e introducir en la práctica social las Tecnologías de la Información y las Comunicaciones para lograr una cultura informática como característica importante del hombre nuevo; le facilitaría a nuestra sociedad alcanzar el objetivo de un desarrollo sostenible.

Nuestro país, se encuentra inmerso en un proceso de transformaciones dentro del cual está priorizado el sector educacional. Se han creado muchas instituciones en el país

que contribuyen al crecimiento informático, y como parte de los aportes que cada día sustentan ese desarrollo se encuentra la elaboración de diversos software educativos. La Universidad de las Ciencias Informáticas (UCI) como un nuevo proyecto de la Revolución ha crecido para desarrollar software y alcanzar un alto nivel en el avance de la economía mundial donde se han especificado los tipos de software que se van a realizar, entre ellos el software educativo uno de los más importantes para alcanzar un logro inmediato y superior al existente en el sector informático educacional. Conque

La UCI se ha unido a los programas con que cuenta nuestra nación, para alcanzar un desarrollo inmediato y superior en la enseñanza de las nuevas generaciones, se ha dedicado tiempo, costo y esfuerzo para la creación de diversos productos que colaboren a conquistar los objetivos anteriormente mencionados, pero el propio hecho de ser una institución que actualmente se encuentra en formación, donde estudios hechos a la universidad demuestran que existe muy poca experiencia por parte de estudiantes y profesores, donde se han tenido que realizar diversas modificaciones a partir de los resultados obtenidos según experiencias anteriores, ha provocado la aparición de diversas dificultades en el desarrollo de los productos de software educativo, las cuales vienen dadas por numeroso factores que atentan contra su desarrollo exitoso por lo que se creo una estrategia de trabajo “Sistema Metodológico para el desarrollo de Software Educativo en la UCI” para poder eliminar estas dificultades.

La UCI constituye hoy día un eslabón esencial en la producción de software educativo y es por ello que necesita la documentación necesaria para guiar dichos procesos, la cual es muy escasa y la existente no resuelve en sí las particularidades de cada uno. Su reciente aparición en el mercado del software trae consigo la no existencia de un conjunto vasto de metodologías que se dediquen especialmente a tratar esta nueva concepción. Al respecto Cataldi afirmó: *“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”.* (CATALDI, 2000)

Como se hizo alusión en la cita anterior, no existe un estándar teórico-metodológico-técnico-científico que pueda ser usado para el desarrollo de software por todas las personas que se encuentren vinculadas al desarrollo de proyectos informáticos, mucho menos, cuando se trata de productos correspondientes al plano educativo, en el cual se hizo necesario tener en cuenta los aspectos pedagógicos-didácticos y de comunicación

con el usuario para cada caso en particular. Es por esta principal razón que se hizo necesario investigar sobre los métodos existentes para la modelación del software educativo en proceso productivo de la UCI.

Para lograr establecer satisfactoriamente estos productos con calidad, los desarrolladores en el área requieren de *Métodos para el desarrollo del SWE* que ofrezcan soluciones completas que satisfagan al cliente, y logren convertir necesidades o requerimientos en software.

Las condiciones descritas hasta el momento, en las cuales se desenvuelve la producción de software educativo a nivel internacional, nacional y de forma particular en la universidad; así como el desarrollo de los métodos ingenieriles de producción de Software, conlleva a que se haya identificado el problema científico: ***Escaso conocimiento de los métodos más utilizados en el entorno productivo de SWE de la Universidad de las Ciencias Informáticas, lo que produce deficiencias en la documentación y modelación de este tipo de SW.***

Para resolver el problema mencionado anteriormente se estableció un análisis comparativo entre los métodos utilizados para definir cual o cuales son los más factibles; trabajando en el objetivo general de ***Priorizar por ordenamiento y comparar, de acuerdo a las necesidades del entorno productivo de la Universidad de las Ciencias Informáticas, los mejores métodos existentes para el desarrollo de SWE.***

Para darle cumplimiento a dicho objetivo fue necesario centrar la investigación en la ***Identificación de los elementos comunes en los métodos utilizados en el entorno productivo de la UCI*** *así como en la* ***Caracterización de las mejores prácticas y tendencias en los métodos para la modelación de SWE en el mundo***; lo cual constituyeron los objetivos específicos.

El objeto de estudio de esta investigación lo constituyeron los ***métodos para la modelación de SWE*****.**

El campo de acción de este trabajo estuvo centrado en el ***Proceso de desarrollo del software educativo en la UCI.***

Con el análisis realizado en esta investigación se obtuvo ***un estudio comparativo de los métodos para la modelación de SWE más utilizados en el entorno productivo mundial,*** *que posibilita defender la idea de que* ***permitirá un mejor conocimiento y establecimiento de las mejores prácticas y tendencias actuales en la modelación de este tipo de aplicaciones en la Universidad de las Ciencias Informáticas.***

Para dar cumplimiento a los objetivos anteriormente planteados se siguieron las

siguientes tareas:

1 Identificar los principales tendencias científicas-técnicas y teóricas, así como el desarrollo histórico del estado actual de la producción de SWE en Cuba.

2 Comparar los métodos de desarrollo nacionales e internacionales aplicables al SWE.

3 Identificar los elementos significativos reutilizables en la producción del software educativo de los métodos de desarrollo utilizados en el entorno productivo mundial.

4 Describir las etapas genéricas en las que se modela el SWE según los métodos estudiados.

5 Identificar las teorías de representación de la información más utilizados en los métodos estudiados.

6 Priorizar por ordenamiento los mejores métodos de modelación de SWE existentes en la universidad.

7 Comparar los métodos de desarrollo más utilizados.

8 Confeccionar las memorias de la investigación para su socialización e inserción en la comunidad científica UCI.

Para el desarrollo de la investigación se utilizaron métodos teóricos y empíricos. Dentro de los métodos teóricos son utilizados los de análisis y síntesis, el histórico-lógico y el causal y dentro de los métodos empíricos se utilizaron la entrevista y la encuesta. A continuación se especifica el por qué de la selección de los mismos.

1. Se aplicó el “Método Histórico” para investigar si existe un método específico planteado en la universidad que haya guiado en la actualidad el proceso de desarrollo de SWE.

2. El método “Causal” para estudiar los factores que provocaron la necesidad de un proceso de este tipo en la institución.

3. El método de “Análisis” para comprender y definir cual o cuales fueron los métodos que se pueden utilizar como guía para el desarrollo de SWE y el de “Síntesis” para plantear, describir y resumir el proceso que se desarrolló a partir de la investigación en la UCI.

4. Entrevistas y Encuesta a líderes y estudiantes de proyectos de SWE en la UCI, para conocer cómo se desarrolló el proceso de desarrollo de SWE de forma general en la universidad.

Para lograr los resultados de esta investigación y tener credencial de la misma, fue necesario seleccionar una población, de la cual se extrajo una muestra que fue analizada. De los resultados obtenidos a partir de este análisis se pudo deducir como se

iba a comportar dicha población.

A continuación se presentan:

Población: Tres líderes de proyecto y los desarrolladores de SWE de la Universidad de las Ciencias Informáticas

Muestra: Tres líderes de proyecto de SWE en la UCI.

Unidad de estudio: desarrollador de SWE en la UCI.

Para la selección de la muestra fue utilizada la técnica estratificada: Por ejemplo si se desea estudiar los proyectos de la UCI que realizan SWE, se parte de los tres Líderes de proyectos existentes en la UCI tomado como población, se selecciona la muestra que representa la población en estudio y finalmente se investiga la unidad de estudio.

El trabajo que se presenta a continuación está estructurado en dos capítulos.

El Capítulo 1 contempla la fundamentación teórica de esta investigación, en la cual se definen los conceptos relacionados con Informática Educativa. Se realizó a su vez una presentación de las características de los software educativos, los tipos de métodos utilizados para el desarrollo del software y los modelos existentes para la modelación del software educativo.

En el Capítulo 2 se realizó un acercamiento al flujo productivo de la UCI donde se abordan los métodos utilizados a partir del análisis de un conjunto de factores que intervienen indistintamente en el desarrollo de SWE, se hizo necesario realizar un análisis comparativo de los métodos más utilizados con el fin de proponer cual es el método más factible para el proceso de desarrollo del software.

Capítulo 1 Fundamentación Teórica

Introducción.

"El problema del software se reduce a la dificultad que afrontan los desarrolladores para coordinar las múltiples cadenas de trabajo de un gran proyecto de software. La comunidad de desarrolladores necesita una forma coordinada de trabajar. Necesita un ***método*** *que integre las múltiples facetas del desarrollo del Software……*". (JACOBSON, y otros, 2000)

En este Capítulo se hizo un análisis de todos los conceptos asociados al dominio del problema, pues este es la base para el entendimiento del objeto de estudio que rige esta investigación, el cual se centró en el desarrollo del software educativo en la Universidad de las Ciencias Informáticas. De ahí se hizo necesario plantear los fundamentos teóricos que contribuyen a esclarecer el origen y desarrollo del problema, hasta llegar al planteamiento y estudio de un conjunto de soluciones que son utilizadas en la actualidad para la producción de software de forma general.

1.1 Algunos conceptos importantes en el área del Software Educativo

Con el objetivo de proporcionarle al lector, la vía más adecuada para una mayor comprensión de los temas que serán abordados en este capítulo, directamente relacionados con el objeto de estudio de la investigación, se describen detalladamente a continuación un grupo de conceptos asociados al dominio del problema, entre los que se encuentran: el software educativo, el proceso de enseñanza – aprendizaje y las Tecnologías de la Información y las Comunicaciones (TIC).

1.1.1 Ingeniería de Software.

La Ingeniería de Software es una tecnología multicapa en la que, según Pressman, se pueden identificar: los métodos (indican cómo construir técnicamente el software), el proceso (es el fundamento de la Ingeniería de Software, es la unión que mantiene juntas las capas de la tecnología) y las herramientas (soporte automático o semiautomático para el proceso y los métodos) (ver Figura 1). (PRESSMAN 2002)

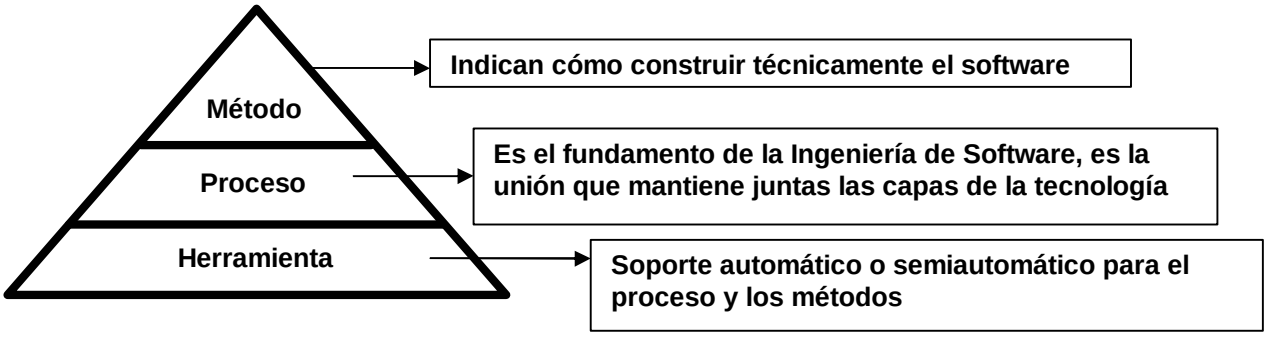


Figura 1. La ingeniería de software es una tecnología multicapas.

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 realizó básicamente de acuerdo al tipo de proyecto y de aplicación, los controles y las entregas a realizar.

La ingeniería del software es la parte que estudia cómo desarrollar software que permite resolver aplicar métodos y técnicas para resolver los problemas, esta aporta herramientas y procedimientos que permite:

- Mejorar la calidad de los productos de software
- Aumentar la productividad y trabajo de los ingenieros del software
- Facilitar el control del proceso de desarrollo de software
- Suministrar a los desarrolladores las bases para construir software de alta calidad en una forma eficiente.
- Definir una disciplina que garantice la producción y el mantenimiento de los productos software desarrollados en el plazo fijado y dentro del costo estimado.

1.1.2 Software Educativo.

El **software educativo**, programas educativos y programas didácticos se usan como sinónimos para designar genéricamente los programas para computadoras, creados con la finalidad específica de ser utilizados como medio didáctico, es decir, para facilitar los procesos de enseñanza y de aprendizaje. (MARQUÉS, 1995)

Esta definición engloba todos los programas que han estado elaborados con un fin didáctico, desde los tradicionales programas basados en los modelos conductistas de la enseñanza, los programas de Enseñanza Asistida por Ordenador (EAO), hasta los programas experimentales de Enseñanza Inteligente Asistida por Ordenador (EIAO), que, utilizando técnicas propias del campo de los Sistemas Expertos y de la Inteligencia Artificial

en general, pretenden imitar la labor tutorial personalizada que realizan los profesores y presentan modelos de representación del conocimiento en consonancia con los procesos cognitivos que desarrollan los alumnos. (MARQUÉS, 1995)

No obstante según esta definición, más basada en un criterio de finalidad que de funcionalidad, se excluyen del software educativo todos los programas de uso general en el mundo empresarial que también se utilizan en los centros educativos con funciones didácticas o instrumentales como por ejemplo: procesadores de textos, gestores de bases de datos, hojas de cálculo, editores gráficos... Estos programas, aunque puedan desarrollar una función didáctica, no han estado elaborados específicamente con esta finalidad.

Se define al software educativo como un *programa computacional cuyas características estructurales y funcionales sirvan de apoyo al proceso de enseñar, aprender y administrar*. Un concepto más restringido de software educativo lo define como "*(…) aquel material de aprendizaje especialmente diseñado para ser utilizado con un computador en los procesos de enseñar y aprender."* (ÁLVAREZ, y otros, 2006)

La definición anterior es reflejada en al artículo: *"Software Educativo. Su influencia en la escuela cubana"*, cuyas autoras expresan posteriormente:

"Finalmente, los Software Educativos se pueden considerar como el conjunto de recursos informáticos diseñados con la intención de ser utilizados en el contexto del proceso de enseñanza – aprendizaje. Se caracterizan por ser altamente interactivos, a partir del empleo de recursos multimedia, como videos, sonidos, fotografías, diccionarios especializados, explicaciones de experimentados profesores, ejercicios y juegos instructivos que apoyan las funciones de evaluación y diagnóstico." (ÁLVAREZ, y otros, 2006)

El software educativo está formado por los programas educativos y didácticos creados con la finalidad específica de ser utilizados para facilitar los procesos de enseñanza – aprendizaje, como se había reflejado en las definiciones anteriores, es decir, tiene la finalidad de lograr que el estudiante adquiera conocimientos, habilidades, procedimientos, y esto es posible gracias a la interactividad que surge entre el estudiante y la computadora, a través de la cual se puede individualizar el trabajo de ese estudiante, pues este tipo de software permite adaptar el ritmo de trabajo a cada uno, así como las actividades que tiene implícitas según la actuación de los mismos.

Es importante destacar que existen programas que son utilizados en los centros educacionales con funciones didácticas, que aunque pueden desarrollar dichas funciones, no han sido elaborados específicamente con esa finalidad, por tanto se considera que está excluido de la categoría de software educativo. Ejemplo de ello tenemos a los gestores de

bases de datos, hojas de cálculo, entre otros.

Características esenciales de los programas educativos:

Los programas educativos pueden tratar las diferentes materias (matemáticas, idiomas, geografía, dibujo...), de formas muy diversas (a partir de cuestionarios, facilitando una información estructurada a los alumnos, mediante la simulación de fenómenos...) y ofrecer un entorno de trabajo sensible a las circunstancias de los alumnos rico en posibilidades de interacción; pero todos comparten cinco **características esenciales**:

1. Son materiales elaborados con una **finalidad didáctica**, como se desprende de la definición.
2. **Utilizan la computadora** como soporte en el que los alumnos realizan las actividades que ellos proponen.
3. **Son interactivos**, contestan inmediatamente las acciones de los estudiantes y permiten un diálogo y un intercambio de informaciones entre el ordenador y los estudiantes.
4. **Individualizan el trabajo** de los estudiantes, ya que se adaptan al ritmo de trabajo cada uno y pueden adaptar sus actividades según las actuaciones de los alumnos.
5. **Son fáciles de usar**. Los conocimientos informáticos necesarios para utilizar la mayoría de estos programas son similares a los conocimientos de electrónica necesarios para usar un vídeo, es decir, son mínimos, aunque cada programa tiene unas reglas de funcionamiento que es necesario conocer. (MARQUÉS, 1996)

Estructura básica de los programas educativos:

La mayoría de los programas didácticos, igual que muchos de los programas informáticos nacidos sin finalidad educativa, tienen tres módulos principales claramente definidos: el módulo que gestiona la comunicación con el usuario (sistema input/output), el módulo que contiene debidamente organizados los contenidos informativos del programa (bases de datos) y el módulo que gestiona las actuaciones del ordenador y sus respuestas a las acciones de los usuarios (motor). (MARQUÉS, 1996)

Las Tipologías:

Los programas educativos se pueden clasificar según diferentes tipologías. En la Tabla 1 se pueden ver algunas de acuerdo a diferentes criterios. Se debe considerar que un aspecto clave de todo buen diseño es tomar en cuenta las características de la interfase 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.

