Universidad de las Ciencias Informáticas

Facultad 4

Título: "Análisis y Diseño del nivel B de la especificación IMS-LD para el módulo de Diseño Instruccional en CRODA 2.0"

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

Autor:
Judiht García Rodríguez.

Tutor:
Ing. Osvaldo E. Stable Vilches
Cotutor:
Lic. Yisell Góngora Parra.

La Habana, junio de 2012
"Año 53 de la Revolución"

Declaración de Autoría

Por este medio declaro que soy la única autora de este trabajo y autorizo a la Universidad de las Ciencias Informáticas (UCI) para que hagan el uso que estimen pertinente con este trabajo.

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

Judiht García Rodríguez	Ing. Osvaldo E. Stable Vilches
_____	_____
Firma del Autor	Firma del Tutor

Agradecimientos

A toda mi familia por el apoyo que siempre me brindaron, especialmente a mis padres por estar siempre conmigo y ser mi guía.

A mi gran amor Raidel, por todo su amor, su cariño y su comprensión, te amo mi vida.

A mi tutor Osvaldo por guiarme, por su confianza y por corregirme siempre que fue necesario.

A mis grandes amigas, a las que adoro Margarita y Mirita.

A mis feas: Karen, Katiuska, Olga, Yanisleydis que han estado conmigo siempre en las buenas y en las malas bridándome su ayuda durante estos 5 años, las quiero.

A mis compañeras de apartamento y compañeros de aula durante todos estos años.

Dedicatoria

A mis padres María Eugenia y Roberto por todo el apoyo que me han brindado siempre, por ser los mejores padres y le mejor ejemplo que he tenido siempre.

A mi hermanito Roberto para que vea lo importante que es forjarse un futuro.

A toda mi familia que siempre ha estado tan orgullosa de mí.

Resumen

Las tecnologías de la información y las comunicaciones han tenido un impacto considerable en la educación a distancia. De la interacción de estas tecnologías parte el concepto de lo que se conoce como *e-learning*. En las actuales tecnologías, los estándares y/o especificaciones ocupan un lugar importante en el campo del *e-learning*, estos favorecen la interoperabilidad entre las diferentes tecnologías que con ellos desarrollan recursos formativos que generan o utilizan.

IMS - LD, es actualmente el estándar más versátil y respaldado para modelar procesos completos de aprendizaje como complemento de estructuras de contenidos educativos. IMS –LD es un lenguaje de modelado educativo que tiene como objetivo definir formalmente una estructura semántica para anotar los procesos de enseñanza y aprendizaje, y así convertirlos en entidades reutilizables entre diferentes cursos y aplicaciones. Los diseños instruccionales creados a partir de este estándar adquieren un papel fundamental como proceso sistémico, estructurado y planificado. Su función no es sólo la de producir materiales educativos sino productos educativos completos, efectivos y eficaces.

El presente trabajo tiene como objetivo elaborar el análisis y diseño del nivel B de la especificación IMS-LD del módulo de DI de la herramienta de autor web CRODA, desarrollada en el Departamento de Herramientas Educativas del centro FORTES de nuestra universidad utilizando el estándar IMS-LD.

El módulo necesita de la incorporación de elementos adicionales que le permitan al diseñador instruccional controlar el flujo del diseño dentro de una unidad de aprendizaje, de tal forma que permita la personalización y secuenciación.

Palabras clave: CRODA, diseño instruccional, e-learning, especificaciones, estándares, IMS-LD.

Índice

Capítulo 1: Fundamentación teórica

Introducción

El inconmensurable desarrollo de las tecnologías de la información y las comunicaciones (TIC) ha traído consigo grandes cambios en la educación. Este nuevo escenario educativo hace que la mayoría de las instituciones educacionales se esfuercen por introducir el uso de las TIC en la enseñanza y en el aprendizaje, para adaptarse a una nueva demanda social y como motor impulsor para la calidad en la educación. Como parte importante de estos cambios revolucionarios se produjo el surgimiento de nuevos términos como es el *e-learning*, que se traduce como aprendizaje electrónico. "*El e-learning es un conjunto de tecnologías, aplicaciones y servicios orientados a facilitar la enseñanza y el aprendizaje a través de Internet/Intranet, que facilita el acceso a la información y la comunicación con otros participantes*".(TTnet and Hernández 2005)

"Diseño Instruccional (DI), proceso sistémico y sistemático por medio del cual a partir del análisis de una necesidad de aprendizaje, se seleccionan y desarrollan las actividades y recursos para satisfacerla, así como los procedimientos para evaluar si dicho aprendizaje fue alcanzado."(Gil 2009)El DI se introduce al entorno del e-learning debido a que es una herramienta esencial para incrementar la calidad de los cursos y programas de formación que no ha sido suficientemente atendida en la teleformación (educación a distancia). (Orellana, Suárez et al. 2001)

Estos producen una variedad de materiales educativos adecuados a las necesidades de los estudiantes, con el fin de mejorar la calidad del aprendizaje. Requiere de la evaluación constante de los resultados, así como de la adaptación a nuevas circunstancias del ambiente de aprendizaje, características y necesidades de los estudiantes y del entorno en general, donde se contextualiza el proceso educativo.

Existen diferentes modelos de diseño instruccional, la aplicación de uno u otro dependen principalmente de los contextos y necesidades específicas de los entornos donde se abordan, así como del grado de complejidad, profundidad y amplitud de los productos educativos. Sin embargo, todos tienen elementos comunes como el establecimiento de metas y objetivos instruccionales. Estos modelos hacen más efectivo el proceso de aprendizaje mediante la realización de una planificación adecuada propiciando así la adquisición de las destrezas necesarias para la utilización de las tecnologías en las actividades que se llevan a cabo.

Capítulo 1: Fundamentación teórica

El *Instructional Management System – Learning Desing* (IMS - LD), es actualmente el estándar más versátil y respaldado para modelar procesos completos de aprendizaje como complemento de estructuras de contenidos educativos.IMS –LD es un lenguaje de modelado educativo que tiene como objetivo definir formalmente una estructura semántica para anotar los procesos de enseñanza y aprendizaje, y así convertirlos en entidades reutilizables entre diferentes cursos y aplicaciones.(Berlanga 2005)

Un Diseño de Aprendizaje (*Learning Desing*) es definido en la especificación IMS como "*una descripción de un método que permite a los alumnos alcanzar ciertos objetivos de aprendizaje por medio del desarrollo de ciertas actividades de aprendizaje en un cierto orden en el contexto de un cierto ambiente de aprendizaje*".(IMS 2003)

También se le define como la aplicación de un modelo pedagógico para un objetivo de aprendizaje, un grupo objetivo y un contexto determinado.(Koper and Olivier 2004)En realidad, *IMS Learning Desing* permite que a través de una especificación aceptada públicamente se exprese un proceso de enseñanza-aprendizaje, lo que equivale a describir las actividades y los recursos de ellas, que serán utilizadas y desarrolladas por personas (alumnos y tutores) para permitir que los alumnos alcancen un objetivo educacional esperado.

Esta especificación de IMS plantea que el Diseño de Aprendizaje, es decir, la descripción del proceso enseñanza-aprendizaje unido a los recursos físicos que dicho proceso hará uso, incluyendo los objetos de aprendizaje, se puede "empaquetar" en una única entidad llamada *Unit of Learning* también conocida como UoL, que traducida a nuestro idioma sería llamada "Unidad de Aprendizaje".

La Universidad de las Ciencias Informáticas (UCI) como institución educacional y centro de desarrollo no está exenta de la creación y utilización de herramientas y plataformas que apoyan el *e-learning* teniendo en cuenta los diferentes estándares que se aplican. En el Departamento de Producción de Herramientas Educativas perteneciente al Centro de Tecnologías para la Teleformación (FORTES) de la universidad se desarrolla la herramienta de autor web CRODA, actualmente en su versión 2.0. CRODA es una herramienta de autor web que permite la creación de objetos de aprendizaje (OA) reutilizables, accesibles, duraderos e interoperables empleando el estándar SCORM1.2. Entre algunas de sus funciones se encuentran la creación de diferentes tipos de preguntas, desarrollo de plantillas con la estructura inicial que puede tener algún OA, además la posibilidad de poder modificarlas y hacerlas públicas a los demás que interactúen con la herramienta. Posteriormente se incluyeron funcionalidades para apoyar el trabajo colaborativo.

Capítulo 1: Fundamentación teórica

El módulo de Diseño Instruccional de la herramienta de autor web CRODA necesita de la incorporación de elementos adicionales que le permitan al diseñador instruccional controlar el flujo del diseño dentro de una unidad de aprendizaje, de tal forma que permita la personalización y secuenciación de las mismas.
El editor de UoL actualmente permite crear estas unidades en su primer nivel, limitando a que los diseños creados presenten un flujo de desarrollo estático. Esto trae consigo un diseño de aprendizaje con una secuencia de actividades poco elaborada y no proporciona garantía alguna del tránsito de los estudiantes por estas actividades, que pueden ser de desarrollo obligatorio. Por lo que no se adecua correctamente a las necesidades de cada estudiante. Además, no facilita la creación de modelos que permitan a un rol como el profesor monitorear a sus estudiantes durante el desarrollo del diseño creado.
Son firmes los argumentos que demuestran la falta de posibilidades de personalización y secuenciación de las UoL creadas en el módulo Diseño del Instruccional de la herramienta de autor web CRODA.

Por todo lo anteriormente expuesto se plantea como **problema de investigación**: ¿Cómo contribuir a la personalización y secuenciación de una unidad de aprendizaje durante su creación en el módulo de Diseño Instruccional de CRODA 2.0?

Se puede definir entonces como **objeto de estudio**, el proceso de creación de secuenciación y personalización en recursos digitales estandarizados dentro el campo de *e-learning*.

Como **campo de acción,** la secuenciación y personalización de las Unidades de Aprendizaje en CRODA.

Para darle solución al problema se plantea como **objetivo general**, elaborar el análisis y diseño del nivel B de la especificación IMS-LD del módulo de Diseño Instruccional de la herramienta de autor web CRODA 2.0.

A partir del objetivo general se desglosan los siguientes **objetivos específicos:**

- Analizar el funcionamiento del nivel B en editores de IMS – LD.
- Realizar análisis del nivel B de la especificación IMS-LD para el módulo de Diseño Instruccional en CRODA 2.0.

- Realizar diseño del nivel B de la especificación IMS-LD para el módulo de Diseño Instruccional en CRODA 2.0.

En vistas de guiar el desarrollo de este trabajo se toma como **idea a defender** que, con la elaboración del análisis y diseño del nivel B de la especificación IMS-LD del módulo de Diseño Instruccional para la herramienta de autor web CRODA 2.0, se crearán las bases para la implementación de funcionalidades que permitan la secuenciación y personalización de una unidad de aprendizaje durante su creación.

Para dar cumplimento a los objetivos expuestos se plantean las siguientes **tareas a cumplir:**

1. Realización de un análisis bibliográfico para conformar la base teórica de la investigación.
2. Análisis del funcionamiento del nivel B en editores de IMS – LD.
3. Análisis de los flujos de trabajo de ingeniería de requerimientos, análisis y diseño de la metodología de desarrollo utilizada en la investigación del nivel A, para la futura investigación del nivel B.
4. Realización la especificación de requisitos funcionales y no funcionales.
5. Realización del análisis del nivel B de la especificación IMS-LD para el módulo de Diseño Instruccional en CRODA 2.0.
6. Elaboración del diseño del nivel B de la especificación IMS-LD para el módulo de Diseño Instruccional en CRODA 2.0.

Con el objetivo de proveer las bases necesarias de la investigación se hizo necesaria la utilización de diversos **métodos de investigación**.

Métodos teóricos:

- **Analítico – sintético**: permitió el estudio y análisis de los diferentes conceptos y definiciones permitiendo la extracción de los elementos más importantes relacionados con el proceso de creación de la secuenciación en e-learning.
- **Análisis Histórico – lógico**: permitió constatar teóricamente cómo ha evolucionado el proceso de creación de la secuenciación en e-learning en diferentes editores, sus características y ventajas, para evaluar posibles mejoras a incluir en nuestra aplicación.

Posibles resultados:

- Obtener la especificación de requisitos funcionales y no funcionales para el nivel B de la especificación IMS-LD del módulo de DI en CRODA 2.0.
- Obtener los artefactos de la fase de análisis y diseño de RUP para el desarrollo del nivel B de IMS-LD del módulo de DI en CRODA 2.0.

Esta investigación se desglosa en tres capítulos, en los cuales se abordan de forma simplificada los siguientes contenidos:

Capítulo I Fundamento teórico: Se aborda de forma general los principales conceptos a tener en cuenta en la investigación. Se hace un estudio de la especificación IMS – LD y de las diferentes vías de cómo desarrollar la secuenciación dentro de una UoL en editores IMS – LD. Además, se hace un análisis de las herramientas y metodologías utilizadas para el desarrollo del análisis y diseño del módulo con el objetivo de proponer el más indicado para su utilización.

Capítulo II Características del sistema: Se hace una descripción de la solución que se propone, donde se presenta primeramente el modelo de dominio para comprender el ámbito donde se desarrolla este módulo. Se desarrolla el flujo de trabajo Requerimientos de la metodología RUP, exponiendo básicamente los requisitos funcionales, llegando así a los casos de uso necesarios, donde además se realiza una descripción textual de los mismos y se efectúa un diseño de los prototipos de interfaz de usuario de cada caso de uso identificado.

Capítulo III Análisis y diseño del sistema: Se enmarca principalmente en el flujo de trabajo Análisis y Diseño de la metodología RUP, donde se observa de forma general el cumplimiento de los objetivos definidos anteriormente. Se presentan los diagramas de clases del análisis y los diagramas de colaboración mediante los cuales se logra una comprensión más precisa de los requisitos y como resultado final se desarrolla el modelo de diseño. Se realiza además la validación de los requisitos y casos de uso utilizando los métodos de métricas para la especificación de requisitos y métricas para la especificación de casos de uso, lo que permite constatar que estos artefactos están correctamente descritos.

Capítulo I Fundamentación teórica

1.1. Introducción

En el presente capítulo se relacionan los conceptos necesarios para poder concebir el módulo de Diseño Instruccional de la herramienta de autor CRODA, basado en la especificación IMS – LD. Se realiza un estudio de las diferentes herramientas de autor que basan su funcionamiento en Diseño Instruccional (DI), y se observa cómo se aplica el nivel B en las mismas, ya que sirve de base para el desarrollo de la propuesta de solución. Además, se presenta un análisis de las diferentes tecnologías y herramientas necesarias, con el objetivo de proponer las más indicadas.

1.2. Diseño instruccional

En la actualidad la utilización de Internet y diversos modelos de enseñanza que vinculan medios electrónicos para el aprendizaje se hacen prácticamente imprescindibles por la disponibilidad de contenidos y materiales que los mismos ofrecen. Estos proporcionan gran variedad de facilidades y materiales educativos, pero han provocado un problema para los estudiantes, el cual consiste en cómo aprovechar de mejor manera estas potencialidades a la hora de asimilar los conocimientos. El diseño instruccional surge para satisfacer las necesidades de los estudiantes.

"Diseño instruccional, en su definición más sencilla, es un proceso sistemático, planificado y estructurado donde se produce una variedad de materiales educativos atemperados a las necesidades de los educandos, asegurándose así la calidad del aprendizaje."(Yukavetsky 2008)

Los modelos de diseño instruccional comprenden un análisis exhaustivo de las necesidades y metas educativas necesarias, a continuación se realiza el diseño e implementación de los métodos necesarios para hacer efectivos los objetivos planteados. Estos métodos están compuestos por actividades y materiales instruccionales, así como pruebas y evaluaciones para verificar las destrezas adquiridas por los estudiantes.

1.3. Especificación IMS - Learning Desing (IMS - LD)

IMS Global Learning Consortium Inc. publica en el año 2003 la especificación *IMS – Learning Desing* desarrollada por la *Open University of Netherlands,* con una amplia gama de especificaciones IMS se

enfocó en desarrollar una especificación que estuviera centrada en el proceso de aprendizaje sin importar el modelo pedagógico, lo que significa, que no tiene modelo pedagógico asociado y puede ser utilizado como meta-modelo ya que cuenta con la capacidad de asimilar cualquier escenario de aprendizaje, intentando asegurar la interoperabilidad de los módulos o UoL que genera.

Esta especificación, pedagógicamente neutra como se explicaba anteriormente tiene su basamento en un lenguaje de modelado educativo en el que prima como objetivo definir formalmente una estructura semántica para constituir el proceso de enseñanza-aprendizaje. Específicamente, *"describe un método que va a permitir a alumnos y profesores alcanzar ciertos objetivos de aprendizaje previamente planteados mediante la realización de distintas actividades de aprendizaje, utilizando determinados recursos didácticos, en un cierto orden, bajo ciertas condiciones, en un cierto ambiente de aprendizaje."*(IMS 2003)IMS – LD refuerza la asociación entre objeto didáctico reutilizable, actividades y roles de personas que intervienen en el proceso educativo, asemejándose más al modelo apreciable en las clases presenciales.(Vázquez 2007)

Por todo lo anteriormente expuesto, y teniendo en cuenta que IMS - LD no es un estándar sino una especificación, se considera el idóneo para lograr la adaptación del diseño instruccional en la herramienta de autor web CRODA independientemente del modelo pedagógico, lo que le confiere un incremento en la flexibilidad del manejo de los OA.

En ocasiones se utilizan de forma errónea los términos estándar y especificación, es importante definir que un estándar es una tecnología, formato o método, reconocido, nacional o internacionalmente, documentado en detalle y ratificado por una autoridad respetada de su campo; por el contrario, una especificación es creada por alguna compañía u organismo, que no ha sido ratificada todavía por ninguna autoridad, y que suele utilizarse de manera provisional pero está suficientemente respaldada.(Burgos 2005)

Según(Singh and Reed 2002), no hay estrictamente un estándar *e-learning*, sino grupos desarrollando especificaciones. La adquisición de conocimientos no depende solamente del diseño instruccional o de la estructura que proporciona la especificación IMS – LD, por excelente que sea el curso la retención de la información y conocimientos depende en gran medida de la utilización que se le dé al curso y del provecho que los estudiantes y profesores sean capaces de obtener del mismo.

