Facultad 2

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

COLECCIÓN DE PLUGINS PARA EL MODELADO DE DIAGRAMAS CON EL LENGUAJE ApEM-L 2.0

Autores: Maydel Jerez Sarzo
Reinaldo Viera del Toro

Tutores: Dr.C Febe Angel Ciudad Ricardo
Ing. Yosnel Herrera Martínez

La Habana, junio de 2014
"Año 56 de la Revolución"

"Hacer o no hacer, no existe el intentar."

Yoda

StarWars.

Maydel:

A mi titi (mama) por ser la mujer más bonita, inteligente y trabajadora, que he conocido en mi vida. No alcanzan palabras para describirte y agradecerte lo suficiente todo lo que representas para mí. Te quiero muchísimo titi.

A mi tito (papa) por ser ese gigante que cuida mis sueños. Por su eterno sacrificio, por enseñarme el significado de la palabra responsabilidad y preocupación. A ti, mi eterno guardián.

A mi hermano Andy que ha sido mi ejemplo, mi compañía y mi risa. Te quiero mucho mi gordito lindo.

A mi abuela por estar siempre pendiente y preocupada de mí, por mi carrera, (...) por todo, y quien es dueña de un pedazote de mi corazón.

A mi tío Renecito y a mi tía Cristina quien no solo se preocupa siempre por mí, sino que también siempre están dispuestos a ayudar si hace falta.

A mis primos Karen y Renecito, que más que primos son como hermanos. Gracias por su buena voluntad, y cariño hacia mí.

A mis mejores amigas Nery y Maylin, que desde hace ya unos cuantos años dejamos de ser amigas, para ser hermanas. Gracias a ustedes que nunca me ha defraudado y siempre están ahí para compartir aquellos secretos e historias tan de nosotras.

Al amor de mi vida, Reinaldo. Gracias por enseñarme los nuevos matices de la vida, aquellos que solo las personas enamoradas pueden ver.

A mis suegros, por recibirme con los brazos abierto y darme la oportunidad de ser parte de su pequeña, pero grandiosa familia.

A mi tutor, por su tiempo y conocimiento volcado en la ayuda para la realización de la presente memoria. Me llena de orgullo, haber conocido una persona de alma tan hermosa y generosa como la de usted.

A mi nuevo primito Luis Gabriel que a pesar de haberte conocido hace poco, te quiero como si te conociera de hace mucho.

A mis compañeros de aula y de apartamento con quienes he pasado estos cinco años de constante aprendizaje.

A Cesar, que nunca ha dejado de demostrar su cariño y alegría cuando llego a la casa.

A mi titi y a mi tito, por ser los dioses que cuidan de mi mundo y me dan su hombro para apoyarme cuando lo necesito.

Reinaldo:

A mamá, por sufrir y disfrutar conmigo en todos los momentos de mi vida.

A pepe, por ser el ejemplo que siempre seguiré para ser una mejor persona.

A mis hermanos René, Cossette, y Thais, por apoyarme y ayudarme siempre que lo necesité.

A mi familia en todos los rincones de Cuba, por acogerme tan bien durante toda mi vida.

A mi tutor, por ser el educador que me enseñó a hacer ciencia y ser un mejor profesional.

A mis amigos de Santiago, por nunca defraudarme y siempre estar ahí para mí.

A mis compañeros de aula en la UCI, por aguantarme todos estos años.

A mis amigos Alfredo y Alamino por su apoyo incondicional en la universidad.

A la revolución y a la UCI, por darme la oportunidad de convertirme en ingeniero.

A mí, por aguantarme toda mi vida.

A mi papapishu Maydel por entrar en mi vida cuando más lo necesitaba, y hacer de ella una dulce locura que solo ella y yo podemos adorar.

Declaramos que somos autores del presente trabajo de diploma y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales sobre la misma, con carácter exclusivo. Para que así conste, firmamos la presente a los ___ días del mes de _____ del año _____.

Maydel Jerez Sarzo

_____

Firma del Autor

Reinaldo Viera del Toro

_____

Firma del Autor

Dr.C Febe Angel Ciudad Ricardo

_____

Firma del Tutor

Ing. Yosnel Herrera Martínez

_____

Firma del Tutor

Dr.C. Febe Angel Ciudad Ricardo. Graduado como Ingeniero Informático en el año 2004 por la universidad de Holguín “Oscar Lucero Moya” (UHOLM) y el Instituto Superior Politécnico “José Antonio Echeverría” (CUJAE). Titulado como Máster en Informática Aplicada en el año 2007 por la Universidad de las Ciencias Informáticas (UCI) y obtuvo el grado científico de Doctor en Ciencias de la Educación – Especialidad Tecnología Educativa en el año 2012 por la Universidad de La Habana (UH). Imparte su docencia de pregrado en las disciplinas de Ingeniería y Gestión de Software, Metodología de la Investigación Científica y Formación Pedagógica. Es miembro de los claustros de las maestrías de Informática Aplicada, Informática Avanzada, Gestión de Proyectos y Educación a Distancia de la UCI. Desarrolla sus investigaciones en las temáticas de Ingeniería y Gestión de Software, con énfasis en el área del Software Educativo; así como en la Tecnología e Informática Educativas. Ha publicado diversos artículos científicos y ha participado en diferentes eventos nacionales e internacionales en estas áreas del conocimiento. Ha sido arquitecto, analista y líder de proyectos de desarrollo de software, jefe de departamento docente y asesor técnico – docente. Actualmente se desempeña como Director del Centro de Innovación y Calidad de la Educación (CICE) de la UCI.

Ing. Yosnel Herrera Martínez. Graduado como Ingeniero en Ciencias Informáticas en el año 2006 por la Universidad de Ciencias Informáticas (UCI). Obtuvo la categoría docente de Profesor Asistente en el año 2013. Imparte su docencia de pregrado en la disciplina de Ingeniería y Gestión de Software. Desarrolla sus investigaciones en las temáticas de Ingeniería y Gestión de Software, con énfasis en el área del modelado de Software Educativo. Ha publicado diversos artículos científicos y ha participado en diferentes eventos nacionales e internacionales en estas áreas del conocimiento. Ha sido analista, diseñador y probador de proyectos de desarrollo de software; así como jefe de asignatura. Actualmente se desempeña como profesor del Dpto. de Ingeniería y Gestión de Software de la Facultad 4 de la UCI

Un lenguaje de modelado es aquel cuya función principal es representar el conocimiento asociado a los sistemas informáticos a través de su metamodelo. ApEM-L, como lenguaje de modelado, surge por la necesidad de emplear un lenguaje notacional que se ajustara a las características del software educativo cubano y que representara en modelos: la estructura lógica, el comportamiento y las funciones del sistema a desarrollar. Para el soporte de los diagramas que se generan mediante los lenguajes de modelado se usan las denominadas herramientas CASE. La presente investigación se enmarca en la actividad de modelado de aplicaciones educativas, mediante herramientas CASE en la UCI. Propone el desarrollo de un generador de diagramas que dé apoyo a la actividad de modelado de los proyectos productivos encargados del desarrollo de aplicaciones educativas en la UCI. Lo cual permitirá aumentar la precisión y la representatividad de los diagramas que se generen. Ambas variables fueron elaboradas a partir de la sistematización de los principales referentes teórico-metodológicos relativos a los diagramas resultantes de la actividad de modelado. La operacionalización del concepto completitud de las herramientas CASE en la actividad de modelado de aplicaciones educativas, posibilitó la identificación de una sistema de características para contribuir con la precisión y representatividad diagramas resultantes de la actividad de modelado. Para valorar dicha contribución se realizaron pruebas de caja blanca y experimento con un caso de estudio.

