Universidad de las Ciencias Informáticas
Facultad 5

Título: **Multimedia interactiva para la enseñanza**

Del Diseño y la Construcción

de Software

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Yahima Martija Pupo.

Renier Alexis Bubaire Riverón.

Tutor: Msc. Yamilis Fernández Pérez

Co-tutor: Lic. Michel Ramos Navarro

Junio-2010

"… el hombre debe transformarse al mismo tiempo que la producción progresa; no realizaríamos una tarea adecuada si fuéramos tan sólo productores de artículos, de materias primas y no fuéramos al mismo tiempo productores de hombres."

Ernesto Che Guevara.

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Yahima Martija Pupo

_____
Firma del Autor

Msc. Yamilis Fernández Pérez

_____
Firma del Tutor

Renier Alexis Bubaire Riverón

_____
Firma del Autor

Lic. Michel Ramos Navarro

_____
Firma del Co-tutor

DATOS DE CONTACTO

Síntesis del Tutor: MSc. Yamilis Fernández Pérez

Graduada de Ingeniera en Sistema Automatizado de Dirección, en 1992 en el ISPJAE, Profesora asistente desde 1995. MSc en Informática Aplicada en 1995. Imparte docencia en universidades desde 1992. Ha desarrollado trabajos con Universidades extranjeras en Brasil, Bolivia y Canadá. Fue la jefa de Departamento Docente Central de Ingeniería y Gestión de Software de la UCI desde 2005 hasta el 2008. Actualmente es la directora de Formación Posgraduada de la UCI.

Síntesis del Tutor: Michel Ramos Navarro.

Graduado de Licenciado en Física en el 2000 en la UH. Profesor asistente desde 2007. Imparte docencia desde 2002 en la Universidad de las Ciencias Informáticas. Trabajó un año y medio en una empresa productora de software (SIMPRO). Ha desarrollado trabajos con entidades extranjeras en Venezuela. Desde el 2002 y hasta el 2009 fungió como jefe de Departamento Central de la disciplina de Práctica Profesional. Actualmente es profesor de la facultad 5.

AGRADECIMIENTOS

A nuestros tutores **Yamilis** y **Michel** por apoyarnos tanto y contribuir a realizar nuestras metas.

A nuestras madres **Anita y Deysi** por apoyarnos en todo momento y estar ahí, cada vez que la necesitamos.

A nuestros hermanos y hermanas por apoyarnos también en todos y cada uno de los momentos en que lo hemos necesitado.

A toda nuestra familia que ha estado pendiente y preocupada porque este sueño se realizara.

A nuestras parejas **Dayamí** y **Leansy** por brindarnos todo el cariño y la comprensión cuando más lo necesitábamos.

A **Jamit** por estar pendiente y por su gran contribución en el desarrollo de esta tesis..

A **Yanay**, **Kenia**, **Liset**, **Marilín**, **Pablito**, **Nelson** y **Odalys** por ser más que amigos y brindarnos su cariño incondicional aún en la distancia.

A **Ernesto** y **Yoandy** por su contribución y su tiempo dedicado a ayudarnos en la realización de la multimedia.

A nuestros compañeros de grupo que aunque no estén a nuestro lado sabemos que su pensamiento ha estado con nosotros en cada momento.

DEDICATORIA

A **Ramón Martija**, **Eduardo Pupo y papito Toñé** que donde quiera que se encuentren se que están orgullosos de verme realizada como una ingeniera.

Dedicada a mi abuela **Eca** que aunque ya no esté conmigo se que donde se encuentre se siente orgullosa de ver que mi sueño se hizo realidad.

En especial a nuestras **madres, padres,** y demás familiares.

Resumen

Con el surgimiento del Internet en la década de los noventa, se ha desatado en la humanidad un gran interés por aprender a interactuar con las Tecnologías de la Información y las Comunicaciones (TIC). La Universidad de las Ciencias Informáticas (UCI) no se ha quedado rezagada en esta revolución tecnológica. Desde sus inicios se ha sumado a la creación de multimedias para apoyar el proceso de enseñanza y aprendizaje en el país y en la propia universidad.

En el presente trabajo se desarrolla una multimedia interactiva enfocada en el proceso de combinación planteado por Nonaka y Takeuchi en el modelo SECI (Socialización, Exteriorización, Combinación e Interiorización) sobre la gestión del conocimiento, para los temas Diseño y Construcción de Software en la UCI. Se apoya en la confección de los mapas conceptuales de cada tema para recoger los principales conceptos relacionados con esta disciplina. Para el desarrollo de la multimedia, se realizó un análisis detallado de las diferentes herramientas y metodologías existentes. Se decide utilizar UML en su extensión OMMMA - L (lenguaje representativo), ActionScript (lenguaje de implementación) y XML (lenguaje para gestionar y agrupar los datos en volúmenes compactos de información).

Palabras Claves:

Diseño, Construcción, multimedia, multimedia interactiva, combinación, gestión del conocimiento, herramienta, metodología y lenguaje.

TABLA DE CONTENIDOS

Índice de Figuras

Índice de Tablas

INTRODUCCIÓN

Con el surgimiento del Internet en la década de los noventa, se ha desatado en la humanidad un gran interés por aprender a interactuar con las Tecnologías de la Información y las Comunicaciones (TIC), las cuales habían sido prácticamente desconocidas por todos hasta ese momento. Dichas tecnologías proveen al hombre de grandes conocimientos y le facilita el trabajo a la hora de crear un proyecto, ya sea en el mundo de la informática o en cualquier otra rama que requiera su uso. Principalmente en el mundo del software es donde se aplican estas tecnologías con todo rigor.

Cuba no se ha quedado rezagada en el desarrollo de las TIC. Esto lo demuestra con la puesta en práctica de la informatización de la sociedad con la creación, específicamente, de centros de desarrollo de software como es la empresa DESOFT. Con la creación de los Joven Club de Computación y Electrónica para el disfrute y aprendizaje de todo aquel que esté interesado en desarrollar sus conocimientos en el área de la informática.

La Universidad de las Ciencias Informáticas es uno de los ejemplos más concretos de esta nueva etapa de desarrollo en el país. Es por ello que en este proyecto de la Revolución se ha puesto gran parte de la tecnología y de las esperanzas, para contribuir al desarrollo económico del país.
No obstante la UCI no está exenta de los problemas que aquejan al mundo del software. En el año 2008 se realizó un diagnóstico a los proyectos productivos en la universidad con el objetivo de:

- Caracterizar el ambiente técnico de las organizaciones productivas,
- Validar los resultados del levantamiento de información realizado.
- Revisar los documentos: Visión, Plan de Desarrollo y Cronogramas de cada proyecto productivo.
- Revisar el ambiente de desarrollo (validar los datos del proyecto, capital humano y entregables).

Por solo citar algunos.
Se encontró inconsistencia en la entrega de la información, 0 a 2 inconsistencias en 108 proyectos, para un 71%; de 3 a 6 inconsistencias en 45 proyectos, para un 29%; y de 7 a 14 inconsistencias ningún proyecto.
El diagnóstico mostró varios indicadores a mejorar en la universidad:

- Planificación: 5%.
- Trabajo en equipo: 7.50%.
- Definición de los requerimientos: 10%.
- Formación del personal: 15%.

- Apoyo de los directivos al proyecto: 15%. (Calidad, 2008)

Analizando específicamente el indicador de formación del personal, se puede decir que es un punto fundamental en el que se debe hacer énfasis. En los últimos tiempos ha crecido la necesidad de desarrollar el intelecto y explotar al máximo los conocimientos de las personas, para lograr así una mayor organización y desarrollo en las entidades. Es por ello que resulta indispensable, en cualquier empresa u organización, fomentar el desarrollo de habilidades para optimizar el aprovechamiento de estos conocimientos.

En la Universidad de las Ciencias Informáticas se realiza un arduo trabajo para lograr el desarrollo de los conocimientos de las personas vinculadas a la producción, de los estudiantes y el claustro de profesores que lo conforma. El mismo se lleva a cabo mediante la gestión del conocimiento, que es la encargada de resolver cómo utilizar mejor los recursos humanos con sus conocimientos, apoyándose para ello en las Tecnologías de la Información y las Comunicaciones (TIC).

Sin embargo existen varios factores que impiden que el conocimiento en el personal que conforma un proyecto sea debidamente desarrollado y/o estimulado, tanto en los estudiantes como en los profesores de la Universidad de las Ciencias Informáticas.

Dicha Universidad cuenta con un Entorno Virtual de Aprendizaje (EVA), el cual sirve de apoyo al proceso de formación de la carrera de Ingeniería en Ciencias Informáticas(http://eva.uci.cu) y a los programas de postgrado desarrollados por parte del claustro de profesores de la universidad (http://evapostgrado.uci.cu). Dicho entorno no está muy enriquecido de materiales interactivos que apoyen la capacitación de los estudiantes y profesores, sobre todo pregrado que es donde más interacción estudiantes-profesor debe existir.

Conjuntamente con esto, está el poco énfasis que se hace en consolidar el aprendizaje de temas de diseño y construcción de software, específicamente debido a que sólo se toman algunos conceptos generales en asignaturas que se comienzan a impartir y no desde el comienzo de la carrera, por ende los estudiantes no adquieren todos los conocimientos acerca de estos temas que son tan importantes a la hora de desarrollar un software.

En los análisis realizados en los consejos docentes quincenales del Departamento Central de Ingeniería y Gestión de software acerca de los resultados obtenidos por parte de los estudiantes en los exámenes parciales de las asignaturas que pertenecen a dicho departamento, y los resultados obtenidos en las exámenes finales de disciplina, se ha llegado a la conclusión de que en los temas de Diseño y Construcción de software son los temas donde más errores se cometen. (Pérez, 2010)

Debido a estos problemas anteriormente explicados, se plantea como **problema científico:** El insuficiente conocimiento del desarrollo del proceso de diseño y construcción implica deficiencias en la Gestión de los proyectos de software en la UCI. Como **objeto de estudio**: El proceso de combinación en la gestión del conocimiento del diseño y la construcción en el desarrollo de software. A partir del problema científico anteriormente planteado, se ha determinado como **Objetivo de la investigación**: Implementar una multimedia interactiva como medio de capacitación en los procesos de diseño y construcción de software en la Universidad de Ciencias Informáticas.
Como **campo de acción**: Desarrollo de una multimedia interactiva para la capacitación del personal en el diseño y la construcción de software.

Para dar cumplimiento al objetivo anteriormente planteado, se realizarán las siguientes **Tareas Investigativas**:

- Análisis de toda la información necesaria sobre el diseño y la construcción de software para la elaboración del contenido del producto a desarrollar.
- Confección de mapas conceptuales que contengan los conceptos necesarios para la comprensión de los temas.
- Análisis de forma crítica de los diferentes lenguajes como XML (Lenguaje de Marcas Extensibles), ActionScript (Lenguaje de programación), y los lenguajes de modelado UML (Lenguaje Unificado de Modelado) y su extensión OMMMA-L (Lenguaje de Modelado Orientado a Objetos para Aplicaciones Multimedia).
- Selección de la metodología de desarrollo de software que mejor se ajuste a las características del producto a desarrollar, a partir de las metodologías existentes para el desarrollo de aplicaciones con tecnología multimedia.
- Realización del análisis, diseño e implementación de una multimedia que contenga mapas conceptuales y diversos medios, que ayuden a la comprensión del tema, utilizando la herramienta y el lenguaje de modelado seleccionado.

Para lograr el objetivo de esta investigación, se definen los siguientes métodos teóricos a utilizar:

- Histórico-lógico: Se utilizó para la realización de una investigación profunda de los procesos vinculados al diseño y la construcción de software.
- Analítico-sintético para el procesamiento de la información recopilada durante la investigación.

Como método empírico se utilizó:

- Entrevista a profesores y especialistas, principalmente a aquellos que están más vinculados a los procesos de diseño y construcción de software en la UCI.

El presente trabajo está conformado por tres capítulos que contienen todo el historial investigativo y de desarrollo del mismo.

Capítulo 1: “Diseño y Construcción de Software”.
En éste se realiza la fundamentación teórica que está basada en el análisis del libro SWEBOK, el cual define los principales conceptos sobre el diseño y la construcción de software y cómo se deben desarrollar dichos temas, argumentando dichos temas con otras bibliografías consultadas. Se hace un análisis sobre la gestión del conocimiento, el cual queda plasmado en un epígrafe. Además, se realiza una comparación entre las diferentes metodologías que se utilizan para construir un software, para seleccionar la más conveniente en este caso. Se analizan las diferentes herramientas a utilizar y se hace referencia a algunos conceptos de interés para el presente trabajo.

Capítulo 2: **“**Descripción de la solución propuesta**”**.
Este capítulo está vinculado al análisis de la multimedia interactiva diseño y construcción de software. Se construye un modelo de dominio. Se definen los requisitos funcionales y no funcionales que debe cumplir, y se modela el sistema para la realización de los casos de uso correspondientes. Además se define el guión de la aplicación, donde va a estar implícito el contenido de la misma.

Capítulo 3: **“**Diseño de la solución**”**.
Este capítulo va dirigido a la definición del modelo de análisis del sistema y modelos de implementación, incluyendo todos los diagramas necesarios. Se hace una descripción de los archivos XML y también se describen las normas de diseño que va a tener la aplicación.

CAPÍTULO 1: DISEÑO Y CONSTRUCCIÓN DE SOFTWARE.

Introducción:

Es fundamental para todo ingeniero informático tener un vasto conocimiento acerca de cómo diseñar y construir un software, así garantizará un producto terminado y con buena calidad.
En este primer capítulo se hace un estudio del libro SWEBOK que describe todo el contenido necesario que debe conocer, comprender y analizar cada Ingeniero Informático. Además se estudia la propuesta del SEEK (Currícula de la Ingeniería de Software) de los temas que se deben impartir en los cursos de Diseño y la Construcción de Software. Se muestra además un estudio realizado sobre la gestión del conocimiento. Se abordan los principales conceptos de multimedia, multimedia interactiva y software educativo. Se exponen aspectos sobre las tecnologías, haciendo énfasis en la metodología, lenguajes y herramientas a utilizar para el desarrollo de la aplicación.

1.1 Swebook y otros estándares

La guía del cuerpo del conocimiento de la ingeniería de software, surgió en el año 2004 por IEEE Computer Society con el objetivo de:

- Caracterizar los contenidos de la Ingeniería de Software.
- Proveer acceso, a través de las temáticas, al conjunto de conocimientos de la ingeniería de software.
- Promover una visión consistente de la ingeniería del software en todo el mundo.
- Clarificar la posición de la ingeniería del software respecto a otras disciplinas como las ciencias de la computación o las matemáticas.
- Proveer una base para su desarrollo curricular y la creación de materiales de certificación. (SWEBOOK, 2004)

A continuación se muestra la descomposición por áreas de conocimiento de dicho libro de una manera gráfica:

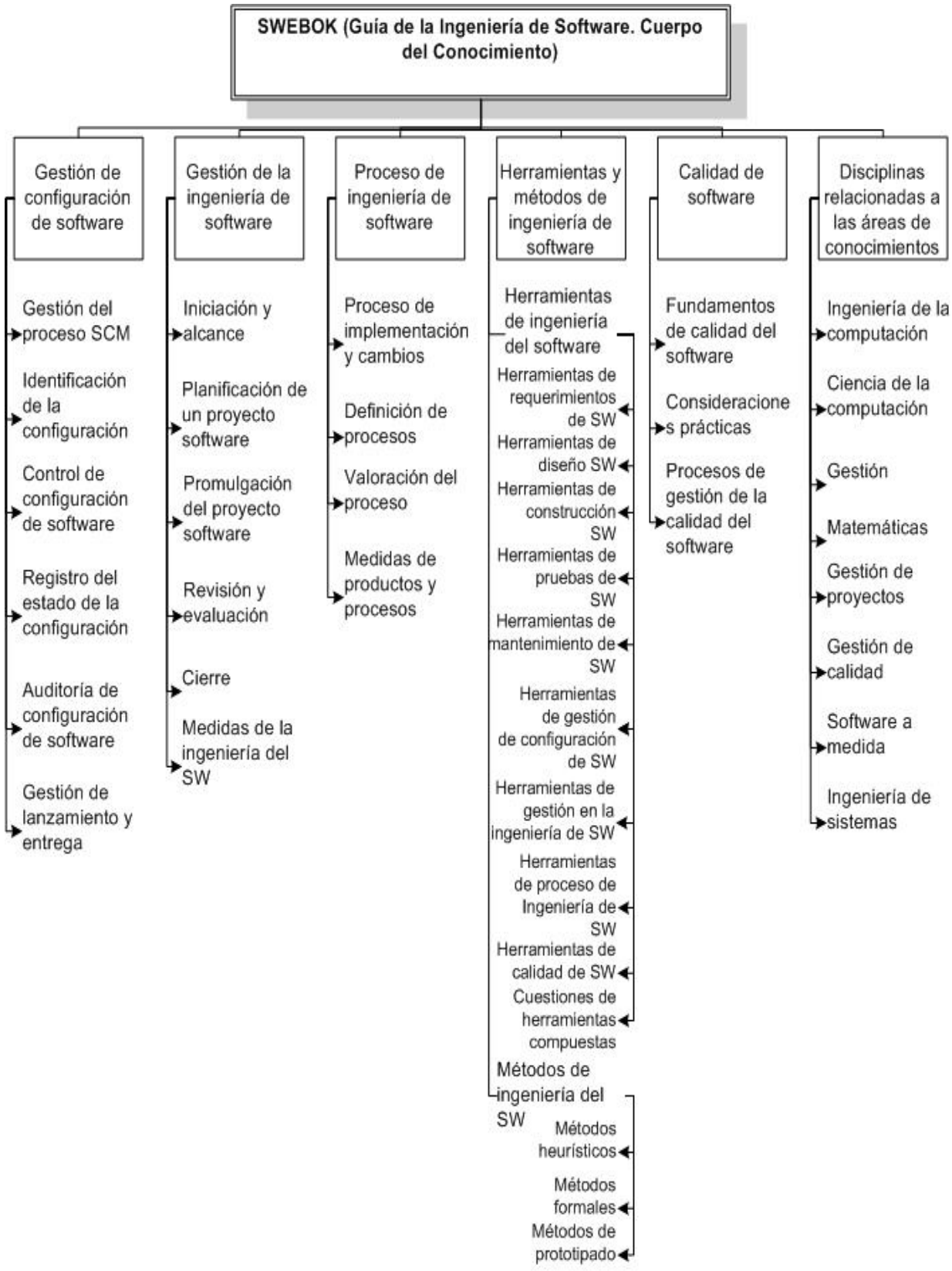


Figura 1. 1Guía de la Ingeniería de Software. Cuerpo del Conocimiento (SWEBOOK, 2004)

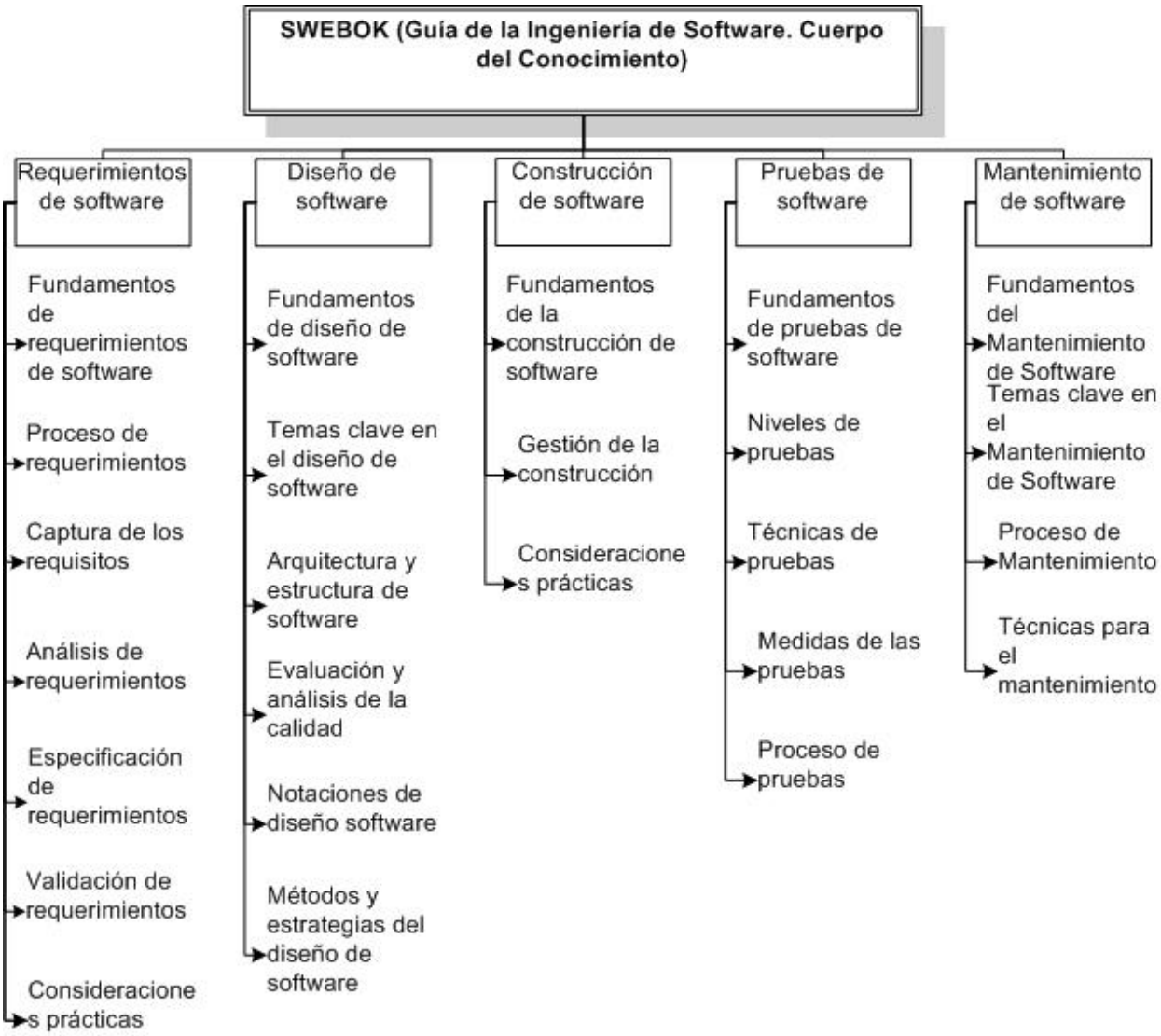


Figura 1. 2 Guía de la Ingeniería de Software. Cuerpo del Conocimiento (SWEBOOK, 2004)

El presente trabajo está basado en el estudio de dos de las áreas de conocimiento mostradas anteriormente, el área de Diseño de Software y el área de Construcción de Software. A continuación se muestra como mismo está expuesto en el SWEBOOK, la descomposición por temas de cada una de estas áreas de conocimiento.