1.3.1. Funcionamiento general de IMS – LD

Una unidad de aprendizaje según (Koper and Tattersall 2005)es "*Una unidad de educación o formación completa y autónoma, curso, módulo o lección. La creación de una unidad de aprendizaje implica la creación de un diseño de aprendizaje y también la compilación de sus recursos asociados, bien como archivos contenidos en la unidad o como referencias web, incluyendo evaluaciones, materiales de aprendizaje e información para configurar el servicio de aprendizaje*".

IMS - LD no incluye los materiales o recursos que utiliza. Una UoL representa la unidad completa de formación y auto-contenida como, por ejemplo, un curso o una lección. En sí, una UoL contiene todos los recursos asociados como ejercicios, objetos u recursos de aprendizaje e información para configurar distintos servicios (ej. conferencia, correo electrónico, búsqueda).

Un escenario de aprendizaje completo viene definido por los recursos, la metodología, las actividades de aprendizaje y soporte, los servicios complementarios, la relación entre los roles, la agrupación de usuarios, entre otros. Todo ello se define en un fichero llamado manifiesto, escrito en XML, que junto con los recursos es empaquetado en un fichero en formato ZIP, consiguiendo la Unidad de Aprendizaje (UoL). Este fichero comprimido podrá ser ejecutado, abierto o modificado en cualquier sistema compatible LD. (Burgos 2005)

Para ejemplificar los conceptos expuestos hasta el momento se proponen los siguientes esquemas que muestran un paquete de contenidos *IMS Content Packaging* (*Ver Figura 1y 2*).

Figura 1-Estructura de un paquete de contenido IMS.

Figura 2 -La estructura de una Unidad de Aprendizaje (UoL) en IMS Learning Desing.

1.3.2. Niveles de IMS – LD

La especificación *IMS Learning Desing* se desglosa en tres niveles de aplicación y cumplimiento. La especificación se centra no en definir un esquema grande solo con un núcleo de elementos obligatorios y numerosos elementos opcionales, sino más bien para definir un núcleo completo lo más simple posible, y luego dos niveles de extensión que captura las características y comportamientos más sofisticados.

La implementación de estos niveles depende en gran medida de las entidades donde se apliquen, ya que cada una debe conocer sus necesidades específicas y en que profundidad debe aplicar la especificación. Lo que indica que no porque existan tres niveles es necesaria la implementación de todos para lograr la implementación dela especificación.

Los niveles se denominan indistintamente como 1, 2, 3 ó A, B, C esta última denominación será la utilizada de ahora en adelante.

1.3.2.1. Nivel A

El nivel A crea la base para el desarrollo de la especificación, en este nivel es donde se define el vocabulario básico necesario, se dice que es el nivel de menor abstracción. La especificación utiliza la metáfora del teatro para representar el proceso de enseñanza-aprendizaje que se realiza dentro de la misma, varios actores todos trabajando con un fin común dentro de una obra en la que pueden tomar diferentes roles y participar en diferentes actos. Este nivel es donde se especifican en detalle cada una de las características de esta obra.

El nivel A, constituye el núcleo y define las actividades de aprendizaje, las de soporte, las estructuras, los entornos, el método, la instancia, los actos, los roles, los recursos y la comunicación entre ellos. Además los usuarios podrán usar diversos recursos externos como enlaces web y algunos servicios como foros y chats para facilitar la colaboración entre estos.(Vázquez 2007)

Elementos del Nivel:(Falcon 2011)

- **Persona:** Es la persona que adopta un determinado rol.
- **Rol:** Por defecto son, aprendiz y grupo de trabajo.
- **Actividad:** Existen dos tipos de actividades, actividades de apoyo y actividades de aprendizaje, así como estructuras de actividades que agrupan dos o más de estas e incluso otras estructuras.
- **Ambiente:** El ambiente es una colección estructurada de servicios, objetos de aprendizaje y sub-ambientes.
- **Método:** El método es el mecanismo que permite relacionar los roles, las actividades y los ambientes. Se compone de tres elementos:
  - **Obra**: Un método se compone de una o más obras, y estas a su vez de uno o varios actos.
  - **Acto:** Es aquí donde se establecen las distintas actividades en Partes de Rol adecuadas. Un acto involucra a un participante o a varios, pero cada uno de ellos lleva a cabo una única actividad.
  - **Parte del Rol:** Asocia únicamente un solo rol a una única actividad, teniendo como objetivo, especificar lo que ese rol debe cumplir en el Acto.

1.3.2.2. Nivel B

Elementos del Nivel:(Falcon 2011)

- **Condiciones**: Son condiciones que permiten una personalización así como secuenciación más elaboradas.
- **Propiedades**: Son propiedades y pueden ser utilizadas para dirigir las actividades de aprendizaje así como también para registrar los resultados.

El nivel B define las propiedades y las condiciones, donde las propiedades son pares de atributo-valor que parten inicialmente con ese valor y pueden transformarse durante la ejecución de la UoL, almacenan información sobre las (preferencias, resultados, información personal), estas una vez definidas se pueden establecer las condiciones, que no son más que, consultas que se realizan sobre las propiedades en un momento determinado, cuya verdad o falsedad puede provocar la visibilidad o invisibilidad de nuevas actividades y, por tanto, la adaptación del flujo de aprendizaje a las necesidades de cada participante, estas le permiten al diseñador instruccional controlar el flujo diseño dentro de una Unidad de Aprendizaje. De esta forma el diseño de aprendizaje se modifica y permite que el flujo de actividades no sea fijo como en el nivel anterior.

Las condiciones se representan como reglas tipo si <guarda> entonces <acción-cierto> en otro caso <acción-falso>, con el significado: si <guarda> es cierta, realiza las acciones indicadas en <acción-cierto> y, si no, realiza las acciones indicadas en <acción-falso>. (*Ver Figura 3*)

Evaluar una condición supone, por tanto, evaluar su guarda y, dependiendo de si dicha guarda es cierta o falsa, ejecutar la acción correspondiente. Las condiciones se evalúan efectivamente:

- Al comienzo de la ejecución de la unidad.
- Siempre y cuando el valor de una propiedad haya cambiado.

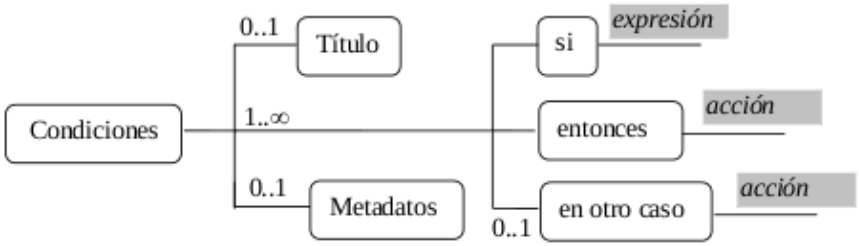


Figura 3 -Estructura de la especificación de las condiciones.

Existen diferentes tipos de propiedades (*Ver Figura 4*), el nivel puede añadir propiedades locales y globales. Las propiedades son una parte esencial de la vigilancia, la personalización, la evaluación y para la interacción del usuario.A las propiedades locales se les dominan propiedades internas ya que existen únicamente durante el proceso de ejecución de la Unidad de Aprendizaje y las propiedades globales son

también conocidas como externas porque estas pueden ser consultadas y utilizadas en diferentes UoL y los datos que almacenan permanecen en el transcurso de varias secciones.

*Learning Design*definecuatro tipos de propiedades: propiedades locales,propiedades globales, propiedades personales y propiedades de roles.

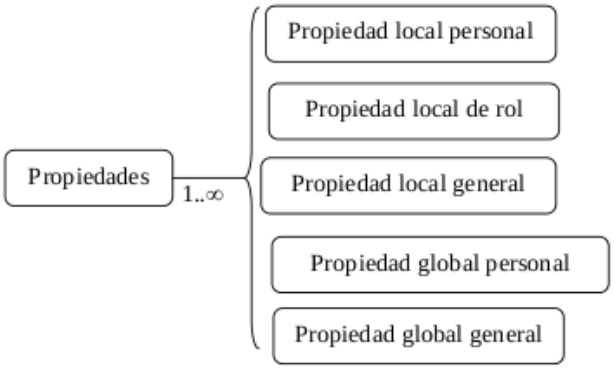


Figura 4 - Estructura de la especificación de las propiedades.

A continuación se detallan cada una de ellas:

- **Propiedades locales:** se almacenan con un alcance local para la ejecución de una unidad de aprendizaje. Se definen y se utilizan en la unidad de aprendizaje. El valor de esta propiedad es el mismo para todos los usuarios en el funcionamiento de la unidad de aprendizaje, pero pueden variar en diferentes ejecuciones.
- **Propiedades globales:** son accesibles fuera del contexto de una unidad de aprendizaje, es decir en más de una unidad de aprendizaje. Se pueden definir en una unidad de aprendizaje y utilizarse en otra. Las propiedades globales una vez definidas no se puede cambiar la definición.
- **Propiedades personales:** son propiedad de una persona (local o global). Estas propiedades se utilizan para la personalización. Por ejemplo, una cartera que funciona a través de unidades de aprendizaje puede ser modelado con propiedades globales personales. Las propiedades personales pueden ser almacenados en un expediente portátil personal.
- **Propiedades de roles:** son propiedad de una función y son siempre locales. Todos los usuarios en un rol específico puede acceder a esta propiedad y tiene el mismo valor en el mismo plazo de la unidad de aprendizaje.

La aplicación en cuestión trabajará específicamente con las propiedades locales, las propiedades personales y las propiedades de roles. Este nivel brinda flexibilidad, ya que posibilita condicionar el flujo de apredizaje, almacenar datos del usuario y de la instancia tanto a nivel local como personal, además de que permite mostrar o esconder estos elementos en el momento deseado.

Los tipos de datos siguientes se utilizan en la declaración de las propiedades. El formato de cada tipo de datos se especifica:

- **Boolean:** Representa la lógica binaria, verdaderas o falsas (alias: sí / no; 1 / 0). Como todos los otros tipos de datos, booleanos pueden tener <no-value>.
- **Entero:** Representa el concepto estándar de matemáticas de números enteros, en representación de números enteros positivos y negativos incluyendo el cero.
- **Real:** Representa el concepto estándar de matemáticas que representan los números decimales de precisión arbitraria, y debe ser capaz de manejar un número de 18 cifras decimales como mínimo.
- **Cadena:** representa cualquier cadena de caracteres legal. El número máximo de caracteres es de 2000.
- **Archivo:** Representa cualquier tipo de datos como archivo binario. La propiedad almacena este archivo.
- **Fecha y hora:** especifica la fecha y hora en el formato: AAAA-MM-DDThh: mm: ss.
- **Duración:** especifica una cantidad de tiempo: la duración de un evento en términos relativos (por ejemplo, la duración, dada la fecha y hora inicio de la ejecución de una unidad de aprendizaje El formato - también se utiliza en la especificación W3C XML Schema - es.: PnYnMnDTnHnMnS where: PnYnMnDTnHnMnS donde:

P es el indicador que debe estar siempre presente.

n es una variable que está lleno de un número entero in

nY representa el número de años.

nM representa el número de meses.

nD representa el número de días.

T es el separador de fecha / hora, que debe estar siempre presente cuando se representa el tiempo.

nH es el número de horas.

nm es el número de minutos.

NS es el número de segundos.

- **Texto:** representa cualquier cadena de caracteres legal. El número máximo de caracteres es de 64.000 (alrededor de 10 páginas de texto A4).

Para las propiedades se pueden especificar ciertas restricciones en los valores de los datos:

- **longitud:** limita la longitud del valor de la propiedad de un tipo de datos textual (cadena, texto) en términos del número de caracteres que puede tener.
- **minLong:** restringe el número mínimo de caracteres que una propiedad de tipo de datos textuales pueden tener.
- **maxLong:** limita el número máximo de caracteres que una propiedad de tipo de datos textuales pueden tener.
- **enumeración:** limita el valor de una propiedad a un valor determinado (el uso de listas alternativas de valor).
- **maxInclusivo:** limita el valor de un orden (entero, fecha y hora real,) la propiedad a un límite superior específico incluido.
- **minInclusivo:** limita el valor de una propiedad de la orden de un límite inferior específico incluido.
- **maxExclusivo:** limita el valor de una propiedad de la orden de un superior específico exclusivo obligado.
- **minExclusivo:** limita el valor de una propiedad de la orden de un límite inferior exclusivo específico.
- **totalDigitos:** limita el valor de una propiedad decimal a un número específico de dígitos que debe contener.
- **fracciónDígitos:** limita el valor de una propiedad decimal para el número máximo de dígitos que puede haber después del punto decimal.
- **patrón:** limita los literales que comprende el valor de una propiedad a un patrón definido por una expresión regular.

1.3.2.3. Nivel C

El nivel C incorpora un sistema de notificaciones (envío de mensajes) a la aplicación, donde estas se ejecutan automáticamente para dar respuesta a eventos que se producen dentro del proceso de enseñanza-aprendizaje.

Con esta opción las actividades pueden ser interrumpidas en plena ejecución de ser necesario incluso por el propio *Learning Management System* (LMS). Con ello es posible realizar flujos de aprendizaje modificables en tiempo real y así alterar la secuenciación de las actividades.

Elementos del Nivel:(Falcon 2011)

- **Notificación**: Son notificaciones, es decir, acciones asociadas a eventos que se disparan automáticamente. Se originan por el resultado de una actividad y pueden producir un comportamiento distinto al esperado en una actividad en tiempo real.

1.4. Herramientas de autor

Las herramientas de autor son aplicaciones que permiten la creación de contenidos educativos como cursos o lecciones que sirven de apoyo a profesores y estudiantes en el proceso de enseñanza-aprendizaje, adecuándolo a las necesidades específicas que el autor desee. Estas permiten controlar el aprendizaje, mediante cuestionarios y marcadores que evalúan el conocimiento que el estudiante vaya adquiriendo.

Estas se pueden clasificar en tres tipos, las que permiten la creación de materiales educativos digitales, las que pueden generar todos los materiales a incluir en el curso y su publicación y las que generan simulaciones. (Montero, J. L; Herrero, E, 2008).

En la actualidad existen una gran variedad de herramientas de autor dedicadas a la creación de contenidos educativos todos compatibles con la especificación IMS-LD, entre las que se pueden mencionar ALFANET, CopperAuthor, RELOAD, CoS-MoS, Collage y ReCourse. A continuación se realiza un análisis más detallado de la herramienta RELOAD, ReCourse y CopperAuthor con las cuales se pueden crear diseños educativos IMS-LD de los tres niveles (del nivel A al nivel C), las otras en su inmensa mayoría solo alcanzan el nivel A.

1.4.1. Reload

Reload es una herramienta de autor que permite la creación de contenidos educativos y posibilita el DI. Concretamente, Reload crea y edita paquetes e inserta metadatos conforme a las especificaciones de ADL e IMS. Con esta herramienta es posible confeccionar y ver los paquetes en un navegador web. Utiliza el estándar SCORM y permite comunicación con un LMS.

Este editor permite la edición de una UoL mediante la interacción con múltiples formularios y estructuras en forma de árbol que representan la estructura de agregación de los conceptos de IMS-LD implícita en el formato XML utilizado para representar las UoL de IMS-LD. Con esta herramienta se pueden crear diseños educativos IMS-LD de los tres niveles (del nivel A al nivel C).(ITE 2010)

1.4.2. ReCourse

ReCourse es un Editor *IMS Learning Desing*, patrocinado por el proyecto *TENCompetence*. Este es una herramienta para crear unidades de aprendizaje compatibles con *IMS Learning Desing*. La especificación *IMS Learning Desing* es muy extensa y compleja, existen muy pocas herramientas disponibles efectivas y fáciles de utilizar que los profesores pueden usar para producir unidades de aprendizaje que sean compatibles con la especificación.(ITE 2010)

Algunas características claves del Editor de Diseño de Aprendizaje ReCourse:

- Una interfaz de creación simplificada. Los autores pueden crear la estructura principal de la Unidad de Aprendizaje de una red de aprendizaje de flujo único.
- Soporte para flujo de trabajo de los autores.
- Las UoLs ahora pueden ser publicadas y se rellenan con los alumnos directamente desde el editor de recursos. Con un solo clic se inicia el UoL publicado en un navegador. Los usuarios también pueden inscribirse en UoLs, si tienen permiso.
- El acceso a los repositorios se ha integrado, por lo que los autores pueden encontrar y almacenar los UoLs que están trabajando.

1.4.3. CopperAuthor

CopperAuthor es una herramienta desarrollada en paralelo al motor de ejecución CopperCore. Esta herramienta permite a los diseñadores construir y navegar sobre la estructura del diseño educativo

mediante una interfaz basada en tablas. En su versión actual la interfaz es un tanto primitiva y permite el desarrollo de diseños educativos IMS-LD de los 3 niveles.(ITE 2010)

La herramienta de autor CopperAuthor define los usuarios, las actividades de aprendizaje, las actividades de soporte y demás elementos de correspondientes al nivel A acorde con la especificación. Ofrece funcionalidades para incorporar el servicio de monitorización y elementos globales, lo que permite al usuario definir estructuras más complejas. Brinda flexibilidad a la hora de la representación didáctica, ya que permite esconder y mostrar elementos, condicionar el flujo de aprendizaje, almacenar datos del usuario, tanto a nivel local como personal. Pero todo a un nivel muy técnico con la necesidad de conocer la estructura y jerarquías de los elementos de la especificación.

Mediante el estudio realizado a estas herramientas se detectaron algunos inconvenientes tanto en las aplicaciones como en la adaptación del nivel B de la especificación. Estas tres herramientas son aplicaciones de escritorio que no brindan a los usuarios las facilidades que estos necesitan. El tratamiento que se le da a las condiciones en las mismas es imitando un lenguaje de pseudocódigo (*If-Then-Else*) los pseudocódigos de los lenguajes de programación, algunas lo adaptan de forma más sencilla que otro pero de igual forma resulta complejo de comprender para los usuarios. La herramienta ReCourse de las antes explicadas es la de mejor entendimiento ya que proporciona una interfaz más amigable, no siendo así CopperAuthor.

