UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 5: Entornos Virtuales, Hardware y Automática

DISEÑO DE UNA HERRAMIENTA DE AUTOR DE OBJETOS DE APRENDIZAJE

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autoras: Katia García Martínez.
Suneidis Vinent Torres.

Tutora: Ing. Irina Elena Argota.

Asesora: Ing. Belkis Grissel González.

Consultante: Lic. Juan Antonio Fung Goizueta.

Ciudad de la Habana, julio 5, 2008.
"Año 50 del triunfo de la Revolución"

"Lo que caracteriza a una inteligencia formada es que puede descansar satisfecha con el grado de precisión que la naturaleza de un asunto permite, y no buscar la exactitud cuando sólo una aproximación de la verdad es posible…"
Aristóteles

DECLARACIÓN DE AUTORÍA

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

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

Katia García Martínez	**Suneidis Vinent Torres**	**Irina E. Argota Vera**
_____	_____	_____
Firma del Autor	Firma del Autor	Firma del Tutor

Ing. Irina Elena Argota Vera.

Graduada de Ingeniera Informática en el 2007 y profesor en adiestramiento. Con 2 años de experiencia como analista de software dentro del proyecto SCADA perteneciente al Polo de Hardware y Automática de la Facultad 5. Correo electrónico: iargota@uci.cu.

.

A mis padres Sandra y Suitberto por todo su amor, comprensión y apoyo desde que nací, a los que adoro junto a mi hermanito Sidney que lo quiero mucho.
Darle las gracias a mi familia que siempre ha estado pendiente de mí:
A mis abuelas y abuelos: Marina, Dulce, Suitberto y Roberto;
A mis tías: Mariela y Niuris y mi tío Manolito;
A mis primas y primos: Dhayana, Marina, Andris y Alaen.
A mi novio Pedro Alberto por todo su apoyo en los momentos difíciles.
A mis amigos que me han ayudado en las buenas y en las malas: a Judith, a Margarita, a Husseyn, a Maria, a Oreidis y un besito para Maykol.
A mi compañera de tesis por toda su paciencia con mi optimismo.
A mi familia en Playa.
Agradecer a los profesores que nos guiaron para que hoy fuéramos profesionales para nuestra sociedad y en especial a los profesores Zenaida y Pedro Raúl.
Agradecer a mí tutora Irina por su preocupación por mis cosas y por la tesis.
Las gracias también a nuestro comandante en jefe, Fidel Castro Ruz por darnos la oportunidad de estudiar en esta universidad.
A todos muchísimas gracias.

Suneidis.

A mi mamá por tantos sacrificios hechos para que yo pudiera estar estudiando, y por ser la mejor madre del mundo.
A Vicente por toda su preocupación por mi estancia en la UCI, y por todo el apoyo brindado.
A mi hermana Kirenia por estar ahí cuando más la necesito.
A Edivaldi por su comprensión y ánimos.
A mi tía Paula Esther por todo su apoyo y preocupación porque yo llegara a realizar este sueño.
A mi abuela Caridad por todas sus atenciones en todo momento…. Te quiero mucho.
A mi tutora Irina por estar siempre dispuesta a aclarar cualquier duda.
Al profe Pedrito por toda la ayuda brindada.
A mi compañera de tesis por soportar todo mi desespero durante la elaboración de la misma.
A mis amigas Yanaris y Yahima por acompañarme siempre en los buenos y malos momentos.
A mis compañeras de apto por ser la familia más cercana con que compartí durante este curso.
A toda mi familia por ser ejemplo para mí.
A todos los que han aportado ideas para la realización de este trabajo de diploma.
A nuestro Comandante en Jefe por haberme dado la posibilidad de estudiar en una Universidad de Excelencia.
A todos muchas gracias.

Katia.

A mi mamá por tanto cariño y apoyo; por ser lo más grande para mí en este mundo.
A mi abuelita querida que se que estaría muy orgullosa de mí en estos momentos. Que EN PAZ DESCANCES.

Katia.

A mis padres sin los cuales no estaría aquí, por todo el amor que me han dado en mi formación.

Suneidis

"Las herramientas de autor son aplicaciones que tienen la intención de reducir el esfuerzo necesario para producir software, cargando con la responsabilidad en los aspectos mecánicos o la tarea, guiando al autor, y ofreciéndole elementos predefinidos que puede relacionar conjuntamente para satisfacer una necesidad particular (Educativa)".

Unido a este concepto aparece la definición de Objetos de Aprendizaje (OA) dada por cualquier recurso con una intención formativa, compuesto de uno o varios elementos digitales, descritos con metadatos, que pueda ser utilizado y reutilizado.

Producto a las necesidades de los especialistas cubanos y del mundo de tener una herramienta que apoye el proceso docente y que permita crear objetos interactivos en los cursos que imparten es que se tuvo la idea de diseñar en software libre (liberándose del software propietario) bajo la licencia GNU/Linux: la **Herramienta de Autor de Objetos de Aprendizaje.**

Dicha herramienta formará parte de un sistema cuyo nombre es Emedia que tiene como fin apoyar el proceso de enseñanza-aprendizaje.

PALABRAS CLAVE

Herramienta de Autor, Objetos de Aprendizaje, metadatos.

INTRODUCCIÓN

Actualmente el desarrollo de la educación favorece directamente al progreso social y económico de una región o un país. Para que esta premisa tenga efecto, es básico y previo un aumento de las capacidades personales de cada individuo. El objetivo fundamental de la educación es proporcionar a los estudiantes, una formación plena que les ayude a estructurar su identidad y a desarrollar sus conocimientos en aras de participar en la construcción de la sociedad. (1)

Internet exige cambios en el ámbito educativo, y los profesionales de la educación tienen múltiples razones para aprovechar las nuevas posibilidades que este medio ofrece para impulsar este cambio hacia un nuevo paradigma educativo más personalizado y centrado en la actividad de los estudiantes. Se hace necesaria la alfabetización digital de los alumnos y el aprovechamiento de las Tecnologías de la Informática y las Comunicaciones (TIC) para alcanzar una mayor productividad, ya que estas pueden suministrar medios para la mejora de los procesos de enseñanza - aprendizaje y para la gestión de los entornos educativos en general. Un ejemplo de estos entornos son los llamados Entornos Virtuales de Aprendizaje (EVA). (2)

Un EVA, también conocido como Sistema de Gestión del Aprendizaje (SGA), es un software instalado en un servidor que se utiliza para crear, gestionar y distribuir cursos destinados a la preparación individual de cada estudiante. Sirve además de contenedor de cursos y de Objetos de Aprendizaje (OA), e incorpora otras herramientas para facilitar la comunicación y el trabajo colaborativo entre profesores y estudiantes permitiendo el seguimiento y evaluación del alumno. (3)

Un OA es una entidad, digital o no digital, que puede ser utilizada, reutilizada y referenciada durante el aprendizaje apoyado con tecnología. (4)

La Universidad de Ciencias Informáticas (UCI), como ente productor y exportador de software, establece como una de sus principales líneas de producción, el desarrollo de aplicaciones multimedia educativas, tanto para el propio desarrollo de formación y aprendizaje de sus estudiantes, como para las diferentes enseñanzas e instituciones, ya sean nacionales o extranjeras.

La comunidad de desarrollo de aplicaciones educativas de la UCI, durante el proceso de elaboración de software educativo ha detectado un conjunto de irregularidades que atentan contra la calidad de las aplicaciones resultantes. Las principales son:

- La utilización de herramientas de propósito general no orientadas al conjunto de especificidades que se requieren para lograr una mayor calidad en el desarrollo de aplicaciones.

- La existencia de herramientas relativamente intuitivas y fáciles de utilizar viene condicionada por las pocas prestaciones que ofrecen. Otras de mayores prestaciones están básicamente orientadas a desarrolladores de software y no a especialistas de la educación. Este fenómeno, en la mayoría de los casos, impide su empleo por parte de los expertos en contenido con su consecuente impacto negativo en las soluciones finales.
- Las Herramientas asociadas a la gestión del software educativo se encuentran dispersas y por lo tanto, no integran sus procesos fundamentales como la planificación, el diseño, desarrollo y despliegue.
- La creciente presencia de la Red Global Mundial , en inglés, World Wide Web(Web) a la hora de distribuir los contenidos educativos olvidándose de que estos pueden distribuirse además mediante Aplicaciones Desktop, CD-ROM entre otras. Este problema afecta de manera directa a los países pobres ya que se ven limitados debido a la velocidad de la red y en algunos casos, como en nuestro país, la forma de acceso a la misma.(5)

Surge así la necesidad de crear un SGA que les facilite a los profesores la forma en que interactúan con los nuevos medios informáticos y contribuya a prescindir de las dificultades existentes en la actualidad. Dándole además la posibilidad de que desarrollen e implementen sus propias ideas, que se sientan cómodos utilizando los nuevos recursos y medios de enseñanza. Los sistemas desarrollados deben permitir una mayor interacción con los usuarios finales.

Para suplir la necesidad planteada anteriormente es que se crea el proyecto Emedia con el objetivo de desarrollar y gestionar herramientas para el aprendizaje.
Emedia se elabora sobre plataforma libre y está compuesto por módulos independientes entre los que se destacan:

- **SGA:** Es la herramienta encargada del despliegue del sistema Emedia.
- **Autor de Contenido:** Posibilita la creación de los guiones de contenidos necesarios para la construcción de cursos.
- **Proyector:** Se ocupa de la visualización de los OA.
- **Intérprete:** Es un software que recibe un programa en lenguaje de alto nivel, lo analiza y lo ejecuta. Para analizar el programa completo, va traduciendo sentencias de código y ejecutándolas si están bien, así hasta completar el programa origen.
- **Repositorio:** Posibilita el almacenamiento, acceso y gestión de los OA desarrollados.

Además de los módulos antes mencionados se hace necesario buscar una vía para crear OA, con el objetivo de poner en práctica las ideas de los Especialistas en Contenido o Expertos. Por consiguiente, se hace imprescindible trazar una línea de desarrollo de una herramienta que permita la realización de dichos recursos, centrado en el diseño de la misma como un primer avance de la aplicación, y que pueda contribuir a un posterior seguimiento.

Para dar solución a la situación problemática existente se tiene que responder la presente interrogante del **Problema Científico:** ¿Cómo diseñar un subsistema para el producto Emedia destinado a la creación de Objetos de Aprendizaje?

El **Objeto de estudio** son las herramientas de Autor.

El **Campo de acción** lo constituye la herramienta Autor de Objetos de Aprendizaje para el sistema Emedia.

El **Objetivo General** que se desea alcanzar es, diseñar una herramienta Autor de Objetos de Aprendizaje.

Para este trabajo se definieron las siguientes **Tareas de la Investigación** en aras de lograr un mayor nivel de detalle en la misma:

- Revisar bibliografías referentes a términos que se tratarán en la investigación.
- Analizar las herramientas que existen en el mundo para apoyar la producción de software educativo.
- Seleccionar la Metodología de Análisis y Diseño del Sistema.
- Definir las tecnologías a usar en la creación de la herramienta.
- Realizar el Modelo de Dominio.
- Identificar los requerimientos del sistema.
- Identificar los casos de usos del sistema.
- Realizar el diagrama de caso de uso del sistema.
- Describir detalladamente los casos de uso arquitectónicamente significativos.
- Diseñar un prototipo no funcional de interfaz de usuario.
- Realizar diagramas de clases del análisis y de diseño de los casos de uso más significativos.
- Realizar diagramas de secuencia del diseño.

De acuerdo con el problema científico planteado la **Idea a Defender** definida es la siguiente:

El diseño de una herramienta de Autor de Objetos de Aprendizaje para el sistema Emedia, destinada a la creación de OA y que permita tener una base para su futura implementación, le facilitará al

Especialista en Contenidos (Experto) la conversión de su experiencia y conocimientos en información fácil de usar, a un costo y esfuerzo menor, posibilitando una mayor calidad en la producción de cursos educativos.

Para el desarrollo de las tareas científicas se combinan diferentes **Métodos y Técnicas en la búsqueda y procesamiento de la información**, los fundamentales son:

A nivel teórico

Métodos de análisis-síntesis e inducción-deducción: Para el estudio de las concepciones y conceptos empleados en el campo; para analizar las teorías y documentos generados por desarrolladores de otras Herramientas de Autor como Atenex, Autore y Dokeos. En la revisión y justificación de la metodología de desarrollo de software y tecnologías a utilizar.

Análisis histórico-lógico: Para conocer, con mayor profundidad, los antecedentes y las tendencias actuales referidas a las herramientas de Autor existentes, además de conceptos, términos y vocabularios propios del campo como herramienta de Autor, Objetos de Aprendizaje, entre otros, que contribuyen en gran medida al entendimiento del trabajo.

Modelación: Para la caracterización de las funcionalidades que tendrá la aplicación cuando se estaban definiendo los requisitos funcionales.

A nivel Empírico

Entrevista: En la búsqueda de fundamentación teórica de la necesidad de creación del sistema Emedia.

Experimentos: En la elaboración de prototipos no funcionales, con el objetivo de validar los Casos de Uso identificados.

El presente documento está estructurado en tres capítulos:

En el **Capítulo 1** Fundamentación teórica. En el se estudian los conceptos y sistemas relacionados con herramientas de Autor, específicamente enfocados en la creación de Objetos de Aprendizaje; así como las tecnologías más utilizadas en plataforma libre y que serán utilizadas en el diseño de la aplicación.

En el **Capítulo 2** Características del sistema. Se aborda a cerca del estado actual del negocio en cuestión, además de las características del subsistema que se pretende diseñar. Se modela el negocio y a partir de las funcionalidades identificadas, se realiza la propuesta de solución.

En el **Capítulo 3** Análisis y diseño. Teniendo en cuenta la metodología RUP, se elaboran un grupo de artefactos que pertenecen a la disciplina y se proponen patrones para el diseño de la aplicación.

CAPÍTULO 1: Fundamentación teórica.

Introducción.

En el presente capítulo se hace referencia a los principales conceptos y características fundamentales a tener en cuenta en el trabajo como las relacionadas con las herramientas de Autor, Objetos de Aprendizaje, entre otros. Se definen aspectos fundamentales de ingeniería de software, y se hace un estudio de las tendencias y tecnologías actuales a tener en cuenta en la investigación.

1.1 Surgimiento de las herramientas de Autor.

El proceso de producción de un curso en formato digital es una tarea difícil que requiere mucho esfuerzo por parte de especialistas de contenido, desarrolladores del currículo y programadores. Es por ello que no estuvo al alcance de la mayoría de los maestros e instructores por mucho tiempo. Esta complejidad se pudo reducir apoyándose en herramientas informáticas con ese propósito, que automatizaron todo el proceso de elaboración de un curso. Así surgen las herramientas de Autor, que inicialmente resultó desalentador para muchos docentes ya que estaban más bien orientadas a usuarios con amplios conocimientos en informática y no al Especialista en Contenidos.

El aumento de la demanda de formación a través de las TIC ha propiciado que se desarrollen herramientas cada vez más fáciles de utilizar por el usuario y los profesores. De esta manera ha ido creciendo en estos últimos años una nueva clase de software que tiene como objetivo facilitar la creación, publicación y gestión de materiales educativos digitales a utilizar en la teleformación. A este se le ha llamado, software educativo.

Las primeras ideas sobre el desarrollo de software educativo aparecen en la década de los 60 con programas hechos con herramientas de programación, lentamente, se desarrollan varias líneas distintas.

Cataldi las divide en:

1. Lenguajes para el aprendizaje (como el Logo).
2. Lenguajes y herramientas que sirven para la generación del producto de software educativo.
3. Las herramientas de Autor.

"*Los intentos para utilizar herramientas digitales en la creación de materiales educativos han sido muchos y han respondido a varios ejes de clasificación. Uno principal ha sido considerar la creación de materiales como una actividad estandarizable o, por el contrario, como una actividad creativa y completamente personal. Entre ambos extremos se han situado muchas opciones. En la actualidad*

una concepción importante es la referida a los OA, así como a la estandarización de materiales educativos y de su descripción (p.e. el modelo de referencia SCORM)." (6)

1.2 Definición de herramienta de Autor.

Las aplicaciones que se utilizan para la creación de materiales educativos y publicación de cursos son las que se denominan herramientas de Autor, y las mismas se pueden dividir en tres tipos; las que generan simulaciones, las que generan los materiales a incluir en el curso y su publicación y las que permiten la creación de materiales educativos digitales. Sobre este último tipo de herramienta es que va encaminada la investigación en este trabajo.

Existen varias definiciones de herramientas de Autor dadas por diferentes especialistas en el tema. Algunas de estas son: *"Las herramientas de autor son herramientas de desarrollo de software que posibilitan a diseñadores instructivos, educadores, maestros y aprendices diseñar un curso multimedia interactivo, y ambientes de aprendizaje en hipermedia sin el conocimiento de lenguajes de programación. Las herramientas de autor tienen como objetivo aplicaciones educativas que contengan generalmente, ya sea implícitamente o explícitamente, un modelo particular de la tarea en el que el usuario final debe estar ocupado, así como un modelo del proceso editorial mismo." Un aspecto muy importante que se tiene en cuenta en esta definición es la relación de dependencia de la herramienta de Autor con el modelo pedagógico del curso a crear y el modelo del proceso de producción en la herramienta de Autor.*

"Según De Leeuwe son herramientas para crear con facilidad contenidos de aprendizaje. Páginas Web con hipervínculos y toda clase de interactividad (y hasta exámenes) son creadas con la misma facilidad de una presentación en Power Point. Con estas herramientas se pueden importar objetos de aprendizaje existentes como texto, imágenes, sonidos y video usando técnicas de arrastrar con el ratón. El contenido puede ser portado en HTML, CD-ROM o al código estándar SCORM y AICC8."

Para fines de este trabajo se tomará la definición que diera Tom Murray en la que, un aspecto importante es la responsabilidad que debe tener este producto de software de guiar el proceso de elaboración de materiales educativos para satisfacer necesidades educativas particulares.

"Las herramientas de autor son aplicaciones que tienen la intención de reducir el esfuerzo necesario para producir software, cargando con la responsabilidad en los aspectos mecánicos o la tarea, guiando al autor, y ofreciéndole elementos predefinidos que puede relacionar conjuntamente para satisfacer una necesidad particular (Educativa)" (…).

La selección de una herramienta de autor dependerá fundamentalmente de dos factores: Las características particulares de la aplicación a desarrollar y la formación y experiencias del propio desarrollador. Además, es deseable que el modelo del sistema facilite la implantación de la aplicación resultante y por otro lado es importante tener en cuenta las posibilidades del autor en materia de programación y la portabilidad de la aplicación hacia los sistemas operativos de los usuarios. (6)

1.3 Herramientas de Autor existentes en el mundo.

Para el desarrollo de este trabajo se evaluaron diferentes herramientas de Autor con el objetivo de conformar una idea general de estas aplicaciones. A continuación se presentan algunas de ellas:

Atenex

Atenex es una plataforma de origen española, destinada a la creación y gestión de materiales multimedia interactivos y para el seguimiento y evaluación del proceso de aprendizaje de los alumnos. Entre sus principales características se encuentran, la facilidad de edición de materiales y la compatibilidad que presenta con los estándares SCORM e IMS. También es importante destacar de ella, que es una herramienta multiplataforma preparada para trabajar tanto en local como en línea. Su facilidad de uso está relacionada a la forma en que se elaboran los materiales por el simple hecho de arrastrar y soltar además de incorporar un completo plantillero de actividades programadas de fácil configuración. Dispone de foros de ayuda y módulos de Preguntas Frecuentes como ayuda en línea. (7)

Autore

Esta herramienta multiplataforma permite generar contenidos e-learning con una interfaz extraordinariamente sencilla e intuitiva. Su uso no requiere ningún tipo de conocimientos en programación. Los contenidos que genera son materiales multimedia estructurados de acuerdo al modelo de un OA (Objetivo o idea, Desarrollo y Evaluación). Autore cuenta con un potente editor de páginas y es capaz de importar textos, imágenes, animaciones, documentos de audio y video, ecuaciones MML. Además, es posible elaborar ejercicios de autoevaluación. Los metadatos se pueden introducir durante el proceso de autoría. Autore ha sido diseñada e implementada por el Campus Virtual de la UPV-EHU y es uno de los elementos que conforman su Learning Content Management System (LCMS). (8)

Dokeos

Dokeos es un entorno de e-learning y una aplicación de administración de contenidos de cursos y también una herramienta de colaboración. Es software libre y está bajo la licencia GNU General Public License o simplemente su acrónimo del inglés GNU GPL, el desarrollo es internacional y colaborativo. También está certificado por la Iniciativa de Código Abierto, en inglés, Open Source Initiative (OSI) y puede ser usado como un sistema de gestión de contenido (CMS) para educación y educadores. Esta característica para administrar contenidos incluye distribución de contenidos, calendario, proceso de entrenamiento, chat en texto, audio y video, administración de pruebas y guardado de registros. Hasta el 2007, estaba traducido en 34 idiomas (y varios están completos) y es usado por más de mil organizaciones. (9)

1.4 Concepción general de un Objeto de Aprendizaje.

Los OA en su concepción fueron pensados para apoyar el proceso de enseñanza-aprendizaje en los diferentes niveles de educación. A continuación se exponen una serie de conceptos relacionados con estos recursos como son su definición, propiedades, reutilización y metadatos.

1.4.1 Definición de Objeto de Aprendizaje.

Los OA son elementos para la instrucción, aprendizaje o enseñanza basada en computadora. Están basados en una filosofía, que según Wiley (2000) se fundamenta en la corriente de las ciencias de la computación conocida como orientación a objetos.

La orientación a objetos se basa en la creación de entidades con la intención de que puedan ser reutilizadas en múltiples aplicaciones. Este método avizora mejoras de amplio alcance en la forma de diseño, desarrollo y mantenimiento del software, ofreciendo una solución a largo plazo a los problemas y preocupaciones que han existido desde el comienzo en el desarrollo de software: la falta de portabilidad del código y reutilización, código que es difícil de modificar, ciclos de desarrollo largos y técnicas de codificación no intuitivas. Esta misma idea se sigue para la elaboración de los OA. Es decir, los diseñadores instruccionales pueden desarrollar componentes instruccionales pequeños que pueden ser reutilizados en diferentes aplicaciones educativas (Wiley, 2000).

Formalmente no hay un único concepto de OA y las definiciones son muy amplias. El Comité de Estandarización de Tecnología Educativa, dice que los OA son *"una entidad, digital o no digital, que puede ser utilizada, reutilizada y referenciada durante el aprendizaje apoyado con tecnología" (4).*

Según Wiley (2000) son *"cualquier recurso digital que puede ser reutilizado para apoyar el aprendizaje".*

Estas definiciones son muy amplias y a la hora de ponerlas en práctica pueden resultar inoperables ya que no hay un elemento claro que distinga a los OA de otros recursos.

Morales & García (2005) definen a los OA como una unidad de aprendizaje independiente y autónomo que está predispuesto a su reutilización en diversos contextos instruccionales. Y por otra parte, JORUM+ Project (2004) dice que *"un OA es cualquier recurso que puede ser utilizado para facilitar la enseñanza y el aprendizaje y que ha sido descrito utilizando metadatos"*.

Estas ideas dan una definición más actual, ya que estas características son componentes intrínsecos para que un OA pueda ser identificado.

Se dan como ejemplos de OA los contenidos multimedia, el contenido instruccional, los objetivos de aprendizaje, software instruccional, personas, organizaciones o eventos referenciados durante el aprendizaje basado en tecnología (IEEE, 2001). Por otro lado, JORUM+ Project (2004) dice que se puede incluir una imagen, un mapa, una pieza de texto, una pieza de audio, una evaluación o más de uno de estos recursos, cabe resaltar que se mencionan extractos o sólo parte de los recursos y es posible no considerar el recurso completo, como asimismo hace hincapié en que un OA también puede ser el conjunto de dos o más recursos.

Teniendo en cuenta la variedad de las definiciones, así como la diversidad de recursos que pueden considerarse como OA, se puede concluir para términos de esta investigación que se considerará como OA a cualquier recurso con una intención formativa, compuesto de uno o varios elementos digitales, descritos con metadatos, que pueda ser utilizado y reutilizado (Figura 1). (10)

Figura 1. Conceptualización de un OA.

1.4.2 Propiedades de los OA.

David Wiley padre de la teoría de los OA, plantea cuatro características básicas que estos recursos deben cumplir; estas son: reusabilidad, accesibilidad, interoperabilidad y durabilidad. Además de estas, (Rehak & Mason, 2003) han agregado una que se refiere a la capacidad que deben cumplir estos objetos para albergarse en cualquier plataforma. Estas propiedades en general se definen de la siguiente forma:

- *Reutilizables:* El recurso debe ser modular para servir como base o componente de otro recurso. También debe tener una tecnología, una estructura y los componentes necesarios para ser incluido en diversas aplicaciones.
- *Accesibles:* Pueden ser indexados para una localización y recuperación más eficiente, utilizando esquemas estándares de metadatos.
- *Interoperables:* Pueden operar entre diferentes plataformas de hardware y software.
- *Portables:* Pueden moverse y albergarse en diferentes plataformas de manera transparente, sin cambio alguno en estructura o contenido.
- *Durables:* Deben permanecer intactos a las actualizaciones (upgrades) de software y hardware.

La creación de OA no es sencilla, pero los esfuerzos y costos de producción se equilibran con las veces que este recurso pueda reutilizarse. (10)

1.4.3 Reutilización de los OA.

La característica que más se distingue en las distintas definiciones que se mencionaron anteriormente de los OA es la reutilización. Esta se trata de utilizar elementos de software previamente desarrollados para generar un nuevo producto de software. Cuando reutilizamos un contenido, este aumenta su valor y produce ahorro, en diferentes sentidos, a nivel institucional o individual. El gran potencial de reutilizar los OA es poder aprovechar los contenidos que han desarrollado otros para formar nuevos recursos.