"Gran parte de los programas educativos pertenecen a un subgrupo denominado hipermediales, y en ellos las bases de datos de imágenes fijas o en movimiento, video clips y sonidos juegan un rol fundamental a la hora de diseñar el programa." (MARQUÉS, 1998)

Tipologías Según:	
Contenidos	Temas, áreas curriculares.
Destinatarios	Por niveles educativos, edad, conocimientos previos.
Estructura	Tutorial, base de datos, simulador constructor, Herramienta.
Bases de datos	Los medios que integra: cerrados o abiertos. Convencional hipermedia, realidad virtual.
Inteligencia	Convencional, sistema experto.
Objetivos educativos que pretende facilitar	Conceptuales, actitudinales, procedimentales
Procesos cognitivos que activa	Observación, identificación, construcción memorización, clasificación, análisis, síntesis, deducción, valoración, expresión, creación, entre otros.
Tipo de interacción que propicia	Recognitiva, reconstructiva, intuitiva, constructiva
Función en el Instructivo, revelador, conjetural, emancipador aprendizaje:	
Comportamiento	Tutor, herramienta, aprendiz

Tratamiento de los errores	Tutorial, no tutorial.
Bases psicopedagógicas sobre el aprendizaje	Conductista, constructivista, cognitivista.
Función en la estrategia Didáctica	Informar, motivar, orientar, ayudar, proveer, recursos, facilitar prácticas y evaluar.
Diseño	Centrado en el aprendizaje, centrado en la enseñanza, proveedor de recursos.

Tabla 1.Algunas tipologías, según Marqués (1998)

Clasificación del Software Educativo:

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

Los programas tutoriales, son aquellos 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 facilitan la exploración y la consulta selectiva, permitiendo extraer datos relevantes para resolver problemas, analizar y relacionar datos y extraer conclusiones.

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 en la clasificación de educativos, pero muchas veces son necesarios para la redacción final de trabajos, informes y monografías. (MARQUÉS, 1996)

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, se describen en forma sintética algunas de las funciones que pueden realizar este tipo de programas.

Función	Descripción
Informativa	Presentan contenidos que proporcionan una información estructurada 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 el interés de los alumnos y enfocarlo hacia los aspectos más importantes de las actividades.
Evaluadora	Al evaluar implícita o explícitamente, el trabajo de los alumnos.
Investigadora	Las más comunes la base de dato, 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 tecnologías más recientes.

Tabla 2.Funciones del Software Educativo. **(MARQUÉS, 1996)**

1.1.4 El proceso de desarrollo de software.

El concepto de proceso de desarrollo de software es unos de los conceptos necesarios para analizar y tener mejor dominio de la problemática que se trabaja.

*"Un **proceso** define quién está haciendo qué, cuándo y cómo alcanzar un determinado objetivo. En la ingeniería del software el objetivo es construir un producto software o mejorar uno existente. Un proceso efectivo proporciona normas para el desarrollo eficiente de software de calidad. Captura y presenta las mejores prácticas que el estado actual de la tecnología permite. En consecuencia, reduce el riesgo y hace el proyecto más predecible. El efecto global es el fomento de una visión y una cultura enormes".* (JACOBSON, y otros, 2000)

Blaya plantea que un proceso es "*un conjunto de actuaciones, decisiones, actividades y tareas que se encadenan de forma secuencial y ordenada para conseguir un resultado que satisfaga plenamente los requerimientos del cliente al que va dirigido*". (BLAYA, 2006)

Los autores del libro "*El Proceso Unificado de Desarrollo de Software"*, con los cuales se concuerda en la investigación presentada, expresan que el proceso de desarrollo de software, se puede definir como "*el conjunto completo de actividades necesarias para convertir los requisitos de usuario en un conjunto consistente de artefactos que conforman un producto software, y para convertir los cambios sobre esos requisitos en un nuevo conjunto consistente de artefactos*". (JACOBSON, y otros, 2000)

Analicemos ahora los dos nuevos conceptos necesarios para entender la definición presentada.

Requisitos: "*Este es uno de los flujos de trabajo más importantes, porque en él se establece qué tiene que hacer exactamente el sistema que construyamos. En esta línea los requisitos son el contrato que se debe cumplir, de modo que los usuarios finales tienen que comprender y aceptar los requisitos que especifiquemos."* (JACOBSON, y otros, 2000)

Artefacto: "*Un producto o artefacto es un trozo de información que es producido, modificado o usado durante el proceso de desarrollo de software. Los productos son los resultados tangibles del proyecto, las cosas que va creando y usando hasta obtener el producto final."* (JACOBSON, y otros, 2000)

Por su parte Roger S. Pressman en "*Ingeniería del Software. Un Enfoque Práctico."* caracteriza el proceso de desarrollo de software como *"(…) un marco de trabajo de las*

tareas que se requieren para construir software de alta calidad." Este proceso se muestra en la Figura 2, Los elementos involucrados en este se describen a continuación:

- Un marco común del proceso, definiendo un pequeño número de actividades del marco de trabajo que son aplicables a todos los proyectos de software, con independencia del tamaño o complejidad.
- Un conjunto de tareas, cada uno es una colección de tareas de ingeniería del software, hitos de proyectos, entregas y productos de trabajo del software, y puntos de garantía de calidad, que permiten que las actividades del marco de trabajo se adapten a las características del proyecto de software y los requisitos del equipo del proyecto.
- Las actividades de protección, tales como garantía de calidad del software, gestión de configuración del software y medición, abarcan el modelo del proceso. Las actividades de protección son independientes de cualquier actividad del marco de trabajo y aparecen durante todo el proceso."(PRESSMAN 2002).

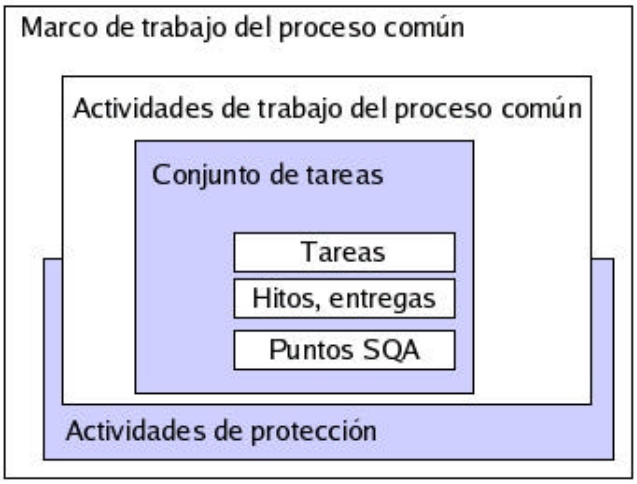


Figura 2. El Proceso Desarrollo del Software. (PRESSMAN, 2002)

En mismo texto mencionado se realizó un análisis interesante de lo que es proceso de desarrollo de software, donde se concluye que *"El proceso se ve influido fuertemente por las herramientas. Las herramientas son buenas para automatizar procesos repetitivos, mantener las cosas estructuradas, gestionar grandes cantidades de información y para guiarnos a lo largo de un camino de desarrollo concreto*". (JACOBSON, y otros, 2000)

Tomando como base los conceptos referenciados anteriormente se puede definir que para desarrollar un producto software es necesario utilizar un proceso soportado por herramientas con calidad y que sean a su vez estándares utilizados por los desarrolladores en amplias regiones del mundo o lo suficientemente contextualizados que permitan resultados beneficiosos para los que lo utilizan.

1.1.4 Tecnología multimedia.

Según la enciclopedia Microsoft Encarta'97 la multimedia se puede clasificar como una forma de presentar información, en su combinación de texto, sonido, imágenes, animaciones y video. Ejemplos de aplicaciones multimedia informáticas son: juegos interactivos, programas de aprendizaje, materiales de referencia, por ejemplo enciclopedias. En los últimos años, varios autores han intentado conceptualizar la tecnología multimedia. Pero una concepción multifocal de la multimedia es la que plantea Rodríguez Lamas: *"Es una nueva plataforma donde se integran componentes para hacer ciertas tareas que proporcionan a los usuarios nuevas oportunidades de trabajo y acceso a nuevas tecnologías. Es un nuevo medio donde la computadora junto con los medios tradicionales dan una nueva forma de expresión. Es una nueva experiencia donde la interacción con los medios es radicalmente diferente y donde tenemos que aprender como usarlos. Es una nueva industria donde, con una nueva plataforma, un nuevo medio y una nueva experiencia, nos llevan a tener nuevas oportunidades de negocios."* (Rodríguez, 2000)

La tecnología multimedia aparece en el año 1984, cuando la Apple Computer lanza la primera variante de Macintosh, la cual tenía amplias capacidades de reproducción de sonidos; apoyándose en su sistema operativo, propicio para el diseño gráfico y la edición. En 1987, con los videojuegos, los avances en las Tecnologías de la Información y las Comunicaciones (TIC) y el desarrollo por la Philips del Disco Compacto (CD), aumentan las posibilidades para el desarrollo de la tecnología multimedia.

En 1992, se presenta en la feria "Consumer Electronic Show" (CES) de Las Vegas, el CD multiusos, que luego dio paso luego a la televisión interactiva. En 1993, el nuevo término de multimedia, obliga a una revisión de los conceptos que permiten el funcionamiento de la nueva tecnología, buscando el desarrollo de estándares para lograr la uniformidad en el desarrollo. Hoy se ha desarrollado una nueva técnica que surgió a partir de la multimedia: la hipermedia, que tiene su base en el hipertexto, permitiendo al usuario un recorrido o navegación por distintos archivos o partes del programa de acuerdo a sus intereses personales.

La multimedia tiene varias aplicaciones entre las cuales se pueden mencionar las siguientes: (CORRALES DIAZ, 1994)

1. *En la diversión y el entretenimiento: donde se sitúan los juegos de video y las aplicaciones de pasatiempos de tipo cultural.*
2. *Multimedia en los negocios: sus principales exponentes están en los kioscos de información, las presentaciones, intercambio y circulación de información.*
3. *En publicidad y marketing: sus ejemplares son; la presentación multimedia de*

negocios, de productos y servicios, la oferta y difusión de los productos y servicios a través de los kioscos de información.

4. *En la difusión del saber y conocimiento: La característica de la interactividad de multimedia, que permite navegar por el programa y buscar la información sin tener que recorrerlo todo, logra que la tecnología se aplique en los nuevos medios de modos diferentes y se use de forma alternativa. Y por último entre los muchos beneficios que ofrecen la tecnología multimedia se puede mencionar: el impacto al incorporar imágenes, efectos de sonido, video y animación en tercera dimensión para crear presentaciones vivas y de extraordinaria calidad. La flexibilidad, ya que el material digital puede ser fácil , rápidamente actualizado y presentado a través de innumerables medios. El control por parte del emisor, al seleccionar la cantidad y tipo de información que desea entregar así como la forma de entregarla al igual que el control por parte del receptor, al elegir la información que quiere recibir y en el momento en que desea recibirla. El ahorro de recursos en materiales impresos difíciles de actualizar y presentándola en innumerables ocasiones sin ninguna restricción.*

1.1.5 Tecnología Hipermedia.

"Vannevar Bush, en la década de los 40, y Theodor Holme Nelson, en los 60, se consideran los artífices de la estructuración no lineal y de la interconexión de la información, asuntos que constituyen conceptos claves para el desarrollo de la interactividad informática aplicada a la comunicación. El hipertexto no es más que un conjunto de bloques de texto interconectados por nexos, que forman diferentes itinerarios para el usuario, donde estos nexos "electrónicos" unen fragmentos te texto internos o externos a la obra (soportes cerrados – off line – o abiertos – on line – respectivamente), creando un texto que el lector experimenta como no lineal o, mejor dicho, como multilineal o multisecuencial." (CIUDAD RICARDO, 2007)

La hipermedia surge como resultado de la fusión de dos tecnologías, el hipertexto y la multimedia. La tecnología multimedia es la que permite integrar diferentes medios (sonido, imágenes, secuencias...) en una misma presentación. La hipermedia, por tanto, es la tecnología que permite estructurar la información de una manera no-secuencial, a través de nodos interconectados por enlaces sobre los diferentes formatos de información.

Estas hipermedias y multimedias pretenden resolver el problema del procesamiento lineal de la información por el receptor, como ocurre en el libro de texto. Por el contrario, la información se puede construir desde diferentes trayectorias y alternativas, y con diferentes tipos de códigos. Estas trayectorias pueden limitarse por el autor del programa, para evitar problemas de desorientación en el usuario (PASTOR, 1997)), pudiendo soportar la autoría de documentos complejos y el procesamiento de ideas, especialmente en

entornos de trabajo colaborativos. Para la aplicación de esta facilidad los alumnos necesitan la capacidad de anotar nodos de información y ser capaces de colaborar con otros estudiantes y con el profesor sobre unidades específicas de información. Aún más importante que esto es que deben ser capaces de construir su propio sistema de conocimiento a través de la investigación, abstracción, readaptación y adición de conocimientos a una base de datos existentes.

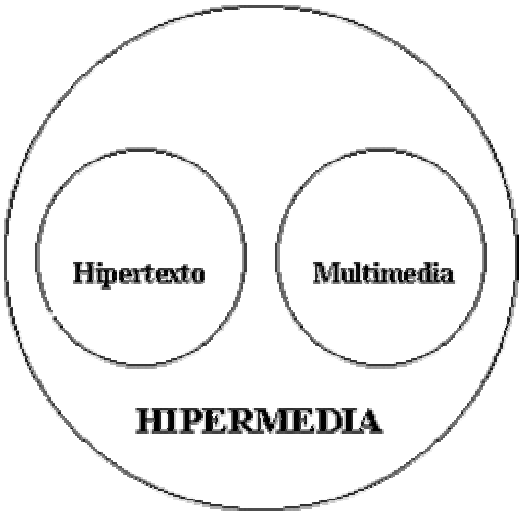


Figura 3. Hipermedia

1.1.6 Proceso de enseñanza – aprendizaje

Hasta ahora se han abordado un grupo de conceptos importantes para el entendimiento del trabajo que se presenta. Se hizo imprescindible entonces, abordar un concepto que tiene amplia relación con el software educativo y su desarrollo; más en el contexto productivo cubano: el proceso de enseñanza – aprendizaje.

"El proceso de enseñanza – aprendizaje constituye la vía mediatizadora esencial para la apropiación de conocimientos, habilidades, hábitos, normas de relación, de comportamiento y valores, legados por la humanidad, que se expresan en el contenido de enseñanza, en estrecho vínculo con el resto de las actividades docentes y extradocentes que realizan los estudiantes". (ZILBERSTEIN, y otros, 1999)

A su vez, Fernández considera que "*El proceso de enseñanza – aprendizaje es un proceso esencialmente interactivo y comunicativo, de intercambio de información, compartiendo experiencias, conocimientos y vivencias, que logran una influencia mutua en las relaciones interpersonales*". (FERNÁNDEZ, 2000)

Mancebo, plantea una idea interesante como conclusión: "Enseñanza y aprendizaje forman parte de un único proceso que tiene como fin la formación del estudiante". (MANCEBO, 1999)

El objetivo final de este proceso es la formación del estudiante y esto se logrará, en mayor

medida; a través de como el profesor muestra a sus alumnos los contenidos, hábitos y habilidades, utilizando medios y con la intención de lograr objetivos fijados desde un inicio en un determinado contexto.

1.1.7 Informática Educativa

Según Raúl Rodríguez Lamas, la Informática Educativa se puede definir como *"(…)la parte de la ciencia de la Informática encargada de dirigir, en el sentido más amplio, todo el proceso de selección, elaboración, diseño y explotación de los recursos informáticos dirigido a la gestión docente, entendiéndose por éste la enseñanza asistida por computadora y la administración docente."* (RODRÍGUEZ, 2000)

La verdadera revolución se ha producido a raíz de la generalización de los CD-ROMs como soporte de información para las aplicaciones multimedia y al mismo tiempo una auténtica revolución en el software, sobre todo en la interfaz de usuario. Vinculando la enseñanza por computadora con las nuevas tecnologías multimedia, surge lo que conocemos como los software multimedia educativos; herramientas poderosas dentro del contexto de la Informática Educativa.

1.2 Modelos de desarrollo de software.

Se comenzará este epígrafe analizando lo que ha sido aceptado por la Real Academia Española de la Lengua como concepto de Modelo; donde en su diccionario electrónico en línea aparece: *"Arquetipo o punto de referencia para imitarlo o reproducirlo. Ejemplar que por su perfección se debe seguir e imitar. Esquema teórico, generalmente en forma matemática, de un sistema o de una realidad compleja, que se elabora para su comprensión y el estudio de su comportamiento."*

La autora de esta memoria coincide con la última cláusula presentada en el párrafo anterior, por considerar que los programas computacionales con sistemas complejos, los cuales necesitamos modelar para hacer posible su comprensión y su estudio.

R. S. Pressman en su libro "*Ingeniería del Software. Un Enfoque Práctico"*, agrupa los modelos de desarrollo de software de la siguiente forma (PRESSMAN, 2002):

- Modelo Lineal Secuencial,
- Modelo de Construcción de Prototipos,
- Modelo DRA y
- Modelos Evolutivos, en él se incluyen:
  - el Modelo Incremental,
  - el Modelo Espiral,
  - el Modelo Espiral WinWin y

- el Modelo de Desarrollo Concurrente, entre otros modelos que hace alusión.