Diseño de Software: está dividido en 6 temas: Fundamentos del Diseño, Elementos Claves, Estructura y arquitectura del software, Análisis de la calidad y evaluación, Notaciones del Diseño y Estrategias y Métodos del Diseño. (Ver Fig. 1.3)

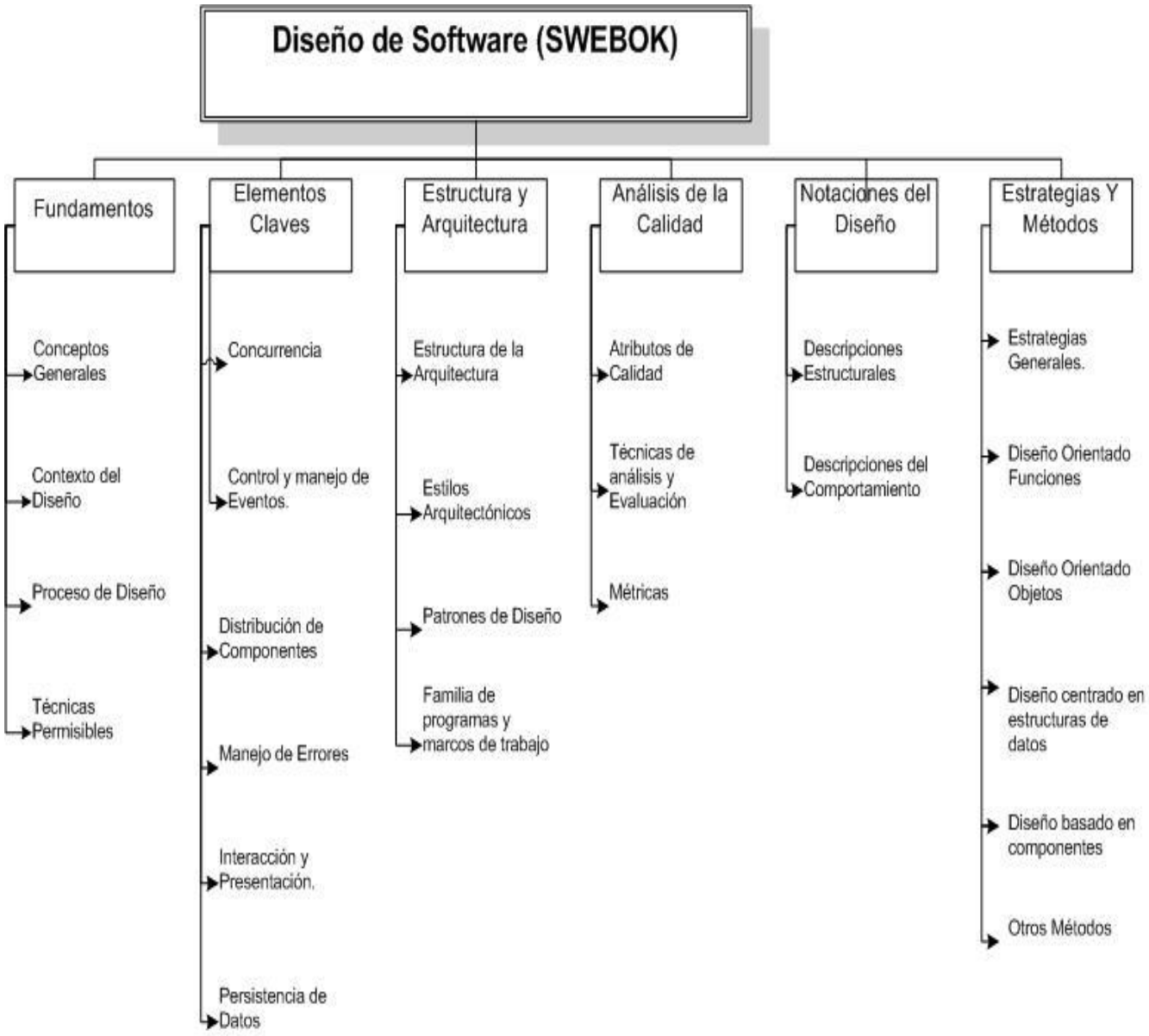


Figura 1. 3 Descomposición de los temas del área de conocimiento de Diseño de Software (SWEBOOK, 2004)

Por su parte Construcción de Software está dividida en tres temas: Fundamentos de la Construcción, Gestión de la Construcción y Consideraciones Prácticas. (Ver Fig. 1.4)

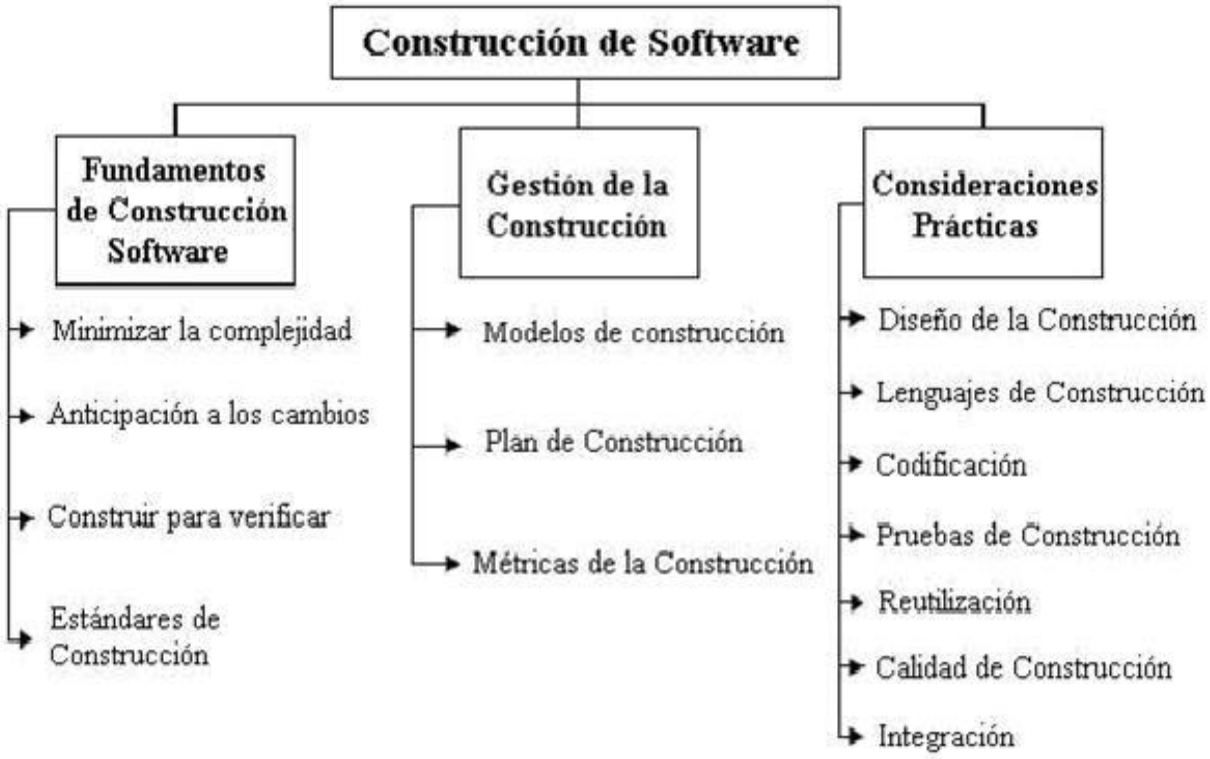


Figura 1. 4 Descomposición de los temas del área de conocimiento de Construcción de Software (SWEBOOK, 2004)

1.1.2 SEEK

El propósito principal de este libro es proporcionar orientación a las instituciones académicas y agencias de acreditación sobre lo que debería constituir un pregrado de ingeniería de software en la educación.

A continuación se hará un breve resumen de lo que propone el SEEK para los temas de Diseño y Construcción de Software.

1.1.2.1 Diseño de Software

El diseño es fundamental para cualquier actividad de ingeniería, y desempeña un papel crítico en lo que respecta al software. Como tal, el diseño proporciona los requisitos previos para el desarrollo físico". Se ocupa de cuestiones, técnicas, estrategias, representaciones y patrones, los cuales utiliza para determinar cómo implementar un componente o sistema. El diseño se ajustará a requisitos funcionales dentro de las limitaciones impuestas por otros requisitos, tales como los recursos, rendimiento, fiabilidad y seguridad. Esta área también incluye especificaciones de las interfaces internas entre los componentes de software, diseño arquitectónico, diseño de datos, diseño de interfaces de usuario, y la evaluación del diseño. (SEEK, 2004)

A continuación se muestra cómo queda conformada la propuesta del curso de Diseño de Software en dicho estándar.

La relevancia del tema está representada de la siguiente manera:

- **Esencial (E):** El tema es parte del núcleo.
- **Deseable (D):** El tema no es parte del núcleo, pero debería ser incluido en el núcleo de un programa en particular, si es posible, de lo contrario, debe ser considerado como parte de la elección de materiales.
- **Opcional (O):** El tema puede incluirse o no en el curso.

Tabla 1. 1 Descomposición del tema de Diseño (SEEK, 2004)

Referencias		**E,D**
DES	***Diseño de Software***	
DES.con	Conceptos del Diseño	
DES.con.1	Definiciones del Diseño	E
DES.con.2	Fundamentos del Diseño	E
DES.con.3	Contexto del Diseño	E
DES.con.4	Principios del Diseño	E
DES.con.5	Diseño de los atributos de calidad	E
DES.con.6	Estilos arquitectónicos, patrones, reutilización	E
DES.str	*Estrategias del Diseño*	
DES.str.1	Diseño Orientado a Funciones	E
DES.str.2	Diseño Orientado a Objetos	E
DES.str.3	Diseño Centrado en Estructura de Datos	D
DES.str.4	Diseño orientado a Aspectos	O
DES.ar	*Diseño arquitectónico*	
DES.ar.1	Estilos arquitectónicos	E
DES.ar.2	Hardware usados en la arquitectura de software	E
DES.ar.3	Requisitos de traceabilidad de la arquitectura	E
DES.ar.4	Arquitectura de dominio específico y líneas de producto	E
DES.ar.5	Notaciones de la arquitectura	E
DES.hci	*Diseño de Interfaz de usuario*	
DES.hci.1	Principios generales del diseño de interfaz de usuario	E
DES.hci.2	Uso de modelos, navegación	E
DES.hci.3	Codificación de las técnicas y de diseño visual	E

1.1.2.2 Construcción de Software

El tema de Construcción está incluido en el curso de Informática Esencial dentro del SEEK, incluye el conocimiento acerca de la transformación de un diseño en una aplicación, las herramientas utilizadas durante este proceso, y métodos de construcción.

A continuación se muestra cómo queda conformada la propuesta del curso de Construcción de Software en dicho estándar.

Tabla 1. 2 Descomposición del tema Construcción de Software (SEEK, 2004)

Referencias		E,D
CMP	*Informática Esencial*	
CMP.cf.1	Fundamentos de programación	E
CMP.cf.2	Algoritmos, estructuras de datos, representación (estática y dinámica) y complejidad	E
CMP.cf.3	Técnicas de resolución de problemas	E
CMP.cf.4	Abstracción (encapsulación, jerarquía, etc.)	E
CMP.cf.5	Lenguajes de programación básicos	E
CMP.cf.6	Semántica de lenguajes de programación	D
CMP.ct	*Tecnologías de la Construcción*	
CMP.ct.1	Reutilización de código y librerías	E
CMP.ct.2	Cuestiones de tiempo de ejecución OO (polimorfismo, etc)	E
CMP.ct.3	Aserciones, diseño por contrato, programación defensiva	E
CMP.ct.4	Manejo de errores, de excepciones, y tolerancia a fallos	E
CMP.ct.5	Gramática de procesamiento (parsing)	E
CMP.ct.6	Métodos de construcción de software distribuido	E
CMP.ct.7	Plataforma de las normas	D
CMP.ct.8	Pruebas de programación	D
CMP.tl	*Herramientas de Construcción*	
CMP.tl.1	Entornos de desarrollo	E
CMP.tl.2	Herramientas de pruebas de unidad	E
CMP.tl.3	Aplicación orientada a los lenguajes	E
CMP.tl.4	Perfiles, análisis de rendimiento y herramientas de corte	D
CMP.fm	*Métodos formales de construcción*	
CMP.fm.1	Aplicación de las máquinas abstractas	E
CMP.fm.2	Aplicación de lenguajes de especificación y métodos	E
CMP.fm.3	Generación automática de código de una especificación	E
CMP.fm.4	Programa de derivación	E
CMP.fm.5	Análisis de las implementaciones candidatas	**E**
CMP.fm.6	Mapeo de una especificación para distintas implementaciones	E

CMP.fm.7	Refinamiento	E

1.1.2.3 Descripciones detalladas de los cursos propuestos:

Fundamentos de Programación: Introduce los conceptos fundamentales de la programación de procedimiento. Los temas incluyen tipos de datos, estructuras de control, funciones, matrices, archivos, y la mecánica de funcionamiento, pruebas y depuración. El curso también ofrece una introducción al contexto histórico y social de la computación y una vista de la informática como disciplina.

Unidades de Diseño y Construcción que se estudian en dicho tema:

DES.con – Conceptos del Diseño de Software.

DES.con.1 - Definiciones del Diseño.

CMP.cf – Informática Esencial.

CMP.cf.1 – Fundamentos de la programación.

CMP.cf.2 – Algoritmos, estructura de datos y representación.

CMP.cf.3 – Técnicas de resolución de problemas.

CMP.cf.6 - Semántica de lenguajes de programación.

CMP.cf.9 – Programación de lenguajes básicos.

CMP.tl – Herramientas de Construcción.

El Paradigma Orientado a Objetos: Introduce los conceptos de programación orientada a objetos a los estudiantes con un fondo en el paradigma de procedimiento. El curso comienza con una revisión de estructuras de control y tipos de datos con énfasis en los tipos de datos estructurados y procesamiento de arreglos. Los temas incluyen una visión general de los principios de la programación del lenguaje, un simple análisis de algoritmos, la búsqueda básica y técnicas de clasificación, y una introducción a cuestiones de ingeniería de software.

Unidades de Diseño y Construcción que se estudian en dicho tema:

DES.con.4 – Principios del Diseño.

DES.hci – Diseño de Interfaz de Usuario.

DES.hci.1 – Principios generales del diseño de interfaz de usuario.

CMP.cf - Informática Esencial.

CMP.cf.1 – Fundamentos de la programación.

CMP.cf.2 – Algoritmos, estructura de datos y representación.

CMP.cf.3 – Técnicas de resolución de problemas.

CMP.cf.4 – Abstracción.

CMP.cf.5 - Lenguajes de programación básicos.

CMP.cf.9 - Programación de lenguajes básicos

CMP.cf.11 – Bases de datos básicas.
CMP.ct Tecnologías de la Construcción.

Estructuras de Datos y Algoritmos: Los temas incluyen la recursividad, la filosofía de programación orientada a objetos, estructuras de datos fundamentales (Incluyendo pilas, colas, listas enlazadas, tablas hash, árboles, y gráficos), los fundamentos del análisis de algoritmos y una introducción a los principios de traducción de lenguajes.
Unidades de Diseño y Construcción que se estudian en dicho tema:
CMP.cf Informática Esencial.
CMP.cf.1 Fundamentos de la programación.
CMP.cf.2 Algoritmos, estructura de datos y representación.
CMP.cf.4 Abstracción.
CMP.cf.9 Programación de lenguajes básicos.

Base de Datos: Presenta los conceptos y las técnicas de los sistemas de base de datos.
Unidades de Diseño y Construcción que se estudian en dicho tema:
CMP.cf Informática Esencial.
CMP.cf.2 Algoritmos, estructura de datos y representación.

Introducción a la Ingeniería de Software y Computación: Estudia los principios generales de la informática: la resolución de problemas, la abstracción, la división del sistema en cómodos componentes, la reutilización, interfaces simples, conceptos de programación: las construcciones de control; expresiones; uso de la API, los datos simples, incluyendo los arreglos y cadenas, las clases y la herencia. Conceptos del Diseño: Evaluación de alternativas. Fundamentos de las pruebas.
Unidades de Diseño y Construcción que se estudian en dicho tema:
DES.con Conceptos del Diseño de Software.
DES.str Estrategias del Diseño de Software.
DES.dd Diseño detallado.
CMP.cf Informática Esencial.
CMP.cf.1 Fundamentos de la programación.
CMP.cf.3 Técnicas de resolución de problemas.
CMP.cf.4 Abstracción.
CMP.cf.5 - Lenguajes de programación básicos.
CMP.cf.6 Semántica de lenguajes de programación.
CMP.ct Tecnologías de la Construcción.

CMP.tl Herramientas de Construcción.

Ingeniería de Software y Computación II: Estudia **r**equisitos, diseño, implementación, revisión y verificación de software simple que interactúa con el sistema operativo, bases de datos y la red, y que implique la interfaces gráficas de usuario. El uso de estructuras de datos simples, tales como pilas y colas. El uso eficaz de las instalaciones de un lenguaje de programación. Diseño y análisis de algoritmos sencillos, incluidos los que utilizan recursividad. El uso de patrones de diseño simple, como delegación. Dibujo simple de clases UML, paquete, y diagramas de componentes. Lidiar con el cambio: evolución de los principios, la tramitación de requisitos de cambios; problema de información y seguimiento.

Unidades de Diseño y Construcción que se estudian en dicho tema:

DES.str Estrategias del Diseño de Software.

DES.dd Diseño detallado.

DES.nst Notaciones del Diseño.

CMP.cf Informática Esencial.

CMP.cf.1 Fundamentos de la programación.

CMP.cf.3 Técnicas de resolución de problemas.

CMP.cf.4 Abstracción.

Ingeniería de Software y Computación III: Presenta el proceso de software, la planificación y seguimiento de los trabajos. Análisis, arquitectura y diseño de los sistemas cliente-servidor usando UML, con énfasis en la clase y diagramas de estado. Diseños de aplicación utilizando las estructuras de datos adecuados, los marcos y las API.

Unidades de Diseño y Construcción que se estudian en dicho tema:

DES.con Conceptos del Diseño de Software.

DES.str Estrategias del Diseño de Software.

DES.ar Diseño arquitectónico.

DES.hci Diseño de Interfaz de usuario.

DES.ev Evaluación del Diseño.

CMP.cf Informática Esencial.

CMP.cf.1 Fundamentos de la programación.

CMP.cf.2 Algoritmos, estructura de datos y representación.

CMP.cf.4 Abstracción.

CMP.cf.9 Programación de lenguajes básicos.

CMP.ct Tecnologías de la Construcción.

Introducción a la Ingeniería de Software: Principios de ingeniería de software: Requisitos, diseño y pruebas. Revisión de los principios de la orientación a objetos. Análisis orientado a objetos usando UML. Macros y APIs. Introducción a la arquitectura cliente-servidor. Análisis, diseño y programación de servidores simple y clientes. Introducción a la tecnología de interfaz de usuario.
Unidades de Diseño y Construcción que se estudian en dicho tema:
DES.con Conceptos del Diseño.
DES.str Estrategias del Diseño.
DES.ar Diseño arquitectónico.
DES.hci Diseño de interfaz de usuario.
DES.dd Diseño detallado.
DES.nst Notaciones del Diseño.
DES.ev Evaluación del Diseño.
CMP.ct Tecnologías de la Construcción.
CMP.ct.1 – Diseño y uso de API.
CMP.ct.2 – Reutilización de código y librerías.
CMP.ct.3 - Aserciones, diseño por contrato, programación defensiva.

Construcción de Software: Principios generales y técnicas de diseño de software de bajo nivel. Conceptos básicos del lenguaje y protocolo de diseño. Estado de transición y diseño de software basado en tablas. Métodos formales para construcción de software. Técnicas para el manejo de la concurrencia y la comunicación entre procesos. Técnicas para el diseño de software numérico. Herramientas para la construcción dirigida por modelos. Análisis y optimización del rendimiento.
Unidades de Diseño y Construcción que se estudian en dicho tema:
CMP.ct Tecnologías de la Construcción.
CMP.ct.6 - Métodos de construcción de software distribuido,
CMP.ct.7 - Plataforma de las normas.
CMP.ct.8 – Pruebas de programación,
CMP.tl Herramientas de Construcción.
CMP.fm Métodos formales de Construcción.

Diseño y Arquitectura de grandes Sistemas de Software: Presenta la arquitectura. Los estilos arquitectónicos y patrones. Configuraciones y gestión de la configuración y las líneas de productos.
Unidades de Diseño y Construcción que se estudian en dicho tema:
DES.str Estrategias del Diseño de Software.

DES.ar Diseño arquitectónico.

Diseño y Arquitectura de Software: Continuación del estudio de patrones de diseño, macros y arquitecturas. Estudio de las arquitecturas middleware actual. Diseño de sistemas distribuidos utilizando el middleware. Medición de la teoría y el uso apropiado de las métricas en el diseño. El diseño de cualidades tales como rendimiento, seguridad, protección, reutilización, fiabilidad, etc. Medición interna, cualidades y complejidad del software. Evaluación y evolución de diseños. Fundamentos de la evolución del software, reingeniería, e ingeniería inversa.

Unidades de Diseño y Construcción que se estudian en dicho tema:

DES.str Estrategias del Diseño.

DES.ar Diseño arquitectónico.

DES.dd Diseño detallado.

DES.nst Notaciones del Diseño.

DES.ev Evaluación del Diseño.

CMP.ct Tecnologías de la Construcción.

CMP.ct.11 Middleware

Diseño de bajo nivel: Presenta software de diseño detallado y la construcción en profundidad. La cobertura en profundidad de los patrones de diseño y refactorización. Introducción a los sistemas de enseñanza escolar para el diseño. Análisis de diseños basados en interior criterios de calidad. Rendimiento y mejora la mantenibilidad. La ingeniería inversa.

Unidades de Diseño y Construcción que se estudian en dicho tema:

DES.dd Diseño detallado.

CMP.ct Tecnologías de la Construcción.

CMP.tl Herramientas de Construcción.

1.2 Diseño de Software.

Los conceptos, nociones, y terminología introducidas en este epígrafe, forman una base para entender el papel y alcance del diseño de software. En este epígrafe se exponen algunas técnicas de diseño, métodos, estrategias, estilos arquitectónicos, patrones, etc., para lograr entender la naturaleza y cómo funciona el proceso de diseñar un software.

1.2.1 Fundamentos del Diseño de Software

El diseño de software, visto como proceso, es la actividad del ciclo de vida de la cual los requisitos del software se analizan para producir una descripción de la estructura interna del software, sentando las bases para la construcción. Desempeña un papel importante en el desarrollo del software pues permite que la ingeniería de software produzca diversos modelos los cuales, después de ser analizados y evaluados para determinar si satisfacen los requisitos planteados, pasan a formar parte de la solución que se pondrá en desarrollo.

Para entender el papel del diseño de software, es importante entender el contexto y el ciclo de vida de la tecnología de dotación lógica. Entender las características principales del análisis de requisitos del software contra el diseño de software, contra la construcción del software y contra la prueba del software.

1.2.1.1 Proceso del Diseño de Software

El diseño de software se considera un proceso dividido en dos etapas, las cuales se encuentran entre el análisis de los requisitos de software y la construcción del software:

- **Diseño de la arquitectura del software (Diseño de Nivel Superior):** Describe la estructura del software, la organización e identificación de varios componentes a nivel superior, además se describe cómo el software se descompone y se organiza en componentes (la arquitectura) del software. Define las relaciones entre los principales elementos estructurales del programa.

- **Diseño detallado del Software:** Se describe cada componente suficientemente para tener en cuenta su construcción, además del comportamiento específico de cada uno de estos componentes. La salida de este proceso es un sistema de modelos y los artefactos que registran las decisiones principales que se han tomado. El diseño detallado se ocupa del refinamiento y la representación arquitectónica que lleva a una estructura de datos refinada y a las representaciones algorítmicas del software.

1.2.1.2 Técnicas Permisibles.

Según el diccionario del inglés de Oxford, un principio es “una verdad básica o una ley general que se utiliza como una base del razonamiento o guía de la acción.” Los principios del diseño de software, también llamados técnicas permisibles, son nociones dominantes que consideran fundamentalmente los diversos acercamientos y conceptos del diseño de software. Las técnicas que lo permiten son las siguientes. A continuación se detallan cada una de ellas.

Abstracción: La abstracción es el proceso de olvidarse de la información para poder tratar las cosas que son diferentes como si fueran iguales. En el contexto del diseño de software, dos mecanismos dominantes de la abstracción son la parametrización y la especificación. La abstracción por la especificación conduce a dos clases importantes de abstracción: abstracción procedimental y abstracción de datos.

- Una **abstracción de datos:** Es un conjunto de datos que describen un objeto como puede ser el carné de identidad de una persona, que está compuesta por conjunto de partes de información, pero que nos podemos referir a todos los datos, mencionando el nombre de la abstracción de datos.
- Una **abstracción procedimental:** Es una determinada secuencia de instrucciones que tiene una función limitada y específica, como puede ser "mover objeto", que supone la secuencia de pasos "abrir pinza", "mover hasta posición de destino 1", "cerrar pinza", "mover hasta posición 2", "abrir pinza", "mover hasta posición origen", "cerrar pinza". Estas abstracciones permiten al diseñador representar un objeto a diferentes niveles de detalle.

Cuando se considera una solución modular para cualquier problema, pueden formularse varios niveles de abstracción.

En el nivel superior se establece una solución de forma general en lenguaje natural. En los niveles superiores se utiliza una orientación más procedimental y en el nivel más bajo se establece una solución de forma que se pueda implementar de forma directa. (Rojas, 2006)

Acoplamiento y Cohesión: El acoplamiento se define como la fuerza de las relaciones entre los módulos, mientras que la cohesión es definida como la relación que existe entre los elementos que componen un módulo.

Descomposición y Modularización:

Descomposición y Modularización del software en partes más pequeñas e independientes, generalmente se realiza con la meta de poner diversas funcionalidades o responsabilidades en diversos componentes.

El software se divide en componentes con nombres y ubicaciones determinadas que se denominan módulos y que se integran para satisfacer los requisitos del proveedor.

El software monolítico (programa grande compuesto de un solo módulo), no puede ser estudiado fácilmente por un lector, ya que el número de caminos de control, el número de variables y la complejidad global harían el código prácticamente indescifrable. Para ello se recomienda dividir el proyecto en módulos para hacer más fácil su comprensión y manejo, llegando a la conclusión que la modularidad del software facilita el desarrollo del mismo, pero hasta un cierto límite, porque si

llegáramos a dividir el problema en infinitos módulos, los módulos tendrían una complejidad y un esfuerzo mucho menor, pero crecería el costo asociado a la creación de interfaces entre los módulos. Las ventajas de la modularización incluyen: Reducir la complejidad, Facilitar los cambios, Implementación más sencilla y permite el desarrollo paralelo de partes diferentes de un sistema. (Rojas, 2006)

Separación de la interfaz y de la puesta en práctica*:* La separación de la interfaz y de la puesta en práctica implica el definir de un componente, especificando una interfaz pública, además de los detalles de cómo se observa el componente.

1.2.1.3 Elementos claves en el Diseño del Software

Se tienen que tener en cuenta una serie de principios claves a la hora de diseñar software. Algunos son preocupaciones que todo el software debe tratar, por ejemplo, el funcionamiento de la calidad, otra cuestión importante es cómo se descomponen, se organizan y constituyen los paquetes de software. Esto es tan fundamental que todos los acercamientos del diseño deben tratarlo de un modo u otro. En cambio, otras ediciones se ocupan de un cierto aspecto del comportamiento del software que no está en el dominio del uso, pero que trata algunos de los dominios de soporte.

Concurrencia: Cómo se descompone el software en procesos, tareas e hilos, además cómo repartir con eficacia, relacionando la atomicidad y la sincronización.

Control y manejo de Eventos: Cómo organizar los datos y controlar el flujo, cómo manejar acontecimientos reactivos y temporales a través de varios mecanismos, tales como invocación y servicios implícitos repetidos.

Distribución de componentes: Cómo distribuir el software a través del hardware, cómo se comunican los componentes, cómo se puede utilizar el middleware para ocuparse de software heterogéneo.

Interacción y presentación: Cómo estructurar y organizar las interacciones con los usuarios y la presentación de la información (por ejemplo, separación de la presentación y de la lógica del negocio usando el modelo Vista-Controlador).

1.2.2 Estructura y arquitectura del software

Una arquitectura de software es una descripción de los subsistemas y de los componentes de un sistema de software y las relaciones entre ellas, procurando así definir la estructura interna. Según el diccionario del inglés de Oxford, […] “la manera en la cual se construye o se organiza cualquier parte del software que resulta”. Durante mediados de los años 90, la arquitectura del software comenzó a emerger como una disciplina más amplia, que implicaba el estudio de las estructuras y de las arquitecturas de software en una manera más genérica. Esto dio lugar a un número de ideas

interesantes sobre diseño del software en diversos niveles de la abstracción. Algunos de estos conceptos pueden ser útiles durante el diseño arquitectónico (por ejemplo, estilo arquitectónico) del software específico, así como durante su diseño detallado (por ejemplo, patrones de nivel inferior del diseño), pero pueden también ser útiles para diseñar sistemas genéricos, conduciendo al diseño de familias de los programas (también conocidos como líneas de productos).