En la Figura 2 se ejemplifica cómo a partir de tres OA independientes se genera otro nuevo, no se tuvo que desarrollar un nuevo contenido, únicamente se reutilizó el que ya existía. Otro tipo de reutilización es la de utilizar un mismo OA pero entre diversas aplicaciones. El contenido no cambia, sólo se incluye en otro programa académico que le da un contexto diferente.

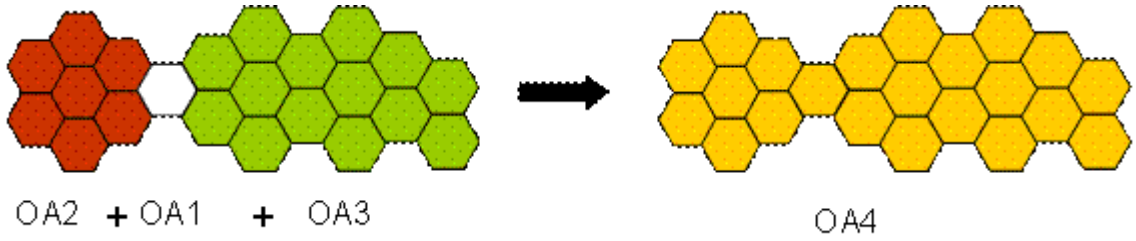


Figura 2. Creación de un nuevo OA a partir de la composición de otros.

Para lograr la reutilización, así como para hacer cumplir las propiedades descritas anteriormente, se hace necesario que el OA cuente con los metadatos que le permitan ser identificado, organizado y recuperado, entre otros aspectos como la categorización y calificación pedagógica del objeto, pero lo más importante es que esos metadatos estén basados en un estándar, a fin de asegurar su compatibilidad e interoperabilidad con los sistema que puedan reutilizarlos (López, García & Pernías, 2005). (10)

1.4.3.1 Metadatos de los OA.

"*Los metadatos son un conjunto de atributos o elementos necesarios para describir un recurso. A través de los metadatos se tiene un primer acercamiento con el objeto, conociendo rápidamente sus principales características. Son especialmente útiles en los recursos que no son textuales y en los que su contenido no puede ser indizado por sistemas automáticos, por ejemplo, los multimedios o un audio.*" (10)

Un ejemplo donde se evidencia de forma más simple los metadatos para el ámbito educativo, se encuentra en una ficha bibliográfica, en la que se tiene toda la información que describe al recurso y se puede decidir si se consulta o no sin haber tenido contacto directo con el libro (u otro recurso documental), esto hace más fácil y ágil ubicar el recurso que se desea consultar dentro de una colección. Algunos de los descriptores que contiene la ficha son: ubicación, título, autor, editorial, año de edición, tema y número de páginas de un libro, estos descriptores tienen su origen en la catalogación bibliotecaria y se conocen ahora también como metadatos.

Los metadatos no sólo son descriptivos, también pueden ser administrativos y de estructura (Caplan 2003): (10)

- *Metadatos descriptivos:* tienen propósito de descubrimiento (cómo se encuentra un recurso), identificación (cómo un recurso puede distinguirse de otro), y selección (cómo determinar que un recurso cubre una necesidad particular). Los metadatos descriptivos sirven también para

formar colecciones de recursos similares. Otras de sus funciones son la evaluación, relación (con otros recursos) y usabilidad.

- *Metadatos administrativos*: es información que facilita la administración de los recursos. Incluyen información sobre cuándo y cómo fue creado el recurso, quién es el responsable del acceso o de la actualización del contenido y también se incluye información técnica, como la versión de software o el hardware necesario para ejecutar dicho recurso.
- *Metadatos estructurales*: sirven para identificar cada una de las partes que componen al recurso, definen la estructura que le da forma. Por ejemplo, un libro, que contiene capítulos y páginas, se puede etiquetar con metadatos que identifican cada parte y la relación que guardan entre ellas. Se usan especialmente para el procesamiento de la máquina y por software de presentación o estilos.

1.5 Metodologías de desarrollo de software.

Las metodologías para el desarrollo de software describen el conjunto de fases, etapas, actividades y tareas asociadas a la producción de software (aplicación) de calidad. La finalidad del uso de metodologías es lograr el desarrollo de un software de calidad.
Rumbaugh dio la siguiente definición:

"Una metodología de ingeniería de software es un proceso para la producción organizada del software, empleando para ello una colección de técnicas predefinidas y convencionales en las notaciones. Una metodología se presenta normalmente como una serie de pasos, con técnicas y notaciones asociadas a cada paso. Los pasos de la producción del software se organizan normalmente en un ciclo de vida consistente en varias fases de desarrollo". (11)

Dentro de las clasificaciones existentes para las mismas sobresalen dos: las metodologías tradicionales y las ágiles.

Las metodologías tradicionales están basadas en normas provenientes de estándares seguidos por el entorno de desarrollo, son generalmente aplicadas a grandes proyectos e indican paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Además detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla.

Por otra parte se encuentran las llamadas metodologías ágiles que se centran más en la obtención del sistema sin importar cuán documentado esté el mismo, generalmente propone que los involucrados participen en el proceso de desarrollo para ir mejorando las funcionalidades del sistema en producción, así como el trabajo en parejas. (12)

Dentro de las metodologías tradicionales y que puede ser visto como una metodología ágil además, pues es tan configurable como deseemos, se encuentra el Rational Unified Process (RUP) haciendo uso del lenguaje UML.

1.5.1 Metodología Orientada a Objetos: RUP.

La metodología RUP, fue desarrollada en 1998 por Grady Booch, Ivar Jacobson y James Rumbaugh, prominentes metodologistas en la industria de la tecnología y sistemas de información. RUP se caracteriza por ser: dirigido por Casos de Uso, centrado en la arquitectura, e iterativo e incremental.

Una metodología de desarrollo de software Orientada a Objeto (OO) consta de los siguientes elementos:

- Conceptos y diagramas (Modelo).
- Etapas y definición de entrega en cada una de ellas.
- Actividades y recomendaciones. (ISW 2006)

Las principales disciplinas de esta metodología son:

Disciplinas	Descripción
Modelado del negocio	Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización. Propone comprender los problemas de la organización e identificar posibles mejoras, evaluar el impacto del cambio introducido con la automatización, asegurar que todos los miembros tienen una misma visión de la organización.
Requerimientos	Esta disciplina se encarga de convertir las peticiones de los clientes en un conjunto de requerimientos de software, que definan el alcance y lo que debe hacer el producto a ser construido. Propone mantener un acuerdo a los interesados de lo que el sistema debe hacer, provee a los desarrolladores de una mejor visión de los requerimientos del sistema, define la frontera del sistema, crea las bases para la planificación y estimación.

Análisis y Diseño	Esta disciplina define como transformar artefactos resultantes del flujo de requerimientos de software en especificaciones del diseño del proyecto a desarrollar, en el se hace evolucionar la arquitectura de modo que se adapte al entorno del usuario final.
Implementación	Define como desarrollar, organizar realizar pruebas de unidad e integración de todos los componentes implementados basado en las especificaciones del sistema.
Pruebas	Este flujo se centra en encontrar y documentar los defectos en la calidad del software, validar y probar todos los supuestos hechos durante el diseño, verificar que el producto se desempeña como fue diseñado y cubre todos los requisitos pactados durante el flujo de "Captura de Requerimientos".
Instalación	Se encarga de garantizar que el producto esta disponible para lo usuarios en su ambiente, se realizan actividades como empaque, instalación, asistencia a usuarios.
Administración del proyecto.	Se centra los aspectos esenciales del desarrollo iterativo como son la planificación, control de riesgos, seguimiento del proceso de desarrollo de software y aplicación de métricas. Las restantes serán utilizadas durante el proceso de administración.
Administración de configuración y cambios	Se encarga de controlar y sincronizar la evolución de todos los productos generados, introduce el uso de herramientas que faciliten el trabajo colaborativo en el equipo.
Ambiente	Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

Tabla1. Principales etapas de RUP.

El RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al final de cada ciclo. (Ver Figura 3).

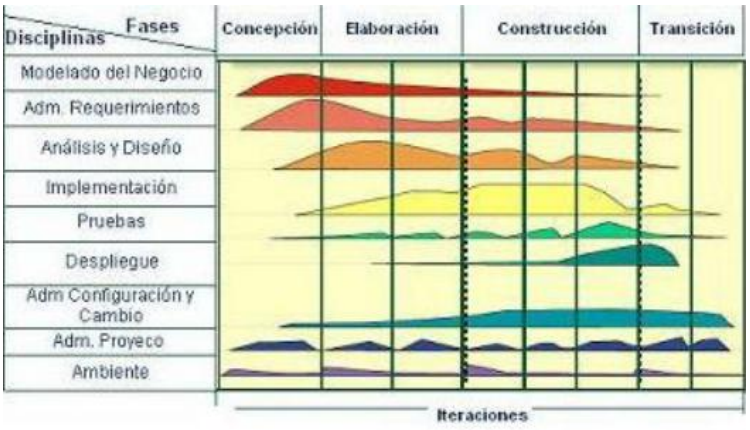


Figura 3. Metodología de desarrollo de software: RUP.

Cada ciclo se divide en fases que finalizan con un hito donde se debe tomar una decisión importante:

• **Conceptualización (Concepción o Inicio):** Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.

• **Elaboración:** Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales) identificados de acuerdo al alcance definido.

• **Construcción:** Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene uno o varios entregables del producto que han pasado las pruebas. Se ponen estos entregables a consideración de un subconjunto de usuarios.

• **Transición:** El entregable ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores.

Se decidió usar RUP como metodología de desarrollo de software; en primer lugar porque el proyecto tiene un amplio alcance y necesita tener el más mínimo detalle de especificación que con la documentación que genera RUP es fácil de lograr, y en segundo lugar se logra organización y calidad del producto, posibilitando que en cada iteración se tenga una visión más clara de lo que se está desarrollando, disminuyendo riesgos de errores en el producto final y tratando de que el cliente quede conforme con lo que se hizo.

1.6 Lenguaje de Modelado UML.

Entre los lenguajes de modelado que define Object Management Group (OMG) el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad es UML (Unified Modelling Language). Es un lenguaje gráfico para especificar, construir y documentar los artefactos que modelan un sistema. Fue diseñado para ser un lenguaje de modelado de propósito general, por lo que se puede utilizar para especificar la mayoría de los sistemas basados en objetos o en componentes, y para modelar aplicaciones de muy diversos dominios de aplicación y plataformas de objetos.

Al establecer una comparación entre UML 1.5 y UML 2.0 se llegó a la conclusión de que este último ofrece a los proveedores de herramientas CASE (Computer-Aided Software Engineering) "la producción automática de programas basado en especificaciones del software".

El UML 1.5 está constituido por 7 diagramas básicos y dos diagramas (Ver Figura 4) que constituyen variaciones de dos de los anteriores:

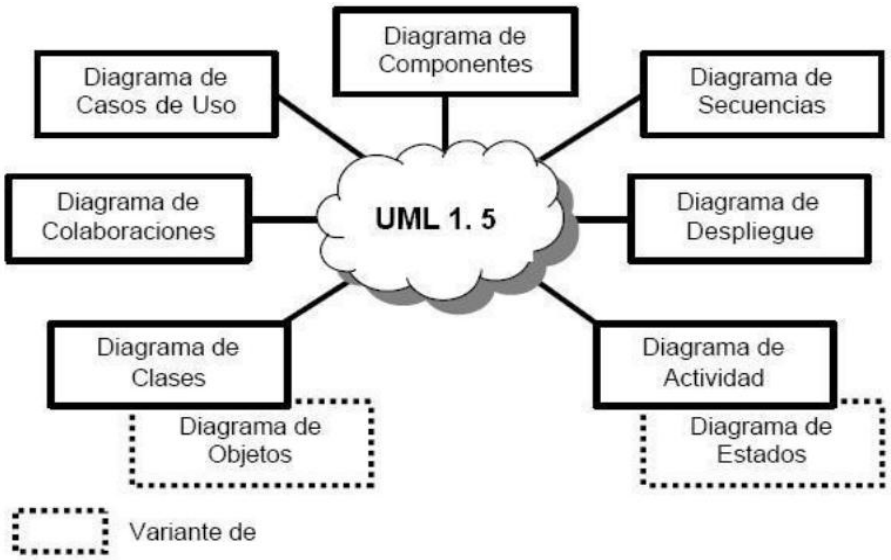


Figura 4. Diagramas de UML 1.5

En UML 2.0 hay 13 tipos diferentes de diagramas, ya que, se definen una serie de diagramas adicionales a los establecidos en UML 1.5. El conjunto de diagramas se encuentra organizado en torno a dos categorías: diagramas estructurales y diagramas dinámicos o de comportamiento. (Ver Figura 5)

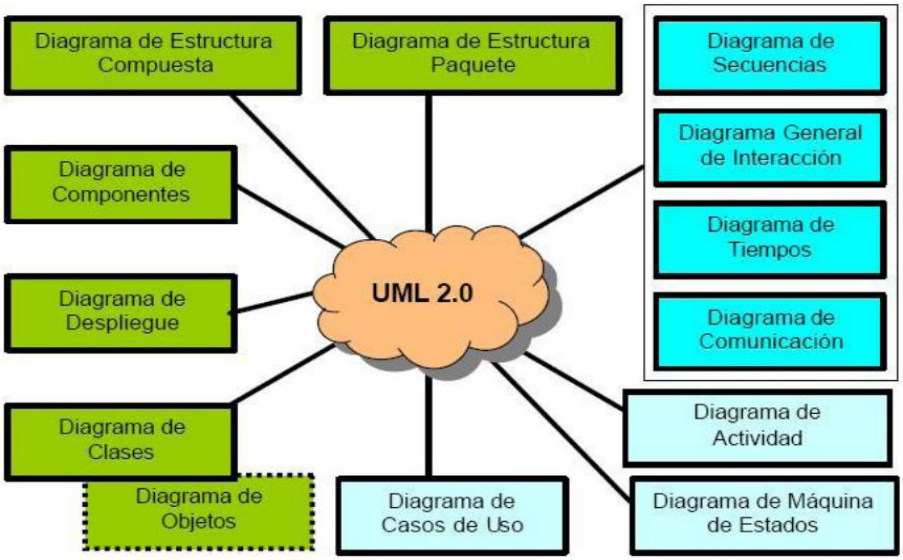


Figura 5. Diagramas de UML 2.0.

UML 2.0 es la mayor revisión que se le ha hecho a UML desde la versión 1.0. El modelo conceptual en el 2.0 ha sido reestructurado completamente y nuevos diagramas han sido incorporados. Los diagramas tradicionales también han sido mejorados, por ejemplo, los diagramas de secuencia y de despliegue; en el primero es posible hacer referencias a otros diagramas de secuencia, además de incluir flujos alternos, opcionales, lazos, entre otros y en el segundo, sobre los diferentes nodos de la infraestructura de red se colocan, a modo de artefactos (elementos físicos simples), los elementos componentes del software. (13)

En UML 2.0 los diagramas aparecen dentro de un marco (frame) que posee una etiqueta para indicar el tipo de diagrama, lo que permite al usuario una referencia rápida al diagrama invocado.

Esta versión proporciona a los analistas, arquitectos y desarrolladores; herramientas cada vez más potentes que les permite aprovechar mejor los modelos y como consecuencia generar una mayor cantidad de código reduciendo significativamente el ciclo de desarrollo de sus aplicaciones.

1.7 Plataforma de desarrollo de software.

Las Plataformas de Desarrollo de Software son una solución completa para desarrollar software y sistemas basados en software. Como parte de la necesidad de implantar una plataforma homogénea de diseño, modelado y desarrollo de software se investigaron tecnologías de licenciamiento libre (Software Libre) y propietario: Umbrello, BOUML y RSA.

Umbrello UML Modeller.

Es una herramienta de modelado en software libre bajo la licencia GPL. Soporta los 9 diagramas principales de diseño, así como un gran número de lenguajes de programación, entre ellos, python. Su interfaz es extremadamente simple y minimalista. En ella los diagramas se agrupan en vistas: vista lógica, vista de componentes, vista de casos de uso, de entidad-relación y las clases que se usan son comunes a todos. Su instalación es muy sencilla y ocupa poco espacio. Se encuentra en los repositorios oficiales de Ubuntu, con buena valoración por parte de la comunidad pero solo está disponible en plataformas GNU/Linux. La pantalla de diseño del diagrama está poco automatizada. Se basa en Drag & Drop, pero es demasiado simple. Las imágenes que genera son demasiado sencillas. (14)

BOUML.

Es una herramienta freeware (con las fuentes disponibles) que ofrece versiones compiladas para los principales sistemas operativos gracias al framework Qt. Es muy ligera y fácil de instalar. Permite visualizar la documentación del proyecto en html e importar el diseño a xml. Soporta los principales lenguajes orientados a objetos: C++, Java y PHP. Además permite incluir y exportar código automáticamente a partir del diseño y aunque no se encuentran en ella herramientas integradas en la propia aplicación lo permite hacer para las herramientas externas escritas en Java o C++. Por otra parte el número de lenguajes con los que permite trabajar es muy limitado y tiene una mala organización de los menús y la forma de agrupar los diagramas. (14)

IBM Rational Software Architect.

Es un entorno de modelación y desarrollo que aprovecha el Lenguaje Unificado de Modelado versión 2.0 (UML v2.0) para diseñar arquitecturas de software soportadas por C++, J2EE y aplicaciones Web. Aprovecha al máximo la plataforma abierta Eclipse 3.2. Ofrece una gama completa de herramientas de desarrollo de software para una variedad de tecnologías de implementación.

El Rational Software Architect (RSA) ayuda a migrar con más facilidad de una herramienta de desarrollo a otra. Permite agregar características a medida que cambien las necesidades, lo que hace del producto una selección consumible para las necesidades de diseño y de construcción. Además es único en su habilidad de dar soporte de modelado a otros dominios tales como WSDL y XSD (XML Schema Definition), permitiéndole crear diagramas que combinan los elementos de UML con estos últimos. (15)

Debido a las características del sistema en el que se trabaja se ha decidido utilizar la suite de herramientas de desarrollo de Rational que presenta ventajas debido a que su desarrollo está basado

en emplear el “framework” Eclipse, muy usado y conocido entre los desarrolladores en plataforma Linux. Además crea modelos en UML 2.0, desde la toma de requerimientos, la definición de las clases y los diagramas de secuencia, hasta la generación del código base que permite iniciar el desarrollo de una aplicación.

1.8 Tendencias y tecnologías actuales.

En la actualidad, durante el proceso de desarrollo de un software educativo en plataforma libre, se toman en cuenta una serie de tendencias y tecnologías informáticas, de manera que se identifican un grupo de ellas para el desarrollo de la investigación.

1.8.1 Sistema Operativo GNU/Linux.

Esta herramienta será desarrollada a partir de las herramientas que brinda el software libre. Entre las ventajas de utilizar este sistema operativo se cuenta el acceso libre, es decir, los usuarios pueden ejecutar, distribuir, modificar y mejorar estos programas, y publicar los cambios en las mismas condiciones de licenciamiento del programa original. Esta filosofía de trabajo busca la independencia tecnológica y la promoción de áreas de investigación y desarrollo nacional.
Sobre el sistema operativo GNU/Linux es importante destacar que posee una interfaz gráfica accesible a cualquier usuario que esté adaptado a otros entornos; brindando igual o similar operatividad y además puede coexistir en un mismo equipo informático con otro sistema operativo. (16)

1.8.1.2 Distribución Debian GNU/Linux.

Debian GNU/Linux es la principal distribución Linux del proyecto Debian, que basa su principio y fin en el software libre. El Proyecto Debian es una asociación de personas que han hecho causa común para crear un sistema operativo (SO) libre, bajo la licencia General Public Licence (LGPL).

Una gran parte de las herramientas básicas que completan el SO, vienen del proyecto GNU; de ahí el nombre: GNU/Linux. Debian es la única distribución importante de GNU/Linux mantenida solamente por sus usuarios lo cual trae ventajas y desventajas.

En primer lugar su distribución es libre y gratuita, tanto del SO como de las actualizaciones del mismo por lo que no es necesario piratear. Es uno de los SO mas estables en estos momentos y casi no existen los virus para el. Posee miles de paquetes pre-compilados estables, ya que, las personas que se dedican a Debian tienen una alta motivación en participar en la misma, y se actualiza la distribución diariamente.

En segundo lugar, dada su actitud abierta a la participación de todos, en el mismo espíritu original de Linux, constantemente hay personas que se unen a Debian para participar aportando su granito de arena, no solamente haciendo paquetes de programas, sino colaborando en todo lo que pueden: en el Servidor de Web, traduciendo documentación de Debian, documentando fallos, o ayudando a los usuarios a través de las listas de correo que mantiene la comunidad.

Como desventajas se puede decir que se necesita un mínimo de conocimiento en Linux para poder usar el sistema con comodidad. Tiene un mayor componente técnico que otras distribuciones. Es largo el tiempo que transcurre entre lanzamientos de versiones estables. Por ejemplo, pasaron casi tres años entre el lanzamiento de Debian3.0 y Debian3.1. Por otra parte la instalación es difícil para un usuario sin conocimientos en Linux. (17)

Se decidió usar la distribución Debian porque es un SO de libre distribución (es decir sin coste alguno). Permite a varios usuarios acceder al mismo tiempo a través de terminales, y distribuye los recursos disponibles entre todos. Puede correr en la mayoría de las plataformas que se encuentran en el mercado. Tiene un magnífico soporte de estabilidad en las aplicaciones (no requieren ser compiladas en la máquina que las esté usando) a diferencia de otras distribuciones. La memoria se gestiona como un recurso unificado para los programas de usuario y para el caché de disco, de tal forma que toda la memoria libre puede ser usada para caché y ésta puede a su vez ser reducida cuando se ejecuten grandes programas. Como es una distribución que ha probado su estabilidad y utilidad, muchos desarrolladores la han tomado para crear otras nuevas como: Knoppix, Ubuntu, Sidux, etc. Y los problemas de seguridad se solucionan rápidamente con parches de seguridad que se actualizan en internet.

1.8.2 Bibliotecas gráficas.

Los sistemas libres están adentrándose con creces en el ambiente de las interfaces gráficas de usuario, dejando atrás el mito del sistema operativo en modo texto, para ello emplean diferentes paquetes de desarrollo y además variadas tecnologías para la representación de los elementos gráficos.

A continuación se caracterizan las librerías para diseño gráfico más usadas actualmente en plataforma libre.

GTK.

GTK (GIMP Toolkit) es un grupo importante de bibliotecas o rutinas para desarrollar interfaces gráficas de usuario (GUI) para principalmente los entornos gráficos GNOME, XFCE y ROX de sistemas Linux.

GTK es la abreviatura de GIMP toolkit (conjunto de rutinas para GIMP).Es software libre (bajo la licencia LGPL), multiplataforma y parte importante del proyecto GNU. Inicialmente fue creado para desarrollar el programa de manejo de imágenes GIMP, sin embargo actualmente es muy usada por muchos otros programas en los sistemas GNU/Linux. Cabe mencionar que Qt es una alternativa a GTK que también es muy utilizada (en el entorno KDE, por ejemplo).GTK se ha diseñado para permitir programar con lenguajes como C, C++, C#, Java, Perl, PHP o Python. Actualmente su última versión es GTK+ 2, con una cantidad importante de mejoras respecto a la primera versión, aunque sin embargo, no es compatible con ella. (18)

QT-Designer.

Qt es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario. Fue creada por la compañía noruega Trolltech. Es utilizada fundamentalmente en KDE, un entorno de escritorio para sistemas como GNU/Linux o FreeBSD, entre otros. Utiliza el lenguaje de programación C++ de forma nativa y además existen bindings para C, Python (PyQt), Java (Qt Jambi), Perl (PerlQt) y Ruby (QtRuby) entre otros.

El API de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL, así como uso de XML y una multitud de otros para el manejo de ficheros, además de estructuras de datos tradicionales.

Qt Cuenta actualmente con un sistema de doble licencia: una GPL para el desarrollo de software de código abierto (open source) y software libre, y otra de pago para el desarrollo de aplicaciones comerciales.

Qt facilita la programación orientada a objetos y la creación de componentes, proporcionando sólidos cimientos para la construcción de cualquier tipo de aplicación gráfica. (19)

Una vez concluido el análisis de las bibliotecas hemos decidido utilizar la biblioteca Qt. La selección se debe a que posee un framework más completo y con mucha más utilidad a la hora de un desarrollo rápido, usando C++ como lenguaje de programación. Es muy respetado, una librería de muchas funcionalidades de alto rendimiento y prestaciones, documentación impecable y licencia GPL. GTK es una librería muy extendida pero con más bugs que Qt y documentación confusa, desactualizada y escasa. Haciendo un análisis profundo se puede observar que los programas que usan Qt son más ligeros y dan la sensación de mayor configurabilidad que sus contrapartes en GTK. Se puede concluir que las aplicaciones Qt son más rápidas, ligeras y óptimas.

CAPÍTULO 2: Características del sistema.

Introducción.

En el presente capítulo se exponen las características fundamentales que debe tener la Herramienta Autor de Objetos de Aprendizaje del sistema Emedia. Se realiza un modelo de Dominio, se capturan los requisitos funcionales y no funcionales que son necesarios para la confección de la aplicación, se identifican los casos de uso y se describen los que son arquitectónicamente significativos.

