

**Universidad de las Ciencias Informáticas**

**Facultad 3**



**Título: “Análisis y Diseño de una Multimedia para niños de la enseñanza preescolar con problemas del lenguaje”**

Trabajo de Diploma para optar por el título de Ingeniero en  
Ciencias Informáticas

**Autor:** Liudmila Licette Pérez Díaz

**Tutor:** Ing. Yinett Hernández Hernández

Ciudad de la Habana

Junio 2009

## **DECLARACIÓN DE AUTORÍA**

Declaro ser autora de la presente tesis y reconozco 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 \_\_\_\_\_.

**Liudmila Licette Pérez Díaz**

**Yinett Hernández Hernández**

\_\_\_\_\_

Firma del Autor

\_\_\_\_\_

Firma del Tutor

## AGRADECIMIENTOS

### Agradezco:

*Primeramente a Dios por guiarme, darme la fuerza, la salud y permitirme ser lo que soy hoy.*

*A Yerandy por su apoyo incondicional, sin ti no sé que me hubiese hecho en la realización de este Trabajo.*

*A mis padres por su amor e incondicionalidad.*

*A mi tutora Yinett por brindarme su ayuda y su paciencia.*

*A mis amigos Balía y Yudel por estar conmigo en las buenas y malas. Dicen que los amigos son los hermanos que Dios olvidó darnos, ustedes son los mejores amigos del mundo.*

*A mis compañeras de aula Dustell, Santos y Ana por su amistad a lo largo de estos años de carrera universitaria, por aceptarme como soy, a ustedes muchísimas gracias por estar ahí siempre que las necesité.*

*De forma general a todas las personas que directa o indirectamente hicieron posible la realización de este trabajo.*

## DEDICATORIA

### Dedico esta Tesis:

*A mis padres por su confianza y sacrificio, porque siempre han esperado de mí lo mejor, porque nunca será suficiente lo que haga por ellos, por ser mi guía, mi apoyo, por todo su amor, comprensión, por darme las fuerzas para seguir adelante, Atribuyo todos mis éxitos en esta vida a la enseñanza moral, intelectual y física que recibí de ellos...*

*A Rosangel por ser la niña más linda y representar para mí el mayor tesoro del mundo...*

*A mi familia en general, por todo el apoyo que he recibido de ellos, por creer en mí...*

*A la Universidad de las Ciencias Informáticas, donde viví momentos inolvidables, donde hice las mejores amistades y me crecí como persona,*

## **RESUMEN**

El desarrollo de las nuevas tecnologías de la informática ha producido un salto fundamental en la productividad de las actividades de la sociedad. Estas son empleadas principalmente en el desarrollo de software, teniendo como objetivo resolver cualquier problemática proporcionada. Por tal motivo el presente trabajo está enmarcado en desarrollar el análisis y diseño de un software con tecnología multimedia para los niños de la enseñanza preescolar que presentan problemas del lenguaje. Este documento recoge todo el proceso de modelación del negocio, levantamiento de requisitos, análisis y diseño del software. Para su desarrollo se analizan las tendencias, tecnologías y las posibles herramientas a utilizar.

Con este trabajo se aporta el análisis y diseño de un Software como material de soporte; consiste en la propuesta de un grupo de actividades con el fin de disminuir los problemas del lenguaje que existen en la actualidad en los niños de la enseñanza preescolar, ofreciendo la posibilidad de utilizar la computadora como medio de enseñanza e instrumento de trabajo.

### **PALABRAS CLAVE**

Multimedia, Software, Tecnología, Herramientas, Problemas del Lenguaje.

# ÍNDICE DE CONTENIDO

AGRADECIMIENTOS .....	II
INTRODUCCIÓN .....	1
CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA .....	5
1.1 Introducción .....	5
1.2 Conceptos Asociados .....	5
1.3 ¿Por qué la Enseñanza Preescolar? .....	5
1.4 Análisis de las Soluciones Existentes .....	6
1.5 Metodologías de Desarrollo de Software.....	7
1.5.1 Rational Unified Process (RUP) .....	7
1.5.2 Extreme Programing (XP) .....	8
1.5.3 SCRUM.....	9
1.6 Lenguajes de Modelado .....	9
1.6.1 Lenguaje Unificado de Modelado (UML) .....	10
1.6.2 Notación para el Modelado de Procesos del Negocio (BPMN).....	10
1.7 Herramientas de Modelado. ....	10
1.7.1 Visual Paradigm.....	11
1.7.2 Rational Rose .....	12
1.7.3 Enterprise Architect .....	12
1.8 Herramientas Propuestas para la Implementación .....	12
1.8.1 Adobe Director.....	12
1.8.2 ToolBook .....	13
1.8.3 Adobe Flash CS3 .....	13
1.9 Ingeniería de Requisitos.....	14
1.9.1 Técnicas para el Levantamiento de Requisitos .....	15
1.10 Patrones de Casos de Uso .....	16
1.11 Patrones de Diseño.....	20
1.12 Principios de Diseño .....	23
1.13 Métodos para la Validación de Resultados. ....	25
1.13.1 Técnica de Prototipado.....	25
1.13.2 Métrica de la Calidad de Especificación de los Requisitos .....	25
1.13.3 Heurística basada en NOAS/NOS.....	26
1.13.4 Métricas Propuestas por Lorenz y Kidd (Tamaño de Clases).....	26
1.14 Conclusiones .....	27
CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA. ....	28
2.1 Introducción .....	28
2.2 Modelo de Dominio .....	28
2.3 Descripción del Sistema Propuesto.....	30
2.3.1 Especificación del Contenido.....	30
2.4 Descripción de la Funcionalidad.....	30
2.4.1 Especificación de los Requisitos del software .....	30
2.4.2 Captura de Requisitos Funcionales .....	31
2.4.3 Requisitos no Funcionales .....	31

2.5 Modelo de Caso de Uso del Sistema.....	32
2.5.1 Determinación y Justificación de los Actores del Sistema .....	33
2.5.2 Determinación y Justificación de los Casos de Uso del Sistema.....	33
2.5.3 Descripción de Casos de Uso.....	35
2.6 Conclusiones .....	44
<b>CAPÍTULO 3: ANÁLISIS Y DISEÑO DE LA SOLUCIÓN PROPUESTA .....</b>	<b>45</b>
3.1 Introducción .....	45
3.2 Análisis .....	45
3.2.1 Modelo de Clases de Análisis por Caso de Uso.....	45
3.2.2 Diagramas de Interacción. Colaboración.....	47
3.3 Diseño .....	50
3.3.1 Diagramas de Interacción. Secuencia. ....	50
3.3.2 Diagramas de Clases de Diseño .....	54
3.4 Conclusiones .....	55
<b>CAPÍTULO 4: VALIDACIÓN DE RESULTADOS .....</b>	<b>56</b>
4.1 Introducción .....	56
4.2 Métrica de la Calidad de Especificación de los Requisitos .....	56
4.3 Técnica de Prototipado.....	58
4.4 Modelo de Métricas Orientadas a Objeto aplicadas al DCUS.....	58
4.5 Métricas Orientadas al Tamaño de Clase (TC).....	59
4.6 Conclusiones .....	60
<b>CONCLUSIONES .....</b>	<b>62</b>
<b>RECOMENDACIONES .....</b>	<b>63</b>
<b>REFERENCIAS BIBLIOGRÁFICAS .....</b>	<b>64</b>
<b>BIBLIOGRAFÍA.....</b>	<b>67</b>
<b>GLOSARIO DE TÉRMINOS .....</b>	<b>70</b>
<b>ANEXOS.....</b>	<b>72</b>
Anexo 1. Ejercicios de Pronunciación .....	72
Anexo 2: Plantilla para la Descripción de Casos de Uso.....	81

## **ÍNDICE DE FIGURAS Y TABLAS**

Figura 1. La Ingeniería de Requisitos como puente entre los espacios del problema y la solución.....	14
Figura 2. Modelo de Dominio.....	29
Figura 3. Diagrama de Casos de Uso.....	35
Figura 4. Diagrama de Clases de Análisis. Navegar por el Sistema .....	46
Figura 5. Diagrama de Clases de Análisis. Consultar Ayuda. ....	46
Figura 6. Diagrama de Clases de Análisis. Visualizar Ejercicio.....	46
Figura 7. Diagrama de Clases de Análisis. Manipular Audio .....	47
Figura 8. Diagrama de Colaboración. Navegar por el sistema. Escenario Cargar Presentación .....	47

Figura 9. Diagrama de Colaboración. Navegar por el sistema. Escenario Ver Inicio .....	48
Figura 10. Diagrama de Colaboración. Navegar por el sistema. Escenario Ver Índice .....	48
Figura 11. Diagrama de Colaboración. Navegar por el sistema. Escenario Salir .....	48
Figura 12. Diagrama de Colaboración. Consultar Ayuda. ....	49
Figura 13. Diagrama de Colaboración. Visualizar Ejercicio. ....	49
Figura 14. Diagrama de Colaboración. Manipular Audio. ....	49
Figura 15. Diagrama de Secuencia. Navegar por el Sistema. Escenario Cargar Presentación .....	50
Figura 16. Diagrama de Secuencia. Navegar por el Sistema. Escenario Ver Inicio .....	51
Figura 17. Diagrama de Secuencia. Navegar por el Sistema. Escenario Ver Índice .....	51
Figura 18. Diagrama de Secuencia. Navegar por el Sistema. Escenario Salir del sistema.....	52
Figura 19. Diagrama de Secuencia. Consultar Ayuda .....	52
Figura 20. Diagrama de Secuencia. Visualizar Ejercicio. ....	53
Figura 21. Diagrama de Secuencia. Manipular Audio. ....	53
Figura 22. Diagrama de Clases de Diseño. ....	54
Tabla 1. Representación de la Métrica aplicada al Diagrama de Casos de Uso .....	58
Tabla 2. Tamaño de las clases. ....	59
Tabla 3. Clasificación de Clases .....	60
Tabla 4. Promedio de Operaciones de Clases. ....	60



### INTRODUCCIÓN

En Cuba pocos años atrás significaba una utopía pensar en emplear el software educativo en el desarrollo de la educación; sin embargo, actualmente los avances tecnológicos permiten el empleo del imprescindible medio informático en las diferentes enseñanzas. Hasta la fecha existen numerosos programas creados para la gestión económica, la esfera militar, las investigaciones, el entrenamiento, la salud, la educación y otros muchos campos de aplicación. Hoy, se ha logrado alcanzar una alta relevancia en la educación, teniendo en cuenta, precisamente, el inmenso volumen de información que dispone el hombre en los momentos actuales y los propios factores que han motivado una masividad en el uso de esta tecnología. A medida que prospera la informatización de la sociedad cubana se incrementa la implementación de recursos informáticos en las escuelas que benefician la calidad de las clases.

En la educación existen grandes perspectivas a causa de la gran diversidad de asignaturas, su forma de programación y su conjugación con otras, entre otros muchos factores; de ahí que la construcción de medios de enseñanza computarizados sea un reto en los momentos actuales y una inversión cuyos resultados se obtienen en tiempo futuro.

El software educativo constituye una evidencia del impacto de la tecnología en la educación pues es la más reciente herramienta didáctica, convirtiéndose en una alternativa válida para ofrecer al usuario un ambiente propicio para que fluya el conocimiento. Hoy se avanza vertiginosamente en su desarrollo en todos los niveles de enseñanza como una vía eficiente para elevar la calidad en la educación. Según los especialistas el software educativo se caracteriza por ser altamente interactivo, 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.

En la actualidad no existe ningún software con tecnología multimedia diseñado para resolver los problemas del lenguaje en los niños de la enseñanza preescolar; sin embargo se conoce de la existencia de muchos niños que presentan estos problemas, por lo que en cada una de las escuelas hay un personal especializado con un proceso de enseñanza–aprendizaje específico y con características determinadas con el objetivo de darles una atención adecuada y diferenciada para contribuir a su formación, concentración y orientación en el espacio, además de desarrollar sus capacidades y habilidades.

La función del logopeda no es enseñar a hablar bien sino es proporcionar a las personas que lo necesitan un sistema de comunicación que les permita expresarse ya sea mediante el lenguaje oral, escrito, gestual y/o pictográfico (dibujos).

Es necesario hacer un análisis previo donde se identifiquen las necesidades del cliente y se traduzcan al lenguaje de los desarrolladores, pues estos últimos no tienen claridad acerca de lo que el cliente quiere y no se delimita qué debe y qué no debe hacer el producto. Es decir, es necesario llegar a un acuerdo entre los involucrados sobre lo que el software debe hacer, para así proporcionar a los desarrolladores un mejor entendimiento de los requisitos del software y transformarlos en un lenguaje entendible para la futura implementación.

Luego del análisis de la situación actual, surge el siguiente **problema científico**: ¿Cómo obtener los artefactos entendibles para los desarrolladores que sirvan de entrada para la futura implementación de un software con tecnología multimedia que contribuya a disminuir los problemas del lenguaje en niños de la enseñanza preescolar?

El problema científico se enmarca en el **objeto de estudio**: Proceso de desarrollo de software. Teniendo en cuenta esto, el **campo de acción** se reduciría al análisis y diseño de un software educativo con tecnología multimedia.

El **objetivo general** es realizar el análisis y diseño de un software educativo con tecnología multimedia para niños de la enseñanza preescolar con problemas del lenguaje.

Lo que conlleva a la siguiente **Hipótesis**:

Si se realiza el análisis y diseño de un software educativo con tecnología multimedia para niños de la enseñanza preescolar con problemas del lenguaje, entonces se obtendrían los artefactos entendibles para los desarrolladores que servirían de entrada para la futura implementación del mismo.

A continuación se muestran las **Tareas** de la investigación trazadas para dar cumplimiento al objetivo general:

- Realizar búsquedas bibliográficas encaminadas a investigar algunos temas relacionados con los problemas del lenguaje en niños de la enseñanza preescolar, así como las soluciones informáticas existentes para disminuirlos.
- Estudiar las metodologías de desarrollo de software, herramientas CASE y lenguajes de modelado más utilizados actualmente para el desarrollo de productos con tecnología multimedia con el fin de seleccionar y aplicar los más adecuados.
- Confeccionar el modelo del negocio.

- Especificar los requisitos del software.
- Elaborar los artefactos de análisis y diseño.
- Demostrar la calidad del análisis y diseño propuestos mediante el uso de métodos de validación.

Para llevar a cabo esta investigación se tuvieron en cuenta algunos métodos, a continuación se describirán los que fueron usados en este trabajo.

### **Métodos Científicos de Investigación:**

#### **Teóricos**

**Métodos de análisis-síntesis e inducción-deducción:** Este método propició el estudio de los conceptos empleados en el campo, permitiendo la extracción de los elementos más importantes que se relacionan con el desarrollo de la aplicación.

**Análisis histórico-lógico:** Permitió conocer, con mayor profundidad, los antecedentes y las tendencias actuales referidas al tema a tratar, además de conceptos, términos y vocabularios propios del campo.

**Método de modelación:** Facilitó la caracterización de las funcionalidades que tendrá la aplicación.

#### **Empíricos**

**Observación:** Se recogió la información de cada uno de los conceptos o variables definidas en la hipótesis.

Además vale resaltar la importancia de las **entrevistas** que propiciaron recoger variadas opiniones de especialista en el tema tratado, así como **búsquedas bibliográficas** que facilitaron la obtención de materiales en beneficio de la aplicación.

Como resultado se espera el modelo de análisis y diseño de una aplicación con tecnología multimedia.

Para la confección del trabajo de diploma que se propone desarrollar, se muestran cuatro capítulos, estructurados de la siguiente forma:

En el **capítulo 1** se analizarán los conceptos que serán necesarios para el conocimiento de los diferentes temas que fundamentan el trabajo. Se llevará a cabo un estudio profundo sobre las soluciones existentes, metodologías y herramientas a utilizar, para establecer una comparación que permita escoger las más adecuadas. También se realizará un estudio acerca de los principios y patrones de diseño y las métricas para la validación de los resultados.

En el **capítulo 2** se comienza a construir el software que se desea desarrollar, describiendo los flujos de trabajo Modelación del Negocio y Captura de Requisitos, elaborando para ello los artefactos correspondientes. Aquí se realizará la presentación del Modelo de Dominio, se capturarán los Requisitos Funcionales y No Funcionales del Software, se definirán los actores y casos de uso del sistema, los cuáles se integrarán en el Diagrama de Casos de Uso. Finalmente se realizarán las descripciones por cada caso de uso para una mayor comprensión de las acciones en cada uno.

En el **capítulo 3** se trabajará con el flujo de trabajo Análisis y Diseño. Se van a representar los artefactos correspondientes a este flujo de trabajo. Para el análisis se realizarán los diagramas de clases de análisis para cada caso de uso y los diagramas de colaboración, y para el diseño los diagramas de secuencia y los diagramas de clases del diseño. Todos estos artefactos servirán para que se tenga un mejor entendimiento del funcionamiento del software a construir.

En el **capítulo 4** se realizará la validación de los resultados. Mediante la aplicación de métricas y otros métodos se evaluará la calidad del trabajo realizado.

### CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

#### 1.1 Introducción

En este capítulo se expondrán las definiciones de Software, Multimedia, Multimedia Educativa, Logopedia como ciencia, entre otros conceptos necesarios para el conocimiento de los diferentes temas que fundamentan el trabajo. También se analizarán las soluciones existentes, entre ellas las que existen en Cuba que ayuden a prevenir o disminuir los problemas del lenguaje. Por otro lado se realizará un estudio acerca de las metodologías y herramientas para establecer una comparación que permita escoger la más adecuada en la realización del software. Finalmente se estudiarán los patrones y las métricas para validación de resultados.

#### 1.2 Conceptos Asociados

En este punto se hablará primeramente de los diferentes conceptos que se tratan en este trabajo y que permiten la familiarización con el tema que se presenta.

Un **Software** es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación. (IEEE, 1993)

**Multimedia** es un sistema que utiliza más de un medio de comunicación al mismo tiempo en la presentación de la información, como el texto, la imagen, la animación, el vídeo y el sonido. (Palomo, 2007)

Una **Multimedia Educativa** representa el conjunto de materiales didácticos de tipo multimedia que orientan y regulan el aprendizaje de los estudiantes pues, explícita o implícitamente, promueven determinadas actuaciones de los mismos encaminadas a este fin. (Palomo, 2007)

**Logopedia** es la ciencia que evalúa, diagnostica y trata los problemas del lenguaje y la voz. Su finalidad es la prevención, el diagnóstico, el pronóstico, el tratamiento y la evaluación integral de los trastornos de la comunicación humana, ya sea que pertenezcan al ámbito del habla o del lenguaje. (Fleitas, 2001)

Los **fonemas** son cada uno de los sonidos simples del lenguaje hablado. (Fleitas, 2001)

#### 1.3 ¿Por qué la Enseñanza Preescolar?

En la actualidad hay una gran cantidad de niños que presentan problemas del lenguaje. Un estudio profundo de documentos publicados por doctores en la especialidad así como entrevistas realizadas demuestran que las alteraciones del habla más frecuentes están presentes en los primeros años de vida del niño.

En el Tercer Congreso Internacional de Logopedia y Foniatría la licenciada Sandra Sánchez León presentó el trabajo titulado: Incidencias de los trastornos del lenguaje en los círculos infantiles de un área de salud del municipio Playa, en el que realizó un pesquisaje que permitió conocer que es en 5to año de vida donde hay más niños con alteraciones del lenguaje, el área más afectada es la del habla y las afectaciones orales se dan en los cuatro niveles articulatorios.

A continuación se muestra la explicación de los 4 niveles de fonemas más afectados en la actualidad.

- Fonemas del 1er nivel. Se encuentra situado entre labios y dientes y los sonidos que pertenecen a él son (P, B, M, F).
- Fonemas del 2do nivel. Situado entre el borde inferior de los incisivos superiores y el límite de la cara interna de la encía superior, donde se puede extender ésta en 1 ó 2 centímetros; los sonidos son (L, N, S, T, D, R).
- Fonemas del 3er nivel. Corresponde a la zona que bordea los límites entre el tercio anterior y el tercio medio de la bóveda palatina y en ella están incluidos los sonidos (Ñ, H, Y).
- Fonemas del 4to nivel. Tomando la base de la lengua, el velo del paladar con sus pilares y la pared faríngea, donde están comprendidos los sonidos (K, G, J).

La detección temprana de estas dificultades permite que se comience la rehabilitación de las mismas y que muchos de estos niños lleguen a la enseñanza primaria con una correcta comunicación oral.

Teniendo en cuenta estos estudios se decidió dirigir este software a niños de la enseñanza preescolar pues en esta etapa el niño ya tiene un poder de asimilación elevado debido a que ha adquirido los conocimientos básicos en el seno de la familia y aún está en el proceso de forjamiento de su personalidad. Además en el paso por la enseñanza preescolar habrá desarrollado una imagen positiva de sí mismo, las normas de relación y comportamiento basadas en la equidad y el respeto, la capacidad de asumir distintos roles a través de juegos, y medios de enseñanzas, la confianza para expresar, dialogar y conversar, habilidades en el proceso de lecto-escritura, la capacidad de observar y describir fenómenos naturales, conocimientos basados en situaciones de experimentación.

### **1.4 Análisis de las Soluciones Existentes**

Existe poca documentación acerca de este tema tanto en el país como internacionalmente. En Cuba hasta el momento no existe ningún software educativo con tecnología multimedia para tratar los problemas del lenguaje. Aunque se conoce de la existencia de manuales de actividades, así como el Visual Voz (por la Dra. Xiomara Rodríguez Fleitas) que son alternativas pedagógicas para el trabajo con la pronunciación en niños con trastornos del habla.

La necesidad de realizar el análisis y diseño de un software educativo con tecnología multimedia como este es imperiosa, pues luego de su implementación se le daría a los niños la posibilidad de hacer uso de las tecnologías informáticas y sería de gran utilidad para la prevención o disminución de los problemas del lenguaje. Además de servir como apoyo a los profesores que trabajan directamente con estos niños.

Este trabajo consiste en la propuesta de un grupo de actividades con el fin de ser utilizadas por el personal adecuado que atiende directamente a los niños con desviaciones potenciales del lenguaje, ofrece la posibilidad de interactuar desde el escenario escolar con el uso de la computadora como medio de enseñanza y herramienta de trabajo. Este conjunto de actividades diseñadas en multimedia se ajustarán a las necesidades y motivaciones de los niños con problemas del lenguaje teniendo en cuenta la diversidad de ejercicios para prevenir estos trastornos y haciendo un análisis profundo a partir de su diagnóstico inicial.