La arquitectura del software se refiere a dos características importantes del software:

- La estructura jerárquica de los módulos del software, que no es más que una representación gráfica de la organización de los módulos del programa desde el punto de vista del control.
- La estructura de los datos que muestra las alternativas de organización, métodos de acceso, capacidad de asociación y procesamiento de la información.

La arquitectura del software se obtiene mediante un proceso de partición que relaciona los problemas del mundo real (definidos en el análisis de requerimientos) con las soluciones del software para resolver los problemas del software. A continuación se muestra el proceso de transición entre el análisis de requerimientos y el diseño. (Rojas, 2006)

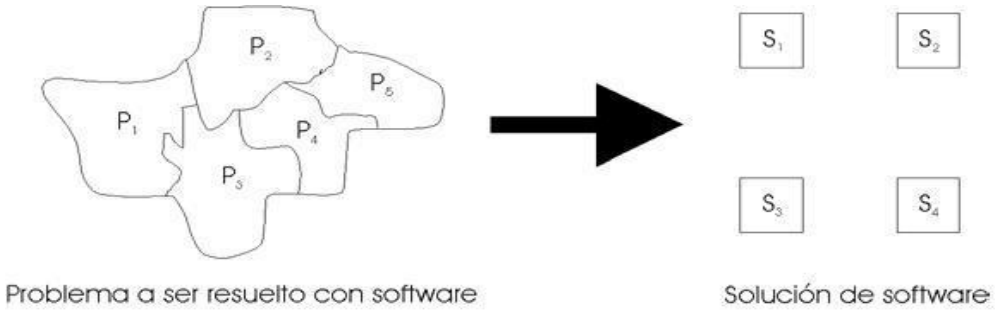


Figura1.6 Proceso de transición entre el análisis de requerimiento y el diseño. (Rojas, 2006)

1.2.2.1 Estructuras de la Arquitectura y Estilos Arquitectónicos.

Un Diseño de Software es un artefacto múltiple producido por el proceso del diseño e integrado generalmente por visiones relativamente independientes y ortogonales. Un estilo arquitectónico es “un sistema de apremios en una arquitectura que define un sistema o una familia de arquitecturas que las satisfagan”. Un estilo arquitectónico se puede considerar así, mientras que un meta-modelo pueda proporcionar la organización de alto nivel del software (arquitectura macro).Varios autores han identificado un número de estilos arquitectónicos importantes.

- Estructura general (capas, pipas, filtros y pizarra).
- Sistemas distribuidos (servidor de cliente, tres capas, corredor).
- Sistemas interactivos (Modelo Vista-Controlador, Presentación-Abstracción-Control).

- Sistemas adaptables (micro-núcleo, reflexión).
- Otros (batch, interpreters, control, basados en las reglas de proceso).

1.2.2.2 Patrones de Diseño (patrones arquitectónicos micro).

Un patrón es "una solución común a un problema común en un contexto dado." Mientras que los estilos arquitectónicos se pueden ver como patrones que describen la mayor parte de estos conceptos, se pueden considerar como tentativas de describir y reutilizar así el conocimiento genérico del diseño. Otros patrones del diseño se pueden utilizar para describir los detalles en un nivel más bajo, más local (arquitectura micro). A continuación se muestra los diferentes tipos de patrones de diseño según sus clasificaciones:

- **Patrones de creación:** (builder, factory, prototipo, y singleton).
- **Patrones estructurales:** (adapter, bridge, composite, decorator, façade, flyweight, y proxy).
- **Patrones del comportamiento:** (command, interpreter, iterator, mediator, memento, observer, state, strategy, template, visitor).

1.2.2.3 Familia de programas y Frameworks.

Una posible opción para permitir la reutilización de los diseños y de los componentes del software es diseñar las familias del software, también conocidas como líneas del producto de software. Estas pueden ser hechas identificando las concordancias entre los miembros de tales familias y por los componentes reutilizables y customizables entre miembros de la familia. En programación orientada a objetos, una clave relacionada es la del marco: un subsistema parcialmente completo del software que puede ser ampliado apropiadamente instalando los plug-ins específicos (también conocidos como puntos calientes).

1.2.3 Análisis de la Calidad y Evaluación.

Esta sección incluye generalidades de la calidad y evaluación que se relacionen específicamente con el diseño de software.

1.2.3.1 Atributos de Calidad.

Varios atributos son importantes para obtener un diseño de software de buena calidad, por ejemplo la capacidad de mantenimiento, la portabilidad, el testeo, la trazabilidad, la corrección y la robustez. Una distinción interesante es la que está entre las cualidades de la calidad discernible en el tiempo de

ejecución, como son el funcionamiento, la seguridad, la disponibilidad, y la utilidad. Las cualidades no discernibles en el tiempo de ejecución como son la modificabilidad, la portabilidad, la reutilización, la integridad y testeabilidad, y las cualidades relacionadas con las calidades intrínsecas de la arquitectura como son la integridad y la corrección.

1.2.3.2 Técnicas de análisis de Calidad y Evaluación.

Existen varias técnicas que pueden ayudar a asegurar la calidad de un diseño de software:

- Revisiones del diseño del software: informal o semi-formal, a menudo basado en grupo, las técnicas para verificar y para asegurar la calidad de los artefactos del diseño (por ejemplo, revisiones de la arquitectura revisiones de diseño, e inspecciones.
- Análisis estático: análisis estático formal o semi-formal (ningún ejecutable) que se puede utilizar para evaluar un diseño (por ejemplo, el análisis o el cross-checking automatizado).
- Simulación y prototipado: técnicas dinámicas para evaluar un diseño (por ejemplo, simulación o prototipo de la viabilidad del funcionamiento).

A lo largo de este proceso, la calidad del diseño se evalúa mediante una serie de revisiones técnicas formales, que son una actividad de garantía del software cuyos objetivos son:

- Descubrir errores en la función, lógica o implementación de cualquier representación del software.
- Verificar el cumplimiento de los requisitos.
- Garantizar el cumplimiento de los estándares.
- Conseguir un desarrollo uniforme del software.
- Obtener proyectos que hagan más sencillo los trabajos técnicos (análisis que permitan buenos diseños, diseños que permitan implementaciones sencillas, estrategias de pruebas que faciliten éstas,…).

Cada revisión técnica formal se lleva a cabo mediante una reunión, pues sólo tendrá éxito si está bien planificada, controlada y atendida.

A continuación se lista una serie de criterios para determinar la calidad del software.

1. Un diseño debe tener una organización jerárquica.
2. Un diseño debe ser modular, es decir, el software debe estar dividido en elementos que realicen funciones específicas.
3. Un diseño debe tener representaciones distintas y separadas de los datos y de los procedimientos.
4. Un diseño debe llevar a módulos que exhiban características funcionales independientes.

5. Un diseño debe conducir a interfaces que reduzcan la complejidad de las conexiones entre los módulos y el exterior.
6. Un diseño debe obtenerse mediante un método que sea reproducible y que esté dirigido por la información obtenida durante el análisis de requerimientos.

Un buen Diseño de Software no se consigue fácilmente, es el resultado de la aplicación de los principios fundamentales de diseño, de una metodología sistemática y de una revisión exhaustiva. (Rojas, 2006)

1.2.3.3 Métricas.

Las medidas se pueden utilizar para determinar o para estimar cuantitativamente varios aspectos del tamaño, de la estructura, o de la calidad de un diseño del software. Estas medidas se clasifican en dos amplias categorías:

- **Diseño de medidas orientada a función (estructuradas):** La estructura del diseño, obtenida sobre todo con la descomposición funcional; representado generalmente como una carta de estructura (a veces llamada un diagrama jerárquico) en la cual varias medidas pueden ser computadas.

- **Diseño de medidas orientada a objetos:** La estructura total del diseño se representa a menudo como el diagrama de la clase, en el cual varias medidas pueden ser computadas. Las medidas en las características del contenido interno de cada clase pueden también ser computadas.

1.2.4 Notaciones del Diseño de Software.

Existen muchas notaciones e idiomas para representar los artefactos del diseño de software. Algunos se utilizan principalmente para describir la organización estructural de un diseño, otras para representar el comportamiento del software. Ciertas notaciones se utilizan sobre todo durante el diseño arquitectónico y otros, principalmente durante el diseño detallado, aunque algunas notaciones se pueden utilizar en ambos pasos. Además, algunas notaciones se utilizan sobre todo en el contexto de métodos específicos.

1.2.4.1 Descripción Estructural (vista estática).

Las siguientes notaciones, sobre todo gráficas, describen y representan los aspectos estructurales del diseño de software, las cuales describen los componentes principales y cómo se interconectan (visión estática):

- **Lenguajes descriptivos de la arquitectura:** Textuales, a menudo formal. Estos lenguajes describen una arquitectura del software en términos de componentes y conectores.
- **Diagramas de la clase y objeto:** Usados para representar un sistema de clases (objetos) y de sus correlaciones.
- **Diagramas de componentes:** Usados para representar un sistema de componentes ("parte física y reemplazable de un sistema al cual conforma y proporciona la realización de un sistema de interfaces") y de sus correlaciones.
- **Tarjetas del colaborador de la responsabilidad de la clase (CRCs):** Denotan los nombres de los componentes (clases), de sus responsabilidades y los nombres de sus componentes de colaboración.
- **Diagramas de despliegue:** Representan un sistema de nodos (físico) y de sus correlaciones, además modelan los aspectos físicos de un sistema.
- **Diagramas de la Entidad-Relación (ERDs):** Representan los modelos conceptuales de los datos almacenados en los sistemas de información.
- **Lenguaje descriptivo de la interfaz (IDLS):** Programación con lenguajes usados para definir las interfaces (nombres y tipos de operaciones exportadas) de los componentes de software.
- **Diagramas de la estructura de Jackson:** Usados para describir las estructuras de datos en términos de secuencia, selección, e iteración.
- **Estructura de cartas:** Usados para describir la estructura que llamaba los programas (el módulo que llama, y que es llamado por otro módulo).

1.2.4.2 Descripciones del comportamiento (visión dinámica).

Las siguientes notaciones y lenguajes, algunos gráficos y otros textuales, se utilizan para describir el comportamiento dinámico del software y de los componentes. Muchas de estas notaciones son útiles sobre todo, pero no exclusivamente, durante el diseño detallado.

- **Diagramas de actividad:** Muestran el flujo del control de la actividad ("ejecución no-atómica en curso dentro de una máquina del estado") a la actividad.
- **Diagramas de colaboración:** Muestran las interacciones que ocurren entre un grupo de objetos, sus acoplamientos, y los mensajes que intercambian en estos acoplamientos.
- **Organigramas de datos:** Muestran los flujos de datos entre un sistema y los procesos.
- **Tablas y diagramas de decisión:** Representan combinaciones complejas de las condiciones y de las acciones.

- **Organigramas y organigramas estructurados:** Representan el control de flujo y de las acciones asociadas que se realizarán.
- **Diagramas de Secuencia:** Muestran las interacciones entre un grupo de objetos, con énfasis sobre el tiempo de ordenación de mensajes.
- **Transición de estado y diagramas de carta de estado:** Demostraban el control de flujo de un estado a otro en una máquina de estados.
- **Lenguajes formales de especificación:** Los lenguajes textuales que utilizan nociones básicas de matemáticas (por ejemplo, lógica, sistema, secuencia), para obtener de forma rigurosa y abstracta, definir interfaces y comportamientos del componente de software, a menudo en términos de pre y post-condiciones.
- **Lenguajes del diseño de pseudo código del programa (PDLs):** Programa estructurado como los lenguajes usados para describir, generalmente en la etapa detallada del diseño, el comportamiento de un procedimiento o el método.

1.2.5 Estrategias y métodos del Diseño de Software

Existen varias estrategias generales para ayudar a dirigir el proceso de diseño. Al contrario que en las estrategias generales, los métodos son más específicos, sugieren y proporcionan generalmente un sistema de notaciones que se utilizarán con el método, una descripción del proceso que se utilizará después del método y un sistema de pautas al usar el método. Tales métodos son útiles como medios de transferir conocimiento y como marco común para los equipos de los ingenieros de software.

Estrategias generales: Algunos de los ejemplos citados de las estrategias generales útiles en el proceso del diseño son divide-y-vencerás y el refinamiento de arriba hacia abajo contra las estrategias buttom-up, abstracción de los datos y el ocultar de la información, uso de la heurística, uso de patrones y lenguajes de patrones, uso de un acercamiento iterativo e incremental.

Diseño (estructurado) Orientado a Funciones: Este es uno de los métodos clásicos del diseño de software, donde los centros de descomposición identifican las funciones del software y después elaboran y refinan de una manera de arriba hacia abajo. El diseño estructurado se utiliza generalmente después de análisis estructurado, produciendo así, entre otras cosas, organigramas de datos y de descripciones de procesos asociados. Los investigadores han propuesto varias estrategias (por ejemplo, análisis de la transformación, análisis de la transacción) y la heurística (por ejemplo, fan-in/fan- out, alcance del efecto contra el alcance del control). (Rojas, 2006)

Diseño orientado a objeto: El campo del diseño de software se ha desarrollado basado en el diseño orientado a objeto de mediados de los años ochenta, (sustantivo = objeto; verbo = método; adjetivo = cualidad) en este diseño la herencia y el polimorfismo juega un papel muy importante. Es una fase de la metodología orientada a objetos para el desarrollo de Software. Su uso induce a los programadores a pensar en términos de objetos, en vez de procedimientos, cuando planifican su código. Un objeto agrupa datos encapsulados y procedimientos para representar una entidad. Un programa orientado a objetos es descrito por la interacción de esos objetos. (Rojas, 2006)

Diseño centrado en estructuras de datos: El diseño centrado en estructura de datos (Jackson, Warnier-Orr) comienza desde las estructuras de datos que un programa manipula, más bien las funciones que realiza. La ingeniería de software primero describe las estructuras de datos de entrada y de salida (que usan los diagramas de la estructura de Jackson) desarrollándose la estructura de control del programa basada en estos diagramas de estructura de datos. La heurística se ha propuesto para tratar como caso especial, cuando hay una unión mal hecha entre la entrada y las estructuras de la salida. (Rojas, 2006)

Diseño basado en componente (CBD): Un componente de software es una unidad independiente, teniendo bien definidos las interfaces y dependencias que se pueden componer y desplegar independientemente. El diseño basado en componente trata las ediciones relacionadas con el abastecimiento, desarrollo, e integración de tales componentes para mejorar la reutilización. (Rojas, 2006)

1.3 Construcción del Software

La construcción de software es un mero proceso de diseño, pero el hecho de que, al terminar, este diseño tome la forma de una especificación final en un lenguaje ejecutable para ordenador, provoca que el diseño mencionado se asimile a un proceso completo que incluye también la fabricación, de manera que el proceso de producción y la comprobación final de la calidad del proceso y del producto son francamente complejos y problemáticos. (García, 2002)

1.3.1 Fundamentos de la Construcción

Los fundamentos de la construcción del software incluyen:

- Minimizar la complejidad.
- Anticiparse a los cambios.

- Construir para verificar.
- Estándares en la construcción.

Los tres primeros conceptos se aplican tanto al diseño como a la construcción. Las siguientes secciones definen estos conceptos y describen cómo se aplican a la construcción.

El término construcción del software hace referencia a la creación detallada de software operativo y significativo, por medio de una combinación de codificación, verificación, pruebas unitarias, pruebas de integración y depuración. El área de conocimiento de la construcción del software está vinculada a todas las otras áreas de conocimiento más fuertemente al diseño del software y a las pruebas del software. Esto se debe a que el proceso mismo de construcción del software cubre tanto el diseño significativo de software como las actividades de pruebas. También utiliza las salidas del diseño y proporciona una de las entradas para las pruebas. Las fronteras detalladas entre el diseño, la construcción y las pruebas (si es que existen) varían dependiendo de los procesos de ciclo de vida del software utilizados en un proyecto.

A pesar de que se pueda realizar parte del diseño detallado antes de la construcción, mucho del trabajo del diseño se lleva a cabo durante la actividad misma de la construcción. Por lo que estas dos áreas están estrechamente relacionadas.

Por medio de la construcción los ingenieros del software realizan tanto pruebas unitarias, como pruebas de integración de su trabajo. De tal manera que el área de conocimiento de construcción del software está también vinculada de cerca al área de conocimiento de pruebas del software. La construcción del software, por lo general, produce el mayor número de elementos de configuración que se necesitan gestionar en un proyecto de software (archivos de código fuente, contenido, casos de pruebas, etc.). De este modo, el área de conocimiento de construcción del software también está vinculada de cerca al área de conocimiento de gestión de la configuración del software.

Dado que la construcción del software tiene una gran dependencia de las herramientas y de los métodos, y de que se trata probablemente del área de conocimiento que más herramientas tiene y utiliza, está vinculada al área de conocimiento de herramientas y métodos de la ingeniería de software.

Aunque la calidad del software es importante en todas las áreas de conocimientos, el código es el último entregable de un proyecto de software y, por tanto, la calidad del software está vinculada de cerca a la construcción del software.

Entre las disciplinas descritas de la ingeniería del software, el área de conocimiento de construcción del software es lo más parecido a la ciencia informática en su uso del conocimiento de algoritmos y de las prácticas detalladas de codificación, ambas son consideradas, con frecuencia, como pertenecientes al dominio de la ciencia informática. También está relacionada con la gestión del proyecto en la medida en que la gestión de la construcción pueda presentar retos considerables.

Minimizar la complejidad

Uno de los más importantes factores dentro de la construcción de software es minimizar la complejidad. Se aplica esencialmente a todos los aspectos de la misma y es de gran importancia para el proceso de verificación y pruebas dentro de esta fase.

Sólo se logra alcanzar una reducida complejidad por medio del énfasis en la creación de código simple y legible. Se logra además mediante el uso de estándares, posteriormente en el apartado de Estándares de Construcción, y mediante numerosas técnicas específicas que también se tratarán más adelante en el apartado Codificación. También se apoya en las técnicas de calidad enfocadas a la construcción resumida en el apartado calidad de la construcción.

Anticiparse a los cambios

Anticiparse a los cambios es una técnica muy importante dentro de la construcción de software ya que los mismos a lo largo del tiempo van a sufrir cambios. El software es parte de los ambientes externos y cualquier cambio que se efectúe en esos ambientes externos, va a afectarlo inevitablemente. Algunas de las técnicas en las que se apoya la anticipación a los cambios, serán tratadas más adelante en el apartado Codificación.

Construir para verificar

Construir para verificar significa construir el software de forma tal que le resulte fácil a los ingenieros del software sacar a relucir los fallos al estar escribiendo el código.

Las técnicas específicas que se utilizan para construir para verificar incluyen el seguimiento de estándares de codificación que permitan las revisiones del código, las pruebas unitarias, la organización del código que permita pruebas automáticas, y el uso restringido de estructuras de lenguaje que sean complejas o difíciles de entender, entre otras.

Estándares en la construcción

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Al comenzar un proyecto de software, se debe establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. El mejor método para asegurarse de que un equipo de programadores mantenga un código de calidad es establecer un estándar de codificación sobre el que se efectuarán luego revisiones del código de rutinas. La adopción de un estándar de codificación sólo es viable si se sigue desde el principio hasta

el final del proyecto de software. No es práctico, ni prudente, imponer un estándar de codificación una vez iniciado el trabajo. (2006)

Los estándares que afectan directamente a elementos de la construcción incluyen:

- Métodos de comunicación (por ejemplo, estándares para los formatos de los documentos y de los contenidos)
- Programación de lenguajes (por ejemplo, estándares de lenguaje para lenguajes como Java y C++)
- Plataformas (por ejemplo, estándares de interfaces del programador para llamadas al sistema operativo).
- Herramientas (por ejemplo, estándares diagramáticos para notaciones como UML (Lenguaje Unificado de Modelado)

Uso de estándares externos. Construir depende del uso de estándares externos para los lenguajes de construcción, las herramientas de construcción, las interfaces técnicas, y las interacciones entre la construcción del software y las otras áreas de conocimiento.

Los estándares provienen de numerosas fuentes, incluyendo las especificaciones de interfaz del hardware y del software, tales como el Object Management Group (OMG) y las organizaciones internacionales tales como la IEEE o ISO.

Uso de estándares internos. Los estándares también pueden crearse partiendo de una base organizacional a un nivel corporativo o para su uso en proyectos específicos. Estos estándares permiten la coordinación de actividades de grupo, el minimizar la complejidad, el anticipar los cambios y el construir para verificar.

1.3.2 Gestión de la Construcción

Es muy importante realizar una adecuada planificación y gestión de la construcción y mantener mediciones acerca del desempeño y calidad de los programadores, de qué es lo que se va a desarrollar, cuándo y cómo. De esta manera se garantiza que el producto final tenga la calidad requerida.

1.3.2.1 Modelos de Construcción

Existen numerosos modelos para el desarrollo del software, algunos ponen más énfasis en la construcción que otros.

Algunos modelos son más lineales que otros desde el punto de vista de la construcción tales como los modelos en cascada y los del ciclo de vida de entregas por etapas. Estos modelos tratan la construcción como una actividad que sucede sólo después de que se haya completado un significativo trabajo con los prerrequisitos incluyendo un trabajo detallado sobre los requisitos, un extensivo trabajo

sobre el diseño y una planificación detallada. Los enfoques más lineales tienden a poner el énfasis en las actividades que preceden a la construcción (requisitos y diseño), y tienden a crear separaciones más marcadas entre las actividades. En estos modelos, la codificación sería el punto de mayor énfasis de la construcción.

Otros modelos son más iterativos, tales como el prototipado evolucionista, Programación Extrema y Scrum. Estos enfoques tienden a tratar la construcción como una actividad que ocurre en estos momentos con otras actividades de desarrollo del software incluyendo los requisitos, el diseño y la planificación. Estos enfoques tienden a mezclar el diseño, la codificación y las actividades de pruebas, y con frecuencia tratan la combinación de actividades como una construcción.

Por consiguiente, lo que está considerado como construcción depende hasta cierto grado del modelo de ciclo de vida utilizado.

1.3.2.2 Planificación de la Construcción

Un aspecto clave de la planificación de la actividad de construcción es la elección de un método de construcción, ya que afecta hasta dónde se realizan los prerrequisitos de construcción, el orden en el que se realizan, y el grado hasta el que se espera que se completen antes de que comience el trabajo de construcción.

El modo de afrontar la construcción afecta a la habilidad del proyecto para reducir la complejidad, anticipar cambios y construir para verificar. Cada uno de estos objetivos puede también afrontarse en los niveles de proceso, requisitos y diseño, pero también influye sobre ellos la elección de un método de construcción.

La planificación de la construcción también define el orden en el que se crean e integran, según el método elegido, los componentes, los procesos de gestión de la calidad del software, la asignación de tareas a ingenieros del software específicos y el resto de las tareas.

1.3.2.3 Métricas de la Construcción

Tradicionalmente se ha medido el tamaño del software mediante distintas métricas: recuento de las líneas de código, número de programas fuente, o técnicas similares que no resultan aceptables como una buena práctica profesional, porque:

- Su resultado depende fuertemente del entorno técnico y el lenguaje de programación utilizado.
- Varía en función de la habilidad de cada programador y del uso de normas y metodologías.
- No resultan significativas al usuario ni a la dirección.

Cuando se trata de establecer métricas de productividad y calidad en la construcción de software, o realizar estimaciones de coste y duración, es imprescindible disponer de una medida fiable y comprensible del tamaño de lo que se construye. (Jones, 2000)

Ejemplos de Métricas:

La **métrica del punto función** es un método utilizado en ingeniería del software para medir el tamaño del software. Fue definida por Allan Albrecht, de IBM, en 1979 y pretende medir la funcionalidad entregada al usuario independientemente de la tecnología utilizada para la construcción y explotación del software, y también ser útil en cualquiera de las fases de vida del software, desde el diseño inicial hasta la explotación y mantenimiento.

La **métrica de Puntos de Casos de Uso** se desarrollo en el año 1993 por Gustav Karner para poder finalmente obtener estimaciones de esfuerzo sobre productos de software orientados a objetos. Es un método de estimación y cálculo de tamaño del software basado en cuentas hechas sobre los casos de uso para un sistema de software.

El método exige la existencia de un modelo de casos de uso, por lo que la labor deberá ser hecha cuando exista algún entendimiento del dominio del problema o cuando se esté realizando las labores de arquitectura y dimensionamiento del tamaño del sistema. Por lo general, estas condiciones están dadas al término de las actividades de Análisis.

En términos simples, el método requiere de casos de uso en modo textual y gráfico sólo en términos de mayor claridad, se revisan en detalle los casos de uso seleccionados en la etapa del proyecto que se defina y se realizan los siguientes pasos:

1. Cuantificación de características funcionales del Sistema:

- Clasificación de Actores, obtención del **P**eso de **A**ctores **S**in **A**justar (PASA).
- Clasificación de los Casos de Uso, obtención del **P**eso de **T**ransacciones **S**in **A**justar (PTSA)
- Obtención del Peso o **P**untos de **C**asos de **U**so **S**in **A**justar (PCUSA).

2. Cuantificación de características no funcionales del Sistema:

- Clasificación de **F**actores de **C**omplejidad **T**écnica (FCT)

- Clasificación de **F**actores **A**mbientales (FA)
- Cálculo de **P**untos de **C**asos de **U**so Ajustados (PCU) (Hernández, 2002)

La métrica **Mark II** (United Kingdom Software Metrics Association) fue desarrollada por KPMG en 1986, definida y publicada por Charles Symons en 1991, Adoptada por la UKSMA (United Kingdom Software Metrics Association). Intenta ser un método de medición continua a lo largo del ciclo de vida de una aplicación, frente a unas mediciones más estáticas del IFPUG-FPA.

Otra métrica es la **FFP** (Full Function Point) la cual fue desarrollada por COSMIC (Common Software Measurement International Consortium) y es una adaptación del FPA con vistas al software real-time (equipos de telecomunicaciones, sistemas operativos y similares). (Herron, 2000)

1.3.3 Consideraciones Prácticas

La construcción es una actividad en la cual el software tiene que enfrentarse a restricciones arbitrarias del mundo real, y aún así, hacer exactamente lo que se pide. Dado que posee restricciones muy parecidas a las restricciones del mundo real, la construcción está guiada por buenas prácticas más que otras áreas de conocimiento.