Palabras clave:

ApEM-L, ApEM-L 2.0, Desarrollo de Software Dirigido por Modelos, Graphic Modeling Framework, MDD.

.

ÍNDICE

INTRODUCCIÓN

Una de las maneras de garantizar la productividad y calidad en las actividades del Proceso de Desarrollo de Software (PDS) es a partir de las herramientas CASE (Computer Aided Software Engineering). Estas intentan automatizar las actividades del PDS o algunas de ellas, de manera tal que se facilite el trabajo de los desarrolladores de software y se aumente la productividad en el proceso.

El modelado es una de las actividades genéricas que componen el PDS. Refiriéndose a la importancia de la actividad de modelado, el autor (Somerville, 2005) expresa que *«debido a las representaciones gráficas utilizadas, los modelos suelen ser más comprensibles que las detalladas descripciones en lenguaje natural de los requisitos del sistema. También son un puente importante entre los procesos de análisis y diseño».* De esta actividad surgen modelos, los cuales se construyen para representar los resultados de las actividades de análisis y de diseño. A su vez, los modelos están constituidos por diagramas.

Las herramientas CASE que apoyan a las actividades genéricas iniciales del PDS (como el modelado), son denominadas Upper-CASE (U-CASE). Una de las principales características de las U-CASE, es que permiten la generación de diagramas a partir de las especificaciones de los usuarios o de manera automática. El autor (Lopez, Pascual Gonzáles y otros, 1999) expresa que en algunos casos, la herramienta CASE permite el chequeo de la consistencia y el refinamiento entre los diagramas.

Por otra parte, la (OMG, 2012) plantea que, para la generación de diagramas es necesario definir primeramente, una sintaxis abstracta (metamodelo) del lenguaje, para luego crear las reglas y restricciones de los modelos. Posteriormente los usuarios, modelan sus aplicaciones creando instancias de los elementos de la sintaxis abstracta, dando como resultado la representación gráfica de los diagramas en la herramienta que se utilice.

Por lo expresado anteriormente, se dedujo que una de las principales características de una U-CASE es su capacidad de cubrir o soportar a través de su metamodelo todas las necesidades de representación del software existente. Además de que pueda describir, a partir de este metamodelo, suficientemente las características sintácticas y semánticas de sus conceptos de modelado. Lo anterior se denominó en la investigación como nivel de completitud de la herramienta CASE.

El hecho de que un lenguaje de modelado cuente con una herramienta CASE que genere sus diagramas, es ventajoso para cualquier equipo de desarrollo. Autores como (Cruz y otros, 2010) y (Rojas y otros, 2008), aseguran que el uso de las U-CASE mejoran la productividad del PDS y la calidad de su producto

final, de manera que permiten hacer más eficiente y coherente la actividad de modelado de sus software. Además otros autores como (Rumbaugh y otros, 2007) expresan que, lenguajes de modelado como el Lenguaje Unificado de Modelado (UML por sus siglas en inglés), proponen que para modelar con su semántica sistemas de tamaño real, es necesario el uso de herramientas, la cuales aporten formas interactivas de ver y editar los modelos. Esto, proporciona un nivel de organización que se encuentra fuera del ámbito del propio UML, pero que aporta comprensión al usuario y le ayuda en el acceso a la información. Estas herramientas facilitan la búsqueda de información en modelos de gran tamaño.

Según (Ciudad y otros, 2012) y (Pedrajas, 2005) una de las áreas a las que ha embebido la informática (desde su surgimiento) ha sido la de los sistemas educacionales; que ha dado lugar a lo que se conoce como Informática Educativa, así como su principal resultado: las aplicaciones educativas.

En Cuba, con más de 20 años de experiencia en su desarrollo, se han logrado avances que proporcionan las herramientas necesarias para fomentar un conjunto de características propias para una nueva alternativa revolucionaria dentro de la enseñanza a todos los niveles: las aplicaciones educativas cubanas. Dichos avances, a criterio de (Ciudad, 2007) son distintivos de los restantes en el mundo y potencian algunas áreas específicas mencionadas en las investigaciones de este autor.

De esta manera surge en el año 2007, en la Universidad de las Ciencias Informáticas (UCI) en Cuba, el Lenguaje de Modelado Orientado a Objetos para Aplicaciones Educativas y Multimedia (ApEM-L 1.0), creado por el Dr.C. Febe Ángel Ciudad Ricardo. Según (Ciudad, 2013) este lenguaje responde a *«la necesidad de emplear un lenguaje notacional que se ajustara a las características del software educativo cubano y que representara en modelos: la estructura lógica, el comportamiento y las funciones del sistema a desarrollar»*. El mismo, es clasificado como un lenguaje de modelado de dominio específico (DSML por sus siglas en inglés), lo cual le atribuye ciertas características para modelar las aplicaciones educativas y de multimedia cubanas.

Según (Herrera, 2014a) y (Muñoz y otros, 2008), una de las razones de la aceptación de ApEM-L fue que los desarrolladores tuvieron la posibilidad de utilizar como soporte, la representación de las herramientas CASE que existen para UML, facilitando el trabajo y ahorrando tiempo para el modelado. De esta manera los desarrolladores que optaron por el uso de este lenguaje pudieron, entre otras funcionalidades, generar los diagramas con las mismas herramientas que existen para UML.

ApEM-L 1.0 trae consigo una serie de diagramas y conceptos, los cuales son resultado de extender el lenguaje UML. Tras la concepción de la versión 2.0 del lenguaje algunos de estos conceptos y diagramas

son modificados y otros son creados. Estos elementos son enriquecidos mediante la especificación textual de su sintaxis y semántica, y la definición de su metamodelo. Entre los diagramas modificados de ApEM-L 1.0 se encuentran los siguientes:

- Diagrama de estructura de presentación. Este según (Herrera, 2014a) permitió modelar la estructura lógica de la capa de presentación de las aplicaciones educativas; específicamente las estructuras de las interfaces de usuario (IU por sus siglas en inglés), sus componentes y las características interactivas de estos últimos.
- Diagrama de estructura de navegación, a partir del cual, según (Herrera, 2014a) se permitió representar gráficamente la estructura navegacional de la aplicación educativa; es decir, todos aquellos componentes que intervienen o influyen en la navegación del sistema.
- Diagrama de casos de uso, el cual extiende del que ya existe de UML, incorporando nuevos conceptos que permiten el modelado, mediante casos de uso, de escenarios en un entorno educativo.
- Diagrama de clases del diseño, el cual extiende del que ya existe de UML, incorporando nuevos elementos para modelar bajo el patrón Modelo-Vista-Controlador-Entidad MVC-E propuesto por (Ciudad, 2006).