## **1.5 Metodologías de Desarrollo de Software**

Las metodologías de desarrollo de software constituyen un factor importante para lograr productos con calidad en el tiempo requerido. Actualmente existen múltiples metodologías, todas con muchas características comunes, pero que comparten también significativas diferencias, por lo que se recomienda hacer un estudio de estas para realizar una correcta selección, garantizando así el éxito del proyecto de software.

Las ventajas de tener una metodología son entre otras: (Gómez, 2006)

Mejora de los procesos de desarrollo.

Actividades de desarrollo apoyadas por procedimientos y guías.

Resultados del desarrollo predecibles

El uso de las metodologías permite mejorar el proceso de desarrollo, a continuación se citarán algunas de las metodologías existentes.

### **1.5.1 Rational Unified Process (RUP)**

La metodología Rational Unified Process (RUP), divide en cuatro fases el desarrollo del software:

Inicio: Determinar la visión del proyecto.

Elaboración: Determinar la arquitectura óptima.

Construcción: Llegar a obtener la capacidad operacional inicial.

Transición: Llegar a obtener el *release* del proyecto.

Vale mencionar que el ciclo de vida que se desarrolla por cada iteración, es llevado bajo dos disciplinas:

## Disciplina de Desarrollo

Modelación del negocio: Tiene como objetivo comprender las necesidades del negocio.

Requisitos: Traslada las necesidades del negocio a un sistema automatizado.

Análisis y Diseño: Traslada los requisitos dentro de la arquitectura de software.

Implementación: Crea software que se ajuste a la arquitectura y que tenga el comportamiento deseado.

Pruebas: Asegura que el comportamiento requerido sea el correcto y que todo lo solicitado esté presente.

## Disciplina de Soporte

Configuración y administración de cambios: Guarda todas las versiones del proyecto.

Administración de proyecto: Administra horarios y recursos.

Ambiente: Administra el ambiente de desarrollo.

Despliegue: Hace todo lo necesario para la distribución del proyecto a los usuarios finales.

Los elementos del RUP son:

Actividades: Procesos que se llegan a determinar en cada iteración.

Trabajadores: Las personas involucradas en cada proceso.

Artefactos: Puede ser un documento, un modelo, o un elemento de modelo.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

### **1.5.2 Extreme Programing (XP)**

La programación extrema (**XP**) o *eXtreme Programming*, es una metodología ágil utilizada en el desarrollo de software. Su filosofía es satisfacer al completo las necesidades del usuario, por eso, lo integra como una parte más del equipo de desarrollo. Está diseñada para el desarrollo de aplicaciones de corto plazo que requieren un grupo de programadores pequeño, donde la comunicación sea más factible. La comunicación es un punto importante y debe realizarse entre los programadores, los jefes de proyecto y los clientes. (Barbone, 2006)



XP, se basa en hacer pruebas unitarias para detectar las fallas que pudieran ocurrir, en la reutilización de código, es decir, en la refabricación y en la programación en pares la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Lo fundamental en este tipo de metodología es:

**Comunicación:** Los programadores están en constante comunicación con los clientes para satisfacer sus requisitos y responder rápidamente a los cambios de los mismos.

**Simplicidad:** Codificación y diseños simples y claros.

**Realimentación (Feedback):** Mediante la realimentación se ofrece al usuario la posibilidad de conseguir un sistema apto a sus necesidades ya que se le va mostrando el proyecto a tiempo para poder ser cambiado y poder retroceder a una fase anterior para rediseñarlo a su gusto.

**Coraje:** Se debe tener coraje o valentía para cumplir los tres puntos anteriores; hay que tener valor para comunicarse con el usuario y enfatizar algunos puntos, a pesar de que esto pueda dar sensación de ignorancia por parte del programador, hay que tener coraje para mantener un diseño simple y no optar por el camino más fácil y por último hay que confiar en que la realimentación será efectiva.

### **1.5.3 SCRUM**

Dentro de las metodologías ágiles se encuentra SCRUM y se enfoca fundamentalmente hacia la planeación iterativa y el seguimiento del proceso. Sus características son muy cercanas a las de XP, sin embargo presenta procesos de iteración con un período de 30 días, denominadas carreras cortas.

La metodología SCRUM se divide de forma tal que cuando se vaya a hacer una primera carrera se debe definir la funcionalidad para la misma y luego se deja al equipo de desarrolladores para que haga entrega de la primera iteración. La actividad principal en esta primera corrida es la de estabilizar los requisitos. Para ello la gerencia se da a la tarea de reunir al equipo todos los días durante 15 minutos para discutir las tareas del próximo día, haciendo que no se pierda el objetivo perseguido durante la carrera: la comunicación entre todos. Durante esta reunión se muestran todos los inconvenientes que impidan el progreso de la carrera y la gerencia debe resolver los mismos. Al igual la gerencia debe tener toda la información diaria que se genera como una forma de mantener la actualización diaria del proyecto. (Furini, 2008)

### **1.6 Lenguajes de Modelado**

El constante avance en el nivel de complejidad de las soluciones informáticas ha hecho de la utilización de los modelos, un mecanismo para facilitar la comprensión de estas. A continuación se citarán algunos de los lenguajes de modelado existentes.

## 1.6.1 Lenguaje Unificado de Modelado (UML)

UML es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje estándar para escribir planos de software. UML puede utilizarse para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra gran cantidad de software.

UML se puede aplicar en una gran variedad de formas para soportar una metodología de desarrollo de software que no especifica en sí mismo qué metodología o proceso usar. Hoy en día, UML está consolidado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo. Mediante UML es posible establecer una serie de requerimientos y estructuras necesarias para plasmar un sistema de software previo al proceso intensivo de escribir código. (Latina, 2007)

UML ayuda al usuario a entender la realidad de la tecnología y la posibilidad de que reflexione antes de invertir y gastar grandes cantidades en proyectos que no estén seguros en su desarrollo, reduciendo el coste y el tiempo empleado en la construcción de las piezas que constituirán el modelo.

## 1.6.2 Notación para el Modelado de Procesos del Negocio (BPMN)

BPMN es un estándar cuyo principal objetivo es proporcionar una notación fácilmente comprensible por todos los usuarios del negocio, desde los analistas, los desarrolladores técnicos, hasta aquellos que monitorizarán y gestionarán los procesos. Es importante tener en cuenta que BPMN abarca únicamente los procesos de negocio, lo que significa que otro tipo de modelos relacionados (estructura de la organización, recursos, modelos de datos, estrategias, reglas de negocio) quedan fuera de la especificación. (Pérez, 2007)

## 1.7 Herramientas de Modelado.

Las herramientas CASE (Computer Aided Software Engineering) tienen como objetivo primordial: (Giraldo, 2003)

- Mejorar la productividad en el desarrollo y mantenimiento del software.
- Aumentar la calidad del software.
- Mejorar el tiempo y coste de desarrollo y mantenimiento de los sistemas informáticos.
- Mejorar la planificación de un proyecto
- Aumentar la biblioteca de conocimiento informático de una empresa ayudando a la búsqueda de soluciones para los requisitos.
- Automatizar, desarrollar el software, generar documentación y código, realizar pruebas de errores.

- Ayudar a la reutilización del software, portabilidad y estandarización de la documentación.
- Gestionar todas las fases de desarrollo de software con una misma herramienta.
- Facilitar el uso de las distintas metodologías propias de la ingeniería del software.

Dentro de las herramientas CASE se pueden mencionar entre otras: Rational Rose, Enterprise Architect y Visual Paradigm.

## 1.7.1 Visual Paradigm

Visual Paradigm es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software. Facilita el dibujo de todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación, así como una serie de tutoriales con demostraciones interactivas y proyectos.

Es importante destacar que el Visual Paradigm como herramienta de modelado posee características como (Sixto, 2003)

- Posibilita la calidad del Producto
- Soporta aplicaciones Web.
- Las imágenes y reportes generados son de muy buena calidad.
- Varios idiomas.
- Generación de código para Java y exportación como HTML.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad; disponibilidad de integrarse en los principales IDEs (Integrated Development Environment).
- Disponibilidad en múltiples plataformas; ofrece un mecanismo general para la organización de los modelos/subsistemas/capas agrupando elementos de modelado y versión gratuita.

### 1.7.2 Rational Rose

Rational Rose es una herramienta de modelación visual que provee el modelado basado en UML. Rose es una herramienta con plataforma independiente que ayuda a la comunicación entre los miembros de equipo, a monitorear el tiempo de desarrollo y a entender el entorno de los sistemas. Una de las grandes ventajas de Rose es que utiliza la notación estándar en la arquitectura de software (UML), la cual permite a los arquitectos de software y desarrolladores visualizar el sistema completo utilizando un lenguaje común, además los diseñadores pueden modelar sus componentes e interfaces en forma individual y luego unirlos con otros componentes del proyecto. (GSI, 2007)

### 1.7.3 Enterprise Architect

Enterprise Architect (EA) es una herramienta que cubre todo el proceso de desarrollo de software. EA es una herramienta multi-usuario, desarrollada para Windows, diseñada para ayudar a construir software robusto y fácil de mantener. Ofrece salida de documentación flexible y de alta calidad.

Esta herramienta además de tener muchas ventajas como rapidez, buena interfaz gráfica, infinidad de funciones, entre otras, también tiene una serie de desventajas como son: el control de versionamiento es muy lento, es complicado hacer combinaciones entre dos importaciones con elementos en común y poca estética a la hora de hacer el modelo entidad relación.

## 1.8 Herramientas Propuestas para la Implementación

En el avance de la tecnología han surgido varias herramientas para desarrollar productos multimedia. A continuación se exponen algunas de estas haciendo énfasis en sus características.

### 1.8.1 Adobe Director

Esta herramienta es un potente ambiente de composición multimedia para construir contenidos y aplicaciones de alta capacidad, enriquecidas e interactivas. Es un programa de autor de fácil manejo. Permite la combinación de texto, gráficos, sonido, animación y vídeo en un documento que se reproduce en el ordenador y que es presentado con múltiples detalles. La filosofía seguida por este programa es la de una línea de tiempo durante el cual irán sucediendo diferentes acontecimientos según se vayan necesitando. Este proceso no tiene por qué ser necesariamente lineal ni continuo sino que permite detenerse en el punto de tiempo y saltar de un punto a otro en esa línea temporal. (Felipe, 2008)

### 1.8.2 ToolBook

ToolBook brinda un ambiente de programación orientado a objeto para de esta manera construir proyectos, con el fin de presentar gráficamente información como textos, animaciones, sonidos, dibujos e imágenes digitalizadas a color. Además ofrece interfaces gráficas al estilo Windows. Presenta dos niveles de trabajo: el lector y el autor. Se ejecutan los guiones a nivel de lector. A nivel de autor se utilizan órdenes para crear nuevos libros, crear y modificar objetivos en las páginas y escribir guiones. ToolBook ofrece opciones de vinculación para botones y palabras claves, de forma que se pueda crear guiones de navegación identificando la página a la que debe ir. (Rodríguez, 2006)

### 1.8.3 Adobe Flash CS3

Flash CS3 es una potente herramienta que ha superado las mejores expectativas de sus creadores. Inicialmente Flash fue creado con el objeto de realizar animaciones vistosas para la web, así como para crear GIFs animados. Los motivos que han convertido a Flash en el programa elegido por la mayoría de los diseñadores web profesionales y aficionados son varios. Las posibilidades de Flash son extraordinarias, cada nueva versión ha mejorado a la anterior, y el actual Flash CS3 no ha sido menos (Aulacllic, 2008)

Flash CS3 presenta mejoras en cuanto a facilidad de manejo, mayor potencia gráfica y de integración con programas de edición de imágenes, facilidad para importar vídeo, posibilidad de emular las películas dirigidas a dispositivos móviles y para los menos avanzados, se recupera el asistente de ActionScript 3.0 que había desaparecido. A continuación se describen estas ventajas y otras muchas con un poco más de detalle.

Integración total con archivos de Photoshop: Flash CS3 ofrece una compatibilidad perfecta a la hora de importar archivos creados en Photoshop que permite modificar y utilizar sus capas y estilos muy fácilmente.

Interfaz mejorada: Flash CS3 cambia su interfaz para integrarse completamente en la suite de productos Adobe, todas las ventanas se encuentran en paneles laterales completamente configurables. Se pueden mostrar las ventanas de un modo expandido o minimizado.

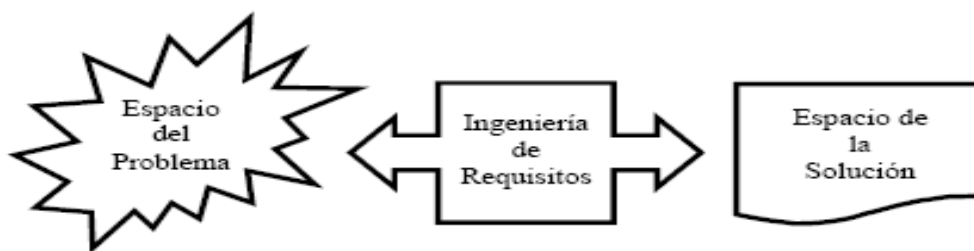
Flash también permite guardar y cargar diferentes tipos de configuración de paneles. Así se puede tener una vista preferida guardada y accesible si en algún momento el espacio de trabajo se vuelve demasiado caótico.

Componentes mejorados: Flash CS3 permite una mayor configuración en los componentes preinstalados. De esta forma se puede cambiar su aspecto. Esta opción permite crear formularios más acorde con los diseños.

Vídeo más potente: Esta versión de Flash introduce un nuevo compresor de vídeo mejorado que aumenta la calidad y disminuye el tamaño del archivo final SWF. Además se introduce el soporte de capas con transparencia que podrán ser tratadas e importadas sin ningún tipo de problema.

## 1.9 Ingeniería de Requisitos

La Ingeniería de Requisitos (IR) debe ser considerada como un proceso de construcción de una especificación de requisitos en el que se avanza desde unas especificaciones iniciales, que no poseen las características oportunas, hasta especificaciones finales completas, formales y acordadas entre todas las partes (Pressman, 2005)



**Figura 1. La Ingeniería de Requisitos como puente entre los espacios del problema y la solución**

El proceso de desarrollo de la IR se divide en cinco etapas:

- Elicitación de los requisitos
- Análisis de los requisitos
- Especificación de los requisitos
- Validación de los requisitos
- Gestión o Administración de los requisitos

### Elicitación

Actividad de la IR en la cual se estudia el dominio del problema y se interactúa con los clientes y usuarios para obtener y registrar información sobre sus necesidades. En esta etapa se debe recolectar toda la información posible y necesaria a partir de la cual se identifican los primeros requisitos candidatos del sistema a desarrollar.

### Análisis

Actividad de la IR en la cual se estudia la información extraída en la etapa previa para identificar la presencia de áreas no detectadas, requisitos contradictorios y peticiones que aparecen como vagas e irrelevantes. En esta etapa se clasifican y analizan los requisitos encontrados en la primera

aproximación y se negocian con el cliente para verificar los puntos de acuerdo y entendimiento de sus necesidades.

## **Especificación**

En esta etapa se realiza una descripción formal de los requisitos que finalmente el sistema va a cumplir.

## **Validación**

Actividad de la IR en la que clientes y usuarios, junto con la ayuda de los ingenieros de requisitos y otros evaluadores, revisan los productos obtenidos en etapas anteriores para comprobar que realmente reflejen sus necesidades, que definen el producto deseado.

## **Gestión o Administración**

Con esta actividad se pretende llevar un control sobre los cambios que pueden sufrir los requisitos debido a que no se hayan hecho las preguntas correctas a los usuarios, haya cambiado el problema que se estaba resolviendo, o simplemente cambiaron las expectativas de los clientes.

### **1.9.1 Técnicas para el Levantamiento de Requisitos**

A lo largo de la historia se han desarrollado algunas técnicas para el levantamiento de requisitos. Éstas son manejadas por un equipo de clientes y desarrolladores que trabajan para comprender el problema, proponer soluciones, negociar diferentes perspectivas o puntos de vista y especificar un conjunto básico de requisitos de la solución. A continuación se citan algunas de estas técnicas.

Entrevista: Es usada con frecuencia para acercarse al cliente y al desarrollador, pues logra una muy buena comunicación entre ambas partes. Es muy aceptada y ampliamente extendida. Permite obtener información sobre el problema en cuestión y comprender los objetivos para su solución. Para realizarla es necesario seleccionar correctamente a los entrevistados, definir previamente las preguntas, pues la forma de redacción puede influir en las respuestas y por último, analizar los resultados. (Koch, 2002)

Tormenta de ideas: Es una técnica de reuniones que se usa para generar ideas. Su objetivo principal es concebir la mayor cantidad de requisitos para el software. Reúne integrantes de varias disciplinas, mientras mayor experiencia y preparación tengan, mayor posibilidad se presenta de generar buenas ideas. En este tipo de reunión no se debe socavar ideas que parezcan locas, ni evaluar o criticar las ideas de los demás, porque puede producir un ambiente hostil que perjudique el dinamismo de la reunión, además de que esa idea puede constituir un gran aporte luego de ser madurada y perfeccionada o relacionada con otras. (Zapata, 2004)

Mapas conceptuales: Es una técnica muy eficiente para obtener una idea del negocio, las relaciones que presenta y el vocabulario, incluso puede ayudar a aportar términos al glosario. Consiste en la representación de conceptos sobre el problema donde se muestran sus asociaciones y atributos mediante un grafo, donde los conceptos son los vértices, y las relaciones son las aristas. Esta técnica es muy usada en la IR, pues es fácil de entender por parte del usuario o cliente, aunque puede llegar a ser ambigua si no se usa correctamente.

Arqueología de documentos: Se inspecciona la documentación de la empresa con el fin de familiarizarse más con el negocio y poder identificar posibles requisitos. (Dávila, 2001)

Introspección: Este es el método más obvio que se utiliza para entender las necesidades del cliente. Recomienda que el entrevistador se ponga en el lugar del cliente y trate de imaginar cómo desearía el sistema, si fuera para él. Y en base a estas suposiciones comenzar a recomendar al cliente sobre la funcionalidad que debería presentar el software.

## **1.10 Patrones de Casos de Uso**

La experiencia en la utilización de casos de uso ha evolucionado en un conjunto de patrones que permiten con más precisión reflejar los requisitos reales, haciendo más fácil el trabajo con los sistemas, y mucho más simple su mantenimiento. Dado un contexto y un problema a resolver, estas técnicas han mostrado ser la solución adoptada en la comunidad del desarrollo de software. Se presentan a modo de herramientas que permiten resolver los problemas que se les planteen a los desarrolladores de una forma ágil y sistemática.

Los patrones de casos de uso son los siguientes: (Colectivo de autores, 2008)

- Reglas de negocio
- Concordancia (Commonality)
- Componente jerárquico (Component hierarchy)
- CRUD (Creating, Reading, Updating, Deleting)
- Caso de uso grande (Large Use case)
- Sistema de Capas
- Múltiples actores
- Servicio opcional
- Vistas ortogonales
- Secuencia de casos de uso.



A continuación se explicarán algunos de estos patrones.

## Reglas de Negocio

Se basan en la extracción de información originada de las políticas, reglas y regulaciones del negocio de la descripción del flujo y describe la información como una colección de reglas del negocio referenciadas a partir de las descripciones de los casos de uso.

### *Definición estática*

Este patrón es aplicado a todos los casos de uso modelando los servicios que son afectados por las reglas del negocio definidas en la organización. Sin embargo, este patrón no influye en la estructura del modelo de casos de uso. Las reglas son descritas en un documento separado, referenciadas por las descripciones de los casos de usos relevantes. Este patrón es apropiado utilizarlo cuando no hay necesidad de cambiar dinámicamente las reglas del negocio mientras el sistema se esté utilizando.

### *Modificación dinámica*

Este modelo del patrón contiene un caso de uso llamado Gestionar regla, que se encarga de crear, actualizar y eliminar las reglas del negocio. Este patrón es útil cuando la colección de reglas sea modificada dinámicamente, o sea, estas pueden ser modificadas mientras el sistema esté corriendo.

## Concordancia (Commonality)

Extrae una subsecuencia de acciones que aparecen en diferentes lugares del flujo de casos de uso y es expresado por separado.

### *Reuso*

Consta de 3 casos de uso. El primero llamado subsecuencia común, modela una secuencia de acciones que aparecerán en múltiples casos de uso en el modelo. Los otros casos de uso modelan el uso del sistema que comparte la subsecuencia común de acciones. De manera que deben existir al menos dos de ellos.

### *Adición*

En el caso de este patrón alternativo, la subsecuencia común de casos de uso, extiende los casos de uso compartiendo la subsecuencia de acciones. Los otros casos de uso modelan el flujo que será expandido con la subsecuencia. Este patrón es preferible usarlo cuando otros casos de uso se encuentran propiamente completos, o sea, que no requieren de una subsecuencia común de acciones para modelar los usos completos del sistema.

### *Especialización*

Otro patrón de concordancia que contiene casos de uso del mismo tipo. En este caso, estos son modelados como una especialización de casos de uso de tipo de uso común. Todas las acciones en estos casos de uso son heredadas por los casos de uso hijos, donde otras acciones serán adicionadas o acciones heredadas que serán especializadas. Este patrón es aplicable cuando la utilización de los casos de uso que han sido modelados son del mismo tipo, y este tipo debe hacerse visible en el modelo.

### *Reuso interno*

Si la subsecuencia de acciones es utilizada en diferentes lugares en un solo caso de uso, no existe la necesidad de extraer la subsecuencia dentro de un caso de uso separado. Además este debe ser descrito en una sub-sección separada en la descripción del caso de uso. Esta sub-sección será referenciada desde diferentes partes en la descripción del caso de uso donde las subsecuencias de acciones sean realizadas. Este patrón se utiliza cuando la subsecuencia común aparece en múltiples lugares en un mismo caso de uso.

### CRUD (Creating, Reading, Updating, Deleting)

Este patrón se basa en la fusión de casos de uso simples para formar una unidad conceptual.

### *CRUD Completo*

Este patrón consta de un caso de uso, llamado Información CRUD o Gestionar información modela todas las operaciones que pueden ser realizadas sobre una parte de la información de un tipo específico, tales como creación, lectura, actualización y eliminación. Suele ser utilizado cuando todos los flujos contribuyen al mismo valor del negocio, y estos a su vez son cortos y simples.

### *CRUD Parcial*

Este patrón alternativo modela una de las vías de los casos de uso como un caso de uso separado. Es preferiblemente utilizado cuando una de las alternativas de los casos de uso es más significativa, larga o más compleja que las otras.

### Múltiples actores

#### *Roles diferentes*

Captura la concordancia entre actores manteniendo roles separados. Consiste en un caso de uso y por lo menos dos actores. Es utilizado cuando dos actores juegan diferentes roles en un caso de uso, o sea, interactúan de forma diferente con el caso de uso.