1.3.3.1 Diseño de la Construcción

Independientemente de la asignación que se realice con respecto al diseño, en el nivel de construcción también se trabaja un poco el diseño detallado y ese trabajo de diseño tiende a seguir restricciones específicas impuestas por un problema del mundo real que está siendo afrontado por el software.
Así como los obreros de una construcción que construyen una estructura física tienen que realizar modificaciones a pequeña escala para cubrir huecos no previstos en los planes del constructor, los obreros de la construcción del software tendrán que hacer modificaciones en una mayor o menor escala para revelar los detalles del diseño de software durante la construcción.
Cada uno de los detalles de la actividad de diseño a nivel de la construcción son esencialmente los mismos que se describen en el área de conocimiento del diseño de software, pero se aplican en una escala inferior.

1.3.3.2 Lenguajes de Construcción

Los lenguajes de construcción incluyen todas las vías de comunicación mediante las cuales un humano puede especificar una solución ejecutable para un problema de un ordenador.
El tipo más simple de lenguaje de construcción es un lenguaje de configuración en el que los ingenieros del software eligen de entre un conjunto limitado de opciones predefinidas para crear nuevas o típicas instalaciones del software. Los archivos de configuración basados en texto utilizados

tanto en los sistemas operativos de Windows como de Unix son ejemplos de esto, y otro ejemplo son las listas de selección en forma de menú de algunos generadores de programas.

Los lenguajes de herramientas se utilizan para construir aplicaciones partiendo de las herramientas (conjuntos integrados de partes reutilizables específicas de las aplicaciones), y son más complejos que los lenguajes de configuración. Los lenguajes de herramientas pueden definirse explícitamente como lenguajes de programación de aplicaciones (por ejemplo, scripts), o pueden simplemente estar implícitos en el conjunto de interfaces de las herramientas.

Los lenguajes de programación son el tipo más flexible de lenguaje de construcción. También son los que menos información contienen acerca de las áreas específicas de la aplicación y los procesos de desarrollo, y por tanto requieren el mayor entrenamiento y destreza posibles para utilizarlos con eficacia.

Existen tres tipos generales de notación utilizados para los lenguajes de programación, a saber:

- Lingüísticos.
- Formales.
- Visuales.

Las **notaciones lingüísticas** se distinguen en particular por la utilización de cadenas de texto del tipo palabra para representar construcciones complejas de software, y por la combinación en patrones de cadenas del tipo palabra que tienen una sintaxis del tipo sentencia. Utilizadas adecuadamente, cada una de estas cadenas debería tener una fuerte connotación ofreciendo un entendimiento intuitivo inmediato de lo que sucedería cuando se ejecutara la construcción del software subyacente.

Las **notaciones formales** se apoyan menos en los significados de las palabras y cadenas de texto intuitivos, y más en las definiciones precisas, sin ambigüedad, y formales (o matemáticas). Las construcciones formales también utilizan modos de combinar símbolos definidos con precisión que evitan la ambigüedad de muchas construcciones del lenguaje natural.

Las **notaciones visuales** se apoyan poco en las notaciones orientadas al texto tanto de la construcción lingüística como de la formal, y en cambio sí se apoyan en una interpretación visual directa y en la colocación de las entidades visuales que representan al software subyacente. La construcción visual tiende a estar un tanto limitada por la dificultad de hacer declaraciones “complejas” utilizando sólo el movimiento de entidades visuales en un despliegue. Sin embargo, puede resultar ser muy importante en los casos en los cuales la principal tarea de programación es simplemente construir y “ajustar” una interfaz visual a un programa, cuyo comportamiento detallado ha sido definido anteriormente.

1.3.3.3 Codificación

Las consideraciones que se mencionan a continuación son aplicadas a la actividad de construcción del código del software:

- Técnicas para crear código fuente comprensible, que incluye la asignación de nombres y el esquema del código fuente.
- Utilización de clases, tipos enumerados, variables, constantes predefinidas, y otras entidades similares.
- Utilización de estructuras de control.
- Tratamiento de las condiciones de error, tanto los errores planeados como las excepciones (la entrada de datos malos, por ejemplo).
- Prevención de brechas en la seguridad a nivel de código (el búfer o el índice de la matriz se desborda, por ejemplo).
- Utilización de recursos por medio del uso de mecanismos de exclusión y disciplina en el acceso serial a recursos reutilizables (incluyendo threads o bloqueos de bases de datos).
- Organización del código fuente (en declaraciones, rutinas, clases, paquetes u otras estructuras).
- Documentación del código.
- Puesta a punto del código.

1.3.3.4 Pruebas de Construcción

Existen dos tipos de prueba a la hora de construir, que por lo general las realiza el mismo ingeniero del software que escribió el código:

- **Pruebas unitarias:** Consisten en probar o testear piezas de software pequeñas; a nivel de secciones, procedimientos, funciones y módulos; aquellas que tengan funcionalidades específicas. Dichas pruebas se utilizan para asegurar el correcto funcionamiento de secciones de código, mucho más reducidas que el conjunto, y que tienen funciones concretas con cierto grado de independencia.
- **Pruebas de integración:** Se realizan una vez que las pruebas unitarias fueron concluidas *exitosamente*; con éstas se intenta asegurar que el sistema completo, incluso los subsistemas que componen las piezas individuales grandes del software, funcionen correctamente al operar e interoperar en conjunto. (Corporation, 2006)

El propósito de las pruebas de construcción es reducir el intervalo de tiempo entre el tiempo en el que se introducen fallos en el código y el tiempo en el que se detectan esos fallos. En algunos casos, las

pruebas de construcción se llevan a cabo después que se ha escrito el código. En otros casos, se pueden elaborar casos de pruebas antes de que se escriba el código.

Es frecuente que las pruebas de construcción incluyan un subconjunto de tipos de pruebas, las cuales se describen en el área de conocimiento de pruebas del software. Por ejemplo, no es típico de las pruebas de construcción el incluir las pruebas del sistema, las pruebas alfa, las pruebas beta, las pruebas de estrés, las pruebas de construcción, las pruebas de posibilidad de uso, u otros tipos de pruebas más especializadas.

Se han publicado dos estándares sobre dicho tema:

_ *IEEE STD 829-1998, IEEE Standard for Software Test Documentation.*

_*IEEE STD 1008-1987, IEEE Standard for Software Unit Testing.*

1.3.3.5 Reutilización

Tal y como se afirma en la introducción del (IEEE1517-99):

"El implementar la utilización del software conlleva algo más que crear y utilizar librerías de recursos. Requiere formalizar la práctica de la reutilización por medio de la integración de procesos y actividades de reutilización en el ciclo de vida del software."

Las tareas relacionadas con la reutilización en la construcción del software durante su codificación y pruebas son:

- La selección de unidades, bases de datos, procedimientos de pruebas o datos de pruebas reutilizables.
- La evaluación de la posibilidad de reutilización del código o de las pruebas.
- Comunicar la información sobre reutilización realizada en el código nuevo, los procedimientos de pruebas o los datos de pruebas.

1.3.3.6 Calidad de la Construcción

La calidad en cualquier ámbito está muy ligada a la consecución de la excelencia. No es algo alcanzable, ni un objetivo en sí mismo, sino un medio para mejorar la productividad y reducir los riesgos. Aunque es una paradoja, la construcción de software sigue teniendo un fuerte componente artesanal, donde la calidad depende mucho de la destreza y experiencia de los programadores.

Si se quiere construir un software correcto, robusto, extensible y reusable se debe ser conscientes que el mejor camino para lograrlo está precisamente en hacer una programación de calidad. Existen además otras razones de mucha importancia para aplicar técnicas de buena codificación. Los desarrolladores no pueden olvidar que el código fuente se mantiene y reutiliza, por lo que para facilitar estas tareas se debe cuidar la programación.

Existen algunas técnicas particulares, así como otras generales, para el ejercicio de una buena codificación entre las que se pueden citar:

- Las pruebas unitarias y las pruebas de integración mencionadas anteriormente.
- Revisiones Técnicas.
- Uso de Aserciones y Validaciones. Uno de los pilares de una programación sólida es conseguir detectar rápidamente los errores durante el desarrollo para evitar que éstos pasen al producto que se entrega al cliente. El mecanismo principal para lograr este objetivo es el uso generalizado y profuso de validaciones en el código para detectar cualquier uso no válido de los métodos, clases, funciones y procedimientos. Una posible clasificación de las validaciones es la siguiente:
  - Validaciones Internas. Se escriben para capturar estados erróneos durante el desarrollo.
  - Validaciones Paramétricas.
  - Validaciones Contextuales a nivel de Clase.
  - Validaciones Contextuales a nivel de Aplicación.
  - Validaciones Externas. Se escriben para evitar que se produzcan estados erróneos en la aplicación debidos a entradas no válidas por parte de los componentes externos a la aplicación (usuarios, ficheros, etc.).

Las únicas validaciones que permanecerán en el producto final son las validaciones externas (sobre todo aquellas concernientes a la interacción con el usuario), las internas son usadas durante el desarrollo del código. (García, 2002)

La elección de la técnica a poner en práctica depende de la naturaleza del software que se está construyendo, así como del conjunto de habilidades que posean los ingenieros del software que llevan a cabo la construcción.

Las actividades de calidad de la construcción tienen un enfoque diferente al de las otras actividades de calidad. Las actividades de calidad de la construcción se centran en el código y en los artefactos que están estrechamente relacionados con el código: diseños en pequeña escala, en oposición a otros artefactos que están menos directamente ligados al código, tales como requisitos, diseños de alto nivel y planes.

1.3.3.7 Integración

Una actividad clave durante la construcción es la integración de rutinas, clases, componentes y subsistemas construidos por separado. Además, un sistema particular del software podría necesitar ser integrado con otros sistemas de software o de hardware.

También se puede citar la integración incremental. Si diariamente se construye el software, se detectan rápidamente los problemas de integración y es mucho más fácil si se corrigen cuando los cambios, que pueden ser la causa, aún están frescos en la memoria. Además, como parte del proceso de

construcción, se pueden ejecutar los test unitarios, lo que proporcionará la posibilidad de detectar errores no sólo relacionados con la integración.

Los intereses relacionados con la integración de la construcción incluyen planificar la secuencia en la que se integrarán los componentes, crear andamiajes que soporten versiones provisionales del software, determinar el grado de pruebas y la calidad del trabajo realizado sobre los componentes antes de que sean integrados, y determinar los puntos en el proyecto en los que se prueban las versiones provisionales del software.

1.4 Gestión del conocimiento

La universidad es uno de los elementos más importantes dentro del sistema de innovación, ya sea por el suministro de capital humano o tecnológico. A medida que crece la importancia de los conocimientos en el campo de las innovaciones, la universidad debe mostrar mayor protagonismo en la innovación industrial.

Según José Alberto Valdés Fernández (…) "Gestión del conocimiento es el proceso caracterizado por una transformación continua de datos en informaciones, de informaciones en conocimientos y de conocimientos en conocimientos." (Valdés, 2005)

Pérez Rodríguez y Coutin Domínguez (…) la definen como "un proceso mediante el cual se desarrolla, estructura y mantiene la información, con el objetivo de transformarla en un activo crítico y ponerla a disposición de una comunidad de usuarios, definida con la seguridad necesaria. Incluye el aprendizaje, la información, las aptitudes y la experiencia desarrollada durante la historia de la organización." (Pérez, 2005)

Objetivos de la Gestión de Conocimientos.

A consideración de varios autores, los objetivos de la gestión del conocimiento son perfectamente resumidos por Díaz Muñante en:

- Formular una estrategia de alcance organizacional para el desarrollo, adquisición y aplicación del conocimiento.
- Implantar estrategias orientadas al conocimiento.
- Promover la mejora continua de los procesos de negocio, enfatizando en la generación y utilización del conocimiento.
- Monitorear y evaluar los logros obtenidos mediante la aplicación del conocimiento.
- Reducir los tiempos de ciclos en el desarrollo de soluciones a los problemas.
- Reducir los costos asociados a la repetición de errores. (Muñante, 2003)

Nonaka y Takeuchi proponen un modelo de proceso de creación de conocimiento para entender la naturaleza dinámica de la creación del conocimiento y manejar tal proceso con eficacia.
Dichos autores clasifican el conocimiento en:
Conocimiento explícito: El conocimiento explicito es formal y sistemático, está codificado por lo que puede ser comunicado y compartido con cierta facilidad. Ejemplos de este conocimiento son los manuales de la empresa, etc.
Conocimiento tácito: El conocimiento tácito o implícito es aquel de difícil expresión y definición, es complicado de formalizar y por lo tanto difícil de comunicar. Es un conocimiento personal formado por experiencias de trabajo, vivencias, etc. (Takeuchi, 1995)

De la consideración de estos dos tipos de conocimiento, se deriva la denominada "espiral de conocimiento" (NONAKA, y otros, 1995), que plantea que la problemática de la generación de conocimiento organizacional reside, principalmente, en cómo extender el conocimiento individual al resto de la organización y que este mismo conocimiento compartido vuelva a generar nuevos conocimientos individuales y colectivos. A continuación se muestran gráficamente los procesos de conversión de unos tipos de conocimiento en otros a través de determinadas fases:

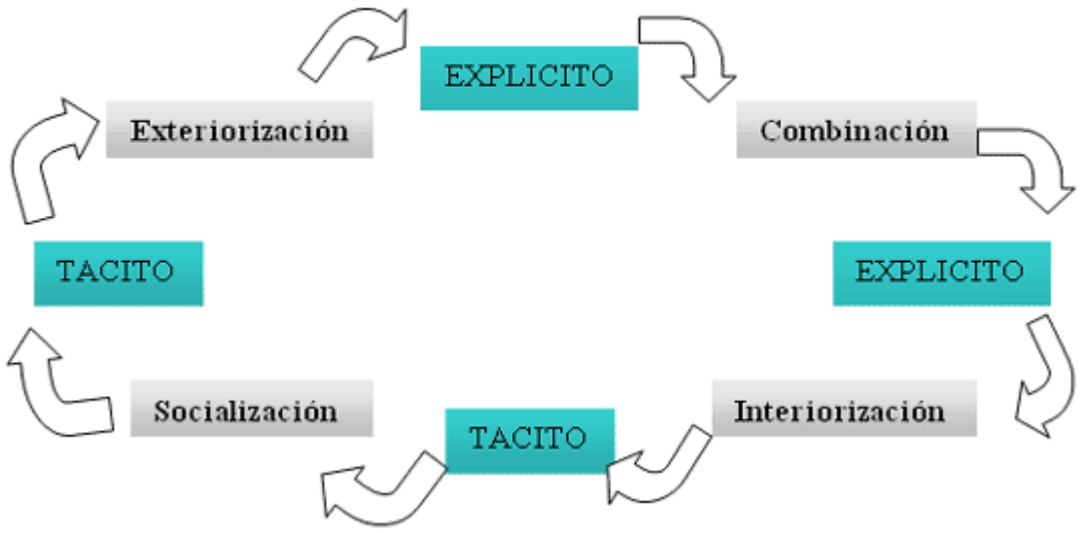


Figura 1. 5 Espiral de conocimiento

- **La Socialización**, es el proceso de adquirir conocimiento tácito a través de compartir experiencias por medio de exposiciones orales, documentos, manuales y tradiciones y que añade el conocimiento novedoso a la base colectiva que posee la organización.

- **La Exteriorización**, es el proceso de convertir conocimiento tácito en conceptos explícitos que supone hacer tangible mediante el uso de metáforas conocimiento de por sí difícil de comunicar, integrándolo en la cultura de la organización; es la actividad esencial en la creación del conocimiento.
- **La Combinación**, es el proceso de crear conocimiento explícito al reunir conocimiento explícito proveniente de cierto número de fuentes, mediante el intercambio de conversaciones telefónicas, reuniones, correos, etc., y se puede categorizar, confrontar y clasificar para formar bases de datos para producir conocimiento explícito.
- **La Interiorización**, es un proceso de incorporación de conocimiento explícito en conocimiento tácito, que analiza las experiencias adquiridas en la puesta en práctica de los nuevos conocimientos y que se incorpora en las bases de conocimiento tácito de los miembros de la organización en la forma de modelos mentales compartidos o prácticas de trabajo." (Acuña, 2002)

Por ello, la retención del conocimiento pasará por:

- El traspaso del conocimiento explícito a las personas.
- El traspaso del conocimiento tácito de una persona a otras personas.
- La transformación del conocimiento tácito en conocimiento explícito.

1.5 Software Educativo y Multimedia

Sánchez J. en su libro "Construyendo y Aprendiendo con el Computador", define el concepto genérico de Software Educativo como cualquier programa computacional cuyas características estructurales y funcionales sirvan de apoyo al proceso de enseñar, aprender y administrar. (Sánchez, 1999)

La literatura define el concepto genérico de Software Educativo como cualquier 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. (Dellamea, 2006)

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.

1.5.1 Características, uso y funciones del software educativo

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. (Rodríguez, 2000)

Los softwares educativos pueden tratar diferentes materias de diversas maneras y ofrecer un entorno de trabajo más o menos sensible a las necesidades de los alumnos y más o menos rico en posibilidades de interacción pero todos comparten las siguientes características:

- Permiten la interactividad con los estudiantes, retroalimentándolos y evaluando lo aprendido.
- Facilitan las representaciones animadas.
- Inciden en el desarrollo de las habilidades a través de la ejercitación.
- Permiten simular procesos complejos.
- Reducen el tiempo del que se dispone para impartir gran cantidad de conocimientos facilitando un trabajo diferenciado, introduciendo al estudiante en el trabajo con los medios computarizados.
- Facilitan el trabajo independiente y a la vez un tratamiento individual de las diferencias.
- Permiten al usuario (estudiante) introducirse en las técnicas más avanzadas.

El uso del software educativo en el proceso de enseñanza - aprendizaje puede ser:

Por parte del alumno. Se evidencia cuando el estudiante opera directamente el software educativo, pero en este caso es de vital importancia la acción dirigida por el profesor.

Por parte del profesor. Se manifiesta cuando el profesor opera directamente con el software y el estudiante actúa como receptor del sistema de información. La generalidad plantea que este no es el caso más productivo para el aprendizaje. (Rodríguez, 2000)

La funcionalidad, ventajas e inconvenientes que pueda traer su uso, serán el resultado de las características del material, de su adecuación al contexto educativo al que se aplica y de la manera en que el profesor organice su utilización.

- Función informativa: La mayoría de los programas, a través de sus actividades, presentan unos contenidos que proporcionan una información estructuradora de la realidad a los estudiantes. Como todos los medios didácticos, estos materiales representan la realidad y la ordenan.
- Función instructiva: Todos los programas educativos orientan y regulan el aprendizaje de los estudiantes ya que, explícita o implícitamente, promueven determinadas actuaciones de los mismos encaminadas a facilitar el logro de unos objetivos educativos específicos. Además, condicionan el tipo de aprendizaje que se realiza pues, por ejemplo, pueden disponer un tratamiento global de la información (propio de los medios audiovisuales) o a un tratamiento secuencial (propio de los textos escritos).

- Función motivadora: Generalmente los estudiantes se sienten atraídos e interesados por todo el software educativo, ya que los programas suelen incluir elementos para captar la atención de los alumnos, mantener su interés y, cuando sea necesario, focalizarlo hacia los aspectos más importantes de las actividades.
- Función evaluadora: La interactividad propia de estos materiales, que les permite responder inmediatamente a las respuestas y acciones de los estudiantes, les hace especialmente adecuados para evaluar el trabajo que se va realizando con ellos. Esta evaluación puede ser de dos tipos:
  - ✓ *Implícita:* cuando el estudiante detecta sus errores, se evalúa, a partir de las respuestas que le da el ordenador.
  - ✓ *Explícita:* cuando el programa presenta informes valorando la actuación del alumno. Este tipo de evaluación sólo la realizan los programas que disponen de módulos específicos de evaluación.

- Función investigadora: Los programas no directivos, especialmente las bases de datos, simuladores y programas constructores, ofrecen a los estudiantes interesantes entornos donde investigar: buscar determinadas informaciones, cambiar los valores de las variables de un sistema, etc.
- Función Expresiva: Dado que los ordenadores son unas máquinas capaces de procesar los símbolos mediante los cuales las personas representamos nuestros conocimientos y nos comunicamos, sus posibilidades como instrumento expresivo son muy amplias.
- Función lúdica: Trabajar con los ordenadores realizando actividades educativas, es una labor que a menudo tiene unas connotaciones lúdicas y festivas para los estudiantes.
- Función innovadora. Aunque no siempre sus planteamientos pedagógicos resulten innovadores, los programas educativos se pueden considerar materiales didácticos con esta función, ya que utilizan una tecnología recientemente incorporada a los centros educativos y, en general, suelen permitir muy diversas formas de uso. Esta versatilidad abre amplias posibilidades de experimentación didáctica e innovación educativa en el aula.

1.5.2 Multimedia. Clasificación

El concepto de multimedia es tan antiguo como la comunicación humana, ya que al expresarnos en una charla normal hablamos (sonido), escribimos (texto), observamos a nuestro interlocutor (video) y accionamos con gestos y movimientos de las manos (animación).

El término multimedia se utiliza para referirse a cualquier objeto o sistema que utiliza múltiples medios de expresión (físicos o digitales) para presentar o comunicar información.
De allí la expresión "multi-medios". Los medios pueden ser variados, desde texto e imágenes, hasta animación, sonido, video, etc. (Diaz, 1994)

Multimedia Interactiva

Se habla de multimedia interactiva cuando el usuario tiene libre control sobre la presentación de los contenidos, acerca de qué es lo que desea ver y cuándo; a diferencia de una presentación lineal, en la que es forzado a visualizar contenido en un orden predeterminado.
Cuanto más alto el grado de interactividad, mayor es la complejidad del producto y, por lo tanto, exige mayor tiempo de desarrollo y tiene un más alto costo. Es por ello que el grado de interactividad a utilizar debe seleccionarse meditadamente de acuerdo a los objetivos del proyecto**.** Esto no quiere decir que cuanto más alto el grado de interactividad mejor el producto, sino que este nivel debe ser adecuado y suficiente. (Multimediam, 2006)
Cuando se habla de multimedia, se hace necesario también definir el concepto de *hipertexto*, el cual ha sido definido como un enfoque para manejar y organizar la información, en el cual los datos se almacenan en una red de nodos conectados por enlaces. Los nodos contienen textos y si contienen además gráficos, imágenes, audio, animaciones y video, así como código ejecutable u otra forma de datos, se les da el nombre de hipermedio, es decir, una generalización de hipertexto. (Bianchini, 2000).
La hipermedia puede considerarse como una forma especial de multimedia interactiva que emplea estructuras de navegación más complejas que aumentan el control del usuario sobre el flujo de la información. (Sabatini, 2008)

Se pueden clasificar según:

- Su sistema de navegación.
- El nivel de control que tiene el profesional.
- Su finalidad y base Teórica.

A continuación se explica cada una de ellas.

1.5.2.1 Clasificación según su sistema de navegación:

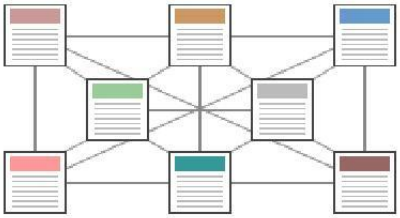
La estructura seguida en una aplicación multimedia es de gran relevancia pues determina el grado de interactividad de la aplicación, por tanto, la selección de un determinado tipo de estructura para la aplicación condicionará el sistema de navegación seguido por el usuario y la posibilidad de una mayor

o menor interacción con la aplicación. No existe una estructura mejor que otra, sino que ésta estará subordinada a la finalidad de la aplicación multimedia.

Los sistemas de navegación más usuales en relación a la estructura de las aplicaciones son:

LINEAL: El usuario sigue un sistema de navegación lineal o secuencial para acceder a los diferentes módulos de la aplicación, de tal modo que únicamente puede seguir un determinado camino o recorrido. Esta estructura es utilizada en gran parte de las aplicaciones multimedia de ejercitación y práctica o en libros multimedia.

RETICULAR: Se utiliza el hipertexto para permitir que el usuario tenga total libertad para seguir diferentes caminos cuando navega por el programa, atendiendo a sus necesidades, deseos, conocimientos, etc. Sería la más adecuada para las aplicaciones orientadas a la consulta de información, por ejemplo para la realización de una enciclopedia electrónica.



JERARQUIZADO: Combina las dos modalidades anteriores. Este sistema es muy utilizado pues combina las ventajas de los dos sistemas anteriores (libertad de selección por parte del usuario y organización de la información atendiendo a su contenido, dificultad, etc.). (Ortí, 2006).

1.5.2.2 Clasificación según el nivel de control que tiene el profesional:

Una de las características más deseables en una aplicación multimedia es su capacidad para poder ser configurada y/o adaptada por el profesional para poder atender las necesidades concretas de los usuarios. Los tipos de software según el menor o mayor nivel de control por parte del profesional son:

- **Programas cerrados:** Los componen los programas informáticos que trabajan sobre un determinado contenido, y el profesional no tiene posibilidad de modificarlo y/o adaptarlo a las características de las personas con las que trabaja. Tienen una estructura secuencial que no puede ser modificada por el usuario.
- **Programas semiabiertos:** Estas aplicaciones permiten que el profesional modifique algunas de las características del programa o tome decisiones sobre el itinerario a seguir. Algunos programas semiabiertos permiten seleccionar diferentes niveles de dificultad en las actividades a realizar. El programa Exler de la escuela de patología del lenguaje, es un ejemplo de este tipo de programa, puesto que permite: seleccionar el tipo de actividades que deseamos realizar, el nivel de dificultad de las actividades y también ajustar la tipografía a las características de los usuarios.
- **Programas abiertos:** Son programas informáticos que, partiendo de un conjunto de posibilidades de actuación, permiten que el profesional fije el contenido concreto a desarrollar, pudiendo adaptarlo a las necesidades de las personas concretas que lo van a utilizar. Un ejemplo de programa abierto es el programa “Clic” que puede ser utilizado por los logopedas para crear ejercicios y actividades orientadas a la intervención de un caso o problema concreto. (Ortí, 2006).