Los creadores de unidades de aprendizaje, que pueden no tener conocimiento de LD, necesitan ser capaces de representar su experiencia pedagógica y los planes de instrucción mediante el uso de sus vocabularios convencionales como el curso, la clase o la presentación.

Para estos profesores sin conocimientos previos de informática y de la especificación, dichas herramientas poseen una organización un tanto compleja y un vocabulario no adecuado.

1.5. Herramientas y tecnologías a utilizar

Para la realización del análisis y diseño del nivel B de la especificación IMS –LD se hace necesaria la descripción de un conjunto de tecnologías y herramientas, las mismas son definidas por el proyecto al cual tributa esta investigación. Además, el análisis y diseño que se propone como solución en la investigación, constituye la continuación y perfeccionamiento del desarrollo del nivel anterior. El desarrollo del nivel A constituyó el trabajo de diploma del estudiante Mario Manuel Falcón García, titulado **“**Implementación del

nivel A de IMS-LD para la herramienta de autor web CRODA", el cual servirá como base para la realización de la presente investigación efectuando las acotaciones que se consideren en caso de ser necesarias.

1.5.1. Metodología de desarrollo

En la actualidad existen un sin número de metodologías para el desarrollo de software, se hace necesaria la aplicación de los procedimientos que proponen las mismas para llevar a cabo un proceso de desarrollo de software bien estructurado, planeado y controlado, y así obtener un producto con una calidad de excelencia.

Se entiende por metodología de desarrollo *"una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software. En Inglés, Software Development Methodology (SDM) o System Development LifeCycle (SDLC).La finalidad de una metodología de desarrollo es garantizar la eficacia (ej. cumplir los requisitos iníciales) y la eficiencia (ej. minimizar las pérdidas de tiempo) en el proceso de generación de software."*(Cuaresma 2008)

1.5.1.1. RUP

Rational Unified Process (RUP por sus siglas en Inglés), es un conjunto de actividades que permiten transformar los requerimientos de un usuario en un sistema de software. *"RUP más que un proceso constituye un marco de trabajo genérico que puede especializarse para una gran variedad de: sistemas de software, áreas de aplicación, tipos de organizaciones niveles de aptitud y tamaños de software."* (Jacobson, et al., 2000)

Esta metodología define "quién" (trabajadores) debe hacer "qué" (artefactos), "cuándo" (flujo de trabajo y fases) y "cómo" (actividades) debe hacerlo.

En RUP se divide el desarrollo de software en 4 fases: Inicio, Elaboración, Construcción y Transición, estas fases son desarrolladas mediante un ciclo de iteraciones las cuales son llevadas bajo dos grupos de disciplinas conocidas también como flujos de trabajo. *(Ver Figura 5)*

RUP cuenta con numerosas características, 3 de estas lo hacen único, dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental. Existen otras no menos importantes entre las que se encuentran que utiliza UML como lenguaje de modelado, el tipo de programación que utiliza es orientado a objeto (OO). Es una metodología adaptable que permite que se le realicen los cambios que se necesite.

Esta metodología es la seleccionada para el desarrollo de la presente investigación. Son bastos los argumentos que se pueden mencionar en relación a las ventajas que proporciona para el desarrollo de un correcto proceso de software.

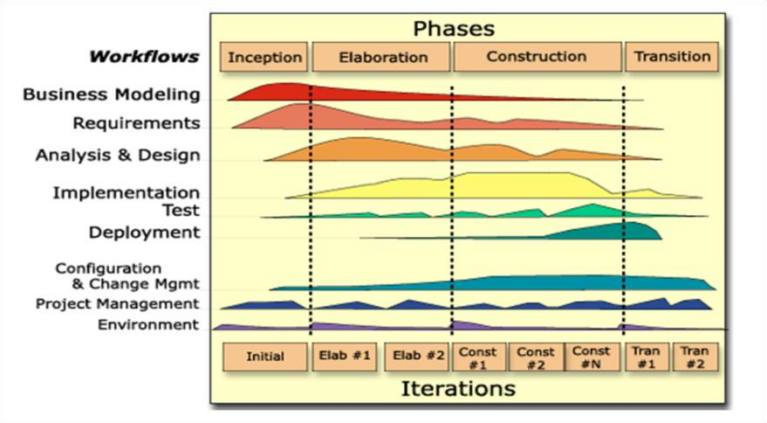


Figura 5 -Fases e Iteraciones de la Metodología RUP.

1.5.2. Herramienta CASE

La herramienta CASE seleccionada para el desarrollo de esta investigación es Visual Paradigm (VP), para dar continuidad al trabajo realizado anteriormente además de tener en cuenta las características que presenta la herramienta y las características del ambiente de trabajo.

Visual Paradigm es una herramienta multiplataforma, una característica de vital importancia ya que en el proyecto CRODA se trabaja sobre el sistema operativo Linux. Esta herramienta modela el ciclo de desarrollo de software que propone RUP completo. Permite esbozar todos los diagramas de clases necesarios, tiene integración con diferentes entornos de desarrollo (IDE), generación de código y documentación.

VP posee un diseño de interfaz gráfica, enfocado a facilitar la interacción y el trabajo del usuario. La herramienta también proporciona abundantes tutoriales, demostraciones interactivas y proyectos de UML. Es en conclusión la herramienta adecuada a las necesidades y características de la investigación en cuestión.

1.5.3. Servidores Web

"Un servidor web puede ser tanto un ordenador de grandes dimensiones y capacidad como un programa informático que utiliza el protocolo de comunicaciones http para recibir las peticiones de información de un programa cliente (navegador) en el ordenador del usuario."(Consumoteca 2009)

El Servidor Apache HTTP es un servidor web de tecnología *Open Source* sólido y para uso comercial desarrollado por la *Apache Software Foundation*, el *Red Hat Enterprise* Linux incluye el Servidor Apache HTTP versión 2.0 así como también una serie de módulos de servidor diseñados para mejorar su funcionalidad.(MIT 2010)

Entre sus principales características se encuentran:

- Multiplataforma.
- Es un servidor web conforme al protocolo HTTP/1.1
- Apache es una tecnología gratuita de código fuente abierta.
- Es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor web Apache.
- Es posible configurar Apache para que ejecute un determinado *script* cuando ocurra un error en concreto.
- Incentiva la realimentación de los usuarios, obteniendo nuevas ideas, informes de fallos y parches para la solución de los mismos.

En la propuesta de solución se utilizará Apache como servidor Web por todas las características y ventajas antes mencionadas.

1.5.4. Gestores de bases de datos

Los sistemas gestores de bases de datos se pueden definir según(Navathe 1997) como "*Un sistema software de propósito general que facilita el proceso de definir, construir y manipular bases de datos para diversas aplicaciones*".

- *Definir: Especificar los tipos de datos, las estructuras y las restricciones de los datos.*
- *Construir: Guardar los datos mismos en algún medio de almacenamiento controlado por el SGBD.*
- *Manipular: Consultar, actualizar la información almacenada.*

1.5.4.1. PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia *Berkeley Software Distribution* (BSD) y con su código fuente disponible libremente.

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

Como otras características de este gestor se encuentran:(Quiñones 2008)

- La atomicidad, esta propiedad asegura que la operación se haya realizado o no, así se asegura que ante un fallo del sistema lo que se realizaba no queda a medias.
- Consistencia en la propiedad, esto asegura que las tareas una vez empezadas se finalicen correctamente y completas.
- Aislamiento en la propiedad, asegura que una operación no puede afectar a las otras.
- Durabilidad, asegura que una vez realizadas las operaciones, aunque exista fallas en el sistema estas se mantienen persistentes.
- Se ejecuta en casi todos los sistemas operativos, tales como Linux, Unit, OS, Beos, Windows, etc.
- Diversidad de documentación, amplia, organizada, libre y además pública.

Por todas las características antes mencionadas y las ventajas que nos proporciona PostgreSQL al ser un sistema multiplataforma, se propone el mismo para el manejo de la base de datos de la investigación en cuestión.

1.5.4.2. Exist-DB

Actualmente debido a la diversidad de información existente algunos datos que trabajamos necesitan un tratamiento especial, existen un grupo de gestores de bases de datos que se encargan del almacenamiento y tratamiento de estos datos.

Puesto que existe un incremento del uso de tales datos específicos, como son los XML, y de la gran utilidad de estos, se ha hecho necesario el trabajo con respecto a ellos. Así, en los últimos tiempos, han aparecido un gran número de bases de datos que permiten almacenar documentos XML, entre otros. (Meier, 2002)

Exist-db es un sistema gestor de base de datos de código abierto que almacena los datos mediante estructuras jerarquizadas, para ello utiliza el formato XML, es una excelente plataforma para el desarrollo de aplicaciones basadas en web.
Exist-db es compatible con muchos estándares de la tecnología Web 2.0 incluyendo la especificación IMS-LD, la cual concibe las Unidades de Aprendizaje en un fichero compactado en formato .rar donde sus datos están escritos en XML. Este es el sistema gestor de base de datos seleccionado para el almacenamiento de estos archivos.

1.5.5. Framework de desarrollo

Symfony es un completo framework, una biblioteca de clases de cohesión escrito en PHP. Proporciona una arquitectura, componentes y herramientas a los desarrolladores para construir aplicaciones web complejas más rápido. La elección de Symfony permite liberar sus aplicaciones anteriores, el anfitrión y la escala sin problema, y mantenerlos en el tiempo con una sorpresa.
Symfony es un framework diseñado para optimizar el desarrollo de aplicaciones web por medio de varias características claves. Para empezar, separa las reglas de una aplicación web de negocios, lógica del servidor, y las vistas de presentación. Contiene numerosas herramientas y clases dirigidas a acortar el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes para que el desarrollador pueda concentrarse por completo en los detalles de una aplicación. El resultado final de estas ventajas es que no hay necesidad de reinventar la rueda cada vez que una nueva aplicación web se construye. Es compatible con la mayoría de los motores de bases de datos disponibles, como MySQL, PostgreSQL, Oracle, y Microsoft SQL Server. Se ejecuta en * nix y Windows. (Zaninotto 2007)

Por toda esta gama de potencialidades y características expuestas anteriormente se propone Symfony como base para el desarrollo. Es un framework cuyas capacidades han sido lo suficientemente probadas mundialmente y está realmente en uso para la alta demanda de sitios web.

1.5.6. Librería de javascript ExtJS

En informática, una librería es una colección o conjunto de subprogramas usados para desarrollar software. En general, las librerías no son ejecutables, pero sí pueden ser usadas por ejecutables que las

necesitan para poder funcionar correctamente. La mayoría de los sistemas operativos proveen librerías que implementan la mayoría de los servicios del sistema. Dichas librerías contienen comodidades que las aplicaciones modernas esperan que un sistema operativo provea.

Con el auge de la web existen múltiples librerías de Javascript que permiten realizar todo lo que deseemos en el navegador web. Una de estas librerías se llama ExtJS, esta permite construir aplicaciones complejas. Esta librería incluye:(Slocum, Moeskau et al. 2007)

- Componentes UI del alto performance y personalizables.
- Modelo de componentes extensibles.
- Un API fácil de usar.

Algunas de las ventajas que nos proporciona ExtJS:

- Nos permite crear aplicaciones complejas utilizando componentes predefinidos
- Provee una experiencia consistente sobre cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada uno (Firefox, IE, Safari, etc.).
- La ventana flotante que provee ExtJS es excelente por la forma en la que funciona. Al moverla o redimensionarla solo se dibujan los bordes haciendo que el movimiento sea fluido lo cual le ofrece ventaja frente a otros.

Es ExtJS la librería seleccionada para brindar a los futuros usuarios de la herramienta de autor web CRODA 2.0, una interfaz de usuario enriquecida

1.5.7. IDE de desarrollo NetBeans

Un entorno de desarrollo integrado, llamado también IDE (por sus siglas en inglés *Integrated Development Environment*), es un programa informático compuesto por un conjunto de herramientas de programación. Es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

El proyecto NetBeans consiste en un IDE de código abierto y una plataforma de aplicaciones que permiten a los desarrolladores crear rápidamente web, empresa, escritorio y aplicaciones móviles utilizando la plataforma Java, así como PHP, JavaScript y Ajax, Groovy y Grails, y C / C + +.

El IDE NetBeans es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier

otro lenguaje de programación. Es un producto libre y gratuito sin restricciones de uso, disponible para Windows, Mac, Linux y Solaris. (Domínguez-Dorado 2005)
Por las características antes mencionadas se selecciona NetBeans como entorno de desarrollo.

1.5.8. Lenguajes de programación y etiquetado

1.5.8.1. PHP

PHP (*Hyper Text Preprocessor*) es un lenguaje de *scripting* que permite la generación dinámica de contenidos en un servidor web. Es de código abierto y especialmente adecuado para desarrollo web. Entre sus principales características cabe destacar su potencia, su alto rendimiento, su facilidad de aprendizaje y su escasez de consumo de recursos. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno.
Lo mejor de usar PHP es que es extremadamente simple para el principiante, pero a su vez, ofrece muchas características avanzadas para los programadores profesionales.
Por estas y otras potencialidades se selecciona a PHP para el desarrollo de esta aplicación web.

1.5.8.2. Javascript

JavaScript es un lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, cabe destacar que existe una forma de JavaScript del lado del servidor. Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF, aplicaciones de escritorio (mayoritariamente *widgets*) es también significativo. Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web.
Este lenguaje interpretado es también la base de la librería ExtJS descrita anteriormente y por tanto parte de la columna vertebral de la aplicación. Es por ello que su selección es lo más aconsejable para la adaptación y mejoras en la librería seleccionada.(Falcon 2011)

1.5.8.3. XML

XML(*eXtensible Markup Language*)es un lenguaje que permite jerarquizar y estructurar la información y describir los contenidos dentro del propio documento, así como la reutilización de partes del mismo. La información estructurada presenta varios contenidos: texto, imágenes, audio, etc. y formas: hojas de cálculo, tablas de datos, libretas de direcciones, parámetros de configuración, dibujos técnicos, etc.

Es un conjunto de reglas que se usan para definir etiquetas semánticas las cuales organizan un documento en diferentes partes. Siendo así un meta lenguaje que define sintaxis para definir otros lenguajes etiquetados estructurados. (Bray 2008)

Su principal característica es que no posee etiquetas definidas previamente, por lo que el propio autor las define como desee. Además de poseer datos compuestos de múltiples aplicaciones.

La extensibilidad y flexibilidad de este lenguaje permite agrupar una variedad amplia de aplicaciones, desde páginas web hasta bases de datos. Posibilita la gestión y manipulación de los datos desde el propio cliente web. Por otro lado es la forma en que se expresa el manifiesto de la especificación IMS – LD por lo que su empleo es imprescindible junto el gestor de base de datos Exist-DB. (Falcon 2011)

1.6. Conclusiones parciales

Con el resultado de los estudios e investigaciones realizados en este capítulo, quedan claramente expuestos los principales conceptos relacionados con el Diseño Instruccional y la especificación IMS – LD, sus características, definiciones e importancia para la educación y los procesos de formación. Se define la metodología de desarrollo de software RUP, la herramienta asociada para el modelamiento de los diagramas correspondientes al flujo de trabajo análisis y diseño de esta investigación Visual Paradigm y el lenguaje de modelado UML. Además, se realiza la propuesta de herramientas y tecnologías necesarias para la futura incorporación de esta investigación al módulo de DI de la herramienta de autor, CRODA.

Capítulo II Características del sistema

2.1. Introducción

En el presente capítulo se expone la propuesta de análisis y diseño del nivel B de la especificación IMS – LD del módulo de DI de la herramienta de autor web CRODA 2.0. Para la descripción de la solución se desarrolla el modelo de dominio, describiendo las clases y relaciones que lo conforman, con el objetivo de conformar un vocabulario común. Se desarrolla el flujo de trabajo Requerimientos de la metodología (RUP) propuesta en el capítulo anterior, donde se especifican los requisitos funcionales, se definen e identifican los actores y los casos de usos así como las relaciones que se establecen entre estos.

2.2. Modelo de Dominio

Un Modelo de Dominio es un artefacto de la disciplina de análisis, construido con las reglas de UML se presenta como uno o más diagramas de clases y que contiene, no conceptos propios de un sistema de software sino de la propia realidad física. Este modelo ayuda a comprender los conceptos que utilizan los usuarios, los conceptos con los que trabajan y con los que deberá trabajar la aplicación.

El Modelo de Dominio "*una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés*".(Fowler 1996)

Es posible capturar un mayor grado de detalle en uno de estos modelos; este modelo puede ser tomado como el punto de partida para el diseño del sistema.

El modelo de dominio que a continuación se presenta constituye una representación gráfica de los conceptos más importantes que forman parte de la versión 2.0 del módulo de DI de CRODA.