#### *Roles comunes*

Puede suceder que los dos actores jueguen el mismo rol sobre el CU. Este rol es representado por otro actor, heredado por los actores que comparten este rol. Es aplicable cuando, desde el punto de vista del caso de uso, solo exista una entidad externa interactuando con cada una de las instancias del caso de uso.

### Sistema de capas

Estructura el modelo de casos de uso de manera tal que cada caso de uso es definido dentro de una capa, y utiliza relaciones entre los casos de uso en diferentes capas que permitan instanciar los casos de uso para expandir múltiples capas.

En cada uno de los patrones del sistema de capas, existen dos paquetes modelando dos capas, un paquete importa las relaciones del paquete que representa las capas superiores para el paquete que representa las capas inferiores. La relación implica que el contenido (público) del paquete de la capa inferior se torne disponible dentro del paquete de la capa superior. Todas las relaciones entre los elementos determinados en diferentes paquetes de capas, son definidas dentro del paquete de capas superior.

### *Reuso*

Contiene dos casos de uso y dos paquetes con un paquete que importa las relaciones entre ellos. El caso de uso definido en la capa superior tiene una relación de inclusión con el caso de uso que es definido en la capa inferior y que es importado hacia la capa superior. El caso de uso definido en la capa inferior es a menudo, pero no siempre, abstracto.

Este patrón es apropiado cuando las instancias de los casos de uso inician en la capa superior, pero utilizarán un servicio definido en la capa inferior. No es conveniente cuando la instancia del caso de uso inicia en la capa inferior.

### *Adición*

En este caso, el caso de uso definido en la capa superior extiende al caso de uso definido en la capa inferior. Aquí la instancia del caso de uso se inicia en la capa inferior, pero los servicios definidos en la capa superior son insertados dentro del mismo. El caso de uso en la capa superior es normalmente abstracto, o sea usualmente no es realizado como una instancia del mismo. La diferencia entre esta alternativa y la anterior está dada por la relación entre los casos de uso superiores y los inferiores. Este patrón es aplicable cuando la instancia del caso de uso inicia en la capa inferior, pero no debe ser utilizado si comienza en la capa superior.

### *Especialización*

En el sistema de capas para el patrón de especialización, el caso de uso definido en la capa superior es una especialización del caso de uso definido en la capa inferior, de manera que el anterior es una generalización del último. Este patrón es aplicable cuando el caso de uso superior es del mismo tipo que el inferior. No es conveniente utilizarlo cuando la instancia del caso de uso siga una combinación de descripciones de dos capas.

### *Embebido*

En este patrón alternativo, los casos de uso de la capa inferior no están disponibles. De otra forma el acceso a la información dentro de la capa inferior es descrito dentro de las descripciones de los casos de uso en la capa superior. Es utilizado cuando la capa superior solo realiza operaciones simples sobre la información en la capa inferior o cuando la capa inferior consta de una plataforma que no pueda ser modificada. La información de la capa inferior es considerada disponible en el sistema cuando se escriben las descripciones de los casos de uso de la capa superior.

### **1.11 Patrones de Diseño**

En términos generales, los patrones de diseño contribuyen a la construcción del software, brindan la posibilidad de conocer cómo utilizar las clases y los objetos de forma correcta. El uso de patrones es muy importante pues establece problemas Pareja-Solución, ayuda a especificar interfaces, y además permite la reutilización del Código.

Los patrones de diseño tienen como características:(Larman, 2004)

- 1 Son soluciones concretas. Proponen soluciones a problemas concretos, no son teorías genéricas.
- 2 Son soluciones técnicas. Indican resoluciones técnicas basadas en Programación Orientada a Objetos (POO). En ocasiones tienen más utilidad con algunos lenguajes de programación y en otras son aplicables a cualquier lenguaje.
- 3 Se utilizan en situaciones frecuentes debido a que se basan en la experiencia acumulada al resolver problemas reiterativos.
- 4 Favorecen la reutilización de código. Ayudan a construir software basado en la reutilización, a construir clases reutilizables. Los propios patrones se reutilizan cada vez que se vuelven a aplicar.
- 5 El uso de un patrón no se refleja en el código. Al aplicar un patrón, el código resultante no tiene por qué delatar el patrón o patrones que lo inspiró. No obstante últimamente hay múltiples esfuerzos enfocados a la construcción de herramientas de desarrollo basados en los patrones y frecuentemente se incluye en los nombres de las clases el nombre del patrón en que se basan facilitando así la comunicación entre desarrolladores.

- 6 Es difícil reutilizar la implementación de un patrón. Al aplicar un patrón aparecen clases concretas que solucionan un problema concreto y que no será aplicable a otros problemas que requieran el mismo patrón.

Los patrones de diseño están divididos en los patrones GRASP (General Responsibility Assignment Software Patterns o patrones generales de software para asignar responsabilidades) y los patrones GOF (conocidos como los patrones de la banda de los cuatro GoF, gang of four).

A continuación se citarán algunos de estos patrones de diseño. (Colectivo de autores, 2008)

### **GRASP:**

**Bajo Acoplamiento:** Este patrón es un principio que asigna la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.) El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. Un error muy común es asignarle demasiada responsabilidad y alto nivel de acoplamiento con el resto de los componentes del sistema.

**Experto:** La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase contiene toda la información necesaria para realizar la labor que tiene encomendada.

**Alta cohesión:** La cohesión es una medida de la fuerza con la que se relacionan las clases y el grado de focalización de las responsabilidades de un elemento. Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable, una clase con baja cohesión hace muchas cosas no relacionadas o hace demasiado trabajo.

**Creador:** Este patrón como su nombre lo indica es el que crea, guía la asignación de responsabilidades relacionadas con la creación de objetos, se asigna la responsabilidad de que una clase B cree un objeto de la clase A solamente cuando

B contiene a A

B es una agregación (o composición) de A

B almacena a A

B tiene los datos de inicialización de A (datos que requiere su constructor)

B usa a A.

**Controlador:** Es un evento generado por actores externos. Se asocian con operaciones del sistema, operaciones del sistema como respuestas a los eventos del sistema, tal como se relacionan los

mensajes y los métodos Normalmente un controlador delega en otros objetos el trabajo que se necesita hacer; coordina o controla la actividad. No realiza mucho trabajo por sí mismo.

### **GOF:**

Creación: Dentro del grupo de los patrones de creación hay existen algunos que abordan problemas comúnmente encontrados al momento de decidir, dónde, cómo y cuándo crear objetos.

Abstract Factory: El primero de la lista, usado para crear un conjunto de objetos que sean independientes del cliente.

Builder: Es una particularidad del Abstract Factory, lo que busca es el diseño flexible de un objeto que va a ser consumido por terceros; el objeto, dependiendo de su creación, tendrá un comportamiento diferente, pero los consumidores no deberán ser afectados por ese cambio. Además, lo que se busca es que el objeto creador esté separado de los objetos a crearse.

Factory Method: Aquí ya no se trabaja con un creador separado de la representación, son las clases derivadas las que deciden la implementación particular. A diferencia del anterior, ya no se tiene el direccionador, se trabaja con el creador directamente.

Prototype: Usado cuando es necesario trabajar con varias instancias de un mismo objeto, pero se busca además cierta independencia entre estos, Se trabaja con clonaciones de un objeto en particular.

Singleton: Cuando se quiere trabajar con solo una instancia de un objeto, este es por cierto el patrón que casi siempre el primero en ser mencionado. Un lugar donde podría ser aplicado es cuando se tienen opciones de menú, y se quiere que al hacer clic sobre el mismo, se muestre solo un formulario asociado.

**Estructurales (Composición):** Dentro de este grupo de patrones se encuentran los que sirven para resolver los problemas que se tienen al trabajar con interacción entre objetos.

Adapter (Adaptador): Adapta una interfaz para que pueda ser utilizada por una clase que de ninguna otra manera podría utilizarla.

Bridge (Puente): Desacopla una abstracción de su implementación.

Composite (Objeto compuesto): Permite tratar objetos compuestos como si de uno simple se tratara.

Decorator (Envoltorio): Añade funcionalidad a una clase dinámicamente.

Facade (Fachada): Proporciona una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema.

Flyweight (Peso ligero): Reduce la redundancia cuando gran cantidad de objetos poseen información idéntica.

Proxy: Mantiene a un representante de un objeto.

**Comportamiento**: aquí se incluyen el conjunto de patrones que serán usados cuando se tengan que resolver problemas afines a las Características que deba tener un objeto.

Chain of Responsibility (Cadena de responsabilidad): Permite establecer la línea que tienen que llevar los mensajes para que los objetos realicen la tarea indicada.

Command (Orden): Permite tratar un método como si de un objeto se tratara.

Iterator (Iterador): Permite realizar recorridos sobre objetos compuestos independientemente de la implementación de éstos.

Memento (Recuerdo): Permite volver a estados anteriores del sistema.

Strategy (Estrategia): Permite disponer de diversos métodos para resolver un problema y elegir cuál utilizar en tiempo de ejecución.

## 1.12 Principios de Diseño

En la elaboración de este Software educativo con tecnología multimedia se tienen en cuenta una serie de principios y normas de diseño. (Bauzá, 2008)

- **Principio de la múltiple entrada**: todo cuanto se puede transmitir desde una aplicación multimedia “viajará” por lo que se llaman los canales de comunicación: texto, imagen o sonido.
- **El principio multicanal** establece que para lograr una buena comunicación hay que utilizar todos los canales. Un sistema multimedia es el que transmite una información mediante imagen, sonido y texto de forma sincronizada, y que hace uso adecuado de la capacidad de usar los diferentes canales de comunicación.
- **Principio de interactividad**: es un recurso propio de sistemas informáticos y permite acceder a cualquier tipo de información rompiendo radicalmente con la linealidad o secuencialidad, con el único objetivo de reforzar el mensaje que se quiere transmitir. En una aplicación multimedia es necesario establecer niveles de interacción siempre y cuando estos no afecten el objetivo del mensaje original. Así, el usuario debe interactuar con la aplicación cuando sea estrictamente necesario. Además, se deben evitar los períodos de tiempo excesivamente prolongados en los que el usuario no interviene, como: una lectura de textos extensos en pantalla, secuencias prologadas de sonido e imagen animada.

También debe evitarse la interacción basada en la repetición de gestos por parte del usuario.

- **Principio de vitalidad:** toda pantalla debe estar viva. Es decir, el usuario debe percibir la aplicación como algo que funciona autónomamente, como un mundo al que se asoma. Con ello se va más allá del principio de interactividad: en la aplicación siempre sucede algo, aunque el usuario no haga nada.
- **Principio de libertad:** una vez que se ha logrado un diseño interactivo, donde el usuario no es un mero espectador de los acontecimientos, se ha conseguido uno de los principales objetivos de la aplicación: convertir al usuario en actor de la misma. El objetivo del diseñador de una aplicación multimedia es que el usuario piense que navega libremente, mientras que en realidad está inmerso en un esquema de etapas predeterminado.
- **Principio de atención:** el objetivo de las aplicaciones multimedia es mantener la atención sostenida, es decir, conseguir que el receptor mantenga una actitud continua de expectación ante la aplicación. Para ello se dispone de dos factores: la naturaleza misma de la aplicación y la apariencia, que generan respectivamente atención cognitiva y afectiva. Atención cognitiva es la que se basa en el valor de la información suministrada, es típica de las aplicaciones profesionales o de contenidos muy particulares. Se hace especialmente atractiva para los usuarios especializados a los cuales va dirigida, y que son capaces de percibir la importancia de la información que se transmite. Para conseguirla hace falta que la información sea relevante y esté bien organizada. La atención afectiva se basa en el lazo afectivo que se establece entre el usuario y la aplicación. Hay que señalar un recurso que contribuirá siempre a conseguir la atención afectiva: el desenlace literario. Esto consiste en que si se empieza a contar una historia se está sembrando en el receptor una inquietud por conocer el final.

Una de las normas más importantes para la gestión de calidad en los recursos multimedia es la ISO 14915 que proporciona orientaciones y recomendaciones para el diseño ergonómico del software de las interfaces de usuario multimedia. El diseño ergonómico mejora la capacidad del usuario para manejar aplicaciones multimedia con eficacia, eficiencia y satisfacción.

**La norma ISO 14915** consiste en las siguientes partes:

### **Parte 1: Principio de diseño y estructura.**

Esta primera parte establece los principios de diseño para interfaces de usuario multimedia así como la estructura para el diseño multimedia. Los principios presentados proporcionan las bases de las recomendaciones específicas para multimedia descritas en otras partes de la norma 14915. Igualmente se incluyen las recomendaciones generales para el proceso de diseño de interfaces de usuario multimedia.

### **Parte 2: Navegación multimedia y control**



La segunda parte proporciona recomendaciones para el control y la navegación en las aplicaciones multimedia. El control de los medios se refiere, especialmente, a las funciones de control de medios dinámicos tales como audio o vídeo. La navegación se refiere a la estructura conceptual de las aplicaciones multimedia y de las interacciones de usuario necesarias para moverse en esa estructura. También incluye recomendaciones para la búsqueda de material multimedia.

**Parte 3: Selección y combinación de medios:** proporciona recomendaciones para la selección de los medios con respecto a los objetivos de comunicación de la tarea, así como respecto de las características de la información. También suministra orientaciones para combinar los diferentes medios. Además, incluye recomendaciones para la integración de los componentes multimedia durante las secuencias de visión y lectura.

### 1.13 Métodos para la Validación de Resultados.

Para efectuar la evaluación de los productos y procesos de software es necesario la utilización de algunos métodos de validación, entre ellos se encuentran las métricas que abarcan un gran escenario en cuanto a medidas de software computacional se refiere. De manera general las métricas del software son las que están relacionadas con el desarrollo del software como son: complejidad, funcionalidad y eficiencia. A continuación se abordará este tema tratando algunos métodos de validación que serán de gran utilidad luego de la obtención de los resultados en posteriores capítulos.

#### 1.13.1 Técnica de Prototipado

Se conoce como prototipo a la implementación concreta de un sistema que se crea para explorar cuestiones sobre aspectos muy diversos durante el desarrollo de un sistema. Un prototipo no funcional es el diseño de la posible interfaz del software a construir, esta se utiliza para tener un mejor entendimiento del problema y validar los requisitos que el usuario solicite. (Pressman, 2005)

#### 1.13.2 Métrica de la Calidad de Especificación de los Requisitos

Para realizar la validación de los requisitos existe una lista de características que sugieren el uso de una o más métricas como son: especificidad (ausencia de ambigüedad), corrección, completión, comprensión, capacidad de verificación, consistencia externa e interna, capacidad de logro, concisión, trazabilidad, capacidad de modificación, exactitud y capacidad de reutilización. Debido a las particularidades de los requisitos capturados, se debe aplicar la métrica que mide la especificidad en los mismos, haciendo que los clientes puedan entender los requisitos de una manera fácil y se puedan probar. Para llevar a cabo este proceso se tiene que:

$$n_r = n_f + n_{nf}$$

Donde

$n_r$  representa el número de requisitos del sistema

$n_f$  es el número de requisitos funcionales

$n_{nf}$  es el número de requisitos no funcionales.

Luego se procede a medir la especificidad de los requisitos con la siguiente fórmula:

$$Q = n_{ui} / n_r$$

Donde

$n_{ui}$  es el número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas.

El valor de Q a medida que se acerca a 1, se va disminuyendo la ambigüedad de la especificación.

### 1.13.3 Heurística basada en NOAS/NOS

Esta heurística se basa en la idea de que un caso de uso sirve básicamente para expresar una interacción actor–sistema. Por ello, el número de pasos del actor deben estar en torno al 50% en cuanto al número de pasos de sistema, considerando también la posibilidad de que existan pasos de inclusión o extensión en los que se realice otro caso de uso.

Las situaciones que llevan a esta métrica fuera del rango habitual son:

1. El hecho de obviar la participación del sistema, por lo que el caso de uso resulta incompleto. Es la situación más habitual.
2. El hecho de haber desglosado demasiado las acciones de un actor determinado. En este caso aparecen varios pasos seguidos del mismo actor, lo cual se podría haber evitado uniéndolos en uno solo, separando las acciones por comas en el texto del paso.
3. El hecho de incluir interacciones de actores con el entorno del sistema o con otros actores. Este hecho, que en principio no puede considerarse un defecto, sino más bien una información interesante aunque no esencial del caso de uso, será un defecto cuando se produzca con excesiva frecuencia.

### 1.13.4 Métricas Propuestas por Lorenz y Kidd (Tamaño de Clases)

En el libro de métricas realizado por Lorenz y Kidd 0.0 “Laranjeira '90”, dividen las métricas basadas en clases en cuatro categorías: tamaño, herencia, valores internos y valores externos. Las métricas

orientadas a tamaños para una clase se centran en cálculos de atributos y de operaciones para una clase individual, y promedian los valores para el sistema.

Tamaño de Clase (TC). El tamaño general de una clase se puede determinar empleando las medidas siguientes:

1. El número total de operaciones (tanto operaciones heredadas como privadas de la instancia) que están encapsuladas dentro de la clase.
2. El número de atributos (tanto atributos heredados como atributos privados de la Instancia) que están encapsulados en la clase.

Si existen valores grandes de TC estos mostrarán que una clase puede tener demasiada responsabilidad, lo cual reducirá la reutilizabilidad de la clase y complicará la implementación y la comprobación, por otra parte cuanto menor sea el valor medio para el tamaño, más probable es que las clases existentes dentro del sistema se puedan reutilizar ampliamente.

### **1.14 Conclusiones**

Basándose en el profundo estudio realizado se seleccionó como metodología de desarrollo RUP por ser una metodología robusta que cubre todo el proceso de desarrollo del software y genera gran cantidad de artefactos, por otra parte proporciona mejor comprensión para futuras implementaciones, quedando documentado todo el desarrollo del software hasta el análisis y diseño. Como herramienta de modelado se optó por Visual Paradigm por ser multiplataforma y las tendencias que hay en la actualidad de migrar a software libre, además de poseerse la licencia para su uso. Como lenguaje de modelado se seleccionó el UML que hoy se conoce como el lenguaje estándar para el modelado en cuanto a sistemas de cómputo se refiere, además de ser fácil de aprender y permite una comunicación fluida entre los desarrolladores de software. Finalmente se propone para la futura implementación del software utilizar Adobe Flash CS3 como herramienta de implementación que incluye el lenguaje de programación ActionScript 3.0 y es considerada una de las herramientas de las más potentes y fácil de usar, además en la actualidad es la que más se utiliza en el desarrollo de productos multimedia por su capacidad de reducir las animaciones a la mínima expresión en cuanto al espacio e incorpora potentes efectos de fácil uso. En este capítulo también se realizó un estudio a cerca de los patrones de casos de uso y de diseño que permitirá la utilización de estos en capítulos posteriores. Vale destacar además, el estudio realizado acerca de las métricas y otros métodos de validación que posibilitarán evaluar la calidad del trabajo realizado en el último capítulo de este trabajo.

### **CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA.**

#### **2.1 Introducción**

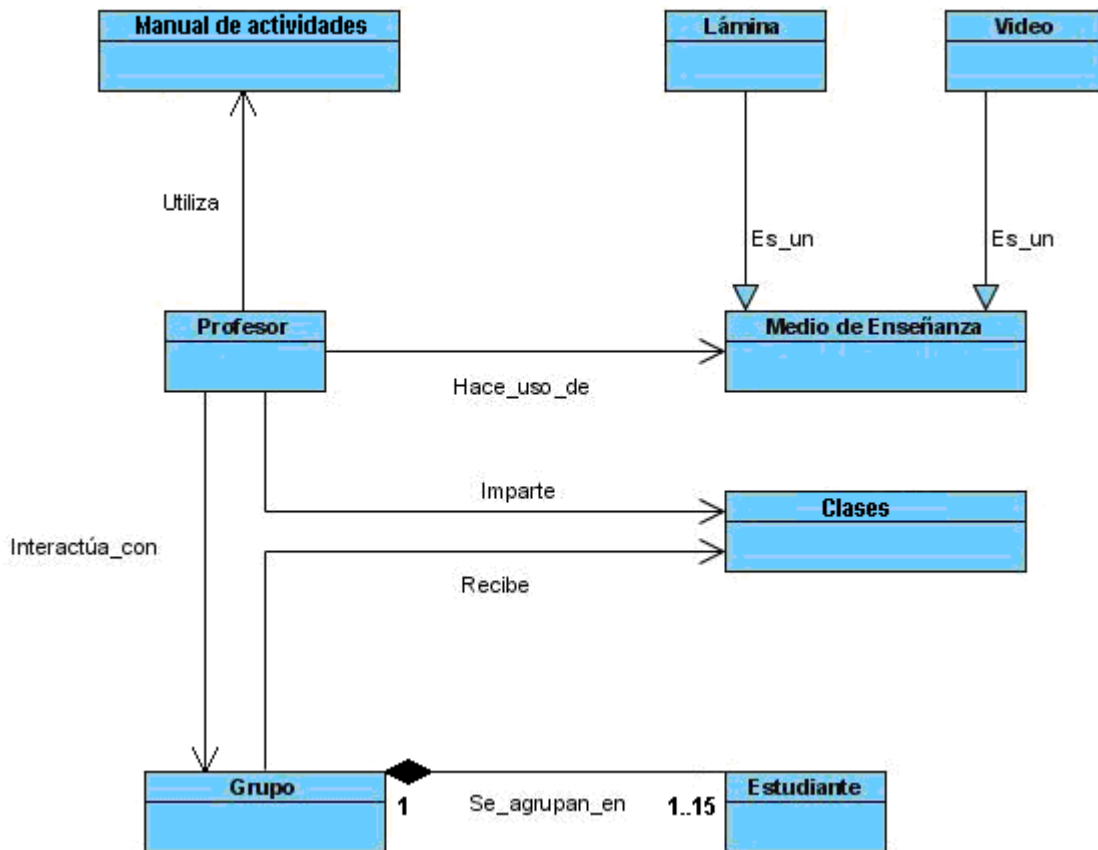
En este capítulo se realiza un modelo de dominio que permite relacionar las clases conceptuales que interactúan en el sistema, también se realiza una descripción de los conceptos asociados y se especifican los requerimientos funcionales y no funcionales que debe cumplir la aplicación. Estos requisitos se encuentran implícitos en los diferentes casos de uso del sistema, e interactúan con los actores, conformando el DCUS (diagrama de casos de uso del sistema). Finalmente se realizará una descripción detallada de cada caso de uso.