1.2.1 Modelo en Cascada.

El Modelo Lineal Secuencial o Modelo en Cascada es el paradigma más antiguo y más extensamente utilizado por la ingeniería del software. Su versión original fue presentada por Royce en 1970, aunque los refinamientos realizados por Boehm (1981), Sommerville (1985) y Sigwart (1990) son mucho más conocidos. Dicho modelo sugiere un enfoque sistemático, secuencial, para el desarrollo del software, es decir, el producto evoluciona a través de una secuencia de fases ordenadas en forma lineal, permitiendo iteraciones al estado anterior. Aunque las etapas o actividades que comprende este modelo suelen variar podríamos resumirlas en: Ingeniería y modelado de Sistemas/Información, Análisis de los requisitos del software, Diseño, Generación de Código, Pruebas y Mantenimiento.(Consultar Anexo 3)

En la práctica se ha logrado demostrar que este modelo carece de eficacia, sus problemas se centran en que en los proyectos reales, raras veces se sigue la linealidad que propone el mismo, además de que se ha demostrado que casi siempre hay iteraciones que siguen más allá de la etapa anterior. Conjuntamente, este modelo requiere que el cliente exponga explícitamente todos los requisitos, cosa que es muy difícil para este, lo que trae consigo numerosas dificultades a la hora de acomodar la incertidumbre natural al comienzo de muchos proyectos. También este modelo se caracteriza por la no disponibilidad de una versión de trabajo del programa hasta que el proyecto esté muy avanzado lo que provoca un efecto desastroso del mismo ante la aparición de un grave error que no haya sido detectado hasta revisar el programa.

Aunque este paradigma tenga estas debilidades es significativamente mejor que un enfoque hecho al azar para el desarrollo del software. (PRESSMAN, 2002)

1.2.2 Modelo de Construcción de Prototipos.

El Modelo de Construcción de Prototipos contribuye a eliminar las debilidades que fueron expuestas en el Modelo en Cascada pues ofrece un mejor enfoque ante situaciones que se puedan presentar tanto por el cliente como por el desarrollador de software. El uso de prototipos se centra fundamentalmente en la idea de comprender los requisitos de entrada, proceso o salida que no han sido identificados por el cliente, el cual, como se había dicho con anterioridad sólo es capaz de definir un conjunto de objetivos generales para el software. Simultáneamente, ayuda al desarrollador de software a comprender mejor lo que se necesita hacer y puede utilizarse cuando este tiene dudas acerca de la viabilidad de la solución pensada. Es importante destacar que una vez se tenga el prototipo construido es utilizado para refinar los requisitos del software a desarrollar, el cual se va incrementando gradualmente y va evolucionando hasta llegar a lo que se suele llamar “primer sistema”. En

este modelo las etapas del ciclo de vida clásico quedan modificadas de la siguiente forma: 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. (Consultar Anexo 4).

Aunque este paradigma le guste a los clientes y a los desarrolladores, puede tornarse problemática, pues cuando se le comunica al cliente sobre la necesidad de reconstruir el producto, que tuvo la oportunidad de ver como una versión del software, pero que en realidad trae asociado un sinnúmero de dificultades que atentan contra la calidad y facilidad de mantenimiento, este no entiende la magnitud del problema, y piensa que para lograr un producto final sólo se necesita realizar unos pequeños ajustes, cuando en verdad esta gestión de desarrollo frecuentemente se hizo muy lenta. Además, se puede presentar el problema de que lo que en un inicio se utilizó por el desarrollador de software como una elección viable y rápida para la construcción del prototipo y que pudo haber sido la selección menos adecuada ahora es una parte integral del sistema. (PRESSMAN, 2002)

1.2.3 Modelo Incremental.

El Modelo Incremental (Consultar Anexo 5) combina elementos (aplicados repetidamente) del modelo lineal secuencial (aunque corrige la problemática de la linealidad del modelo en cascada) con la filosofía interactiva de construcción de prototipos. Cada secuencia lineal produce un incremento del software. El modelo de proceso incremental, como la construcción de prototipos y otros enfoques evolutivos, es iterativo por naturaleza. Con cada incremento se va logrando una versión del producto y aunque en los primeros incrementos estas versiones del producto final sean incompletas, a diferencia del modelo de construcción de prototipos, proveen al usuario la funcionalidad que precisa, pues se centra en la entrega de un producto operacional con cada incremento. Para lograr esto 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. (PRESSMAN, 2002)

1.2.4 Modelo Espiral

El modelo en espiral propuesto originalmente por Boehm, es un modelo de proceso de software evolutivo que conjuga la naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial. Proporciona el

potencial para el desarrollo rápido de versiones incrementales del software. En el modelo espiral, el software se desarrolla en una serie de versiones incrementales que permiten su mejoría e incremento gradual y progresivo.

Se divide en regiones de tareas de las cuales existen generalmente entre tres y seis (Consultar Anexo 6). Estas se definen de la siguiente forma:

- **Comunicación con el cliente:** las tareas requeridas para establecer comunicación entre el desarrollador y el cliente.
- **Planificación:** las tareas requeridas para definir recursos, el tiempo y otra información relacionadas con el proyecto.
- **Análisis de riesgos:** las tareas requeridas para construir una o más representaciones de la aplicación.
- **Construcción y acción:** las tareas requeridas para construir, probar, instalar y proporcionar soporte al usuario (por ejemplo: documentación y práctica)
- **Evaluación del cliente:** las tareas requeridas para obtener la reacción del cliente según la evaluación de las representaciones del software creadas durante la etapa de ingeniería e implementada durante la etapa de instalación.

Cada una de estas regiones está compuesta por un conjunto de tareas que se adaptan a las características del proyecto que se emprende. En todos los casos se aplican actividades de protección como gestión de configuración de software y garantía de calidad de software.

"El primer circuito de la espiral puede producir el desarrollo de una especificación de productos; los pasos siguientes en la espiral se podrían utilizar para desarrollar un prototipo y progresivamente versiones más sofisticadas del software. Cada paso por la región de planificación produce ajustes en el plan del proyecto. El costo y la planificación se ajustan con la realimentación ante la evaluación del cliente".(PRESSMAN 2002)

El modelo en espiral puede adaptarse y aplicarse a lo largo de la vida del software, a diferencia del modelo de proceso clásico que termina cuando se entrega el software.

Durante cada ciclo de la espiral, aparece el análisis de riesgos, identificando situaciones que pueden hacer fracasar el proyecto, demorarlo o incrementar su costo. Es por ello que Pressman expresa que este modelo es un enfoque realista del desarrollo de sistemas y de software a gran escala, pues como el software evoluciona, a medida que progresa el proceso, el desarrollador y el cliente comprenden y reaccionan mejor ante los riesgos en cada uno de los niveles evolutivos. Para la reducción del riesgo aplica el enfoque de la construcción de prototipo y esta construcción se puede llevar a cabo en cualquier etapa del producto.

1.3 Los métodos de propósito general para el desarrollo del software.

Es pertinente plantear lo que define la Real Academia Española de la Lengua como Método: *"Modo de decir o hacer con orden. Modo de obrar o proceder, hábito o costumbre que cada uno tiene y observa."* (http://buscon.rae.es/draeI/)

Se hizo notar primero que todo en el concepto presentado, el hecho que *se hizo con orden*, elemento que en la disciplina de la Ingeniería de Software es de suma importancia. Al mismo tiempo presentamos el concepto ofrecido por Angel Puello en su libro "Foreing Language Teaching", con el cual se concuerda en la investigación: *"Conjunto de técnicas organizadas en determinada secuencia, mientras que un enfoque estaba íntimamente relacionado con principios teóricos. El primero era más concreto y tangible que el segundo, el cual tenía características más abstractas."* (PUELLO, 2002)

1.3.1 ¿Qué es un método de desarrollo de Software?

Analicemos en este acápite tres definiciones de Métodos de Ingeniería de Software que nos ubiquen en los elementos fundamentales de este concepto:

Según R. S. Pressman: "*Los métodos de la Ingeniería Software indican ¿cómo? construir técnicamente el software. Abarcan una gran gama de tareas que incluyen análisis de requisitos, diseño, construcción de programas, pruebas y mantenimiento. Dependen de un conjunto de principios básicos que gobiernan cada área de la tecnología e incluyen actividades de modelado y otras técnicas descriptivas."* (PRESSMAN, 2002)

A su vez, Zabala plantea que: *"Los métodos de la ingeniería del software son procedimientos, reglas, la documentación asociada y los datos que pertenecen a un sistema de cómputo".* Según el mismo autor, *"es la rama de la ingeniería que aplica los principios de la ciencia de la computación y las matemáticas para lograr soluciones costo-efectivas (eficaces en costo o económicas) a los problemas de desarrollo de software es decir, permite elaborar consistentemente productos correctos, utilizables y costo-efectivos."* (Zabala, 2000)

Smorville enuncia que: *"Un método de ingeniería de software es un enfoque estructurado para el desarrollo de software cuyo propósito es facilitar la producción de software de alta calidad de una forma costeable"*. (SMORVILLE, 2002)

Al leer estas definiciones puede notarse como elementos comunes en estas, aspectos como: aplicación ordenada de los elementos de la informática y la computación a la solución de problemas reales de la sociedad en todos sus órdenes, teniendo en cuenta la relación costo-beneficio y apoyándose en notaciones para la representación del desarrollo de las soluciones. Todos los métodos se basan en la idea de modelos gráficos de desarrollo de un sistema y en el uso de estos modelos como un sistema de especificación

o diseño.

La introducción de esta investigación refleja que el objeto de estudio sobre el cual versa esta tesis es acerca de los métodos para la modelación de SWE, enmarcado específicamente en la Universidad de las Ciencias Informáticas.

Dada la diversidad de los contextos educativos en el mundo, así como los procesos de enseñanza y aprendizaje disímiles que rigen las escuelas pedagógicas en el planeta, las formas de hacer software educativo por los desarrolladores tienen que responder a variados entornos, lo que produce también variedad en los métodos de desarrollo de software. Dados estos variados métodos de desarrollo, por ejemplo, podemos encontrar software con gran capacidad de aprendizaje que en realidad constituye todo un portento de programación pero de una pobreza enorme en su capacidad de enseñar o bien con intenciones didácticas de una pobreza en los algoritmos empleados en su producción que conllevan a errores significativos en la interacción de software con el usuario.

Es necesario entonces, que para la producción de los mismos, el equipo de trabajo que los produce tenga en cuenta los métodos que va a utilizar para su desarrollo y su intención, que es ayudar o ser un instrumento de ayuda en el proceso de enseñanza-aprendizaje.

1.3.2 Clasificación de los Métodos de desarrollo de Software.

En el presente epígrafe se hizo alusión a las dos clasificaciones más utilizadas hoy día en la Ingeniería de Software para la clasificación de los métodos de desarrollo. La primera de ella ofrecida por Piatini en 1996 y la segunda descrita por Pressman en el año 2002.

Piatini clasificó los métodos de la siguiente forma:

Estructurados:

- Representan los procesos, flujos y estructuras de datos, de una manera jerárquica, descendente ven el sistema como entradas-proceso-salidas

Orientados a procesos:

- Se centran en la parte proceso.
- Constan de (fundamentalmente) Diagrama de flujo de datos (DFD), Diagramas de datos (DD), miniespecificaciones de proceso, Diagrama de Entidad Relación (DED)

Orientados a datos:

- Se orientan más a las entradas y salidas.
- Primero se definen los datos.
- A partir de ellos, los componentes procedimentales.
- “Los datos son más estables”.

Se pueden distinguir seis escuelas principales de pensamiento en relación con las técnicas y métodos de desarrollo de ingeniería del software. (PIATTINI, 1996)

Orientadas a procesos	Fundamentan en el modelo básico *entrada/proceso/salida* de un sistema, de forma que los datos introducen en el sistema y éste responde ante ellos transformándolos para obtener salidas. Estas metodologías enfocan fundamentalmente en la parte de proceso y, por esto, se describen como un enfoque de desarrollo de software orientado al proceso.
Orientadas a datos:	Al contrario que en el caso anterior, estas metodologías se centran más la parte de entrada/salida dentro del modelo básico entrada/proceso/salida. En estas metodologías las actividades de análisis comienzan evaluando en primer lugar los datos y sus interrelaciones para determinar la arquitectura de datos subyacente. Cuando esta arquitectura está definida, se definen las salidas a producir y los procesos y entradas necesarios para obtenerlas.
Orientadas a estados y transiciones (tiempo real):	Estas metodologías están dirigidas a la especificación de sistemas en tiempo real y sistemas que tienen que reaccionar continuamente a estímulos internos y externos (eventos o sucesos). Las extensiones de las metodologías de análisis y diseño estructurado de Ward y Mellor y de Hatley y Pirbhai son dos buenos ejemplos de estas metodologías.
Diseño basado en el conocimiento:	Es una aproximación que utiliza técnicas y conceptos de inteligencia artificial para especificar y generar sistemas de información. El método KADS (*Knowledge Acquisition and Development Systems*) es un ejemplo de esta categoría.
Orientadas a objetos:	Estas metodologías se fundamentan en la integración de los dos aspectos de los sistemas de información: datos y procesos. En este paradigma un sistema se concibe como una sociedad de objetos que se comunican entre sí mediante mensajes. El objeto encapsula datos y operaciones. Este enfoque permite un modelado más natural del mundo real y facilita enormemente la reutilización y extensibilidad del software.
Basadas en métodos formales:	Estas metodologías implican una revolución en los procedimientos de desarrollo, ya que a diferencia de las anteriores, estas técnicas se basan en teorías matemáticas que permiten una verdadera aproximación científica y rigurosa al de sistemas de información

Tabla 3.Clasificación de los métodos.

Según Pressman en el año 2002, la clasificación sugerida es la siguiente:

- Métodos formales.
- Concurrentes.
- Basados en Objetos.

Métodos Formales:

Estos métodos permiten al ingeniero del software crear una especificación, sin ambigüedades que sea más completa y constante que las que se utilizan en los métodos

convencionales u orientados a objetos. La teoría de conjuntos y las notaciones lógicas se utilizan para crear una sentencia clara de hechos(o de requisitos).

Los métodos formales son importantes en sistemas críticos para la misión y para la seguridad, un fallo puede pagarse muy caro.

Los métodos formales reducen drásticamente los errores de especificación, y consecuentemente son la base del software que tiene poco errores una vez que le cliente comienza a utilizarlo.

Los métodos formales ofrecen un fundamento para entornos de especificación que dan lugar a modelos de análisis más completos, consistentes y carentes de ambigüedad, que aquellos que se producen empelando métodos convencionales u orientados a objetos. (PRESSMAN, 2002)

Métodos Formales Concurrentes.

Los métodos formales concurrentes son aquellos donde se ejecuta una serie de procesos al mismo tiempo y frecuentemente con un grado elevado de comunicación entre estos procesos.

Los métodos formales que se utilizan para desarrollar sistemas de computadoras son técnicas de base matemática para describir las propiedades del sistema. Estos métodos formales proporcionan marcos de referencia en el seno de los cuales las personas pueden especificar, desarrollar y verificar los sistemas de manera sistemática, en lugar de hacerlo.

Se dice que un método es formal si posee una base matemática estable, que normalmente vendrá dada por un lenguaje formal de especificación. Esta base proporciona una forma de definir de manera precisa nociones tales como la consistencia y completitud, y, lo que es aún más relevante, la especificación, la implementación y la corrección. (PRESSMAN, 2002)

Métodos Formales basados en objetos.

El interés creciente en la tecnología de objeto ha supuesto que los que trabajan en el área de métodos formales hayan comenzado a definir notaciones matemáticas que reflejan las construcciones orientadas a objetos, llamadas clases, herencia e instanciación. Se han propuesto diferentes variantes notaciones existentes principalmente *Object Z.* (PRESSMAN, 2002)

1.3.1 Etapas genéricas en los métodos de desarrollo de software.

"*El trabajo que se asocia a la ingeniería del software se puede dividir en tres fases genéricas, con independencia del área de aplicación, tamaño o complejidad del proyecto. Cada fase se encuentra con una o varias cuestiones de las destacadas anteriormente*". (PRESSMAN 2002).

Dichas fases se encuentran representadas en la figura 5.

Las fases a las cuales hace referencia Pressman en la cita anterior son: fase de definición, fase de desarrollo y fase de mantenimiento. Durante la ***fase de definición*** se trata de identificar qué información debe ser procesada, función y rendimiento que se desea, comportamiento del sistema, interfaces que van de ser establecidas, restricciones de diseño existentes y criterios de validación que se necesitan para definir un sistema correcto. Durante la ***fase de desarrollo*** se intenta definir cómo han de diseñarse las estructuras de datos, implementarse la función dentro de una arquitectura de software, implementarse los detalles procedimentales, caracterizarse las interfaces, traducirse el diseño en un lenguaje de programación o lenguaje no procedimental y realizarse las pruebas. Durante la ***fase de mantenimiento*** es necesario trabajar en función de los cambios asociados a la corrección de errores, a las adaptaciones que se requieren conforme evoluciona el entorno del software y a los cambios que dependen de las mejoras producidas por los requisitos cambiantes del cliente. Estas fases se complementan con un número de actividades protectoras, entre las cuales se incluyen: seguimiento y control del proyecto de software, revisiones técnicas formales, garantía de calidad del software, gestión de configuración del software, preparación y producción de documentos, gestión de reutilización, mediciones y gestión de riesgos.

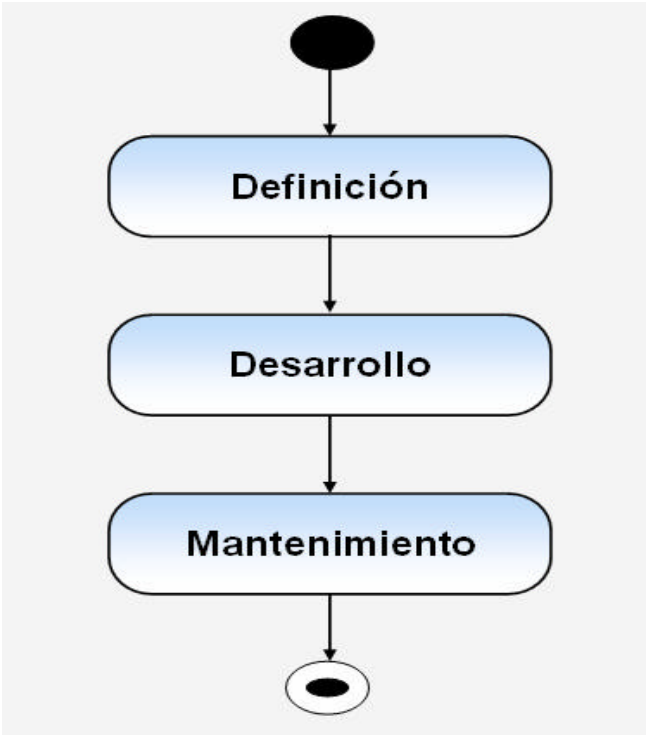


Figura 4. Fases genéricas asociadas a la ingeniería del software

.