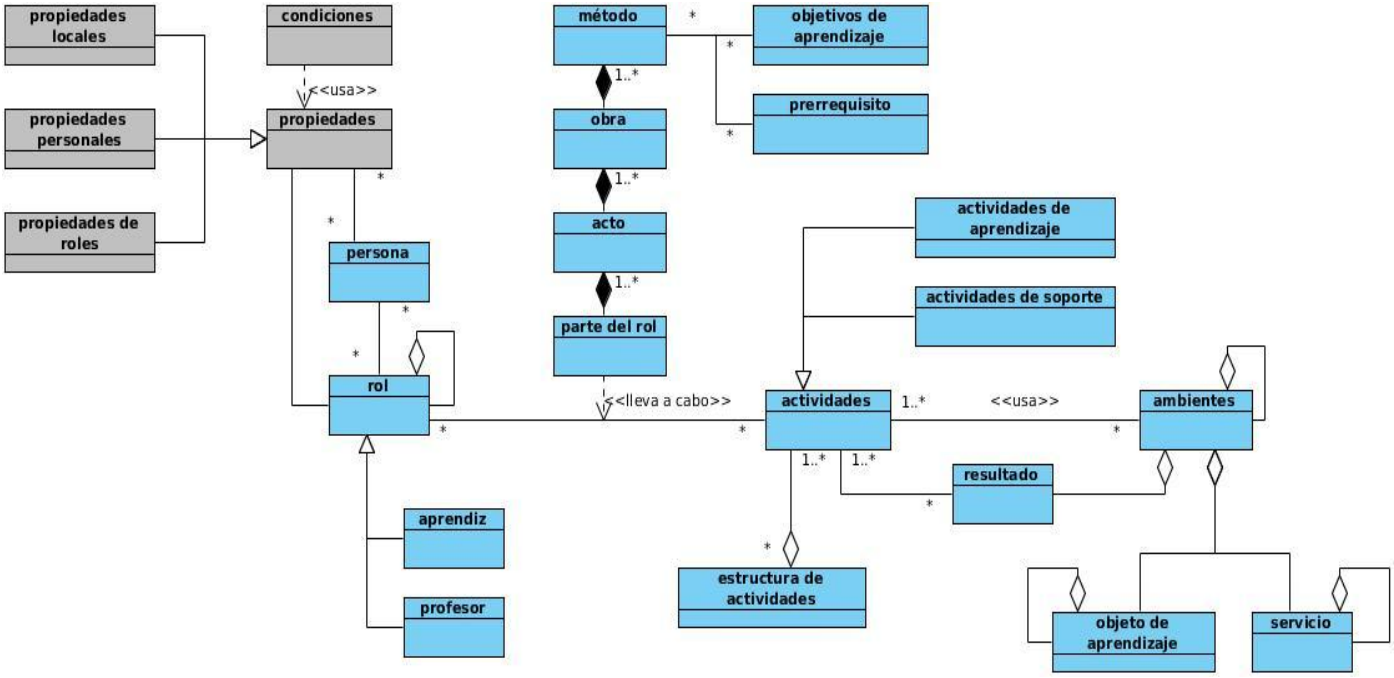


Figura 6 –Modelo de dominio.

2.2.1. Definición de los conceptos fundamentales

Propiedades: Son utilizadas para dirigir las actividades de aprendizaje así como también para registrar los resultados.

Propiedades Locales: Se crean y se mantienen únicamente dentro de cada ejecución de la unidad de aprendizaje.

Propiedades Personales: Propiedades cuyo valor puede ser distinto para cada participante.

Propiedades de Roles: Propiedades cuyo valor es el mismo para todos los miembros de un mismo rol, pero que puede variar entre distintos roles.

Condiciones: Son consultas que se realizan sobre las propiedades (previamente establecidas) que permiten una personalización así como secuenciación más elaboradas.

2.3. Requerimientos

"*Los requerimientos son una especificación de lo que debe ser implementado. Estos son descripciones de cómo el sistema se debe comportar, de las propiedades y atributos del mismo. Deben ser una restricción del proceso de desarrollo del sistema*".(Sommerville and Sawyer 1997)

Requerimientos funcionales

Los requerimientos funcionales definen qué hace el sistema, describen entradas y salidas, es decir, las funciones del sistema. A continuación se presentan los requerimientos funcionales identificados durante el proceso de levantamiento de requisitos correspondiente al flujo de trabajo Requerimientos de la metodología RUP.

RF1. Definir las propiedades.

RF1.1 El sistema debe permitir definir propiedades locales.

RF1.2 El sistema debe permitir definir propiedades locales personales.

RF1.3 El sistema debe permitir definir propiedades locales de roles.

RF2. Definir propiedad de Límite de tiempo.

RF2.1 El sistema debe permitir definir la propiedad de "límite de tiempo" en las Actividades, en los Actos, las Obras y el Método.

RF3. Definir cuando la propiedad de valor es transformado.

RF3.1 El sistema debe permitir definir completamiento, "cuando se transforme la propiedad" en las Actividades, en los Actos, las Obras y el Método.

RF4. Definir cambio de valor de la propiedad.

RF4.1 El sistema debe permitir definir al finalizar el "cambio de valor de la propiedad" en las Actividades, en los Actos, las Obras y el Método.

RF5. Definir servicio monitor.

RF5.1 El sistema debe permitir definir los Servicios en los ambientes de trabajo según la especificación, del tipo Monitor.

RF6. Definir extensión de correo electrónico.

RF6.1 El sistema debe permitir definir la ampliación de los datos del modelo de correo electrónico.

RF7. Definir condiciones de los elementos.

RF7.1 El sistema debe permitir ampliar el método de los elementos para definir las condiciones de los elementos.

RF8. Redefinir el modelo de completar acto.

RF8.1 El sistema debe permitir redefinir el modelo de completar acto para incluir cuando la condición es verdad.

Requerimientos no funcionales

Un requisito no funcional es, en la ingeniería de software, un requisito que especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos.

RNF1.Usabilidad

RNF1.1. El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora y de un ambiente Web en sentido general.

RNF1.2. Debe presentar una interfaz especial para usuarios no expertos en informática muy amigable y fácil de trabajar y que pueda ser usada por los usuarios fácilmente.

RNF1.3. En la interacción con el software el sistema mostrará mensajes que sean de entendimiento común con el usuario, para de esta forma elevar la comunicación con el mismo.

RNF1.4. En los campos que sean complejos se mostrarán notas indicando ejemplos concretos para guiar el proceso.

RNF2.Fiabilidad

RNF2.1. Una vez publicado el sistema, deberá estar disponible para los usuarios en cualquier momento.

RNF2.2. Las salidas que ofrece el sistema como resultado del manejo y procesamiento de la información estarán adecuadamente validadas durante los diferentes hitos de prueba, siendo estas fidedignas y con total exactitud a los valores esperados.

RNF3.Eficiencia

RNF3.1. Tiempos de respuestas rápidas como máximo 10 segundos al igual que la velocidad de procesamiento de la información.

RNF3.2. El sistema funcionará de manera óptima en los navegadores Web Mozilla Firefox, Internet Explorer 6 o superior y Opera.

- **Hardware**

RNF3.3. La PC del servidor debe de tener al menos 1GB de RAM.

RNF3.4. La PC del servidor debe ser Pentium 4 o superior.

RNF3.5. La capacidad de almacenamiento del disco duro del servidor debe ser como mínimo 120GB.

RNF3.6. La PC cliente debe tener como mínimo 80GB de capacidad de almacenamiento.

RNF3.7. La PC cliente debe tener al menos 1GB de RAM.

RNF3.8. La PC cliente debe ser Pentium 4 o superior.

- **Software**

RNF3.9. Se utilizará PosgreSQL versión 8.4 como gestor de base de datos.

RNF3.10. Se utilizará Exist-db versión 1.4 como base de datos de XML nativo.

RNF4. Soporte

RNF4.1. El soporte del sistema estará a cargo del Dpto. Instalación y Soporte del centro FORTES.

RNF5. Accesibilidad

RNF5.1. El sistema debe permitir aumentar y disminuir el tamaño de la letra.

RNF5.2. Las imágenes presentes en el sistema deben poseer una descripción.

RNF5.3. Las interfaces deben tener vínculos entre ellas para facilitar el acceso.

RNF6. Restricciones de diseño

RNF6.1. Para el modelado visual se deberá usar Visual Paradigm 6.4 como herramienta CASE.

RNF6.2. Se utilizará para la programación el lenguaje PHP 5.

RNF6.3. Se hará uso de Netbeans 6.9 como entorno de desarrollo.

RNF7. Interfaz de software

RNF7.1. El sistema debe interactuar con el repositorio de objetos de aprendizaje RHODA. Para ello se ofrecerá una interfaz de comunicación haciendo uso del estándar SQI e IMS-DRI. Dicha interfaz contará con la vía para acceder a las funcionalidades o información que forman parte de la integración.

- **Interfaces de usuario**

RNF7.2. El sistema debe ofrecer una interfaz fácil de operar. Igualmente tiene que mantener la línea de diseño de CRODA 1.0.

- **Interfaces de Comunicación**

RNF7.3. El sistema debe poseer una interfaz de comunicación haciendo uso del estándar SQI e IMS-DRI.

RNF8. Ayuda y documentación en línea

RNF8.1. El sistema debe contener un manual de usuario que les permita a sus usuarios documentarse para el trabajo con el mismo.

RNF9. Requisitos Legales, de Derecho de Autor, Normas o estándares aplicables al sistema, por ejemplo estándares legales, de calidad, normas de usabilidad, otros.

- La plataforma escogida para el desarrollo de la aplicación está basada en la licencia GNU/GPL.
- Para lograr la usabilidad de la herramienta de autor se empleará la norma ISO 9241-11.
- Para lograr la interoperabilidad tanto de los contenidos creados a partir de la herramienta como la interoperabilidad de la propia herramienta con otros sistemas se utilizarán los estándares SCORM y QTI; e IMS-DRI y SQI respectivamente.
- Para la descripción de los contenidos se utilizará el estándar LOM.
- Para el derecho de autor se implementará de forma obligatoria el llenado de la subcategoría Contribución en la categoría Ciclo de vida.

RNF10. Componentes incorporados al sistema.

No se necesita la compra o adquisición de algún componente para su incorporación en el sistema.

2.3.1. Definición de actores y casos de uso del sistema

2.3.1.1. Definición de actores del sistema

Actores	Descripción
Diseñador Instruccional	Es el usuario responsable de definir el diseño de aprendizaje, debe especificar qué métodos, estrategias, actividades y recursos deberá utilizar para que los estudiantes asimilen la información.

2.3.1.2. Definición de casos de uso del sistema

CU-1	**Gestionar propiedades del Diseño de Aprendizaje.**
Actor	Diseñador Instruccional
Descripción	Permite al diseñador instruccional definir las propiedades del sistema.
Referencia	RF1

CU-2	**Completar actividades de aprendizaje y/o apoyo.**
Actor	Diseñador Instruccional
Descripción	Permite al diseñador instruccional completar las actividades de aprendizaje y apoyo.
Referencia	RF2, RF3, RF4

CU-3	**Completar Acto.**
Actor	Diseñador Instruccional
Descripción	Permite al diseñador instruccional completar los actos.
Referencia	RF2, RF3, RF4

CU-4	Completar Obra.
Actor	Diseñador Instruccional
Descripción	Permite al diseñador instruccional completar las obras.
Referencia	RF2, RF3, RF4

CU-5	Completar Método.
Actor	Diseñador Instruccional
Descripción	Permite al diseñador instruccional completar los métodos.
Referencia	RF2, RF3, RF4

CU-6	Definir extensión de límite de tiempo.
Actor	Diseñador Instruccional
Descripción	Permite al diseñador instruccional definir la propiedad de límite de tiempo.
Referencia	RF2

CU-7	Definir cuando la propiedad de valor es transformado.
Actor	Diseñador Instruccional
Descripción	Permite al diseñador instruccional definir el completamiento, cuando se transforme la propiedad.
Referencia	RF3

CU-8	Definir cambio de valor de la propiedad.
Actor	Diseñador Instruccional
Descripción	Permite al diseñador instruccional definir al finalizar el cambio de valor de la propiedad.
Referencia	RF4

CU-9	Gestionar servicio Monitor.
Actor	Diseñador Instruccional
Descripción	Permite al diseñador instruccional crear, modificar o eliminar un servicio del tipo monitor en un ambiente de trabajo.
Referencia	RF5

CU-10	Gestionar extensión de correo electrónico.
Actor	Diseñador Instruccional
Descripción	Permite al diseñador instruccional definir los datos del modelo de correo electrónico.
Referencia	RF6

CU-11	Gestionar condiciones del Diseño de Aprendizaje.
Actor	Diseñador Instruccional
Descripción	Permite al diseñador instruccional definir las condiciones de los elementos.
Referencia	RF7

CU-12	Completar acto para condición.
Actor	Diseñador Instruccional
Descripción	Permite definir las características de la opción cuando la condición es verdad en el completar acto.
Referencia	RF8

2.3.2. Diagrama de casos de uso del sistema

Los diagramas de casos de uso describen las relaciones y las dependencias entre un grupo de casos de uso y los actores que participan en el proceso. Estos facilitan la comunicación con los futuros usuarios del

sistema, y con el cliente, y resultan especialmente útiles para determinar las características necesarias que tendrá el sistema. En sí, los diagramas de casos de uso describen qué es lo que debe hacer el sistema.

Como se mencionaba con anterioridad el análisis y diseño que se propone como solución en esta investigación, constituye la continuación y perfeccionamiento del desarrollo del nivel anterior. El desarrollo del nivel A servirá como base para la realización de la presente investigación. A continuación se muestran ambos diagramas, primeramente el diagrama de casos de uso correspondiente al nivel A, seguido del diagrama de casos de uso del nivel B. Este último está desarrollado sobre el anterior ya que en el nivel B se realizan muchas transformaciones a lo que previamente se había definido en el nivel A.

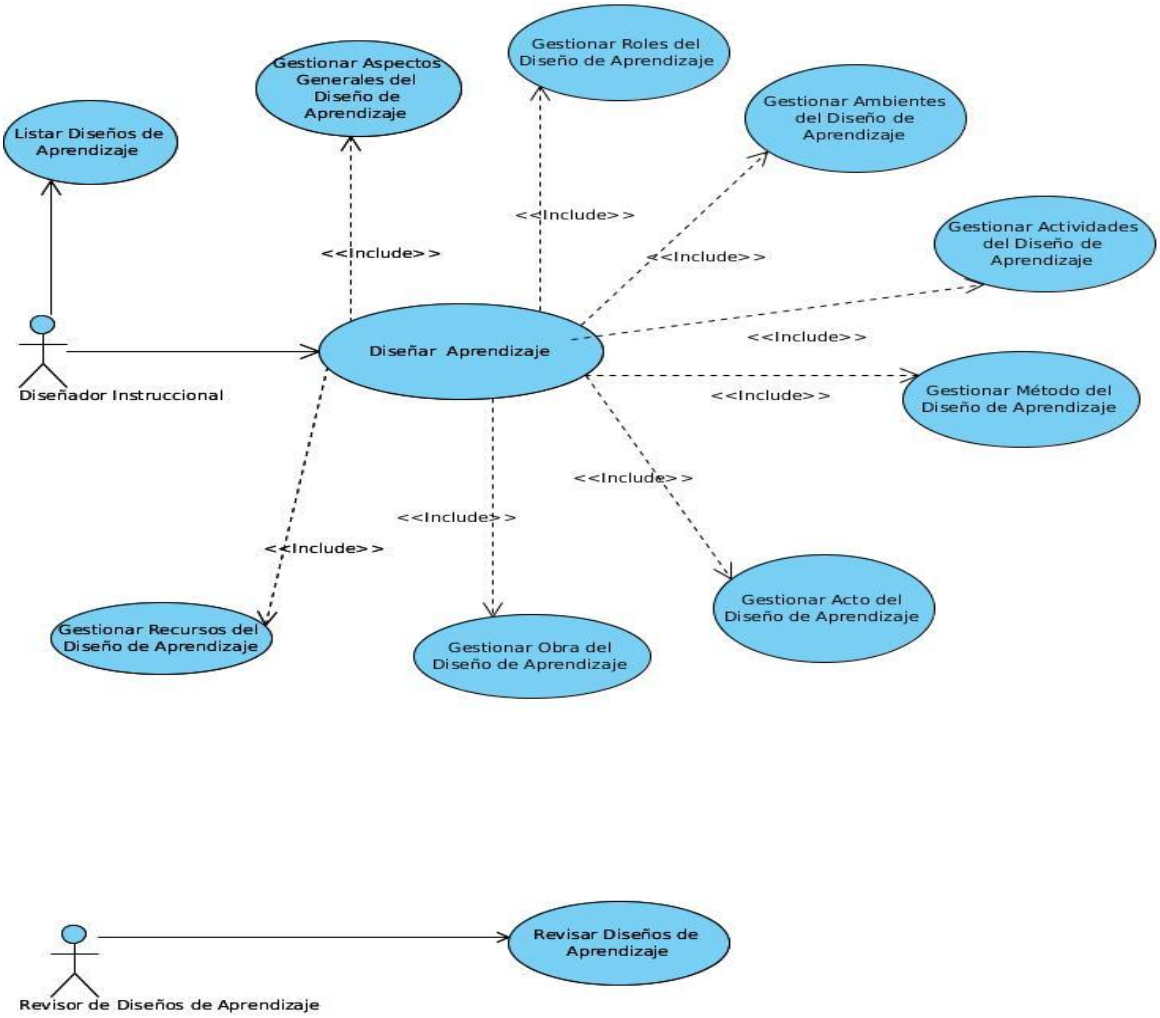


Figura 7 –Diagrama de casos de uso del sistema del nivel A.