2.1 Sistema Emedia.

Emedia es un sistema que está en proceso de elaboración el cual contendrá cursos y OA, e incorporará herramientas para facilitar la comunicación y el trabajo colaborativo entre profesores y estudiantes, permitiendo el seguimiento y evaluación del alumno. Con estas herramientas se intenta lograr que el sistema no necesite la utilización de herramientas foráneas; sino que acceda a las herramientas integradas en su paquete para realizar el producto final. Esta característica brinda una facilidad de uso plena, debido a que la instalación de las funcionalidades para la producción del curso, sólo se hace una vez.

En una primera versión este sistema se implementará para escritorio (Desktop) con el objetivo de contribuir a su distribución mediante esta vía y posteriormente se llevará a la Web. Será una herramienta integrada que para su confección está compuesto por diferentes módulos; entre ellos se encuentra la herramienta Autor de Objetos de Aprendizaje que se encarga de crear estos objetos posibilitando poner en práctica los conocimientos del Experto a un costo y esfuerzo menor, y lograr la calidad que requiere la producción de cursos educativos.

El sistema Emedia usa como alternativa el Software Libre y las oportunidades que GNU/Linux le ofrece para su instalación, reduciendo los costos de informatización.

2.1.1 Herramienta Autor de Objetos de Aprendizaje.

Como solución a los objetivos propuestos en la presente investigación se propone desarrollar una Herramienta de Autor de Objetos de Aprendizaje, que funcionará como un componente de diseño independiente para la creación de OA, que se integrará al sistema Emedia.

La arquitectura propuesta persigue que la herramienta que se elabore pueda dar respuesta a los requerimientos de portabilidad, comprensibilidad, extensibilidad y eficiencia. Uno de los requerimientos que más estresan la solución es la cualidad de que el sistema pueda desplegarse en ambiente consola

tipo Escritorio y tipo Web. Se impone una dualidad de conceptos en los que se aboga por el desempeño que se logra con las aplicaciones nativas en modo de consolas gráficas, y por otro lado el empeño en la facilidad, ubiquidad y portabilidad que brindan las tecnologías Web.

En esta primera etapa del desarrollo de la herramienta Autor de Objetos de Aprendizaje, no se aborda la capacidad de la herramienta de ser visualizada sobre la Web, sino que funcione como una herramienta escritorio. Es necesario garantizar que la futura integración de este módulo al sistema Emedia esté en correspondencia con la arquitectura definida para la herramienta principal.

La herramienta Autor de Objetos de Aprendizaje del sistema Emedia permite importar un guión de contenido previamente realizado por el Experto en el cual se muestra la forma en que el mismo quiere que se muestren los contenidos. El Montador es el encargado de poner en práctica estos contenidos creando OA que reflejen lo expresado por el Especialista. Los OA se crean con metadatos que garantizan una mejor localización para la hora de realizar una búsqueda. Luego se le van agregando componentes que se pueden importar como video, audio, imágenes y animaciones, recursos que se encuentren en la aplicación así como también con plantillas predefinidas en la misma.

2.2 Modelo de Dominio.

Al no existir un negocio que defina los procesos del módulo de Herramienta Autor de OA del sistema Emedia; y para poder entender el contexto en que se desarrolla el mismo, es necesario definir conceptos que se puedan agrupar en un modelo de dominio.

El Modelo de Dominio (o Modelo Conceptual) es una representación de los conceptos u objetos del mundo real significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema. A continuación se realiza una descripción de los conceptos más significativos que se utilizan en este dominio para un mayor entendimiento del sistema.

Guión de Contenido: Incluirá todo lo relativo a la organización y estructuración de los contenidos. Como punto de partida podemos considerarlo como una primera aproximación a una representación hipertextual de los contenidos.

Objeto de Aprendizaje: Entidad informativa digital desarrollada para la generación de conocimiento. Un recurso digital que puede ser rehusado para ayudar en el aprendizaje.

Componente: Elementos que serán utilizados por el Montador para confeccionar los OA. Estos elementos pueden ser extraídos de los OA y reutilizados dentro de otros, por ejemplo: videos, imágenes, textos, crucigramas, test, encuestas, etc.

Montador: Es el único capacitado para el trabajo en la aplicación. Se ocupa de poner en práctica las ideas del Experto expresadas en el guión de contenido.

Plantilla: Patrón que sirve como modelo para organizar una información determinada.

El Modelo de Dominio se describe mediante un diagrama UML, específicamente con un diagrama de clases conceptuales significativas en el dominio del problema.

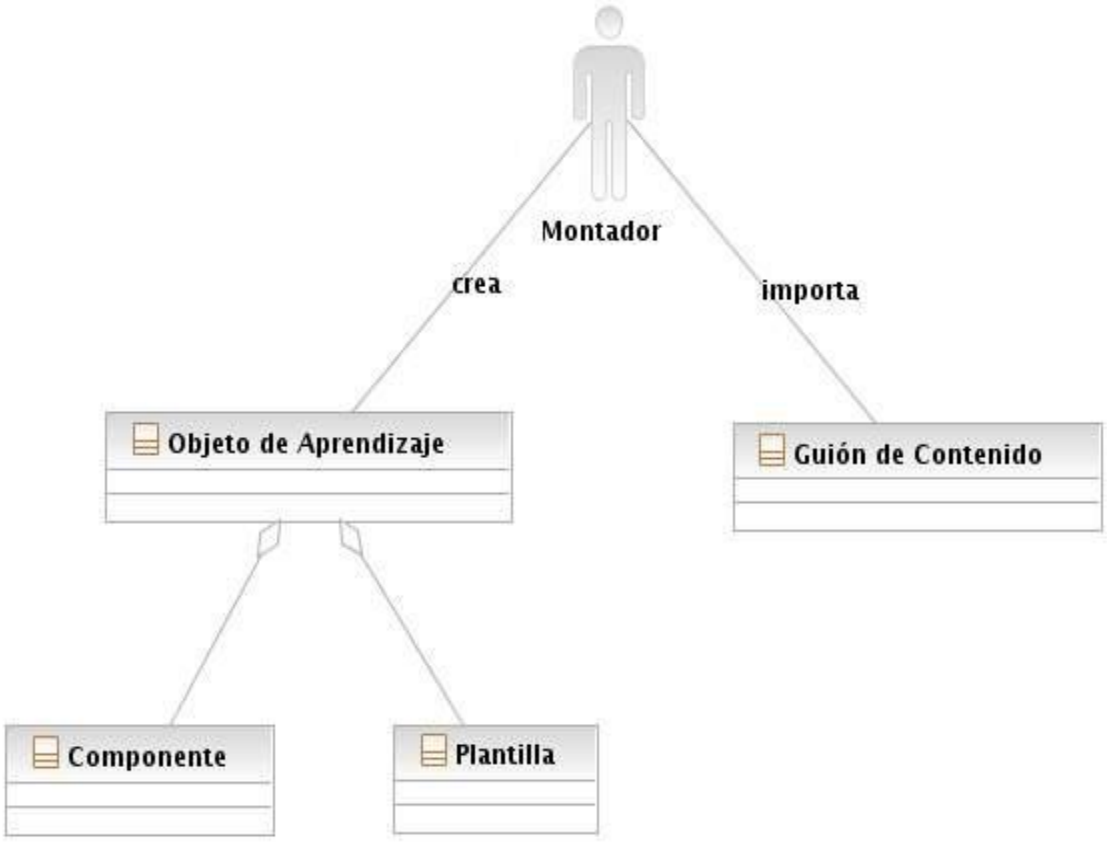


Figura 6. Modelo de Dominio

2.3 Modelación del Sistema.

En el presente epígrafe se identifica el actor que interactúa con la herramienta Autor de OA del sistema Emedia así como también se exponen un conjunto de requisitos funcionales y no funcionales que debe cumplir esta aplicación para su correcto funcionamiento.

2.3.1 Actor del Sistema.

Los actores representan personas o sistemas externos que interactúan con el sistema. Actualmente para la Herramienta Autor de OA se considera un perfil de usuario que es el único que estará capacitado para el trabajo en la aplicación.

Actor del Sistema	Justificación
Montador	Es quién interactúa con el sistema. Es el encargado de crear los Objetos de Aprendizaje, apoyado de las funcionalidades brindadas por la aplicación.

Tabla 2. Actor del sistema.

2.3.2 Requerimientos funcionales.

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Para el módulo de Herramienta Autor de OA que se está desarrollando ellos son:

RF 1. Gestionar guión de contenido.

RF 1.1 Importar guión de contenido.

RF 1.2 Cerrar guión de contenido.

RF 2 Crear OA (autor, título, fecha de creación, dirigido a, objetivos).

RF 3. Gestionar OA.

RF 3.1 Cargar OA existente.

RF 3.2 Modificar OA.

RF 4. Gestionar Proyecto.

RF 4.1 Adicionar proyecto.

RF 4.2 Abrir proyecto.

RF 4.3 Guardar proyecto.

RF 4.3 Cerrar proyecto.

RF 5. Gestionar Elemento Taxonómico.

RF 5.1 Adicionar un elemento taxonómico.

RF 5.2 Eliminar elemento taxonómico.

RF 6. Gestionar plantilla.

RF 6.1 Configurar plantilla (tipo de plantilla, título, puntuación, tiempo e intentos).

RF 6.2 Eliminar plantilla.

RF 7. Gestionar componente.

RF 7.1 Importar componente.

RF 7.2 Adicionar componente.

RF 7.3 Modificar propiedades del componente.

RF 7.4 Eliminar componente.

2.3.3 Requerimientos no funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Muestran las características que hacen al producto atractivo, usable, rápido o confiable. No alteran la funcionalidad del producto, esto quiere decir que los requerimientos funcionales se mantienen invariables sin importarle con que propiedades o cualidades se relacionen.

- **Requerimiento de Portabilidad.**

El sistema debe funcionar con sistemas operativos Windows o Linux.

- **Requerimiento asociado al Licenciamiento.**

Se debe garantizar que el sistema sea software libre y, por tanto, cualquier componente software que se utilice también deberá ser libre.

- **Requerimiento de Rendimiento y Disponibilidad del Sistema.**

Se debe garantizar el tiempo de acceso a las diferentes ventanas y despliegues del sistema en menos de dos segundos.

Se debe garantizar que las modificaciones realizadas sobre la configuración del algún objeto (datos, despliegues, entre otros) sean actualizadas y reflejadas en un tiempo de un segundo.

- **Requerimiento de Apariencia o interfaz externa.**

Se debe garantizar que el color sea gris como las aplicaciones en GNU/Linux. La interfaz debe ser de fácil comprensión en su funcionamiento permitiendo la utilización del sistema sin mucho entrenamiento.

- **Requerimiento de Interfaces de Usuario.**

La aplicación debe proveer al usuario una gran versatilidad en la presentación de la información en las pantallas, en cuanto a disponibilidad de los recursos y organización de los mismos. Debe permitir el

manejo de ambientes multi-ventanas o despliegues, visualizándose varios despliegues simultáneamente, sin generar efectos negativos en el rendimiento del sistema.

- Posibilidad de ocultar y mostrar las ventanas de configuración del proyecto en un área determinada; o sea, que aunque se oculten éstas se mantengan activas. También debe ofrecerse la posibilidad de cerrarlas.
- Mostrar el nombre de la aplicación en la parte superior izquierda mediante una barra de título.
- La aplicación debe estar sectorizada en varias áreas funcionales:
  - Para el Menú y las Barras de Herramientas, utilizar la parte superior.
  - Para los paneles de plantillas, videos, componentes, audio y animación utilizar el lateral izquierdo.
  - Para el panel de dependencia de los recursos de proyectos y del guión utilizar el lateral izquierdo.
  - Para la barra de dibujo y otros símbolos utilizar la parte inferior.
  - Para realizar el diseño de los OA, utilizar la parte central.
- Permitir mostrar opciones estándares (copiar, cortar, pegar…), pudiendo acceder mediante el clic derecho sobre un componente o por la combinación de teclas calientes.
- Permitir tener un menú que posea una serie de opciones que garanticen realizar acciones para el manejo y control de los objetos gráficos.

- Mostrar ventana de propiedades de cada recurso y componente.

- **Requerimiento de Diseño e Implementación.**

Se utilizará la biblioteca gráfica Qt-Designer para el diseño del prototipo no funcional y el IDE Eclipse para su futura implementación.

- **Requerimiento legal.**

La biblioteca se regirá por la licencia GNU/GPL.

- **Requerimiento de seguridad.**

La información manejada por el sistema será objeto de cuidadosa protección contra estados inconsistentes, esto se refiere a que debe tener validados todos los datos de entrada, así como también la definición de los tipos de variables que se almacenaran.

2.4 Diagramas de casos de uso.

Luego de haber definido el actor y los requerimientos funcionales del sistema, se modela la relación que existe entre el actor Montador con los casos de usos definidos a partir de la captura de los requerimientos.

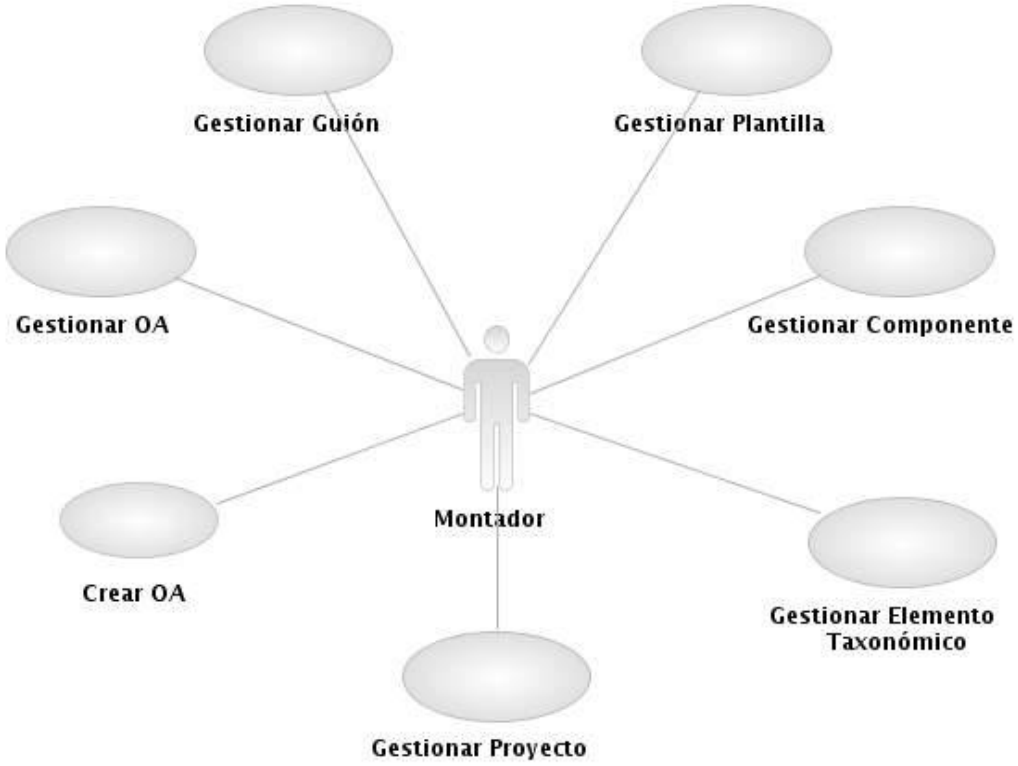


Figura 7. Diagrama de casos de uso del sistema.

2.5 Descripción de los casos de usos.

El objetivo principal de detallar cada caso de uso es describir su flujo de sucesos en detalle, incluyendo cómo comienza, termina e interactúan con los actores. (11)

A continuación se muestra la descripción de los casos de uso más significativos del módulo Herramienta Autor de Objetos de Aprendizaje.

CU Gestionar guión de contenido.

Nombre del CU	**Gestionar guión de contenido.**
Actor	Montador
Resumen	El CU inicia cuando el Montador decide importar un guión de contenido para comenzar a crear sus respectivos Objetos de Aprendizaje.
Referencia	**RF 1, RF 1.1, RF 1.2.**
Precondición	
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El Montador desea importar o cerrar un guión de contenido.	1.1 Si decide importar un guión ir a la Sección "Importar". 1.2 Si decide cerrar un guión ir a la Sección "Cerrar Guión".
Sección "Importar Guión"	
2. El Montador va al menú Archivo y da clic en la opción de Importar.	2.1 El Sistema muestra una interfaz de búsqueda donde permite localizar el directorio en que se encuentra el guión.
3. El Montador localiza el directorio, selecciona el guión que desea y presiona el botón "Aceptar".	3.1 El Sistema muestra el guión de contenido en el Explorador de Proyecto que se encuentra en el lateral izquierdo de la aplicación.
Sección "Cerrar Guión"	
4. El Montador da clic en la carpeta contenedora del guión, va al menú Archivo y selecciona la opción de Cerrar.	4.1 El Sistema muestra una ventana de confirmación.
5. El Montador da clic en el botón Aceptar.	5.1 El sistema elimina el guión de la aplicación.
Curso Alterno	
3- El Montador da clic en el botón "Cancelar".	3.1 El Sistema regresa a su estado inicial.
5. El Montador da clic en el botón "Cancelar".	5.1 El Sistema mantiene su estado.
Poscondición	Se importó y cerró un guión de contenido.
Prioridad	Crítica.

CU Gestionar OA.

Nombre del CU	**Gestionar OA.**
Actor	Montador
Resumen	El CU inicia cuando el Montador desea cargar para realizar modificaciones sobre él, con el objetivo de satisfacer necesidades de un guión. El caso de uso finaliza cuando se ha cargado o modificado un OA.
Referencia	**RF 3, RF 3.1, RF 3.2, RF 3.3.**
Precondición	Se debe haber creado un proyecto nuevo.
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El Montador desea cargar o modificar un OA.	1.1 Si decide cargar un OA ir a la Sección "Cargar OA".

	1.2 Si decide modificar un OA ir a la Sección "Modificar un OA".
Sección "Cargar OA"	
2. El Montador selecciona en el menú Archivo la opción de Cargar.	2.1 El Sistema muestra una interfaz en la que brinda la posibilidad de localizar el directorio en el que se encuentra el OA.
3. El Montador localiza el directorio y da clic en el botón Aceptar.	3.1 El Sistema visualiza el proyecto de OA en el Explorador de Proyecto.
Sección "Modificar un OA"	
4. El Montador va al elemento de la taxonomía del proyecto de OA que desea modificar y da clic.	4.1 El Sistema muestra el contenido del elemento.
5. El Montador modifica el contenido del elemento que estima pertinente.	5.1 El Sistema guarda las modificaciones realizadas.
Curso Alterno	
3- El Montador presiona el botón Cancelar.	5.1 El Sistema regresa a su estado inicial.
Poscondición	Se cargó y modificó un OA.
Prioridad	Crítica

CU Crear OA.

Nombre del CU	**Crear OA.**
Actor	Montador
Resumen	El CU inicia cuando el Montador decide crear un OA para satisfacer las necesidades de un guión de contenido.
Referencia	**RF 2.**
Precondición	Debe haberse adicionado un elemento taxonómico al proyecto.
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El Montador selecciona el elemento taxonómico, va al menú Objeto y da clic en la opción de Crear.	1.1 El Sistema muestra una interfaz para llenar datos como Autor, Título, Fecha de creación, Dirigido a y Objetivos.
2. El Montador introduce los datos correspondientes	2.1 El Sistema verifica que todos los campos han sido llenados. 2.2 El Sistema guarda los datos entrados.
Curso Alterno	
	2.1- Si no están introducidos todos los datos, el Sistema muestra la ventana nuevamente indicando entrar los datos en los campos vacíos.
3. El Montador introduce los campos faltantes.	3.1 El Sistema guarda los datos entrados.
Poscondición	Se creó un OA.
Prioridad	Crítica.

CU Gestionar componente.

Nombre del CU	**Gestionar componente.**
Actor	Montador
Resumen	El CU inicia cuando el Montador decide importar un componente para adicionar a un OA.
Referencia	**RF 7, RF 7.1, RF 7.2, RF 7.3, RF 7.4.**
Precondición	Debe haberse creado un OA.
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1 El Montador desea importar o modificar las propiedades de un componente.	1.1 Si decide importar un componente ir a la Sección "Importar Componente". 1.2 Si decide modificar las propiedades de un componente ir a la Sección "Modificar Propiedades".
Sección "Importar Componente"	
2. El Montador va al menú Archivo y selecciona la opción de Importar.	2.1 El Sistema muestra una ventana de búsqueda, la cual permite localizar la ruta donde se encuentra el componente que desea importar.
3. El Montador localiza la ruta, selecciona el componente y presiona el botón Aceptar.	3.1 El sistema realiza una de estas acciones: 3.1.1 Si el componente que selecciona el Montador es un video ir a la Sección "Video". 3.1.2 Si el componente que selecciona el Montador es una animación ir a la Sección "Animación". 3.1.3 Si el componente que selecciona el Montador es de audio ir a la Sección "Audio". 3.1.4 Si el componente que selecciona el Montador es de imagen ir a la Sección "Imagen".
	Sección "Video"
	3.1.1 El Sistema activa el panel de video y muestra el componente en la biblioteca de video de dicho panel.
	Sección "Animación"
	3.1.2 El Sistema activa el panel de animación y muestra el componente en la biblioteca de animación de dicho panel.
	Sección "Audio"
	3.1.3 El Sistema activa el panel de audio y muestra el componente en la biblioteca de audio de dicho panel.
	Sección "Imagen"
	3.1.4 El Sistema activa el panel de imagen y muestra el componente en la biblioteca de imagen de dicho panel.
Sección "Modificar Propiedades"	
4. El Montador selecciona el componente al que desea modificar sus propiedades y da clic en el panel de Propiedades.	4.1 El Sistema muestra un conjunto de propiedades que se refieren a largo, ancho y cambiar el nombre al componente.
5. El Montador modifica las propiedades que	5.1 El Sistema muestra el componente con las

desea y da clic en el botón Aplicar.	modificaciones realizadas.
Curso Alterno	
3- El Montador pulsa el botón “Cancelar”.	3.1 El Sistema regresa a su estado inicial.
Poscondición	Se importó, y modificó las propiedades de un componente.
Prioridad	Crítica.

CU Gestionar plantilla.

Nombre del CU	**Gestionar plantilla.**
Actor	Montador
Resumen	El CU inicia cuando el Montador decide crear una plantilla para un OA.
Referencia	**RF 6, RF 6.1, RF 6.2, RF 6.3, RF 6.4.**
Precondición	Debe haberse creado un OA.
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El Montador desea configurar o eliminar una plantilla.	1.1 Si decide configurar una plantilla ir a la Sección “Configurar Plantilla”. 1.2 Si decide eliminar una plantilla ir a la Sección “Eliminar Plantilla”.
Sección “Configurar Plantilla”	
2. El Montador da clic en el panel de plantilla, selecciona la que desea y la arrastra hacia el OA.	2.1 El Sistema muestra una interfaz donde le da la posibilidad al Montador de configurar propiedades de la plantilla. Estas propiedades se refieren a datos como: tipo de plantilla, título, puntuación, tiempo e intentos.
3. El Montador llena los datos y da clic en el botón Aceptar.	3.1 El Sistema verifica que todos los campos han sido llenados. 3.2 El Sistema guarda los datos entrados. 3.3 El Sistema muestra la plantilla con las configuraciones realizadas.
Sección “Eliminar Plantilla”	
4. El Montador selecciona la plantilla que desea eliminar del OA y va a la opción de Eliminar en el menú Plantilla y da clic.	4.1 El Sistema elimina la plantilla del OA.
Curso Alterno	
	3.1- Si no están introducidos todos los datos, el Sistema muestra la ventana nuevamente indicando entrar los datos en los campos vacíos.
4. El Montador introduce los campos faltantes.	4.1 El Sistema guarda los datos entrados.
3. El montador pulsa el botón “Cancelar”	3.1 El sistema regresa a su estado inicial.
Poscondición	Se creó y eliminó una plantilla.
Prioridad	Crítica.

CU Gestionar Elemento Taxonómico.

Nombre del CU	**Gestionar Elemento Taxonómico.**
Actor	Montador
Propósito	Permitir al montador adicionar y eliminar un elemento taxonómico.
Resumen	El CU inicia cuando el montador decide adicionar a un proyecto un nuevo elemento taxonómico para crear un Objeto de Aprendizaje. Finaliza cuando se adiciona o elimina un elemento taxonómico.
Referencia	**RF 5, RF 5.1, RF 5.2**
Precondición	Debe haberse creado un proyecto.
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El montador necesita adicionar o eliminar un elemento taxonómico.	1.1 Si decide adicionar un elemento taxonómico ir a la Sección “Adicionar Elemento”. 1.2 Si decide eliminar un elemento taxonómico ir a la Sección “Eliminar Elemento”.
Sección “Adicionar Elemento”	
2. El montador da clic derecho en el árbol de la taxonomía, donde desea adicionar un elemento y selecciona la opción de Nuevo.	2.1 El sistema le adiciona al árbol un nuevo elemento taxonómico.
Sección “Eliminar Elemento”	
5. El Montador da clic derecho en el elemento taxonómico que desea eliminar y escoge la opción Eliminar.	5.1 El Sistema elimina el elemento. 5.2 El Sistema actualiza el Explorador de Proyecto.
Curso Alterno	
Poscondición	Se ha adicionado y eliminado un elemento taxonómico.
Prioridad	Media

CU Gestionar Proyecto.

Nombre del CU	**Gestionar Proyecto.**
Actor	Montador
Resumen	El CU inicia cuando el Montador decide adicionar su propio proyecto para a partir de ahí crear Objetos de Aprendizaje.
Referencia	**RF 4, RF 4.1, RF 4.2, RF 4.3, RF 4.4.**
Precondición	
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El Montador necesita adicionar, o abrir un proyecto.	1.1 Si decide adicionar un proyecto ir a la Sección “Adicionar Proyecto”. 1.2 Si decide abrir un proyecto ir a la Sección “Abrir Proyecto”. 1.3 Si decide cerrar un proyecto ir a la Sección “Cerrar proyecto”.