Fase de Definición

Según Pressman la etapa de definición se divide en tres tareas fundamentales, representadas (Ver Anexo 8): ingeniería de sistemas o de información, planificación del proyecto del software y análisis de los requisitos. “La ingeniería de sistemas se centra en diversos elementos, analizando, diseñando y organizando esos elementos en un sistema que pueden ser un producto, un servicio o una tecnología para la transformación de información o control de información”. (PRESSMAN 2002).

La ingeniería de sistemas ayuda a traducir las necesidades del cliente en un modelo de sistemas que utiliza uno o varios componentes de hardware, software, personas, bases de datos, documentación y procedimientos. Durante esta etapa se realizaron un conjunto de actividades conocidas como ingeniería de requisitos y la misma comprende la identificación, análisis y negociación, especificación, modelización, validación y gestión de requisitos operacionales. Por su parte la planificación del proyecto del software implica la gestión de proyectos, que comprende la planificación, supervisión y control del personal, del proceso y de los eventos que ocurren durante la evolución del software; también involucra las actividades relacionadas con la estimación como el costo, esfuerzo, recursos y tiempo que supone construir un sistema o producto de software; comprende además las acciones para el análisis y la gestión de riesgo, como la identificación, evaluación de la probabilidad de aparición, la estimación del impacto que pueda producir y el plan de contingencia ante la aparición de problemas; y por último abarca todo lo relacionado con la planificación temporal y el seguimiento del proyecto.

Durante el análisis de los requisitos se especifican las características operacionales del software (función, datos y rendimiento), se indica la interfaz del software con otros elementos del sistema y se establecen las restricciones que debe cumplir el software. En esta se crean modelos, son particionados los problemas y se desarrollan representaciones para mostrar las particularidades de los requisitos y los detalles de implementación. Además, se crean prototipos que permiten la refinación de los requisitos y se desarrolla la especificación de los requisitos del software. En esta fase se utiliza el modelo de datos y de flujos como base para crear el modelo de análisis, se involucran otros diagramas como el de entidad-relación, el de transición de estados, el modelo de comportamiento y el de contenido de datos con un diccionario de datos.

Teniendo en cuenta que la gestión eficaz de un proyecto de software se centra en las cuatro P´s (persona, producto, proceso y proyecto) se hizo necesario que en esta investigación se aludiera a la incidencia de las actividades que forman parte de las tareas correspondientes a la fase de definición definida por Pressman para el desarrollo del software, sobre los elementos que forman parte de dichas cuatro P´s. Para ello fue

necesario agrupar dichas características de acuerdo a su correspondencia con cada uno de estos elementos con el fin de determinar cuáles de ellos toman mayor participación y son imprescindibles para la ejecución de un proyecto determinado.

Fase de Desarrollo:

La etapa de desarrollo se divide en tres tareas fundamentales. El diseño del software comprende el diseño arquitectónico y el de interfaz de usuario. Durante el diseño arquitectónico es necesario el diseño de datos, el cual comprende el modelado de datos, las estructuras de datos, las bases de datos y el almacén de datos; incluye la selección del estilo arquitectónico que se necesita escoger para un sistema determinado, la estructura y las propiedades que ese sistema comprende, así como las interrelaciones que tienen lugar entre todos los componentes arquitectónicos del sistema; el diseño arquitectónico agrupa inicialmente un conjunto de actividades de diseño que conducen a un modelo completo del diseño del software. Por su parte el diseño de interfaz de usuario da especial énfasis a la identificación de los requisitos del usuario, de las tareas y el entorno, para a partir de aquí crear los escenarios de usuario y definir el conjunto de objetos y acciones de la interfaz. “La interfaz de usuario es la ventana del software. En muchos casos, la interfaz modela la percepción que tiene un usuario de la calidad del sistema”. (PRESSMAN 2002). La segunda tarea: generación de código, se centra en el diseño a nivel de componentes, a partir del cual se representan las estructuras de datos, las interfaces y los algoritmos con suficiente detalle como para servir de guía en la generación de códigos fuente de lenguajes de programación. En el caso de las pruebas del software, su objetivo está dirigido a determinar las pruebas que se deben hacer, diseñar la prueba, aplicar las diferentes técnicas de diseño de casos de prueba: prueba de caja blanca y de caja negra y por último la recodificación de errores. (Consultar Anexo 9)

Fase de Mantenimiento:

La fase de mantenimiento se centra en el cambio, contemplando 4 tipos de cambios: Corrección, Adaptación, Mejora y Prevención. “El mantenimiento correctivo cambia el software para corregir los defectos”. “El mantenimiento adaptativo produce modificación en el software para acomodarlo a los cambios de su entorno externos”. El mantenimiento perfectivo lleva al software más allá de sus requisitos funcionales originales”. “El mantenimiento preventivo también llamado reingeniería del software, se debe conducir a permitir que el software sirva para las necesidades de los usuarios finales. En esencia, el mantenimiento preventivo hace cambios en programas de computadora a fin de que se puedan corregir, adaptar y mejorar más fácilmente”. (PRESSMAN 2002) (Consultar Anexo 10)

1.4 Muestra de Métodos utilizados en el entorno productivo mundial.

En los últimos 20 años se han estado utilizando mayoritariamente tres grandes metodologías o métodos de desarrollo de software educativos o que aunque no han sido definidos para este fin pueden utilizarse para el desarrollo de las aplicaciones con estos fines, los cuales se describen brevemente a continuación.

1.4.1 RMM: Metodología de Administración de Relaciones (Relationship Management Methodology).

La metodología RMM, fue desarrollada por T. Isakowitz, en la Universidad de Nueva York en el año 1995. Se realizó la modelación de las aplicaciones a través de RMDM (Relationship Management Data Model), basado en el modelo Entidad – Relación y posee una herramienta CASE denominada: Relationship Management Case Tool – RMCase.

"RMM (Relationship Management Methodology) contiene el diseño y la construcción de aplicaciones hipermedia en un proceso de siete pasos. Es al mismo tiempo un enfoque "top down" y "bottom up". Durante la fase del diseño Entidad – Relación, entidades y relaciones son identificadas las cuales se convertirán en nodos y enlaces en la hipermedia resultante. El segundo paso, diseño de cortes (slices), involucra el agrupamiento de atributos de entidades para la presentación. Los cortes (slices) son "unidades de presentación" que aparecen como páginas de una aplicación hipermedia. La separación del contenido y los aspectos de la presentación no son satisfechos en este paso. RMM especifica la navegación con primitivas de acceso, como enlaces (links), agrupamiento (menus), índices (index) y recorridos guiados (guided tours). La técnica propuesta para el diseño de la interfaz de usuario es la elaboración de maquetas y proptotipos." (Baumeistier, y otros, 2001)

1.4.2 OOHDM: Metodología de Diseño Hipermedia Orientada a Objetos (Object – Oriented Hypermedia Design Methodology).

Desarrollada por Schwabe y Rossi, en la Universidad de Rio de Janeiro, Brasil y Universidad Nacional de la Plata, Buenos Aires respectivamente en el año 1996. Adopta la notación y los mecanismos de abstracción de la Programación Orientada a Objetos (POO) y consta de cuatro pasos para su ejecución: diseño conceptual, diseño navegacional, diseño de interfaz abstracta e implementación. Trabaja la representación a través de los siguientes modelos: Esquema de clases, esquema de navegación, esquema contextual de navegación y vista abstracta de datos.

"El Modelo de Diseño de Hipermedias Orientado a Objetos: OOHDM (Object – Oriented Hypermedia Design Model) comprende cuatro actividades; estas son modelo conceptual,

diseño de navegación, diseño de interfaces abstractas e implementación. Estas actividades son ejecutadas en un estilo de desarrollo mixto a partir de los modelos incremental, iterativo y basado en prototipos. Este método trata a la aplicación como una vista superior al modelo conceptual. El concepto de contexto de navegación es introducido para describir la estructura de navegación. Es un concepto potente que permite diferentes agrupamientos de objetos de navegación con el propósito de navegar en ellos en diferentes contextos. Una notación especial es utilizada para la representación de la estructura de navegación. En trabajos tempranos de investigación, OMT se propuso como la notación para el esquema conceptual; un poco más tarde los trabajos ya utilizan UML. Sin embargo, los diagramas de OOHDM no obedecen los patrones UML, sino que utilizan una notación propia para la perspectiva de los atributos en los diagramas de clase y proponen otros tipos de diagramas para el diseño de la navegación y de las interfaces de usuarios abstractas." (Baumeistier, y otros, 2001)

1.4.3 WSDM: Método de Diseño de Sitios Web (Web Site Design Method).

El Método de Diseño de Sitios Web (WSDM) fue desarrollado por Vrije De Troyer, en la Universidad de Brucelas, en el WISE Research Group, en el año 1997; siendo "(…) un enfoque centrado al usuario definiendo objetos de navegación basados en información de requerimientos del usuario de una aplicación Web. WSDM consiste en tres fases principales: modelación del usuario, diseño conceptual y diseño de implementación. En la fase de modelación del usuario, el usuario potencial del Sitio Web es identificado y clasificado. Diferentes perspectivas son definidas para las clases de usuario, siendo estas diferentes formas en las cuales las clases de usuario utilizan la misma información. El modelo de navegación consiste en un número de rutas de navegación que expresan como los usuarios de perspectiva particular pueden navegar a través de la información disponible. Este método usa su propia notación gráfica para los objetos del modelo de navegación: el diseño de navegación alcanza aplicaciones Web que tengan una marcada estructura jerárquica." (Baumeistier, y otros, 2001)

Conclusiones Parciales

"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).

La necesidad de desarrollar métodos que guíen el desarrollo de productos software, ha conducido a la creación del capítulo que acaba de ser presentado, como referencia fundamental para el análisis de posibles soluciones que contribuyan a la mejor elaboración

de un conjunto de actividades planificadas en un orden lógico para el desarrollo de productos de software educativo en la Universidad de las Ciencias Informáticas.

El estudio del marco conceptual, que rodea al objeto de estudio que rige esta investigación, ha permitido que se tenga una visión general de la situación actual de los productos software en esta institución, la cual ha brindado información por el escaso conocimiento de los métodos más utilizados en el entorno productivo para la modelación del SWE en Cuba.

Aunque las propuestas presentadas no se puedan adecuar íntegramente a las características particulares del tipo de software que se desarrolla en esta institución, identificando los elementos comunes y diferentes entre los métodos existentes en la UCI y estableciendo una comparación entre dichos elementos desarrollados por el producto podría formar parte de la propuesta de ordenamiento lógico de dichos método y organizarlos por su efectividad que será mostrará en el próximo capítulo.

Capítulo 2. Comparación entre métodos para el desarrollo de software educativo.

Introducción.

En la actualidad el desarrollo del software ha aumentado considerablemente, lo que hace que sea muy difícil lograr un posicionamiento y un reconocimiento en el mercado internacional. Esto trae consigo que sea un reto para la industria del software desarrollar las estrategias que le permitan alcanzar un nivel de calidad realmente alto, por lo que se hizo necesario todo un estudio para la elección e implantación del Modelo o Estándar de Calidad indicado. (ALLIANCE 2007). (RAMÍREZ, y otros, 2007)

"El software educativo es uno de los pilares en los que se soporta el sistema educativo a distancia y será la herramienta fundamental de las próximas generaciones de educandos. Son escasos los trabajos de investigación centrados en la problemática del software aplicado a la educación y aunque algunos de ellos son notablemente significativos se carece de un estudio actualizado y en profundidad. Las primeras ideas sobre desarrollo de software educativo aparecen en la década de los 60, lentamente, se desarrollan tres líneas distintas. La primera corresponde a los lenguajes para el aprendizaje y de ella nace el Logo, este lenguaje fue utilizado en un sentido constructivista del aprendizaje. Es de decir, el alumno no descubre el conocimiento, sino que lo construye, en base a su maduración, experiencia física y social. A partir de ahí se ha desarrollado infinidad de software de acuerdo a las diferentes teorías, tanto conductuales, constructivistas como cognitivistas. La segunda línea corresponde a la creación de lenguajes y herramientas que sirvan 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 y las herramientas de autor, el campo desarrollo del software se ha hecho muy complejo, razón por la cual se necesita de una metodología unificada para su desarrollo." (CATALDI, 2000)

En este capítulo se exponen ejemplos de métodos para el desarrollo del software educativo que hoy día se utilizan en la universidad. La carencia de un conjunto de elementos que forman parte y definan pautas a la hora de llevar a cabo la realización de un producto software educativo son algunas de las razones que pudieran comenzar a justificar la necesidad de descripción de un método o priorización de acuerdo a un conjunto de indicadores de estos para su mejor uso en la construcción de este tipo de aplicaciones informáticas.

2.1 ¿Cómo se organiza el desarrollo del software educativo en la UCI?

Desde los inicios de la Universidad de las Ciencias Informáticas se identificó un conjunto de problemas provocados por diversos factores que atentan contra el correcto desarrollo de los proyectos de producción de software educativo. Estas afectaciones fueron provocadas fundamentalmente por la inexperiencia del personal y la falta de procedimientos para guiar a los encargados de dichos proyectos, lo cual se justifica si se tiene en cuenta que estamos en presencia de una nueva institución, en la cual se creó las bases poco a poco para el desarrollo de software, una institución que aún se encuentra en formación y que por tanto existe un desconocimiento en algunas áreas relacionadas con la producción del software . Esta situación empeora cuando se requiere que los productos que sean elaborados correspondan al ámbito educativo, para el cual existen muy pocas metodologías a nivel internacional, y que además no pueden utilizarse de forma directa para la guía de los procesos de desarrollo en la universidad, pues cada proceso de producción se comporta de forma diferente en cada entidad, dependiendo de un conjunto de factores que se asocian al personal involucrado en el proyecto, a la concepción pedagógica, a las características de cada empresa y a su organización.

En la revisión de documentos históricos sobre el marco que se aborda se detectó que "aunque fue elaborada una propuesta para guiar el proceso de desarrollo de software educativo en la UCI, aún siguen existiendo numerosas dificultades en la gestión de los proyectos que se identifican con los mismos, por lo que se ha decidido detectar, cuáles son los aspectos que no han sido contemplados en el flujo de trabajo para el desarrollo de software educativo:" (RAMÍREZ, y otros, 2007)

- La inexistencia de procesos de gestión de configuración y salvas.
- La ausencia de sistemas de chequeo y control a los proyectos por parte de las entidades involucradas.
- La carencia de mecanismos de control relacionados con la seguridad de la información y de los procesos de protección a la propiedad intelectual.
- La falta de definición de los momentos de aprobación de los componentes y los responsables de esta tarea en cada momento del desarrollo del SWE.

Estas deficiencias no se encuentran sólo en el desarrollo de SWE, sino en muchas tipologías de software, y vienen dado en gran medida por la falta de un modelo que evalúe el proceso de desarrollo del mismo y que permita detectar los problemas a tiempo, garantizando así la calidad del producto final.

Es por ello que en la universidad se han tomado medidas con el objetivo de erradicar estas dificultades, dentro de las que se destacan:

- La creación de grupos de calidad por cada una de las facultades.
- El establecimiento de los lineamientos mínimos que deben cumplir todos los proyectos.
- La realización de pruebas al producto final para evaluar la calidad de este.
- El establecimiento de un conjunto de documentos y plantillas que tributan a la organización de cada proyecto.

2.1.1 Sistema Metodológico para el desarrollo de Software Educativo. (SMDSE)

La Universidad de las Ciencias Informáticas ha decidido adoptar una estrategia para la producción de los productos de software educativo y multimedia, basándose en las experiencias de proyectos de este tipo hechos anteriormente, que se concreta en un "**Sistema Metodológico para el desarrollo de Software Educativo**". Dicha estrategia está conformada por un conjunto de elementos que son integrados de forma armónica, en un flujo general de trabajo, compuesto por siete procesos: (Consultar Figura 4)

- Definición de Proyecto.
- Gestión de Requisitos y Análisis.
- Evaluación Técnica.
- Diseño Gráfico
- Gestión de Medias.
- Construcción.
- Aceptación Final del producto.
- Gestión de Configuración.