Además, esta nueva versión de ApEM-L trae consigo una serie de cambios en su estructura, los cuales son mencionados a continuación:

- Es definido el metamodelo de los diagramas pertenecientes a la vista estática y a la vista de presentación de ApEM-L.
- Es definida la sintaxis de los diagramas pertenecientes a la vista estática y a la vista de presentación de ApEM-L.
- Es definida la semántica de los diagramas pertenecientes a la vista estática y a la vista de presentación de ApEM-L.

En general, el uso de los diagramas mencionados, en el modelado de las aplicaciones educativas cubanas permitió una visión más amplia del sistema que se desea modelar. Los elementos de ApEM-L fueron creados para su uso en dominio descrito, respondiendo a necesidades que no se podían resolver con el uso de otros lenguajes de modelado. Además la concepción de la versión 2.0 ApEM-L le brinda formalidad y completitud a dicho lenguaje.

Sin embargo, las nuevas especificaciones que trae consigo la nueva versión del lenguaje no permiten su correcto modelado en las herramientas CASE existentes. Esta situación sucede debido a las siguientes razones:

- No es posible representar todos los nuevos estereotipos decorativos en la construcción de los diagramas, solo los que son decorados mediante estereotipos de UML.
- Las restricciones expresadas en el metamodelo no son interpretadas.
- No existe la posibilidad de definir el metamodelo de la nueva versión del lenguaje en las herramientas, para su correcto diagramado.

Estas deficiencias hacen que el uso de ApEM-L 2.0 en las herramientas existentes hoy para UML pueda traer consigo inconsistencias en los diagramas que se generen. Además disminuye la representatividad de los diagramas, pues los estereotipos decorativos creados no se pueden representar en su totalidad. Lo cual dificulta el uso y la inserción del lenguaje en los proyectos productivo que necesiten de su uso.

Mediante la observación, y el análisis bibliográfico de tesis de grado desarrolladas para el dominio de aplicaciones educativas en la UCI se pudo identificar las siguientes deficiencias en los diagramas que con su uso representan los sistemas que se construyen:

- Los diagramas de las investigaciones donde se usó UML, no logran expresar las características del dominio en el que se desarrollan.
- No se modelan con exactitud todas las características de las aplicaciones, que finalmente se observan en los productos finales.
- El alcance de los diagramas es limitado debido al uso de notaciones y conceptos de modelado no concebidos por el lenguaje.
- Incorrecta interpretación de diagramas que no está en correspondencia con su propósito u objetivo según el metamodelo.

Lo anterior, influye en que los diagramas generados presenten deficiencias, que no le permitan modelar con exactitud las características de las aplicaciones que intentan representar. Dichas deficiencias se traducen en una disminución de la precisión y representatividad de los diagramas del modelado de las aplicaciones educativas en la UCI. Así se presentó la necesidad de usar la versión 2.0 del ApEM-L, pues permitió resolver los problemas anteriormente descritos. Sin embargo, las herramientas usadas para el modelado de aplicaciones educativas en la UCI, no permiten el uso de este lenguaje. Esto permite inferir

que existe un bajo grado de completitud en la actividad de modelado de aplicaciones educativas en la UCI, mediante el uso de herramientas CASE.

A partir de lo planteado anteriormente, se definió el siguiente **problema científico**: Insuficiencias en la completitud de la actividad de modelado, de aplicaciones educativas en la UCI, disminuyen el grado de precisión y representatividad de los diagramas generados en este dominio de aplicación.

Se planteó como **objeto de estudio:** La actividad de modelado de aplicaciones educativas, mediante herramientas CASE, en la UCI. Como **campo de acción**: Los generadores de diagramas de las herramientas CASE para el modelado de dominio específico.

El **objetivo de la investigación** es: desarrollar un generador de diagramas, que dé apoyo a la actividad de modelado de los proyectos productivos encargados del desarrollo de aplicaciones educativas en la UCI y contribuya a aumentar la precisión y la representatividad de los diagramas que se generen.

De esta manera se planteó como **hipótesis de trabajo:** la utilización de un generador de diagramas en una herramienta CASE, en la actividad de modelado de las aplicaciones educativas en la UCI, contribuirá a aumentar la precisión y la representatividad en los diagramas resultantes de su uso.

Para el cumplimento del objetivo de la investigación se plantean las siguientes **tareas investigativas**:

1. Establecimiento de los referentes teóricos-metodológicos para la construcción de generadores de diagramas en herramientas CASE para el modelado de dominio específico.
2. Caracterización del actividad de modelado para el desarrollo de aplicaciones educativas en la UCI, mediante herramientas CASE.
3. Construcción de un generador de diagramas para una herramienta CASE que de apoyo a la actividad de modelado para aplicaciones educativas.
4. Validación de la propuesta de solución mediante su aporte al grado de precisión y representatividad de los diagramas del modelado que se generen.

Las unidades de análisis de la investigación son los diagramas resultantes de la actividad de modelado, específicamente los pertenecientes a la vista estática y a la vista de presentación. Para la ejecución de la investigación se trabajó con una **población** donde cada unidad muestral tuviera ambas de las siguientes características:

- Documentos científicos de la UCI que contuvieran los resultados de investigaciones desarrolladas utilizando el lenguaje ApEM-L en cualquiera de sus versiones 1.0 o 1.5.

- Documentos científicos o cualquier otra forma de socialización de resultados científicos en la UCI que contuvieran los diagramas como parte de los resultados de investigaciones desarrolladas utilizando el lenguaje ApEM-L en cualquiera de sus versiones 1.0 y 1.5.

La **muestra** es probabilística (aleatoria simple), pues es interés de la investigación, que todos los elementos tengan la misma probabilidad de ser seleccionados y porque los resultados que se obtengan, deben ser generalizables a toda la población y a los futuros trabajos de modelado.

Durante el desarrollo de la investigación se usaron los siguientes **métodos científicos**:

Métodos teóricos:

Hipotético – Deductivo: la presente investigación tiene como presupuesto hipotético la hipótesis, y la misma será la que guíe la investigación al cumplimiento de sus objetivos y tareas de trabajo. De esta manera el método hipotético deductivo jugará un papel fundamental en la comprobación de la hipótesis.

Análisis Histórico-Lógico: posibilitará hacer un estudio de los antecedentes que puede tener el proceso que se desea automatizar. Con la ayuda de este método se logrará realizar un análisis de otras soluciones existentes a problemas similares.