Sección "Adicionar Proyecto"	
2. El Montador va al menú Archivo y selecciona la opción de Nuevo.	2.1 El Sistema muestra una interfaz en la que da la posibilidad de poner el nombre del proyecto y una breve descripción.
3. El Montador llena los datos y da clic en el botón Aceptar.	3.1 El Sistema crea una carpeta con el nombre entrado en el Explorador de Proyecto.
Sección "Abrir Proyecto"	
4. El Montador da clic en el signo de más que indica desplegar el proyecto.	4.1 El Sistema muestra el contenido del proyecto.
Sección "Cerrar Proyecto"	
6. El Montador se sitúa en la carpeta con el nombre de proyecto, va al menú Archivo y selecciona la opción de Cerrar.	6.1 El Sistema elimina el proyecto del Explorador de Proyecto.
Curso Alterno	
3. El montador pulsa el botón "Cancelar"	3.1 El sistema regresa a su estado inicial.
5. El montador pulsa el botón "Cancelar"	5.1 El sistema regresa a su estado inicial.
Poscondición	Se creó, abrió y cerró un proyecto.
Prioridad	Media

CAPÍTULO 3: Análisis y diseño.

Introducción.

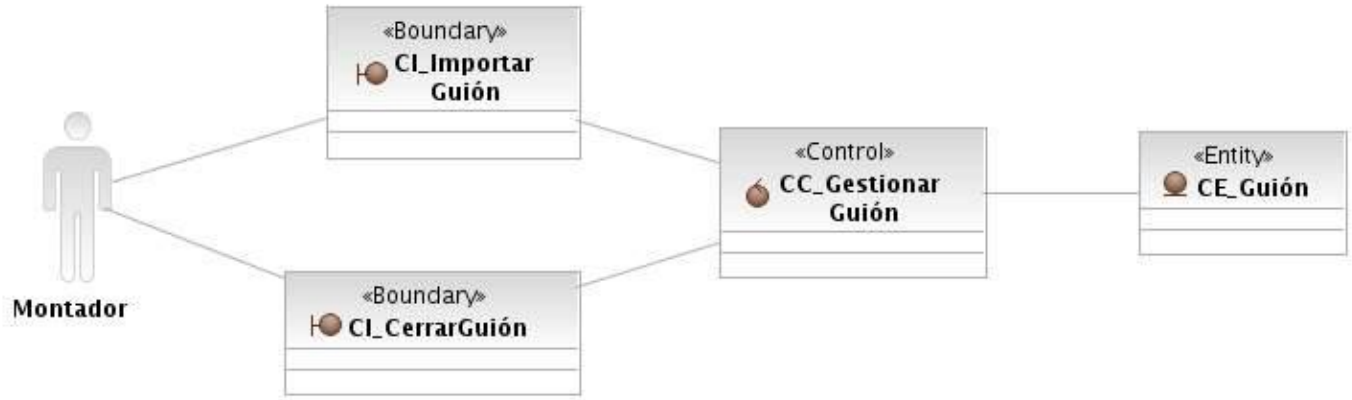
A partir del diagrama de casos de uso obtenido como parte del flujo de trabajo de requerimientos, se puede establecer una traza con la disciplina que continúa el ciclo de vida de RUP: análisis y diseño, esta se encarga de describir en términos de clases cómo debe funcionar cada caso de uso y las colaboraciones que existen entre las clases definidas. De esta forma se obtienen los diagramas de clases de análisis correspondientes a cada caso de uso, conformando las realizaciones de los mismos conjuntamente con los diagramas de interacción hasta llegar a refinar este análisis en el diseño esperado para la aplicación a desarrollar, con el objetivo de transformar los requerimientos, al diseño que el sistema debe tener y con la finalidad de adaptarlo al ambiente de implementación, trazando su funcionamiento. Para el desarrollo satisfactorio de esta disciplina la metodología propone el uso de patrones con el objetivo de lograr una arquitectura robusta para el sistema en cuestión.

3.1 Diagramas de clases del Análisis.

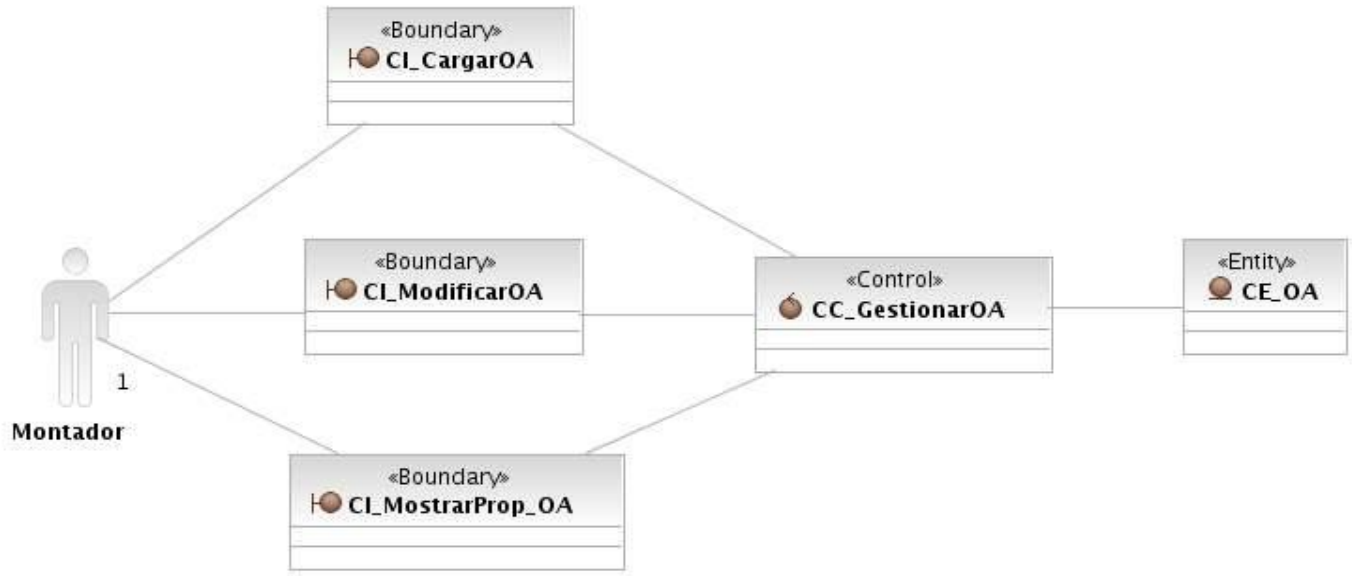
El análisis consiste en obtener una visión del sistema que se preocupa de ver que hace el mismo, de modo que sólo se interesa por los requisitos funcionales.

Los diagramas de clases de análisis, expresan la definición y relación entre las clases. Estas clases pueden ser de tres tipos fundamentales: interfaz, controladora y entidad. Las clases interfaz son las que se encargan de brindar un entorno gráfico amigable y entendible para que el usuario interactúe con el sistema de manera que modelan la interacción entre el sistema y sus actores. Las clases controladoras se encargan de captar toda la información que es enviada a través de las clases interfaz y guardar esos datos a través de métodos en la Base de Datos (BD), coordinando la realización de uno o unos pocos casos de uso así como las actividades de los objetos que implementan la funcionalidad del caso de uso. Las clases entidades generalmente modelan información que posee larga vida y que es a menudo persistente, estas clases son las futuras tablas de la BD. A continuación se muestran los diagramas de clases del análisis de la herramienta Autor de Objetos de Aprendizaje.

CU Gestionar Guión.

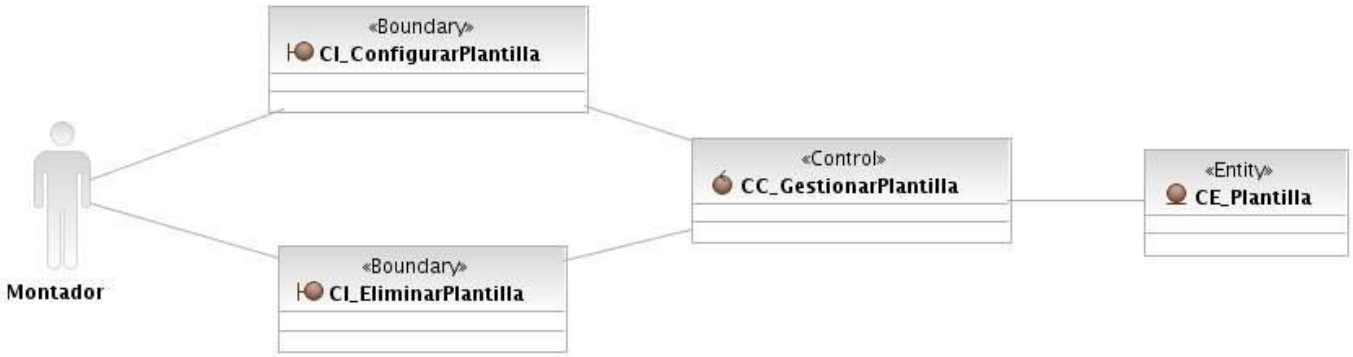


CU Gestionar OA.

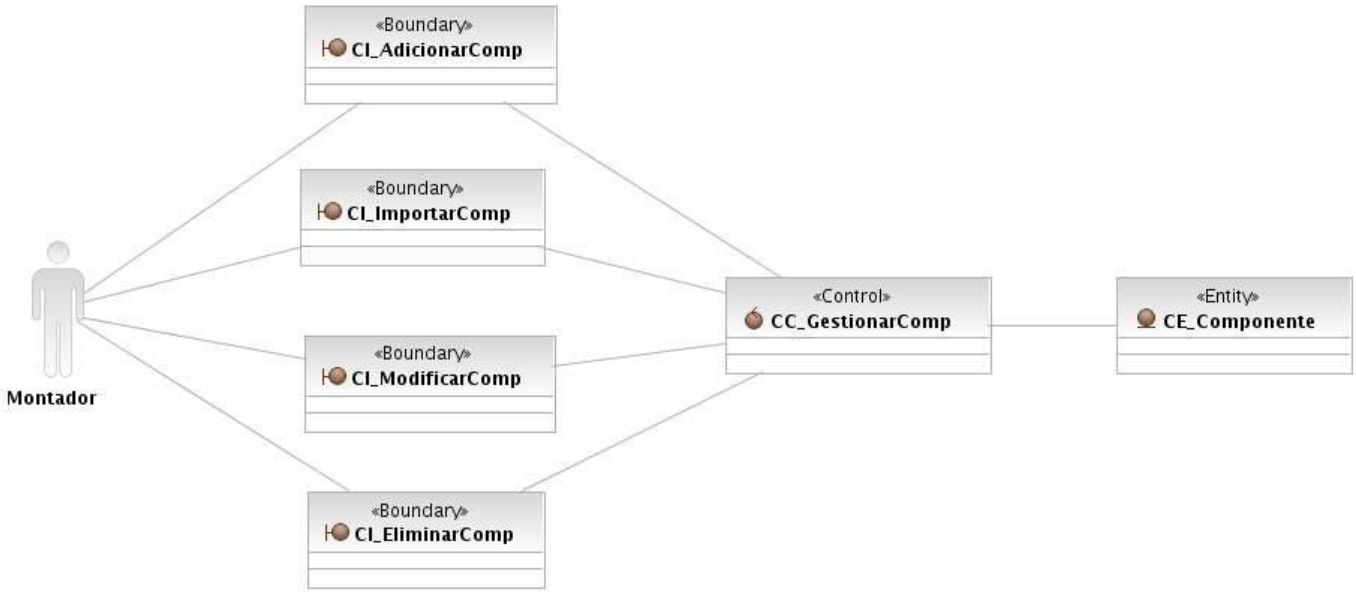


CU Crear OA.

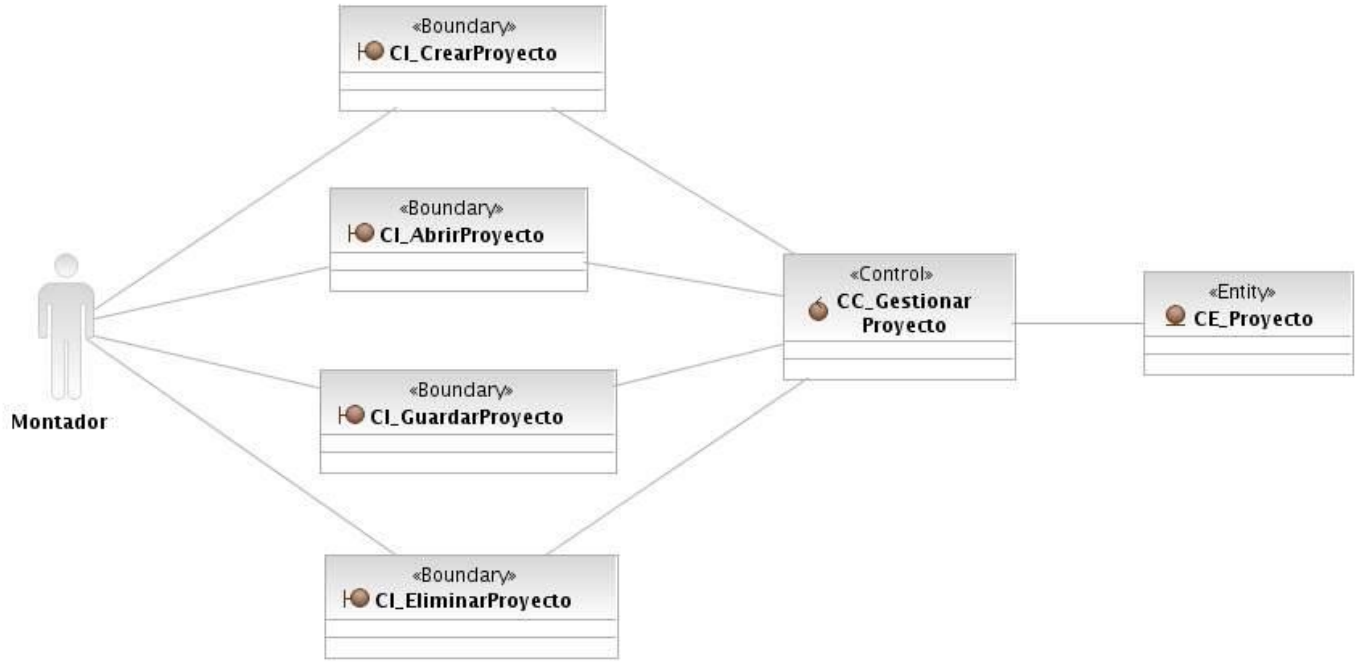
CU Gestionar plantilla.



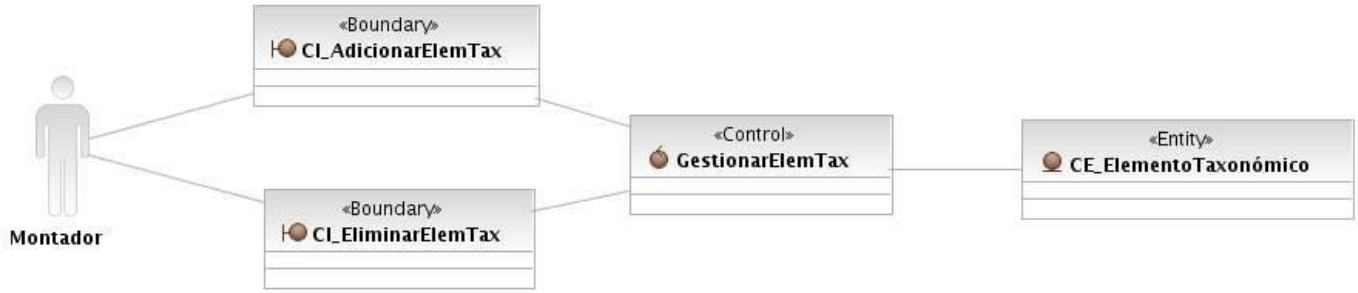
CU Gestionar Componente.



CU Gestionar proyecto.



CU Gestionar Elemento Taxonómico.



3.2 Diseño Arquitectónico.

El diseño es una actividad en la que se toman decisiones importantes, frecuentemente de naturaleza estructural. Comparte con la programación un interés por la abstracción de la información y de las secuencias del procesamiento, pero el nivel de detalle es muy diferente en ambos casos. El diseño

constituye representaciones coherentes y bien planificadas de los programas, concentrándose en las interrelaciones de los componentes de mayor nivel y en las operaciones lógicas implicadas en los niveles inferiores. La arquitectura de software es la estructura general de un sistema. Esta comprende los componentes de software, las propiedades de esos componentes visibles externamente, y las relaciones de ellos. Razones diferentes fundamentan la importancia de la arquitectura de software:

- Las representaciones de la arquitectura de software facilitan la comunicación entre todas las partes interesadas en el desarrollo de un sistema basado en computadoras.
- La arquitectura destaca decisiones tempranas de diseño que tendrán un profundo impacto en todo el trabajo de ingeniería de software que sigue, y es tan importante en el resultado final del sistema como una entidad operacional.
- Constituye un modelo pequeño e intelectualmente comprensible de como está estructurado el sistema y de cómo trabajan juntos sus componentes. (22)

3.2.1 Patrón Modelo-Vista-Controlador.

La arquitectura que se decidió escoger para el diseño de la herramienta Autor de Objetos de Aprendizaje fue la del Modelo Vista Controlador (MVC) divide una aplicación interactiva en tres áreas: procesamiento, salida y entrada. Para esto, utiliza las siguientes abstracciones:

- Modelo (Model): Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.
- Vista (View): Muestra la información del usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.
- Controlador (Controller): Reciben las entradas, usualmente como eventos que codifican los movimientos o pulsaciones de botones del mouse, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicios (“services requests”) para el modelo o la vista.

Se han desarrollado a lo largo de los años, desde la presentación de este patrón a la comunidad científica variantes fundamentales, que se presentan brevemente a continuación.

Variante I: Variante en la cual no existe ninguna comunicación entre el Modelo y la Vista, y esta última recibe los datos a mostrar a través del Controlador.

Variante II: Variante en la cual se desarrolla una comunicación entre el Modelo y la Vista, donde esta última al mostrar los datos los busca directamente en el Modelo, dada una indicación del Controlador, disminuyendo el conjunto de responsabilidades este último.

Variante III: Variante en la cual se diversifican las funcionalidades del Modelo teniendo en cuenta las características de las aplicaciones multimedia, donde tiene un gran peso las medias utilizadas en estas. (23)

Para la realización de la arquitectura de la presente investigación se utilizó la **Variante I** del patrón MVC. (Ver Figura 10)

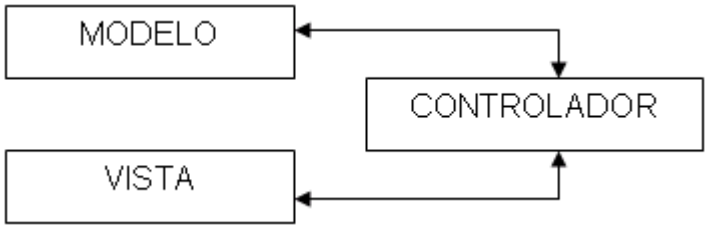


Figura 10: Variante inicial del Patrón MVC.

3.3 Patrones de Diseño.

De manera general los patrones constituyen soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos, basados en la experiencia y que se ha demostrado que funcionan. (24)

Los patrones son parejas de problema/solución con un nombre, que codifican nuevos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. Asignar correctamente las responsabilidades es muy importante en el diseño orientado a objetos. (25)

3.3.1 Patrones GOF.

Los Patrones GOF generalmente se evidencian en clases que son creadas debido al uso de un patrón específico. Existe un grupo de patrones de este tipo definidos para el diseño y con el propósito de crear una arquitectura robusta para el sistema a desarrollar. Del gran número de patrones propuestos por la pandilla de los cuatro o simplemente GOF, se propone el uso del patrón Fábrica Pura. Este tiene como principal función crear objetos sin importar en qué momento y a que clase pertenecen.

3.3.1.1 Fábrica Pura.

Tipo: Creación, a nivel de clases.

Propósito: Definir una interfaz para la creación de un objeto, pero permitiendo a las subclases decidir de que clase instanciarlo. Permite, por tanto, que una clase difiera la instanciación en favor de sus subclases.

Cuando usarlo:

Cuando una clase no puede adelantar las clases de objetos que debe crear.

Cuando una clase pretende que sus subclases especifiquen los objetos que ella crea.

Cuando una clase delega su responsabilidad hacia una de entre varias subclases auxiliares y queremos tener localizada a la subclase delegada.

Ventajas: Se gana en flexibilidad, puedes crear los objetos dentro de una clase con un "Método de Fábrica" es siempre más flexible que hacerlo directamente, debido a que se elimina la necesidad de atar nuestra aplicación a unas clases de productos concretos.

Se facilitan futuras ampliaciones, puesto que se ofrece a las subclases la posibilidad de proporcionar una versión extendida de un objeto, con sólo aplicar a los Productos la misma idea del "Método de Fábrica".

Se facilita, en cuanto a que se hace natural. La conexión entre jerarquías de clases paralelas, son aquellas que se generan cuando una clase delega algunas de sus responsabilidades en una clase aparte. Ambas jerarquías de clases paralelas son creadas por un mismo cliente y el patrón *Método de Fábrica* establece la relación entre parejas de subclases concretas en las mismas.

3.4 Diagramas de clases del diseño.

Para la elaboración del diseño, el patrón Fábrica se modela dentro de un paquete al que acceden todas las clases controladoras del negocio a través de un repositorio que gestiona los procesos fundamentales y que al hacer uso del subsistema fábrica estaría llamando a todas las interfaces de cada una de las entidades del negocio. Estas interfaces contienen métodos abstractos y se encargan de realizar las operaciones básicas sobre los datos: insertar, modificar, eliminar y seleccionar.

Para el desarrollo del módulo de Herramienta Autor de Objetos de Aprendizaje del sistema Emedia, se ha creado un paquete que contiene las realizaciones de cada uno de los casos de uso arquitectónicamente significativos definidos en el flujo de trabajo de requerimientos.

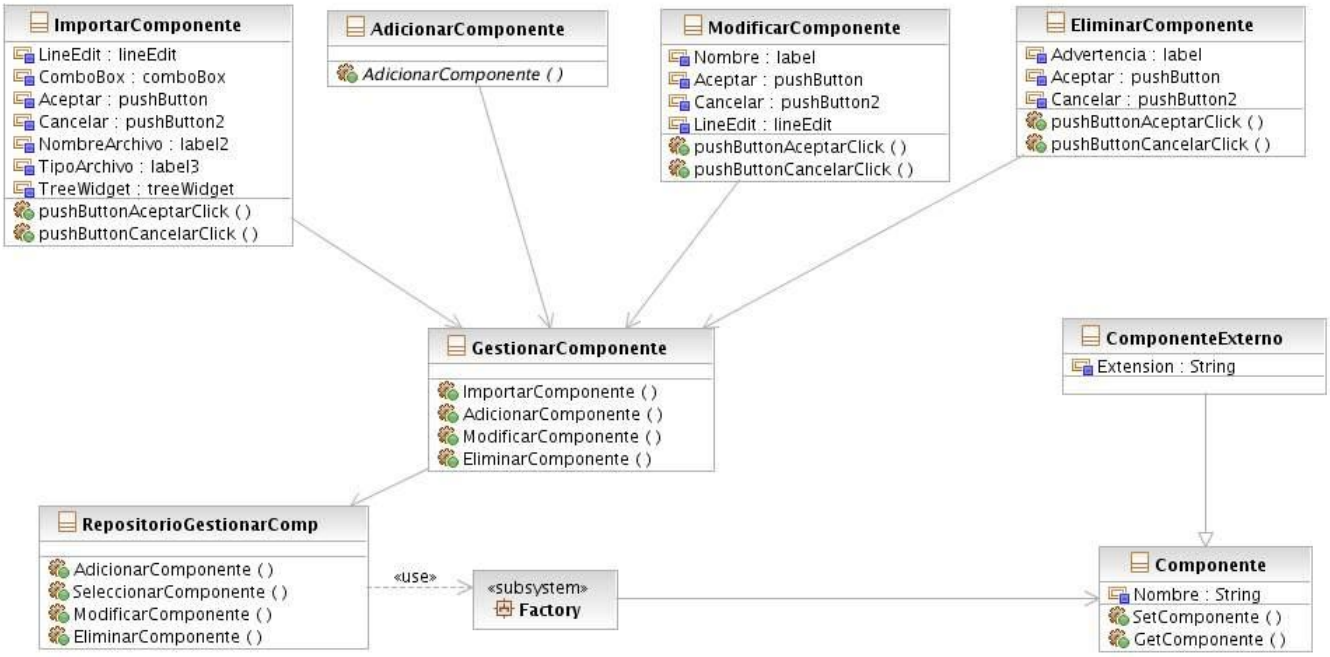
Estas realizaciones contienen los diagramas de clases del diseño que evidencian las clases que se definen para el desarrollo del sistema, así como sus correspondientes diagramas de interacción que en

este caso se ha decidido optar por la elaboración de diagramas de secuencia que manejan objetos de las diferentes clases y muestran una secuencia lógica de llamadas entre los objetos, brindándole a los futuros desarrolladores una imagen clara de cómo deben implementar el software.