1.5.2.3 Clasificación según su finalidad y base Teórica

Se han desarrollado multitud de aplicaciones multimedia con diferentes objetivos y funciones pedagógicas. Así tenemos: enciclopedias multimedia, cuentos interactivos, juegos educativos, aplicaciones multimedia tutoriales, etc. La finalidad de las aplicaciones multimedia puede ser predominantemente informativa o formativa. (Multimediam, 2006)

Multimedias informativas

- **Libros o cuentos multimedia:** Se parecen a los libros convencionales en formato papel en cuanto a que mantienen una estructura lineal para el acceso a la información, pero en sus contenidos tiene un mayor peso o importancia el uso de diferentes códigos en la presentación de esta información (sonidos, animaciones,...).

- **Enciclopedias y diccionarios multimedia:** Al igual que las enciclopedias y diccionarios en papel son recursos de consulta de información, por lo que su estructura es principalmente reticular para favorecer el rápido acceso a la información. Las enciclopedias y diccionarios multimedia utilizan bases de datos para almacenar la información de consulta de forma estructurada, de modo que el acceso a la misma sea lo más rápido y sencillo.
- **Hipermedias:** Son documentos hipertextuales, con información relacionada a través de enlaces que presentan información multimedia. Su estructura es en mayor o menor grado jerarquizada, utilizando diferentes niveles de información. No obstante, los usuarios tienen gran libertad para moverse dentro de la aplicación atendiendo a sus intereses. (VAUGHAN, 1994)

Multimedias formativas

- **Programas de ejercitación y práctica**: Presentan un conjunto de ejercicios que deben realizarse siguiendo la secuencia predeterminada del programa. Se basan en la teoría de la conducta y utilizan un feedback externo para el refuerzo de las actividades. Han sido muy cuestionados desde la perspectiva pedagógica, aunque tienen un importante desarrollo y uso en actividades que exigen el desarrollo y ejercitación, de destrezas concretas.
- **Tutoriales:** Son semejantes a los programas de ejercitación pero presentan información que debe conocerse o asimilarse previamente a la realización de los ejercicios. En muchos tutoriales se presenta una imagen animada o video que va guiando el proceso de aprendizaje. Siguen los postulados del aprendizaje programado.
- **Simulaciones:** Tienen por objeto la experimentación del usuario con gran variedad de situaciones reales. Básicamente, el programa muestra un escenario o modelo sobre el que el estudiante puede experimentar, bien indicando determinados valores para las variables del modelo, o bien realizando determinadas acciones sobre el mismo, comprobando a continuación los efectos que sus decisiones han tenido sobre el modelo propuesto. De este modo, el usuario toma un papel activo en su proceso de aprendizaje, decidiendo qué hacer y analizando las consecuencias de sus decisiones. Se basan en el aprendizaje por descubrimiento.
- **Talleres creativos:** Promueven la construcción y/o realización de nuevos entornos creativos a través del uso de elementos simples, por ejemplo juegos de construcción, taller de dibujo, etc.
- **Resolución de problemas:** Estas aplicaciones multimedia tienen por objeto desarrollar habilidades y destrezas de nivel superior, basándose en la teoría constructivista. Para ello, se plantean problemas contextualizados en situaciones reales los cuales requieren el desarrollo de destrezas tales como comprensión, análisis, síntesis, etc. Para ello, se proporcionan materiales

y recursos para su solución, junto a materiales adicionales para profundizar en el tema planteado.

- **Caza del tesoro:** Una caza del tesoro es un documento hipermedia (página web) en la que se presentan una serie de preguntas sobre un determinado tema, junto a una lista de direcciones web en las que se pueden buscar las respuestas. Como punto final, se incluye una pregunta "la gran pregunta", que los alumnos deben responder a partir de la comprensión e integración de lo aprendido durante la búsqueda y resolución de las preguntas, pues no es posible encontrar la respuesta de forma directa. "*Las cazas del tesoro son estrategias útiles para adquirir información sobre un tema determinado y practicar habilidades y procedimientos relacionados con las tecnologías de la información y la comunicación en general y con el acceso a la información a través de la Internet en particular*".
- **WebQuest:** La metodología WebQuest desarrollada por Bernie Dodge y Tom March, es una actividad orientada a la investigación, en la que parte o toda la información con la que interaccionan los alumnos, proviene de Internet. WebQuest usa el mundo real y tareas auténticas para motivar a los alumnos. Están compuestas por seis partes esenciales: Introducción, Tarea, Proceso, Recursos, Evaluación y Conclusión. Su estructura es constructivista y por tanto fuerza a los alumnos a transformar la información y entenderla. Sus estrategias de aprendizaje cooperativo ayudan a los estudiantes a desarrollar habilidades y a contribuir al producto final del grupo.
- **Wiki:** Es una aplicación orientada al aprendizaje colaborativo. Básicamente consiste en la elaboración de documentos multimedia de forma colaborativa. Los documentos (páginas wiki) se alojan en un servidor y pueden ser escritos por un conjunto de personas a través de un navegador, utilizando una notación sencilla para dar formato, crear enlaces, etc. Cuando alguien edita una página wiki, sus cambios aparecen inmediatamente en la web, sin pasar por ningún tipo de revisión previa. (Díaz, 1994)

1.6 Tecnología a utilizar

En la siguiente sección se analizará la metodología, lenguaje de modelado, lenguaje de programación y herramienta a utilizar para desarrollar la multimedia.

1.6.1 Metodologías a considerar para el desarrollo del software

Se analizaron varias metodologías existentes, para elegir la más adecuada a partir de las ventajas que presenta cada una de ellas con respecto a las demás.

Este análisis se realizará de acuerdo a los siguientes criterios tomados:

- Tamaños de los equipos.
- Nivel de programación y flexibilidad.
- Obtención de requisitos.
- Carga de trabajo.
- Conocimiento del autor.
- Antecedentes en la colección de multimedia.

En la siguiente tabla se muestran las principales características de cada una de estas metodologías teniendo en cuenta los anteriores puntos.

Tabla 1. 3 Comparación entre las diferentes metodologías

	RUP	XP	MFS	FDD
1	Proyectos grandes con roles designados.	Proyectos cortos, rotables.	Proyectos cortos, corto plazo.	Proyectos medianos. A mayor necesidad de código, mayor organización.
2	Tiene 4 fases que se desarrollan mediante un ciclo de iteraciones, cada ciclo exige el uso de artefactos.	Programación rápida o de extrema, funcionalidades retroalimentación continua.	Un poco rígida.	Es flexible, incluye un monitoreo permanente.
3	Se basa en casos de uso, donde se definen los requisitos.	Se basa en historias de casos de uso, solo define detalles técnicos.	Durante la fase de planificación se preparan las especificaciones funcionales.	No hace énfasis en la obtención de requerimientos.
4	Se basa en la documentación. Presenta un plan de desarrollo donde se reconocen los errores y se pueden corregir.	Proceso ligero, no se asignan roles organizativos al equipo.	Tiene definido un modelo de roles con varias personas, ningún rol es omitido.	Se genera una documentación, depende del jefe que es el máximo responsable.
5	Muy conocido, es el usado en toda la trayectoria.	Poco conocido.	Poco conocido.	Desconocido.
6	Si	No	No	No

RUP: Rational Unified Process.

XP: Extreme Programming.

MFS: Microsoft Solution Framework.

FDD: Feature Driven Development.

Luego de hacer estas comparaciones, RUP es la metodología más idónea. Además de las características anteriormente expuestas en la tabla, se puede decir que es un proceso dirigido por casos de uso, éste avanza a través de una serie de flujos de trabajo (requisitos, análisis, diseño, implementación, prueba) que parten de los casos de uso; está centrado en la arquitectura y es iterativo e incremental, también cubre el ciclo de vida de desarrollo de un proyecto y toma en cuenta las mejores prácticas a utilizar en el modelo de desarrollo de software. (Evaristo, 2006).

A continuación se muestran estas prácticas.

- El desarrollo de software de una forma iterativa.
- El manejo de los requerimientos.
- La utilización de la arquitectura basada en componentes.
- El modelado del software de forma visual.
- La verificación de la calidad del software.
- El control de los cambios.

1.6.2 Lenguajes de modelado

UML es un lenguaje de modelado estándar para escribir planos de software que se utiliza para visualizar, especificar, construir y documentar los artefactos de un mismo sistema que involucra una gran cantidad de software.

Los elementos de UML se clasifican en estructurales (clases, interfaces, colaboraciones, casos de uso, clases activas, componentes y nodos), de comportamiento (interacciones y máquinas de estado), de agrupación (paquetes) y de anotación (notas). A su vez, hay cuatro tipos de relaciones: de dependencia, de asociación, de agrupación y de realización. Para construir un plano de software que tenga sentido, lo que se hace es combinar los elementos estructurales con sus respectivas relaciones, según sea el caso, obteniendo como resultado uno de los nueve diagramas que existen en UML: De clases, De objetos, de casos de uso, de secuencia, de colaboración, de estados, de actividades, de componentes y de despliegue. (Pérez, 2006)

Lenguaje Orientado a Objetos para el Modelado de Aplicaciones Multimedia (OMMMA-L)

En medio de una búsqueda para una modelación adecuada, el Lenguaje de Modelado Orientado a objetos de Aplicaciones Multimedia (OMMMA - L) se lanza como una propuesta de extensión de UML para la integración de especificaciones de sistemas multimedia basados en el paradigma orientado a objetos, y MVC (Modelo Vista Controlador) para la interfaz de usuario, siendo este un patrón de diseño

de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos de forma que las modificaciones al componente de la vista pueden ser hechas con un mínimo impacto en el componente del modelo de datos.

OMMMA-L está sustentado en cuatro vistas fundamentales, donde cada una se asocia a un tipo de diagrama en particular. Estas vistas son:

- **Vista Lógica:** modelada a través del diagrama de clases de OMMMA-L, extendido del diagrama de clases de UML, utilizando las mismas notaciones, pero incorporando las clases correspondientes a las medias: media continua y media discreta, generalizadas en una clase medias. Divide en dos áreas dicho diagrama: una para la jerarquía de los tipos de media y otra para la modelación de la estructura lógica del dominio de la aplicación.
- **Vista de Presentación espacial:** modelada a través de los diagramas de presentación de OMMMA-L, los cuales son de nueva aparición en la extensión de UML, dado que este último no contiene un diagrama apropiado para esta tarea. Estos diagrama tienen el propósito de declarar las interfaces de usuario con un conjunto de estructuras delimitadas en tamaño y área, dividiéndose en objetos de visualización (texto, gráfico, video, animación) e interacción (scrolls, barras de menú, botones, campos de entrada y salida, hipertextos con hipervínculos). Estos diagramas de presentación pueden ser divididos en capas virtuales de presentación donde en cada uno de ellas sólo se haga referencia al desarrollo de una clase específica de componentes (por ejemplo, una vista para los objetos de visualización y otra para los de interacción, u otro tipo de división para la representación de los intereses de los desarrolladores).
- **Vista de Comportamiento temporal predefinido:** modelada por el diagrama de secuencia de OMMMA-L, extendido a partir del diagrama de secuencia de UML. El Diagrama de secuencia modela una secuencia de una presentación predefinida dentro de una escena, donde todos los objetos dentro de un diagrama se relacionan al mismo eje del tiempo. En este diagrama se hace un refinamiento del eje del tiempo con la introducción de marcas de tiempo a través de diferentes tipos de intervalos; marcas de inicio y fin de ejecución que permite soportar su reusabilidad; marcas de activación y desactivación de demoras en objetos de tipo media, posibilitando la modelación de las tolerancias de la variación de las restricciones de sincronización para los objetos media; activación compuesta de objetos media para la agrupación de objetos concurrentemente activos.
- **Vista de Control Interactivo:** modelado a través del diagrama de estado, extendido a partir del diagrama de estado de UML, sintácticamente igual a este último, mas con la diferencia semántica de unir los controles interactivos y predefinidos, no interrumpidos de los objetos, las acciones internas de estados simples tienen que llevar nombres de diagrama de secuencia en

vez de diagramas de estado empotrados; queriendo esto decir que el comportamiento especificado por el diagrama de secuencia se provoca automáticamente cuando se entra al estado correspondiente donde se hace referencia. (Pérez, 2006)

1.6.3 Lenguajes de programación

Como lenguajes de programación se utilizan:

XML: Es un Lenguaje de Etiquetado Extensible muy simple, pero estricto, el cual juega un papel fundamental en el intercambio de una gran variedad de datos. Es un lenguaje muy similar a HTML, pero su función principal es describir datos y no mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones. Las tecnologías XML son un conjunto de módulos que ofrecen servicios útiles a las demandas más frecuentes por parte de los usuarios. XML sirve para estructurar, almacenar e intercambiar información. (W3C 2008).

ActionScript: Es un lenguaje orientado a objetos que permite ampliar las funcionalidades que Flash ofrece en sus paneles de diseño y además permitir la creación de películas o animaciones con altísimo contenido interactivo. Provee a Flash de un lenguaje que permite al diseñador o desarrollador añadir nuevos efectos o incluso construir la interfaz de usuario de una aplicación compleja puesto que está basado en el estándar ECMAScript. La versión 3.0 de ActionScript ha marcado un cambio significativo en este lenguaje, puesto que en esta versión prácticamente se ha decidido prescindir de los prototipos y se lo ha encaminado a ser un lenguaje orientado a objetos solamente a través de clases. También se han hecho grandes cambios en cuanto a la sintaxis del lenguaje. (AulaClic, 2004)

1.6.4 Herramientas

Se realizó un estudio de algunas herramientas que se utilizan en la creación de multimedias tales como: Authorware, Director, Scala multimedia, ToolBook y Flash 8. Después del análisis realizado se decidió optar por Flash 8 pues además de las ventajas que posee con respecto a las demás, es la seleccionada por el cliente y la Multimedia Interactiva para la enseñanza del Diseño y la Construcción de Software forma parte de un paquete de productos multimedia que se viene desarrollando desde años anteriores. A continuación se describen algunas de las principales características de cada una de ellas.

Director: Es una herramienta ideal para la creación de presentaciones. Tanto es así que en Director se trabaja sobre una línea de tiempo, en lugar de trabajar sobre pantallas estáticas. Es tal el dinamismo de Director que cuando se desea crear una pantalla estática, lo que se hace es programar que cuando se salga del frame que contiene la pantalla, vuelva a entrar en el mismo.

La nueva versión de Director incluye numerosas mejoras, pero la mayoría se enfocan en el entorno de trabajo y el Internet. (AulaClic, 2006)

ToolBook: Es una herramienta que principalmente está destinada a la creación de CBTs. Para ello hace uso de la metáfora de un libro y utiliza un lenguaje de programación propio: OpenScript. Las principales mejoras incluidas en esta nueva versión tienen que ver con el mundo de Internet, aunque también son muchas las mejoras incluidas en el propio entorno de trabajo. Nada más comenzar se proporcionan las posibilidades típicas: crear un libro nuevo, abrir uno ya creado o crear un libro utilizando un asistente. Esta última es una de las novedades de la versión 6.0. Con el Book Specialist se va guiando paso a paso al usuario por la creación de una nueva aplicación. Este asistente contempla los múltiples parámetros a tener en cuenta a la hora de realizar un CTB. Por ejemplo, se puede interactuar con parámetros que controlan el aspecto y el comportamiento de la aplicación, tales como la composición de la página, las puntuaciones y el método de distribución, entre otros. Una interesante característica de este asistente es que permite ser configurado a gusto propio, de forma que la siguiente vez que se tenga que crear un nuevo proyecto, el asistente trabajará acorde a lo que se haya especificado. (AulaClic, 2006)

Authorware: Es una herramienta que se caracteriza por emplear líneas de flujo para la definición de la estructura y la participación de los distintos contenidos multimedia en la aplicación. Cada línea de flujo contiene una serie de iconos que representan las acciones que se van a llevar a cabo cuando la ejecución de la aplicación llegue al punto correspondiente. Authorware está concebida principalmente como una herramienta para el desarrollo de materiales educativos interactivos. (AulaClic, 2006)

Scala Multimedia: Es un producto principalmente enfocado a la realización de presentaciones espectaculares, compitiendo en cierta medida con Director, pero no con Authorware y ToolBook. A diferencia de Director, Scala Multimedia es un producto que saca el máximo rendimiento a la máquina donde se ejecute. Hay que tener en cuenta que el objetivo perseguido por el producto es conseguir efectos espectaculares, muy parecidos a los que se utilizan en televisión. Para conseguirlo, es imprescindible que se tenga instalado DirectX, pues sin este módulo el producto no funcionará. Al utilizar DirectX y ActiveMovie, todos los formatos de recursos soportados por las tecnologías de Microsoft son soportados por el programa. Por ejemplo, es posible manejar directamente archivos MPEG sin necesidad de tener el hardware de descompresión, aunque si se tiene, se utiliza para optimizar la reproducción. A diferencia de la versión anterior, MM200 ya está completamente integrada en Windows 95 y NT. MM200 soporta todas las profundidades de color partiendo de 8 bits. (AulaClic, 2006)

Flash 8 es el entorno más avanzado en el mercado. Supera las expectativas debido a sus características y las ventajas que ofrece, ya que Flash ofrece de forma explícita una calidad final del

trabajo y es multiplataforma. A continuación una serie de características que fomentan la elección de la herramienta. (AulaClic, 2006)

- Carga dinámica de imágenes (JPEG, PNG, GIF, JPG).
- Ayuda tanto para la programación (ActionScript) como para el diseño de animaciones.
- Incluye componentes ya creados que ayudan a la hora de crear animaciones.
- Sencilla de usar y con muchas herramientas.
- Biblioteca de símbolos.
- Soporte de audio MP3.
- Soporta video.
- Posee un lenguaje de programación propio, muy conocido (ActionScript).

Conclusiones Parciales

En el capítulo que concluye se introducen los temas: Diseño y Construcción de Software, se exponen las definiciones, conceptos, fundamentos de estos temas. Se detallan las relaciones que existen entre estas áreas de conocimiento y a su vez cómo están relacionadas con las demás áreas. Se realizó un estudio del SWEBOK y el SEEK, otro estándar internacional que aborda dichos temas y, tras un análisis de los puntos esenciales que tienen en común, se determinó el contenido a presentar en la multimedia y quedó conformado de la siguiente forma: Diseño de Software presentará los temas:

- Fundamentos y Elementos Claves del Diseño.
- Estructura y Arquitectura.
- Análisis de la Calidad.
- Notaciones del Diseño.
- Estrategias y Métodos.

Por su parte Construcción de Software abordará los temas:

- Fundamentos de la Construcción.
- Gestión de la Construcción.
- Consideraciones Prácticas.

En el presente capítulo se definió además la Gestión del conocimiento, explicando los tipos de conocimiento y el modelo SECI propuesto por Nonaka y Takeuchi, la cual posee la fase de Combinación que es específicamente a la que está enfocada este trabajo. Se realizó la definición de multimedia y multimedia interactiva. Se determinó como herramienta a utilizar: Flash 8.0, como metodología a RUP, y los lenguajes de modelado y programación a utilizar para el desarrollo del producto: XML y ActionScript 2.0.

CAPÍTULO 2: DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

Introducción

Partiendo de lo planteado en el capítulo anterior, en este capítulo se realizará una serie de actividades para dar solución al problema de investigación. Primeramente se hará la selección de la audiencia a la cual estará dirigido el producto, luego se realizará el modelo del dominio correspondiente, especificando los conceptos que lo conformarán, planteándose de esta forma la necesidad de seleccionar los requisitos tanto funcionales (RF) como los no funcionales (RNF) y realizándose finalmente un diagrama de casos de uso (CU) del sistema, dándose una breve descripción de ellos para facilitar su comprensión.

2.1 Identificación de la Audiencia

Este paso es de suma importancia, ya que es necesario saber a quién estará destinado el producto final para poder de esta forma superar sus expectativas y de esta forma lograr la satisfacción del usuario final.

En este caso, esta aplicación está dirigida a los estudiantes y profesores de la Universidad de las Ciencias Informáticas, con el objetivo de fomentar el proceso de gestión del conocimiento e incrementará la calidad del proceso de diseño y construcción de software en los proyectos productivos de la universidad.

2.2 Mapa Conceptual

Un mapa conceptual no es más que una representación de varios conceptos en forma de red que se encuentran interrelacionados entre sí. Cada nodo será un concepto diferente y los enlaces serán las relaciones entre ellos.

El mapa conceptual es un instrumento útil para la organización y la representación visual del conocimiento, cuya elaboración puede ejecutar el docente para mostrar al estudiante cómo se relacionan determinados conceptos o puede ejecutar el alumno, individualmente o en grupo, con el objetivo de alcanzar una mayor comprensión de los conceptos que estudia.

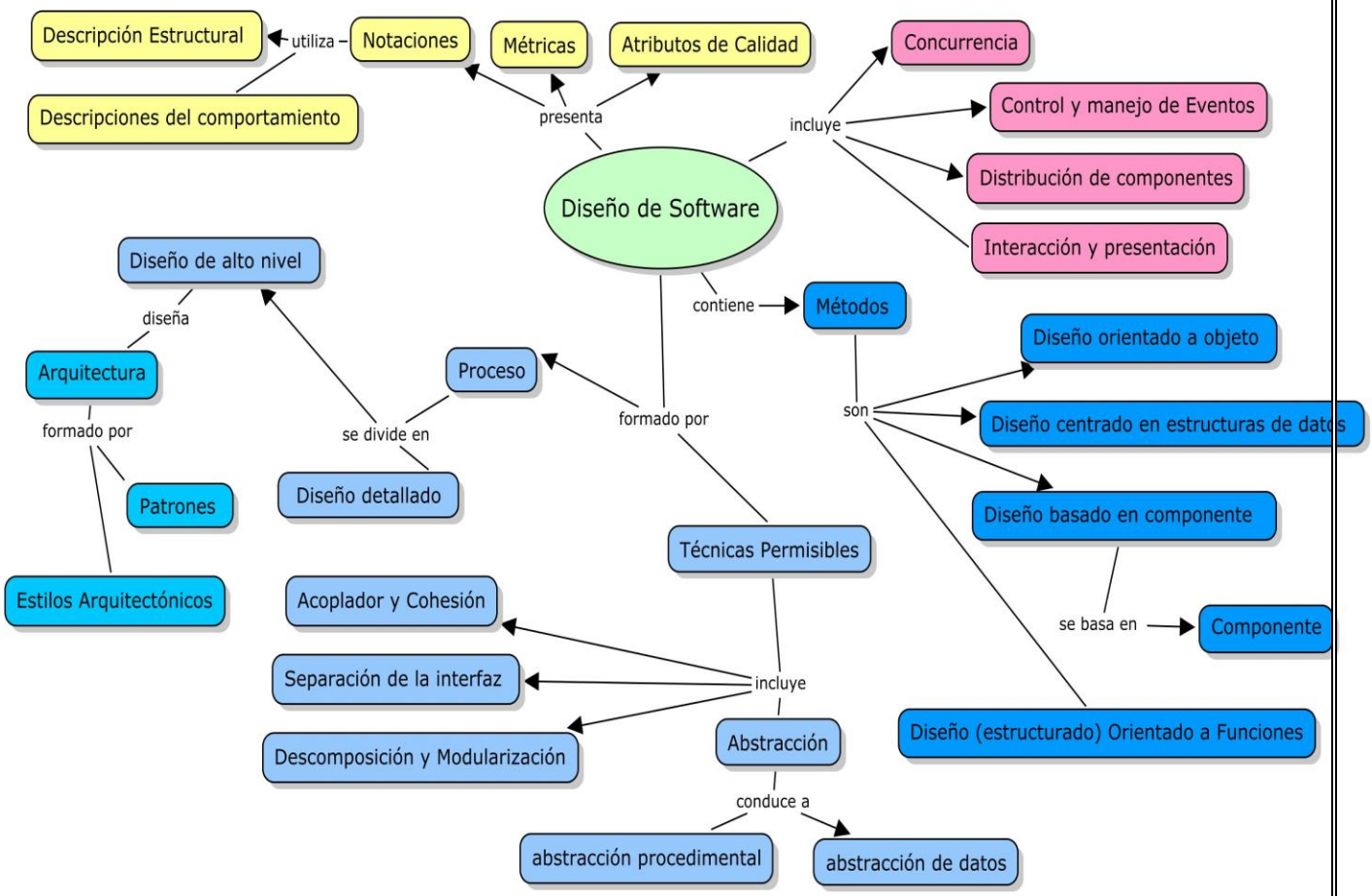


Figura 2. 1 Mapa Conceptual de Diseño de Software

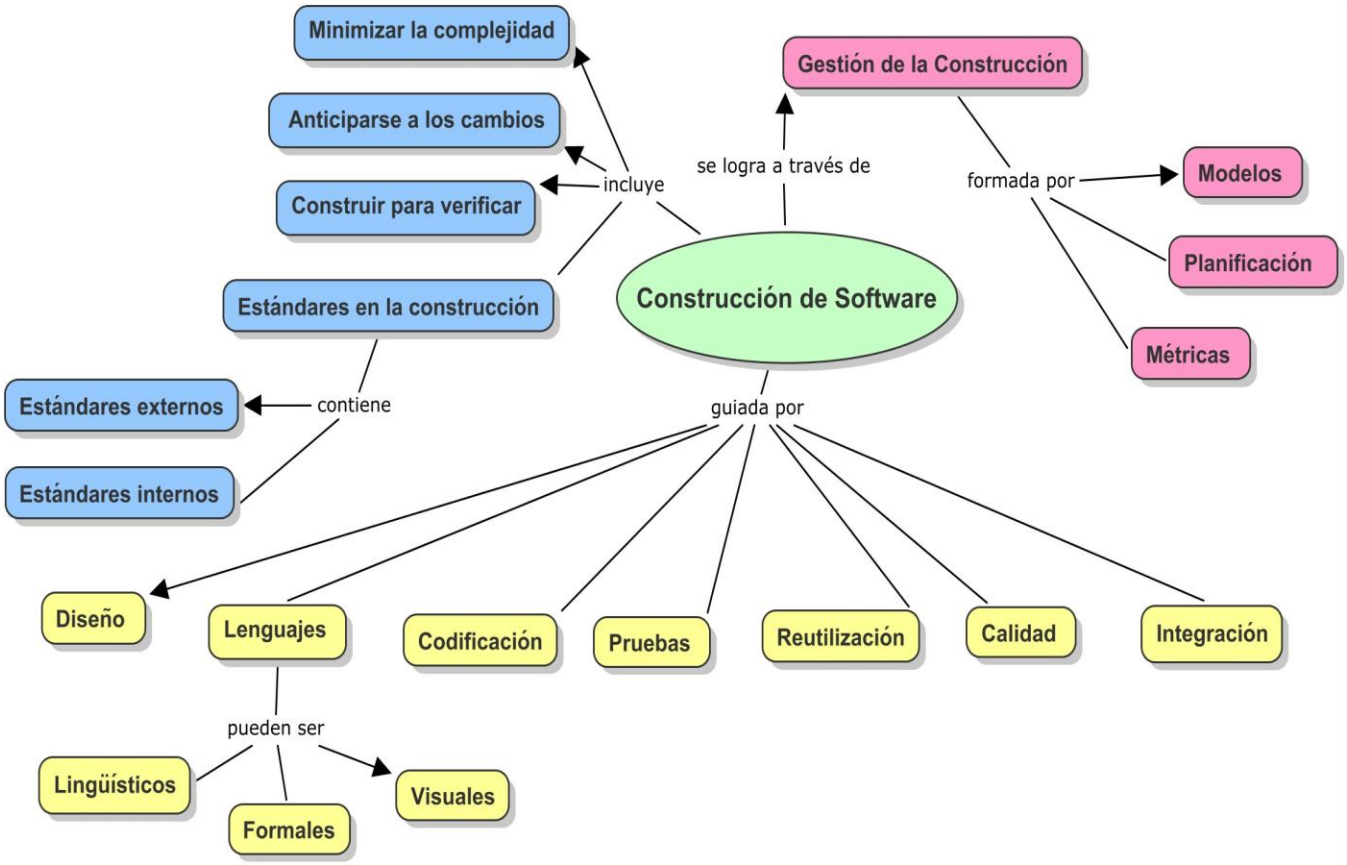


Figura 2. 2 Mapa Conceptual de Construcción de Software

2.3 Descripción del Modelo del Dominio

Debido el bajo nivel de estructuración del negocio, se propone un modelo de dominio que permite, mediante la representación visual de conceptos, un mejor entendimiento del sistema.