#### **2.2 Modelo de Dominio**

El modelo de dominio, constituye una representación visual para el usuario de los conceptos más significativos para un problema o área de interés. Representa conceptos del mundo real, no de los componentes del software. A diferencia de los otros modelos propuestos por la metodología RUP, este no está formado por un concepto de diagramas que describen clases u objetos de software con responsabilidades, sino que pueden considerarse como un diccionario visual de las abstracciones más relevantes en el dominio de definición del producto. (Colectivo de autores, 2008)

En la actualidad los niños que presentan problemas logopedas se agrupan en un conjunto de 15 estudiantes aproximadamente y el profesor que interactúa con cada uno de ellos realiza ejercicios específicos según su dificultad, para esto utiliza manuales de actividades y medios de enseñanza como láminas y videos que contribuyen al desarrollo del proceso docente en las escuelas. Para una mayor comprensión se realiza un diagrama de clases UML que describa las relaciones de las principales clases conceptuales que interactúan. Se decide realizar el modelo de dominio debido a que no pueden ser identificados los procesos del negocio asociados al contexto del sistema, como consecuencia de poca estructuración y fronteras de los mismos. Además los conceptos que serán expuestos representan objetos del mundo real a los cuales el sistema debe darles un estricto seguimiento a su proceso de desarrollo.

Junto al modelo de dominio se agrupan algunos conceptos con el objetivo de ayudar a los usuarios, clientes y desarrolladores a conocer algunos términos que se utilizan en el diagrama.



**Figura 2. Modelo de Dominio**

### Análisis de los conceptos del dominio.

A continuación se muestran los conceptos de las clases que interactúan en el modelo de dominio.

- Se denomina Grupo al conjunto de estudiantes.
- Se le denomina Profesor a la persona que imparte clases a los estudiantes.
- Se denomina Estudiante a la persona que recibe clases.
- Se denomina Clases a las asignaturas impartidas por un profesor a los estudiantes.
- Se denomina Video a la media de tipo video que se utiliza como medio de enseñanza.
- Se denomina Lámina a la media de tipo gráfico o imagen que se utiliza como medio de enseñanza.
- Se denomina Medio de Enseñanza a los medios que se utilizan para enriquecer una clase.
- Se denomina Manual de Actividades al conjunto de descripciones de actividades recogidas en un documento.

### 2.3 Descripción del Sistema Propuesto

La solución propuesta es la elaboración de un producto multimedia compuesto por cinco secciones, éstas son los distintos ejercicios en dependencia del tema a tratar.

#### 2.3.1 Especificación del Contenido

A modo general este software contendrá una serie de actividades evaluativas para niños de la enseñanza preescolar en sus primeras cuatro secciones, la quinta y última sección será sobre los conocimientos adquiridos en las secciones anteriores. Las secciones se conforman por ejercicios de distintos contenidos, como son:

- Sección 1: Fonemas del 1er nivel (P, B, M, F)
- Sección 2: Fonemas del 2do nivel (L,N,S,T,D,R)
- Sección 3: Fonemas del 3er nivel (Ñ, H, Y)
- Sección 4: Fonemas del 4to nivel (K, G, J)
- Sección 5: Adivinanzas y Rimas.

Cada sección contiene un grupo de actividades para tratar de forma separada los problemas de pronunciación del niño. En cada una de ellas se tratarán los ejercicios específicos según el nivel y el tema seleccionado. (Ver Anexo 1)

Para presentar la información no se muestran demasiados objetos en la pantalla, y los que existen están bien distribuidos. Cada elemento visual influye en el usuario no sólo por sí mismo, sino también por su combinación con el resto de los elementos presentes en la pantalla.

Los colores utilizados no son solo decorativos pues comunican información. Se utilizan combinaciones adecuadas para que el color atraiga la atención y no cansa a la persona que interactúa con la aplicación después de un largo rato de trabajo.

### 2.4 Descripción de la Funcionalidad

#### 2.4.1 Especificación de los Requisitos del software

El objetivo fundamental de este flujo de trabajo es guiar el desarrollo hacia el sistema correcto. Siempre se hace necesario especificar los requisitos en un lenguaje que pueda entender tanto el cliente como los desarrolladores. Los requisitos se pueden clasificar en funcionales y no funcionales. La especificación de los requisitos funcionales debe ser lo más precisa posible, esta tarea juega un papel muy importante en la etapa de elaboración del software pues da la posibilidad de conocer lo que

se desea producir, estableciendo con mayor claridad el comportamiento del futuro software y minimizando los problemas derivados en su desarrollo.

### 2.4.2 Captura de Requisitos Funcionales

Los Requisitos Funcionales son capacidades o condiciones que el sistema debe cumplir. Para la captura de los requisitos funcionales se debe preguntar ¿Qué debe hacer el sistema? (Colectivo de autores, 2008) A continuación se muestran los requisitos funcionales.

RF 1. Cargar presentación general del software.

RF 2. Permitir ir a la pantalla Inicio desde cualquier pantalla del software.

RF 3. Mostrar los temas que conforman el software (Índice).

RF 4. Permitir salir del software en cualquier momento.

RF 5. Mostrar mensaje de ayuda en cualquier pantalla que el usuario desee.

RF 6. Permitir al usuario seleccionar un ejercicio específico de algún tema

RF 7. Cargar un ejercicio seleccionado.

RF 8. Mostrar animación de letras correspondiente a cada ejercicio

RF 9. Mostrar imágenes correspondientes a cada ejercicio.

RF 10. Reproducir grabación correspondiente a cada ejercicio.

RF 11. Permitir al usuario reproducir, detener y pausar la grabación correspondiente al ejercicio

### 2.4.3 Requisitos no Funcionales

Los requisitos no funcionales son cualidades que debe poseer el producto y el medio donde se usará. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. (Colectivo de autores, 2008) A continuación se muestran los requisitos no funcionales.

#### **Apariencia o interfaz externa**

1 El producto multimedia se ejecutará en pantalla completa.

2 El producto tendrá una pantalla de Presentación, donde se muestra su nombre y contiene un botón para entrar al sistema.

3 La pantalla principal contará con las siguientes opciones: botón inicio, botón índice, botón ayuda y botón salir.

4 Las pantallas Índice e Inicio contarán con opciones similares (botón inicio, botón índice, botón ayuda, botón salir, botón siguiente y botón atrás).

5 Todas las pantallas de ejercicios tendrán las opciones de botón atrás, botón inicio, botón índice, botón ayuda, botón salir, botón siguiente, botón reproducir/pausar y botón detener grabación de audio correspondiente al ejercicio.

### **RNF 2. Usabilidad**

6 Desde cualquier pantalla se podrá acceder a la pantalla donde se encuentra el listado de temas (índice).

7 Desde cualquier pantalla se podrá salir del producto multimedia.

### **RNF 3. Software**

8. Se requiere un ordenador con el Flash Player 9, o superior instalado.

### **RNF 4. Portabilidad**

9. Se requiere un ordenador con Microsoft Windows 98 SE o superior, o cualquier distribución de Linux con ambiente gráfico.

### **RNF 5. Hardware**

10 Tarjeta de sonido.

11 Procesador de 800 MHz o superior.

12 Memoria RAM de 32 MB o superior.

13 Tarjeta VGA 8 MB o superior.

## **2.5 Modelo de Caso de Uso del Sistema.**

El modelado de casos de uso es la técnica más concreta para modelar los requisitos. Los casos de uso se utilizan para modelar el funcionamiento o cómo el cliente desea que funcione el software. Para determinar los casos de uso del sistema primero se definen cuál o cuáles serían los actores que van a interactuar con el sistema, y luego los casos de uso que van a representar las funcionalidades del mismo.



### 2.5.1 Determinación y Justificación de los Actores del Sistema

Los actores del sistema son las personas que interactúan con el software. Representan terceros fuera del sistema que colaboran con el mismo. Al identificar los actores del sistema se identifica el entorno externo de este.

Actor	Justificación
Usuario	Representa a la persona que va a interactuar con el sistema.

### 2.5.2 Determinación y Justificación de los Casos de Uso del Sistema

Los casos de uso son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Por lo tanto, establece un acuerdo entre clientes y desarrolladores sobre las condiciones que debe cumplir el sistema. (Colectivo de autores, 2008)

A continuación se muestran los casos de uso del sistema.

#### Casos de Uso

1. Navegar por el Sistema.
2. Consultar Ayuda.
3. Visualizar Ejercicio.
4. Manipular Audio.

<b>CU -1</b>	Navegar por el Sistema
<b>Actor</b>	Usuario
<b>Descripción</b>	Se inicia cuando el usuario desea navegar por la aplicación. Recoge los escenarios: Ver Presentación, Ver Inicio, Ver Índice y Salir.
<b>Referencia</b>	CU 3, RF 1, RF 2, RF 3, RF 4, RNF 1, RNF 2, RNF 3, RNF 4, RNF 6,

<b>CU - 2</b>	Consultar Ayuda
<b>Actor</b>	Usuario
<b>Descripción</b>	Se inicia cuando el usuario desea consultar la ayuda de la aplicación
<b>Referencia</b>	RF 5, RNF 1, RNF 5.

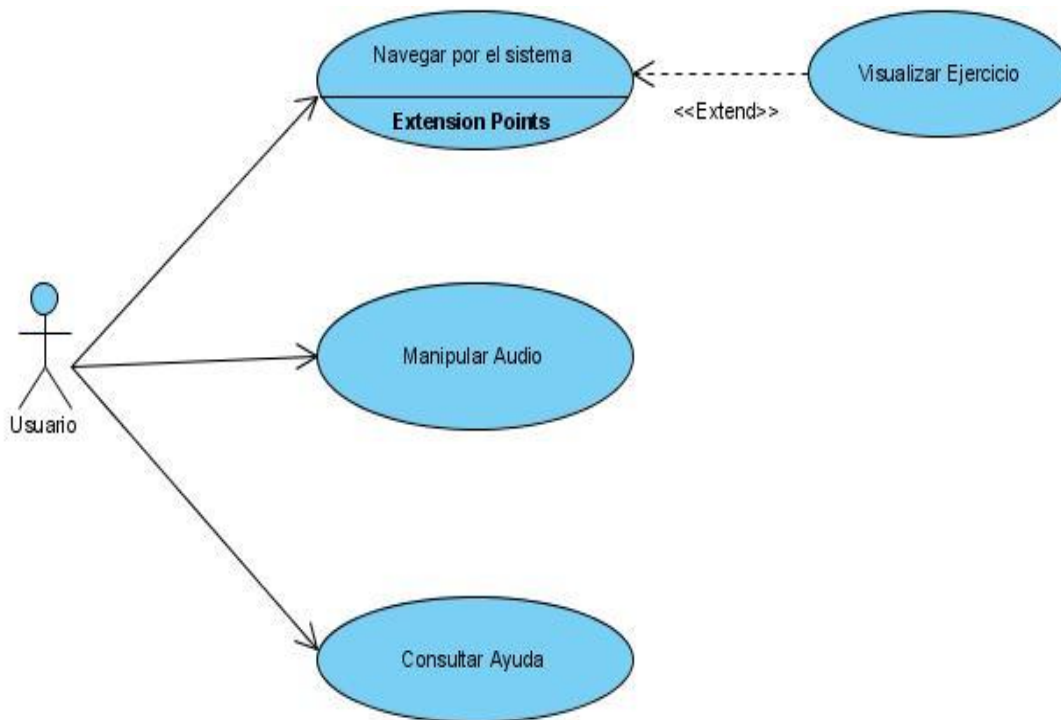
<b>CU - 3</b>	Visualizar Ejercicio
<b>Actor</b>	Usuario
<b>Descripción</b>	Se inicia cuando el usuario está navegando por el sistema y desea interactuar con alguna de las pantallas que contienen ejercicio.
<b>Referencia</b>	CU 1, RF 6, RF 7, RF 8, RF 9, RNF 1.

<b>CU - 4</b>	Manipular Audio
<b>Actor</b>	Usuario
<b>Descripción</b>	Se inicia cuando el usuario solicita reproducir, pausar o detener el audio del sistema.
<b>Referencia</b>	RF 10, RF 11, RNF 1, RNF 10.

Cada forma en que los actores usan el sistema se representa con un caso de uso. Los casos de uso son fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores. Un caso de uso especifica una secuencia de acciones que el sistema puede llevar a cabo interactuando con sus actores, incluyendo alternativas dentro de la secuencia. (Jacobson, 2004).

Los casos de uso son el componente clave del modelado. Su propósito es ilustrar como un sistema permite a un actor cumplir una meta, ilustrando todos los posibles caminos apropiados que ellos pueden tomar para cumplirla, así como las situaciones que podrían hacerlo fallar.

A continuación se representa el Diagrama de Casos de Uso del Sistema donde se muestran las relaciones entre el actor del sistema y los CU. Como se puede observar el CU Visualizar Ejercicio extiende al CU Navegar por el Sistema, evidenciándose el uso de patrones de CU anteriormente estudiado en el Capítulo 1.




**Figura 3. Diagrama de Casos de Uso**

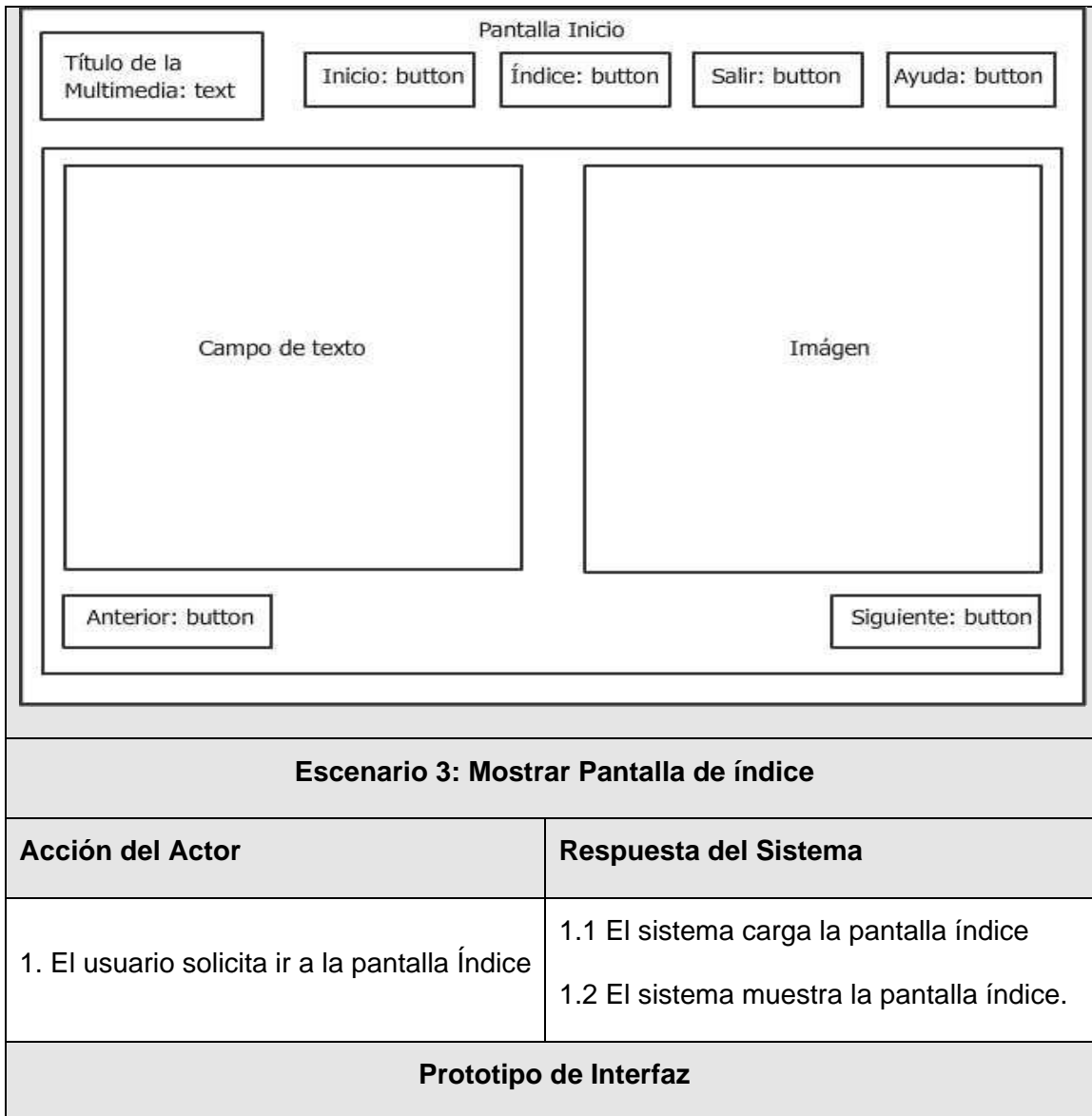
### 2.5.3 Descripción de Casos de Uso

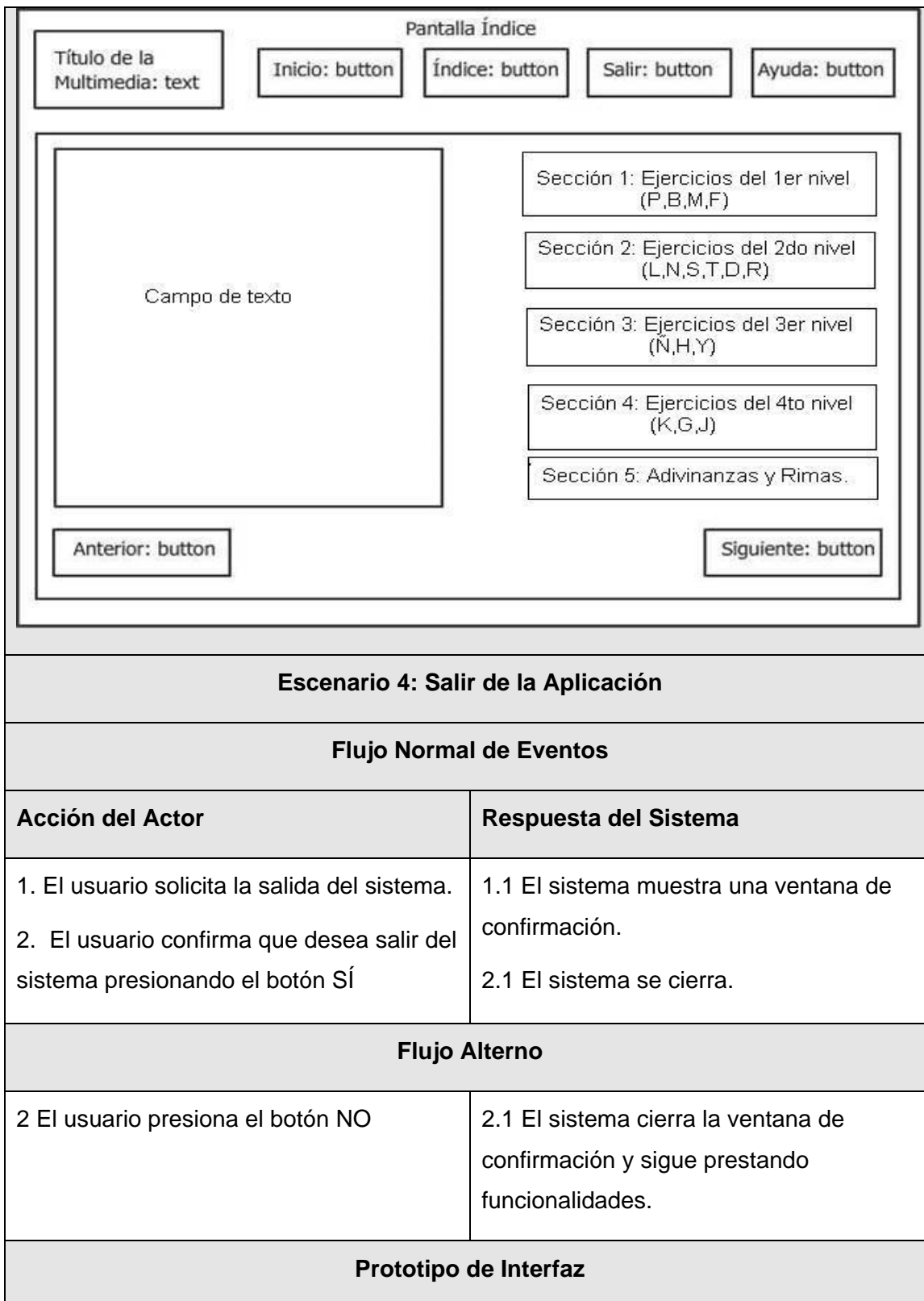
Para efectuar la descripción de los casos de uso del sistema se hace necesario que la misma se realice en un formato expandido y lo más especificada posible, lo que posibilitará una mejor comprensión de los mismos una vez que haya quedando bien detallada la información referente a ellos.

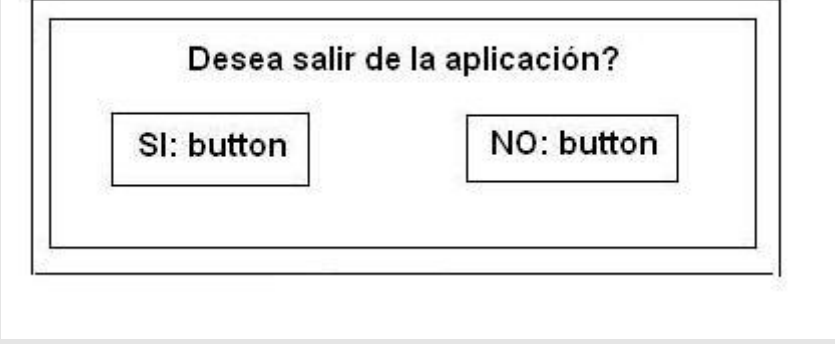
A continuación se muestra la descripción de los casos de uso del sistema.

<b>Caso de Uso</b>	Navegar por el sistema	
<b>Actores</b>	Usuario	
<b>Resumen</b>	El caso de uso se inicia cuando el usuario desea navegar a través del contenido de la multimedia.	
<b>Precondiciones</b>	La aplicación debe haberse iniciado.	
<b>Referencias</b>	RF 1, RF 2, RF 3, RF 4, RF 5, RF 8, RF 9.	
<b>Prioridad</b>	Crítico	
<b>Flujo Normal de Eventos</b>		
<b>Acción del Actor</b>	<b>Respuesta del Sistema</b>	
1. El usuario solicita dirigirse a otra pantalla.	1.1 El sistema muestra la pantalla solicitada.	
<b>Escenario 1: Mostrar Pantalla de Presentación</b>		
<b>Acción del Actor</b>	<b>Respuesta del Sistema</b>	
1. El usuario solicita comenzar a trabajar en la multimedia.	1.1 El sistema carga la presentación de la Multimedia.	
	1.2 El sistema muestra la pantalla de presentación.	
<b>Prototipo de Interfaz</b>		

<p>Pantalla Presentación</p>  <p>Animación: MovieClip</p> <p>Entrar: button</p>	
<b>Escenario 2: Mostrar Pantalla de Inicio</b>	
<b>Acción del Actor</b>	<b>Respuesta del Sistema</b>
1. El usuario oprime el botón entrar	1.1 El sistema carga la pantalla de inicio 1.2 El sistema muestra la pantalla de inicio
<b>Prototipo de Interfaz</b>	





	
<b>Poscondiciones</b>	El sistema se cierra.