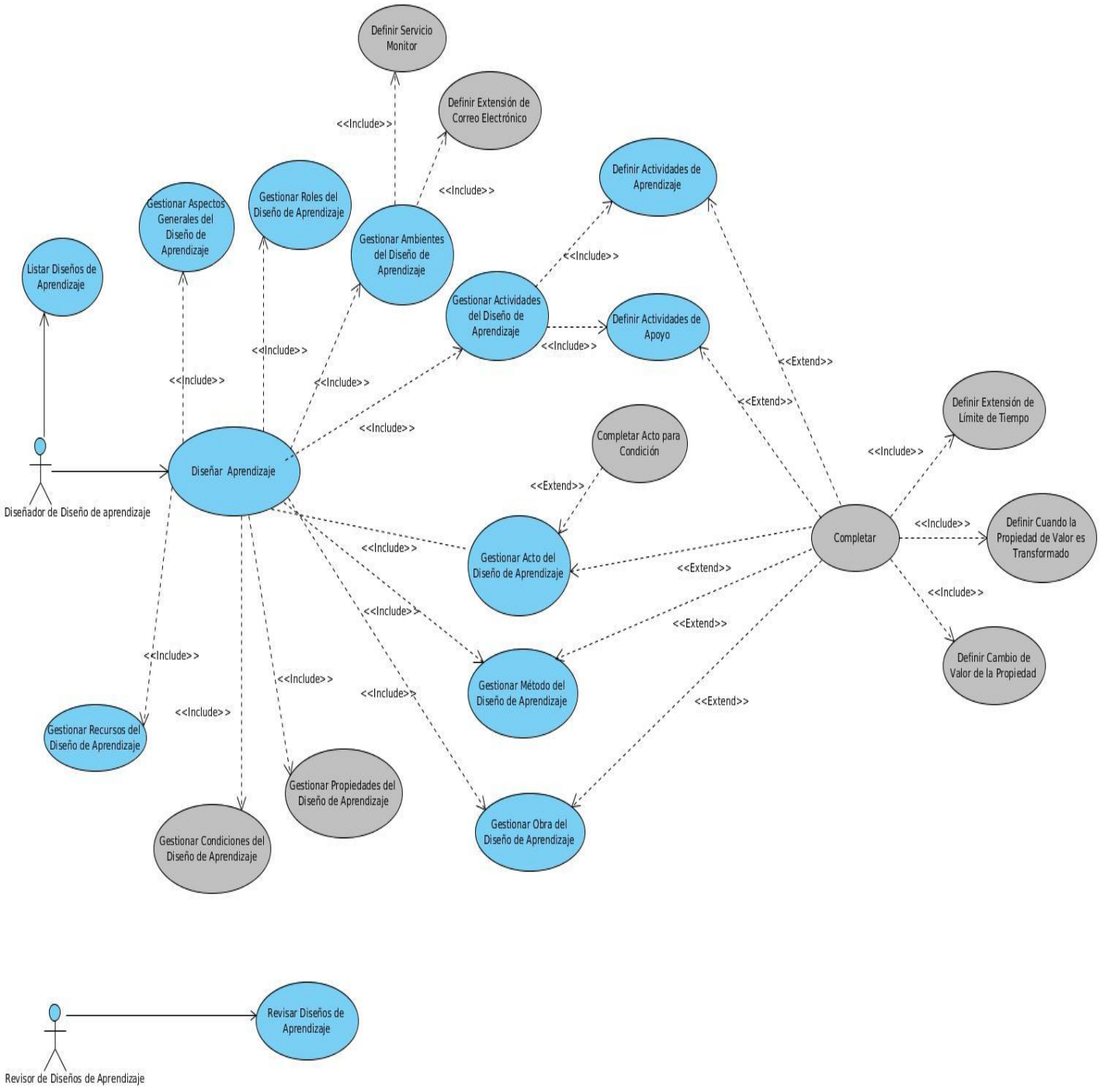
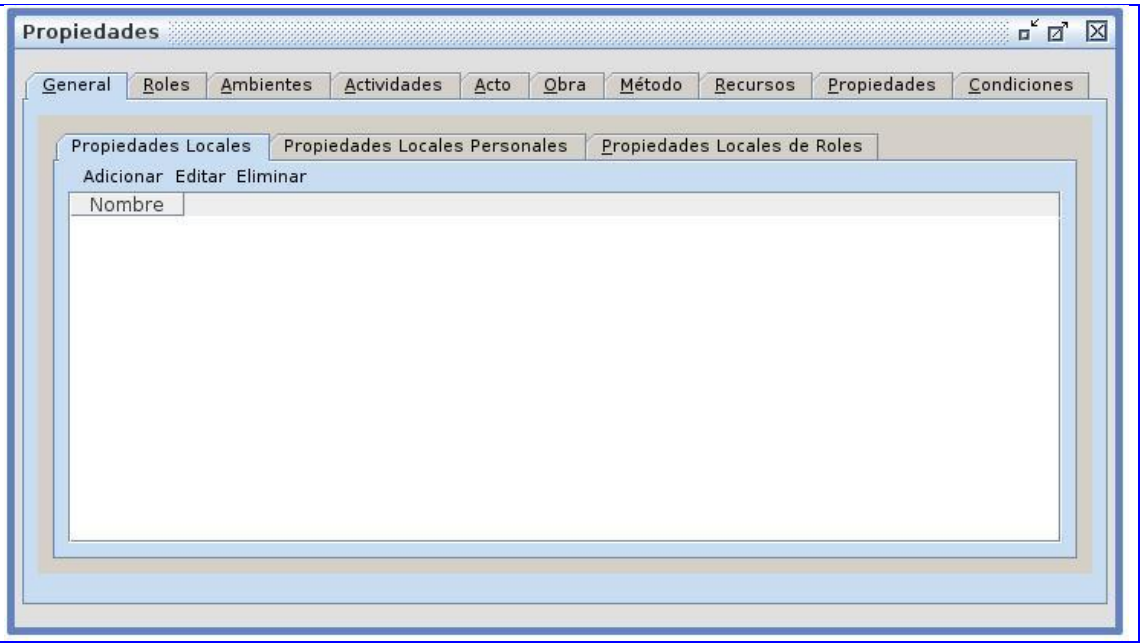


Figura 8 –Diagrama de casos de uso del sistema del nivel B.

2.4. Descripciones textuales de los casos de uso del sistema

A continuación se presentan las descripciones de los casos de usos del sistema que presenta el módulo de DI de la herramienta de autor web CRODA en su versión 2.0.

Caso de Uso:	Gestionar propiedades del Diseño de Aprendizaje.	
Actores:	Diseñador Instruccional.	
Resumen:	El CU inicia cuando el diseñador instruccional decide definir las propiedades.	
Precondiciones:	El usuario debe estar autenticado en la aplicación.	
Referencias	RF1	
Prioridad	Crítica.	
Flujo Normal de Eventos		
Acción del Actor		**Respuesta del Sistema**
1. El diseñador instruccional selecciona en la pestaña “Propiedades” el tipo de propiedad que desea, tales como: - Propiedades Locales. - Propiedades Locales Personales. - Propiedades Locales de Roles.		1.1. El sistema muestra la sección correspondiente al tipo de propiedad seleccionada.
2. El diseñador instruccional puede seleccionar cualquiera de las opciones para gestionar una propiedad del tipo que sea, en los 3 casos se gestiona las propiedades de la misma forma, las opciones se muestran a continuación: - Adicionar. (Ver sección “Adicionar”) - Editar. (Ver sección “Editar”) - Eliminar. (Ver sección “Eliminar”)		
Prototipo de Interfaz		

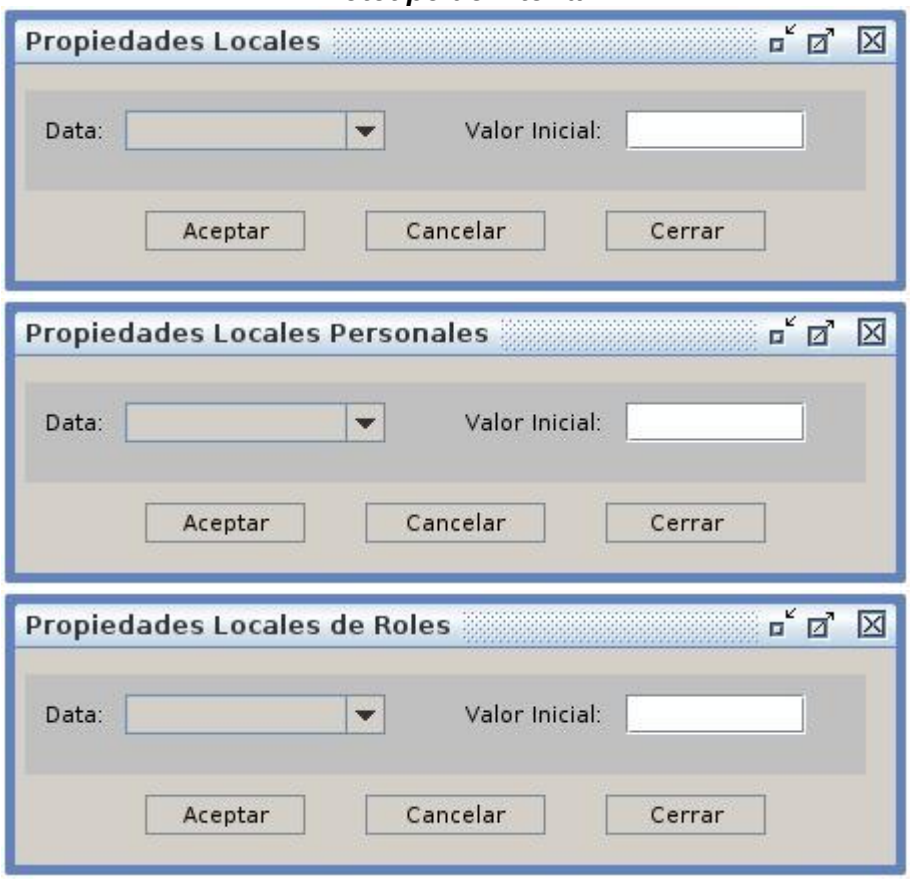


Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	
Poscondiciones	
Sección "Adicionar"	
Acción del Actor	**Respuesta del Sistema**
1. El diseñador instruccional selecciona la opción "Adicionar".	1.1. El sistema muestra una nueva ventana con los campos necesarios para crear la propiedad específica que se seleccionó. Los campos necesarios son: - Data. - Valor Inicial.
2. El diseñador instruccional introduce los datos necesarios y presiona el botón "Aceptar".	2.1. El sistema almacena el tipo de propiedad y regresa a la sección anterior.
Prototipo de Interfaz	

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1. El diseñador instruccional selecciona otra opción.	1.1. El sistema activa los campos pertenecientes a la opción seleccionada.
2.aEl diseñador instruccional introduce incorrectamente los datos solicitados y presiona el botón “Aceptar”. 2.bEl diseñador instruccional no introduce todos los datos solicitados y presiona el botón “Aceptar”.	2.1.aEl sistema muestra un mensaje de error. 2.1.b El sistema muestra un mensaje de error.

Prototipo de Interfaz

Poscondiciones	
Sección "Editar"	
Acción del Actor	**Respuesta del Sistema**
1. El diseñador instruccional selecciona la propiedad que desea modificar y luego la opción "Editar".	1.1. El sistema muestra los datos y los campos de la propiedad previamente creada.
2. El diseñador instruccional modifica los datos necesarios y presiona el botón "Aceptar".	2.1. El sistema almacena los datos y regresa a la sección anterior.
Prototipo de Interfaz	

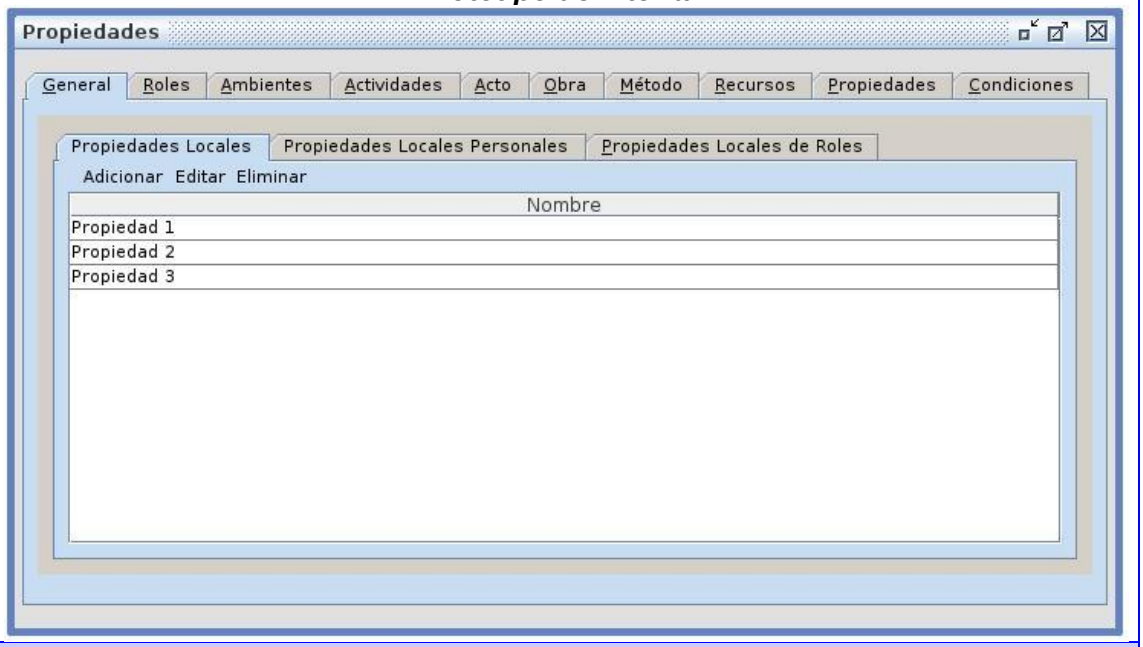


Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
2. El diseñador instruccional modifica incorrectamente los datos y presiona el botón "Aceptar".	2.1.El sistema muestra un mensaje de error.
Prototipo de Interfaz	

Poscondiciones	

Sección "Eliminar"	
Acción del Actor	**Respuesta del Sistema**
1. El diseñador instruccional selecciona del listado la propiedad y luego selecciona la opción "Eliminar".	1.1. El sistema envía un mensaje de confirmación para eliminar los datos.
2. El diseñador instruccional presiona el botón "Aceptar".	2.1 El sistema elimina la propiedad del listado.

Prototipo de Interfaz



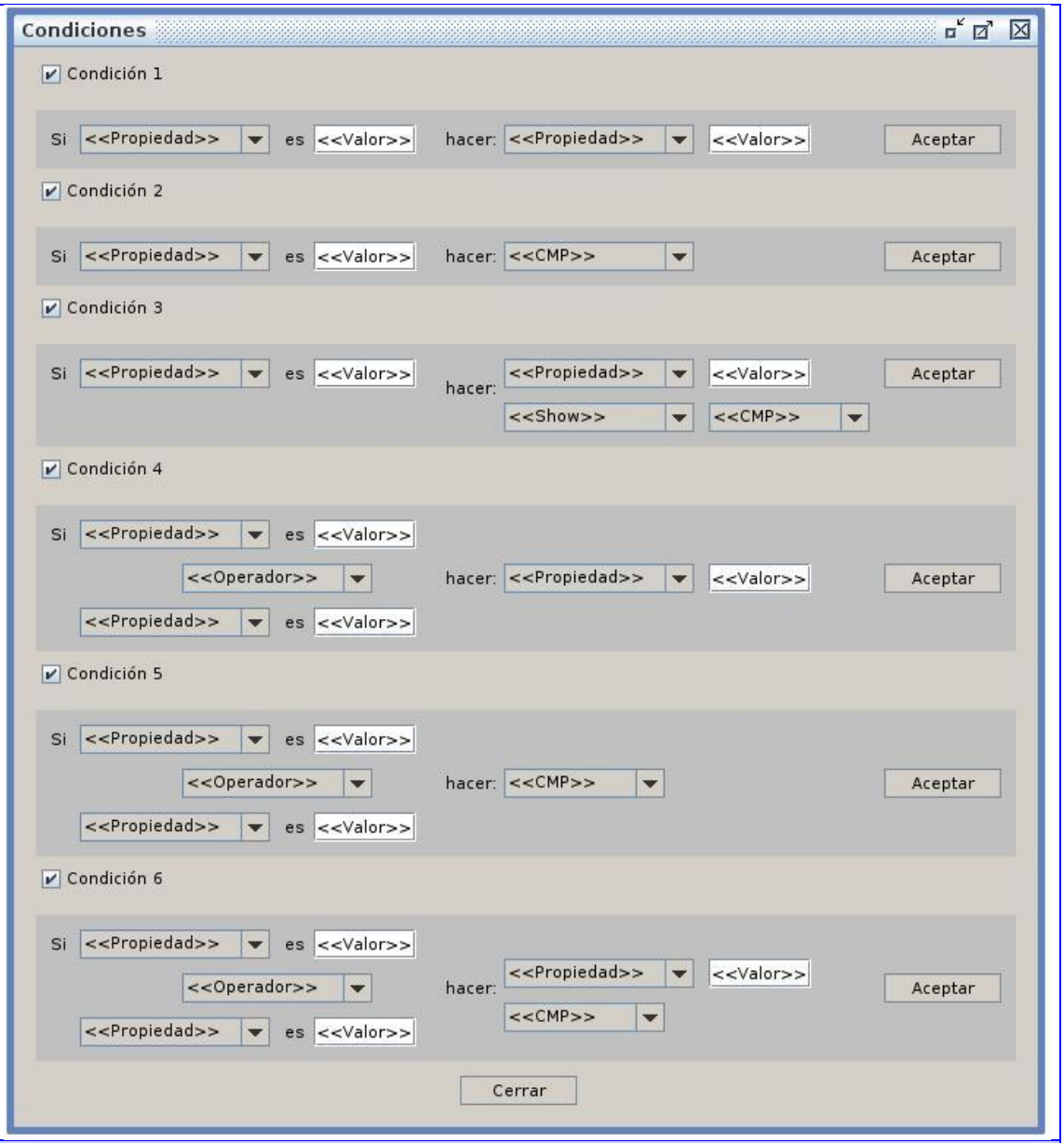
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
2. El diseñador instruccional presiona el botón "Cancelar".	2.1. El sistema regresa a la interfaz anterior.