Modelación: permitirá una abstracción completa del objeto con la realidad, captando la esencia del problema que se desea automatizar. Este método permitió al equipo conocer con exactitud cuáles son las características del sistema mediante la modelación de los procesos del negocio, utilizando para este fin diagramas y figuras.

Sistémico: para lograr que los elementos que forman parte de la solución funcionen de manera armónica como un sistema.

Métodos Empíricos:

Observación: este método permitió analizar el comportamiento lógico de las variables de la investigación en el de cursar del tiempo, dentro del campo del objeto de estudio seleccionado, permitiendo comparar resultados en las distintas fases de la investigación.

Entrevistas: a partir de las mismas se logró conocer el comportamiento lógico de las variables de la investigación en el decursar del tiempo, a partir del punto de vista de los analistas de los proyectos seleccionados.

.

CAPÍTULO I: LENGUAJES DE MODELADO Y HERRAMIENTAS PARA SU VALIDACIÓN

El siguiente capítulo aborda un análisis de los conceptos más importantes que fundamentan la manera en que las herramientas CASE generan diagramas para la actividad de modelado. Se realizó además, un estudio de las herramientas de modelado usadas en el ámbito internacional y en UCI. También se definieron las herramientas y la metodología a utilizar en el desarrollo de la investigación

1.1 Principales conceptos asociados a la actividad de modelado

Modelos y Modelado

Según el diccionario de la real academia española (DREA, 2014), se puede definir que el modelo detalla algo que ya está elaborado o bien puede estar por elaborarse. Los campos de la ingeniería de software, desarrollan sus aplicaciones basándose en modelos, al igual que muchas acciones en la vida. Autores como (Pons y otros, 2010) explican que en general el modelo debe señalar de manera clara y correcta lo que se quiere lograr, sin sobrecargar de información al mismo. Agregan que éste debe ser completo.

Autores como (Pons y otros, 2010) expresan que los modelos son utilizados en la ingeniería para varios propósitos. Dentro de estos se puede mencionar que son utilizados para captar y enumerar exhaustivamente los requisitos y el dominio de conocimiento, de forma que todos los implicados puedan entenderlos y estar de acuerdo. También, son utilizados para ayudar a pensar el diseño de un sistema. Además, son utilizados para capturar decisiones del diseño de manera tal que puedan ser cambiadas a partir de los requisitos. Otros de sus usos es el de generar productos aprovechables para el trabajo y facilitar el manejo de los sistemas complejos. Los modelos de los sistemas se construyen utilizando un lenguaje de modelado que puede variar desde lenguaje natural o diagramas hasta fórmulas matemáticas.

Autores como (Portillo, 2004) y (Llanes y otros, 2008) arriban al concepto de modelado como un ejercicio de abstracción entendida, ésta como una simplificación y generalización de la realidad.

Varios autores como (Llanes y otros, 2008), (Pons y otros, 2010), (Montenegro, Carlos E. y otros, 2011b), (Portillo, 2004) coinciden en que existen cuatro espacios de modelado o niveles; los cuales se construyen en jerarquías de abstracción, en las que cada nivel se apoya en los inferiores:

- Niveles base o M0 (Información): Agregación informal de datos a manejar en un entorno concreto (aplicación). Se separa un subconjunto de datos con características comunes o interesantes desde un determinado punto de vista.
- Niveles M1 (Modelo): Agregación informal de metadatos (datos sobre los datos) que describen una información concreta. Se describen las características comunes de los datos dando lugar a metadatos que se agrupan, describiéndose las relaciones entre ellos, para formar un modelo.
- Niveles M2 (Metamodelo): Agregación informal de modelos o meta-metadatos (descripción informal de los metadatos) Descripciones que definen la estructura y la semántica de los metadatos. Se describen las características comunes de subconjuntos de metadatos dando lugar a meta metadatos o modelos que se agrupan, describiéndose las relaciones entre ellos, para formar un metamodelo.
- Niveles M3 (Meta-metamodelo): Definición de la estructura y la semántica de los meta-metadatos. Se capturan las características comunes de subconjuntos de modelos dando lugar a metamodelos que se agrupan, describiéndose las relaciones entre ellos, para formar un meta-metamodelo.

(Portillo, 2004) Expresa que el propósito de modelar estos niveles de abstracción tan altos, es proporcionar un mecanismo común que permita la interoperabilidad de los sistemas a través de la transformación de un modelo a otro. Para un mejor entendimiento, la siguiente figura sirve como guía.

NIVELES DE ABSTRACCIÓN	ENTIDADES MANEJADAS	LENGUAJES DEFINIDOS
Meta-metamodelo	Meta-modelos	Lenguaje para la descripción de diferentes tipos de modelos. Lenguaje de modelado
Meta-modelo	Meta-metadatos ó Modelos	Lenguaje para la descripción de diferentes tipos de datos (meta-datos)
Modelo	Meta-datos	Lenguaje para la descripción de datos concretos
Información	Datos	

Figura 1: Niveles de abstracción del modelado,[Tomado de (Portillo, 2004)]

Metamodelo y Meta-metamodelo

Como muestra la figura 2, los modelos describen los elementos del mundo real, sin embargo estos necesitan de una guía formal para su realización. De manera tal que no existan inconsistencias en la especificación de los mismos, y sean comprensibles para quien los lea. Esta guía es denominada como metamodelo.

Científicos como (Montenegro, Carlos E. y otros, 2011b) revelan que un metamodelo son las capas que almacenan y que permiten la creación de un modelo, como una descripción de uno o varios elementos del dominio o mundo real. Finalmente, el meta-metamodelo describe a esos metamodelos planteados, generando un grado de abstracción supremamente alto en el cual coinciden todos los modelos. Así, de la misma manera que un metamodelo es necesario para la construcción de un modelo, un meta-metamodelo es necesario para la construcción de un modelo, solo que en un nivel superior de abstracción.

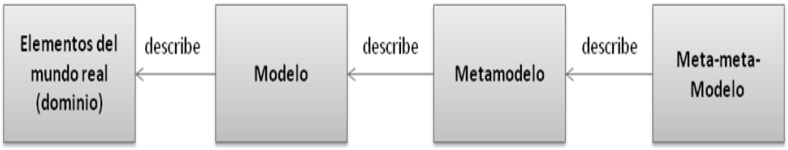

Figura 2: Modelo, metamodelo y meta-metamodelo, [Tomado de (Montenegro, Carlos E. y otros, 2011b)]

Otro concepto abordado es el de (Gonzales, 1998) quien plantea que *«el metamodelo es la capa donde se define el lenguaje que sirve para especificar los modelos que serán creados».* O sea, que sirve para describir los elementos que van a componer los diagramas.

Desarrollo de software dirigido por modelos (MDD)

Varios autores como (Pons y otros, 2010) y (Montenegro, Carlos E. y otros, 2011b) afirman que el desarrollo de software dirigido por modelos (Model Driven software Development, MDD o Model Driven Engineering, MDE como se le conoce en otras bibliografías), se ha convertido en un paradigma para el desarrollo de software. Este, mejora el proceso de construcción del software, al guiarlo por modelos y soportarlo en potentes herramientas. Los modelos se van generando desde los más abstractos a los más concretos, a través de pasos de transformación y/o refinamiento, hasta llegar al código.