<b>Caso de Uso</b>	Consultar Ayuda	
<b>Actores</b>	Usuario	
<b>Resumen</b>	El caso de uso se inicia cuando el usuario solicita la opción de consultar la ayuda de la aplicación.	
<b>Precondiciones</b>	La aplicación debe estar iniciada.	
<b>Referencias</b>	RF 7.	
<b>Prioridad</b>	Crítico	
<b>Flujo Normal de Eventos</b>		
<b>Acción del Actor</b>	<b>Respuesta del Sistema</b>	
1. El usuario estando en cualquier pantalla, solicita la opción de ayuda del sistema.	1.1. El sistema muestra la pantalla Ayuda.	
<b>Prototipo de Interfaz</b>		



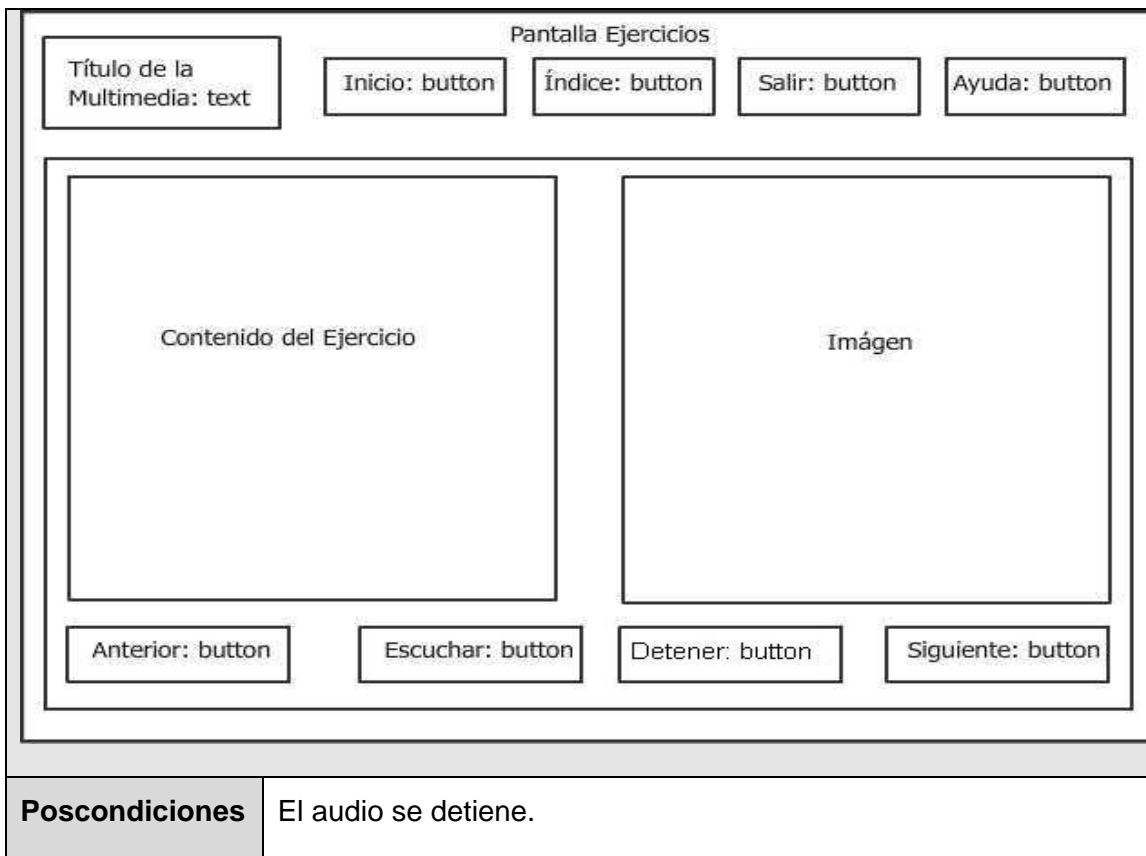
<b>Poscondiciones</b>	Se muestra la Pantalla Ayuda

<b>Caso de Uso</b>	Visualizar Ejercicio
<b>Actores</b>	Usuario
<b>Resumen</b>	El caso de uso se inicia cuando el usuario desea interactuar con un ejercicio determinado.
<b>Precondiciones</b>	La aplicación debe estar iniciada.
<b>Referencias</b>	RF 6, RF 10, RF 11, RF 12, RF 13.
<b>Prioridad</b>	Crítico
<b>Flujo Normal de Eventos</b>	
<b>Acción del Actor</b>	<b>Respuesta del Sistema</b>

<p>1. El usuario solicita interactuar con un ejercicio determinado que se encuentra en otra pantalla.</p>	<p>1.1. El sistema a partir de la selección realizada muestra la pantalla correspondiente.</p>
<p><b>Prototipo de Interfaz</b></p>	
<div style="border: 1px solid black; padding: 10px; text-align: center;"> <p>Pantalla Ejercicios</p> <div style="display: flex; justify-content: space-between; align-items: center;"> <div style="border: 1px solid black; padding: 5px; width: 15%;">Título de la Multimedia: text</div> <div style="border: 1px solid black; padding: 5px; width: 15%;">Inicio: button</div> <div style="border: 1px solid black; padding: 5px; width: 15%;">Índice: button</div> <div style="border: 1px solid black; padding: 5px; width: 15%;">Salir: button</div> <div style="border: 1px solid black; padding: 5px; width: 15%;">Ayuda: button</div> </div> <div style="display: flex; justify-content: space-around; margin-top: 20px;"> <div style="border: 1px solid black; width: 40%; height: 150px; text-align: center; vertical-align: middle;">Contenido del Ejercicio</div> <div style="border: 1px solid black; width: 40%; height: 150px; text-align: center; vertical-align: middle;">Imágen</div> </div> <div style="display: flex; justify-content: space-between; align-items: center; margin-top: 10px;"> <div style="border: 1px solid black; padding: 5px; width: 15%;">Anterior: button</div> <div style="border: 1px solid black; padding: 5px; width: 15%;">Escuchar: button</div> <div style="border: 1px solid black; padding: 5px; width: 15%;">Detener: button</div> <div style="border: 1px solid black; padding: 5px; width: 15%;">Siguiente: button</div> </div> </div>	
<p><b>Poscondiciones</b></p>	<p>Se muestra la Pantalla Ejercicio</p>

<p><b>Caso de Uso:</b></p>	<p>Manipular Audio</p>
<p><b>Actores:</b></p>	<p>Usuario</p>
<p><b>Resumen:</b></p>	<p>El caso de uso se inicia cuando el usuario solicita la opción de manipular el audio del sistema.</p>

<b>Precondiciones</b>	Se debe desear manipular el audio del sistema.	
<b>Referencias</b>	RF 14, RF 15.	
<b>Prioridad</b>	Crítico	
<b>Flujo Normal de Eventos</b>		
<b>Acción del Actor</b>	<b>Respuesta del Sistema</b>	
1. El usuario estando en una pantalla de ejercicios oprime el botón escuchar audio.	1.2 El sistema reproduce el sonido correspondiente.	
<b>Escenario 1: Pausar Audio</b>		
<b>Precondiciones</b>	El audio debe estar reproduciéndose.	
<b>Acción del Actor</b>	<b>Respuesta del Sistema</b>	
2 El usuario solicita pausar el audio presionando el botón escuchar.	2.1 El sistema pausa la reproducción.	
<b>Poscondiciones</b>	El Audio se pausa.	
<b>Escenario 2: Detener Audio</b>		
<b>Precondiciones</b>	El audio debe estar reproduciéndose.	
2 El usuario oprime el botón detener audio.	2.1 El sistema detiene la reproducción del audio correspondiente.	
<b>Prototipo de Interfaz</b>		



### 2.6 Conclusiones

El trabajo realizado en este capítulo permitió demostrar entre otros aspectos que la realización del modelo del dominio sirvió para tener una mejor interpretación de los conceptos asociados, demostró además que el flujo de trabajo captura de requisitos permite tener una visión de las capacidades o condiciones que el sistema debe cumplir así como las cualidades que debe poseer y el medio donde se usará, para que con ello el usuario se encuentre con una aplicación multimedia rápida, atractiva y fácil de utilizar. En este capítulo también se determinó el actor del sistema así como los casos de uso, estos últimos fueron descritos en formato expandido, quedando bien detallada la información referente a ellos. Es válido destacar la importancia del estudio de los patrones de casos de uso realizado en el capítulo 1, pues en el DCUS se evidencia el uso de patrones en el CU Visualizar Ejercicio que extiende al CU Navegar por el Sistema. Ya culminado este flujo de trabajo, se puede comenzar a desarrollar el análisis y diseño para dar cumplimiento a los requisitos y su acabado sea con la mayor calidad posible.

### **CAPÍTULO 3: ANÁLISIS Y DISEÑO DE LA SOLUCIÓN PROPUESTA**

#### **3.1 Introducción**

En este capítulo se mostrarán los artefactos del flujo de trabajo Análisis y Diseño, como son: los Diagramas de Clases de Análisis y Diagramas de Colaboración para el análisis; y los Diagramas de Clases de Diseño y de Secuencia para el diseño. Estos artefactos contribuirán a un mayor entendimiento para los desarrolladores en la futura implementación de la aplicación.

#### **3.2 Análisis**

El análisis forma parte del proceso de desarrollo de software, cuyo propósito es formular el modelo del dominio del problema. El objetivo es desarrollar una serie de modelos que describan el software al trabajar para satisfacer el conjunto de requisitos definidos por el cliente. El análisis debe lograr describir lo que quiere el cliente y establecer una base para la creación de un diseño de software. (Colectivo de autores, 2008)

La realización del análisis proporciona una visión general del sistema que puede ser más difícil de obtener mediante el estudio de los resultados del diseño y la implementación, por contener demasiados detalles.

##### **3.2.1 Modelo de Clases de Análisis por Caso de Uso.**

Las clases de análisis representan un modelo conceptual temprano para “cosas en el sistema que tienen responsabilidades y comportamientos”.

Las clases del análisis son usadas para capturar las principales “concentraciones de responsabilidad” en el sistema. Ellas representan las clases prototipo del sistema y son un primer paso en la principal abstracción que el sistema debe manejar. Las clases del análisis deben ser mantenidas en su propio medio, si se desea una vista conceptual de alto nivel del sistema. (Colectivo de autores, 2008)

A continuación se representan dichos diagramas.

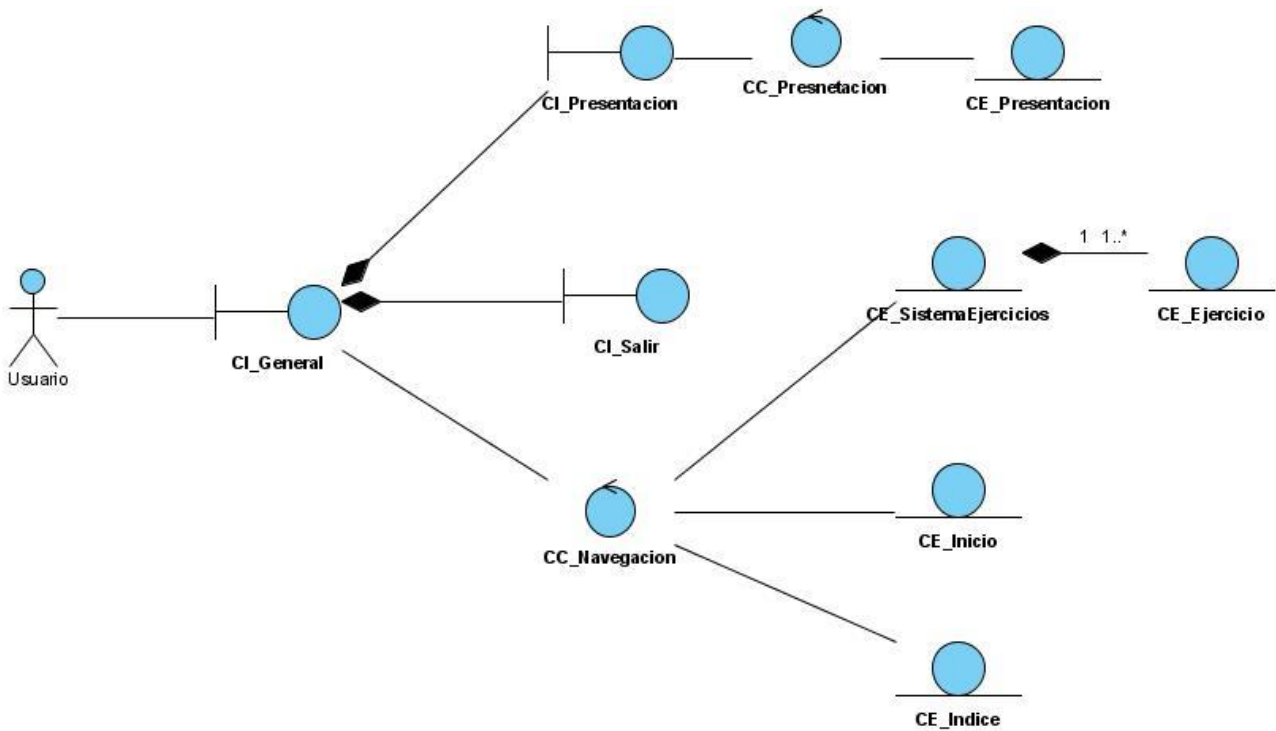


Figura 4. Diagrama de Clases de Análisis. Navegar por el Sistema

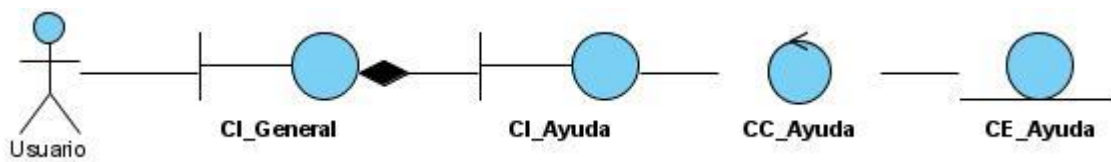


Figura 5. Diagrama de Clases de Análisis. Consultar Ayuda.

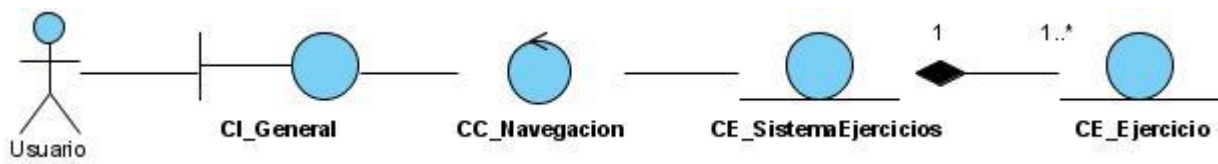


Figura 6. Diagrama de Clases de Análisis. Visualizar Ejercicio

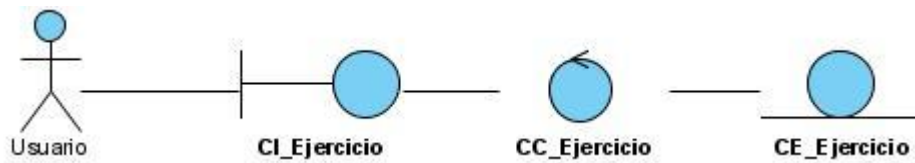


Figura 7. Diagrama de Clases de Análisis. Manipular Audio

### 3.2.2 Diagramas de Interacción. Colaboración.

Para la realización de los casos de uso del análisis se utilizarán los diagramas de interacción. De dichos diagramas se escogieron los de colaboración. Un diagrama de colaboración es un diagrama de interacción que destaca la organización estructural de los objetos que envían y reciben mensajes.

Tienen dos características principales (Colectivo de autores, 2008)

1. En primer lugar, el camino. Para indicar cómo se enlaza un objeto a otro, se puede asociar un estereotipo de camino al extremo más lejano de un enlace. Normalmente, sólo se necesita representar explícitamente el camino del enlace.
2. En segundo lugar, está el número de secuencia. Para indicar la ordenación temporal de un mensaje, se precede de un número (comenzando con el mensaje número 1), que se incrementa secuencialmente por cada nuevo mensaje en el flujo de control (2, 3, etc.)

A continuación se muestran estos diagramas por caso de uso.

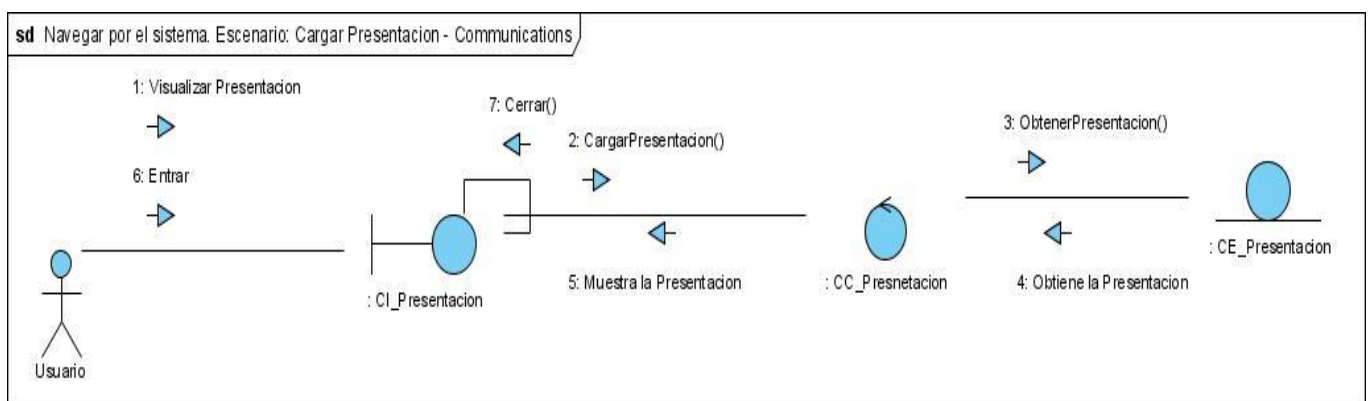


Figura 8. Diagrama de Colaboración. Navegar por el sistema. Escenario Cargar Presentación

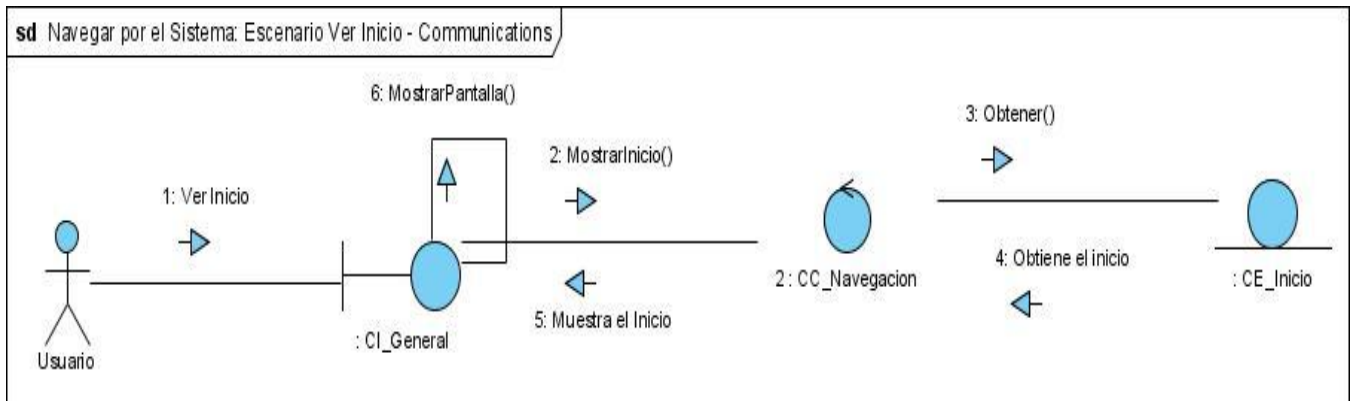


Figura 9. Diagrama de Colaboración. Navegar por el sistema. Escenario Ver Inicio

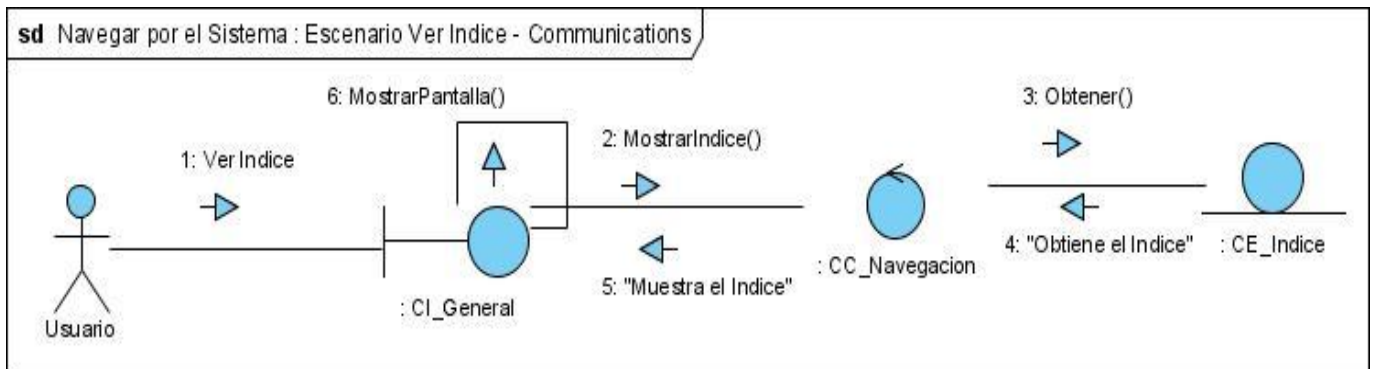


Figura 10. Diagrama de Colaboración. Navegar por el sistema. Escenario Ver Índice

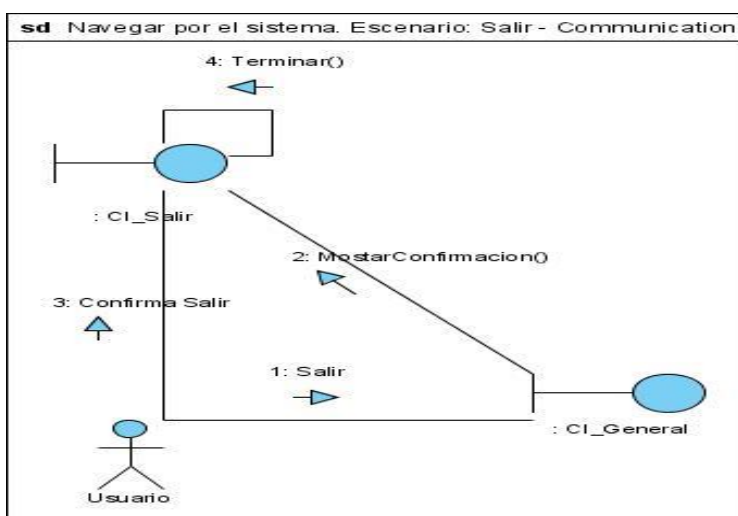


Figura 11. Diagrama de Colaboración. Navegar por el sistema. Escenario Salir



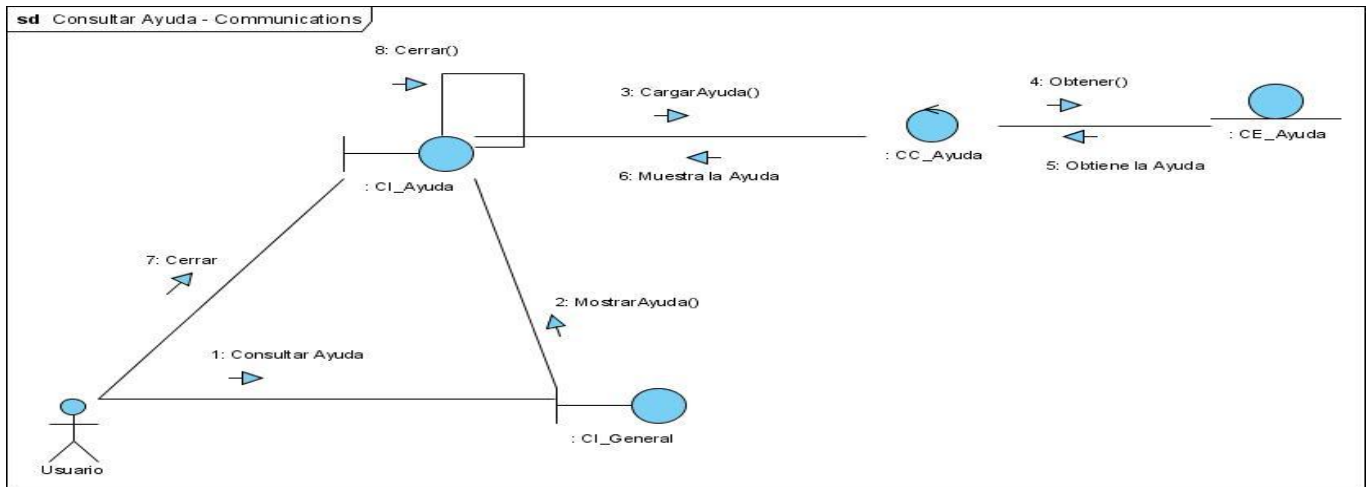


Figura 12. Diagrama de Colaboración. Consultar Ayuda.