"La Estrategia Integral para el Trabajo Técnico del SWE se rige por métodos que contienen mejor diseño y fiabilidad que seguridad de la información totalmente integrada al flujo productivo general." (PIÑERO, y otros, 2006)

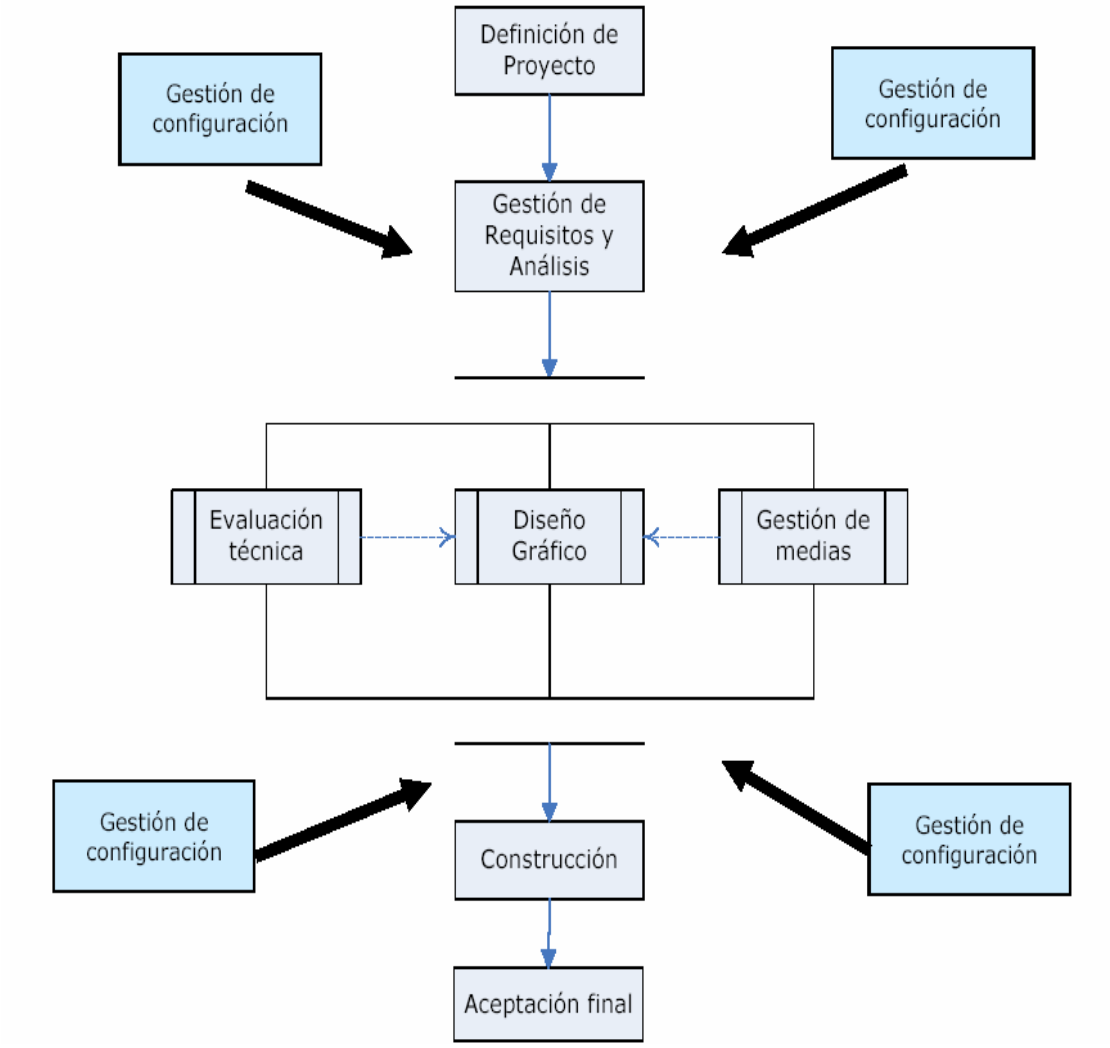


Figura 5. Flujo de trabajo del Sistema Metodológico para el desarrollo de Software Educativo. (PIÑERO, y otros, 2006)

Definición de Proyecto

El cliente entrega su solicitud y el guión de contenidos si lo tiene elaborado. En este flujo se traza la estrategia que se seguirá para el desarrollo de proyecto equipo de proyecto, incluyendo trabajo con el cliente.

Gestión de Requisitos y Análisis

Se hizo un estudio de la solicitud del cliente y del guión de contenido, y se aclaran las posibles dudas sobre la documentación entregada y se revisa el guión hasta tener una total comprensión del mismo, cuando se logra esto se elabora el guión técnico.

Evaluación Técnica

En este proceso se realizó un análisis del proyecto, las especificaciones de los productos a elaborar y se realizó una recomendación de la arquitectura a usar para la producción de software educativo así como de la arquitectura organizativa para acometer la producción.

Gestión de Diseño Gráfico

La gestión de diseño gráfico comprende los procesos de concepción realización del diseño gráfico de un producto de software educativo en función de los requisitos dados en el guión técnico.

Gestión de Medias

La gestión de medias comprende los procesos de búsqueda o producción de determinados recursos audiovisuales en función de los requisitos dados en el guión técnico. Constituye en un flujo de trabajo con alto grado de complejidad.

Construcción

Se realizó la programación para obtener entregables parciales, el cual es sometido a revisiones hasta estar apto para ser entregado al cliente.

Aceptación final del producto.

Una vez realizadas todas las entregas se termina el proceso con una carta de aceptación del producto que manifiesta la aceptación del producto por parte del cliente.

Participan: Cliente, Director de Producción o Vicedecano de producción de la facultad

Resultados: Carta de aceptación del producto

Necesidades: Todas las entregas anteriores aceptadas.

Gestión de Configuración

En esta área se definen las políticas para el desarrollo de los proyecto, además de las herramientas para la gestión de control de versiones, las herramientas de trabajo colaborativo Web, los servidores en los cuales podrán encontrar dichas herramientas, la documentación básica para el uso de estas herramientas, etc. Todas estas definiciones de políticas y procedimientos están en total concordancia con las definiciones establecidas por la Infraestructura productiva para todos los proyectos productivos. Teniendo en cuenta estas políticas generales se han realizado algunas especificaciones para los proyectos de software educativo y multimedia.

En el Sistema Metodológico para el Desarrollo del Software Educativo (SMDSE) mencionado se trabajan con un grupo de artefactos que se mencionan a continuación:

- Lista de chequeo para la etapa de modelación pedagógica.

- Descripción general del software.
- Descripción de las pantallas.
- Descripción de los recursos.
- Acta de conformidad de entrega del guión de contenido.
- Planilla de control de imágenes.
- Planilla de control de sonidos, locuciones y audio.
- Planilla de videos.
- Acta de conformidad de la entrega de los recursos mediático.
- Lista de chequeo para la aceptación del prototipo de software.
- Acta de conformidad de las entregas de los elementos de diseño.

La **lista de chequeo** se utiliza una vez que se concluya la etapa de concepción y se redacte un informe con toda la modelación pedagógica del software. En ella se recoge un conjunto de factores y subfactores necesarios para la evaluar la misma. y una valoración, en caso de que sea necesario, en cada uno de los aspectos y al final una valoración global donde se emitirá un criterio global de la modelación y dar otros elementos que no se tuvieron en cuenta y que enriquecerían el trabajo.

En la **descripción general del software** se detalla la concepción general del producto dando una descripción de sus objetivos, la audiencia para el cual fue concebido especificando en este caso edad, sexo, conocimiento precedente, etc. Además se describe cual es el tema que se va a enseñar expresando su importancia y de que forma fue concebida la estructuración de los contenidos o temáticas. También se hizo alusión de mascotas presentes en el producto que guíen en aprendizaje o apoyen el mismo. Otro de los aspectos que se citan en la descripción general del software es el mapa de navegación que nos da una visión gráfica de la estructuración del contenido y como se puede transitar de una pantalla a otra.

La **descripción de las pantallas** es un artefacto que se desarrollará por cada una de las pantallas descritas en el mapa de navegación que ayudará a describir de forma organizada y precisa el contenido de cada una de las pantallas para su posterior implementación. En esta descripción se especifica el actor que interactúa con la misma así como también el propósito de la misma para dar una idea de porque fue concebida. A estos dos se le suma un listado de los recursos mediáticos que estarán en la pantalla divididos por tipos (textos, imágenes, animación, locuciones, audio, sonidos, videos, diaporamas) y una descripción de la acción (usuario –respuesta) del sistema donde se explique con claridad las diferentes reacciones del sistema a

interacciones del usuario para que posteriormente estas sean implementadas. Se incluye el listado de recursos mediáticos para determinar cuales son los que estarán en la misma.

La **descripción de los recursos mediáticos** no es más que un listado que se formará por cada unos de los recursos mediáticos empleados en la construcción del software que posteriormente serán gestionados o creados por los demás especialistas: diseñadores y audiovisuales, compositores, artistas, etc. Hay que tener en cuenta que para la descripción de cada uno de los recursos se debe describir muy bien puesto esto será el material por el cual los especialistas se guiaran para la gestión de los mismos y sino se describe bien pues esto provocaría que se busque recursos que no son los válidos y se gastaría de forma innecesaria tiempo y recursos. Esta es una de las etapas más costosas en la creación de un software educativo.

Una vez concluida esta etapa se firmará por las partes involucradas un **acta de conformidad del guión de contenido** donde se expresó que todas las partes involucradas en el proceso de desarrollo, y se está de acuerdo con el guión escrito para el desarrollo del producto.

Las **planillas de control** ayudan a controlar el proceso. Cada una de las planillas recoge un conjunto de parámetros que le confieren calidad a cada uno de los recursos mediáticos. La aceptación de cada uno de los recursos mediáticos que se elaboren en esta etapa será evaluada por especialistas.

Acta de conformidad de los recursos mediáticos donde aparecerán los elementos que serán entregados y una firma de las partes que reciben en señal de que se encuentran de acuerdo con los mismos.

Para lograr que en esta etapa la producción de los materiales gráficos (etapa de diseño) contribuya al éxito del software hay que tener en cuenta algunas consideraciones en cada uno de los elementos realizados. Unido a ella una descripción o pauta donde se refleja estas características: el tipo de letra a utilizar, los colores, el tamaño de los objetos, distancia entre ellos, etc. Una vez graficada estas pantallas deberá ser presentadas al cliente y usuarios los cuales a través de una **lista de chequeo** evaluará el prototipo y dirán las consideraciones necesarias al respecto para ello se propone una lista de chequeo la cual tendrán un conjunto de preguntas relacionadas con la interfaz diseñada por el usuario. El mismo podrá seleccionar en cada caso si es excelente, adecuado, o no adecuado. En el caso de seleccionar excelente indicará que la propuesta es aceptada. En caso de que se seleccione adecuada es porque habrá que realizar algunos cambios que el usuario precisara en consideraciones. Si por el contrario se selecciona no adecuado pues no habrá que

realizar consideraciones puesto que el diseñador tendrá que desarrollar una nueva propuesta.

Una vez concluido el conjunto de elementos necesarios para el montaje de una pantalla, los mismos serán expuestos a los clientes y estos expresarán su acuerdo o desacuerdo con los mismos. Cada vez que se realice una entrega al equipo de programación, ésta será avalada con las partes involucradas: el cliente, el programador y el diseñador, y se llenará la siguiente **acta de conformidad.**

(GUTIÉRREZ MOMPIÉ, y otros, 2007)

2.1.1 Nuevo Enfoque para la Producción de Software Educativo en la UCI. (NEPSE).

En el año 2007, como análisis del proceso de desarrollo de software educativo en la UCI, como parte de una investigación científica (PIÑEIRO GÓMEZ, y otros, 2007) se identificaron un conjunto de principios necesarios a tener en cuenta para la construcción de este tipo de software, como resultado de agrupar las actividades principales del Sistema Metodológico para el desarrollo de Software Educativo dentro de las etapas genéricas descritas por Pressman en esta misma memoria: Definición, Desarrollo y Mantenimiento. Alguno de los resultados principales de este investigación fueron los siguientes:

Fase de Definición: Definición del proyecto

- **Solicitud del proyecto:** Establece el proceso de comunicación entre el cliente y representantes de la dirección de producción, así como demás interesados en el desarrollo del proyecto, con el objetivo de realizar la solicitud formal de adquisición de un producto de software educativo. Esta petición queda registrada en una planilla de solicitud de proyecto definida en el “Sistema Metodológico para el desarrollo de software educativo” (SMDSE).
- **Selección del líder de proyecto:** Establece que se debe seleccionar el jefe del equipo (líder), teniendo en cuenta que debe tener habilidades para motivar, organizar y aglutinar personas, además de ser innovador y gran conocedor de su trabajo, con capacidades para resolver cualquier problema que se pueda presentar.
- **Identificar la necesidad educativa:** Se debe especificar la necesidad del programa educativo y seleccionar la teoría educativa a utilizar para el desarrollo del software de acuerdo a esta necesidad.
- **Análisis y gestión de riesgo:** Establece que desde el inicio del proyecto se realice el análisis y gestión del riesgo, el cual comprende la identificación,

estimación, refinamiento y reducción-supervisión-gestión del riesgo, ya que el éxito de un proyecto depende en gran medida de la preparación que se tenga desde el principio sobre las cosas que pueden ir mal en el proyecto, pues su aparición afecta proporcionalmente el plan concebido para el mismo. El resultado de esta actividad debe quedar plasmado en un documento donde figure la lista de los riesgos identificados, además de concebir un plan de contingencia y mitigación por si llegara a ocurrir el problema.

Fase de Definición: Gestión de requisitos y análisis

- **Elaboración del guión técnico:** Establece que una vez estudiada la solicitud del cliente, el guión de contenido, y sean aclaradas las posibles dudas sobre la documentación entregada, se realizó una revisión del guión hasta tener una total comprensión del mismo, cuando se logra esto se elabora el guión técnico.
- **Identificación y clasificación de requisitos:** Establece que a partir del guión de contenido y las entrevistas con el cliente es posible realizar una identificación de requisitos, los cuales deben quedar registrados en el guión técnico, además de ser clasificados en requisitos funcionales o no funcionales dependiendo del papel que jueguen en el sistema.
- **Creación de prototipos del software:** Establece que como no es posible especificar completamente un problema en una etapa tan temprana, se deben crear prototipos de software pues esto contribuye a que se pueda producir un modelo ejecutable del software a partir del cual se pueden refinar los requisitos. Esto se evidencia a través de la creación de pantallas que son especificadas en el guión técnico.
- **Priorizar e integrar los requisitos educativos (pedagógicos) con los del software:** Establece que se deben identificar un conjunto de requerimientos pedagógicos de acuerdo a las necesidades educativas especificadas con anterioridad, relacionados con el contenido y la población estudiantil a la que va dirigido el software e integrarlos al conjunto de requerimientos que han sido especificados con anterioridad.

Fase de Definición: Evaluación técnica

- **Estudio de los requisitos complementarios tanto de software como educativos:** Establece que una vez identificados los requisitos críticos se hizo necesario que se realice un estudio exhaustivo de los requisitos de software y educativos que complementan el software a desarrollar.

- **Definición, análisis y diseño de la arquitectura basándose en la teoría educativa elegida:** Establece que inicialmente se debe realizar una recomendación de la arquitectura apropiada a utilizar para el desarrollo del software educativo de acuerdo a la teoría educativa que fue seleccionada con anterioridad así como de la arquitectura organizativa para acometer la producción. De acuerdo a las características particulares de dicho software se decidirá si la propuesta realizada es factible y a partir de ello se realizaron los análisis necesarios para un posterior diseño de la misma. Esta arquitectura quedará especificada en el documento de diagnóstico de producción perteneciente al sistema metodológico.

Fase de Desarrollo: Evaluación técnica

- **Seleccionar estilo arquitectónico:** Establece que para la construcción de un software se debe seleccionar un patrón arquitectónico, con el cual se obtendrá una visión general de la estructura del software a partir de su arquitectura.
- **Realización de la modelación del diseño:** Establece que a partir del diseño arquitectónico se agrupan inicialmente las actividades de diseño que conducen a la modelación del diseño del software. Para la realización del modelo del diseño es necesario el guión técnico.

Fase de Desarrollo: Diseño gráfico

- **Definir pautas del diseño:** Establece que se debe tener en cuenta la organización de menús, tipos de íconos a usar, efectos a usar, selección de textos, diseño de pantallas y menús, entre otros.
- **Diseñar las interfaces de usuario:** Establece que para diseñar la interfaz de usuario se deben tener en cuenta las características del público al que se dirige el software, las necesidades educativas identificadas y la cumplimentación de las funcionalidades requeridas para la correcta ejecución del software.
- **Definir criterios de navegación:** Establece que dependiendo de la estructura que se deba seguir para lograr los objetivos pedagógicos que se persiguen en el contenido, se deben determinar los criterios de navegación correspondientes a dicha necesidad.
- **Creación de un prototipo a partir de la implementación del modelo de diseño:** Establece que a partir de la implementación del modelación del diseño definido con anterioridad será creado un prototipo funcional del software que se está desarrollando.

- **Evaluación del diseño:** Establece que a partir de la creación de un prototipo de interfaz de usuario, se deberá pasar a la evaluación del mismo, con el objetivo de determinar si cumple o no las necesidades del usuario.

Fase de Desarrollo: Gestión de Medias

- **Búsqueda o producción de recursos audiovisuales:** Establece que a partir de la solicitud de medias realizada con anterioridad, se debe proceder a la búsqueda o producción de los recursos especificados en dicha solicitud, la cual es posible realizarla de acuerdo a los requisitos especificados en el guión técnico.

Fase de Desarrollo: Construcción

Etapa 1: Implementación

- **Realizar diseño a nivel de componentes:** Establece que a partir del diseño de datos, arquitectura definida e interfaz se debe realizar el diseño a nivel de componentes, lo cual es evidenciado en el guión técnico.
- **Generación de código fuente de lenguaje de programación:** Establece que las representaciones detalladas de las estructuras de datos, interfaces y algoritmos servirán de guía en la generación de códigos fuente de lenguaje de programación.
- **Concebir la integración de los módulos:** Establece que una vez elaborados los módulos se debe pasar a la integración de los mismos para así lograr el correcto funcionamiento del software.