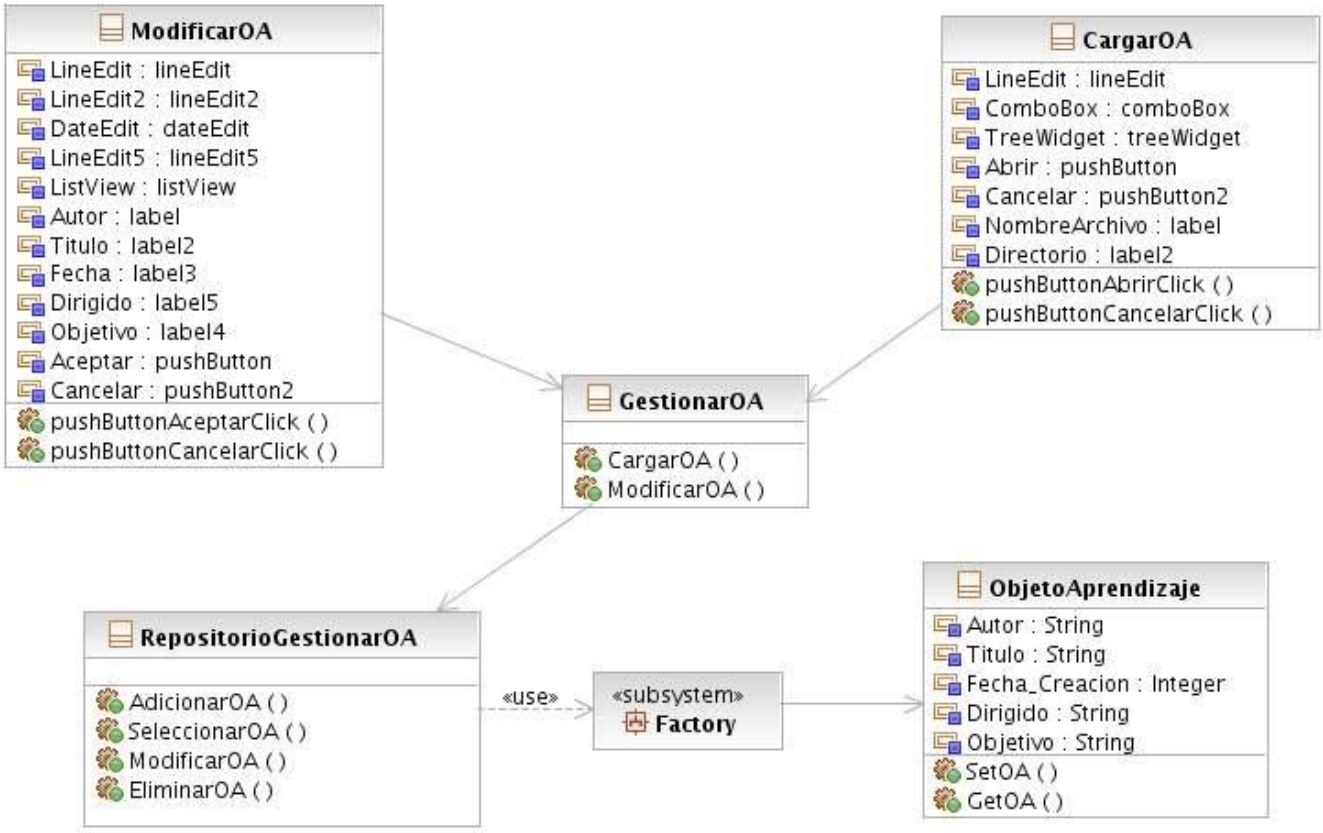
En cada uno de los diagramas que se muestran se incluye el paquete Fábrica, necesario e indispensable para poder acceder a través de alguna de las interfaces del mismo a las entidades definidas en el diseño.

Cada caso de uso contiene un repositorio que actúa como una clase controladora y que es el encargado de seleccionar la interface exacta que va a utilizar para implementar cada proceso descrito.

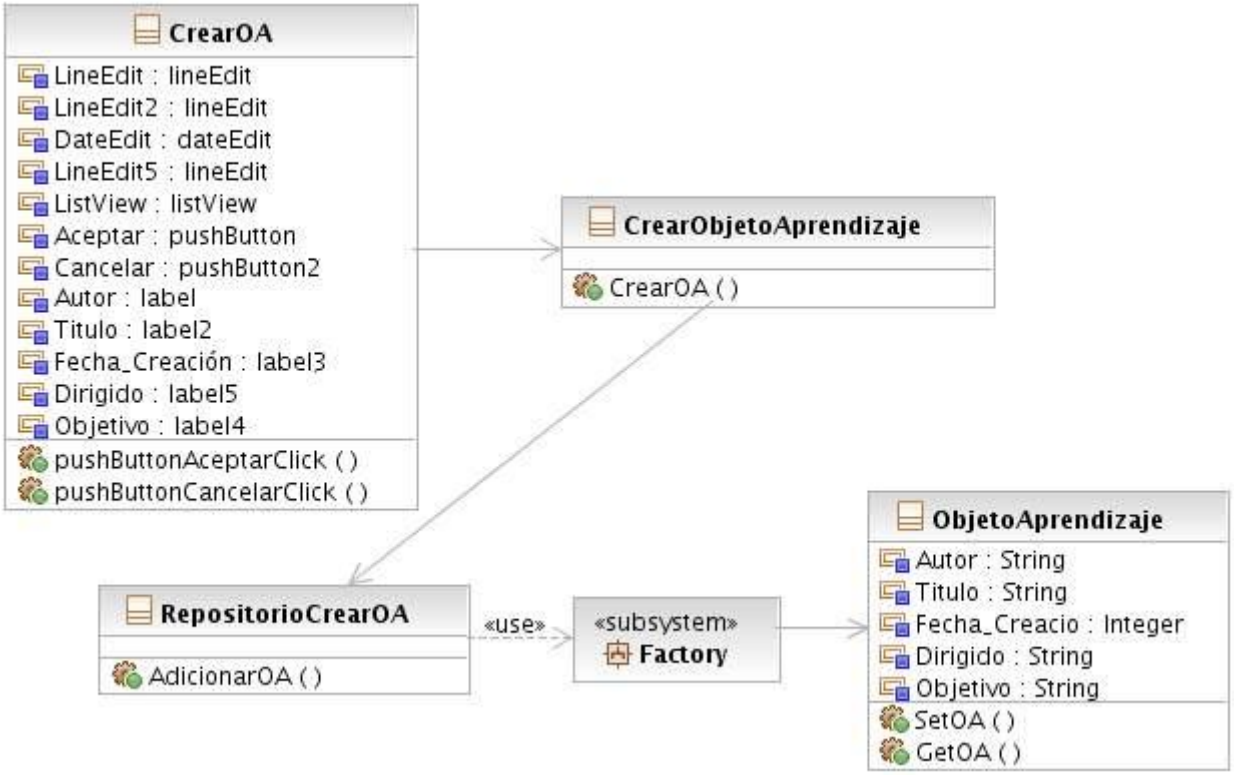
CU Gestionar Guión.



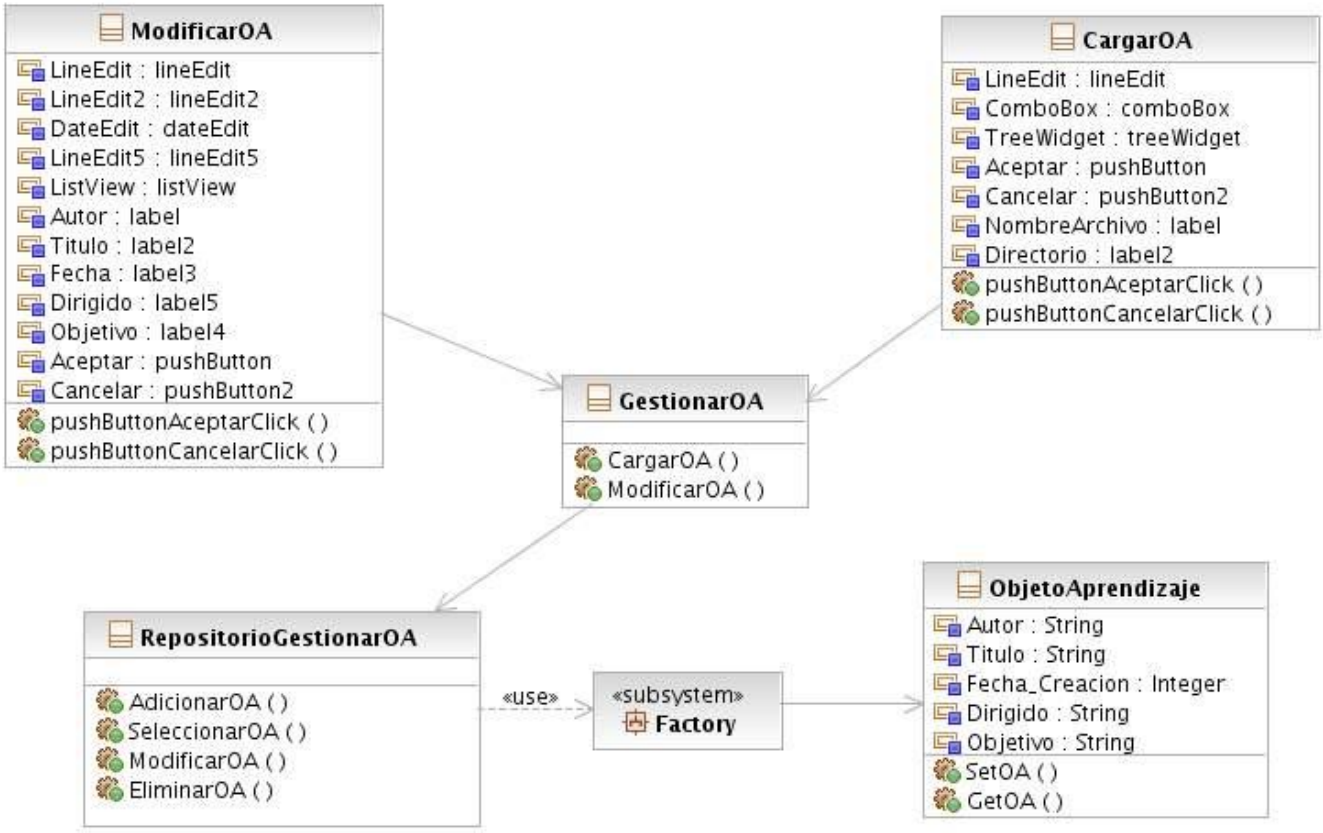
CU Gestionar OA.



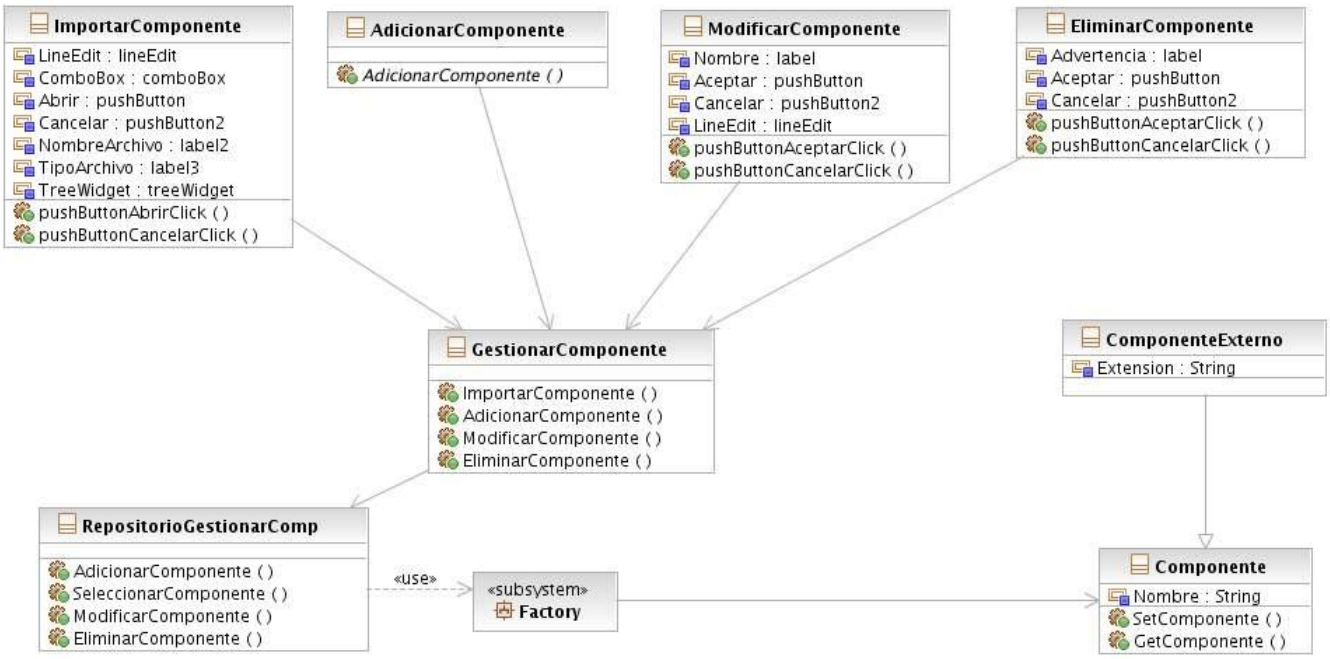
CU Crear OA.


CrearOA
LineEdit : lineEdit
LineEdit2 : lineEdit
DateEdit : dateEdit
LineEdit5 : lineEdit
ListView : listView
Aceptar : pushButton
Cancelar : pushButton2
Autor : label
Titulo : label2
Fecha_Creación : label3
Dirigido : label5
Objetivo : label4
pushButtonAceptarClick ( )
pushButtonCancelarClick ( )
CrearObjetoAprendizaje
CrearOA ( )
RepositorioCrearOA
AdicionarOA ( )
«use»
«subsystem»
Factory
ObjetoAprendizaje
Autor : String
Titulo : String
Fecha_Creacio : Integer
Dirigido : String
Objetivo : String
SetOA ( )
GetOA ( )

CU Gestionar plantilla.

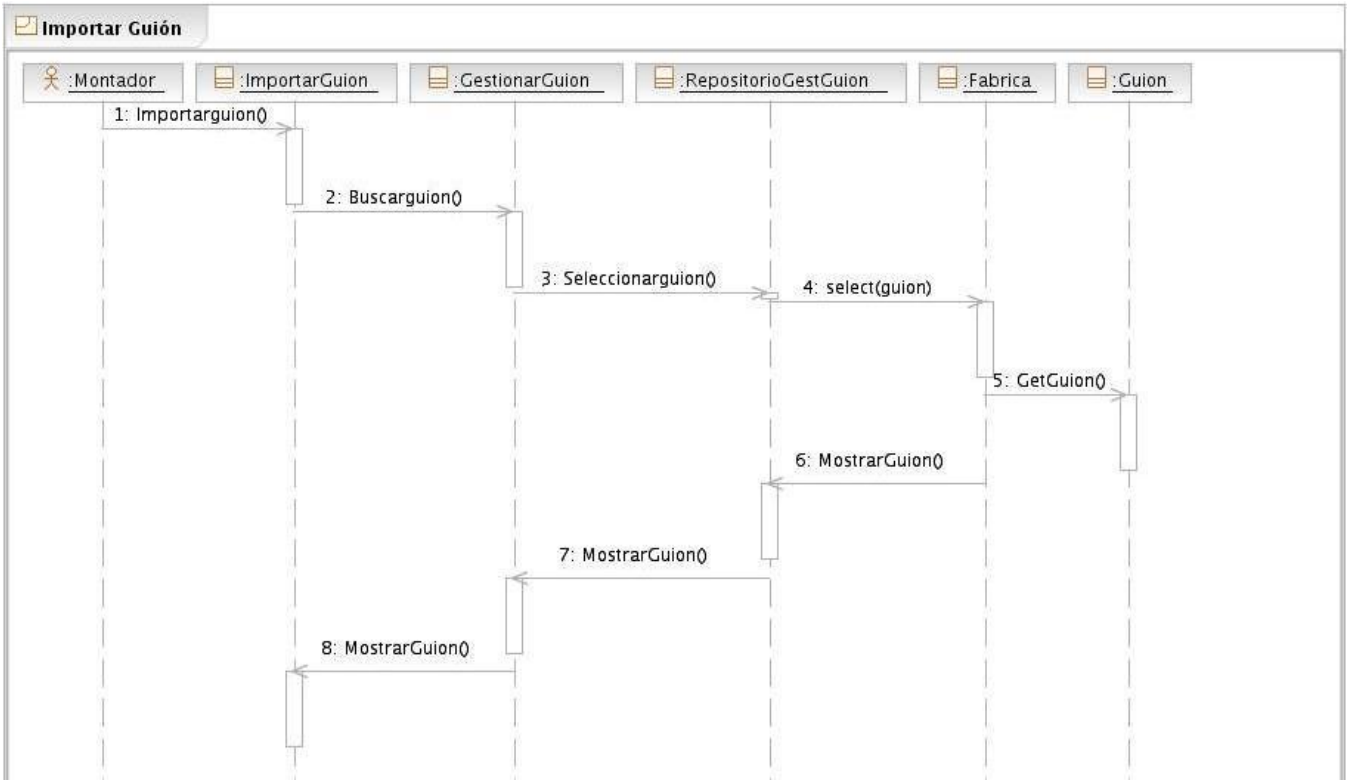


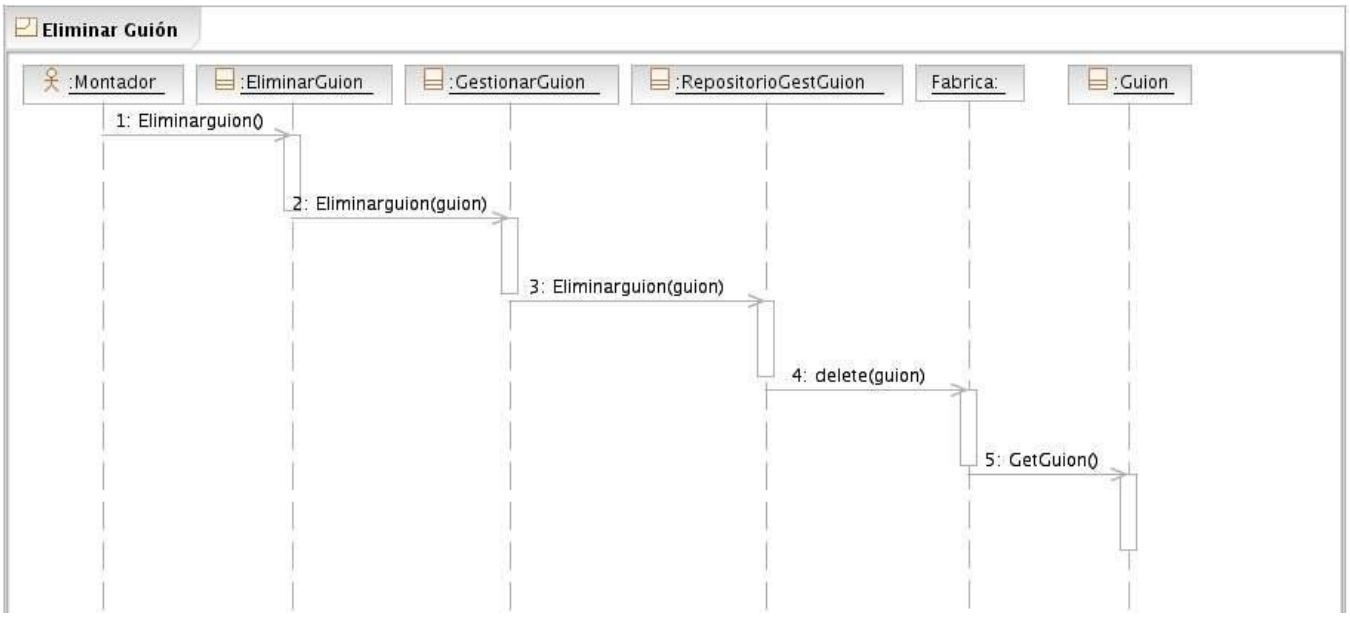
CU Gestionar componente.



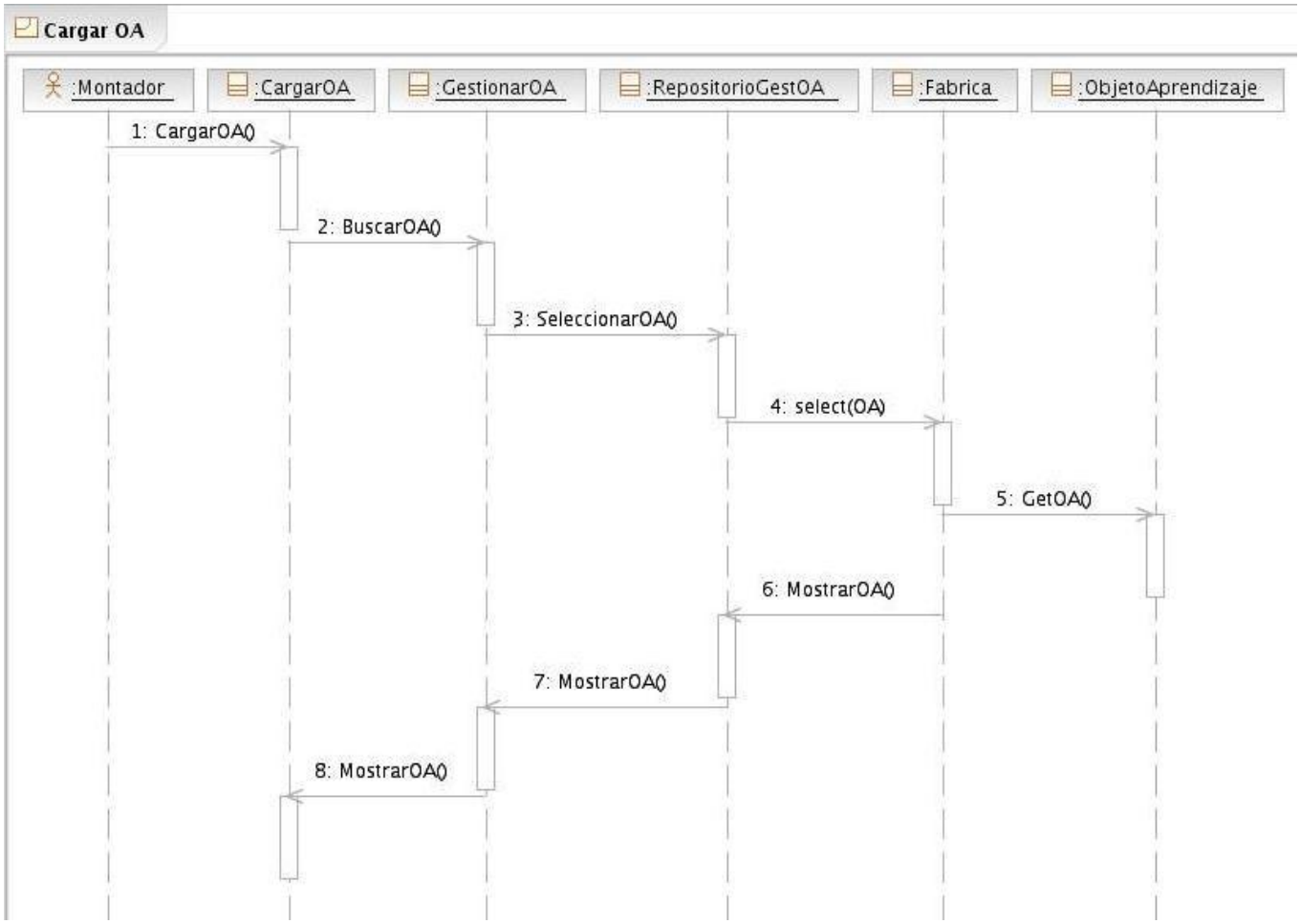
Con la elaboración del diseño, se definen las clases fundamentales que permitirán que el sistema funcione satisfactoriamente, así como los atributos y métodos que deben tener las mismas, para brindarle al desarrollador una idea clara de lo que debe implementar. El uso de patrones de diseño contribuyó a reutilizar la experiencia de los desarrolladores, ya que clasifican y describen formas de solucionar problemas, que ocurren de forma frecuente en el desarrollo de una aplicación. La realización de cada caso de uso es un artefacto muy importante que da entrada al flujo de trabajo que le sigue al diseño en la metodología RUP. La disciplina que continúa, es la de implementación cuyo principal objetivo es crear componentes más específicos y asignarles a estos las clases del diseño correspondientes, en dependencia de la trazabilidad definida. Dentro de las realizaciones de los casos de uso se elaboraron los diagramas de interacción. Estos explican la colaboración que existe entre las clases y cómo son llamados los métodos y sentencias dentro de cada una. Con la obtención de estos artefactos se arriba a los resultados esperados en el diseño del subsistema Herramienta Autor de Objetos de Aprendizaje para el sistema Emedia.