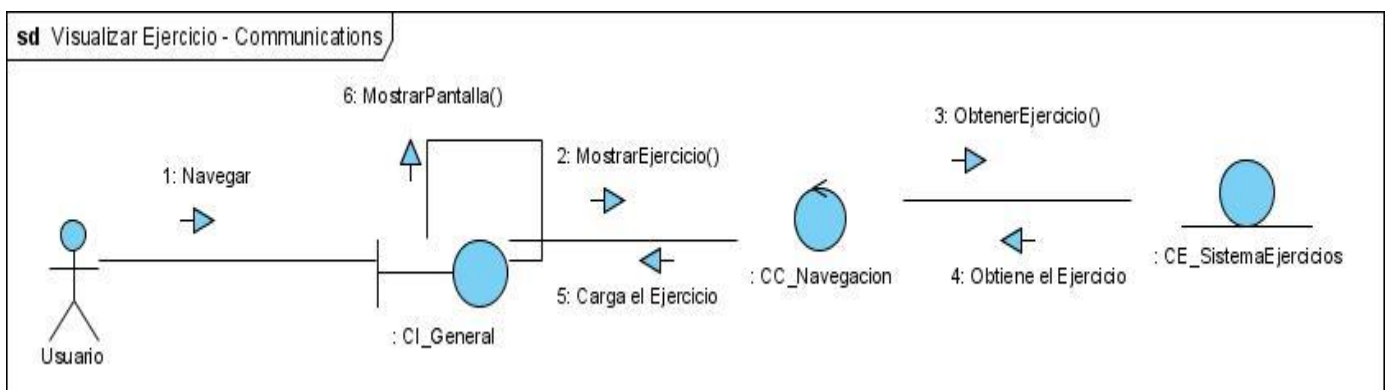


Figura 13. Diagrama de Colaboración. Visualizar Ejercicio.

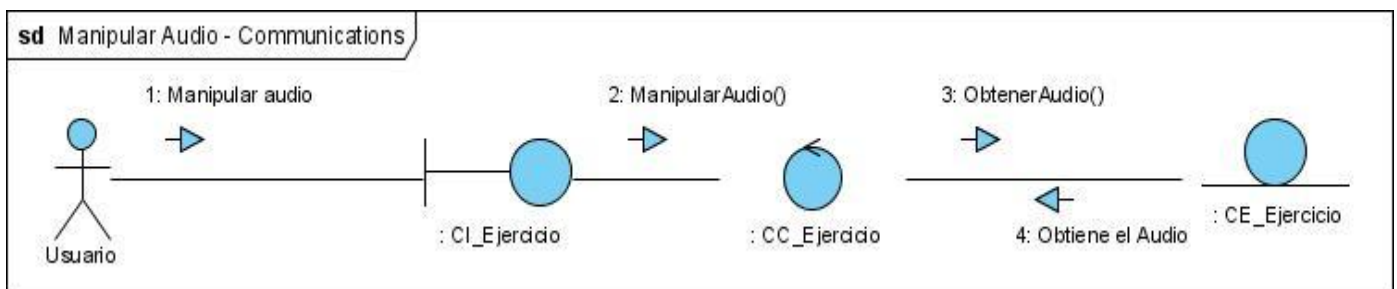


Figura 14. Diagrama de Colaboración. Manipular Audio.

### 3.3 Diseño

El diseño se utiliza para modelar el sistema y encontrar su forma para que soporte todos los requisitos, incluyendo los requisitos no funcionales y demás restricciones. El diseño debe ser una guía legible y comprensible para aquellos que generan código y para aquellos que comprueban y consecuentemente, dan soporte al software; deberá proporcionar una imagen completa del mismo, enfrentándose a los dominios de comportamiento, funcionales y de datos desde una perspectiva de implementación. (Colectivo de autores, 2008)

#### 3.3.1 Diagramas de Interacción. Secuencia.

Para la realización de los casos de uso se escogieron además los diagramas de secuencia como instrumento para lograr un mayor nivel de detalle en el diseño planteado. Un diagrama de secuencia es un diagrama de interacción que destaca la ordenación temporal de los mensajes. (Colectivo de autores, 2008)

A continuación se muestran los diagramas de secuencia por cada caso de uso.

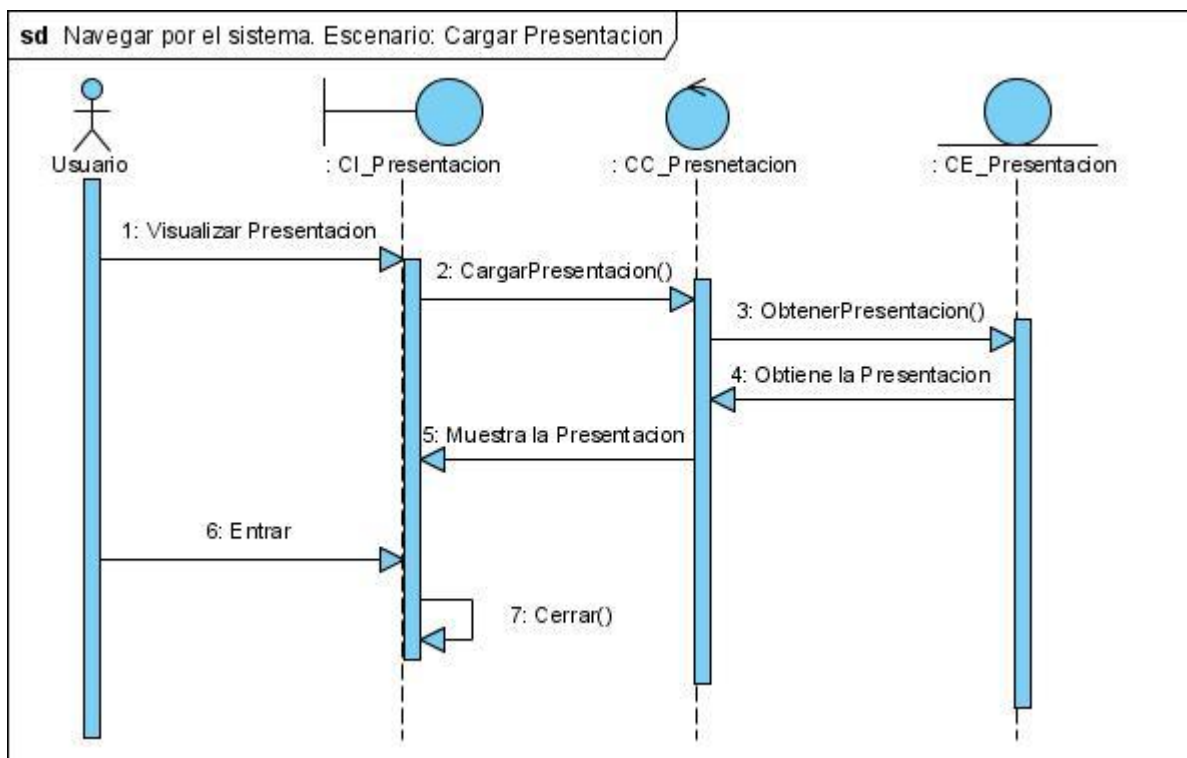


Figura 15. Diagrama de Secuencia. Navegar por el Sistema. Escenario Cargar Presentación

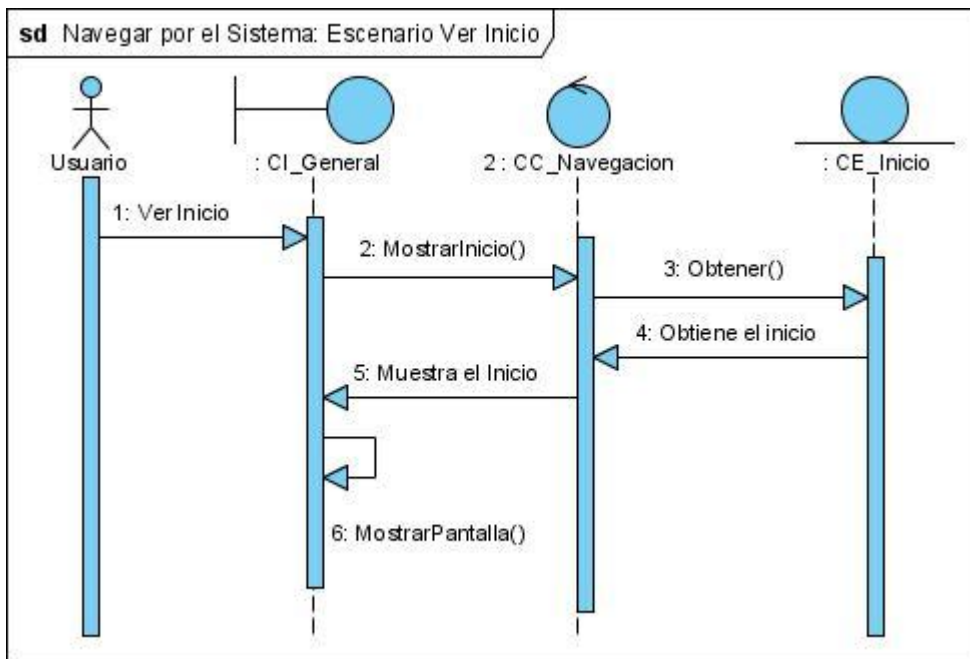


Figura 16. Diagrama de Secuencia. Navegar por el Sistema. Escenario Ver Inicio

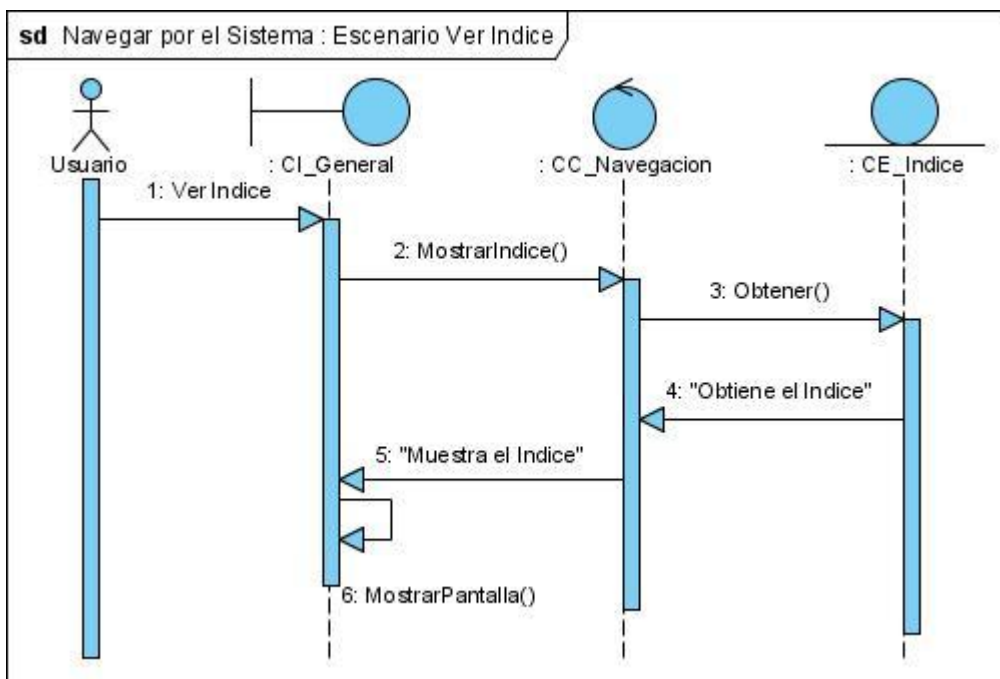


Figura 17. Diagrama de Secuencia. Navegar por el Sistema. Escenario Ver Índice

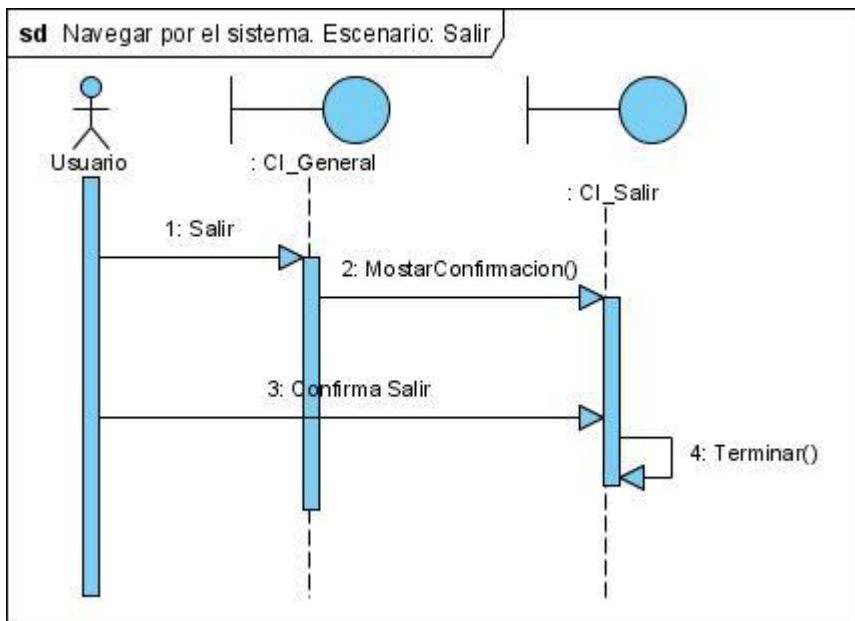


Figura 18. Diagrama de Secuencia. Navegar por el Sistema. Escenario Salir del sistema

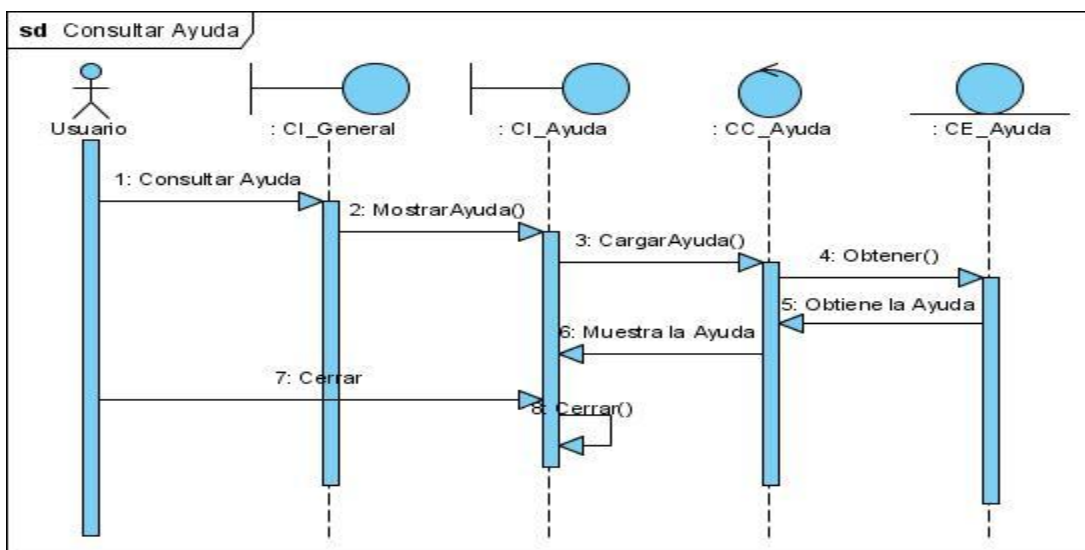


Figura 19. Diagrama de Secuencia. Consultar Ayuda

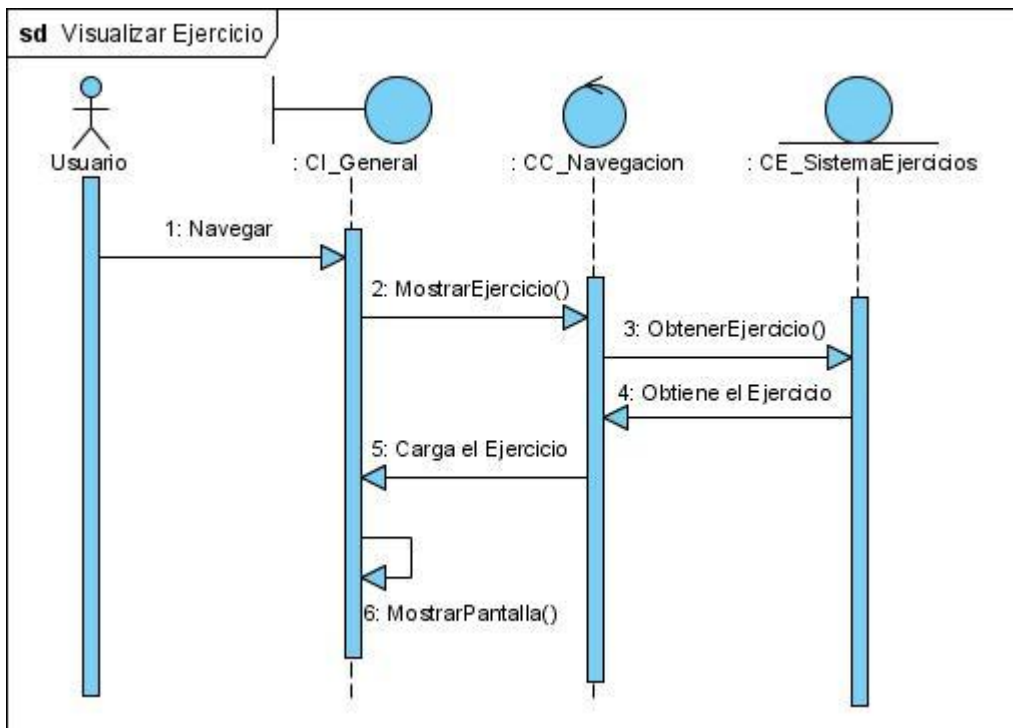


Figura 20. Diagrama de Secuencia. Visualizar Ejercicio.

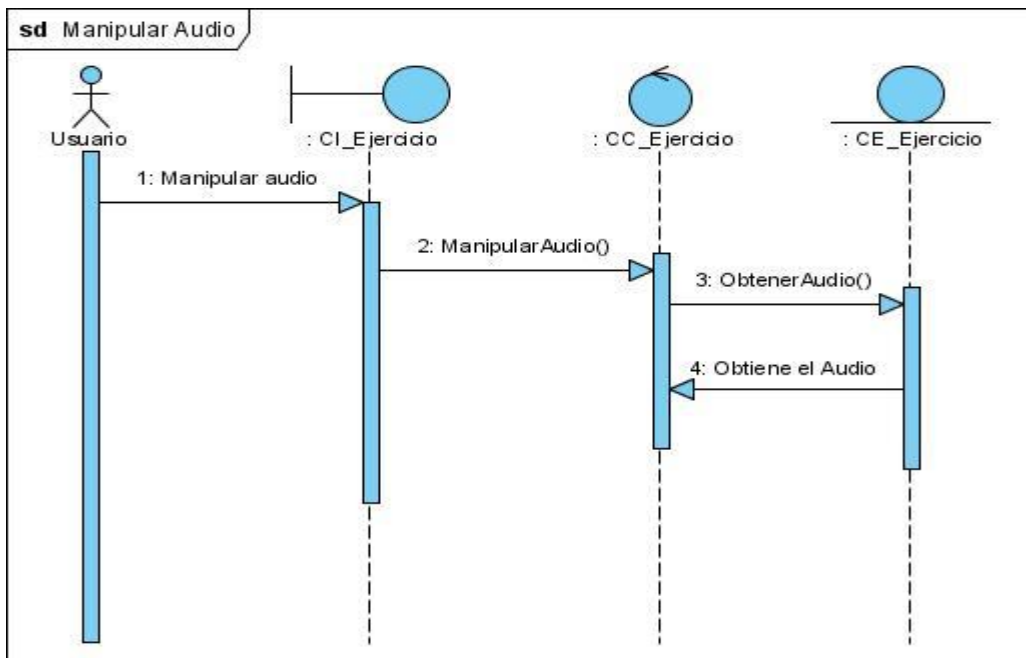


Figura 21. Diagrama de Secuencia. Manipular Audio.