En el proceso de desarrollo de software sin el uso de MDD, se realizan transformaciones de modelo a modelo, o de modelo a código. Sin embargo estas transformaciones no van más allá de la generación de algún esqueleto de código, que luego se debe completar manualmente. Lo novedoso que propone MDD es que las transformaciones entre modelos sean automatizadas.

(Pons y otros, 2010) Plantean que en la actualidad existen dos propuestas concretas; las cuales son las más conocidas y utilizadas en el ámbito de MDD. Estas son: Arquitectura dirigida por modelos (Model Driven Architecture, MDA por sus siglas en inglés) desarrollada por el Grupo OMG (Object Management Group) y por otro lado el modelado de domino específico (Domain Specific Modeling, DSM por sus siglas en inglés) acompañado por los lenguajes de dominio específico (Domain Specific Language, DSL por sus siglas en inglés). Ambas iniciativas guardan una fuerte conexión con los conceptos básicos de MDD. Específicamente, MDA tiende a enfocarse en lenguajes de modelado basados en estándares del OMG, mientras que DSM utiliza para definir sus modelos otras notaciones no estandarizadas.

Según (Pons y otros, 2010) tiene cuatro tipos de modelos fundamentales para el modelado dentro el proceso de desarrollo de software, que se explicarán a continuación.

- Modelo independiente de la computación (Computation Independent Model, CIM): un CIM es una vista del sistema desde un punto de vista independiente de la computación. Un CIM no muestra detalles de la estructura del sistema. Usualmente al CIM se lo llama modelo del dominio y en su construcción se utiliza un vocabulario que resulta familiar para los expertos en el dominio en cuestión. Se asume que los usuarios a quienes está destinado el CIM no poseen conocimientos técnicos acerca de los artefactos que se usarán para implementar el sistema. Este modelo, juega un papel muy importante en reducir la brecha entre los expertos en el dominio y sus requisitos por un lado, y los expertos en diseñar y construir artefactos de software por el otro.
- Modelo independiente de la plataforma (Platform Independent Model, PIM): un PIM es un modelo con un alto nivel de abstracción que es independiente de cualquier tecnología o lenguaje de implementación. Dentro del PIM el sistema se modela desde el punto de vista de cómo se soporta mejor al negocio, sin tener en cuenta cómo va a ser implementado: ignora los sistemas operativos, los lenguajes de programación, el hardware, la topología de red, etc. Por lo tanto un PIM puede luego ser implementado sobre diferentes plataformas específicas.

- Modelo específico de la plataforma (Platform Specific Model, PSM): un PSM se transforma en uno o más modelos específicos de plataforma. Cada PSM representa la proyección del PIM en una plataforma específica. Un PIM puede generar múltiples PSM, cada uno para una tecnología en particular. Generalmente, los PSM deben colaborar entre sí para una solución completa y consistente. Por ejemplo, un PSM para Java contiene términos como clase, interfaz, etc. Un PSM para una base de datos relacional contiene términos como tabla, columna, clave foránea, etc.
- Modelo de la implementación (Código): el paso final en el desarrollo, es la transformación de cada PSM a código fuente, debido a que el PSM está orientado al dominio tecnológico específico. Esta transformación es bastante directa.

Ejemplo gráfico del ciclo de vida del software dirigido por modelos.

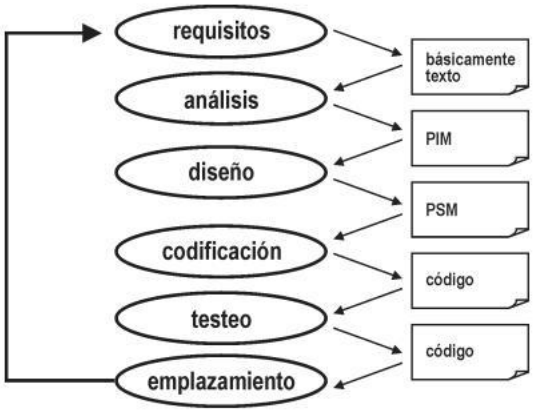

Figura 3: Ciclo de vida del desarrollo de software dirigido por modelos. [Tomado de (Pons y otros, 2010)]

1.2 Lenguajes de modelado

Autores como (Amatriain y otros, 2011) expresan que el lenguaje de modelado *«es aquel cuya función principal es representar el conocimiento asociado a los sistemas informáticos a través de su metamodelo; es decir, el modelo que lo especifica técnicamente y que a su vez lo define»*.

Lenguajes de metamodelado

Algunas bibliografías se refieren a los lenguajes de modelado más abstractos, como lenguajes de metamodelado y otras lo mencionan como metalenguajes. Estos lenguajes se utilizan para describir otros

lenguajes. Autores como (Pons y otros, 2010) explican que la definición de un leguaje a través de otro lenguaje de metamodelado se llama metamodelo. Como un metamodelo es también un modelo, el primero en sí mismo debe estar escrito en un lenguaje bien definido. Este lenguaje se llama metalenguaje o lenguaje de metamodelado. Los metalenguajes o lenguajes de modelado más utilizados actualmente son: MOF (Meta Object Facility), EMOF (Ecore-Meta Object Facility), y OCL (Object Constraint Language).

Según (Llanes y otros, 2008) con el uso de leguajes de metamodelado se pueden definir los lenguajes específicos para determinados dominios de aplicación, los cuales al estar restringidos a un contexto particular, ofrecen técnicas más potentes de análisis y generación de código, aumentando la productividad y mejorando la calidad de los elementos generados.

Lenguajes de modelado de dominio específico

Uno de los lenguajes de modelado más utilizados a nivel internacional es el UML. En la investigación realizada por (Herrera, 2014a) se plantea que desde el surgimiento de UML, los avances científico-tecnológicos en el área del modelado; han trasformado aceleradamente los lenguajes al punto de convertirse en el centro de atención de uno de los enfoques incipientes de la ISW, el MDD. Definiciones como DSML y perfil UML aparecen en MDD para enfatizar el modelado de conceptos de ciertos dominios particulares. UML como lenguaje de propósito general, ha sido uno de los estándares inspiradores en este sentido. El calificativo de “dominio específico” por ejemplo, se ha hecho común en la ciencia, como muestra de una clara necesidad de los ingenieros de representar sistemas informáticos de una determinada área del conocimiento. Ejemplos de DSML son: el Lenguaje Orientado a Objetos para la Modelación de Aplicaciones Multimedia (OMMMA-L por sus siglas en inglés) por (Engels y otros, 1999) y el ApEM-L por (Ciudad, 2007).