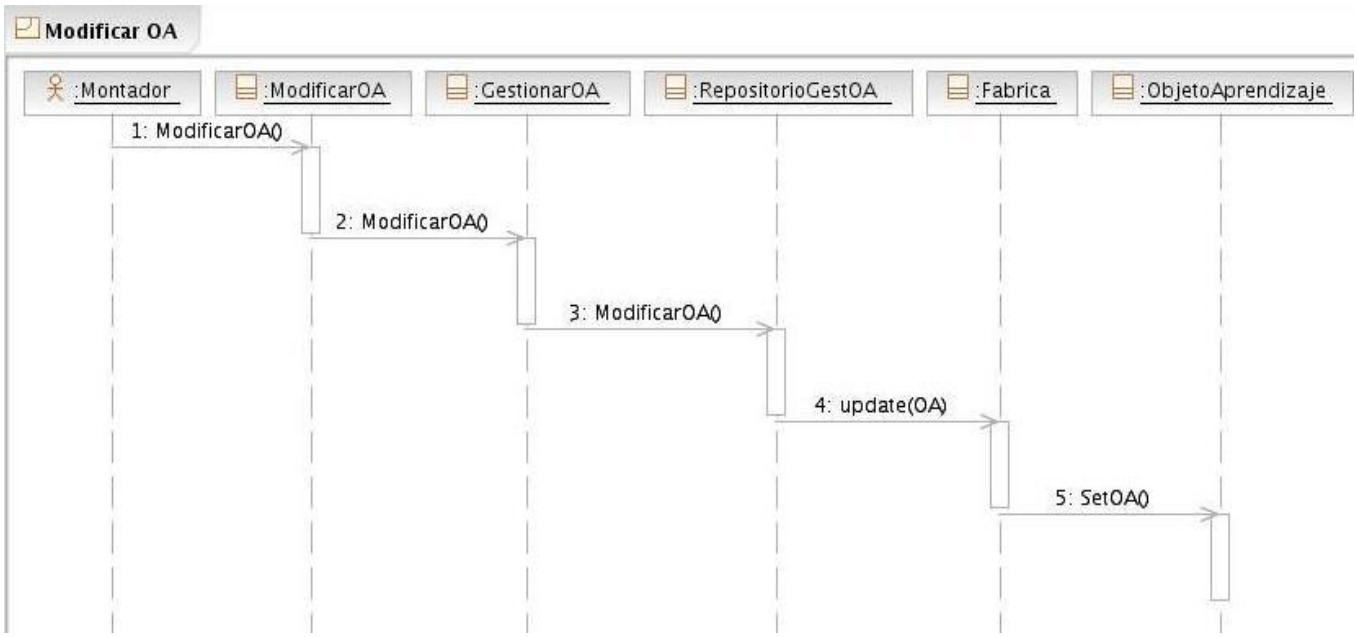
CU Gestionar Guión



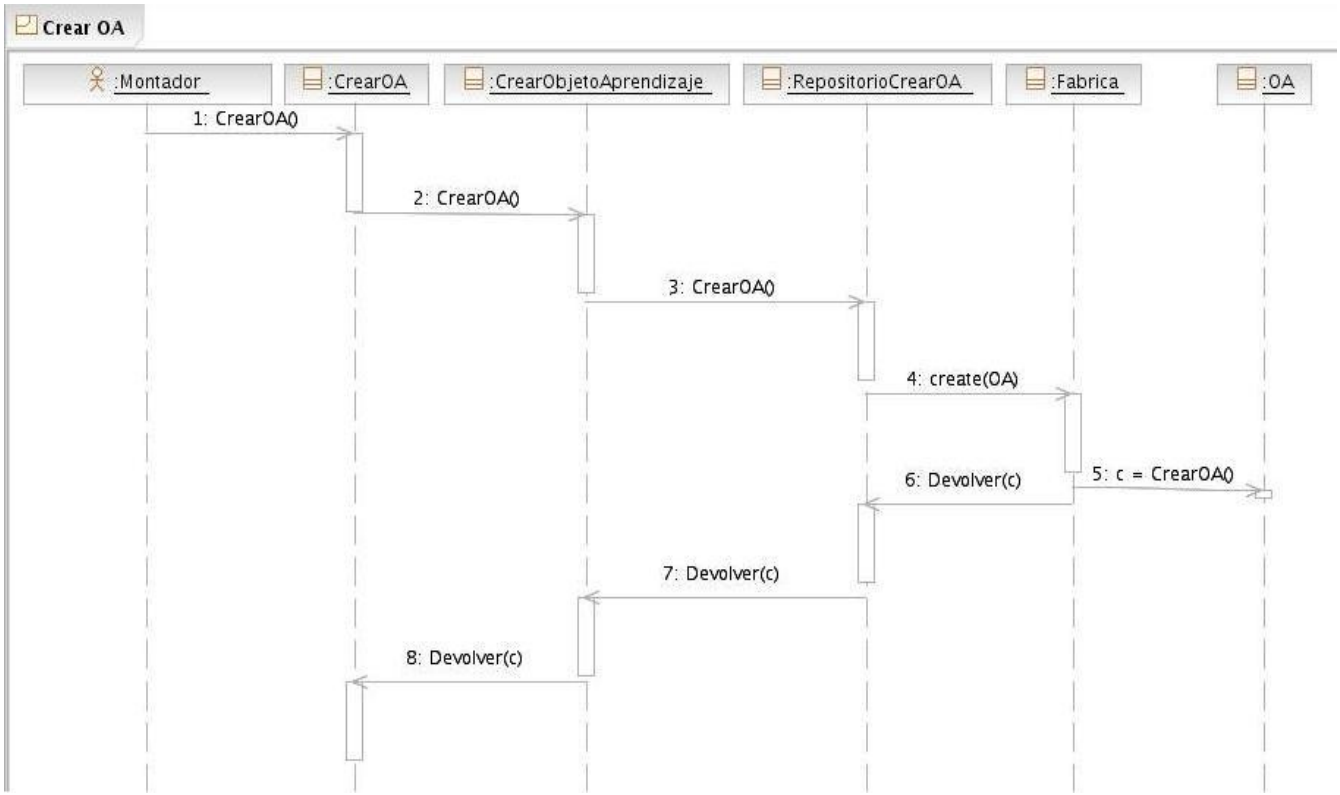

Eliminar Guión
:Montador
:EliminarGuion
:GestionarGuion
:RepositorioGestGuion
Fabrica:
:Guion
1: Eliminarguion()
2: Eliminarguion(guion)
3: Eliminarguion(guion)
4: delete(guion)
5: GetGuion()

CU Gestionar OA.

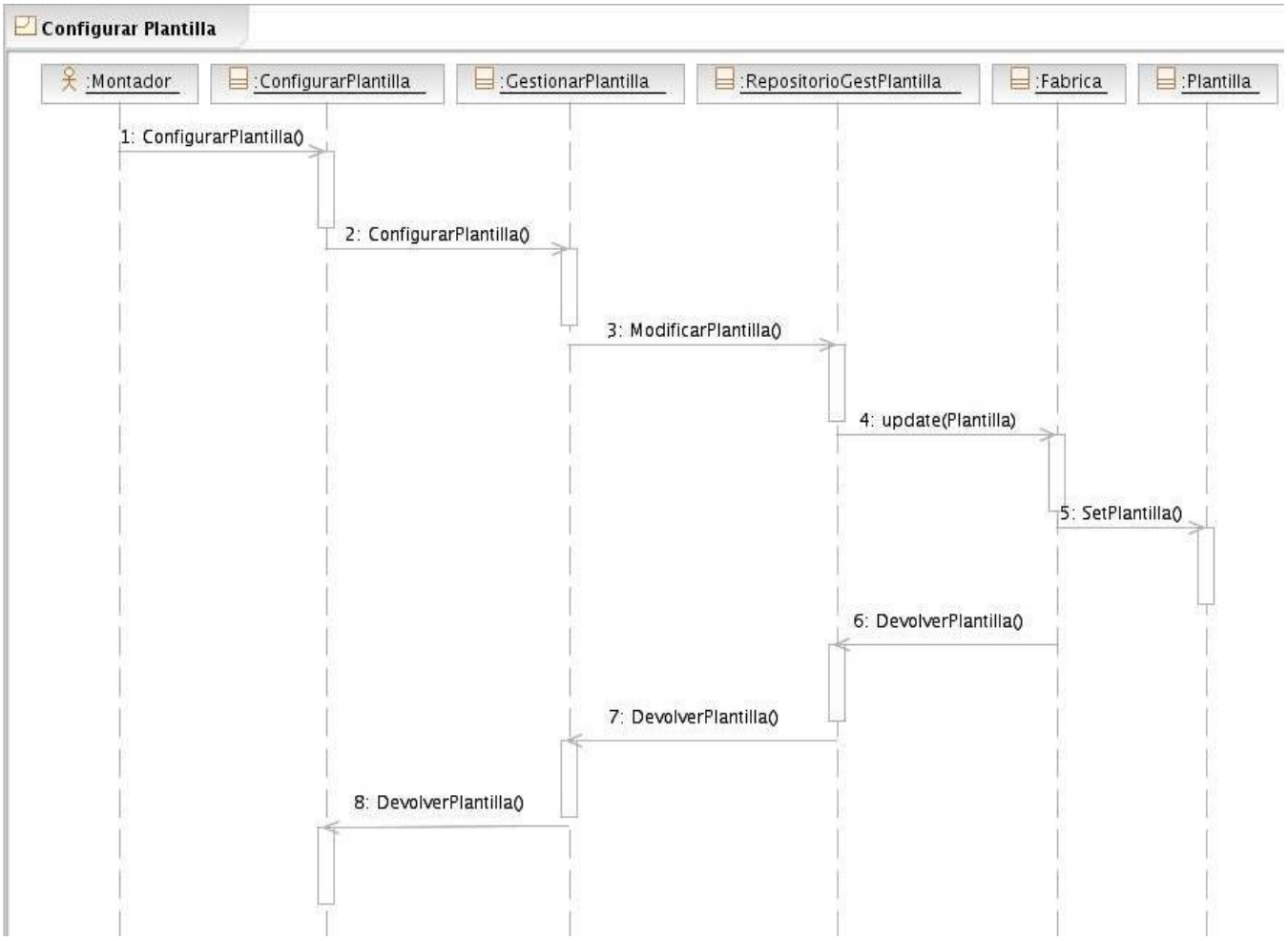



Modificar OA
:Montador
:ModificarOA
:GestionarOA
:RepositorioGestOA
:Fabrica
:ObjetoAprendizaje
1: ModificarOA()
2: ModificarOA()
3: ModificarOA()
4: update(OA)
5: SetOA()

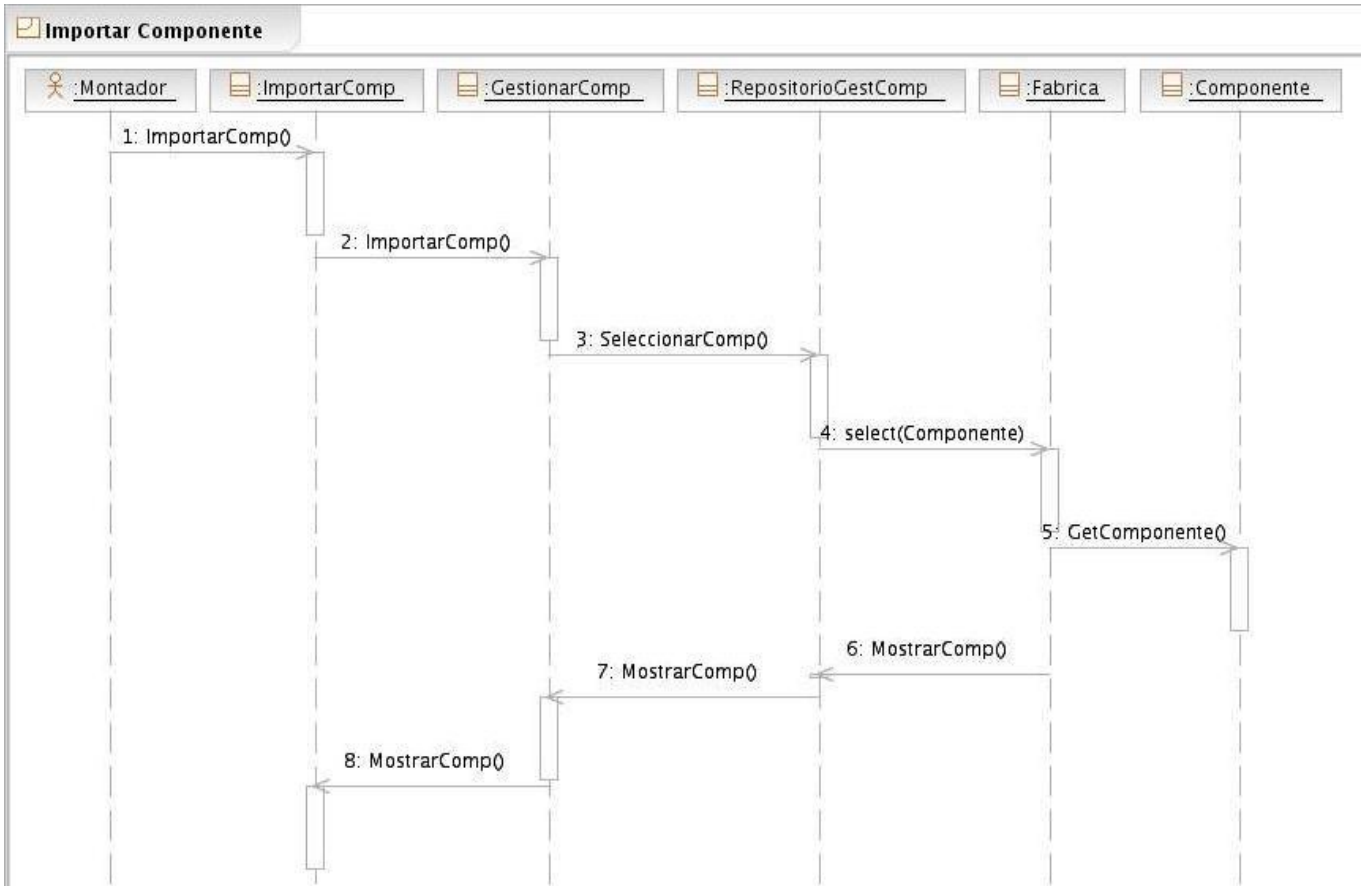
CU Crear OA.

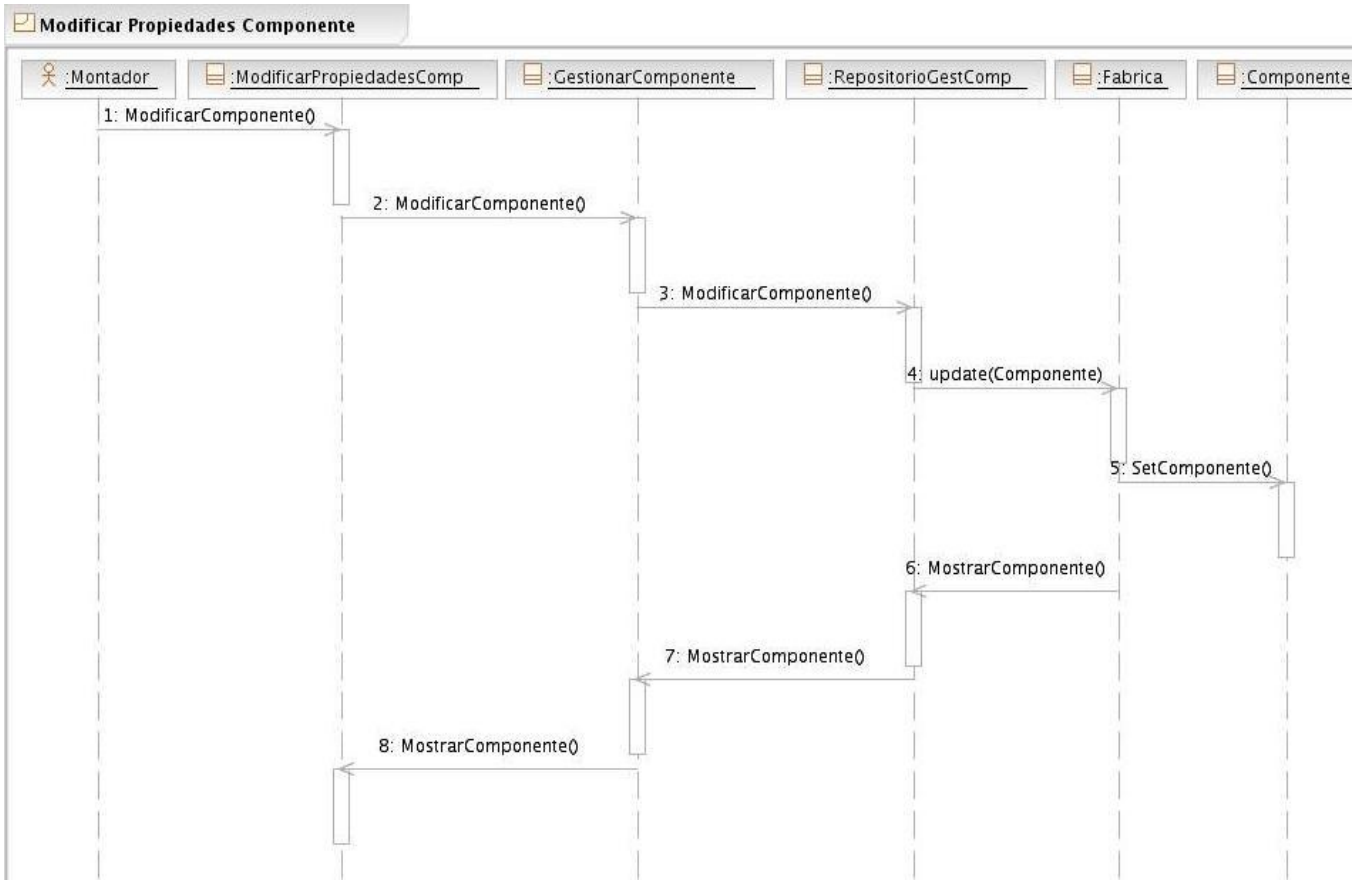

Crear OA
:Montador
:CrearOA
:CrearObjetoAprendizaje
:RepositorioCrearOA
:Fabrica
:OA
1: CrearOA()
2: CrearOA()
3: CrearOA()
4: create(OA)
5: c = CrearOA()
6: Devolver(c)
7: Devolver(c)
8: Devolver(c)

CU Gestionar plantilla.



CU Gestionar componente.





3.5 Descripción de las clases.

Nombre: Crear OA.	
Tipo de clase: Interfaz.	
Atributo	**Tipo**
LineEdit	lineEdit
LineEdit2	lineEdit
DateEdit	dateEdit
LineEdit5	lineEdit
ListView	listView
Aceptar	pushButton

Cancelar	pushButton2
Autor	label
Título	label2
Fecha_Creación	label3
Dirigido	label5
Objetivo	label4
Por cada responsabilidad:	
Nombre:	public void pushButtonAceptarClick
Descripción:	Función que define la programación de un evento de un botón.
Nombre:	public void pushButton2CancelarClick
Descripción:	Función que define la programación de un evento de un botón.

Nombre: Importar Componente.	
Tipo de clase: Interfaz.	
Atributo	**Tipo**
LineEdit	lineEdit
ComboBox	comboBox
Aceptar	pushButton
Cancelar	pushButton2
NombreArchivo	label2
TipoArchivo	label3
TreeWidget	treeWidget
Por cada responsabilidad:	
Nombre:	public pushButtonAceptarClick

Descripción:	Función que define la programación de un evento de un botón.
Nombre:	public pushButton2CancelarClick
Descripción:	Función que define la programación de un evento de un botón.

Nombre: Adicionar Componente.	
Tipo de clase: Interfaz.	
Atributo:	**Tipo:**
–	–
Por cada responsabilidad:	
Nombre:	public virtual void AdicionarComponente =0
Descripción:	Función abstracta que define la adición de un componente.

Nombre: Modificar Componente.	
Tipo de clase: Interfaz.	
Atributo:	**Tipo:**
Nombre	label
Aceptar	pushButton
Cancelar	pushButton2
LineEdit	lineEdit
Por cada responsabilidad:	
Nombre:	public pushButtonAceptarClick
Descripción:	Función que define la programación de un evento de un botón.
Nombre:	public pushButton2CancelarClick
Descripción:	Función que define la programación de un evento de un botón.

Nombre: Eliminar Componente	
Tipo de clase: Interfaz.	
Atributo:	**Tipo:**
Advertencia	label
Aceptar	pushButton
Cancelar	pushButton2
Por cada responsabilidad:	
Nombre:	public pushButtonAceptarClick
Descripción:	Función que define la programación de un evento de un botón.
Nombre:	public pushButton2CancelarClick
Descripción:	Función que define la programación de un evento de un botón.

Nombre: Importar Guión.	
Tipo de clase: Interfaz	
Atributo:	**Tipo:**
LineEdit	lineEdit
Directorio	comboBox
TreeWidget	treeWidget
Aceptar	pushButton
Cancelar	pushButton2
NombreArchivo	label
Por cada responsabilidad:	
Nombre:	public pushButtonAceptarClick
Descripción:	Función que define la programación de un evento de un botón.

Nombre:	public pushButton2CancelarClick
Descripción:	Función que define la programación de un evento de un botón.

Nombre: Eliminar Guión.	
Tipo de clase: Interfaz.	
Atributo	**Tipo**
Advertencia	label
Aceptar	pushButton
Cancelar	pushButton2
Por cada responsabilidad:	
Nombre:	public pushButtonAceptarClick
Descripción:	Función que define la programación de un evento de un botón.
Nombre:	public pushButton2CancelarClick
Descripción:	Función que define la programación de un evento de un botón.

Nombre: Modificar OA.	
Tipo de clase: Interfaz.	
Atributo	**Tipo**
LineEdit	lineEdit
LineEdit2	lineEdit
DateEdit	dateEdit
LineEdit5	lineEdit
ListView	listView
Aceptar	pushButton

Cancelar	pushButton2
Autor	label
Título	label2
Fecha_Creación	label3
Dirigido	label5
Objetivo	label4
Por cada responsabilidad:	
Nombre:	public pushButtonAceptarClick
Descripción:	Función que define la programación de un evento de un botón.
Nombre:	public pushButton2CancelarClick
Descripción:	Función que define la programación de un evento de un botón.

Nombre: Cargar OA.	
Tipo de clase: Interfaz.	
Atributo	**tipo**
LineEdit	lineEdit
ComboBox	comboBox
TreeWidget	treeWidget
Aceptar	pushButton
Cancelar	pushButton2
NombreArchivo	label
Directorio	label2
Por cada responsabilidad:	
Nombre:	public pushButtonAceptarClick

Descripción:	Función que define la programación de un evento de un botón.
Nombre:	public pushButton2CancelarClick
Descripción:	Función que define la programación de un evento de un botón.

Nombre: Configurar Plantilla.	
Tipo de clase: Interfaz.	
Atributo	**Tipo**
Tipo	label
Título	label2
Plantilla	label6
Interacciones	label7
Puntuación	label3
Tiempo	label4
Intentos	label5
Guardar	pushButton2
Cancelar	pushButton3
ComboBox	comboBox
LineEdit	lineEdit
GroupBox	groupBox
GroupBox2	groupBox
Configurar	pushButton
Por cada responsabilidad:	
Nombre:	public pushButtonGuardarClick
Descripción:	Función que define la programación de un evento de un botón.

Nombre:	public pushButton2CancelarClick
Descripción:	Función que define la programación de un evento de un botón.

Nombre: Eliminar Plantilla.	
Tipo de clase: Interfaz.	
Atributo	**Tipo**
Advertencia	label
Aceptar	pushButton
Cancelar	pushButton2
Por cada responsabilidad:	
Nombre:	public pushButtonAceptarClick
Descripción:	Función que define la programación de un evento de un botón.
Nombre:	public pushButton2CancelarClick
Descripción:	Función que define la programación de un evento de un botón.

Las demás descripciones se pueden encontrar en el **Anexo1.**

3.6 Prototipo no funcional de Interfaz de Usuario.

Los prototipos de interfaz de usuario nos ayudan a comprender y especificar las interacciones entre los actores humanos y sistema durante la captura de requisitos. No sólo nos ayuda a desarrollar una buena interfaz gráfica, sino también a comprender mejor los casos de usos. A la hora de especificar la interfaz gráfica de usuario también pueden utilizarse otros artefactos, como los modelos de interfaz gráfica y los esquemas de pantalla. (11) El prototipo no funcional de la herramienta de Autor de Objetos de Aprendizaje se puede encontrar en el **Anexo 2**.

CONCLUSIONES

- Con el estudio realizado de las herramientas de Autor se logró comprender conceptos relacionados con el ambiente donde se encuentra enmarcada la investigación posibilitando la elaboración de un modelo de dominio que permitió establecer relaciones entre dichos conceptos.
- Para realizar el diseño sobre plataformas de Software Libre existen un conjunto de tecnologías que de aplicarse correctamente perfeccionan la comprensión y desarrollo de flujos de trabajo dentro del ciclo de vida de RUP.
- Se determinaron las principales funcionalidades del sistema a diseñar y se obtuvo un prototipo no funcional de la aplicación, lo que facilita la comprensión de los casos de uso descritos.

RECOMENDACIONES

A continuación se muestran algunas recomendaciones a tener en cuenta para darle seguimiento al trabajo realizado:

- Implementar el sistema diseñado utilizando el patrón propuesto.
- Utilizar el estándar SCORM en la futura implementación de la herramienta garantizando que las propiedades generales de los objetos de aprendizaje se vean reflejados en el mismo.
- Integrar la herramienta al sistema Emedia.