Etapa 2: Prueba

- **Diseño de casos de pruebas:** Establece que se deben diseñar pruebas que tengan la mayor probabilidad de encontrar el mayor número de errores con la mínima cantidad de esfuerzo y tiempo posible. Estas pruebas es lo que conocemos como pruebas de caja negra y pruebas de caja blanca.
- **Aplicación de pruebas de caja blanca:** Establece que se deben realizar pruebas dirigidas al código con el objetivo de encontrar errores procedimentales. Con estas pruebas se comprueban los caminos lógicos del software y se proponen casos de prueba que ejerciten conjuntos específicos de condiciones y/o bucles. Permite que se pueda examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado.
- **Aplicación de pruebas de caja negra:** Establece que se deben realizar pruebas sobre la interfaz del software para detectar los errores que puedan

existir en la misma. Estos pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada, que se produce un resultado correcto y que la integridad de la información externa se mantiene. Dichas pruebas se centran en los requisitos funcionales del software.

Etapa 3: Revisión técnica

- **Revisión técnica del entregable:** Establece que se debe realizar una revisión de la versión del producto tomando como referencia el guión técnico, los casos de prueba y las pautas técnicas.
- **Realización de un plan de corrección de errores:** Establece que una vez detectado los errores en las revisiones técnicas debe ser realizado inmediatamente un plan para corregir dichos errores.

Fase de Mantenimiento: Construcción

Etapa 1: Corrección de errores

- **Se corrigen los errores detectados:** Durante la revisión técnica, a partir del listado de errores y del guión técnico son corregidos los defectos que fueron detectados en el software.

Etapa 2: Entrega y revisión del cliente

- **Entrega del producto al cliente:** Establece que una vez corregidos los errores se realizó una revisión de contenido del entregable por parte del cliente. En caso de que este entregable contenga errores se define un plan de corrección y se regresa a la **Etapa 1** (corrección de errores), de lo contrario se entrega al cliente una carta de aceptación para la entrega parcial.

Fase de Mantenimiento: Aceptación final del producto

- **Aceptación del producto:** A las entregas realizadas se le hacen revisiones y pruebas de aceptación. Las revisiones y prubas de aceptación deben tener en cuenta los resultados de las revisiones conjuntas, auditorias, pruebas del software, entre otros. Se debe documentar los resultados de las pruebas y revisiones de aceptación. Este proceso culmina con una carta de aceptación del producto emitida por el cliente.

Actividades de Apoyo

- **Revisiones técnicas formales:** Establece que se deben efectuar revisiones técnicas con el fin de evaluar los productos o servicios software bajo consideración.

- **Realización de la Gestión de configuración del software:** Establece que se deben definir las políticas para el desarrollo de los proyectos, las herramientas para la gestión de control de versiones, las herramientas de trabajo colaborativo Web, los servidores donde se pueden encontrar dichas herramientas., la documentación básica para su uso. Este proceso de gestión de la configuración es ampliado en el Sistema Metodológico.
- **Preparación y producción de documentación didáctica:** Se debe planificar la documentación didáctica y elaborar una guía didáctica, con ejemplos de uso para los futuros usuarios del software. Dicha documentación debe tener la información detallada de las características del programa que haya sido realizado, su forma de uso y posibilidades didácticas. Se debe adjuntar esta información didáctica, los caminos pedagógicos, teorías del aprendizaje y programación didáctica al programa que se ha ido desarrollando.
- **Realizar gestión de reutilización:** Establece que se debe definir todo lo referente a la gestión de elementos reutilizables.
- **Seguimiento de la gestión de riesgo:** Establece que a medida que avanza el proyecto se debe realizar un seguimiento de los riesgos que fueron planteados desde el inicio del mismo para determinar si estos se van haciendo más o menos probables. Se supervisa si se están realizando correctamente los pasos de reducción de riesgo.

2.2 Aspectos teóricos para la comparación.

Para poder establecer una comparación entre los métodos desarrollados por el *Sistema Metodológico para el desarrollo de Software Educativo* y el *Nuevo enfoque para la producción de software educativo en la UCI (NEPSE)* se deben de identificar las características comunes y diferentes de acuerdo a los siguientes indicadores y valores de ponderación expresados en la siguiente tabla:

Tabla 4: Indicadores, resultados y valores de ponderación para la comparación entre los métodos de desarrollo de software utilizados en la UCI.

No	Indicador	Resultado con valor de ponderación 3	Resultado con valor de ponderación 2	Resultado con valor de ponderación 1
1	Actividades conformantes del método.	Agiles	Ligeras	Pesadas

2	Artefactos necesarios para el desarrollo de las actividades.	Muchos	Suficientes	Pocos
3	Artefactos que se generan como consecuencia de las actividades.	Muchos	Suficientes	Pocos
4	Correspondencia con las etapas genéricas definidas para el desarrollo de productos informáticos.	Alta	Mediana	Baja
5	Integración de las soluciones que se pueden generar.	Alta	Mediana	Baja
6	Seguimiento del producto.	Alto	Parcial	Bajo
7	Adecuación al contexto productivo para la educación.	Alta	Parcial	Baja
8	Capacidad de utilización de lenguajes de programación en la generación de las soluciones.	Alta	Parcial	Baja
9	Esfuerzo por el personal necesario para acometer las actividades definidas.	Bajo	Mediano	Alto
10	Utilización de estándares de calidad y definición de procesos utilizados.	Alta	Parcial	Baja
11	Áreas de la Ingeniería de Software que abarca.	Todas	Algunas	Ninguna

2.2.1 Análisis de los indicadores en el Sistema Metodológico para el desarrollo de Software (SMDSE).

En la tablas se muestran los parámetros que se establecieron para la comparación del método SMDSE, se analizó cada indicador teniendo en cuenta sus perspectivas para el desarrollo del software educativo y las aplicaciones de dichas actividades a la hora de construir el mismo, en el cual se le asignó un valor cualitativo según cada parámetro analizado que tuviera correspondencia con todas las acciones que se emplearon en el método:

- Actividades conformantes del método**:** este indicador establece todas las actividades del método así como, la cantidad del conjunto de operaciones o tareas secuenciales en tiempo y espacio propias que son ejecutadas por un desarrollador, que se deben de seguir para el cumplimiento del método.
- Artefactos necesarios para el desarrollo de las actividades: este indicador establece la cantidad de productos tangibles indispensables del proyecto que son producidos, modificados y usados por las actividades; pueden ser modelos, elementos entre modelos, documentos, código fuentes y ejecutables.
- Artefactos que se generan como consecuencia de las actividades: establece los productos tangibles del proyecto que son producidos, modificados y usados que se generan como consecuencia de los artefactos necesarios para el desarrollo de las actividades.
- Correspondencia con las etapas genéricas definidas para el desarrollo de productos informáticos**:** muestra si existe compatibilidad entre los estándares internacionales establecidos para el desarrollo del software y los estándares utilizados en la UCI basados en las etapas genéricas definidas por Pressman para hacer un producto software con calidad.
- Integración de las soluciones que se pueden generar***:*** este establece que las soluciones que se obtengan de este método puedan integrarse a otros sistemas de software educativo.
- Seguimiento del producto*:* este indicador establece el comportamiento del desarrollo del proceso del flujo de trabajo y las etapas genéricas por el cual atraviesa el software.
- Adecuación al contexto productivo para la educación*:* este indicador muestra el acercamiento y la fidelidad del contexto productivo al proceso docente-educativo para la creación de este tipo de software
- Capacidad de utilización de lenguajes de programación en la generación de las soluciones: indica la utilización de la diversidad de lenguajes de programación para la confección del producto y la capacidad de integración de los mismos.
- Esfuerzo por el personal necesario para acometer las actividades definidas*:* este indicador muestra que el personal destinado a la producción, se determine el empeño y la capacidad en llevar adelante el producto, y terminarlo en fecha establecida.
- Utilización de estándares de calidad y definición de procesos utilizados: este indicador muestra los niveles estándares de calidad como son los tipos de

prueba de software para la corrección de errores u omisión de funcionalidades del software.

- Áreas de la Ingeniería de Software que abarca: este indicador muestra si los métodos cumplen con todas las actividades del proceso de la ingeniería del software.

Indicador	Resultado	Valor de Ponderación
Actividades conformantes del método	Ligera	2
Artefactos necesarios para el desarrollo de las actividades	Suficientes	2
Artefactos que se generan como consecuencia de las actividades:	Pocos	3
Correspondencia con las etapas genéricas definidas para el desarrollo de productos informáticos.	Mediana	2
Integración de las soluciones que se pueden generar:	Alta	3
Seguimiento del producto	Parcial	2
Adecuación al contexto productivo para la educación	Alta	3
Capacidad de utilización de lenguajes de programación en la generación de las soluciones	Alta	3
Esfuerzo por el personal necesario para acometer las actividades definidas	Alta	3
Utilización de estándares de calidad y definición de procesos utilizados:	Parcial	2
Áreas de la Ingeniería de Software que abarca	Algunas	2
Totales		**27**

Tabla 5 Valores de los indicadores por ponderación el Método SMDSE.

En la tabla 5 del Métodos en el *Sistema Metodológico para el desarrollo de Software Educativo* (SMDSE) se realizó las siguientes valoraciones:

- En el indicador **Actividades conformantes del método,** se valoró de ligera, porque este método tiene definidas generales, por tanto insuficientes las actividades para el desarrollo del método.
- En el indicador **Artefactos necesarios para el desarrollo de las actividades**, se valoró de suficiente, porque el hecho de tener pocas actividades no se generan los artefactos necesarios para el posterior iteración y mantenimiento del software.
- En el indicador **Artefactos que se generan como consecuencia de las actividades,** se valoró de pocos, porque lleva implícito insuficientes artefactos necesarios para el para la modelación del software.
- En el indicador **Correspondencia con las etapas genéricas definidas para el desarrollo de productos informáticos**, se valoró de mediana, porque si

existe escasa compatibilidad entre los estándares internacionales establecidos para el desarrollo del software.

- En el indicador **Integración de las soluciones que se pueden generar**, se valoró alta, porque las soluciones que se obtengan de este método puedan integrarse a otros sistemas de software educativo.
- En el indicador **Seguimiento del producto**, se valoró parcial, porque no contempla todas las actividades del desarrollo del proceso del flujo de trabajo y las etapas genéricas por el cual atraviesa el software.
- En el indicador **Adecuación al contexto productivo para la educación**, se valoró de alta, porque existe un altísimo acercamiento y fidelidad entre el contexto productivo y el proceso docente-educativo para la creación de este tipo de software.
- En el indicador **Capacidad de utilización de lenguajes de programación en la generación de las soluciones,** se valoró de alta, porque se utilizaron diferentes lenguajes de programación y la integración para la confección del producto y la capacidad de integración de los mismos.
- En el indicador **Esfuerzo por el personal necesario para acometer las actividades definidas,** se valoró de alta, porque el personal destinado a la producción no tiene las actividades especializadas lo que produce que se necesite mayor empeño y capacidad en llevar adelante el producto, y terminarlo en fecha establecida**.**
- En el indicador **Utilización de estándares de calidad y definición de procesos utilizados,** se valoró de parcial, porque no se mostraron todos los estándares de calidad, como son los tipos de prueba de software para la corrección de errores u omisión de funcionalidades del software.
- En el **indicador Áreas de la Ingeniería de Software que abarca**, se valoró de algunas, porque el método no cumple con todas las actividades del proceso de desarrollo de la ingeniería del software.

2.2.2 Análisis de los indicadores en el Nuevo enfoque para la producción de software educativo en la UCI (NEPSE).

- Actividades conformantes del método:
- Artefactos necesarios para el desarrollo de las actividades:
- Artefactos que se generan como consecuencia de las actividades:
- Correspondencia con las etapas genéricas definidas para el desarrollo de productos informáticos:

- Integración de las soluciones que se pueden generar:
- Seguimiento del producto:
- Adecuación al contexto productivo para la educación:
- Capacidad de utilización de lenguajes de programación en la generación de las soluciones:
- Esfuerzo por el personal necesario para acometer las actividades definidas:
- Utilización de estándares de calidad y definición de procesos utilizados:
- Áreas de la Ingeniería de Software que abarca:

Indicador	**Resultado**	**Valor de Ponderación**
Actividades conformantes del método	Ágiles	3
Artefactos necesarios para el desarrollo de las actividades	Muchos	3
Artefactos que se generan como consecuencia de las actividades:	Muchos	3
Correspondencia con las etapas genéricas definidas para el desarrollo de productos informáticos.	Alta	3
Integración de las soluciones que se pueden generar:	Alta	3
Seguimiento del producto.	Alta	3
Adecuación al contexto productivo para la educación:	Alta	3
Capacidad de utilización de lenguajes de programación en la generación de las soluciones	Alta	3
Esfuerzo por el personal necesario para acometer las actividades definidas:	Bajo	3
Utilización de estándares de calidad y definición de procesos utilizados:	Todas	3
Áreas de la Ingeniería de Software que abarca	Alta	3
Totales		**33**

Tabla 6: Valores de los indicadores por ponderación el Método NEPSE

- En indicador **Actividades conformantes del método,** se valoró de ágiles, porque este método tiene definidas las actividades específicas, por tanto hay suficientes actividades para guiar paso a paso el desarrollo del software educativo.
- En el indicador **Artefactos necesarios para el desarrollo de las actividades**, se valoró de muchos, porque el hecho de tener muchas actividades específicas se generan los artefactos necesarios para el mantenimiento del software.
- En el indicador **Artefactos que se generan como consecuencia de las actividades,** se valoró de muchos, porque lleva implícito los artefactos necesarios para la modelación del software.
- En el indicador **Correspondencia con las etapas genéricas definidas para el desarrollo de productos informáticos**, se valoró de alta, porque existe

compatibilidad entre los estándares internacionales con los del método establecido para el desarrollo del software.

- En el indicador **Integración de las soluciones que se pueden generar,** se valoró alta, porque las soluciones que se obtuvieron de este método puedan integrarse a otros sistemas de software educativo.
- En el indicador **Seguimiento del producto**, se valoró de alta, porque contempla todas las actividades del desarrollo del proceso del flujo de trabajo y las etapas genéricas por el cual atraviesa el software.
- En el indicador **Adecuación al contexto productivo para la educación**, se valoró de alta, porque existe un altísimo acercamiento y fidelidad entre el contexto productivo y el proceso docente-educativo para la creación de este tipo de software.
- En el indicador **Capacidad de utilización de lenguajes de programación en la generación de las soluciones,** se valoró de alta, porque se utilizaron diferentes lenguajes de programación y la integración para la confección del producto y la capacidad de integración de los mismos.
- En el indicador **Esfuerzo por el personal necesario para acometer las actividades definidas,** se valoró de bajo, porque el personal destinado a la producción tiene las actividades especializadas lo que produce que se haga un producto con calidad y terminado en la fecha establecida**.**
- En el indicador **Utilización de estándares de calidad y definición de procesos utilizados**, se valoró de todas, porque se mostraron todos los estándares de calidad, como son los tipos de prueba de software para la corrección de errores u omisión de funcionalidades del software.
- En el indicador **Áreas de la Ingeniería de Software que abarca**, se valoró de alta, porque el método cumple con todas las actividades del proceso de desarrollo de la ingeniería del software.

2.3 Ordenamiento por prioridad de uso de los métodos utilizados para el desarrollo del software educativo en la UCI.

Indicador	SMDSE		NEPSE	
	Resultado y valor	Prioridad	Resultado y valor	Prioridad
Actividades conformantes del método	Ligera – 2	2	Ágiles-3	1
Artefactos necesarios para el	Suficientes- 2	2	Muchos-3	1

desarrollo de las actividades				
Artefactos que se generan como consecuencia de las actividades:	Pocos- 3	2	Muchos-3	1
Correspondencia con las etapas genéricas definidas para el desarrollo de productos informáticos.	Mediana-2	2	Alta-3	1
Integración de las soluciones que se pueden generar:	Alta-3	1	Alta-3	1
Seguimiento del producto	Parcial-2	2	Alta-3	1
Adecuación al contexto productivo para la educación:	Alta-3	1	Alta-3	1
Capacidad de utilización de lenguajes de programación en la generación de las soluciones	Alta-3	1	Alta-3	1
Esfuerzo por el personal necesario para acometer las actividades definidas:	Alta-3	2	Bajo-3	1
Utilización de estándares de calidad y definición de procesos utilizados:	Parcial-2	2	Todas-3	1
Áreas de la Ingeniería de Software que abarca	Algunas-2	2	Alta-3	1
Totales	**27**	**19**	**33**	**11**

Tabla 7: Resultados de la comparación por priorización entre SMDSE y NEPSE

Con la premisa de cumplir el objetivo que ha guiado el desarrollo de esta investigación se hizo necesario realizar una referencia sintetizada de las bases sobre las cuales descansa la solución de la misma

Inicialmente se realizó un estudio riguroso de un conjunto de modelos, métodos y procesos de desarrollo de software, dentro de los cuales se hizo especial énfasis en los dedicados al desarrollo de software educativo, por sus aspectos novedosos en el ámbito pedagógico-didáctico. Posteriormente, se realizó un estudio de los métodos para el desarrollo de software nacionales e internacionales, y finalmente, constituyendo el eslabón fundamental para conformar la solución de esta investigación, se tomó como referencia la comparación realizada entre las actividades contenidas en las fases del NEPSE y las actividades contempladas en el Sistema Metodológico para el Desarrollo del Software Educativo en la UCI (SMDSE).