Los DSML cuentan con un universo limitado de aplicación. No obstante, gracias a esto, presentan facilidades y ventajas para abordar los problemas de modelado de software para los que fueron diseñados, es decir, se vuelven especialistas del dominio y ahorran tiempo de análisis, diseño y codificación de software. (Herrera, 2014a)

1.3 Herramientas para la generación de diagramas

Las herramientas para la generación de diagramas tienen una relevancia enorme en el proceso de desarrollo de software. Expresa (Somerville, 2005) que estos instrumentos permiten la construcción y el manejo de los artefactos de la actividad de análisis y diseño del software, además de elevar notablemente la calidad, la eficiencia y la eficacia del software a desarrollar.

Herramientas CASE

Actualmente se evidencian de forma notable en el proceso de desarrollo de software, el uso de las herramientas CASE. Autores como (Somerville, 2005) explica que las tecnologías CASE comprenden un amplio abanico de diferentes tipos de programas, que se utilizan para ayudar en actividades del proceso de desarrollo de software. Existen varias formas de clasificar las herramientas CASE, cada una de las cuales nos proporciona una perspectiva distinta.

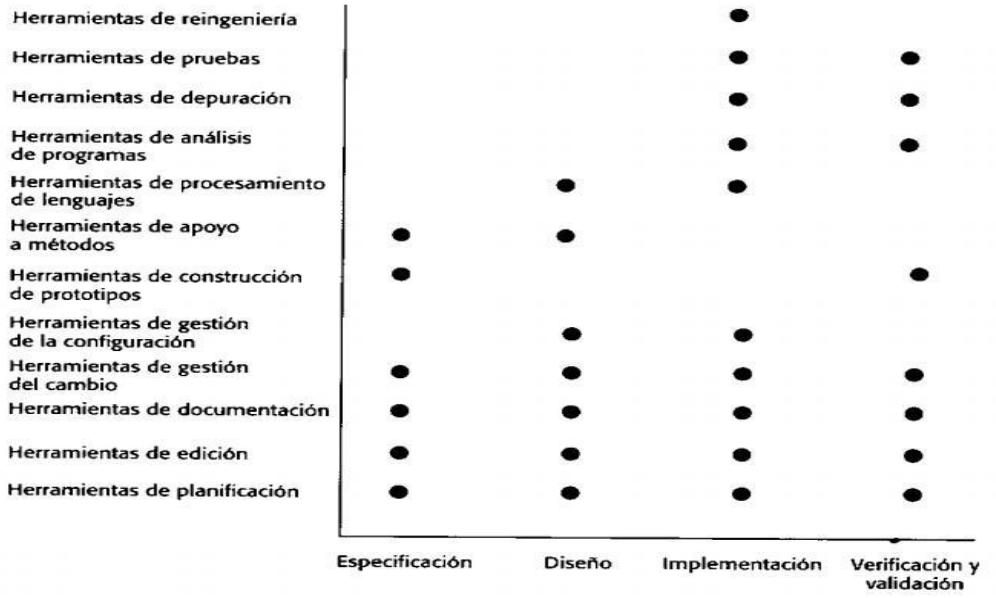

Figura 4: Clasificación de las herramientas CASE basadas en las actividades, [Tomado de (Somerville, 2005)]

La figura 4 representa una clasificación de las herramientas CASE. Esta muestra las actividades del proceso que reciben ayuda por varios tipos de herramientas CASE. Las herramientas para la planificación y estimación, edición de texto, preparación de documentos y gestión de la configuración, pueden utilizarse durante todo el proceso de software.

Herramientas CASE para la actividad de modelado

Para que la investigación sea completa se procedió a realizar un estudio a un conjunto de herramientas de modelado que existen actualmente tanto en el ámbito nacional como en el extranjero. Se realizó el análisis valorando indicadores como:

- Soporte: para conocer qué sistema operativo es compatible para cada herramienta. El valor esperado en la herramienta que se busca es que la misma sea multiplataforma.
- Código Abierto: al presentar la facilidad de ser de código abierto, la herramienta brinda la posibilidad de aprender acerca de la realización de la misma, así como facilitar la integración de nuevas funcionalidades.
- Licencia: al conocer este indicador, se puede saber la accesibilidad a dicha herramienta.
- Lenguaje de programación: esto indica los lenguajes de programación utilizados en la construcción de la herramienta.
- Arquitectura: indica si la arquitectura que presenta la herramienta es dirigida por modelos o no.
- Lenguaje de modelado soportado: se refiere al lenguaje de modelado utilizado en la herramienta.

Tabla 1: Comparación de las herramientas de modelado analizadas.

Herramientas	Soporte	Código Abierto	Licencia	Lenguajes de programación	MDA	Lenguaje modelado soportado
ArgoUML	Multiplataforma	sí	Libre	Java	-	UML
ArgoUWE	Windows	si	Libre	Java	sí	UWE
CORBA tracer	Multiplataforma	-	Libre	Java	-	CORBA
Eclipse	Multiplataforma	sí	Libre	Java	sí	UML 2.0
Enterprise Architect	Windows, (compatible con la instalación de Linux y Mac)	no	Comercial	C, C++, C#, Java, PHP, Delphi, VB, ActionScript	sí	UML 2.0
MagicDraw	Multiplataforma	no	Comercial	Java	sí	UML 2.0
MagicUWE	Windows	no	Comercial	Java	-	UWE

Herramientas	Soporte	Código Abierto	Licencia	Lenguajes de programación	MDA	Lenguaje modelado soportado
Modelio	Multiplataforma	sí	Comercial	Java, C++	sí	UML 2.0
Microsoft (Visual Studio)	Windows	no	Comercial	-	-	Plugin UML 2.0
Poseidon for UML	Multiplataforma	no	Comercial	Java	-	UML 2.0
Power Designer	Windows	no	Comercial	-	sí	UML 2.0
Prosa UML Modeler	Windows	no	Comercial	C++	sí	UML 2.0
Rational Rose	Multiplataforma	no	Comercial	-	-	UML
The rCOS Modeler	-	no	Comercial	-	-	rCOS
Visual Paradigm	Multiplataforma	no	Comercial, Edición libre solo para la comunidad	Java	-	UML 2.0

Con la ayuda de sitios como la página de lenguajes de modelado (www.modeling-languages.com/uml-tools/) y la página de herramientas de desarrollo de software (www.softdevtools.com);se realizó la tabla anterior, la cual muestra una comparación de las herramientas estudiadas, en cuanto a los indicadores descritos anteriormente.

Gracias a ArgoUWE y MagicUWE, se evidenció la realización de herramientas con un propósito similar al trazado en la investigación. Estas son plugins, los cuales usan las funcionalidades de ArgoUML y MagicDraw respectivamente, agregando nuevas para el correcto diagramado del lenguaje UWE (UML-Based Web Engineering).

Tras el análisis de (Lopez y otros, 2004) se pudo constar que para la extensión de ArgoUML es necesario la definición del metamodelo del lenguaje a modelar y las restricciones del mismo.