REFERENCIA BIBLIOGRÁFICA

1. **Rodríguez, José Palos.** www.oei.es. [En línea] asenmac , 2008. http://www.oei.es/valores2/palos2.htm.
2. **García, Lic. Dialina Emelina Luis.** www.monografias.com. [En línea] 1997. http://www.monografias.com/trabajos47/profesor-sociales/profesor-sociales.shtml.
3. **Anónimo.** (s.f.). Recuperado el 27 de 11 de 2007, de http://64.233.169.104/search?q=cache:yhu4hDsWusIJ:tecnologias.gio.etsit.upm.es/elearning/lms--learning-management-system--sistema-de-gestion-de-aprendizaje--
4. **Learning Technology Standards Committee.** ieee.org. [Online] 2005. http://ltsc.ieee.org/wg12/.
5. **Lorente Rodríguez, Abel Ernesto.** *Plataformas para el desarrollo y gestión de Cursos Educativos Multimedia.* [CD-ROM] La Habana : Memorias Uciencia, Universidad de las Ciencias Informáticas (UCI), 2006.
6. **MSc. José L. Montero O'Farrill, Dra. Elsa Herrero Tunis.** www.cujae.edu.cu. [Online] 2006. http://www.cujae.edu.cu/eventos/cittel/trabajos/Trabajos/Comision%205/CITTEL-72.pdf.
7. **Bernal, Rosa.** materialesvirtuales.blogspot.com. [Online] 8 2007. http://materialesvirtuales.blogspot.com/2007/08/atenex_9684.html.iografía
8. **Hernández, Pepe.** murciaobjetosdeaprendizaje.blogspot.com. [Online] 1 2, 2006. http://murciaobjetosdeaprendizaje.blogspot.com/2006/01/autore-10.html.
9. **Wikimedia Foundation, Inc.** es.wikipedia.org. [Online] 6 5, 2008. [Cited: 6 13, 2008.] http://es.wikipedia.org/wiki/Dokeos.
10. **Guzmán, Clara López.** www.biblioweb.dgsca.unam.mx. [Online] 2005. http://www.biblioweb.dgsca.unam.mx/libros/repositorios/objetos_aprendizaje.htm.
11.**RUMBAUGH, J.; I. JACOBSON**, et al. El proceso unificado de desarrollo de software. La Habana, Addison Wesley Longman, 2000. p.
12. **Barzanallana, Rafael.** www.um.es. [Online] 12 30, 2006. http://www.um.es/docencia/barzana/IAGP/Iagp2.html#BM1.
13. **ANACHE, I. and M. JOEL**. *OMG UML 2.0 -Marcando un hito en el desarrollo de software.* Habana, 2005. p.
14. **Agüera, Ivan Obeso.** www.euitio.uniovi.es. [Online] http://petra.euitio.uniovi.es/~i2133798/hd/archivos/disenio/estudioHerramientasDisenio.pdf.
15. **IBM Corporation.** ftp.software.ibm.com. [Online] 2006. ftp://ftp.software.ibm.com/software/rational/rational_es/Rational_Architect_07_esp.pdf.
16. MINISTERIO DE EDUCACION ARGENTINA **.** www.somoslibres.org. [Online] 2008. http://www.somoslibres.org/modules.php?name=News&file=article&sid=1910.
17. **Taffernaberry Carlos**. *www.dirinfo.unsl.edu.ar.* [Online] 2008 .http://www.dirinfo.unsl.edu.ar/~sortecarpro/practicos/presentaciones/DEBIAN.pdf.
18. **Wikimedia Foundation, Inc.** es.wikipedia.org. [Online] 4 27, 2008. [Cited: 5 8, 2008.] http://es.wikipedia.org/wiki/Gtk.
19. **Wikimedia Foundation. 2007.** wikipedia.org. [En línea] 2007. http://es.wikipedia.org/wiki/Qt_%28biblioteca%29.
20. plataformaeclipse.com. [Online] http://plataformaeclipse.com/.
21. **Fundation Member.** [Online] 2008. http://www.easyeclipse.org/site/home/.
22. **BASS L, P Clements y R.Kazman.1998**. "Software Architecture in Practice", Addison-Wesley.
23. **Ricardo, F.Á.C.** Consultado 30/5/2008. Disponible en: http://www.monografías.com/trabajos43/patron-modelo-vista.

24. Patrón de Diseño. 2007. Disponible en:
http://www.ingenierossoftware.com/analisisydiseno/patrones-diseno.php
25. Larman, C. UML y Patrones. Vol. Tomo I.

ANEXOS

ANEXO 1: Descripciones de las clases.

Nombre: CrearObjetoAprendizaje	
Tipo de clase: Controladora.	
Atributo	**Tipo**
-	**-**
Por cada responsabilidad:	
Nombre:	CrearOA.
Descripción:	Función que define la creación de un OA.

Nombre: GestionarComponente.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
-	**-**
Por cada responsabilidad:	
Nombre:	ImportarComponente
Descripción:	Función que define como se importa un componente.
Nombre:	AdicionarComponente
Descripción:	Función que define como se adiciona un componente.
Nombre:	**ModificarComponente**
Descripción:	Función que define como se modifica un componente.
Nombre:	**EliminarComponente**
Descripción:	Función que define como se elimina un componente.

Nombre: GestionarGuión.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
-	**-**
Por cada responsabilidad:	
Nombre:	ImportarGuión.
Descripción:	Función que define como se importa un guión.
Nombre:	EliminarGuión.
Descripción:	Función que define como se elimina un guión.

Nombre: GestionarOA.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
-	**-**
Por cada responsabilidad:	
Nombre:	CargarOA
Descripción:	Función que define como se carga un OA.
Nombre:	ModificarOA.
Descripción:	Funcion que define como se modifica un OA.

Nombre: GestionarPlantilla.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
-	**-**

Por cada responsabilidad:	
Nombre:	CrearPlantilla.
Descripción:	Función que define como se crea una plantilla.
Nombre:	GuardarPlantilla.
Descripción:	Función que define como se guarda una plantilla.
Nombre:	ModificarPlantilla.
Descripción:	Función que define como se modifica una plantilla.
Nombre:	EliminaPlantilla.
Descripción:	Función que define como se elimina una plantilla.

Nombre: RepositorioCrearOA.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
-	**-**
Por cada responsabilidad:	
Nombre:	AdicionarOA.
Descripción:	Función que permite adicionar un nuevo OA.

Nombre: RepositorioGestionarComponente.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
Por cada responsabilidad:	
Nombre:	AdicionarComponente.

Descripción:	Función que permite adicionar un nuevo Componente.
Nombre:	SeleccionarComponente.
Descripción:	Función que permite definir la selección de Componente.
Nombre:	ModificarComponente.
Descripción:	Función que permite modificar los datos de un Componente.
Nombre:	EliminarComponente.
Descripción:	Función que permite eliminar un Componente.

Nombre: RepositorioGestionarGuión.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
-	**-**
Por cada responsabilidad:	
Nombre:	AdicionarGuión.
Descripción:	Función que permite adicionar un nuevo Guión.
Nombre:	SeleccionarGuión.
Descripción:	Función que permite definir la selección de Guión.
Nombre:	EliminarGuión.
Descripción:	Función que permite eliminar un Guión.

Nombre: RepositorioGestionarOA.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
-	**-**

Por cada responsabilidad:	
Nombre:	AdicionarOA.
Descripción:	Función que permite adicionar un nuevo OA.
Nombre:	SeleccionarOA.
Descripción:	Función que permite definir la selección de OA.
Nombre:	ModificarOA.
Descripción:	Función que permite modificar los datos de un OA.
Nombre:	EliminarOA.
Descripción:	Función que permite eliminar un OA.

Nombre: RepositorioGestionarPlantilla.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
-	**-**
Por cada responsabilidad:	
Nombre:	AdicionarPlantilla.
Descripción:	Función que permite adicionar un nuevo Plantilla.
Nombre:	Seleccionar Plantilla.
Descripción:	Función que permite definir la selección de Plantilla.
Nombre:	Modificar Plantilla.
Descripción:	Función que permite modificar los datos de un Plantilla.
Nombre:	Eliminar Plantilla.
Descripción:	Función que permite eliminar un Plantilla.

Nombre: ObjetoAprendizaje.	
Tipo de clase: Entidad.	
Atributo	**Tipo**
Autor	String
Título	String
Fecha_Creación	Integer
Dirigido	String
Objetivo	String
Por cada responsabilidad:	
Nombre:	SetOA.
Descripción:	Método que se encarga de modificar los valores de cada atributo de la clase.
Nombre:	GetOA.
Descripción:	Devuelve los valores que contienen los atributos de la clase.

Nombre: Componente.	
Tipo de clase: Entidad.	
Atributo	**Tipo**
Nombre	String
Por cada responsabilidad:	
Nombre:	SetComponente.
Descripción:	Método que se encarga de modificar los valores de cada atributo de la clase.
Nombre:	GetComponente.

Descripción:	Devuelve los valores que contienen los atributos de la clase.

Nombre: ComponenteExterno.	
Tipo de clase: Entidad.	
Atributo	**Tipo**
Extensión	String
Por cada responsabilidad:	
Nombre:	SetComponente.
Descripción:	Método que se encarga de modificar los valores de cada atributo de la clase.
Nombre:	GetComponente.
Descripción:	Devuelve los valores que contienen los atributos de la clase.

Nombre: Guión.	
Tipo de clase: Entidad.	
Atributo	**Tipo**
Título	String
Autor	String
Fecha_Creación	Integer
Versión	Integer
Por cada responsabilidad:	
Nombre:	SetGuión

Descripción:	Método que se encarga de modificar los valores de cada atributo de la clase.
Nombre:	GetGuión.
Descripción:	Devuelve los valores que contienen los atributos de la clase.

Nombre: Plantilla.	
Tipo de clase: Entidad.	
Atributo	**Tipo**
Tipo	String
Título	String
Puntuación	Integer
Tiempo	Integer
Intentos	Integer
Por cada responsabilidad:	
Nombre:	SetPlantilla.
Descripción:	Método que se encarga de modificar los valores de cada atributo de la clase.
Nombre:	GetPlantilla.
Descripción:	Devuelve los valores que contienen los atributos de la clase.

ANEXO 2: Prototipo no funcional.

CU Gestionar Guión de Contenido.

Figura 1: Vista de la ventana de importar guión.

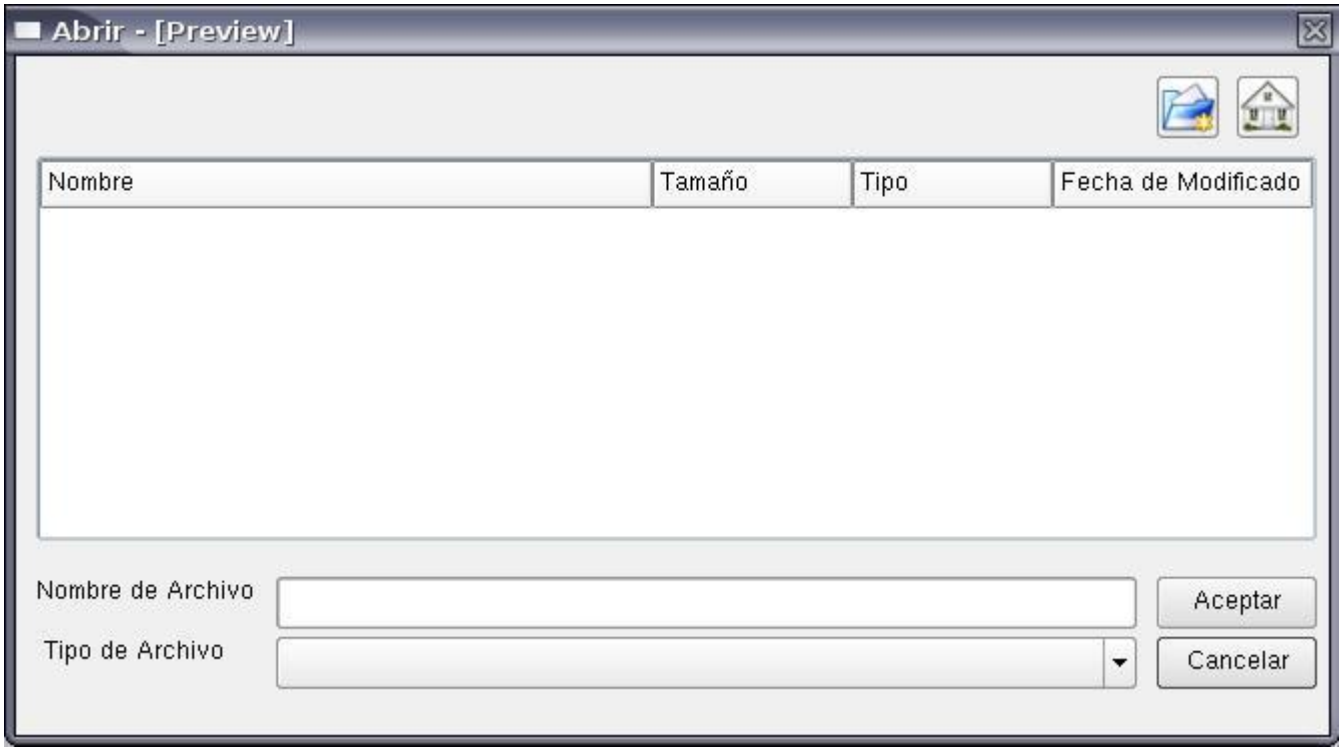


Figura 2: Vista del guión importado.

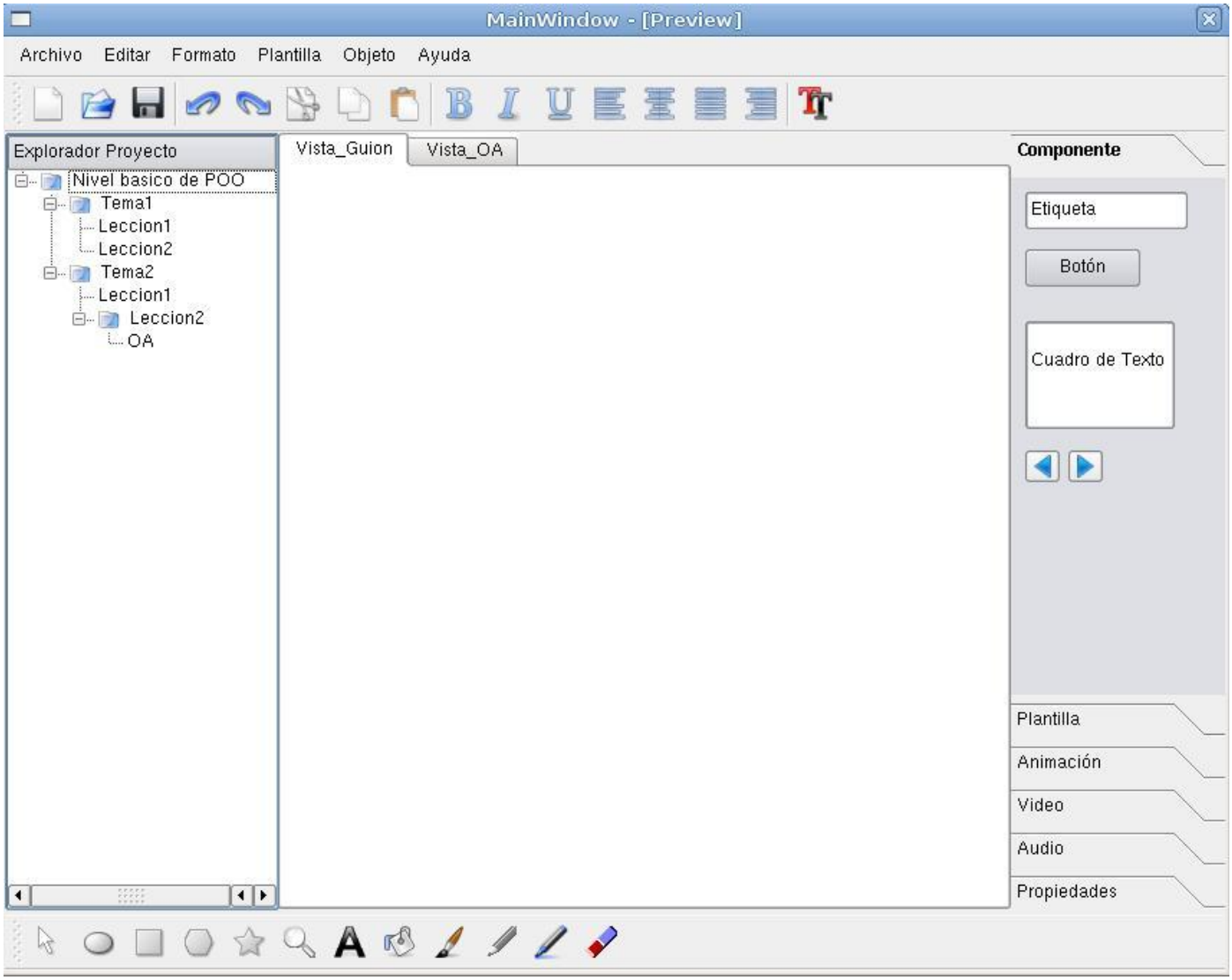
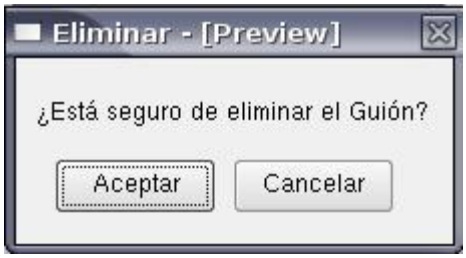


Figura 2: Vista de la interfaz de confirmación para eliminar un guión.

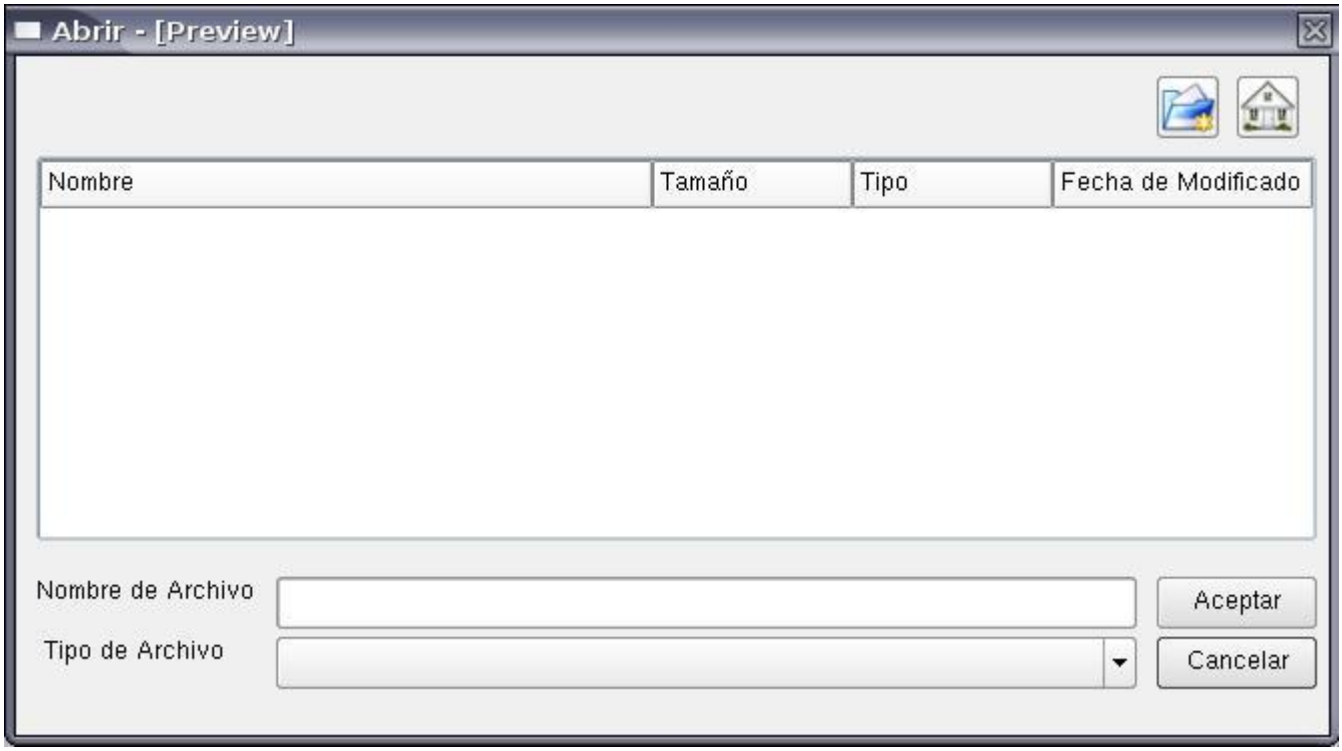


CU Crear OA.

Figura 1: Vista para crear un OA.

CU GESTIONAR OA.

Figura 1: Vista para Cargar un OA.

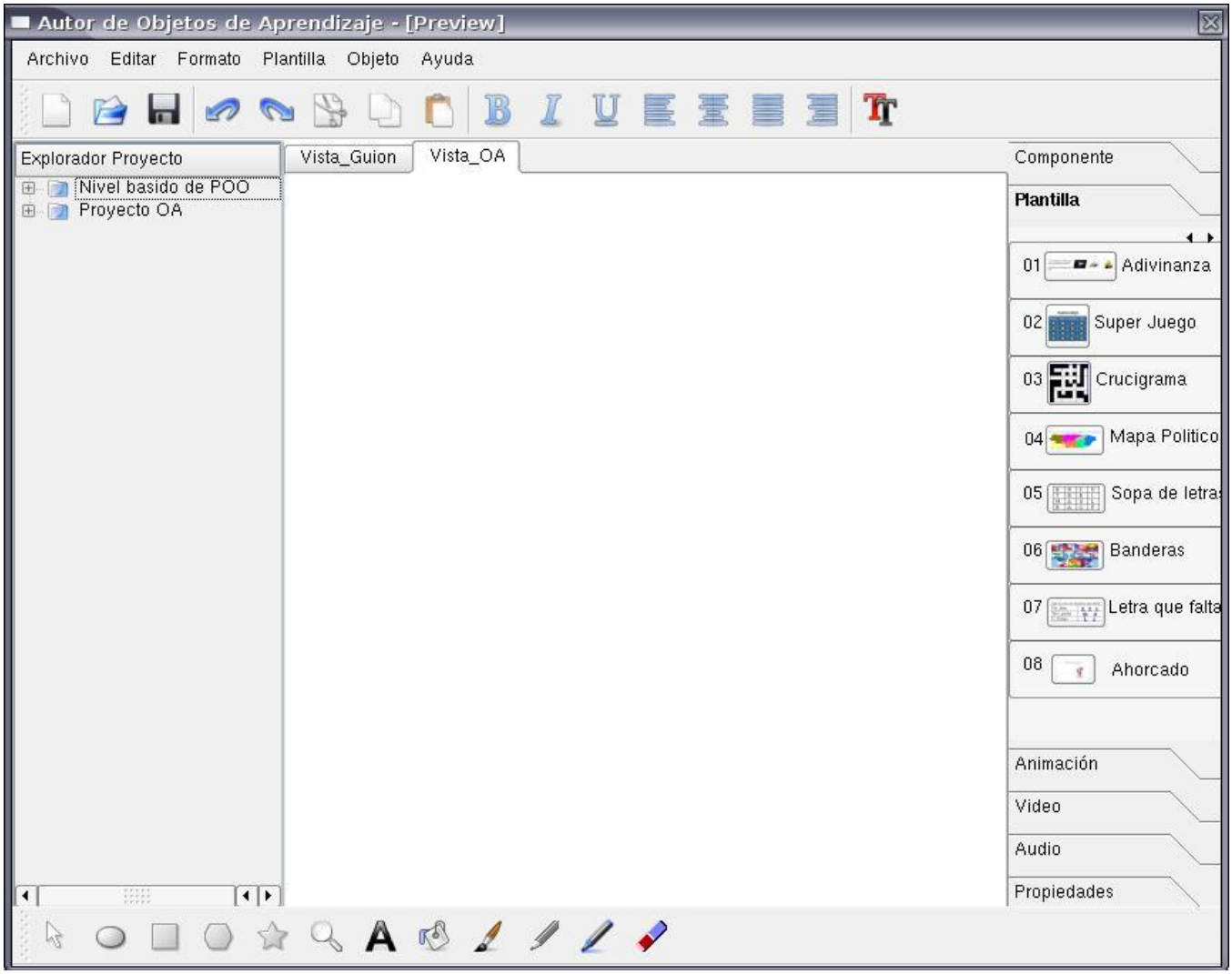


CU Gestionar proyecto.

Figura 1: Vista del modelo a llenar para crear un proyecto.

CU Gestionar Plantilla.

Figura 1: Vista principal para acceder a las plantillas.

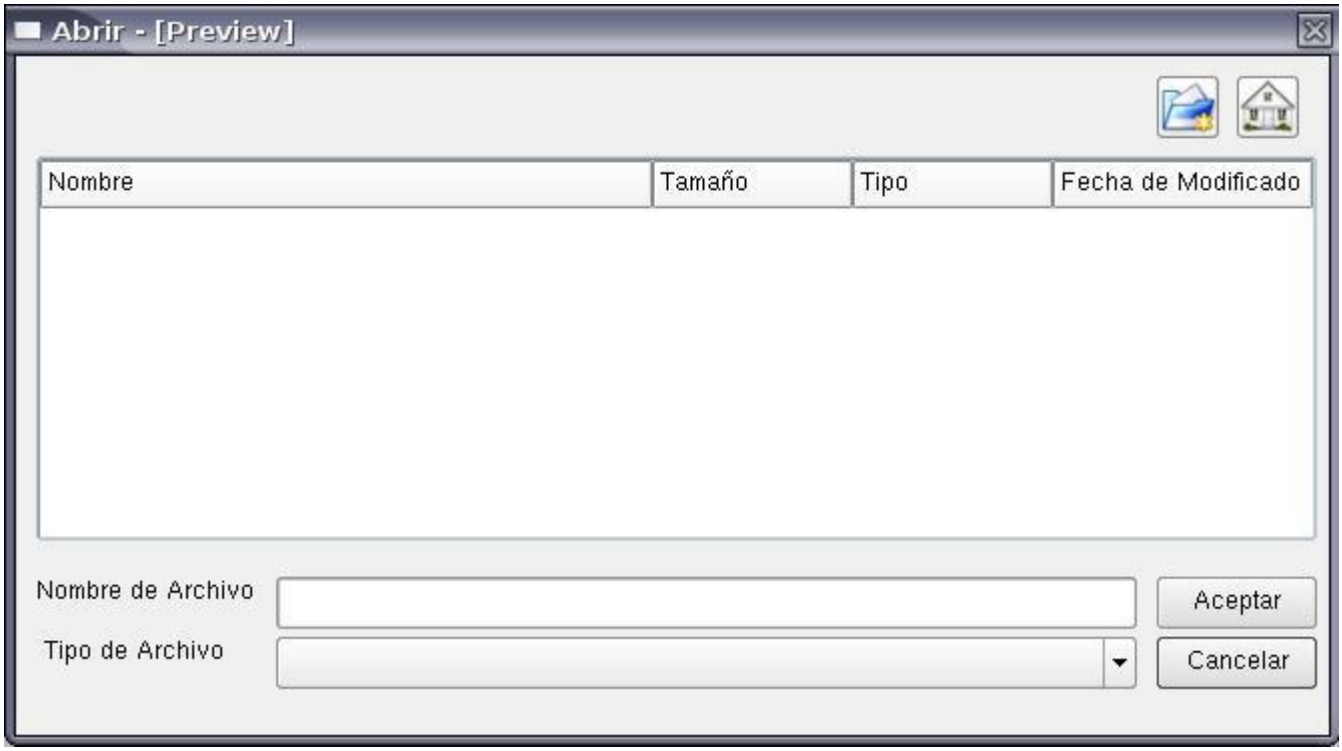


CU Gestionar Plantilla.