Analizando los métodos NEPSE y SMDSE contextualizados a la UCI (con los indicadores: *Actividades conformantes del método, Artefactos necesarios para el desarrollo de las actividades, Artefactos que se generan como consecuencia de las actividades, Correspondencia con las etapas genéricas definidas para el desarrollo de productos informáticos, Integración de las soluciones que se pueden generar, Seguimiento del producto, Adecuación al contexto productivo para la educación,*

Capacidad de utilización de lenguajes de programación en la generación de las soluciones, Esfuerzo por el personal necesario para acometer las actividades definidas, Utilización de estándares de calidad y definición de procesos utilizados, Áreas de la Ingeniería de Software que abarca) establecidos en las tablas, se ordenaron por su prioridad. Nuevo enfoque para la producción de software educativo en la UCI (NEPSE) obtuvo una puntuación de 11 puntos en la tabla de prioridad y 33 puntos en la tabla de los indicadores por ponderación; por otro lado, el Sistema Metodológico para el desarrollo de Software Educativo (SMDSE) obtuvo 22 puntos en la tabla de prioridad y 27 puntos en la tabla de los indicadores por ponderación. Por lo tanto se concluye que el método Nuevo enfoque para la producción de software educativo en la UCI (NEPSE) adquiere la prioridad número uno, pues explica claramente todos los aspectos necesarios para construir un software educativo, se logre que los procesos productivos se realicen según un marco de trabajo factible, donde se disminuyan los costos y el tiempo, y se brinde información a la entidad para realizar mejoras a partir de los errores encontrados y además se ayude grandemente a la mitigación de los riesgos. Se le otorga la segunda prioridad al método *"Sistema Metodológico para el desarrollo de Software Educativo (SMDSE)"* pues este contempla un grupo de aspectos pedagógicos que son de gran importancia pero exime un significativo número de actividades técnicas (entiéndase que esté bien estructurado, soportado sobre una buena herramienta y que haya disminuido su costo y esfuerzo en su construcción), lo provoca deficiencia en la construcción del software.

Conclusiones parciales.

1- El desarrollo del capítulo presentado ha permitido que la comparación establecida entre las actividades y tareas contempladas en las fases definidas por el *"Nuevo enfoque para la Producción de Software Educativo en la UCI"*. (NEPSE) unido a las actividades especificadas en el *"Sistema Metodológico para el desarrollo de Software Educativo"* permitió el ordenamiento de los mismos teniendo en cuenta los principios trazados por cada método de desarrollo cumpliendo de esta forma el objetivo por el cual se trabajó en esta investigación.

2- Las encuestas y entrevistas realizadas a líderes de proyectos de software educativo, arrojaron un significativo número de deficiencias que atentan contra el correcto desarrollo de los proyectos productivos, del desconocimiento de estos métodos existentes lo cual conduce a la necesidad de analizar este ordenamiento y aplicar dichos métodos para mejorar la producción de software educativo en la UCI.

Conclusiones Generales

El desarrollo de esta investigación abordó temas relacionados con el proceso del software educativo en la UCI, por lo que se valoró que existe una pobre aplicación de modelos, métodos y procesos de desarrollo de software en los proyectos productivos de software educativo. Es necesario no solo crear una cultura de desarrollo industrial, sino darle el máximo de atención a los desarrolladores y a los riesgos potenciales que pueden aparecer ante el afán de trabajar sin preparación alguna. Aunque la UCI ha adoptado una estrategia que se concreta en un "Sistema Metodológico para el Desarrollo de Software Educativo", es importante destacar que esta no contempla un significativo número de actividades que son necesarias para el desarrollo de software educativo, además resalta el desconocimiento que sobre la misma se posee por parte de los líderes y desarrolladores de los proyectos de software educativo. Teniendo en cuenta que no existe un método de desarrollo de software ideal o universal, se puede recurrir al estudio de una de las soluciones existentes en la UCI (*"Nuevo enfoque para la Producción de Software Educativo en la UCI")* para la elaboración de productos software educativo en función de aplicarlo en la producción.

El estudio inexorable de un conjunto de modelos, métodos y procesos de desarrollo del software, permitió el ordenamiento después de haber hecho una comparación entre los métodos existentes para que guíen el proceso productivo de software educativo en la Universidad de las Ciencias Informáticas, lo cual constituye el objetivo fundamental que guió esta investigación.

Recomendaciones

Al realizar un estudio profundo de un significativo número de aspectos que contribuyeron a la confección de un método para la modelación del desarrollo de software educativo que guíe este proceso en la UCI, se hizo necesario que se listen un grupo de recomendaciones que permita mejorar considerablemente la producción de software educativo con mayor eficacia y alta calidad.

1. Preparar a los líderes y grupos de trabajo de desarrollo de proyectos en este caso estudiantes de la facultades que desarrollan software educativo sobre todo lo relacionado con los modelos y métodos de desarrollo de este tipo de software, con el fin de que adquieran un conocimiento amplio en la elaboración de un producto SW, para lograr que este salga con calidad al mercado.
2. Profundizar en el estudio de las metodologías de desarrollo de software con el objetivo de analizar el método NEPSE que ha sido propuesto como de solución para contribuir a la minimización del tiempo de desarrollo del producto con mayor calidad.
3. Analizar el ordenamiento de los métodos para el desarrollo del software educativo y aplicarlo en áreas de la universidad y en otras empresas dedicadas al desarrollo de productos de este tipo.

Referencias bibliográficas

HEVIA, D. C. I. A. E. C. *EL PAPEL DE LAS TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES (TICs) EN EL PROCESO DE ENSEÑANZA APRENDIZAJE A COMIENZOS DEL SIGLO XXI*.

JACOBSON, I.; G. BOOCH*, et al.* 2003. *El Proceso Unificado de Desarrollo de Software*. Addison - Wesley (Edición en español por la Pearson Educación S.A. traducido de The Unified Software Development Process, 1999). Madrid, 2000. p.

LAMAS, M. R. R. *Introducción a la Informática Educativa* ISPJAE, Ciudad de la Habana, 2000.

MANCEBO, D. E. *Proyecto Docente*, 1999. [http://www.infor.uva.es/~descuder/docencia/pd/pd.html]

MARTÍNEZ, I. Y.; PIÑERO, Y.*,* *et al. Sistema metodológico para el desarrollo del Software Educativo*. Ciudad de la Habana, Universidad de las Ciencias Informáticas, 2007. p.

PALACIO, J. *Gestión y procesos en empresas de software*, 2005. [http://www.navegapolis.net]

PIATTINI, M. *Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión*, 1996.

PRESSMAN, R. S. *Ingeniería del Software. Un enfoque practico*. Quinta edicion Mc Graw-Hill/Interamericana de España, S.A, 2002. p.

ZILBERSTEIN, J.; PORTELA*, et al.* *Didáctica Integradora de las Ciencias. Experiencia cubana*. en. Editorial Academia, Cuba. 1999. p.

Bibliografía

ÁLVAREZ, L y RODRÍGUEZ, C. 2006.
http://www.monografias.com/trabajos31/software-educativo-cuba/software-educativo-cuba.shtml#softeducat. *http://www.monografias.com/trabajos31/software-educativo-cuba/software-educativo-cuba.shtml#softeducat.* [En línea] 2006.

BATES, T. 2001. *" Como gestionar el cambio tecnológico. Estrategias para los responsables de centros universitarios2.* PRIMERA. s.l. : Gedisa, 2001.

BLAYA. 2006.
http://www.upm.es/innovacion/calidad/documentos/Gestion_procesos.pp. [En línea] 2006.

CATALDI, Z. 2000. *" Metodología de diseño, desarrollo y evaluación de software educativo".* 2000.

CIUDAD RICARDO, FEBE. 2007. *"ApEM – L como una nueva solución a la modelación de aplicaciones educativas multimedias en la UCI"*. Tesis en opción al grado científico de Máster en Informática Aplicada. Universidad de las Ciencias Informáticas. La Habana : s.n., 2007.

CORRALES DIAZ,. 1994.
http://iteso.mx/~carlosc/pagina/documentos/multidef.htm#inicio. *http://iteso.mx/~carlosc/pagina/documentos/multidef.htm#inicio.* [En línea] 1994.

FERNÁNDEZ, A. 2000. *" El formador de Formación Profesional y Ocupacional. Octaedro". .* Barcelona : s.n., 2000.

Gutiérrez Mompié, L y Deniz, Báez. 2007. *"Estrategia para la protección legal durante el proceso de producción de Software Educativo en la Universidad de las Ciencias Informáticas".* LA HABANA : s.n., 2007.

GUTIÉRREZ MOMPIÉ, L y DENIZ, BAÉZ. 2007. *"Modelo de Producción de Software Educativo en la Universidad".* Habana : s.n., 2007.

HEVIA. *" EL PAPEL DE LAS TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES (TICs) EN EL PROCESO DE ENSEÑANZA APRENDIZAJE A COMIENZOS DEL SIGLO XXI".*

JACOBSON, I y BOOCH, G. 2000. *El Proceso Unificado de Desarrollo de Software. Addison - Wesley (Edición en español por la Pearson Educación S.A. traducido de The Unified Software Development Process, 1999).* Madrid : s.n., 2000.

MANCEBO, D. 1999. http://www.infor.uva.es/~descuder/docencia/pd/pd.html. [En línea] 1999.

MARQUÉS, P. 1998. *"La informática como medio didáctico: software educativo, posibilidades e integración curricular.".* 1998. Vols. Volume, 93-109 DOI.

MARQUÉS, P. (1995). *"Guía de uso y metodología de diseño"* (Estel ed.). Barcelona, España.

MARQUÉS, Pere. 1996. *"El Software Educativo".* Barcelona : s.n., 1996.

MARTÍNEZ, R y SAULEDA, N. 1993. *"La evaluación de software educativo en el escenario de la evolución de los paradigmas educativos." Volume, 161-174 DOI:.* 1993.

PALACIO, J. 2005. http://www.navegapolis.net. *Gestión y procesos en empresas de software [http://www.navegapolis.net].* [En línea] 2005.

PASTOR, JUAN ANTONIO. 1997. http://www.ucm.es/info/multidoc/multidoc/revista/cuad6-7/saorin.htm. [En línea] 1997.

PÉREZ, YANEISIS. 2007. *"Metodologías para el desarrollo del Software Educativo: Un estudio comparativo".* Cuidad de la Habana : s.n., 2007.

PIATTINI, M. 1996. *"Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión.".* 1996. Vol. DOI.

PIÑA, J.L, CALZADO, M y PIÑA, R. 2007. \\ponencias\tsem\PDF\Multimedia sobre el hardware de las comput-4212942593\Multimedia sobre el hardware de las computadoras.pdf. *"Multimedia sobre el hardware de las computadoras".* [En línea] 2007.

PIÑERO, Y y MARTÍNEZ, I. 2006. *."Sistema metodológico para el desarrollo del Software Educativo". .* La Habana : s.n., 2006.

PIÑEIRO GÓMEZ, YORDANIS and GALLARDO COLLAZO, JAQUELINE. 2007. *¨PRINCIPIOS ESTRATÉGICOS PARA LA GUÍA DEL PROCESO DE DESARROLLO DE SOFTWARE EDUCATIVO EN LA UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS ¨.* Tesis en opción al título de Ingeniero en Ciencias Informáticas. Universidad de las Ciencias Informáticas. La Habana : s.n., 2007.

PRESSMAN, ROGER. 2002. *" Ingeniería del Software. Un enfoque práctico.".* Quinta Edición. 2002. Vol. 1.

PUELLO, A. (2002). "*Foreign Language Teaching" .*

RAMÍREZ, R y MARTÍNEZ, Y. 2007. *"Modelo de Evaluacion del Proceso de desarrollo del Software Educativo".* Habana : s.n., 2007.

RODRÍGEZ, RAÚL. 2000. *Introducción a la Informática Educativa (conferencia).* s.l. : ISPJAE, 2000.

SMORVILLE, I. 2002. *"Ingeniería del Software.".* 2002. Vol. DOI.

ZILBERSTEIN, J y MACPHERSON, P. 1999. *"Didáctica Integradora de las Ciencias". .* s.l. : Academia, 1999.

Zabala. (2000). " *La Ingenieria del Software*". Tesis en opción al título de grado científico de Máster en Sistemas Informáticos. Barcelona.

Anexos

Anexo 1: Entrevista realizada a líderes de proyecto.

1. ¿Cómo fueron seleccionados los estudiantes y/o profesores que pertenecen al proyecto?
2. ¿Los integrantes del proyecto tenían preparación previa, se encontraban familiarizados con el lenguaje de programación, las herramientas a utilizar para la elaboración del producto requerido?
3. ¿Una vez en el proyecto, cómo se gestiona la preparación de los integrantes del mismo?
4. ¿Cómo valorarías la preparación y motivación de los integrantes del proyecto?
5. ¿Tiene usted experiencia alguna dirigiendo proyectos de esta índole?
6. ¿Cómo fue seleccionado usted para ocupar el cargo de líder de proyecto?
7. ¿Cuáles son las tareas que usted debería cumplir como líder de proyecto?
8. ¿Cómo esta organizado el equipo de desarrollo de software?
9. ¿Cómo se comporta la comunicación entre los miembros del equipo?
10. ¿Al inicio del proyecto se realizaron estimaciones cuantitativas para determinar el tiempo aproximado que les llevaría la construcción del producto?
11. ¿Se concibe un plan organizado para el desarrollo del proyecto?
12. ¿Desde un inicio de establece el ámbito del software?
13. ¿Para la elaboración del software como descomponen el problema?
14. ¿Se realizaron estimaciones para determinar el costo del producto?
15. ¿Seleccionan algún modelo de proceso apropiado para la ingeniería del software que debe aplicar el equipo de proyecto?
16. ¿Sigue usted alguna metodología de desarrollo de software para la elaboración del producto?
17. ¿Cómo usted tiene organizado al equipo de desarrollo, Cuáles roles tiene definidos?
18. ¿Cómo se comporta el flujo de trabajo que ustedes siguen para la elaboración del producto?
19. ¿Se realizó al inicio del proyecto una gestión de riesgo del proyecto?
20. ¿Existe una comunicación fluida entre los desarrolladores y el cliente?

Anexo 2: Encuesta realizada a líderes de proyecto.

Fecha de enviada:

Fecha de respondida:

Nombre y apellidos del encuestado:

Nombre del proyecto:

Preguntas:

1. ¿Cuántos métodos (no metodologías) usted conoce para el desarrollo de SWE? Menciónelos.

2. Se utilizan métodos para el desarrollo del SWE.

_____si _____no

3. ¿En caso de utilizar algún método, cuál es el empleado?

_____

4. ¿Qué tipología de SWE se producen en la universidad? Marque con una x

_____Ejercitación ____Tutorial ____Base de Datos

____Libro ____Simulador ____Juego

____Herramientas

5. Existe la participación de un evaluador educativo o pedagógico en la elaboración del producto SWE.

____ si ____ no

6. ¿En que momento se realiza la evaluación?. Marque con una x.

____Durante el proceso de diseño y desarrollo (con el fin de corregir y perfeccionar el programa).

____Durante su utilización real por los usuarios (para juzgar su eficiencia y los resultados que con él se obtienen).

7. ¿Se evalúa el programa como objeto pedagógico? ___Si ____No

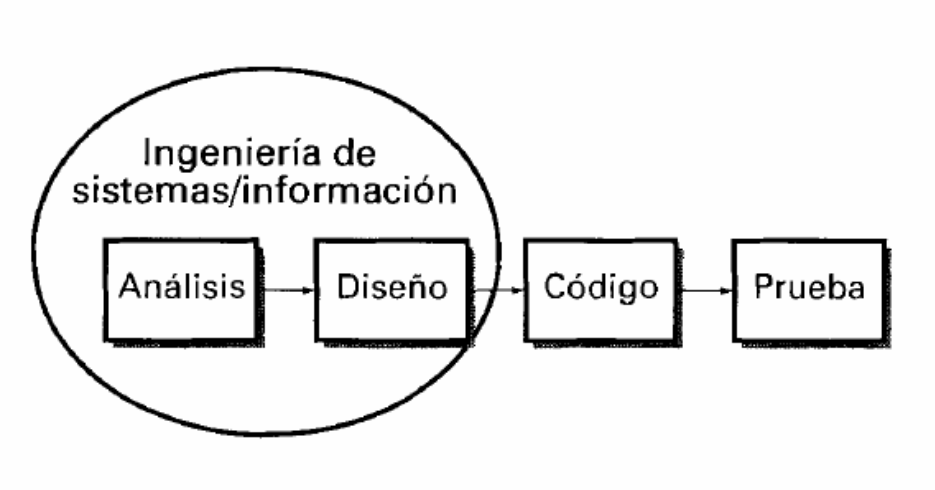
8. Marque con una x las funciones que realiza su producto Software Educativo.

- ? Se hacen preguntas a los estudiantes.
- ? El ordenador adopta el papel de juez poseedor de la verdad y examina al alumno.
- ? Se producen errores cuando la respuesta del alumno está en desacuerdo con la que el ordenador tiene como correcta.

- El ordenador no juzga las acciones del alumno, se limita a procesar los datos que éste introduce y a mostrar las consecuencias de sus acciones sobre un entorno
- El software proporciona un esqueleto, una estructura, sobre la cual los alumnos y los profesores pueden añadir el contenido que les interese.
- Proporciona a los alumnos una serie de herramientas de búsqueda y de proceso de la información que pueden utilizar libremente para construir la respuesta a las preguntas del programa.
- Proporciona datos organizados, en un entorno estático, según determinados criterios, y facilita su exploración y consulta selectiva.
- La investigación de los estudiantes/experimentadores puede realizarse en tiempo real o en tiempo acelerado.