Herramientas CASE para el desarrollo de aplicaciones educativas en la UCI

Con el apoyo de la muestra extraída de la población de 160 trabajos de diplomas que desarrollaron aplicaciones educativas; se identificaron cuáles han sido históricamente las herramientas CASE más utilizadas en el dominio específico que se estudia. Estas resultaron ser las que se muestran en la Tabla 2:

Tabla 2: Comparación entre las herramientas CASE para el desarrollo de aplicaciones educativas en la UCI

Herramientas	Usabilidad	Accesibilidad	Código	Licencia

			Abierto	
ArgoUML	Bajo	Alta	sí	Libre
Rational Rose	Bajo	Alta	no	Comercial
Visual Paradigm	Alto	Alta	no	Comercial, Edición libre solo para la comunidad
Eclipse	Alto	Alta	sí	Libre

A las herramientas analizadas se les incorpora dos nuevos indicadores para su evaluación:

- Usabilidad: pretende numerar cuales son las herramientas que más uso tienen dentro de la comunidad de desarrollo en la universidad.
- Accesibilidad: indica la facilidad de acceso que se tiene a las herramientas propuestas dentro de la universidad.

A partir del análisis realizado, se arrojaron los siguientes resultados:

- ArgoUML: según su página oficial (www. *argouml.tigris.org*), posee mecanismos para su extensión por su condición de código abierto. Sin embargo esta herramienta se encuentra en discontinuidad desde el año 2010. Además la documentación existente para su extensión es escasa.
- Rational Rose: por su condición de software privativo dificulta su posible extensión.
- Visual Paradigm: Según su página oficial (www.visual-paradigm.com) no permite modificación en el metamodelo que tiene establecido. Por lo que no es posible extenderlo para el tipo de solución que se pretende alcanzar en la presente investigación.
- Eclipse: es usado en la universidad como IDE. Aun así permite la actividad de modelado con UML 2.0 mediante plugins existentes para ello, según su página oficial (www.eclipse.org). Permite además la definición de metamodelos para la construcción de entornos gráficos de modelado, gracias al GMF (Graphical Modeling Framework).

Herramientas de Metamodelado.

Las herramientas CASE han evolucionado en otras más flexibles. Autores como (Zapata y otros, 2007) explica que estas herramientas surgieron para superar algunas limitaciones identificables de las tecnologías CASE.

(Pons y otros, 2010) Expresa que, gran parte de las herramientas de modelado que se utilizan hoy día, tanto aquellas que soportan al lenguaje UML, como las que brindan soporte a otros lenguajes de modelado, están basadas en una arquitectura de dos niveles (Figura 5). Los modelos son almacenados en archivos o en un repositorio cuyo esquema está programado y compilado dentro de la herramienta. Esto determina la clase de modelos que se pueden hacer y la manera en que se procesan pero no pueden ser modificados. La tecnología basada en metamodelos, por otra parte elimina esta limitación y permite lenguajes de modelado más flexibles. Esto se lleva a cabo adoptando la arquitectura de capas definida por el OMG, lo cual agrega un nivel sobre el nivel de los lenguajes de modelado, como se ve en la figura 5.

Figura 5: Arquitectura de las herramientas de modelado y las herramientas de metamodelado. [Tomado de (Pons y otros, 2010)]

Las tecnologías de metamodelado más recomendadas por (Pons y otros, 2010) son: Graphical Modeling Framework (GMF), para la herramienta Eclipse, DSL tools de Visual Studio y por último la herramienta MetaEdit+. Estas herramientas se han evaluado en cuanto a los indicadores: soporte, código abierto, licencia, usabilidad y accesibilidad, los cuales han sido descritos anteriormente.

Tabla 3: Establecimientos de valores para las herramientas

Herramientas	Soporte	Código Abierto	Licencia	Usabilidad	Accesibilidad
GMF (de Eclipse)	Multiplataforma	sí	Libre	Alto	Alto
MetaEdit+	Windows	no	Comercial	Bajo	Bajo

DSL tools (Integrado al SDK de Visual Studio)	Windows	no	Comercial	Medio	Alto

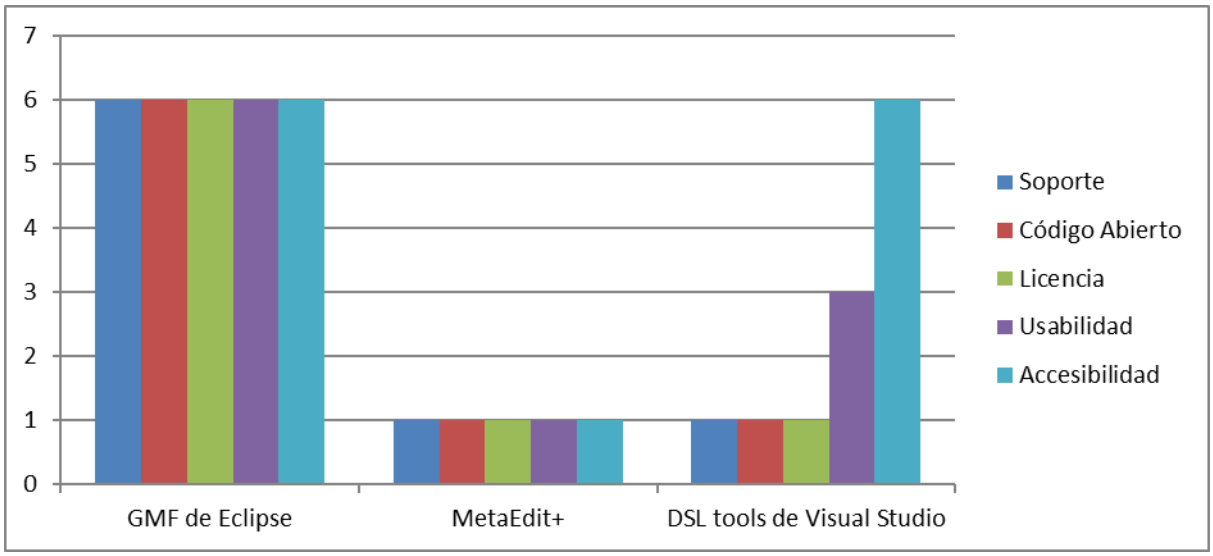

Figura 6: Comportamiento de los indicadores de las herramientas

El número 6 representa el valor alto, 3 es medio, y 1 es bajo. Donde el valor alto se le asigna al soporte que sea de multiplataforma, al código que sí sea abierto y a la licencia que sea libre. Mediante la gráfica se puede percibir que la herramienta con valores más altos es GMF de Eclipse, le sigue DSL tools de Visual Studio y por último la herramienta MetaEdit+. A partir del estudio realizado se arrojaron los siguientes resultados:

- Graphical Modeling Framework (GMF): es un framework de código abierto que permite construir editores gráficos, el cual está desarrollado para el entorno Eclipse. Está basado en los plugins framework de modelado de Eclipse (EMF, por sus siglas en inglés) y el framework gráfico de Eclipse (GEF por sus siglas en inglés). Los editores gráficos generados con GMF están completamente integrados a Eclipse y comparten las mismas características con otros editores como: vista overview, la posibilidad de exportar el diagrama como imagen, cambiar el color y la

fuente de los elementos del diagrama, hacer zoom animado del diagrama e imprimirlo. Estas funcionalidades están disponibles desde la barra de herramientas, como en cualquier otro editor.