### 3.3.2 Diagramas de Clases de Diseño

Un Diagrama de Clases de Diseño es una construcción similar en la implementación del sistema. El lenguaje utilizado para especificar una clase del diseño es lo mismo que el lenguaje de programación. Las operaciones, atributos, tipos, visibilidad (public, protected, private) se pueden especificar con la sintaxis del lenguaje elegido. (Colectivo de autores, 2008)

A continuación se representa el diagrama de clases de diseño, donde se muestran las relaciones de composición entre las clases que intervienen en el sistema. Se muestra además la relación de herencia entre todas las clases de interfaz de usuario y la clase "Sprite". Todas las clases que son hijas de Sprite representan las clases con las que va a interactuar el usuario. La clase Sprite se encuentra en el espacio de nombre flash.display y es la clase idónea para almacenar interfaces de usuario debido a que posee todas las características de un objeto gráfico y carece de línea de tiempo.

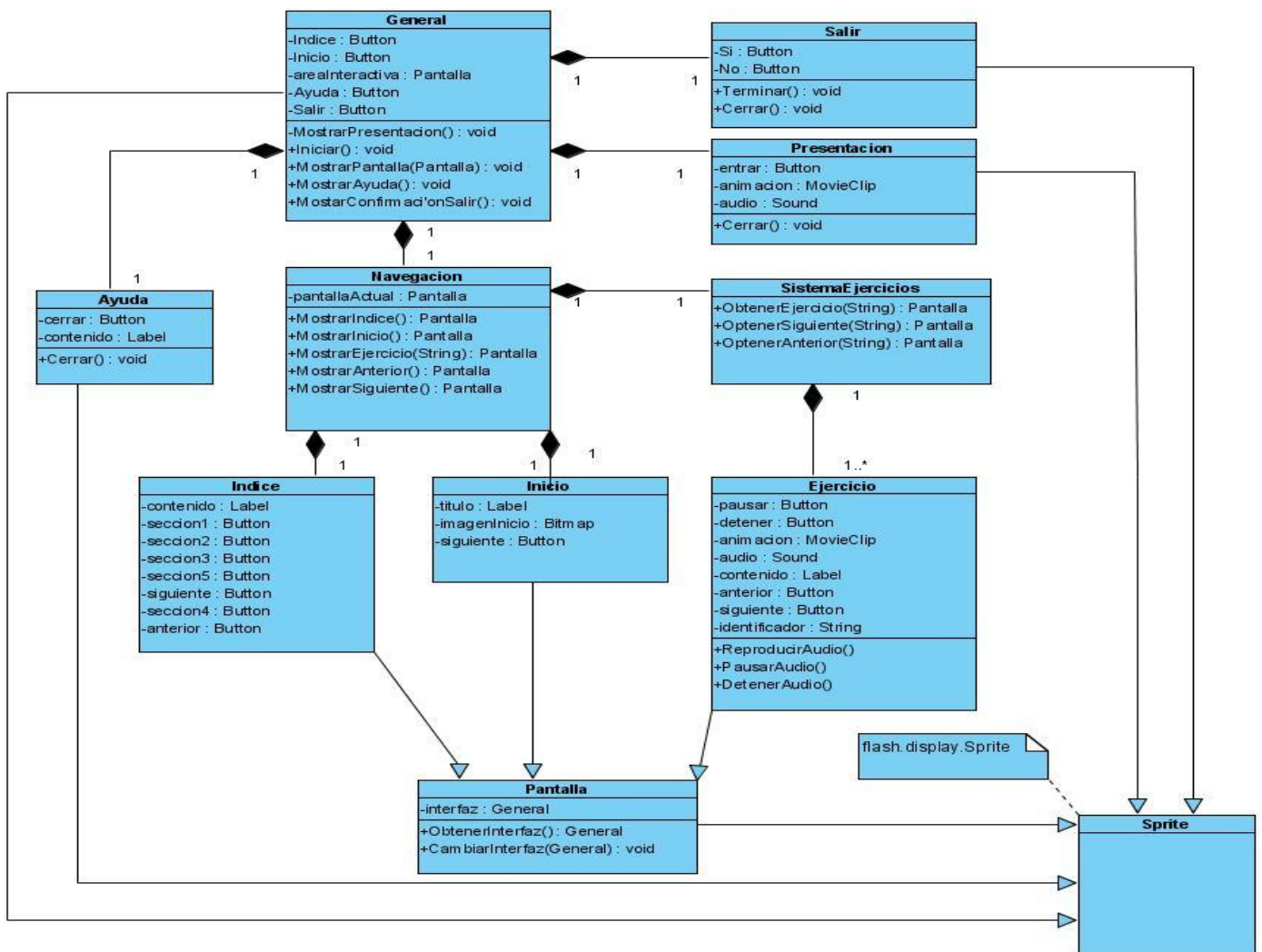


Figura 22. Diagrama de Clases de Diseño.

En el diagrama de clases del diseño se evidencia el uso de algunos patrones de diseño como el experto que permitió asignar responsabilidades a las clases más competentes para llevarlas a cabo, el patrón creador que permitió la creación de instancias de algunas clases en otras, el patrón alta cohesión que contribuyó a darle a las clases el menor número de responsabilidades pero con mucha relación lógica entre ellas y finalmente el patrón bajo acoplamiento permitiendo que la dependencia entre las clases fuera la menor posible, de este modo si cambiara la lógica de alguna clase no se afectarían las demás.

### **3.4 Conclusiones**

El trabajo realizado en este capítulo permitió demostrar que los artefactos generados en el flujo de trabajo de análisis y diseño posibilitarán una mayor comprensión por parte de los desarrolladores en la futura implementación del software. Además se puede afirmar que el uso de los patrones de diseño anteriormente explicados contribuyó a que la realización del diagrama de clases del diseño tuviera la mayor calidad posible. Luego de obtener todos estos artefactos, para conocer la calidad del trabajo realizado se debe pasar a la etapa de validación de los resultados obtenidos.

### CAPÍTULO 4: VALIDACIÓN DE RESULTADOS

#### 4.1 Introducción

A raíz del trabajo realizado se obtuvieron numerosos artefactos, para la validación de estos se hace necesario la aplicación del método correspondiente en cada caso. De manera general se obtuvieron una serie de resultados que en el presente capítulo se validarán poniendo en práctica el estudio realizado en el capítulo 1 referente a los métodos de validación.

#### 4.2 Métrica de la Calidad de Especificación de los Requisitos

El objetivo de esta métrica de validación es eliminar la ambigüedad de los requisitos. Esto constituye un aspecto esencial para la elaboración del producto pues permite que los requisitos cumplan con la redacción correspondiente y que todo el que tenga contacto con los mismos tenga la misma interpretación para cada uno de ellos.

Para la aplicación de esta métrica se hizo necesario conocer los valores de  $n_r$ ,  $n_f$  y  $n_{nf}$

Luego sustituyendo los valores en la fórmula:

$$n_r = n_f + n_{nf}$$

Se obtuvo que:

El número de requisitos = el número de requisitos funcionales + el número de requisitos no funcionales

Donde:  $n_r = 11 + 13$

El número de requisitos es 24.

Para darle soporte a lo planteado anteriormente se realizaron 2 revisiones, con 3 revisores consultados, todo ello con el objetivo de obtener la mayor claridad posible en los requisitos. A continuación se mencionan estos revisores, que fueron seleccionados por su nivel de preparación y experiencia en el tema:

Ing. Daniel Varona Cordero. Asesor de Calidad. Facultad 3. UCI. Se desempeñó como revisor técnico y auditor durante seis meses en el grupo de calidad de la facultad 3 y como jefe de calidad interna en el proyecto Registro y Notaría durante dos años.

Ing. Daylén Benítez Matos. Analista. Facultad 3. UCI. Se desempeñó como analista durante dos años en el proyecto Registro y Notaría y actualmente ocupa el rol de gestor de configuración en el proyecto Tribunales Populares Cubanos.



Ing. Yinett Hernández Hernández. Analista. Facultad 3. UCI. Se desempeñó como analista durante dos años en el proyecto Registro y Notaría y actualmente ocupa el rol de analista en el proyecto Tribunales Populares Cubanos.

### Primera Revisión

En la primera revisión efectuada se pudo constatar que uno de los requisitos no cumplía con la redacción correspondiente y presentaba problemas de ambigüedad.

Por lo tanto para un total de 24 requisitos, los revisores técnicos tuvieron la misma interpretación para 23 de ellos.

Luego se procedió a medir la especificidad de los requisitos calculando el valor de Q en la fórmula:

$$Q = n_{ui} / n_r$$

Donde  $n_{ui}$  es el número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas.

Luego se obtuvo que

$$Q = 23 / 24$$

$$Q = 0.96$$

Como se puede ver la especificidad de los requisitos tuvo un buen valor en la primera revisión pero teniendo en cuenta que el número de requisitos no es tan elevado se procede a una segunda revisión buscando un total entendimiento de los revisores.

### Segunda Revisión

En la segunda y última revisión no se detectaron problemas, se realizó un estudio de los requisitos mucho más profundo y para una totalidad de 24 requerimientos, los revisores tuvieron la misma interpretación para todos.

Luego, calculando el valor de la especificidad de los requisitos (Q) se tiene que:

$$Q = 24 / 24$$

$$Q = 1$$

Luego de realizar un estudio cuantitativo de los resultados arrojados posterior a las revisiones, se puede llegar a la conclusión que los requisitos no eran ambiguos. Una vez aplicada la métrica se contribuyó a la eliminación de la ambigüedad, por lo tanto, el nivel de calidad del proceso de

refinamiento de requisitos fue bastante alto, algo fundamental para desarrollar con éxito un software que sea capaz de cumplir con las expectativas y necesidades del cliente.

### 4.3 Técnica de Prototipado

Para la elaboración y construcción de la interfaz gráfica se hizo uso de Adobe Flash CS3 modelo que permitió validar todos los requisitos funcionales que fueron capturados durante la etapa de Requerimientos. La principal ventaja que presentan las interfaces de usuarios es la de reflejar la presentación e interacción de las necesidades del usuario final en un entorno amigable y fácil de entender. Luego de la construcción de la interfaz principal, se hace necesaria la fabricación de otras con pequeñas modificaciones que complementen el producto del software final.

### 4.4 Modelo de Métricas Orientadas a Objeto aplicadas al DCUS.

#### Heurística basada en NOAS/NOS

El objetivo de esta métrica es calcular la precisión con que fueron descritos los casos de uso. Para obtener un buen resultado final es necesario que el número de pasos del actor esté en torno al 50 % respecto al número total de pasos.

#### Aplicación de esta métrica para la descripción de los casos de uso del sistema:

Para aplicar la heurística basada en NOAS / NOS es necesario conocer que:

NOAS es el número de pasos del actor.

NOS es el número total de pasos.

Posteriormente se calcula el valor de esta métrica dividiendo en valor de NOAS por el valor de NOS.

Luego, sustituyendo los valores en cada caso de uso por separado se obtiene la siguiente tabla.

Caso de Uso	Representación en % de NOAS respecto a NOS
CU 1	30 %
CU 2	50 %
CU 3	50 %
CU 4	50 %

**Tabla 1. Representación de la Métrica aplicada al Diagrama de Casos de Uso**

Haciendo un análisis de la tabla anterior se comprueba que en las descripciones de los casos de uso nunca se obvió la participación del sistema, lo que permitió que ninguno resultara incompleto, además las acciones de del actor se desglosaron correctamente.

Luego de aplicar esta métrica, se puede concluir diciendo que la descripción de los casos de uso del sistema es buena pues los resultados en su mayoría igualan al 50 % y ninguno está fuera de rango (el rango es entre 30% y 60%).

### 4.5 Métricas Orientadas al Tamaño de Clase (TC).

Las métricas orientadas a tamaños para una clase se centran en cálculos de atributos y de operaciones para una clase individual, y promedian los valores para el sistema.

La siguiente tabla ilustra las clases del sistema aplicándole la métrica seleccionada.

No	Nombre de la Clase	Cantidad de Atributos	Cantidad de Operaciones	Tamaño
1	General	9	5	Pequeño
2	Navegación	4	5	Pequeño
3	Índice	8	0	Pequeño
4	Inicio	3	0	Pequeño
5	Salir	2	2	Pequeño
6	Presentación	3	1	Pequeño
7	Ayuda	2	1	Pequeño
8	Pantalla	1	2	Pequeño
9	SistemaEjercicio	1	3	Pequeño
10	Ejercicio	8	0	Pequeño

**Tabla 2. Tamaño de las clases.**

Cuando existe un TC grande se afectan los parámetros de calidad definidos por esta métrica. Se reduce la reutilización de las clases, la implementación se hace más compleja, las pruebas son difíciles de realizar y aumenta la responsabilidad de las clases.

Como se muestra en la tabla todas las clases que conforman el sistema están dentro de la categoría de pequeñas, lo que demuestra que el sistema no es complejo. Los resultados obtenidos son positivos según esta métrica.

A continuación se realiza la clasificación de las clases dentro de las categorías de Pequeño, Medio y Grande (Tabla 3) Así como una representación del promedio de atributos que componen dichas clases (Tabla 4)

<b>Clasificación</b>	<b>Cantidad de Clases</b>
Pequeño	10
Medio	0
Grande	0

**Tabla 3. Clasificación de Clases.**

<b>Cantidad Clases</b>	<b>Promedio Atributos</b>	<b>Promedio Operaciones</b>
10	4.1	1.9

**Tabla 4. Promedio de Operaciones de Clases.**

### 4.6 Conclusiones

Con la aplicación de estos métodos se validaron los resultados obtenidos en los capítulos anteriores y se arribaron a las siguientes conclusiones:

El nivel de ambigüedad en los requisitos disminuyó, lográndose el nivel de especificidad requerido y el entendimiento total por parte de los revisores.

Los diagramas de presentación reflejaron la interacción de las necesidades del usuario final en un entorno amigable y fácil de entender.

La descripción de los casos de uso del sistema sin dudas, es buena pues los resultados en su mayoría igualan al 50 % y ninguno está fuera de rango.

El 100% de las clases que conforman el diseño están dentro de la categoría de pequeñas, lo que demuestra que el proceso de construcción y pruebas de la aplicación no será complejo.

En fin, las métricas aplicadas a los artefactos obtenidos validan de forma general la calidad del análisis y diseño del software.

### CONCLUSIONES

Después de haber realizado un exhaustivo estudio para el análisis y diseño de esta aplicación y haber concluido la fase de elaboración del producto, se ha determinado lo siguiente:

1. Las búsquedas bibliográficas posibilitaron obtener gran parte de la información necesaria para la realización de la investigación.
2. El estudio de las metodologías, herramientas y lenguajes de modelado posibilitó la elección de los más adecuados para el desarrollo del trabajo, seleccionándose:
  - Como metodología de desarrollo: RUP por ser robusta y generar gran cantidad de artefactos que posibilita mejor entendimiento para futuras implementaciones.
  - Como herramienta de modelado: Visual Paradigm, teniendo en cuenta las tendencias actuales de migrar a software libre pues es multiplataforma además de generar reportes e imágenes de muy buena calidad y de poseerse la licencia para uso.
  - Como lenguaje de modelado: UML por ser el más utilizado, conocido e ideal para modelar sistemas de cómputo.
  - Como herramienta de Implementación: Adobe Flash CS3 que incorpora el lenguaje de programación ActionScript 3.0, además de ser una de las más potentes y fácil de usar y la ideal para el desarrollo de productos multimedia que por su capacidad de reducir las animaciones a la mínima expresión en cuanto al espacio e incorpora potentes efectos de fácil uso.
3. La confección del Modelo del Dominio permitió definir y relacionar todos los conceptos del mismo.
4. La captura de los requisitos funcionales y no funcionales permitió tener una visión de las capacidades o condiciones que el sistema debe cumplir así como las cualidades que debe poseer y el medio donde se usará, para que con ello el usuario se encuentre con una aplicación multimedia rápida, atractiva y fácil de utilizar.
5. La elaboración de los artefactos correspondientes a cada uno de los flujos de trabajo por los que pasó el producto permitió convertir las necesidades del cliente en un lenguaje entendible para los desarrolladores lo que posibilita un mejor entendimiento en la futura implementación del software.
6. La validación de los resultados obtenidos propició medir la calidad del trabajo realizado.

### **RECOMENDACIONES**

1. Elaborar los artefactos que no se hicieron en las etapas por las que pasó el producto.
2. Desarrollar la implementación del producto utilizando la herramienta propuesta.
3. Aplicar la ingeniería inversa utilizando la herramienta Visual Paradigm para actualizar los diagramas de clases del diseño.

## REFERENCIAS BIBLIOGRÁFICAS

**Rodríguez. 2006.** MULTIMEDIA, ToolBook. [Online] 2006. [Cited: diciembre 12, 2008.] <http://www.monografias.com/trabajos10/mmedia/mmedia.shtml#toolbook>.

**2006.** AULACLIC. [Online] 2006. [Cited: mayo 3, 2008.] [http://www.aulaclit.es/flashcs3/t\\_1\\_1.htm](http://www.aulaclit.es/flashcs3/t_1_1.htm).

**Carlos Zapata y J. Arango. 2004.** Alineación entre Metas Organizacionales y Elicitación de Requisitos del Software. [Online] 2004. [Cited: enero 25, 2009.] <http://www2.unalmed.edu.co/dyna2005/143/ALINEACI%D3N%20ENTRE%20METAS.pdf>.

**Colectivo de autores. 2008.** Ingeniería de Software I , Conferencia 5 Profundización del flujo de trabajo de requerimientos. *Teleformacion.uci.cu*. [Online] 2008. [Cited: enero 24, 2009.] [http://teleformacion.uci.cu/file.php/102/Curso\\_2008-2009/Materiales\\_Basicos/Materiales\\_Basicos\\_Conf\\_5/Conf5\\_Pro](http://teleformacion.uci.cu/file.php/102/Curso_2008-2009/Materiales_Basicos/Materiales_Basicos_Conf_5/Conf5_Pro).

—. **2008.** Ingeniería de Software II , Materiales Complementarios. Técnicas de análisis y desarrollo de Software. *Teleformación.uci.cu*. [Online] 2008. [Cited: enero 5, 2009.] <http://teleformacion.uci.cu/mod/resource/view.php?id=14085>.

—. **2008.** Ingeniería de Software II , Tema 1, Semana 1, Materiales Básicos. Continuación del FT Análisis y Diseño. Modelo de Diseño. Material de Apoyo. Conferencia de Diseño. [Online] 2008. [Cited: enero 5, 2009.] [http://teleformacion.uci.cu/file.php/259/CURSO\\_2009/Mate](http://teleformacion.uci.cu/file.php/259/CURSO_2009/Mate).

—. **2008.** Materiales Complementarios. Patrones GRASP. *Ingeniería de Software II*. [Online] 2008. [Cited: febrero 4, 2009.] <http://teleformacion.uci.cu/mod/resource/view.php?id=14080>. Colectivo de autores, 2008-2009, Ingeniería de Software II , Materiales Complementarios..

**Dávila, Nicolás. 2001.** *Ingeniería de Requerimientos una guía para extraer, analizar, especificar y validar los requerimientos de un proyectos*. 2001.

**Dra. Nelcys Rodríguez Peña, Norma Regal Cabrera, Belkis Correa Mozo, y René Suárez Martínez. 2008.** *Trabajo de Revisión Hospital Docente "Juan Manuel Marquez" Anomalías de la oclusión y otros trastornos en la agricultura de la palabra*. 2008.

**Fleitas, Dra. Xiomara Rodríguez. 2001.** *Sistema Integral para el tratamiento logopédico*. Habana : s.n., 2001.

**Furini, Gastónx. 2008.** Ingeniería del software - Nivel Intermedio. [Online] febrero 2, 2008. <http://www.clubdesarrolladores.com/articulos/mostrar/63-metodologia-scrum..>

**Giraldo, Z. 2003.** *Herramientas de Ingeniería de software*. 2003.



- Gonzalez, Jose Barreras. 2009.** *Profes.net*. [Online] 2009.  
[http://www.primaria.profes.net/archivo2.asp?id\\_contenido=43406](http://www.primaria.profes.net/archivo2.asp?id_contenido=43406).
- Grupo Soluciones Innova.S.A. (GSI). 2007.** Rational Rose Enterprise. [Online] 2007. [Cited: enero 7, 2009.] <http://www.rational.com.ar/herramientas/roseenterprise.html>..
- Guillem Bou Bauzá. 2008.** LICENCIATURA EN COMUNICACIÓN SOCIAL “Reglas fundamentales del diseño de guiones”. [Online] 2008. [Cited: febrero 4, 2009.] <http://www.unsl.edu.ar/~techo/multimedia/guionmultimedia.htm>.
- I. Jacobson, G. Booch y J. Rumbaugh. 2004..** “*El Proceso Unificado de Desarrollo de Software*”. La Habana : Félix Varela, 2004.
- IEEE. 1993.** s.l. : Instituto de Ingenieros Eléctricos y Electrónicos, 1993. 729.
- J. Gómez. 2006.** ¿Metodología?... Sí, pero, ¿cuál? [Online] 2006. [Cited: diciembre 12, 2008.] <http://www.versioncero.com/articulo/469/metodologiasi-pero-cual..>
- Jesus Ruiz Felipe. 2008.** COMPARATIVA DE SISTEMAS MULTIMEDIA EN EL DISEÑO DE UNIDADES DIDÁCTICAS EN CIENCIAS, Multimedia en la enseñanza de las Ciencias. [Online] 2008. [Cited: febrero 3, 2009.] <http://www.sociedadelainformacion.com/fisica/multimedia/multimedia.htm>.
- Koch, Nora. 2002.** *Ingeniería de Requisitos en Aplicaciones para la Web-Un estudio comparativo*. España : Universidad de Sevilla Lenguajes y Sistemas Informáticos, 2002. pág. 26..
- Larman, C. 2004.** “*UML y PATRONES. Introducción al análisis y diseño orientado a objetos*”. La Habana : Félix Varela, 2004.
- Latina, O. 2007.** Importancia de UML. [Online] 2007. [Cited: diciembre 2, 2008.] <http://www.osmosislatina.com/lenguajes/uml/basico.htm>.
- Palomo, Frank David Avalos. 2007.** *Multimedia Educativa para los niños de la Enseñanza Primaria con disgrafía escolar*. Habana : s.n., 2007.
- Pérez, Durán, & Ruiz. 2007.** ¿Por qué OMG ha elegido BPMN para modelar procesos de negocio si ya existe UML? España : s.n., 2007.
- Pressman, Roger. 2005.** *Ingeniería del Software. Un enfoque práctico*. La Habana : Félix Varela, 2005.
- Sixto. 2003.** ¿Patrones? 2003. [Online] 2003. [Cited: marzo 29, 2009.] <http://www.code4net.com/archives/000030.html>..