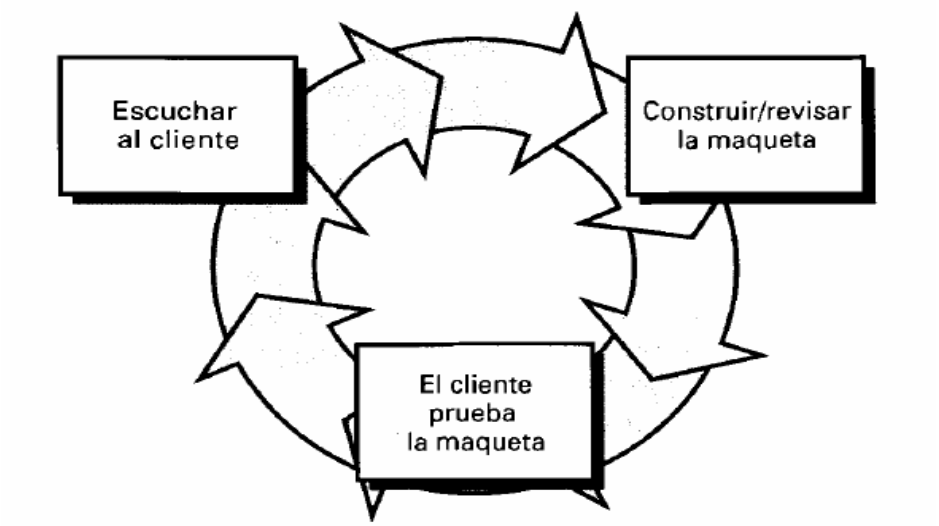
Anexo 3: Modelo lineal secuencial

(Tomado de: PRESSMAN, R. S. *Ingeniería del Software. Un enfoque practico*. Quinta edición Mc Graw-Hill/Interamericana de España, S.A., 2002.)



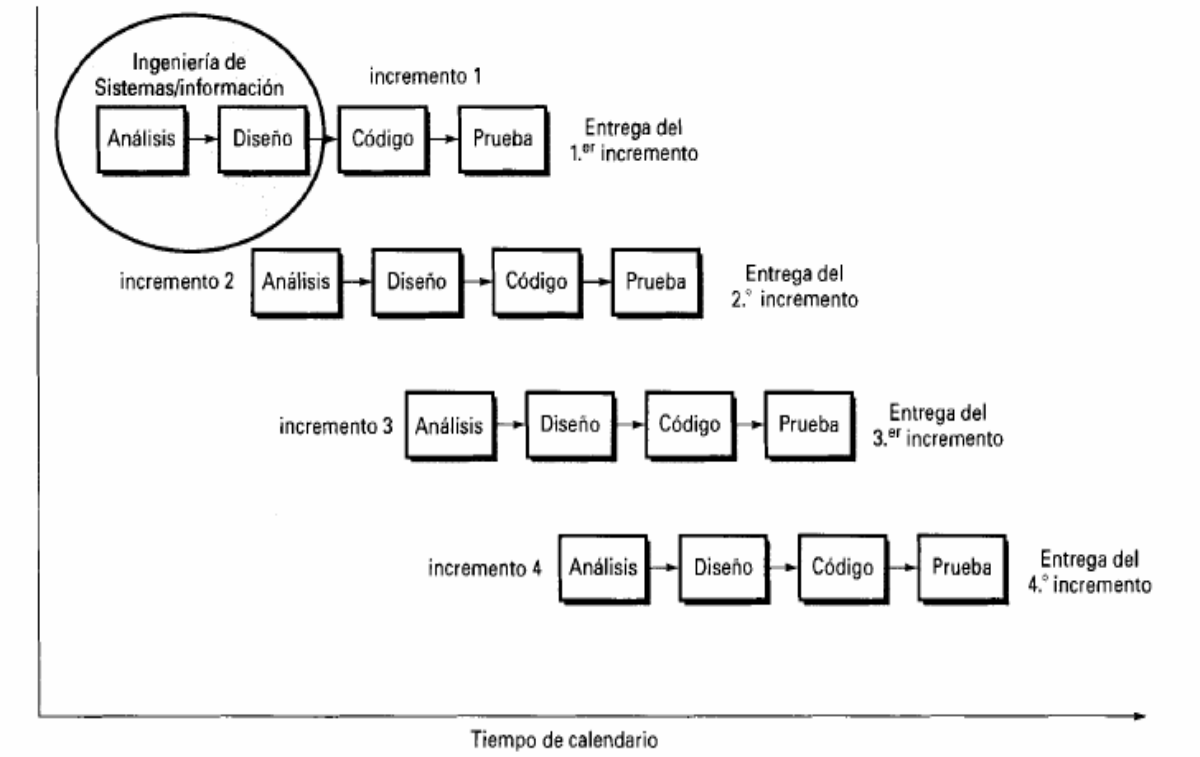
Anexo 4: Paradigma de construcción de prototipos

(Tomado de: PRESSMAN, R. S. *Ingeniería del Software. Un enfoque practico*. Quinta edición Mc Graw-Hill/Interamericana de España, S.A., 2002.)



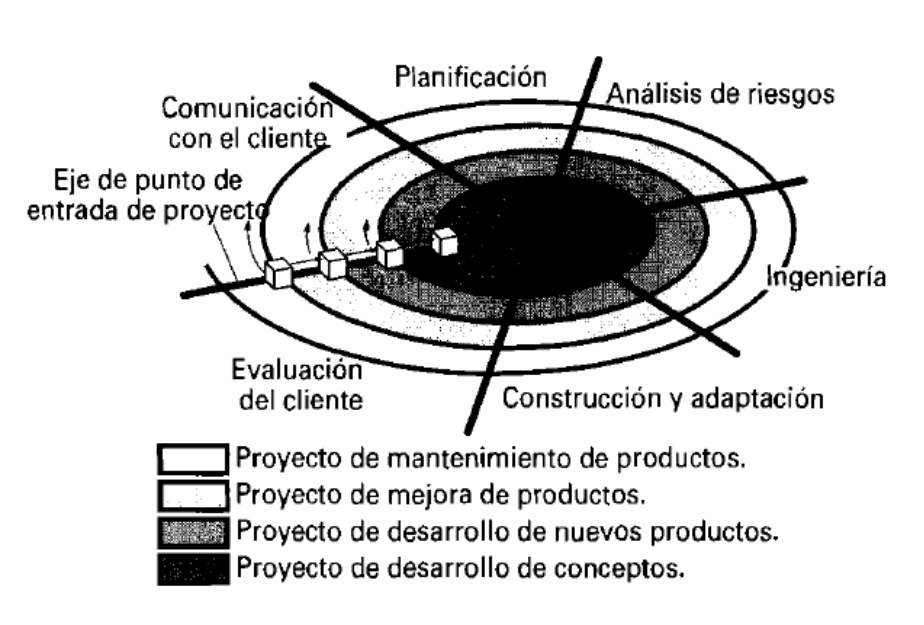
Anexo 5: Modelo incremental

(Tomado de: PRESSMAN, R. S. *Ingeniería del Software. Un enfoque practico*. Quinta edición Mc Graw-Hill/Interamericana de España, S.A., 2002.)

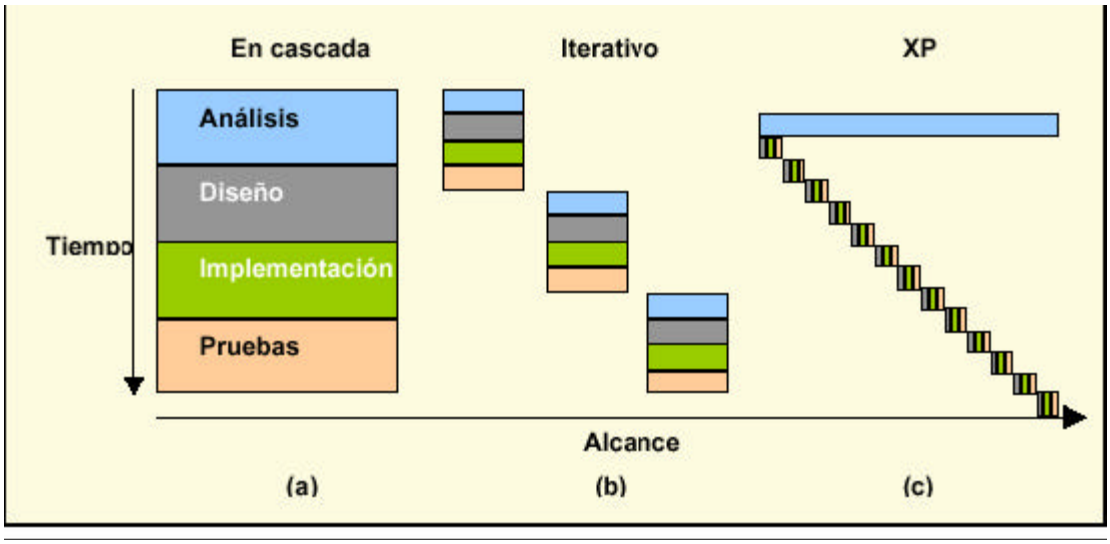


Anexo 6: Modelo espiral típico

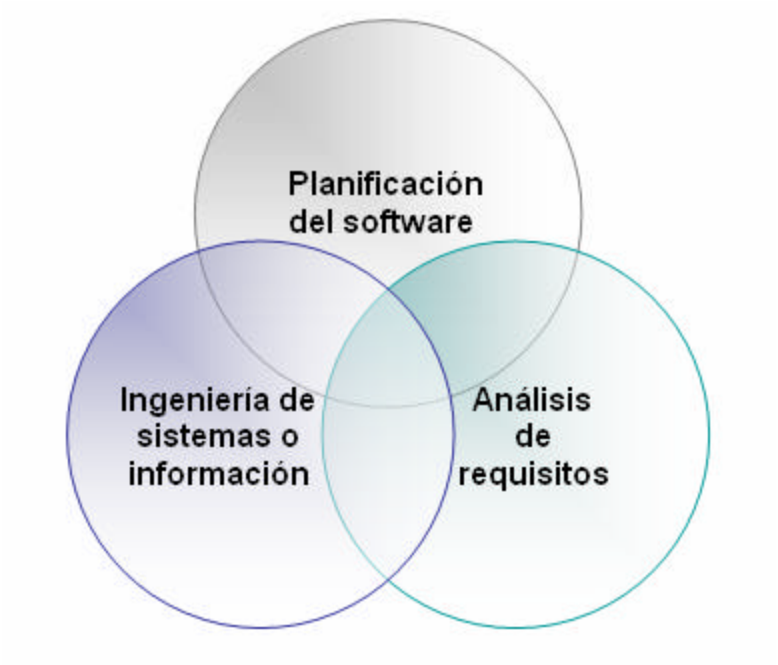
(Tomado de: PRESSMAN, R. S. *Ingeniería del Software. Un enfoque practico*. Quinta edición Mc Graw-Hill/Interamericana de España, S.A., 2002.)



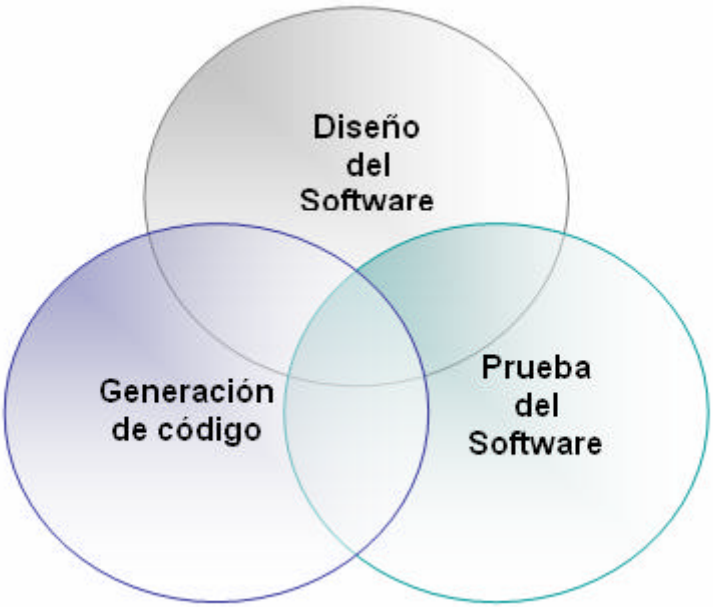
Anexo 7: Representación de los largos ciclos de desarrollo en cascada (a) a ciclos más cortos (b) y a la mezcla que hace XP (c)



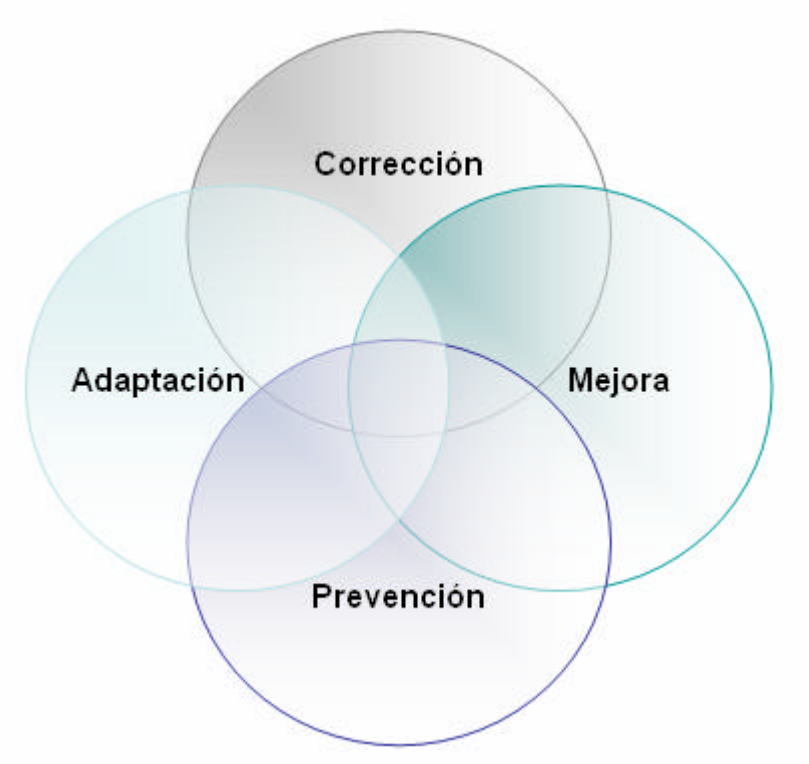
Anexo 8: Tareas principales de la fase de definición



Anexo 9: Tareas principales de la fase de desarrollo



Anexo 10: Tareas principales de la fase de mantenimiento



Glosario de Términos

A:

Actividad: Unidad tangible de trabajo realizada por un trabajador en un flujo de trabajo, de forma que implica una responsabilidad bien definida para el trabajador, produce un resultado bien definido basado en una entrada bien definida , y representa una unidad de trabajo con limites bien definidos. También puede verse como una ejecución de una operación por un trabajador.

Análisis (flujo de trabajo): Flujo de trabajo fundamental cuyo propósito principal es analizar los requisitos descritos en la primera captura de requisitos, mediante su refinamiento y estructuración.

Aprendizaje: El aprendizaje es uno de los procesos más importantes para la psicología científica actual. El aprendizaje es un cambio casi permanente en el comportamiento organismo, mediante el aprendizaje es posible modificar lo que se ha aprendido anteriormente. A diferencia de los animales que nacen con instrucciones genéticas para la supervivencia los humanos tenemos la capacidad de aprendizaje la cual nos da más flexibilidad para adaptarnos al medio ambiente, podemos aprender a resguardarnos de cambios climáticos y adaptarnos a cualquier ambiente, nuestra capacidad de aprendizaje nos permite afrontar cambios.

Arquitectura: Conjunto de decisiones significativas acerca de la organización de un sistema software. La arquitectura no solo se interesa por la estructura y el comportamiento, sino también por las restricciones y compromisos de uso, funcionalidad, funcionamiento, flexibilidad al cambio, reutilización, compresión, economía y tecnología, así como por aspectos estéticos del software.

Artefacto: Pieza de información tangible que es creada, modificada y usada por los trabajadores al realizar actividades. Un artefacto puede ser un modelo, un elemento de un modelo o un documento.

C:

Ciclo de vida del software: Ciclo que cubre 4 fases por las que transita un software.

Construcción: Versión ejecutable del sistema, por lo general, una parte especifica del mismo. El desarrollo transcurre a través de una sucesión de construcciones.

D:

Diseño (flujo de trabajo): Flujo de trabajo fundamental cuyo propósito principal es el de formular modelos que se centran en los requisitos no funcionales y el dominio de la solución, y que prepara para la implementación y pruebas del sistema

F:

Fase: Periodo de tiempo entre dos hitos principales de un proceso de desarrollo.

Flujo de trabajo: Realización de un caso de uso o parte de él. Puede describirse en términos de diagrama de actividad, que incluye a los trabajadores participantes, las actividades que realizan y los artefactos que producen.

H:

Herramienta CASE: Conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software.

I:

Ingeniería de Software: Es una tecnología multicapa en la que, se pueden identificar: los métodos (indican cómo construir técnicamente el software), el proceso (es el fundamento de la Ingeniería de Software, es la unión que mantiene juntas las capas de la tecnología) y las herramientas (soporte automático o semiautomático para el proceso y los métodos)

Implementación (flujo de trabajo): Flujo de trabajo fundamental cuyo propósito esencial es implementar el sistema en términos de componentes, es decir, código fuente, guiones, ficheros binarios, ejecutables etc.

J:

JavaScript: Lenguaje para la programación web.

M:

Metodologías de Desarrollo: Se define como un conjunto de filosofías, etapas, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas de información.

P:

P´s: persona, producto, proceso y proyecto

Proyecto: Esfuerzo de desarrollo para llevar un sistema a lo largo de un ciclo de vida.

R:

Requisito: Condición o capacidad que debe cumplir un sistema.

S:

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

Software: Es la suma total de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de cómputo.

Software Educativo (SWE.): Se define como un programa automatizado diseñado con una intencionalidad educativa para ser utilizado en el proceso de aprendizaje, utiliza procedimientos para que el estudiante aprenda, se fomenta el análisis de problemas, facilita el trabajo en grupo, provee soporte en actividades docentes y en el sentido más amplio, mejora las habilidades del pensamiento y la resolución de problemas.

T:

TIC: Tecnologías de la Información y las Comunicaciones**.**