Prototipo de Interfaz

Poscondiciones	

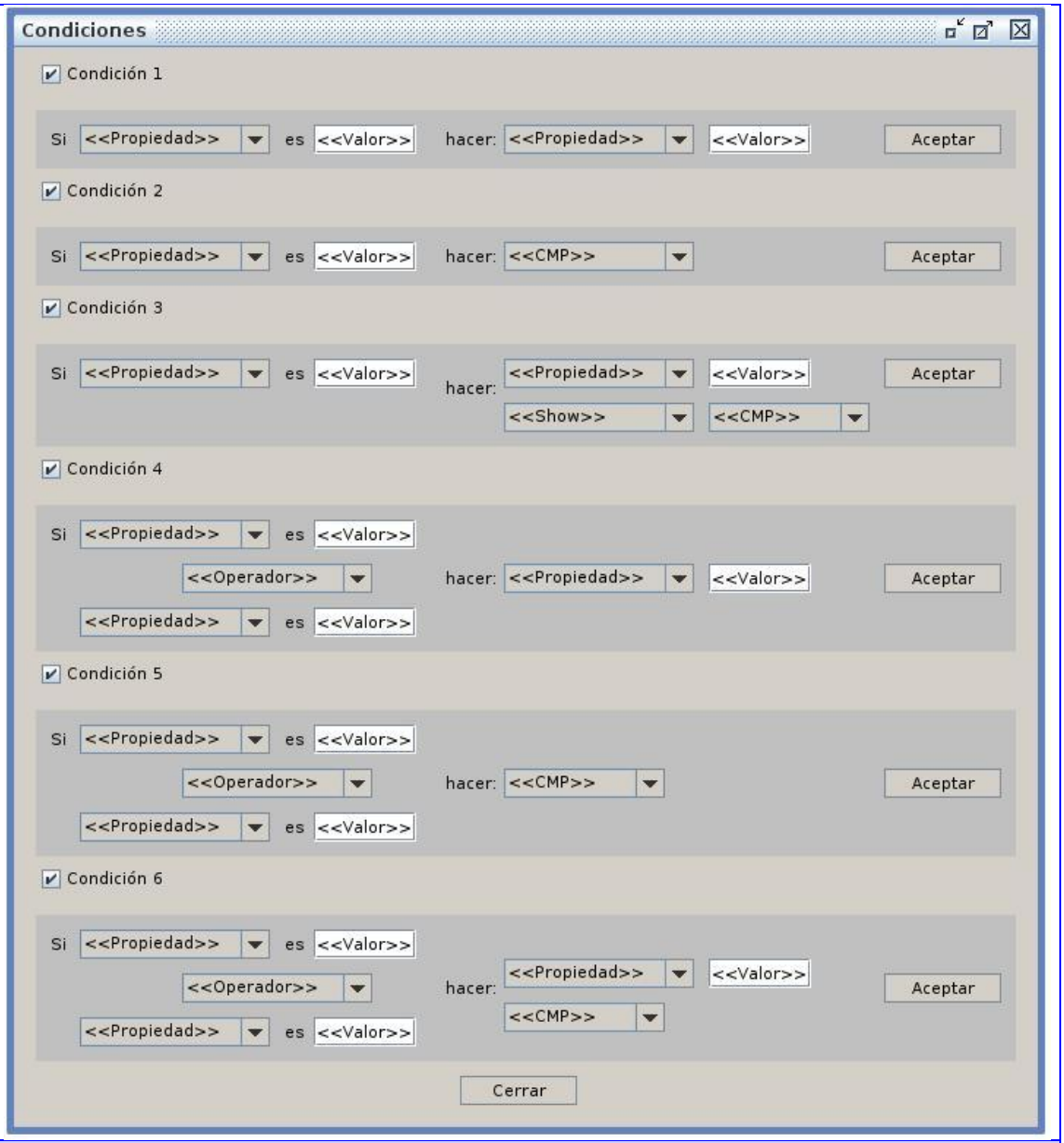
Caso de Uso:	Gestionar condiciones del Diseño de Aprendizaje.
Actores:	Diseñador Instruccional.
Resumen:	El CU inicia cuando el diseñador instruccional decide definir las condiciones de los elementos.
Precondiciones:	El usuario debe estar autenticado en la aplicación.
Referencias	RF7
Prioridad	Crítica.

Flujo Normal de Eventos	
Sección "Adicionar"	
Acción del Actor	**Respuesta del Sistema**
1. El diseñador instruccional selecciona la opción "Adicionar" en la pestaña "Condiciones".	1.1. El sistema muestra una nueva ventana con los campos necesarios para crear las condiciones. Estos varían en dependencia del tipo de propiedad que se escoja, los posibles campos a llenar son: - Propiedad. - Valor. - CMP. - Operador.
2.El diseñador instruccional selecciona el tipo de condición que desea crear. (Existen 6 tipos de condiciones)	2.1 El sistema activa los campos correspondientes a la condición seleccionada.
3. El diseñador instruccional introduce los datos necesarios y presiona el botón "Aceptar".	3.1. El sistema almacena el tipo de condición y regresa a la sección anterior.
Prototipo de Interfaz	



Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1. El diseñador instruccional selecciona otra opción.	1.1. El sistema activa los campos pertenecientes a la opción seleccionada.
3.aEl diseñador instruccional introduce incorrectamente los datos solicitados y presiona el botón “Aceptar”. 3.bEl diseñador instruccional no introduce	3.1.aEl sistema muestra un mensaje de error. 3.1.b El sistema muestra un mensaje de error.

todos los datos solicitados y presiona el botón “Aceptar”.	
Prototipo de Interfaz 	
Poscondiciones	
Sección “Editar”	
Acción del Actor	**Respuesta del Sistema**
1. El diseñador instruccional selecciona la condición que desea modificar y luego la opción “Editar”.	1.1. El sistema muestra los datos y los campos de la condición previamente creada.
2. El diseñador instruccional modifica los datos necesarios y presiona el botón “Aceptar”.	2.1. El sistema almacena los datos y regresa a la sección anterior.
Prototipo de Interfaz	



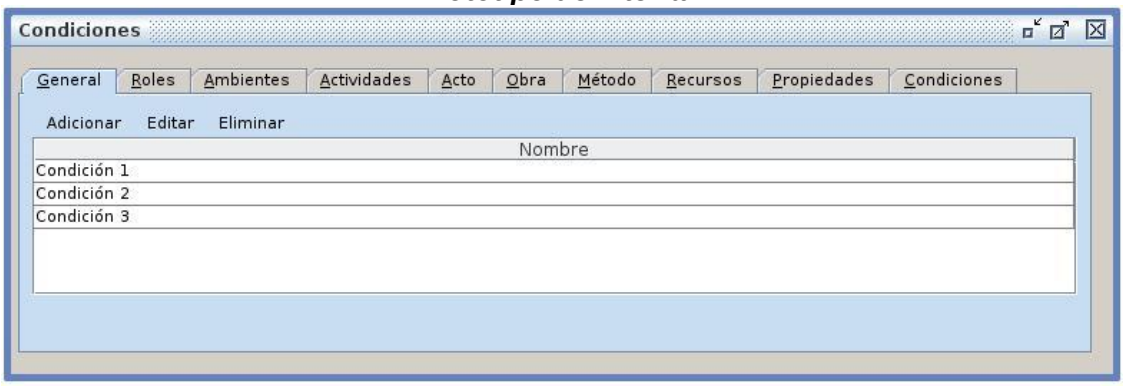
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
2. El diseñador instruccional modifica incorrectamente los datos y presiona el botón “Aceptar”.	2.1.El sistema muestra un mensaje de error.

Prototipo de Interfaz

Poscondiciones	

Sección "Eliminar"	
Acción del Actor	**Respuesta del Sistema**
1. El diseñador instruccional selecciona del listado la condición y luego selecciona la opción "Eliminar".	1.1. El sistema envía un mensaje de confirmación para eliminar los datos.
2. El diseñador instruccional presiona el botón "Aceptar".	2.1 El sistema elimina la condición del listado.

Prototipo de Interfaz



Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
2. El diseñador instruccional presiona el botón "Cancelar".	2.1. El sistema regresa a la interfaz anterior.

Prototipo de Interfaz

Poscondiciones	

Ver en el Anexo 1 las restantes descripciones de los CU del sistema

2.5. Conclusiones parciales

En este capítulo se expresaron las principales clases del dominio para una mayor comprensión del negocio, se identificaron los artefactos actor, cosas de uso y se representó mediante el diagrama de casos de uso las relaciones que se establecen entre estos, se realizó las descripciones detalladas de cada caso de uso del sistema ofreciendo mayor claridad en cuanto a la compresión de cómo debe funcionar el mismo; siendo posible comenzar a realizar el análisis y diseño de la aplicación teniendo en cuenta los requerimientos especificados en el capítulo.

Capítulo III Análisis y diseño del sistema

3.1. Introducción

El presente capítulo se centra en modelar el flujo de trabajo Análisis y Diseño que propone la metodología RUP, donde se representan los artefactos correspondientes. De forma general este capítulo abarca los diagramas de clases del análisis, los diagramas de interacción específicamente el diagrama de colaboración y el diagrama de clases del diseño con estereotipos web; basados en las descripciones textuales propuestas en el capítulo anterior. Además se realiza la validación de algunos de estos artefactos utilizando métricas.

3.2. Modelo del Análisis

Análisis es uno de los flujos de trabajo que se encuentra dentro de la fase de elaboración. Esta disciplina tiene como propósito:

- Conseguir una comprensión más precisa de los requisitos, refinarlos y estructurarlos.
- Utilizar el lenguaje de los desarrolladores para analizar con profundidad los requisitos funcionales.
- Proporcionar una visión general del sistema.

Durante el análisis, es importante realizar un estudio de los requisitos obtenidos, con el objetivo de tener una mejor comprensión antes de entrar al diseño de dicho software, garantizando así una arquitectura robusta, eficaz, eficiente y capaz de sobrevivir a cambios.

3.2.1. Diagrama de clases del análisis

En la construcción del modelo de análisis se deben identificar las clases que describen la realización de los casos de uso, los atributos y las relaciones entre ellas. Con esta información se construye el diagrama de clases del análisis, siendo el principal diagrama de análisis y diseño para un sistema. Este diagrama se representa mediante tres estereotipos:

- Interfaz: Se encargan de la modelación de toda la interacción que puede existir entre los actores y el sistema.

- Control: Representan la coordinación, secuenciación, transacciones y a veces la lógica del negocio; se emplean a menudo para encapsular el control referido a un caso de uso.
- Entidad: Representa la información de larga duración y persistente que se maneja en el sistema.

A continuación se muestran los diagramas de clases de análisis correspondientes a los casos de usos descritos anteriormente.

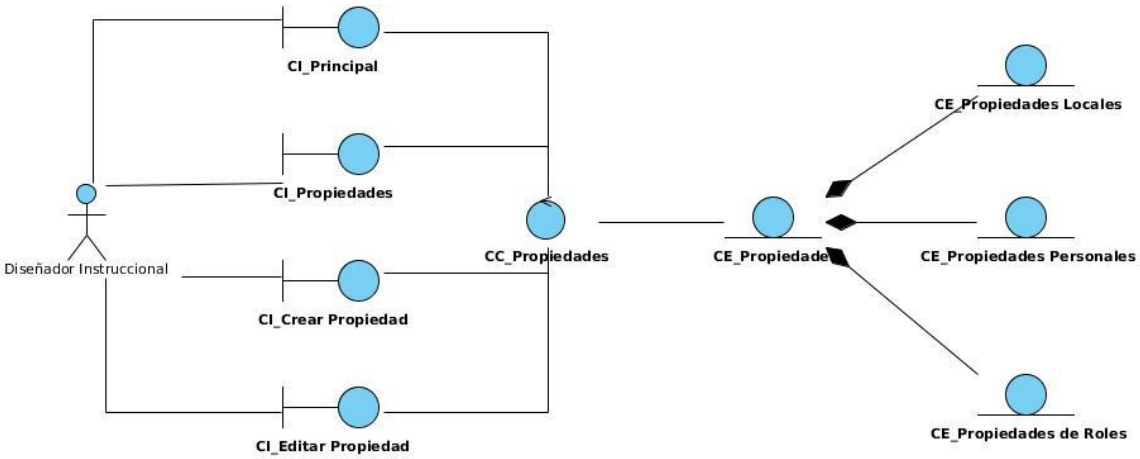


Figura 9 –Diagrama de clases del análisis "Gestionar propiedades del Diseño de Aprendizaje".

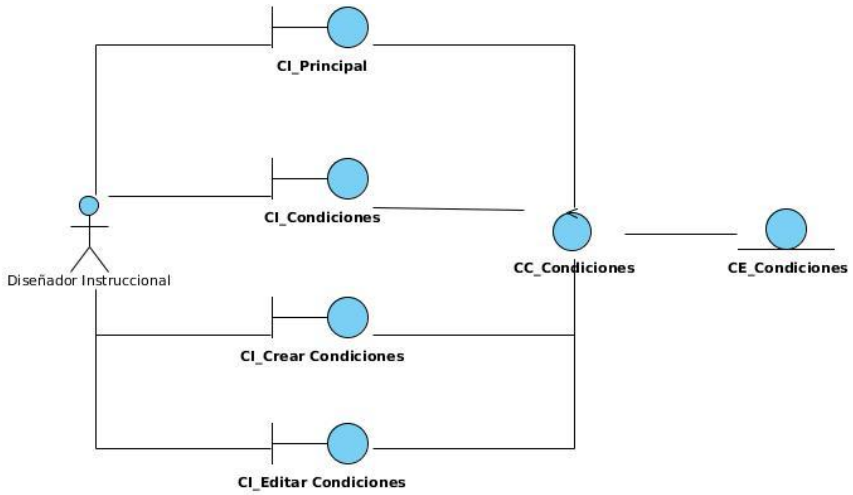


Figura 10 –Diagrama de clases del análisis "Gestionar condiciones del Diseño de Aprendizaje".

Ver en el Anexo 2 las figuras de la 17 a la 24 los diagramas de clases del análisis.

3.3. Diagramas de interacción

Los diagramas de interacción muestran cómo se comunican los objetos. Estos objetos interactúan para realizar colectivamente los servicios ofrecidos por las aplicaciones. Existen dos tipos de diagramas de interacción: los diagramas de secuencia y los diagramas de colaboración que son los que estaremos viendo en este punto.

Los Diagramas de Colaboración se prestan al descubrimiento de abstracciones pues permite representar los objetos en una disposición próxima a la realidad. En esencia, su misión es localizar el comportamiento de los objetos.

En el diseño en ocasiones es preferible representar la interacción entre los objetos o subsistemas con los diagramas de secuencia ya que el centro de atención principal es el encontrar secuencias de iteraciones detalladas y ordenadas en el tiempo.(Jacobson and Booch 2000)

A continuación se muestran los diagramas de colaboración y secuencia correspondientes a los casos de usos descritos.

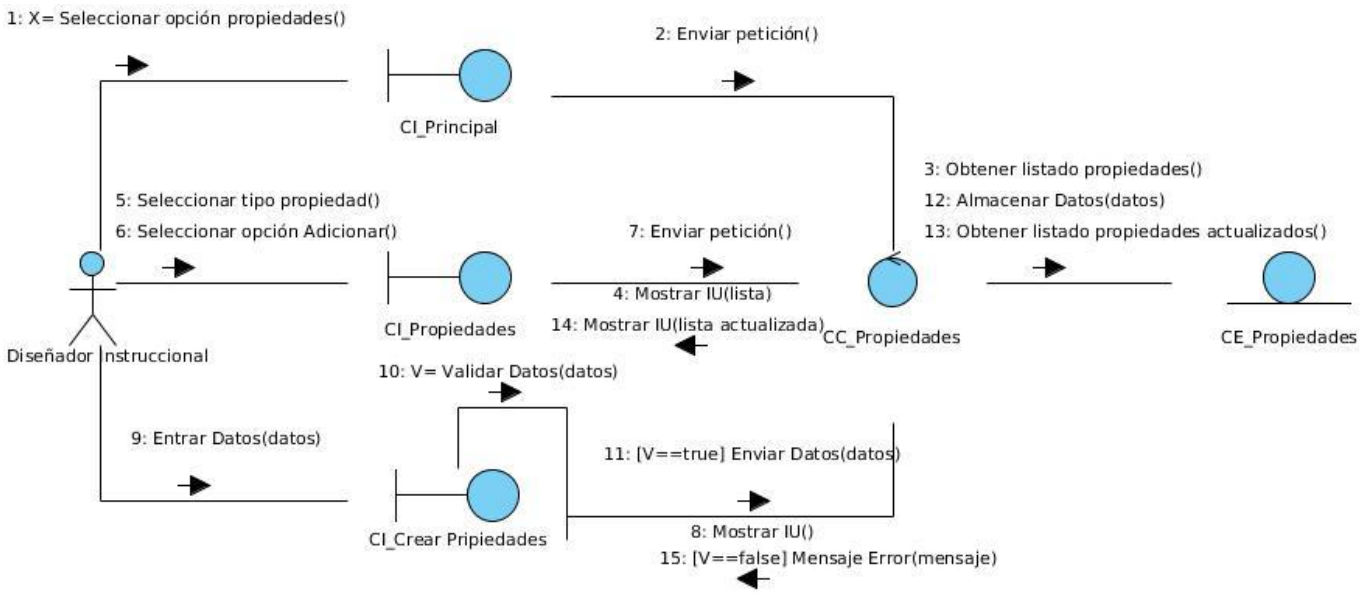


Figura 11 –Diagrama de colaboración "Gestionar propiedades del Diseño de Aprendizaje (Crear)".