La descripción del modelo de dominio se realiza a través de un modelo conceptual que se representa en un diagrama de clases UML. Para realizar el mismo, se listaron todos los conceptos, se representaron en un diagrama y se agregaron las asociaciones para registrar las relaciones entre los mismos.

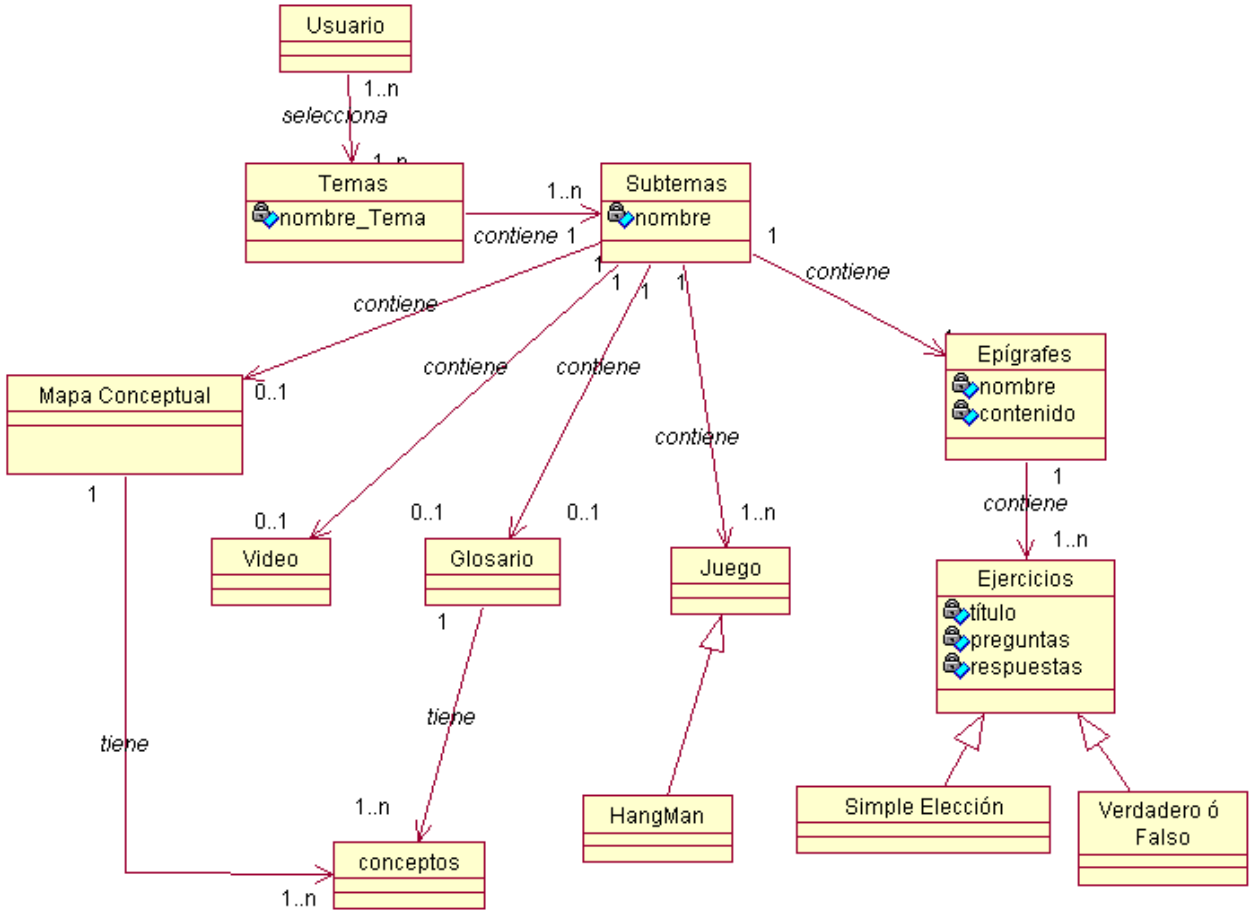


Figura 2. 3 Modelo del Dominio

2.3.1 Conceptos del dominio

En este epígrafe se hace una descripción de los conceptos que fueron utilizados para la elaboración del diagrama del modelo del dominio.

- **Usuarios:** Toda aquella persona que actúa de forma directa con el sistema.
- **Video:** Representa alguna conferencia referente al diseño o la construcción de Software u otro material relacionado con el tema tratado.
- **Tema:** Hace referencia a los temas generales que son tratados en la aplicación. (Diseño y construcción)
- **Mapa Conceptual:** Mapa de conceptos del contenido tratado, mostrando la relación existente entre cada uno de ellos.

- **Juego:** Objetos didácticos que se utilizan para ejercitar los conocimientos que se han adquirido al interactuar con la aplicación.
- **Conceptos:** Son estructuras mediante las cuales se identifican las características esenciales para definir las cosas.
- **Ejercicios:** Representa un conjunto de actividades que se realizan para evaluar los conocimientos adquiridos.
- **Subtemas:** Se denominan subtemas a los diferentes contenidos tratados dentro de otro contenido.
- **Epígrafes:** Hace referencia al contenido existente dentro de cada subtema.
- **Glosario de Términos:** Representa el conjunto de términos que pueden ser de poco conocimiento para el usuario.

2.4 Diagrama de navegación general

La aplicación da inicio con una pantalla de presentación del curso. La misma da paso a la página principal donde el usuario podrá visualizar los temas (diseño y construcción de software), el mapa conceptual, una videoconferencia, un juego de entretenimiento, el glosario de términos o consultar la ayuda. Solamente desde la pantalla de los temas, podrá acceder a los ejercicios de cada tema en específico. El usuario podrá abandonar la aplicación desde cualquier pantalla que desee y una vez realizada esta operación se visualizarán los créditos.

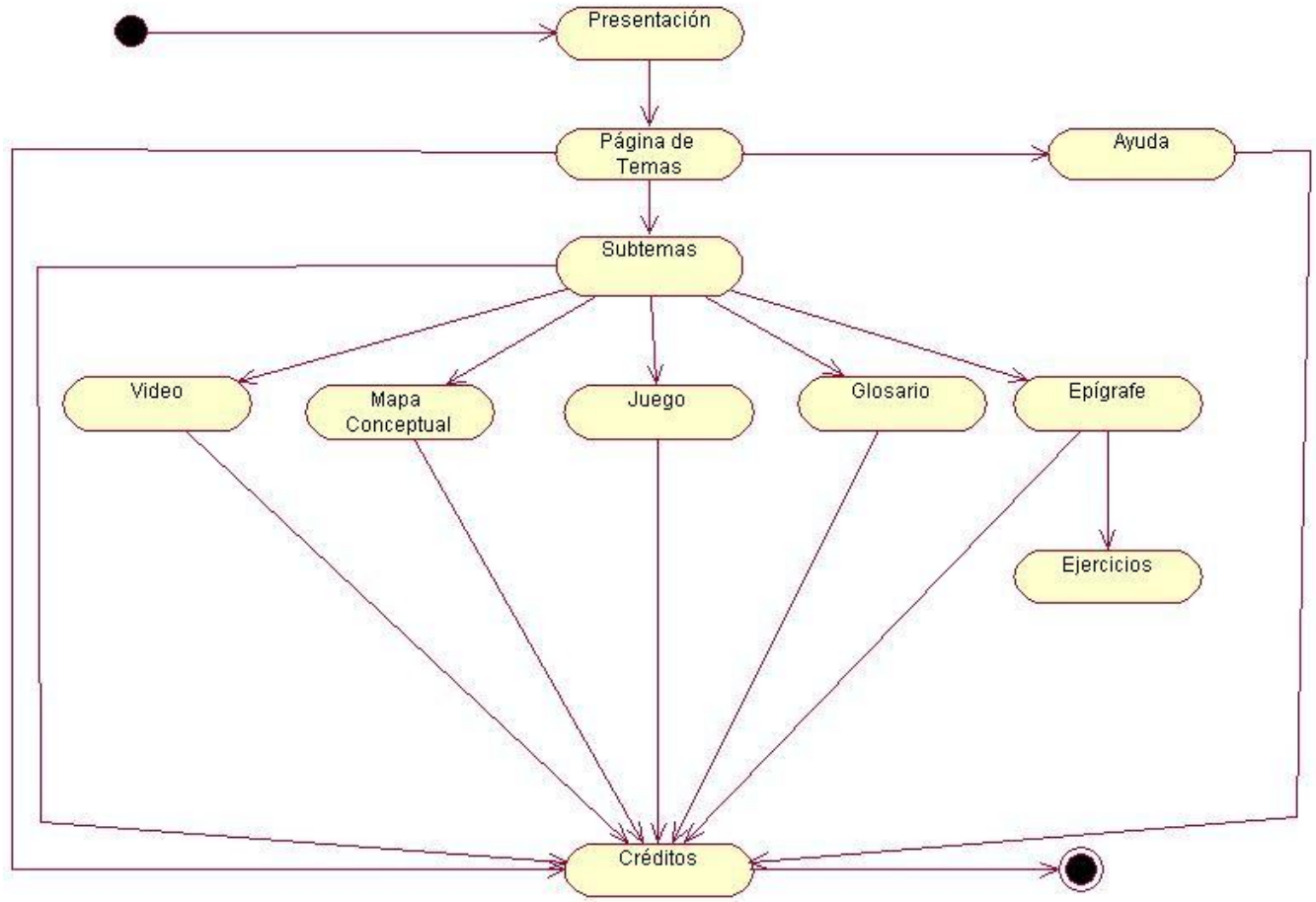


Figura 2. 4 Diagrama de navegación general

2.5 Descripción de las Funcionalidades

A continuación se muestra un listado de los requisitos funcionales y no funcionales a tener en cuenta para desarrollar la multimedia.

2.5.1 Requisitos Funcionales

A continuación se muestran los requisitos funcionales planteados por el cliente.
En la siguiente tabla se exponen los requisitos funcionales planteados por el cliente.

Tabla 2. 1 Requisitos funcionales

Requisitos	Función
R 1	Mostrar una pantalla de presentación que de inicio a la página principal de la Multimedia.
R 2	Reproducir música de fondo de forma predeterminada.
R 3	Permitir al usuario (pausar y reanudar) la música de fondo.
R 4	Permitir al usuario navegar por los temas que brinda la aplicación.
R 5	Dar acceso a los usuarios a navegar por los subtemas por los cuales están compuestos los temas de la multimedia.
R 6	Mostrar información contenida dentro de los subtemas de un tema en específico.
R 7	Permitir al usuario seleccionar algún subtema contenido en un tema en específico
R 8	Permitir al usuario utilizar el Scroll del mouse para acceder a toda la información contenida en un tema en específico.
R 9	Permitir al usuario realizar cualquier ejercicio de cualquier tema que se encuentre en la multimedia.
R 10	Mostrar mensajes indicando al usuario si su respuesta fue correcta o incorrecta.
R 11	Permitir que el usuario ejecute cualquier juego que se encuentre disponible en la multimedia.
R 12	Dar acceso al usuario a toda la documentación de los materiales complementarios.
R 13	Permitir al usuario visualizar la videoconferencia.
R 14	Permitir al cliente tener el control del volumen de la videoconferencia.
R 15	Permitir al usuario maximizar/minimizar la videoconferencia.
R 16	Permitir al usuario visualizar el mapa conceptual de la aplicación.
R 17	Mostrar concepto relacionado con el nodo seleccionado en el mapa conceptual.
R 18	Permitir al usuario visualizar el glosario de términos.
R 19	Mostrar los términos del glosario ordenados alfabéticamente.
R 20	Permitir al usuario regresar a la pantalla principal desde cualquier pantalla en que se encuentre.
R 21	Mostrar al usuario un manual de ayuda en la que se dé una breve explicación de todas las pantallas y botones que se encuentra en la multimedia.
R 22	Permitir al usuario salir de la aplicación desde cualquier lugar de la multimedia en que se encuentre.
R 23	Mostrar los créditos de la aplicación al salir de la misma.

2.5.2 Requisitos no Funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como características que hacen al producto atractivo, usable, rápido y confiable.

2.5.2.1 Requisitos de Apariencia

- La aplicación debe utilizar como idioma principal el español excepto aquellas paradas técnicas que no puedan ser traducidas.
- Utilizar botones que expresen su función, ya sea que se intuya o expresados con texto.
- La opción salir de la aplicación debe estar disponible desde cualquier parte de la aplicación, la cual con solo hacer solo un clic sobre ella se saldrá de la misma.

2.5.2.2 Requisitos de Usabilidad

- Los usuarios que interactúen con la aplicación deben tener conocimiento previo del manejo de la computadora así como del manejo de sistemas operativos visuales.

3.5.2.3 Requisitos de Portabilidad

A continuación se muestran los sistemas operativos sobre los cuales la aplicación podrá ser ejecutada, así como los navegadores en los cuales podrá ser visualizada.

Tabla 2. 2 Requisitos de Portabilidad

Windows	
Plataforma	**Navegador**
Microsoft® Windows® Vista	Microsoft Internet Explorer 7, Firefox 2.0, AOL 9, Safari 3.x o superior.
Microsoft Windows XP	Microsoft Internet Explorer 6.0 o superior, Firefox 1.x, Firefox 2.x, Mozilla 1.x o superior, Netscape 7.x o superior, AOL 9, Opera 7.11 o superior, Safari 3.x o superior.
Windows Server® 2003	Microsoft Internet Explorer 6.0 o superior, Firefox 1.x, Firefox 2.x
Windows 2000	Microsoft Internet Explorer 5.x, Firefox 1.x, Firefox 2.x, Mozilla 1.x, Netscape 7.x o superior, AOL 9, Opera 7.11 o superior.
Windows Me	Microsoft Internet Explorer 5.5, Firefox 1.x, Mozilla 1.x, Netscape 7.x o posterior, AOL 9, Opera 7.11 o superior.
Windows 98	Microsoft Internet Explorer 6.0 o superior, Firefox 1.x, Mozilla 1.x, Netscape 7.x o superior, Opera 7.11 o superior.
Linux	
Plataforma Navegador Red Hat® Enterprise Linux® (RHEL) 3 actualización 8, RHEL 4 actualización 4 (AS/ES/WS)	Firefox 1.5.0.7 o superior; Mozilla 1.7.x o superior; SeaMonkey 1.0.5 o superior.
Novell SUSE 9.x o 10.1	Firefox 1.5.0.7 o superior; Mozilla 1.7.x o superior; SeaMonkey 1.0.5 o superior.

2.5.2.4 Requisitos de Hardware

Para la ejecución de la aplicación se debe tener como requisitos mínimos: Procesador Intel Pentium ||| de 800 MHz (o equivalente) y versiones más avanzadas y de 128 MB de RAM. Pantalla de 16 bits de 1024 x 768 (se recomienda de 32 bits) y 710 MB de espacio disponibles en el disco duro, además la computadora donde se ejecute la multimedia deberá tener tarjeta de video, tarjeta de sonido y demás aditamentos para la reproducción de sonido.

2.5.2.5 Requisitos de Diseño e Implementación

- La herramienta a usar para el diseño grafico de la aplicación es Flash 8.
- Para la carga de los contenidos que formarán parte de la multimedia, se hará uso del lenguaje de programación Action Script en su versión 2.0 o 3.0.

2.6 Modelos de Casos de Uso del Sistema

El modelado de casos de uso es una de las técnicas que se utilizan para modelar los requisitos del sistema. Los casos de uso son fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores, permitiendo modelar el funcionamiento del mismo de la forma que el cliente desee. Utilizando UML, se pueden representar las funcionalidades con las que debe cumplir el producto, o sea, los requisitos funcionales y los actores que van a interactuar con él, mediante un diagrama de casos de uso.

2.6.1 Determinación y justificación de los actores del sistema

Tabla 2. 3 Justificación de los actores del sistema

Actor	Justificación
usuario	Persona que va a entrar a la aplicación a consultar información y aprender sobre los temas Diseño y Construcción de Software.

2.6.2 Diagrama de Caso de Uso

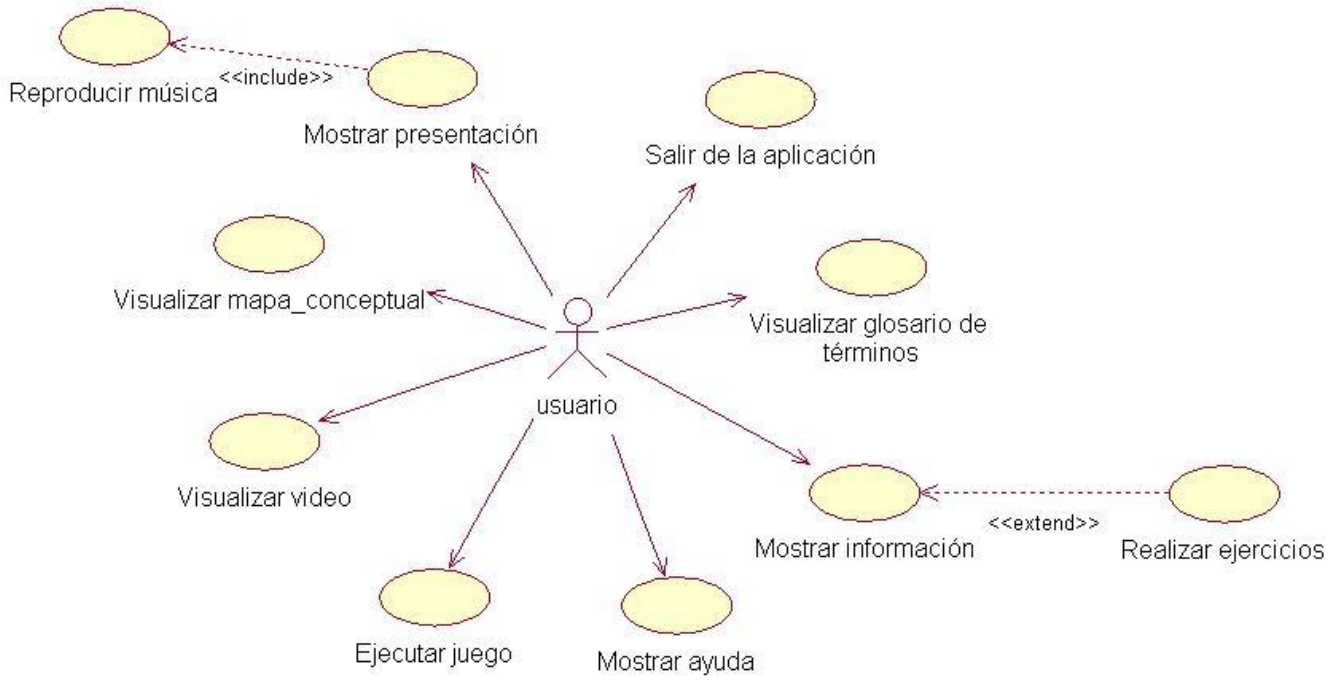


Figura 2. 5 Diagrama de Caso de Uso del Sistema

2.6.3 Breves explicaciones de los casos de Uso del sistema

Tabla 2. 4 Caso de Uso Mostrar Presentación

Caso de Uso:	**Mostrar presentación**
Actor:	usuario
Propósito:	Mostrar la presentación de la multimedia.
Resumen:	Al ejecutarse la aplicación, se muestra la presentación del producto, en la cual se reproduce un sonido de fondo de forma automática. Al concluir la presentación, pasa de forma automática al menú principal de la misma.
Referencias:	R1,R2,R3

Tabla 2. 5 Caso de Uso Mostrar Información

Caso de Uso:	**Mostrar información**
Actor:	usuario
Propósito:	Mostrar toda la información contenida en la multimedia, (textos, imágenes, conceptos pertenecientes al mapa conceptual y ejercicios).
Resumen:	El sistema muestra la información para que el usuario seleccione cuál información desea ver. La aplicación contiene varios subtemas dentro de cada uno de los temas, conceptos pertenecientes al mapa conceptual, ejercicios, etc. Para poder visualizar la información que desea, tendrá que hacer clic sobre el tema y subtema específico contenido dentro del tema que contiene la información deseada y el sistema se encargará de obtener y mostrar la información.
Referencias:	R4, R5, R6, R7, R8, R9, R13, R22

Tabla 2. 6 Caso de Uso Mostrar Video

Caso de Uso:	**Visualizar video**
Actor:	usuario
Propósito:	Mostrar la videoconferencia contenida en la multimedia.
Resumen:	El caso de uso se inicia cuando el usuario decide visualizar la videoconferencia contenida en la aplicación. El usuario se dirige al icono que da la opción de visualizar la misma y con sólo hacer clic, se muestra la pantalla en la cual se verá dicho video y desde donde podrá realizar diferentes actividades como parar, pausar, reproducir, subir o bajar el volumen del mismo.
Referencias:	R15, R16, R17, R22

Tabla 2. 7 Caso de Uso Realizar Ejercicios

Caso de Uso:	**Realizar ejercicios**
Actor:	usuario
Propósito:	Mostrar y ejecutar los ejercicios de los temas estudiados en el curso.
Resumen:	El caso de uso se inicia cuando el usuario, después de haber consultado un determinado tema, decide realizar ejercicios del tema estudiado haciendo clic sobre el último botón del menú desplegable que se llama Ejercicios. Seguidamente se muestra en pantalla con un cuestionario que el usuario deberá responder haciendo clic sobre la respuesta que considere correcta. Para este cuestionario se utilizarán preguntas de verdadero y falso o de selección simple. De manera automática, la aplicación irá pasando de una pregunta a otra hasta llegar al final del cuestionario. Si el usuario no responde correctamente, se le mostraran las respuestas correctas.
Referencias:	R10, R 11, R22

Tabla 2. 8 Caso de Uso Ejecutar Juego

Caso de Uso:	Ejecutar juego
Actor:	usuario
Propósito:	Ejecutar el juego disponible en la aplicación.
Resumen:	El caso de uso comienza cuando el actor selecciona el icono de juego. En ese momento aparecerá el juego del ahorcado, el cual consiste en descubrir una palabra relacionada con el tema que se trata en el curso. El juego irá guardando en la memoria de la máquina un historial que será con valores positivos o negativos, en dependencia de la cantidad de aciertos o fallas con respecto a las palabras a adivinar.
Referencias:	R12, R22

Tabla 2. 9 Caso de Uso Visualizar Mapas

Caso de Uso:	Visualizar mapa conceptual
Actor:	usuario
Propósito:	Mostrar el mapa conceptual que contiene los principales conceptos del curso de estudio.
Resumen:	El caso de uso comienza cuando el actor hace clic sobre el icono Mapa. Automáticamente se carga una pantalla que contiene el mapa conceptual. El usuario se puede desplazar por el mismo, e ir viendo cada uno de los conceptos que el mismo muestra a través de nodos interrelacionados. Haciendo clic sobre un nodo, podrá visualizar en una ventana flotante con el concepto del mismo.
Referencias:	R18, R19, R22

Tabla 2. 10 Caso de Uso Visualizar Ayuda

Caso de Uso:	Mostrar ayuda
Actor:	usuario
Propósito:	Mostrar una guía al usuario que ayude a la navegación por la aplicación
Resumen:	El caso de uso comienza cuando el actor decide consultar la ayuda, porque necesita una mejor orientación por los contenidos de la aplicación. Dando clic sobre el botón de ayuda, aparece una guía de navegación al usuario.
Referencias:	R23, R22

Tabla 2. 11 Caso de Uso Visualizar Glosario de Términos

Caso de Uso:	Mostrar glosario
Actor:	usuario
Propósito:	Mostrar un glosario de términos ordenados alfabéticamente que ayuden a la comprensión de algunos de los conceptos de difícil entendimiento al usuario.
Resumen:	El caso de uso comienza cuando el actor, después de haber consultado información referente a un tema determinado, decide consultar el concepto de una determinada palabra que desconoce su significado. Dando clic sobre el botón glosario, aparece un glosario de todos los términos de comprensión más compleja que contiene la aplicación, ordenados alfabéticamente.
Referencias:	R20,R21, R22

Tabla 2. 12 Caso de Uso Salir de la Aplicación

Caso de Uso:	Salir de la aplicación
Actor:	usuario
Propósito:	Permitir salir de la aplicación en cualquier momento deseado y desde cualquier lugar de la multimedia.
Resumen:	El caso de uso inicia cuando el usuario desea salir de la aplicación. En cualquier instante que el usuario lo desea, al hacer clic en el botón de salir, la aplicación le mostrará un mensaje de confirmación y el usuario puede aceptar o cancelar la opción de salir. Una vez confirmada la salida, se mostrarán los créditos automáticamente.
Referencias:	R24, R25

Tabla 2. 13 Caso de Uso Reproducir música

Caso de Uso:	Reproducir música
Actor:	usuario
Propósito:	Reproducir música automática
Resumen:	El caso de uso comienza cuando el usuario ejecuta la multimedia. La aplicación reproduce la música de forma automática y le da la posibilidad al usuario de pausar, reanudar y controlar el volumen de la misma.
Referencias:	R2, R3

Conclusiones Parciales

En el capítulo se definió la audiencia a la cual está dirigida la aplicación y se especifican los contenidos que se mostrarán en la misma, en este caso son dos temas con varios subtemas dentro de los mismos. Queda definido el modelo del dominio en forma de diagrama junto a los conceptos que incluye el mismo. Quedaron también plasmados los requisitos funcionales y no funcionales que debe tener el producto. Finalizando de esta forma el capítulo, se puede proceder al diseño e implementación de la aplicación.