**Víctor Barbone. 2006.** XP: Extreme Programming (Programación Extrema). [Online] 2006. [Cited: febrero 4, 2009.] <http://ie.fing.edu.uy/~nacho/blandos/seminario/XProg1.html>..

---

**BIBLIOGRAFÍA**

**Hernández León, R. A., & Coello González, S. (2002).** EL PARADIGMA CUANTITATIVO DE LA INVESTIGACIÓN CIENTÍFICA. Ciudad de la Habana.

**García Rubio, F. Ó.** Metodologías de desarrollo de software. S/A.

**Canós, J. H., Letelier, P., & Penadés, M. C.** (n.d.). Metodologías Ágiles en el Desarrollo de Software. Retrieved marzo 1, 2009, from <http://www.willydev.net/descargas/prev/TodoAgil.pdf>

**Monzón, A.** Calidad de la Especificación: ¿Se pueden medir los Requisitos?

**B. Bernárdez, A., & Durán Toro, M. (2004).** UNA PROPUESTA PARA LA VERIFICACIÓN DE REQUISITOS BASADA EN MÉTRICAS. Revista de Procesos y Métricas de las Tecnologías de la Información (RPM) , España.

**Ivar Jacobson, Grady Booch, James Rumbaugh.** El Proceso Unificado de desarrollo de Software. 2000. <http://bibliodoc.uci.cu/pdf/reg00060.pdf>

**Anónimo. (2007-2008).** Asignatura: Ingeniería de Software 1. Retrieved enero 2009, from Título: Introducción a la Ingeniería de Software.: <http://teleformacion.uci.cu>

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

**Rodríguez. 2006.** MULTIMEDIA, ToolBook. [Online] 2006. [Cited: diciembre 12, 2008.] <http://www.monografias.com/trabajos10/mmedia/mmedia.shtml#toolbook>.

**2006. Aulaclic.** [Online] 2006. [Cited: mayo 3, 2008.] [http://www.aulaclic.es/flashcs3/t\\_1\\_1.htm](http://www.aulaclic.es/flashcs3/t_1_1.htm).

**Carlos Zapata y J. Arango. 2004.** Alineación entre Metas Organizacionales y Elicitación de Requisitos del Software. [Online] 2004. [Cited: enero 25, 2009.] <http://www2.unalmed.edu.co/dyna2005/143/ALINEACI%D3N%20ENTRE%20METAS.pdf>.

**Dávila, Nicolás. 2001.** *Ingeniería de Requerimientos una guía para extraer, analizar, especificar y validar los requerimientos de un proyectos.* 2001.

**Dra. Nelcys Rodríguez Peña, Norma Regal Cabrera, Belkis Correa Mozo, y René Suárez Martínez. 2008.** *Trabajo de Revisión Hospital Docente “Juan Manuel Marquez” Anomalías de la oclusión y otros trastornos en la agricultura de la palabra.* 2008.

**Fleitas, Dra. Xiomara Rodríguez. 2001.** *Sistema Integral para el tratamiento logopédico.* Habana : s.n., 2001.

- Furini, Gastónx. 2008.** Ingeniería del software - Nivel Intermedio. [Online] febrero 2, 2008. <http://www.clubdesarrolladores.com/articulos/mostrar/63-metodologia-scrum..>
- Giraldo, Z. 2003.** *Herramientas de Ingeniería de software*. 2003.
- Gonzalez, Jose Barreras. 2009.** *Profes.net*. [Online] 2009. [http://www.primaria.profes.net/archivo2.asp?id\\_contenido=43406](http://www.primaria.profes.net/archivo2.asp?id_contenido=43406).
- Grupo Soluciones Innova.S.A. (GSI). 2007.** Rational Rose Enterprise. [Online] 2007. [Cited: enero 7, 2009.] <http://www.rational.com.ar/herramientas/roseenterprise.html..>
- Guillem Bou Bauzá. 2008.** LICENCIATURA EN COMUNICACIÓN SOCIAL “Reglas fundamentales del diseño de guiones”. [Online] 2008. [Cited: febrero 4, 2009.] <http://www.unsl.edu.ar/~tecno/multimedia/guionmultimedia.htm>.
- I. Jacobson, G. Booch y J. Rumbaugh. 2004..** “*El Proceso Unificado de Desarrollo de Software*”. La Habana : Félix Varela, 2004.
- IEEE. 1993.** s.l. : Instituto de Ingenieros Eléctricos y Electrónicos, 1993. 729.
- J. Gómez. 2006.** ¿Metodología?... Sí, pero, ¿cuál? [Online] 2006. [Cited: diciembre 12, 2008.] <http://www.versionzero.com/articulo/469/metodologias-i-pero-cual..>
- Jesus Ruiz Felipe. 2008.** COMPARATIVA DE SISTEMAS MULTIMEDIA EN EL DISEÑO DE UNIDADES DIDÁCTICAS EN CIENCIAS, Multimedia en la enseñanza de las Ciencias. [Online] 2008. [Cited: febrero 3, 2009.] <http://www.sociedadelainformacion.com/fisica/multimedia/multimedia.htm>.
- Koch, Nora. 2002.** *Ingeniería de Requisitos en Aplicaciones para la Web-Un estudio comparativo*. España : Universidad de Sevilla Lenguajes y Sistemas Informáticos, 2002. pág. 26..
- Larman, C. 2004.** “*UML y PATRONES. Introducción al análisis y diseño orientado a objetos*”. La Habana : Félix Varela, 2004.
- Latina, O. 2007.** Importancia de UML. [Online] 2007. [Cited: diciembre 2, 2008.] <http://www.osmosislatina.com/lenguajes/uml/basico.htm>.
- Palomo, Frank David Avalos. 2007.** *Multimedia Educativa para los niños de la Enseñanza Primaria con disgrafía escolar*. Habana : s.n., 2007.
- Pérez, Durán, & Ruiz. 2007.** ¿Por qué OMG ha elegido BPMN para modelar procesos de negocio si ya existe UML? España : s.n., 2007.
- Pressman, Roger. 2005.** *Ingeniería del Software. Un enfoque práctico*. La Habana : Félix Varela, 2005.

**Sixto. 2003.** ¿Patrones? 2003. [Online] 2003. [Cited: marzo 29, 2009.]  
<http://www.code4net.com/archives/000030.html>..

**Víctor Barbone. 2006.** XP: Extreme Programming (Programación Extrema). [Online] 2006.  
[Cited: febrero 4, 2009.] <http://iie.fing.edu.uy/~nacho/blandos/seminario/XProg1.html>..

### GLOSARIO DE TÉRMINOS

**Actor:** Abstracción de las entidades externas a un sistema, subsistemas o clases que interactúan directamente con el sistema. Un actor participa en un caso de uso o en conjunto coherente de casos de usos para llevar a cabo un propósito global.

**Artefacto:** Pieza de información utilizada o producida por un proceso de desarrollo de software como un documento externo o el producto de un trabajo. Un artefacto puede ser un modelo, elementos dentro del modelo, una descripción, o el software.

**Animación:** Representación sucesiva de una secuencia de imágenes que produce la sensación de estar viendo imágenes en movimiento. Para ello, a cada imagen de una animación se le modifica un pequeño detalle para mantener el movimiento tan fluido como sea posible.

**Aplicación:** Agrupa elementos de media y unifica sus funcionalidades como una entidad.

**Escenario:** representa un conjunto de pantallas que muestran una información a través de objetos con similar funcionalidad.

**Gif:** Graphics Interchange Format. Formato de Intercambio de Gráficos. Formato de archivos de imágenes digitales muy utilizado en la elaboración de productos con tecnología multimedia por ser de reducidas dimensiones.

**GRASP:** Patrones Generales de Software para la Asignación de Responsabilidades.

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

**Interfaz:** Frontera convencional entre dos sistemas o dos unidades, que permite intercambio de informaciones.

**Interfaz de usuario:** Conjunto de elementos que permiten al usuario dialogar con una aplicación interactiva. Estos elementos incluyen tanto el hardware (teclado, ratón, pantalla táctil) como el diseño de las pantallas y la navegación por el contenido.

**Media:** hace referencia a sonido, texto, imágenes, animaciones, video.

**Metodologías:** Se refiere a los métodos de investigación en una ciencia. Se entiende como la parte del proceso de investigación que permite sistematizar los métodos y las técnicas necesarios para llevarla a cabo. Define Quién debe hacer, Qué, Cuándo y Cómo debe hacerlo.

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

**Metodología Ágil:** Constituyen un nuevo enfoque en el desarrollo de software, mejor aceptado por los desarrolladores de proyectos que las metodologías convencionales debido a la simplicidad de sus reglas y prácticas, su orientación a equipos de desarrollo de pequeño tamaño, su flexibilidad ante los cambios y su ideología de colaboración .

**Pantalla:** es un grupo de elementos de medias visuales que están comprendidos en una vista determinada.

**Patrón:** Es una solución a un problema no trivial que es efectiva (ya se resolvió el problema satisfactoriamente en ocasiones anteriores) y reusable (se puede aplicar a diferentes problemas en distintas circunstancias).

**Release:** Lanzamiento de la versión definitiva de un producto final a menos que aparezcan errores que lo impidan.

**RUP:** Metodología para el desarrollo de software, que en español sería Proceso Unificado de Desarrollo.

**Software:** Sistemas o Aplicaciones expresadas en un lenguaje de máquina.

**UML:** “Unified Modeling Language”. Lenguaje gráfico que brinda un vocabulario y reglas para especificar, construir, visualizar y documentar los artefactos de un sistema utilizando el enfoque orientado a objetos.

**Visual Paradigm:** Herramienta CASE (Computer-Aided Software Engineering) que permite realizar ingeniería tanto directa como inversa. Utiliza UML como lenguaje de modelado y RUP como base de desarrollo.

**XP:** eXtreme Programming. Metodología de desarrollo de software basada en valores como simplicidad, comunicación, retroalimentación.

## ANEXOS

### Anexo 1. Ejercicios de Pronunciación

Ejercicios de Pronunciación

#### EJERCICIOS “S”

Sigmoterapia inicial: Con los dientes unidos y los labios en posición de sonrisa forzada emitir silbidos suaves, largos y finos:

Ssssss

Sigmoterapia + vocal (extender sigmoterapia): Con los dientes unidos y los labios en forma de sonrisa forzada, emitir silbidos finos, suaves y largos terminando en vocal acentuada:

Sssá, sssé, sssí, sssó, sssú

Sigmoterapia extensa: Con los dientes unidos y los labios en forma de sonrisa forzada emitir silbidos suaves, largos y finos terminando en vocales y después comenzando con vocales y por último acentuando la última sílaba:

Sssá asss asssssá

Sssé esss essssé

Sssí isss issssí

Sssó osss ossssó

Sssú usss usssssú

Secuencia funcional “S” simple: Con los dientes unidos y los labios en forma de sonrisa forzada emitir silbidos suaves, largos y finos acompañados de vocales:

Sssassssassá

Sssessssessé

Sssisssissí



Sssossssosssó

Sssussssusssú

Secuencia funcional “S” compleja: Realizar una secuencia de sonidos en la cual se comienza por una vocal a la cual siguen silbidos suaves, largos y finos con los labios en forma de sonrisa forzada, se repite la vocal, el silbido y se termina con la vocal acentuada:

Asss sssa sssá

Esss ssse sssé

Isss sssi sssí

Osss sso sssó

Usss sssu sssú

“S-J” y vocales: Con los dientes unidos y los labios en posición de sonrisa forzada emitir silbidos finos, suaves y largos terminados en una sílaba de “J” y vocal acentuada:

sssjá sssjé sssjí sssjó sssjú

Automatizar “S”: Repetir palabras que tengan “S” en distintas posiciones articulatorias

Nota: Debe comenzarse con palabras bisílabas agudas y llanas de uso cotidiano, luego en frases y oraciones, lectura y conversaciones.

## EJERCICIOS “J”

Funcionalismo “J”: Con la boca abierta emitir soplos de jadeo amplios y prolongados:

Jjjj

Funcionalismo “J” + Vocal: Con la boca abierta emitir soplos de jadeo amplios y prolongado terminando en vocal acentuada:

jjjá jjjé jjjí jjjó jjjú

Secuencia funcional “J”: Abrir la boca, emitir soplos de jadeo amplios suaves y

prolongados en series de tres:

Jjjj jjjj jjjj

Secuencia funcional “J” + vocal: Abrir la boca, emitir soplos de jadeo amplios suaves y prolongados terminando en vocal acentuando la última vocal en series de tres:

Jjja jaja jjjá

Jjje jje jjjé

Jjji jji jjjí

Jjjo jjo jjjó

Jjju ju jjjú

Yotatoterapia extensa: Abrir la boca, emitir soplos de jadeo amplios suaves y prolongados primero terminando en vocal después comenzando en vocal y por último el jadeo entre vocal terminando en vocal acentuada:

Jjá ajj ajjá

Jjé ejj ejjé

Jjí ijj ijjí

Jjo ojjo ojjo

Jju ujju ujjú

## EJERCICIOS “L”

Funcionalismo “L”: Colocar la punta de la lengua detrás de los incisivos superiores y dar un sonido de “L” largo y mantenido.

Funcionalismo “L” + vocal: Colocar la punta de la lengua detrás de los incisivos superiores y dar un sonido de “L” + vocal acentuada:

Llá llé llí lló llú

Lambdacismo complejo: Disponer los órganos articulatorios (labios, lengua, dientes) en posición para emitir el fonema correspondiente (P,B,F,K,G,T) sin llegar a emitirlo salir rápidamente de esa posición y caer en la sílaba de “L” + vocal acentuando esta

última:

p-lá            b-lá

p-le            b-lé

p-lí            b-lí

p-ló            b-ló

p-lú            b-lú

Secuencias de sílabas con “L”: Hacer secuencias rapidísimas de sílabas con consonantes P,B,F,G,K y “L” + vocal acentuando esta última:

Palapalapalapalalá

Pelepelepelepelelé

Pilipilipilipililí

Polopolopolopololó

Pulupulupulupululú

### **EJERCICIOS “R”**

Vibración bilabial: Hacer vibrar cuidadosamente los labios de manera larga y mantenida imitando el sonido de un motor:

Brrrrrrrrrrrrrrrr

Vibración bilabial + vocal: Hacer vibrar cuidadosamente los labios de manera larga y mantenida imitando el sonido de un motor:

Brrrá brrré brrrí brrró brrrú

Vibración bilabial en serie: Hacer vibrar ruidosamente los labios de manera larga y mantenida imitando el sonido de un motor en serie de tres:

Brrr brrr brrr

Vibración lingual: Con la punta de la lengua detrás de los dientes de arriba hacerla vibrar como un motor

rrrrrrrrrrrrrrrrrr

Vibración bilabial + vocal en serie: Hacer vibrar ruidosamente los labios de manera larga y mantenida imitando el sonido de un motor terminando en vocal en serie de tres acentuando la última vocal:

Brrrá brrrá brrrá

Brrré brrré brrré

Brrrí brrrí brrrí

Brrró brrró brrró

Brrrú brrrú brrrú

“L-r”: Abrir la boca, llevar la lengua arriba y hacia atrás, emitiendo los siguientes sonidos:

larr lerr lirr lorr lurr

L - r +vocal: Abrir la boca, llevar la lengua arriba y hacia atrás, emitiendo los siguientes sonidos

lará leré lírí loró lurú

L -R +vocal: Abrir la boca, llevar la lengua arriba y hacia atrás, emitiendo los siguientes sonidos

Lar-rrá ler-rré lir-rrí lor-rró lur-rrú

R-R: Decir rápido y claro:

Ar-rrá er-rré ir-rrí or-rró ur-rrú

r-R: Decir rápido claro:

ara-RRÁ ere-RRÉ iri-RRÍ oro-RRÓ uru-RRÚ

r-r-R: Decir rápido y claro:

arar-Rar-RRÁ erer-Rer-RRÉ irir-Rir-RRÍ oror-Ror-RRÓ urur-Rur-RRÚ

r directa: Decir rápido y claro:

ará eré írí oró urú

AR-AR-AR: Decir rápido y claro:

Ar,ar,ar

Er,er,er

Ir,ir,ir

Or,or,or

Ur,ur,ur

Serie td'd'd: Dar golpecitos rápidos en el cielo de la boca con la punta de la lengua dirigiéndolos hacia atrás Diciendo tddddd

Serie tereré: Dar golpecitos rápidos en el cielo de la boca con la punta de la lengua dirigiéndolos hacia atrás Diciendo tererereré

.Td'd'd'-RR: Similar al ejercicio anterior:

TerererereRRÁ

TerererereRRÉ

TerererereRRÍ

TerererereRRÓ

TerererereRRÚ

Rotacismo complejo: Colocar los órganos articulatorios en posición para emitir el fonema correspondiente ( P,B,F,G,K,T) y sin llegar a admitirse salir rápidamente de esa posición y caer en la sílaba de "r" más vocal acentuando esta última:

p-rá p-ré p-rí p-ró p-rú

b-rá b-ré b-rí b-ró b-rú

f-rá f-ré f-rí f-ró f-rú

g-rá g-ré g-rí g-ró g-rú

k-rá k-ré k-rí k-ró k-rú

t-rá t-ré t-rí t-ró t-rú

### **EJERCICIOS CON "K":**

Funcionalismo "K": Abrir la boca y realizar una contracción fuerte con la garganta en

forma de chasquido diciendo:

Ak ek ik ok uk

Funcionalismo “K” + vocal: Abrir la boca y realizar una contracción fuerte con la garganta en forma de chasquido terminando en vocal acentuada:

AKKKÁ EKKKÉ IKKKÍ OKKKÓ UKKKÚ

K-K: Abrir la boca y realizar una contracción fuerte con la garganta separando un tanto la última sílaba acentuándola fuertemente:

AKKKA-KÁ

EKKKE-KÉ

IKKKI-KÍ

OKKK-KÓ

UKKK-KÚ

K-P: Abrir la boca y realizar una contracción fuerte con la garganta en forma de chasquido y caer en una sílaba explosiva de “P +vocal”:

G-K: Abrir la boca y emitir sílabas de “G” largas y suaves diferenciándolas inmediatamente de sílabas con “K” y vocales acentuadas:

Gggaká gggeké gggikí gggokó gggukú

Kapacismo terapia instrumental: Ejercer una presión media sobre la parte delantera de la lengua del paciente con un depresor manteniendo la boca abierta. En esas condiciones pedirle al paciente que pronuncie la “T”. Al no poder hacerlo se produce la emisión de la “K” al contracción lingual.

### **EJERCICIOS CON “F”:**

Funcionalismo “F”: Con la boca abierta levantar el labio superior con los dedos índices empujar al mismo tiempo con los dedos pulgares el labio inferior contra los dientes superiores. En esa posición dar silbidos de forma fina, suave y prolongada.

Funcionalismo “F” + vocal: Con la boca abierta levantar el labio superior con los dedos índices empujar al mismo tiempo con los dedos pulgares el labio inferior contra

los dientes superiores. En esa posición dar silbidos de forma fina, suave y prolongada terminando en vocales acentuadas:

Fffá fffé ffí fffó fffú

Fismoterapia extensa: Con la boca abierta poner los órganos fonoarticulatorios en forma correcta para dar el sonido “F” con silbidos suaves, finos y largos primero terminando en vocal acentuada, después comenzando con vocal y por último el silbido “F” entre vocales acentuando la última

Fffa afff afffá

Fffa efff efffé

Fffi ifff ifffí

Fffo offf offfó

Fffu ufff ufffú

### **EJERCICIOS CON “P”:**

Funcionalismo “P” + vocal: Abuchar bien las mejillas y después mantener un instante el aire en la boca expulsarlo bruscamente dando un sonido “P” bien fuerte y explosivo acompañado de una vocal acentuada

PÁ PÉ PÍ PÓ PÚ

Funcionalismo “P”: Abuchar bien las mejillas y después mantener un instante el aire en la boca expulsarlo bruscamente dando un sonido “P” bien fuerte y explosivo

### **EJERCICIOS CON “T”:**

Funcionalismo “T”:

Primer paso: Exteriorizar la lengua aprisionándola con los labios con cierta presión sin dejarla retroceder y decir:

Tá té tí tó tú

Segundo paso: Exteriorizar la lengua aprisionarla con los dientes con cierta presión y sin dejarla retroceder decir:

Tá té tí tó tú

Tercer paso: Apoyando la lengua fuertemente detrás de los dientes superiores decir.

Tá té tí tó tú

### **EJERCICIOS CON “D”:**

Funcionalismo “D”: Colocar la lengua detrás de los incisivos superiores y emitir un sonido “D” largo y sostenido haciendo vibrar la laringe

Funcionalismo “D” + vocal: Colocar la lengua detrás de los incisivos superiores y emitir un sonido “D” largo y sostenido haciendo vibrar la laringe terminando en vocal acentuada:

**EJERCICIO CON “M”:** Aproximar suavemente los labios hasta hacer contacto y en esa posición musitaciones suaves y largas.

mmmmmmmmmm

**EJERCICIO CON “Ñ” y “N”:** Colocar la punta de la lengua detrás de los incisivos superiores y emitir vibraciones laríngeas y nasales frunciendo exageradamente el ceño.

Ññññññññññññññññ

Nnnnnnnnnnnnnn

Funcionalismo “G”: Con la boca abierta dar el sonido imitando la manera de gárgaras.

### **EJERCICIOS CON “CH”:**

Funcionalismo “CH”: Proyectar los labios en forma de hocico y en esa posición emitir soplos en chorro:

Funcionalismo “CH” + vocal: Proyectar los labios en forma de hocico y en esa



posición emitir soplos en chorro terminando en vocal acentuada.

### EJERICICIOS CON “Y”

Hacer diptongaciones rápidas con “i” y las vocales acentuando fuertemente aumentando la rapidez de la dicción hasta la mayor posible buscando la aparición del sonido “ll”

lá ié íí ió iú

## Anexo 2: Plantilla para la Descripción de Casos de Uso

<b>Caso de Uso</b>	
<b>Actores</b>	
<b>Resumen</b>	
<b>Precondiciones</b>	
<b>Referencias</b>	
<b>Prioridad</b>	
<b>Flujo Normal de Eventos</b>	
<b>Sección “”</b>	
<b>Acción del Actor</b>	<b>Respuesta del Sistema</b>
<b>Prototipo de Interfaz</b>	
<b>Flujos Alternos</b>	

---

Acción del Actor	Respuesta del Sistema
<i>Prototipo de Interfaz</i>	
Poscondiciones	