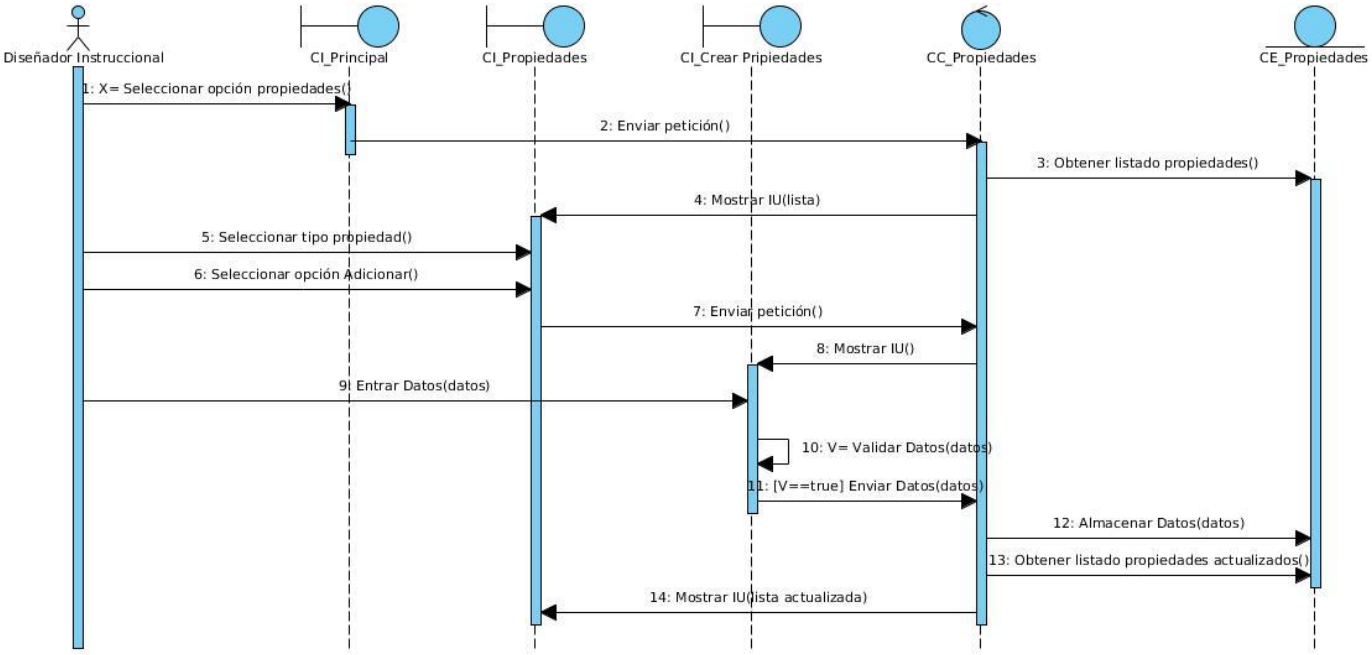


Figura 12 –Diagrama de secuencia "Gestionar propiedades del Diseño de Aprendizaje (Crear)".

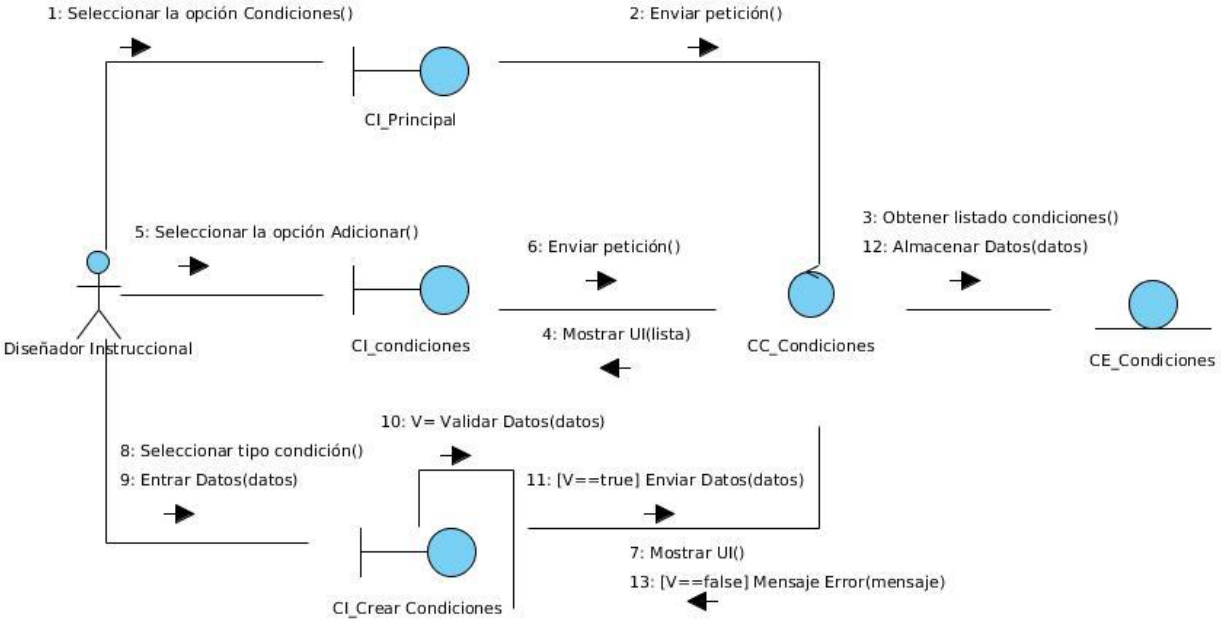


Figura 13 –Diagrama de colaboración "Gestionar condiciones del Diseño de Aprendizaje (Crear)".

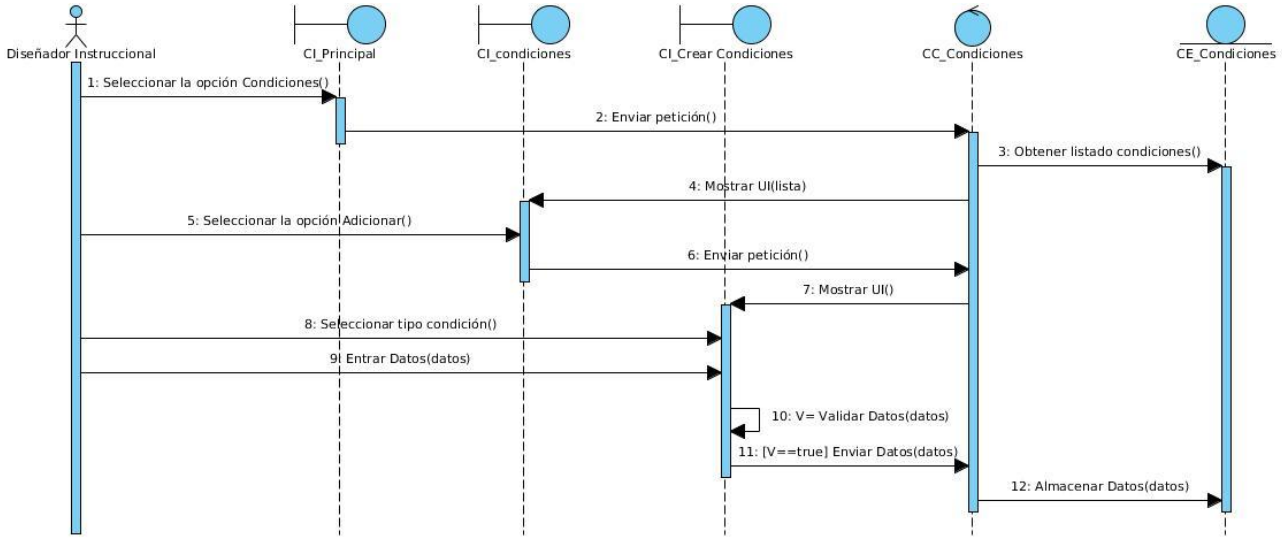


Figura 14 –Diagrama de secuencia "Gestionar condiciones del Diseño de Aprendizaje (Crear)".

Ver en el Anexo 3 las figuras de la 25 a la 40los diagramas de colaboración.

Ver en el Anexo 4 las figuras de la 41 a la 56 los diagramas de secuencia.

3.4. Arquitectura propuesta

"La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución."(IEEE-Std-1471-2000 2000)

La arquitectura de software establece algunas reglas y conceptos claves para que el equipo de desarrollo pueda trabajar en una misma dirección permitiendo alcanzar los objetivos pertinentes.

En este trabajo se propone para la realización del diseño y la implementación el patrón clásico del diseño web conocido como arquitectura Modelo - Vista - Controlador (MVC). El framework Symfony, propuesto para la implementación de este módulo, está basado en este patrón conformado por tres niveles.

El patrón Modelo - Vista - Controlador está catalogado como un patrón de diseño de software donde:

- **Modelo:** Es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.
- **Vista:** Presenta el modelo en un formato adecuado para interactuar con la interfaz de usuario.
- **Controlador:** Responde a eventos, acciones del usuario e invoca cambios en el modelo y en la vista.

3.5. Modelo de diseño

El modelo del diseño constituye el refinamiento del análisis. El diseño crea una representación o modelo del software, proporciona detalles acerca de las estructuras de datos, las arquitecturas, las interfaces y los componentes del software que son necesarios para implementar el sistema. (Pressman 2002)

"Una clase del diseño es una abstracción sin costuras de una clase o construcción similar".(Jacobson and Booch 2000) Las clases del diseño están caracterizadas por (Jacobson and Booch 2000) como:

- El lenguaje utilizado para especificar una clase del diseño es el mismo que el lenguaje de programación. Consecuentemente las operaciones, parámetros, atributos, tipos y demás sin especificados utilizando la sintaxis del lenguaje de programación elegido.
- Pueden realizar y proporcionar interfaces si tiene sentido hacerlo en el lenguaje de programación que se utiliza.

3.5.1. Patrones de diseño utilizados

Son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan.

Es evidente que a lo largo de multitud de diseños de aplicaciones hay problemas que se repiten o que son análogos, es decir, que responden a un cierto patrón. Los patrones de diseño no son fáciles de entender, pero una vez entendido su funcionamiento, los diseños serán mucho más flexibles, modulares y reutilizables.(Larman 2004)

Entre los patrones de diseño utilizados se puede hacer alusión a:

- Patrones de asignación de responsabilidades (Grasp): Alta cohesión, Bajo acoplamiento, Creador, Experto, Controlador.

Varios de estos patrones se evidencian en los diagramas de interacción, ya que son algunos de los artefactos más importantes que se preparan en el análisis y diseño de un software. Es muy importante asignar acertadamente las responsabilidades al momento de elaborar los diagramas de interacción. Mediante estos diagramas se puede ver claramente cómo actúan los patrones bajo acoplamiento y alta cohesión, ya que se refleja la medida de la fuerza con que una clase está conectada a otras clases y cuan enfocadas están las responsabilidades de una clase.

Experto es un patrón que se usa más que cualquier otro al asignar responsabilidades y una aplicación requiere del cumplimiento de cientos de tareas. Este patrón se ve reflejado claramente a la hora de concebir una clase, la cual debe contar con la información necesaria para cumplir las responsabilidades que se le asignen. En la clase propiedades, condiciones y demás clases entidades identificadas se evidencia este patrón.

La creación de nuevas instancias de una clase es una de las actividades más frecuentes en un sistema, a esto hace referencia el patrón creador. El propósito fundamental de este patrones encontrar un creador que se conecte con el objeto producido en cualquier evento. En la clase AccionLD se ve reflejado este patrón.

El patrón controlador define quién debería encargarse de atender un evento del sistema. En el diseño de la propuesta de solución en cuestión las clases encargadas de desempeñar este papel son la clase controladora AccionLD y los formularios.

3.5.2. Diagrama de clases del diseño

Los diagramas de clases muestran el diseño del sistema desde un punto de vista estático, a través de una colección de elementos declarativos, como clases, colaboraciones y sus relaciones.
El módulo en cuestión se diseña para una herramienta web por lo que se hace necesario modelar las páginas, los enlaces entre estas, así como el contenido dinámico de estas, para ello en este trabajo se emplean los estereotipos de UML para diseñar los diagramas de clases del diseño para aplicaciones web. A continuación se muestran los estereotipos con una breve descripción y los diagramas realizados para cada caso de uso identificado.

Tabla de estereotipos web empleados.

Estereotipo	Explicación
<<Client Page>> 	Es la página web de formato HTML, que le brinda al usuario toda la interfaz de la aplicación. Cada página cliente es construida por una sola página de servidor.
<<Server Page>> 	Es la clase encargada del código ejecutable en el servidor. Esta clase es la encargada de construir las páginas clientes mediante una relación *<<build>>*.
<<Form>> 	Es una colección de elementos de entradas que están contenidos en la página cliente. Estas no tienen operaciones y se comunican con las páginas servidoras a través de una relación *<<submit>>*.

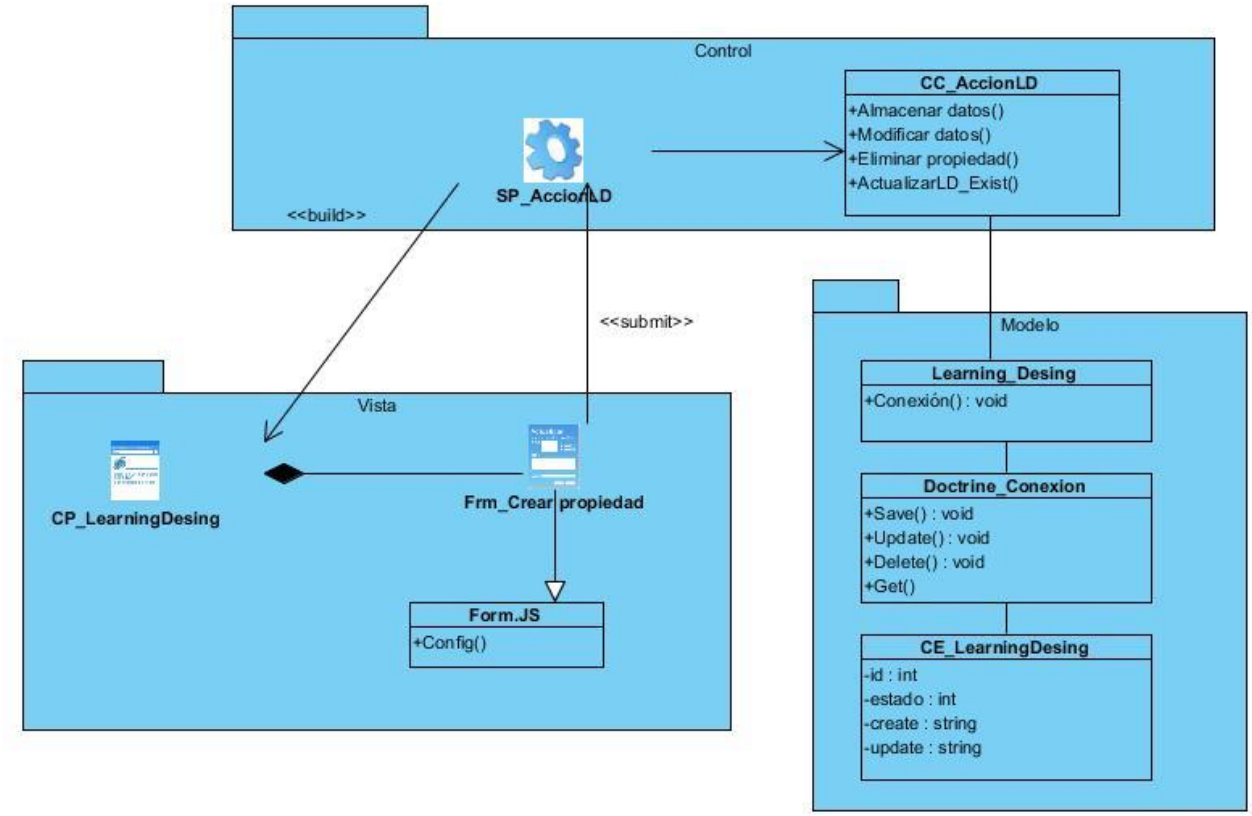


Figura 15 – Diagrama de clases del diseño "Gestionar propiedades del Diseño de Aprendizaje (Crear)".

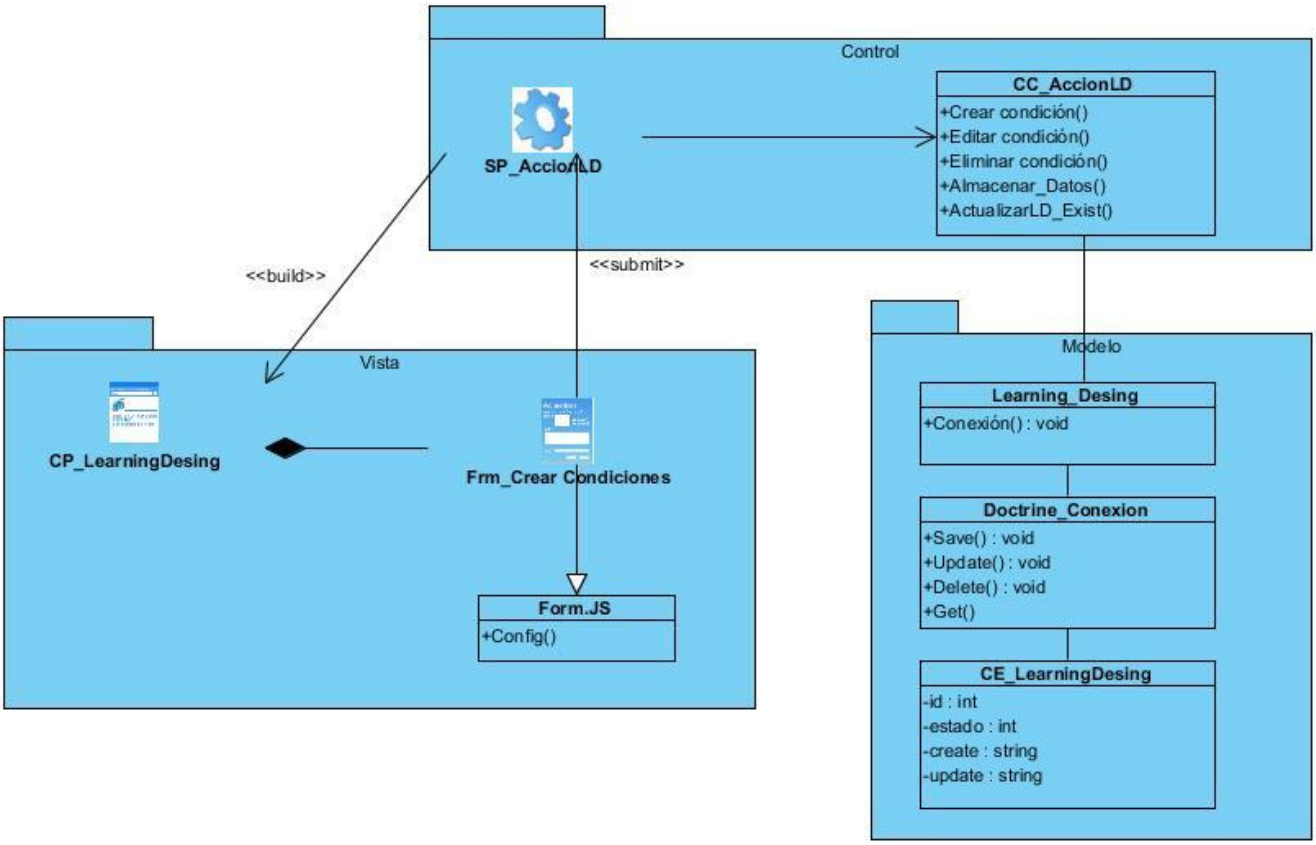


Figura 16 – Diagrama de clases del diseño "Gestionar condiciones del Diseño de Aprendizaje (Crear)".