CAPITULO 3 DISEÑO DE LA SOLUCIÓN PROPUESTA

Introducción

En el presente capítulo se definirán las normas y principios de diseño que se tendrán en cuenta en la implementación de la aplicación. Se muestran los diagramas de presentación como parte de OMMMA-L, el diagrama de componentes como parte de la implementación. También se presenta la estructura de los archivos XML que se utilizarán para mostrar el contenido del producto. Se le aplicarán además las pruebas correspondientes a la multimedia para comprobar su correcto funcionamiento y verificar que cumpla con todos los requisitos que se definieron en el capítulo anterior.

3.1 Análisis de la Arquitectura utilizada

El diseño de interfaces es una de las labores más importante dentro del proceso de desarrollo de un software. La calidad de la interfaz de usuario puede ser uno de los motivos que conduzca al éxito o fracaso de un sistema, es por eso que uno de los aspectos más relevantes de la usabilidad de un sistema es la consistencia de su interfaz de usuario.

3.1.1 Principios de Diseño

Con el objetivo de desarrollar la interfaz del sistema, se analizaron aspectos importantes que posibilitan el bienestar de todo usuario. Para ello se tuvo en cuenta la organización, la distribución de la información, el público al que va dirigido y la temática que se aborda. Los usuarios que interactúan con el sistema lo hacen por medio de una interfaz gráfica, por tal razón la misma es amena, sencilla, presenta facilidad de uso, logrando que el usuario se sienta identificado con ella. Las pantallas del sistema contienen la información necesaria para evitar la sobrecarga, además de mantener las opciones principales en el mismo lugar de la interfaz para una mejor interacción y adaptabilidad del usuario con la aplicación. El sistema combina colores oscuros y claros y poco llamativos, pues son los que están a fin con la temática. Tiene como color predeterminado el azul.

3.1.2 Estándares para el diseño de la interfaz de usuario

La interfaz de la aplicación fue elaborada teniendo en cuenta algunos de los estándares comunes para aplicaciones multimedia. La aplicación contiene navegación, ayuda, opción de salir y opción de control del sonido.

Navegación: En primer lugar, aparece un menú principal, el cual permite el libre acceso a los contenidos lineales de forma rápida. En segundo lugar se considera el árbol de contenidos, puesto que al profundizar en las diferentes ramas, mediante diferentes pulsaciones del ratón, permitirá acceder a los diferentes subtemas y, al pulsar sobre uno de ellos, acceder a la página de contenidos del mismo.

Ayuda: El usuario puede consultarla desde cualquier pantalla ya que está disponible siempre. Ésta muestra una explicación del curso y cómo navegar en la multimedia para evitar que el usuario se pierda.

Salir: La opción salir es accesible desde cualquier pantalla, por lo que el usuario puede salir cuando lo desee.

Control del sonido: La salida y entrada del fondo musical puede ser activada y desactivada por el usuario.

Color: Los colores que predominan son el azul y blanco, debido a que el sistema debe usar colores serios y poco llamativos.

3.2 Diagrama de Presentación del modelo de análisis

Este es un nuevo artefacto dentro de la extensión OMMMA – L del lenguaje UML y sirve para describir la parte estática del modelo a través de una descripción intuitiva de la distribución espacial de objetos visuales de la interfaz de usuario.

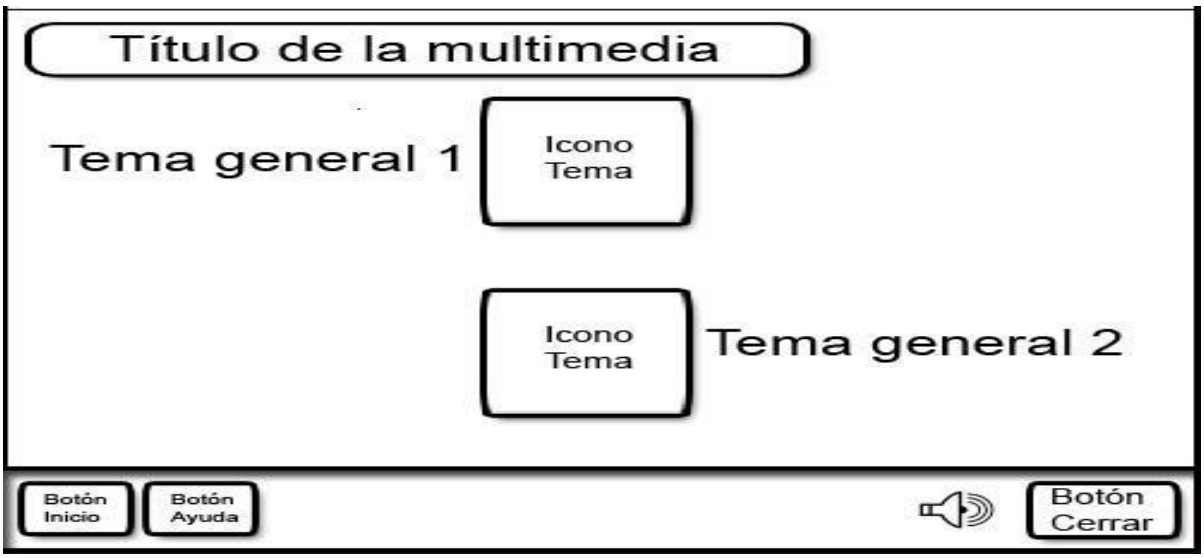


Figura 3. 1 Diagrama de Presentación Pantalla Principal

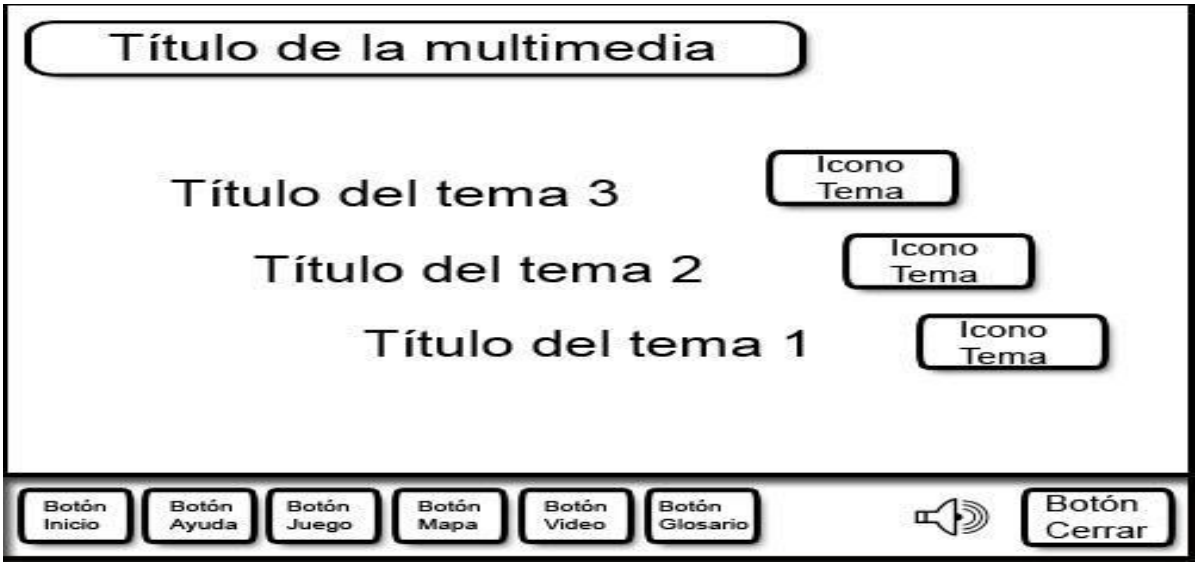


Figura 3. 2 Diagrama de Presentación Pantalla Temas Generales

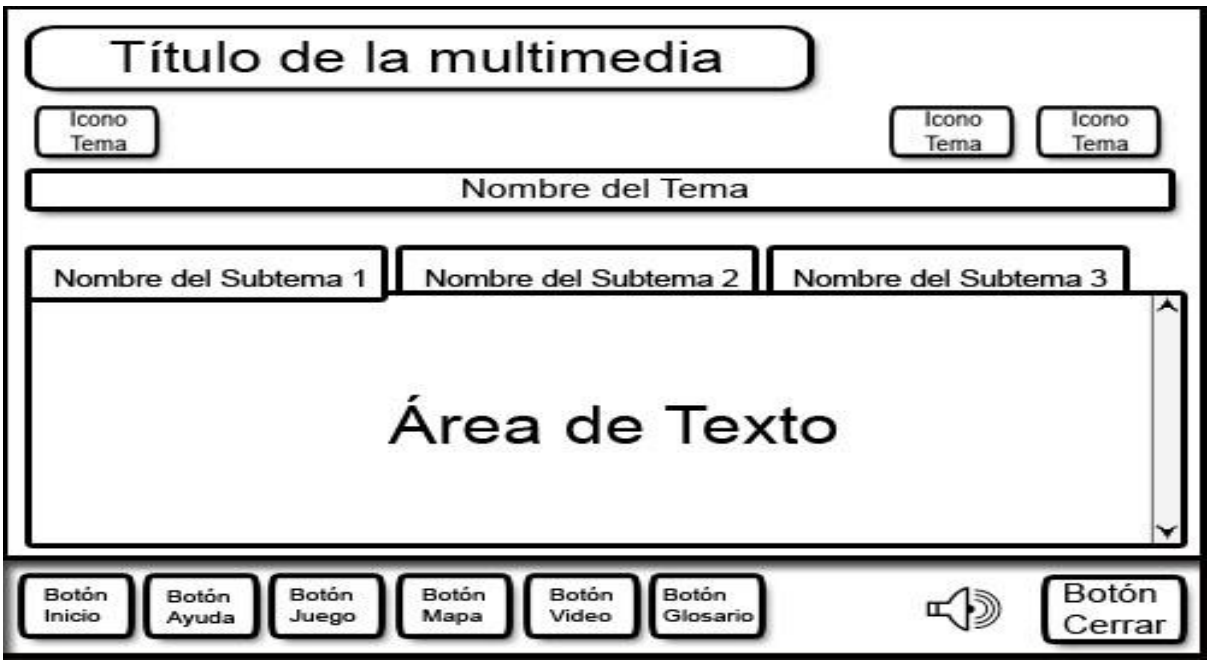


Figura 3. 3 Diagrama de Presentación Pantalla Temas Específicos

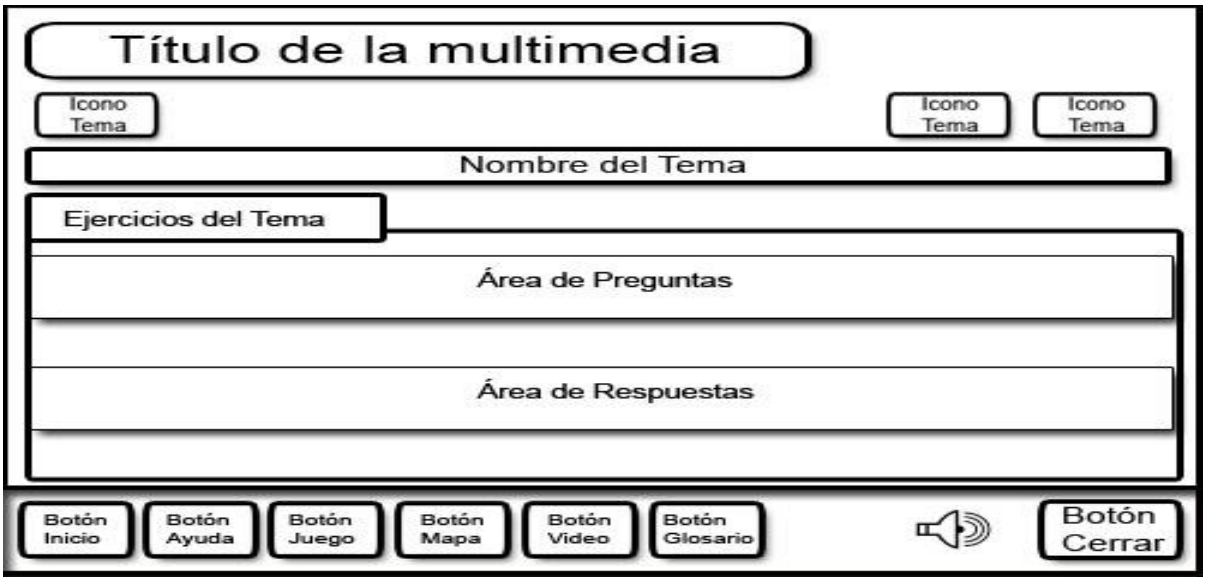


Figura 3. 4 Diagrama de Presentación Pantalla Ejercicio Marcar

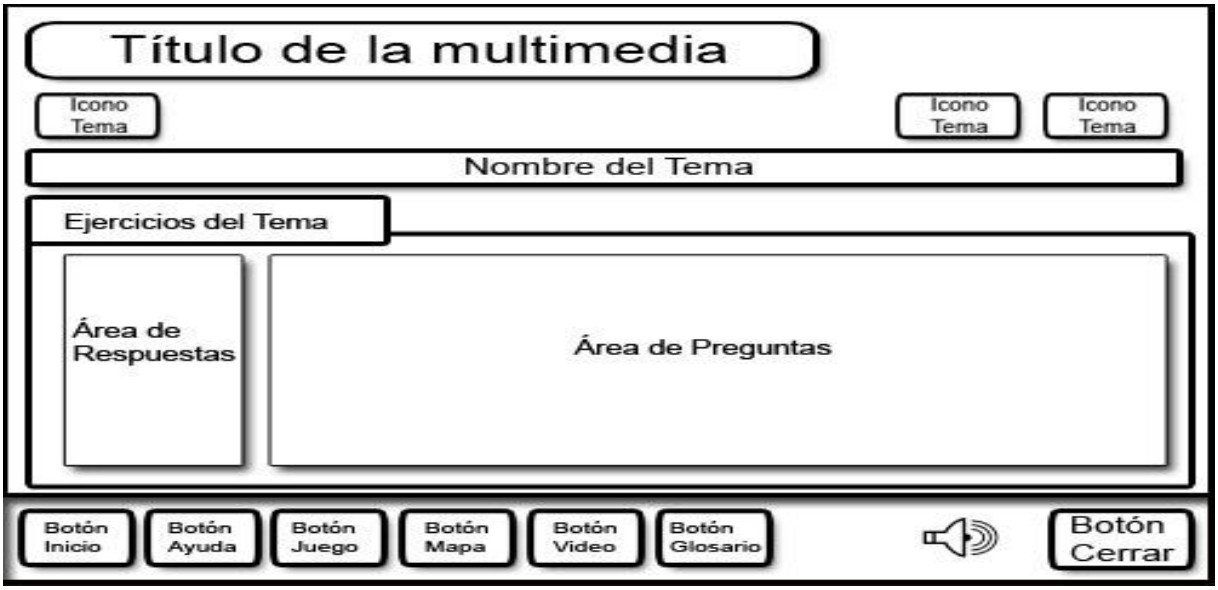


Figura 3. 5 Diagrama de Presentación Pantalla Ejercicio Verdadero/Falso

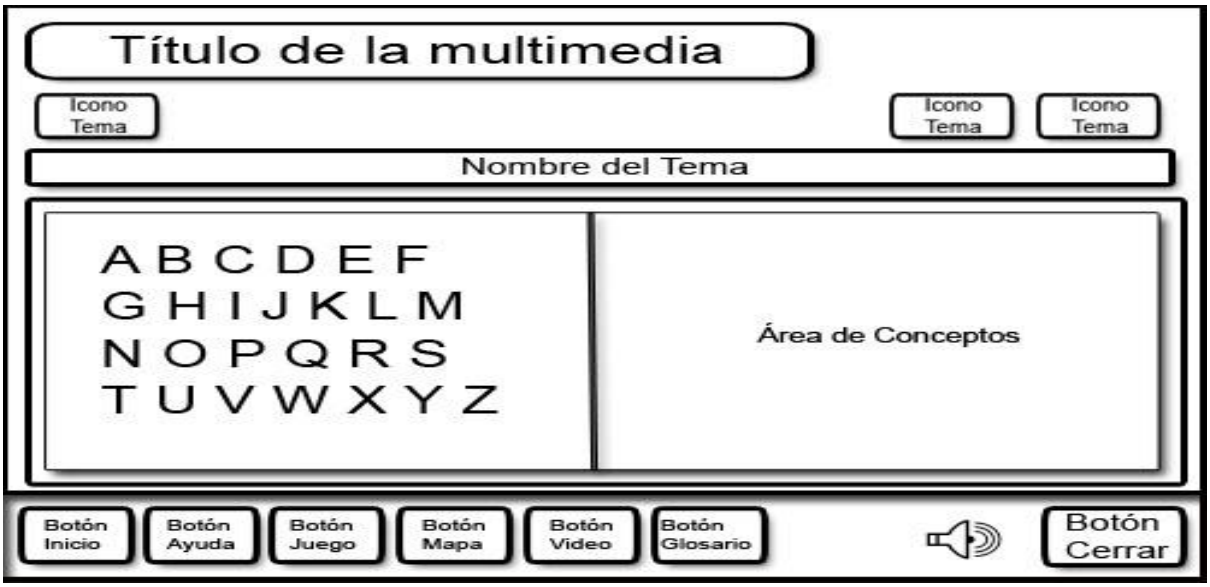


Figura 3. 6 Diagrama de Presentación Pantalla Glosario

Figura 3. 7 Diagrama de Presentación Pantalla Juego

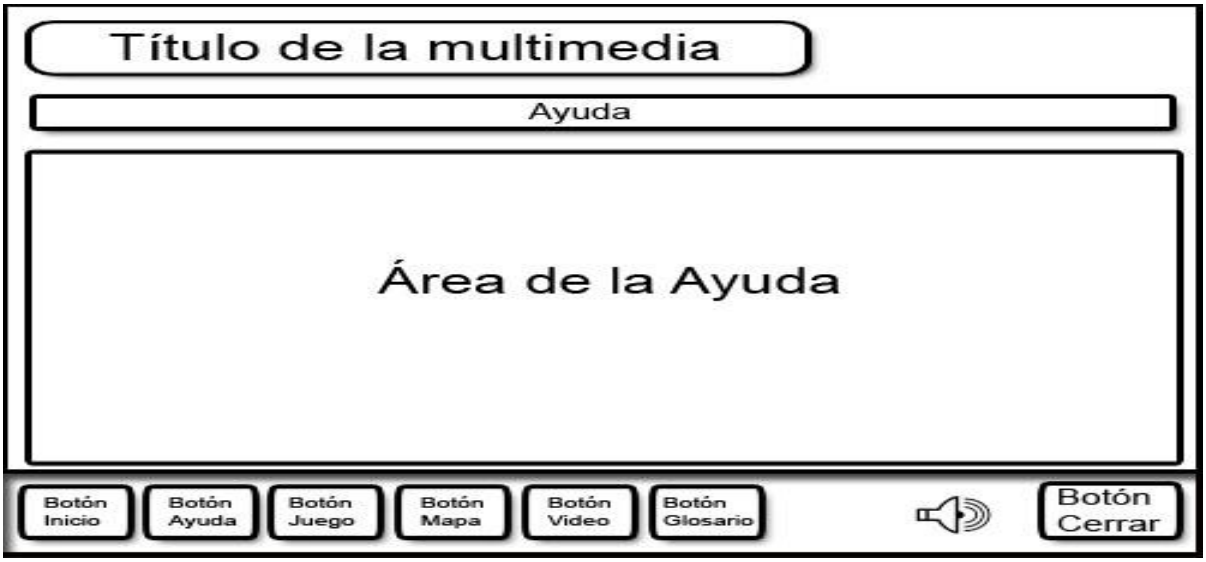


Figura 3. 8 Diagrama de Presentación Pantalla Ayuda

Figura 3. 9 Diagrama de Presentación Pantalla Mapa

Figura 3. 10 Diagrama de Presentación Pantalla Video

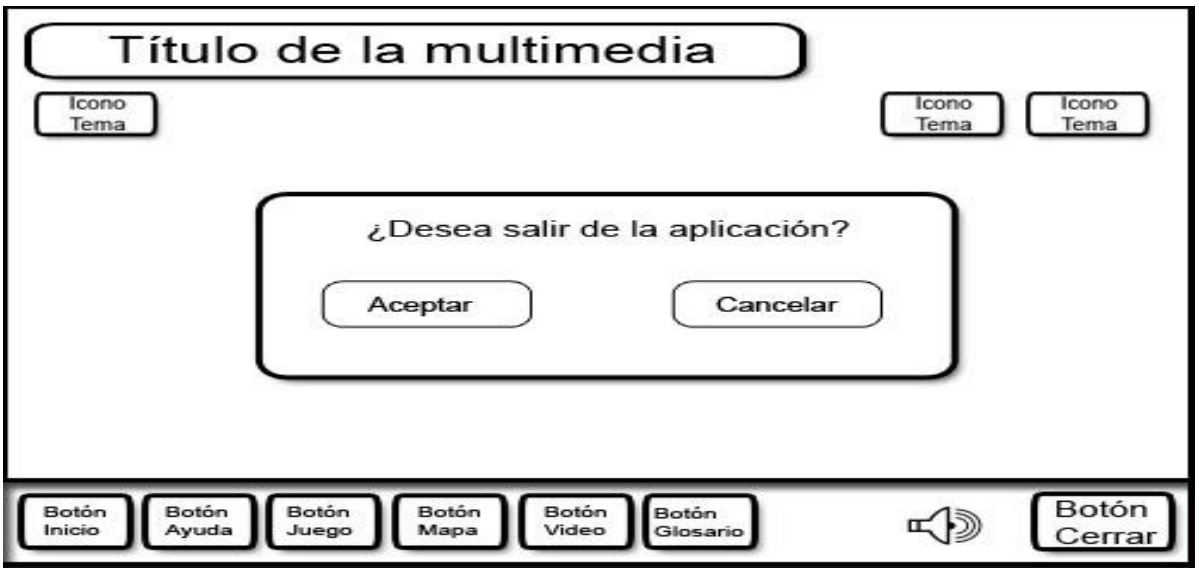


Figura 3. 11 Diagrama de Presentación Pantalla Salir

3.3 Descripción de los archivos XML

Para almacenar la información (textos e imágenes) de la multimedia se utiliza la tecnología XML, lo que permite modificar la información sin necesidad de interactuar con la aplicación.

3.3.1 Descripción de archivos XML Tema

Para cada tema se crea un archivo XML con la siguiente estructura:

```
<tema>
      <nombreTema><p align='center'><i><b>Nombre del tema</b></i>
 </p></nombreTema>
 <subtema1><b><i>Nombre del subtema 1</i></b> </subtema1>
.
< subtema_n><b><i>Nombre del subtema n</i></b> </ subtema_n>
<texto_ subtema1> Texto que pertenece al subtema 1 del tema</texto_ subtema1>
.
.
.
.
.
<texto_ subtema_n> Texto que pertenece al subtema 1 del tema</texto_ subtema_n>
</tema>
```

Los XML de cada tema están estructurados por un nodo padre, por n nodos hijos donde n representa la cantidad de subtemas que tenga cada tema, y por n nodos hijos más que contendrán el texto de cada uno de estos subtemas.

3.3.2 Descripción de los archivos XML ayuda

```
<ayuda>

<bienvenida><p align='center'><font color='#000000' size='22'></font></p>

Bienvenido al curso de Diseño y Construcción!!! </bienvenida>

<descripción><p align='center'><font color='#FF0000' size='15'</font></p>

Se explica al usuario el contenido que encontrará dentro de la multimedia y además cómo puede navegar por la misma.

</descripción>

</ayuda>
```

Descripción de archivos XML Ejercicios

```
<quiz>

    <title> Nombre del título del ejercicio </title>

    <items>
        <item1>

<question>Aquí estará la pregunta que se realizará en el ejercicio que será al estilo Verdadero o Falso </question>

<answer correct="y">Falso</answer>

<answer>Verdadero</answer>

</item1>

<item2>

<question> Aquí estará la pregunta que se realizará en el ejercicio donde el usuario seleccionará la respuesta correcta</question>

<answer>Aquí se pondrán las opciones de respuesta para el usuario </answer>
```

```
<answer correct="y">Aquí se pondrá la respuesta correcta </answer>
</ítem2>
<ítem n>……</ítem n>
</ítems>
</quiz>
```

A continuación se explica detalladamente la función de cada una de las etiquetas usadas en los archivos XML que se cargan en la multimedia.