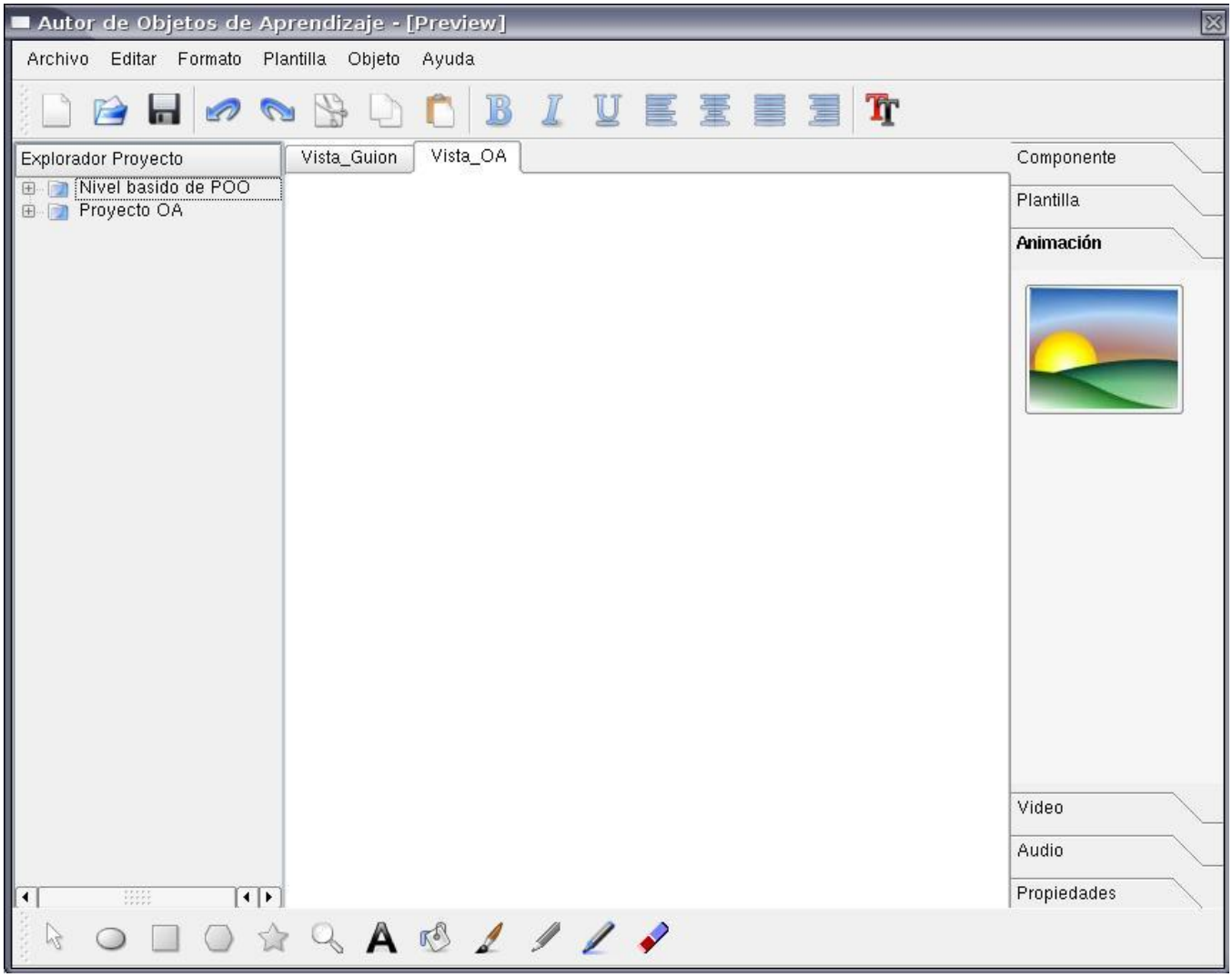
Figura 2: Vista para configurar la plantilla.

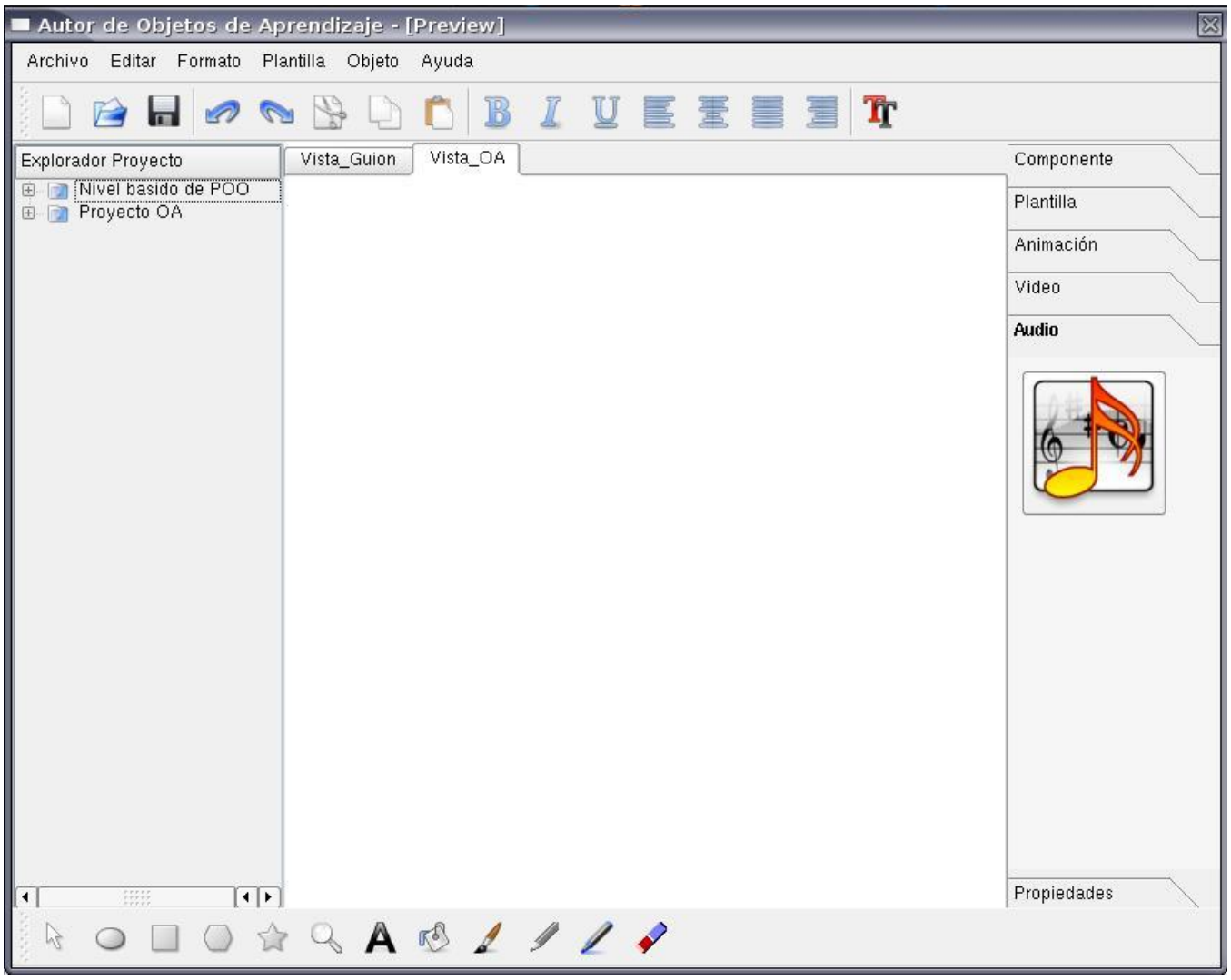
CU Gestionar Componente.

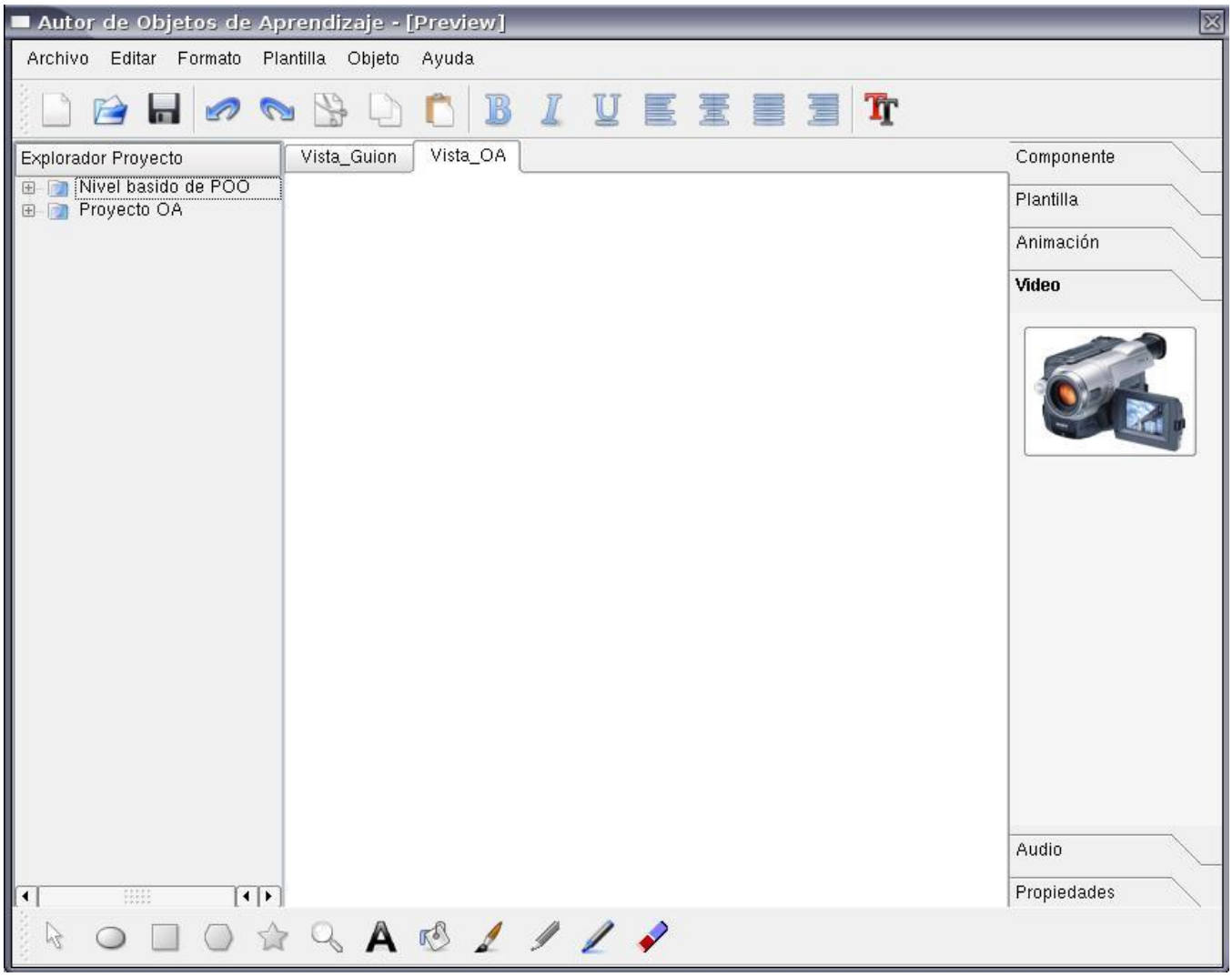
Figura 1: Vista para importar un componente a la aplicación.

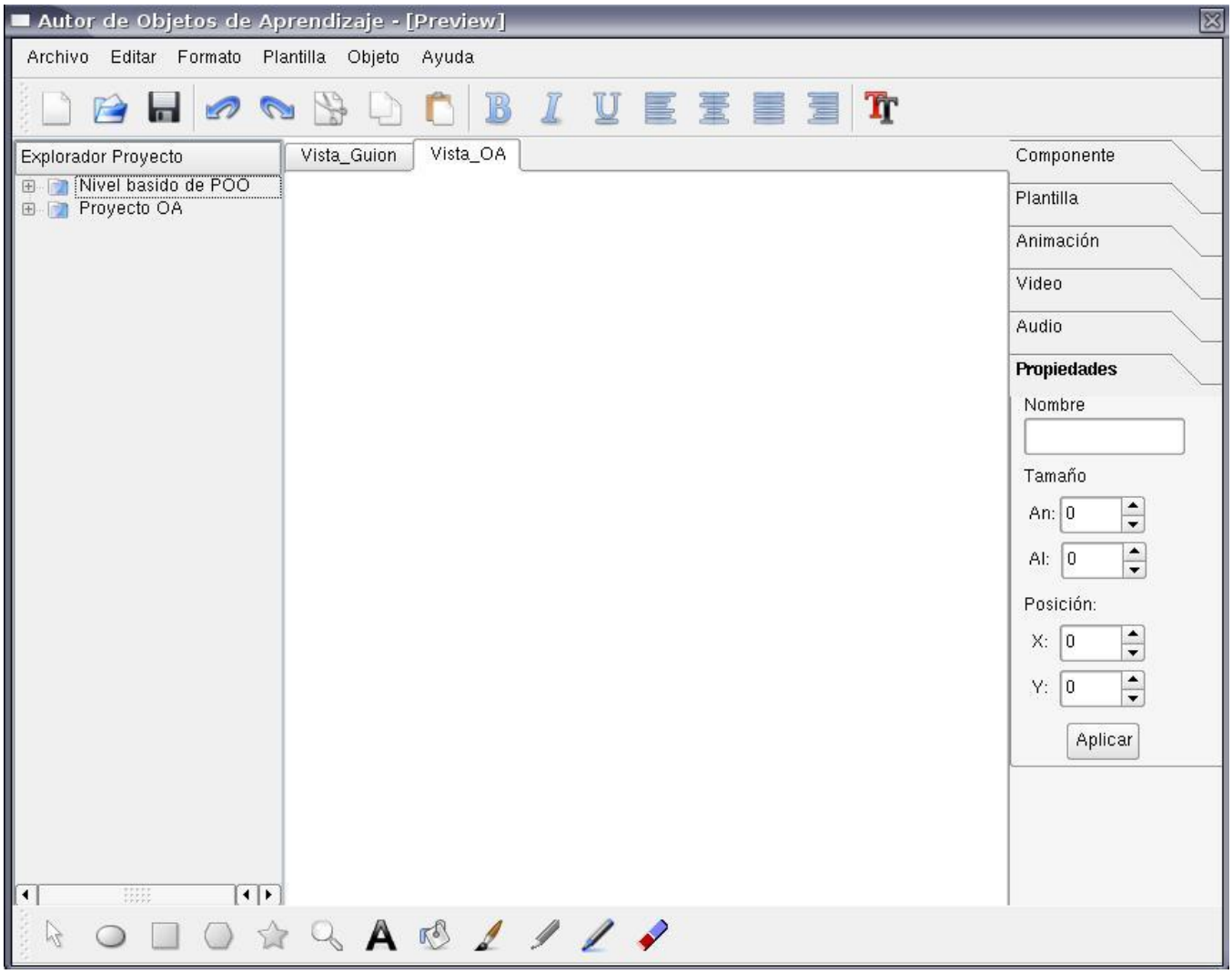


Vistas Generales de la Aplicación.




Autor de Objetos de Aprendizaje - [Preview]
Archivo
Editar
Formato
Plantilla
Objeto
Ayuda
Explorador Proyecto
Nivel basido de POO
Proyecto OA
Vista_Guion
Vista_OA
Componente
Plantilla
Animación
Video
Audio
Propiedades


Autor de Objetos de Aprendizaje - [Preview]
Archivo
Editar
Formato
Plantilla
Objeto
Ayuda
Explorador Proyecto
Nivel basido de POO
Proyecto OA
Vista_Guion
Vista_OA
Componente
Plantilla
Animación
Video
Audio
Propiedades


Autor de Objetos de Aprendizaje - [Preview]
Archivo
Editar
Formato
Plantilla
Objeto
Ayuda
Explorador Proyecto
Nivel basido de POO
Proyecto OA
Vista_Guion
Vista_OA
Componente
Plantilla
Animación
Video
Audio
Propiedades
Nombre
Tamaño
An:
Al:
Posición:
X:
Y:
Aplicar

ANEXO 2: Descripciones de las clases.

Nombre: CrearObjetoAprendizaje	
Tipo de clase: Controladora.	
Atributo	**Tipo**
-	**-**
Por cada responsabilidad:	
Nombre:	CrearOA.
Descripción:	Función que define la creación de un OA.

Nombre: GestionarComponente.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
-	**-**
Por cada responsabilidad:	
Nombre:	ImportarComponente
Descripción:	Función que define como se importa un componente.
Nombre:	AdicionarComponente
Descripción:	Función que define como se adiciona un componente.
Nombre:	**ModificarComponente**
Descripción:	Función que define como se modifica un componente.
Nombre:	**EliminarComponente**
Descripción:	Función que define como se elimina un componente.

Nombre: GestionarGuión.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
-	**-**
Por cada responsabilidad:	
Nombre:	ImportarGuión.
Descripción:	Función que define como se importa un guión.
Nombre:	EliminarGuión.
Descripción:	Función que define como se elimina un guión.

Nombre: GestionarOA.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
-	**-**
Por cada responsabilidad:	
Nombre:	CargarOA
Descripción:	Función que define como se carga un OA.
Nombre:	ModificarOA.
Descripción:	Funcion que define como se modifica un OA.

Nombre: GestionarPlantilla.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
-	**-**

Por cada responsabilidad:	
Nombre:	CrearPlantilla.
Descripción:	Función que define como se crea una plantilla.
Nombre:	GuardarPlantilla.
Descripción:	Función que define como se guarda una plantilla.
Nombre:	ModificarPlantilla.
Descripción:	Función que define como se modifica una plantilla.
Nombre:	EliminaPlantilla.
Descripción:	Función que define como se elimina una plantilla.

Nombre: RepositorioCrearOA.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
-	**-**
Por cada responsabilidad:	
Nombre:	AdicionarOA.
Descripción:	Función que permite adicionar un nuevo OA.

Nombre: RepositorioGestionarComponente.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
Por cada responsabilidad:	
Nombre:	AdicionarComponente.

Descripción:	Función que permite adicionar un nuevo Componente.
Nombre:	SeleccionarComponente.
Descripción:	Función que permite definir la selección de Componente.
Nombre:	ModificarComponente.
Descripción:	Función que permite modificar los datos de un Componente.
Nombre:	EliminarComponente.
Descripción:	Función que permite eliminar un Componente.

Nombre: RepositorioGestionarGuión.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
-	**-**
Por cada responsabilidad:	
Nombre:	AdicionarGuión.
Descripción:	Función que permite adicionar un nuevo Guión.
Nombre:	SeleccionarGuión.
Descripción:	Función que permite definir la selección de Guión.
Nombre:	EliminarGuión.
Descripción:	Función que permite eliminar un Guión.

Nombre: RepositorioGestionarOA.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
-	**-**

Por cada responsabilidad:	
Nombre:	AdicionarOA.
Descripción:	Función que permite adicionar un nuevo OA.
Nombre:	SeleccionarOA.
Descripción:	Función que permite definir la selección de OA.
Nombre:	ModificarOA.
Descripción:	Función que permite modificar los datos de un OA.
Nombre:	EliminarOA.
Descripción:	Función que permite eliminar un OA.

Nombre: RepositorioGestionarPlantilla.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
-	**-**
Por cada responsabilidad:	
Nombre:	AdicionarPlantilla.
Descripción:	Función que permite adicionar un nuevo Plantilla.
Nombre:	Seleccionar Plantilla.
Descripción:	Función que permite definir la selección de Plantilla.
Nombre:	Modificar Plantilla.
Descripción:	Función que permite modificar los datos de un Plantilla.
Nombre:	Eliminar Plantilla.
Descripción:	Función que permite eliminar un Plantilla.

Nombre: ObjetoAprendizaje.	
Tipo de clase: Entidad.	
Atributo	**Tipo**
Autor	String
Título	String
Fecha_Creación	Integer
Dirigido	String
Objetivo	String
Por cada responsabilidad:	
Nombre:	SetOA.
Descripción:	Método que se encarga de modificar los valores de cada atributo de la clase.
Nombre:	GetOA.
Descripción:	Devuelve los valores que contienen los atributos de la clase.

Nombre: Componente.	
Tipo de clase: Entidad.	
Atributo	**Tipo**
Nombre	String
Por cada responsabilidad:	
Nombre:	SetComponente.
Descripción:	Método que se encarga de modificar los valores de cada atributo de la clase.
Nombre:	GetComponente.

Descripción:	Devuelve los valores que contienen los atributos de la clase.

Nombre: ComponenteExterno.	
Tipo de clase: Entidad.	
Atributo	**Tipo**
Extensión	String
Por cada responsabilidad:	
Nombre:	SetComponente.
Descripción:	Método que se encarga de modificar los valores de cada atributo de la clase.
Nombre:	GetComponente.
Descripción:	Devuelve los valores que contienen los atributos de la clase.

Nombre: Guión.	
Tipo de clase: Entidad.	
Atributo	**Tipo**
Título	String
Autor	String
Fecha_Creación	Integer
Versión	Integer
Por cada responsabilidad:	
Nombre:	SetGuión

Descripción:	Método que se encarga de modificar los valores de cada atributo de la clase.
Nombre:	GetGuión.
Descripción:	Devuelve los valores que contienen los atributos de la clase.

Nombre: Plantilla.	
Tipo de clase: Entidad.	
Atributo	**Tipo**
Tipo	String
Título	String
Puntuación	Integer
Tiempo	Integer
Intentos	Integer
Por cada responsabilidad:	
Nombre:	SetPlantilla.
Descripción:	Método que se encarga de modificar los valores de cada atributo de la clase.
Nombre:	GetPlantilla.
Descripción:	Devuelve los valores que contienen los atributos de la clase.

GLOSARIO

AICC: (Aviation Industry Computed Based-Training Comitee) es una asociación internacional de capacitación de profesionales basada en tecnología.

API: (del inglés Application Programming Interface - Interfaz de Programación de Aplicaciones) es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Bindings: es una adaptación de una biblioteca para ser usada en un lenguaje de programación distinto de aquél en el que ha sido escrita.

CU: (Casos de Uso), constituyen una forma de representación visual de las funcionalidades del sistema.

Elemento Taxonómico: elemento que compone un proyecto de Objetos de Aprendizaje.

Framework: es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

FreeBSD: es un sistema operativo libre para computadoras basado en las CPU de arquitectura Intel, incluyendo procesadores 386, 486 (versiones SX y DX), y Pentium. También funciona en procesadores compatibles con Intel como AMD y Cyrix.

GPL: Son las siglas de General Public License, Licencia Pública General, definida por la Fundación para el Software Libre (FSF) para proteger los derechos de copia del software libre.

HTML: (HyperText Markup Language, lenguaje de marcas hipertextuales): Lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web.

IBM: (Information Management System) es un gestor de bases de datos jerárquicas y un gestor transaccional con alta capacidad de proceso.

IDE: (Entorno de Desarrollo Integrado o en inglés Integrated Development Environment) es un programa compuesto por un conjunto de herramientas para un programador.

KDE: K Desktop Environment o Entorno de Escritorio K, es un entorno de escritorio e infraestructura de desarrollo para sistemas Unix/Linux.

LCMS: abreviatura de Learning Content Management System, en inglés, es un sistema de gestión de contenidos (CMS) que se utiliza para la enseñanza.

Multiplataforma: Es un término utilizado frecuentemente en informática para indicar la capacidad o características de poder funcionar o mantener una interoperabilidad de forma similar en diferentes sistemas operativos o plataformas.

Patrón de diseño: son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Plataforma: Base, elemento de apoyo.

SCORM: (Sharable Content Object Reference Model) es una especificación que permite crear objetos pedagógicos estructurados.

SQL: (Lenguaje de consulta estructurado en inglés Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas.

WSDL: son las siglas de Web Services Description Language, un formato XML que se utiliza para describir servicios Web.

WYSIWYG: es un acrónimo de What You See Is What You Get (en español, "lo que vez es lo que obtienes"). Se aplica a los procesadores de texto y otros editores con formato (como los editores de HTML) que permiten escribir un documento viendo directamente el resultado final, frecuentemente el resultado impreso. En el caso de editores de HTML este concepto se aplica a los que permiten escribir la página sobre una vista preliminar similar a la de un procesador de textos, ocupándose en este caso el programa de generar el código fuente en HTML.

XML: (eXtensible Markup Language, 'lenguaje de marcas extensible'): Metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Permite definir la gramática de lenguajes específicos, por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG y MathML.

XSD: XML Schema es un lenguaje de esquema utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa, más allá de las normas sintácticas impuestas por el propio lenguaje XML.