Ver en el Anexo 5 las figuras de la 57 a la 72 los diagramas de clases del diseño.

3.6. Validación de la solución propuesta

Uno de los procesos más importantes que se llevan a cabo dentro del ciclo de vida de un software es el proceso de Verificación y Validación (V&V). Este es el nombre que se da a los procesos de comprobación y análisis que aseguran que el software que se desarrolla está acorde a su especificación y cumple las necesidades de los clientes. La V&V es un proceso de ciclo de vida completo. Inicia con las revisiones de los requerimientos y continúa con las revisiones del diseño y las inspecciones del código hasta la prueba del producto.

La utilización de determinados métodos como métricas en algunos de los artefactos generados permitirá identificar y erradicar errores en los requisitos y los casos de uso.

3.6.1. Métricas de la Calidad de Especificación de los Requisitos

La métrica de la Calidad de Especificación de los Requisitos mide la especificidad de los requisitos, haciendo que la parte interesada pueda entenderlos de manera fácil y se puedan probar. (Pressman 2005) Es necesario conocer el número total de requisitos para comenzar a utilizar la técnica de validación. Para ver el resultado emitido por los revisores ver Anexo 6.

Rf: número de requisitos funcionales.

Rnf: número de requisitos no funcionales.

Rt: total de requisitos.

Rt = Rf + Rnf

Rt = 8 + 10 = 18

Especificidad

Para determinar la especificidad o falta de ambigüedad de los requisitos, se sugiere una métrica basada en la consistencia de la interpretación de los revisores de cada requisito.(Vargas 2007)

Q1 = Rii/ Rt

Donde Rii es el número de requisitos que todos los revisores interpretaron igual. Cuanto más cercano esté el valor de Q1 (grado de especificación de los requisitos) a 1, menor será la ambigüedad de la especificación.

Primera revisión

Q1 = 15 /18

Q1 = 0.83

Resultado: se detectó que el 16.7% de los requerimientos presentaban ambigüedades, por lo que se realizó una reelaboración de los requerimientos, con términos más entendibles por todos.

Segunda revisión

Q1 = 18/18

Q1 = 1

Resultado: no se detectaron ambigüedades.

Compleción

El resultado de esta métrica está siempre entre 0 y 1. El valor óptimo de esta métrica es el más cercano a 1 e indica un alto nivel de completitud en la definición de los requisitos. Este valor se calcula como se muestra a continuación:

Q2 = Rc/(Rc + Rpe)

Donde Q2 es el grado de completitud de los requisitos, Rc es el número de requisitos completos y Rpe el número de requisitos pobremente especificados.

Primera revisión

Q2 = Rc/(Rc + Rpe)

Q2 = 16 / (16 + 2)

Q2 = 0.89

Resultado: se detectó que el 11.1% de los requisitos definidos no estaban correctamente especificados, se realizó una reelaboración de estos requisitos y para la segunda revisión realizada no se registró ningún inconveniente.

Corrección

El resultado de esta métrica está siempre entre 0 y 1. El valor óptimo de esta métrica es el más cercano a 1 e indica un alto nivel de corrección en la definición de los requisitos. Este valor se calcula como se muestra a continuación:

Q3 = (Rc-Ri)/Rc

Donde Q3 es el grado de validación de los requisitos, Rc número de requisitos validados como correctos y Ri número de requisitos validados como incorrectos.

Primera revisión

Q3 = (Rc-Ri)/Rc

Q3 = (15 - 3)/15= 0.8

Resultado: se concluyó que el 20% de los requisitos no estaban correctamente definidos, se realizó una nueva descripción de estos requisitos arrojando un resultado satisfactorio.

Consistencia interna

El resultado de esta métrica está siempre entre 0 y 1. El valor óptimo de esta métrica es el más cercano a 1 y expresa que no existen subconjuntos de requisitos contradictorios. El valor óptimo de esta métrica es el más cercano a 1. Este valor se calcula como se muestra a continuación:

Q4 = (Re - Rc)/Re

Donde Q4 es el grado de consistencia interna, Re es el número de requisitos especificados y Rc número de requisitos en conflicto con otros requisitos.

Q4 = (Re - Rc)/Re

Q4 = (18 - 0)/18 = 1

Resultado: no se hallaron requisitos en conflictos con otros.

Estabilidad

Para medir la estabilidad de los requisitos de software, en el presente trabajo se aplicó la métrica propia para esto, la cual ofrece valores entre 0 y 1. El mejor valor de es el más cercano a 1. La estabilidad de los requisitos se calcula de la siguiente forma:

E = (Rt - Rm)/Rt

Donde E es el valor de la estabilidad de los requisitos y Rm es el número de requisitos modificados.

Clasificación de la estabilidad

Clasificación	Intervalo
Alta	0.90 <= E <= 1
Media	0.80 <= E < 0.90
Baja	E < 0.80

E = (Rt - Rm)/Rt

E = (18 - 0)/18 = 1

Resultado: se detectó que los requisitos no han sido modificados en cuanto al indicador estabilidad, por lo que la estabilidad se clasifica como alta.

3.6.2. Métricas de casos de uso

Para validar los casos de uso se realizó la técnica de las métricas, en la que se emplearán 4 atributos: (B. Bernárdez 2004)

Completitud

Un caso de uso es completo si especifica todo lo que deben hacer el actor y el sistema para alcanzar el objetivo del caso de uso y si se consideran todas las respuestas del sistema a situaciones anormales. Para verificarlo se plantean las siguientes interrogantes:

- ¿Hay respuestas a todas las peticiones que el actor del caso de uso hace al sistema y viceversa?
- ¿Se contemplan todos los posibles escenarios para poder alcanzar el objetivo del caso de uso?
- ¿Se especifican todas las secuencias alternativas a la secuencia normal?
- ¿Se contemplan todas las posibles excepciones a la secuencia normal?

Comprensibilidad

Un caso de uso es comprensible si todos los tipos de lectores pueden entenderlo fácilmente con una mínima explicación del autor. Para verificarlo se plantean las siguientes interrogantes:

- ¿Es posible leer el caso de uso sin volver atrás en repetidas ocasiones?

- ¿Es difícil seguir la secuencia normal del caso de uso por la presencia de las relaciones *include* o *extend*?
- ¿Es difícil seguir la secuencia de pasos por la existencia de demasiados pasos alternativos?
- ¿Se han desglosado demasiado los pasos de algún actor o del sistema provocando que el caso de uso avance a un ritmo muy lento?
- ¿Aparecen pasos condicionales para expresar que el sistema comprueba una situación que permite al caso de uso continuar su realización?

Concisión

Un caso de uso es conciso si no incluye información innecesaria. Para verificarlo se plantean las siguientes interrogantes:

- ¿Podría el caso de uso ser expresado con menos palabras?
- ¿Existen elementos que se puede obviar o aparecen anotaciones innecesarias y que dificultan la lectura del caso de uso?
- ¿Aparecen demasiadas interacciones entre el actor principal del caso de uso y otros elementos del entorno?

No trivialidad

Un caso de uso es no trivial si su secuencia de pasos conduce al actor a conseguir el objetivo que persigue la realización del caso de uso. Para verificarlo se plantean las siguientes interrogantes:

- ¿Expresa el nombre del caso de uso un objetivo de un usuario que el sistema debe implementar?
- ¿Conduce el caso de uso al actor a conseguir alguno de sus objetivos sin representar un conjunto de interacciones triviales?

Al aplicar la técnica Métricas de Casos de Uso, se hizo necesario realizar dos revisiones ya que en la primera de estas se detectaron inconvenientes tales como:

Primera revisión:

- En el análisis de la completitud se encontró que el caso de uso Gestionar condiciones del diseño de aprendizaje y Completar Acto no estaban acabados ya que no se especificaban todas las

acciones del actor y del sistema para alcanzar el objetivo del caso de uso; lo que representa un error del 17% y un nivel de cumplimiento del 83% para el indicador.

- Análisis de la comprensibilidad: se encontró que el caso de uso Definir cambio de valor de la propiedad contenía información que dificultaba el entendimiento, lo que arrojó un error del 8%, lo que implica que el 92% de los casos de uso cumplen debidamente con el indicador.

Segunda revisión:

En la segunda iteración, luego de haber corregido los errores encontrados en la primera revisión, no se detectaron casos de uso que arrojaran algún índice de error sobre alguno de los indicadores por los que los casos de uso estaban 100% correcto en cuanto completitud, comprensibilidad, concisión y no trivialidad.

3.7. Conclusiones parciales

Como resultado del estudio realizado en este capítulo, el cual corresponde al flujo de trabajo de análisis y diseño del sistema se identificaron todas las clases, se desarrollaron los diagramas de clases del análisis de los casos de usos del sistema, se confeccionó los diagramas de colaboración y los diagramas de secuencia para tener una mejor visión de la interacción de los objetos. Además se elaboraron los diagramas de clases del diseño definiendo los objetivos que debe cumplir el sistema. Se realizó la validación a los artefactos especificación de los requerimientos y casos de uso.

Conclusiones Generales

Con la realización de esta investigación se analizaron aspectos relacionados con la especificación IMS-LD, destacando la importancia de la misma para el modelamiento del proceso de enseñanza-aprendizaje, llegando así a las siguientes conclusiones:

La incorporación de los elementos correspondientes al nivel B de esta especificación brindan una mayor interacción, consiguiendo la personalización y secuenciación de las Unidades de Aprendizaje, adaptando el flujo de aprendizaje a las necesidades de cada participante, permitiéndole al diseñador instruccional controlar el flujo de las actividades del diseño dentro de una Unidad de Aprendizaje.

Con los artefactos generados en los flujos de trabajo Requerimientos, Análisis y Diseño de la metodología RUP, se crearon las bases necesarias para la implementación de las funcionalidades para el desarrollo del nivel B de la especificación IMS-LD en el módulo de DI de CRODA 2.0.

Recomendaciones

Se recomienda para la futura incorporación de esta investigación en la herramienta:

- Realizar la implementación de las funcionalidades para el desarrollo del nivel B de la especificación IMS-LD en el módulo de DI de CRODA 2.0.
- Validar la propuesta realizada para la gestión de propiedades y condiciones en CRODA 2.0.
- Realizar la integración del nivel C de la especificación IMS – LD en el módulo de DI en CRODA 2.0, de forma tal que posibilite el envío de mensajes y el establecimiento de nuevas actividades basadas en hechos ciertos.

Bibliografía

Berlanga, A. J. (2005) IMS Learning Design: Hacia la Descripción Estandarizada de los Procesos de Enseñanza.

Bray, T. (2008) Extensible Markup Language (XML).

Burgos, D. (2005) IMS Learning Design desde dentro. Una especificación para crear escenarios de aprendizaje online (I Entrega).

Burgos, D. (2005) IMS Learning Design desde dentro. Una especificación para crear escenarios de aprendizaje online (II Entrega).

Consumoteca (2009). “Servidor Web. Consumoteca.” from http://www.consumoteca.com/diccionario/servidor-web.

Cuaresma, S. B. (2008). “Metodologías de desarrollo.”. from http://www.marblestation.com/?p=644.

eXe, e. (2010). “eXe eXeLearning.” from http://exelearning.org/wiki.

Falcon, M. M. (2011) Implementación del nivel A de IMS-LD para la herramienta de autor web CRODA

Gil, M. C. (2009). “Ateneo Empresarial.” from http://ateneo-empresarial.com/archives/category/diseno-instruccional.

IMS (2003). “IMS Learning Design Information Model.” from http://www.imsglobal.org/Learning Desing/ldv1p0/imsld_infov1p0.html#1495294.

ITE (2010). “Instituto de Tecnologías Educativas. HERRAMIENTAS DE SOPORTE A IMS-LD.” from http://ares.cnice.mec.es/informes/20/contenidos/10.htm.

MIT (2010). “MIT. Servidor Apache HTTP.” from http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-httpd.html.

Orellana, J. M. Suárez, et al. (2001) EL DISEÑO INSTRUCCIONAL, UNA DIMENSIÓN CLAVE INSUFICIENTEMENTE ATENDIDA EN LA TELEFORMACIÓN.

Quiñones, E. (2008) Introducción a postgreSql.

TTnet, R. and A. M. Hernández (2005). “La Formación Sin Distancia.”

Vázquez, J. M. M. (2007) Estado del arte del eLearning. Ideas para la definición de una plataforma universal.

Yukavetsky, G. J. (2008) La elaboración de un módulo instruccional.

B. Bernárdez, A. D., M. Toro (2004). "Revista de Procesos y Métricas de las Tecnologías de la Información.".

Berlanga, A. J. (2005) IMS Learning Design: Hacia la Descripción Estandarizada de los Procesos de Enseñanza.

Bray, T. (2008) Extensible Markup Language (XML).

Burgos, D. (2005) IMS Learning Design desde dentro. Una especificación para crear escenarios de aprendizaje online (I Entrega).

Burgos, D. (2005) IMS Learning Design desde dentro. Una especificación para crear escenarios de aprendizaje online (II Entrega).

Consumoteca (2009). "Servidor Web. Consumoteca." from http://www.consumoteca.com/diccionario/servidor-web.

Cuaresma, S. B. (2008). "Metodologías de desarrollo.". from http://www.marblestation.com/?p=644.

Domínguez-Dorado, M. (2005). Todo Programación. Nº 13.

Falcon, M. M. (2011) Implementación del nivel A de IMS-LD para la herramienta de autor web CRODA.

Fowler, M. a. S., Kendal (1996). UML Gota a Gota. s.l. : Prentice Hall, 1996.

Gil, M. C. (2009). "Ateneo Empresarial." from http://ateneo-empresarial.com/archives/category/diseno-instruccional.

IEEE-Std-1471-2000 (2000). Recommended Practice for. Architectural Description of Software-Intensive Systems.

IMS (2003). "IMS Learning Design Information Model." from http://www.imsglobal.org/Learning Desing/ldv1p0/imsld_infov1p0.html#1495294.

ITE (2010). "Instituto de Tecnologías Educativas. HERRAMIENTAS DE SOPORTE A IMS-LD." from http://ares.cnice.mec.es/informes/20/contenidos/10.htm.

Jacobson, I. and G. Booch (2000). El proceso unificado de desarrollo de software.

Koper, R. and B. Olivier (2004). "Que representa el diseño de aprendizaje de las unidades de aprendizaje. Tecnología Educativa y Sociedad." from http://jime.open.ac.uk/jime/article/viewArticle/2005-8/275.

Koper, R. and C. Tattersall (2005) Diseño de aprendizaje: un manual sobre el modelado y la entrega de la educación y la formación en red.

Larman, C. (2004). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. 3ra Edición.

MIT (2010). "MIT. Servidor Apache HTTP." from http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-httpd.html.

Navathe, S. B. (1997) Fundamentos de Sistemas de Bases de Datos.

Orellana, J. M. Suárez, et al. (2001) EL DISEÑO INSTRUCCIONAL, UNA DIMENSIÓN CLAVE INSUFICIENTEMENTE ATENDIDA EN LA TELEFORMACIÓN.

Pressman, R. (2005). Ingeniería del Software. “Un enfoque práctico“.

Pressman, R. S. (2002) Ingenieria de Software Un Enfoque Práctico.

Quiñones, E. (2008) Introducción a postgreSql. .

Singh, H. and C. Reed (2002) Estándares y Especificaciones para los Entornos e-learning: Convergencia en Contenidos y Sistemas.

Slocum, J., B. Moeskau, et al. (2007). "Sencha_Ext_JS."

Sommerville, I. and P. Sawyer (1997). "Requirements Engineering A good practice guide." from http://i.f.alexander.users.btopenworld.com/reviews/sommerville_and_sawyer.htm.

TTnet, R. and A. M. Hernández (2005). "La Formación Sin Distancia."

Vargas, M. A. A. (2007) Métricas del software.

Vázquez, J. M. M. (2007) Estado del arte del eLearning. Ideas para la definición de una plataforma universal.

Yukavetsky, G. J. (2008) La elaboración de un módulo instruccional.

Zaninotto, F. P. F. (2007). The Definitive Guide to symfony.

Glosario de Términos

- **e-learning:** Enseñanza a distancia caracterizada por una separación física entre profesores y alumnos, donde se usa preferiblemente Internet como medio de comunicación y de distribución del conocimiento.

- **SCORM**: Modelo de referencia para el desarrollo e integración de contenidos educativos.

- **TIC:** Se denominan Tecnologías de la Información y las Comunicaciones (TIC), al conjunto de Tecnologías que Permiten la adquisición, producción, almacenamiento, tratamiento, comunicación, registro y presentación de informaciones.

- **UML:** Lenguaje Unificado de Modelado es el lenguaje de modelado de sistemas de software más conocido en la actualidad.

- **Reutilización:** capacidad de los OA para integrarse con otros formando una sola entidad. En términos de reutilización de metadatos se refiere a la capacidad de reutilizar los metadatos de un recurso para describir otro.

- **Accesibilidad:** capacidad que hace identificable y ubicable cuando se necesite al contenido de un recurso, para los requerimientos formativos necesarios, que se conozca su adecuación a los objetivos sin necesidad de obtener el propio contenido o pagar derechos por él, mediante la provisión de información suficiente.

- **Open Source:** código abierto. Manera de nombrar también a las aplicaciones desarrolladas bajo el amparo de un producto con licencia GPL.

- **Interoperabilidad:** capacidad de las plataformas para comunicarse e integrarse de una forma efectiva aun siendo diferentes.