<quiz> este nodo da inicio al XML de contenido del ejercicio.

</quiz> cierre de nodo.

<title> nombre del título del ejercicio.

<ítems> nodo que va a contener todos los elementos de las preguntas de los ejercicios

</ítems> cierre de nodo.

<item> nodo que va a contener la pregunta y las respuestas como elemento.

</item> cierre de nodo.

<question> nodo que va a contener la pregunta en cuestión.

</question> cierre de nodo.

<answer> nodo que va a contener una posible respuesta a seleccionar.

<answer correct = “y”> nodo que va a contener la respuesta correcta, ya sea una selección de falso o verdadero o una selección de respuesta simple.

</answer> cierre de nodo.

3.4 Modelo de Implementación

El modelo de implementación denota la implementación actual del sistema en términos de componentes y subsistemas de implementación. Es natural mantener el modelo de implementación a lo largo de todo el ciclo de vida del software. Éste describe cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros.

3.4.1 Diagrama de Componentes

El diagrama de componentes representa la separación de un sistema de software en componentes físicos. Se usa para modelar la estructura del software, incluyendo la dependencia entre los componentes del software. Éste muestra las dependencias lógicas entre componentes software, sean éstos componentes fuentes, binarios o ejecutables. Para la realización de este diagrama, los componentes han sido agrupados por paquetes, incluyendo el paquete de archivos XML, ya que esta tecnología fue usada para mostrar la información del producto.

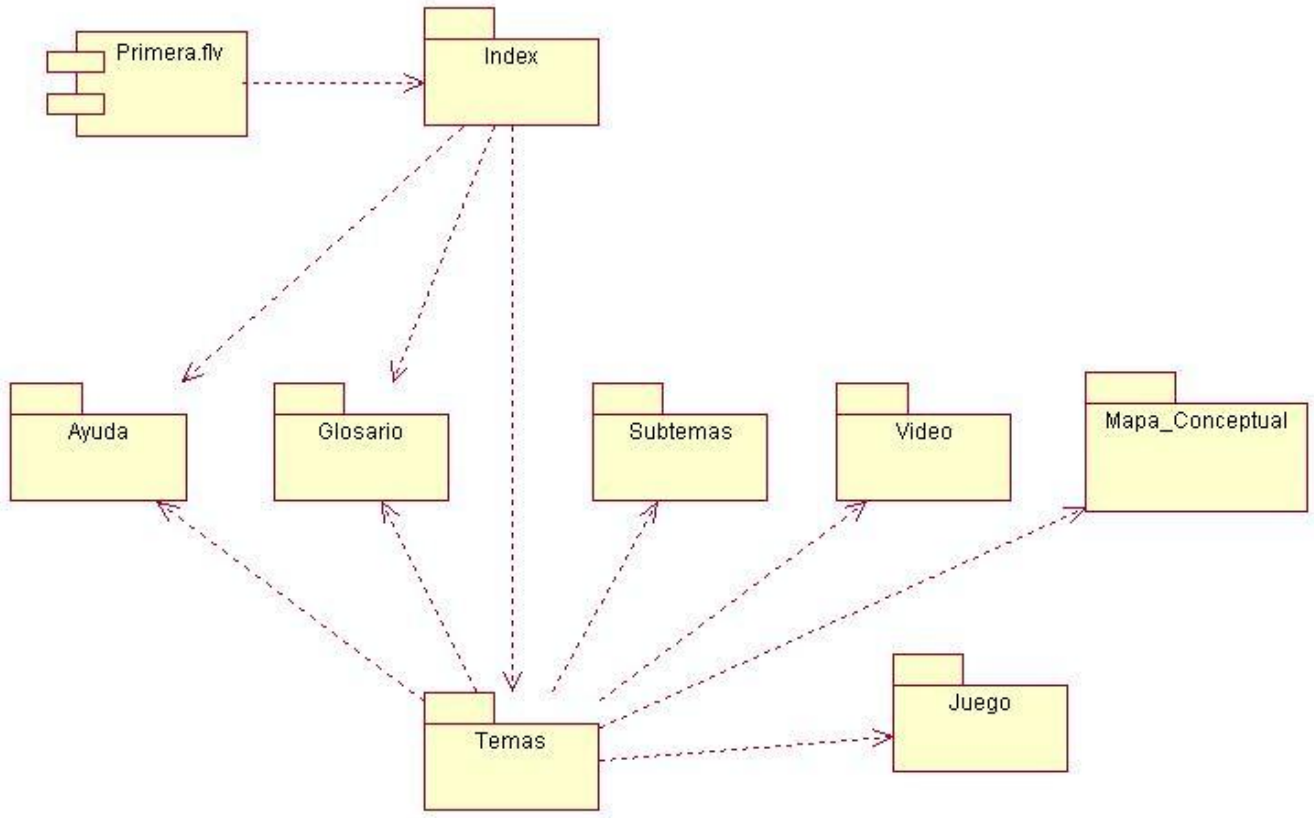


Figura 3. 12 Diagrama de componentes.

Los diagramas de cada paquete se encuentran en los anexos.

3.5 Pruebas al software

A la aplicación se le realizaron pruebas de unidad usando el método de caja blanca y se detectaron 5 no conformidades en la primera iteración de las cuales 2 fueron significativas, en la segunda iteración de pruebas todas las no conformidades fueron solucionadas.

También se realizaron pruebas de sistema donde se implementaron tanto pruebas funcionales como no funcionales, para llevar a cabo las pruebas funcionales se diseñaron 13 casos de pruebas. Para verificar los requisitos no funcionales se diseñaron y aplicaron pruebas de usabilidad, verificando la existencia de un mapa de navegación y el acceso intuitivo a la información.

Cuando se realizó un análisis de las normas que se utilizan a nivel nacional en referencia a la calidad de software, en especial las orientadas a multimedia, el resultado fue la norma ISO 9126 que consta de cuatro partes:

1. ISO/IEC – 9126-1.
2. ISO/IEC – 9126-2.
3. ISO/IEC – 9126-3.
4. ISO/IEC – 9126-4.

La norma 9126-1 trabaja el modelo de calidad el cual abarca fundamentalmente la planificación, los requisitos, el diseño y desarrollo, producción y servicios, seguimiento y medición. Está más orientada a la producción de software tipo multimedia.

Es por eso que se toma la decisión de regirse por dicha norma. Para ello se aplicó el dictamen técnico de evaluación al producto. Se aplicó en una fecha previa a la terminación del producto y nuevamente una vez terminada la multimedia. Ambos dictámenes se encuentran documentados en los anexos.

Conclusiones Parciales

En este capítulo se determinaron todas las normas de diseño e implementación necesarias para cumplir con los requisitos de la aplicación. Se realizó una descripción de los XML usados en la aplicación. En este capítulo es donde se le dio uso al lenguaje OMMMA-L con la elaboración de los diagramas de navegación y presentación. Se realizó el diagrama de componentes ajustado por paquetes y no se realizó un diagrama de despliegue debido a la ausencia de la necesidad de un servidor que almacene la aplicación.

CONCLUSIONES GENERALES

Después de haber realizado un análisis de todos los elementos necesarios para el desarrollo de esta aplicación, y haberle dado cumplimento a las tareas planteadas en la investigación, se puede dar por concluido que:

- Se realizó un estudio del SWEBOK y el SEEK, estándares que abordan los temas de Diseño y Construcción de Software y, tras un análisis de los puntos esenciales que tienen en común, se determinó el contenido a presentar en la multimedia.

- Se desarrolló la Multimedia Interactiva para la enseñanza del Diseño y la Construcción de Software que cumple con los requisitos que se plantearon en la descripción de la solución, utilizando la metodología de desarrollo RUP, la herramienta Flash 8, como lenguaje de modelado OMMMA-L y de programación ActionScript.

- En la multimedia se definieron e implementaron herramientas que apoyan el proceso de enseñanza – aprendizaje.

- La multimedia cumple con todas las normas de diseño e implementación que se definieron.

RECOMENDACIONES

- Se recomienda integrar en un único producto las aplicaciones multimedia: Gestión de Calidad de Software, Métricas de Software, Gestión de la Configuración, Auditorías y Revisiones, Mantenimiento de Software y Diseño y Construcción de Software, con el objetivo de que los usuarios puedan acceder desde una misma aplicación a cualquiera de los cursos que desee consultar.
- Enriquecer el contenido de cada uno de los temas que conforman el producto.
- Valorar la posibilidad de migrar la aplicación a software libre.
- Utilizar el software propuesto como material de apoyo de los cursos de Diseño y Construcción de software.

TRABAJOS CITADOS

Acuña, Luis Alvarado. 2002. *La Gestión del Conocimiento y la utilización de las Tecnologías de la Información y de las Comunicaciones en la creación de valor en los proyectos de innovación.* 2002.

AulaClic. 2006. Lo nuevo de flash 8. [En línea] 2006.

Bianchini, Adelaide. 2000. 2000.

Calidad, Dirección de. 2008. *Diagnóstico.* 2008.

Carrión, Juan. *Gestión del Conocimiento.*

Corporation, IBM. 2006. *Decisiones importantes de implementación.* 2006.

Dalton, Bill. 2007. *Clasificación de software educativo.* 2007.

Dellamea, Esther. 2006. 2006.

Diaz, Carlos. 1994. *LA TECNOLOGIA MULTIMEDIA: Una Nueva Tecnología de Comunicación e Información. Características, concepciones y aplicaciones.* 1994.

Evaristo, Víctor Hugo Reyes. 2006. *Propuesta de metodología para el diseño, desar rollo y evaluación de.* 2006.

García, Miquel Barceló. 2002. *El proyectoinformático de construcción de software.* 2002.

Herron, David. 2000. *Function Point Analysis: Measurement Practices for Successful Software Projects.* 2000.

Jones, Capers. 2000. *Software Assessments Benchamarks, and Best Practices.* 2000.

Lamas, Rguez. 2000. 2000.

Marques Graell, Pere. El software educativo. [En línea] http://www.xtec.es/-pmarques/edusof.htm.

Multimediam. 2006. ¿Qué es Multimedia Interactiva? [En línea] 2006.

Muñante, Jorge Díaz. 2003. *Modelo de Gestión del Conocimiento aplicado a la Universidad Pública de.* 2003.

Pérez, Y. 2005. *La Gestión del conocimiento un nuevo enfoque en la gestión empresarial.* 2005.

Pérez, Yamilis Fernández. 2010. 2010.

Pérez, Yancy Martínez. 2006. *PLANTILLA PARA EL MONTAJE DINÁMICO DE LOS.* 2006.

2006. *Revisiones de código y estándares de codificación.* 2006.

Rodriguez, Douglas Francisco Zambrano. 2007. *Multimedia.* 2007.

Rodríguez, Kethicer Castellanos. 2000. *Software educativo. Su influencia es la escuela cubana.* 2000.

Rojas, Juan Carlos Olivares. 2006. *Fundamentos del diseño de software.* 2006.

ROMERO, José Mª GARRIDO. 1991. *Diseño y creación de software educativo. .* 1991.

Sabatini, Alejandro Gustavo. 2008. 2008.

—. 2008. *Hipermedia (Hiperfilmes + Hipertexto + Hipergrama).* 2008.

Sánchez, J. 1999. *Construyendo y Aprendiendo con el Computador.* 1999.

SEEK. 2004. *SEEK.* 2004.

Society, IEEE Computer. 2004. *SWEBOOK.* 2004.

Takeuchi, Nonaka y. 1995. 1995.

Valdés, José Alberto. 2005. *La Gestión del Conocimiento y la Educación Superior. Fundamentos teóricos.* 2005.

VAUGHAN, Tay. 1994. *Todo el poder de la Multimedia.* 1994.

TRABAJOS CONSULTADOS

IBM Corporation. 2006. *Rational Unificied Process versión 7.2*. 2006.

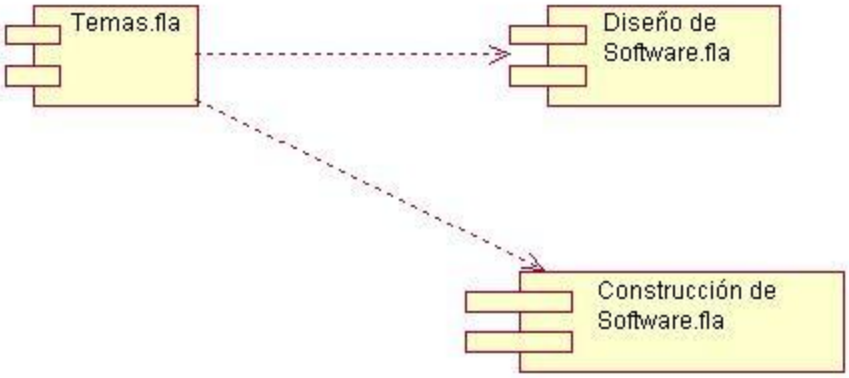
IEEE. 1990. *IEEE Standard Glossary of Software Engineering Terminology.* 1990.

Jacobson, Ivar. 2004. *El Proceso Unificado de Desarrollo de Software*. s.l. : Félix Varela, 2004.

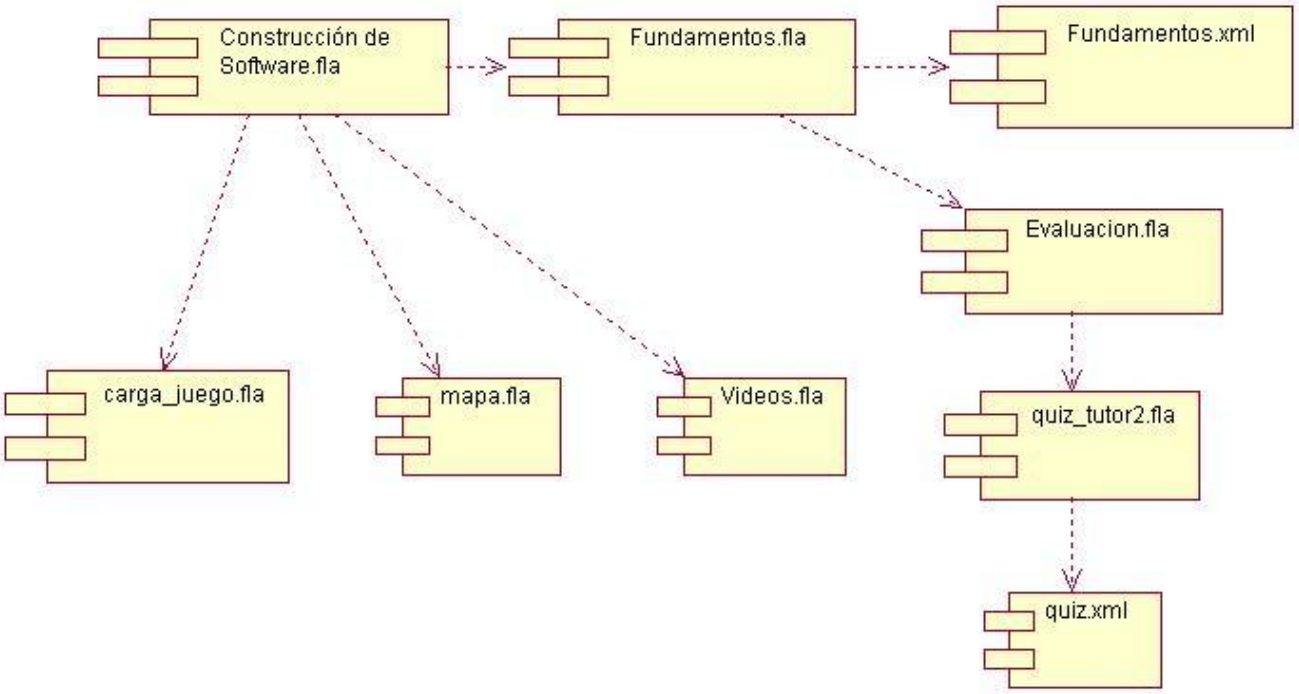
OMG. 2007. Titulo Introduccion a los OMG Lenguaje unificado de modelado(UML). [En línea] 9 de 11 de 2007. www.uml.org.

Pressman, R. 1998. *Ingeniería de software. Un enfoque práctico*. Madrid : Mc Graw-Hill Interamericana de España S.A : s.n., 1998.

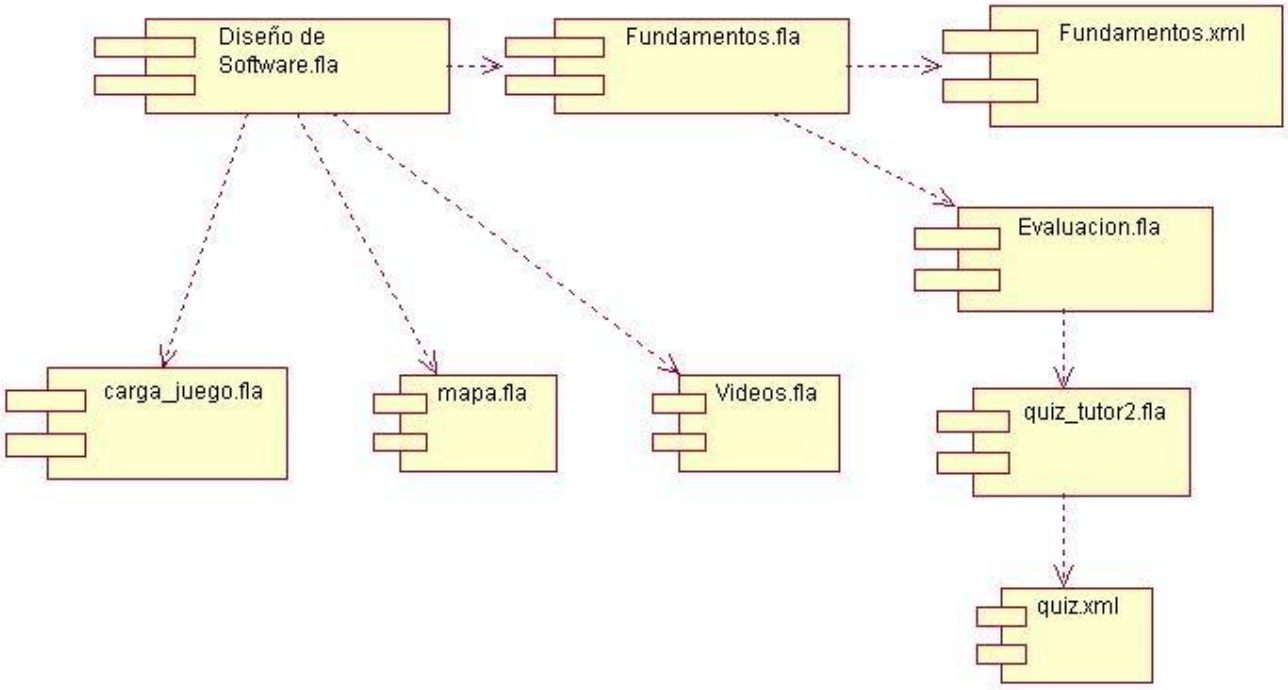
ANEXOS



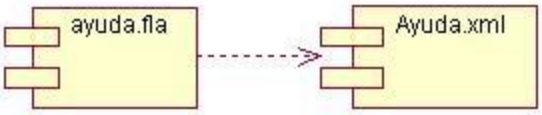
Anexo 1 Paquete temas.



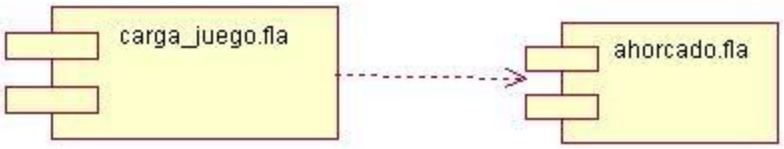
Anexo 2 Paquete tema Construcción.



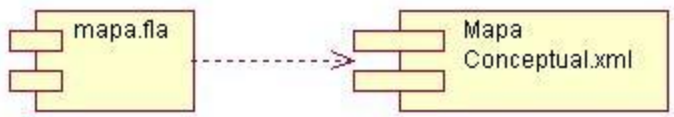
Anexo 3 Paquete tema Diseño.



Anexo 4 Paquete ayuda.

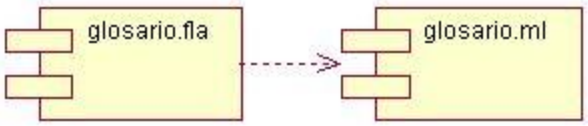


Anexo 5 Paquete juego.



Anexo 6 Paquete mapa conceptual.

Anexo 7 Paquete videos.



Anexo 8 Paquete glosario.

Dictamen Técnico No.1

Producto: Multimedia Interactiva para la enseñanza del Diseño y la Construcción de Software.
Fecha: 7/5/10
La revisión se realizó en una Pentium IV de 248 MB de RAM con W/XP.

La evolución del producto se realizó teniendo en cuenta lo siguiente:

- Norma cubana NC ISO 9126-1:2005 Ingeniería de Software. Calidad del producto.
- Procedimientos establecidos en el Sistema de Gestión de la calidad de CITMATEL para la producción de software:
    1. P03.01 - Sobre la evaluación de productos de software.

2. P03.02 - Sobre los créditos institucionales.
3. P03.05 – Sobre la entrega de productos terminados a comercial.

Atributos medidos: Funcionalidad, confiabilidad, usabilidad, eficiencia, mantenibilidad, portabilidad y reusabilidad.

- **Funcionalidad:**
  - La multimedia no está terminada por lo que no están la totalidad de las funcionalidades acordadas en los requisitos establecidos.
  - No se cumple el requisito funcional de mostrar la presentación que de inicio a la página principal de la multimedia.
  - No presenta objeto de activar o desactivar audio.
  - Los ejercicios aun no son accesibles.

- **Confiabilidad:**
  (No hay señalamientos)

- **Usabilidad:**
  - Los niveles de ayuda aun no están implementados.

- **Eficiencia:**
  (No hay señalamientos)

- **Mantenibilidad:**
  - En caso de fallos no tiene implementada ninguna técnica que indique al usuario qué sucede.

- **Portabilidad:**
  (No hay señalamientos)

- **Reusabilidad:**
  (No hay señalamientos)

Sobre los diseños del producto:

(No hay señalamientos)

Criterio: La multimedia Interactiva para la enseñanza del Diseño y la Construcción de Software puede continuar en la etapa de implementación ya que es una multimedia muy necesaria para la educación. No posee errores críticos de contenido ni de funcionamiento. Los autores deben corregir los señalamientos registrados en el dictamen.

Dictamen Técnico No.2

Producto: Multimedia Interactiva para la enseñanza del Diseño y la Construcción de Software.
Fecha: 7/6/10
La revisión se realizó en una Pentium IV de 248 MB de RAM con W/XP.

La evolución del producto se realizó teniendo en cuenta lo siguiente:

- Norma cubana NC ISO 9126-1:2005 Ingeniería de Software. Calidad del producto.
- Procedimientos establecidos en el Sistema de Gestión de la calidad de CITMATEL para la producción de software:
    1. P03.01 - Sobre la evaluación de productos de software.
    2. P03.02 - Sobre los créditos institucionales.
    3. P03.05 – Sobre la entrega de productos terminados a comercial.

Atributos medidos: Funcionalidad, confiabilidad, usabilidad, eficiencia, mantenibilidad, portabilidad y reusabilidad.

- **Funcionalidad:**
  (No hay señalamientos)

- **Confiabilidad:**
  (No hay señalamientos)

- **Usabilidad:**
  (No hay señalamientos)

- **Eficiencia:**

  (No hay señalamientos)

- **Mantenibilidad:**
  - En caso de fallos no tiene implementada ninguna técnica que indique al usuario qué sucede.

- **Portabilidad:**

  (No hay señalamientos)

- **Reusabilidad:**

  (No hay señalamientos)

Sobre los diseños del producto:

(No hay señalamientos)

Criterio: La multimedia Interactiva para la enseñanza del Diseño y la Construcción de Software no posee errores significativos de contenido ni de funcionamiento.

GLOSARIO DE TÉRMINOS

Buenas Prácticas: La práctica innovadora que contribuye a la mejora del desempeño de una organización en el contexto dado. (IEEE Std 610.12-1990).

Diagrama: Representación gráfica en la que se muestran las relaciones entre las diferentes partes de un conjunto o sistema.

Estándar: Se refiere a los requisitos formales obligatorios desarrollados y utilizados para prescribir enfoques coherentes al desarrollo (por ejemplo, normas ISO / IEC, las normas de IEEE, normas de organización). (CMMI, 2006).

Eficiencia: El grado en que un sistema o componente designado desempeña sus funciones con un mínimo consumo de recursos. (IEEE Std 610.12-1990).

Gestión: Disciplina del proceso de ingeniería de software, cuyo propósito es planificar y gestionar el proyecto de desarrollo. (IBM Corporation, 2006).

Interfaz: (1) Un límite compartido a través del cual la información se transmite. (2) Un componente de hardware o software que conecta dos o más componentes con el propósito de transmitir información de uno a otro. (3) Para conectar dos o más componentes con el propósito de transmitir información de uno a otro. (IEEE Std 610.12-1990).

Multimedia: Es un sistema que utiliza más de un medio de comunicación al mismo tiempo en la presentación de la información, como texto, imagen, animación, vídeo y sonido.

Proceso: Conjunto de recursos y actividades relacionadas entre sí que transforman elementos entrantes en elementos salientes. (ISO 9000, 2000)

Requisito funcional: Un requisito que especifique una función que un sistema o componente del sistema debe ser capaz de realizar. Contrasta con: requisito de diseño, requisito de aplicación, la interfaz requisito; requisito de desempeño; exigencia física. (IEEE Std 610.12-1990)

Usuario: Un individuo u organización que utiliza el sistema operativo para ejecutar una función específica. (NC ISO/IEC 12207: 2005)

1