- MetaEdit+: es una herramienta comercial, basada en repositorios, que usa una arquitectura cliente/servidor. Permite diseñar un lenguaje específico de dominio, para luego construir herramientas de modelado y generadores de código para ese lenguaje. Con esta herramienta el experto define un lenguaje de dominio específico a través de un metamodelo, que contiene los conceptos del dominio y las reglas que deben cumplir. Además del metamodelo, el experto debe especificar su notación gráfica y el código que se quiere generar a partir de las instancias de ese metamodelo. Esta definición se almacena en el repositorio, así el resto del equipo de desarrollo puede usarlo para hacer modelos con el lenguaje definido por el experto y generar automáticamente el código para esos modelos.
- DSL tools: es un conjunto de herramientas las cuales están integradas al programa Visual Studio. Este producto de Microsoft, contiene herramientas para definir e implementar tanto DSL, como generadores de código asociados. Las herramientas previstas en Visual Studio son denominadas colectivamente "DSL Tools" y utilizan técnicas de desarrollo específicas del dominio para crear e implementar DSL para Visual Studio.

Las herramientas de metamodelado proveen funcionalidades para dar soporte al proceso de metamodelado. Entre estas funcionalidades se puede mencionar que la herramienta debe permitir definir un metamodelo indicando cuáles son los elementos del lenguaje y sus relaciones, así como las propiedades para cada elemento. Además, debe permitir la especificación de reglas básicas, como por ejemplo, qué objetos se pueden conectar a través de cuál relación.

1.4 Metodología, Herramientas y Lenguajes a utilizar.

Herramientas y lenguajes de desarrollo a utilizar.

Para la selección de la herramienta de desarrollo de la aplicación, se contó con opciones como: Graphical Modeling Framework (GMF) para la herramienta eclipse, MetaEdit+ y DSL tools para Visual Studio. Gracias a la comparación realizada en el epígrafe anterior, se seleccionó para la realización de la herramienta al framework de desarrollo: **GMF**. Como consecuente se selecciona a **Eclipse** como Entorno de Desarrollo Integrado (IDE por sus siglas en inglés), ya que, para la utilización de GMF es necesario su

uso. Además según la página oficial de GMF para el desarrollo con el mismo, es necesario el uso de **java** como lenguaje de desarrollo.

El resultado del trabajo con GMF es un **plugin** para el IDE Eclipse que se mostrará como un editor gráfico de diagramas. A partir de cada plugin que se cree, se podrá generar un diagrama específico que responda a un metamodelo establecido. Un plugin es considerado como la unidad mínima de funcionalidad de Eclipse, que puede ser distribuida de manera separada. Estos consisten en código Java empaquetado en un archivo de Java (JAR), con algunos archivos de solo lectura, y otros recursos como imágenes, catálogo de mensajes, entre otros. Cada plugin posee un manifiesto del plugin, en el cual se describe su interconexión con otros plugins, o sea, se declara un número determinado de puntos de extensión y a su vez, se exponen las extensiones que se han realizado de otros puntos de extensión definidos por otros plugins. En el caso de la presente investigación se realizaran extensiones para el uso del metamodelo de UML implementado por eclipse. El manifiesto del plugin está representado por un par de ficheros: el MANIFEST.MF y el plugin.xml.

Tomando como base la tabla de comparación realizada para las herramientas de aplicaciones educativas en la UCI, se seleccionó como herramienta de modelado **Visual Paradigm**. En la tabla a pesar de ser Eclipse la herramienta de más altos valores, se seleccionó Visual Paradigm por ser de fácil manejo por los desarrolladores. Como consecuente se escoge a **UML 2.0** como lenguaje de modelado.

Metodología a utilizar

Para la selección de la metodología de desarrollo se definió primeramente el enfoque de la ingeniería de software por el que se optó en la investigación. A partir de esta selección, se analizó la metodología escogida perteneciente a dicho enfoque.

¿Enfoque ágil, pesado o híbrido?

Tras un análisis de la literatura se pudo constar que existen tres enfoques en la forma en que se desarrolla la ingeniería de software en un proyecto: enfoque ágil, pesado e híbrido. A continuación se presentan las principales características de estos enfoques descritas por (Ciudad, 2011).

Un enfoque pesado presenta las siguientes características:

- Elevada documentación: este rasgo permite que la comunicación entre los miembros del equipo sea a través de la documentación, y no necesariamente de manera directa.

- Procesos rígidos y secuenciales: procesos los cuales necesitan necesariamente de las salidas del procedimiento anterior para comenzar.
- Alta modularidad y trabajo por separado.
- Equipos grandes y con pocas necesidades de comunicación.
- Lento ante la resolución de problemas confusos.
- Alta jerarquización de roles.

Por otra parte un ágil presenta las siguientes características, las cuales descansan en elementos subjetivos como las características del equipo de desarrollo:

- Competencia: el equipo de desarrollo debe tener capacidades y habilidades altamente desarrolladas para poder acometer la tarea de construcción del software.
- Enfoque común: todos los miembros del equipo de software deben manejar un mismo lenguaje y estén de acuerdo en las ideas, que desde el punto de vista tecnológico, se van a manejar para la solución de la problemática a la que se enfrentan.
- Colaboración: debe haber amplias habilidades y condiciones de colaboración entre los miembros del equipo, para disminuir de esta manera las amplias etapas de comunicación y planificación
- Habilidad para la toma de decisiones: capacidad del equipo de tomar las decisiones correctas para el desarrollo del software de manera eficaz y eficiente.
- Capacidad de resolución de problemas confusos: es la manera de desarrollar un software en una situación donde el problema no está bien definido, así como sus requisitos.
- Organización propia: dinámica de funcionamiento eficiente y eficaz.

En los últimos años ha venido surgiendo lo que se conoce como el enfoque híbrido el cual intenta introducir entre sus características las mejores de los enfoques anteriores:

- Mantener la documentación.
- Reforzar el mantenimiento y la escalabilidad.
- Garantizar la interoperabilidad.
- Desarrollos rápidos (componentes, aspectos, servicios)
- Equipos grandes y altamente distribuidos.
- Solución de problemas confusos.

Los enfoques pesados se usan fundamentalmente en equipos de trabajos con gran cantidad de personal, para procesos rígidos y secuenciales, donde no importa el coste en términos de tiempo. Estas

características no se adecuan a las de la presente investigación, ya que el tiempo para el desarrollo es de seis meses, y la cantidad de personal es de dos miembros. Por otra parte el enfoque ágil descuida la documentación para actividades de modelado y construcción, generando problemas en la sostenibilidad, el mantenimiento y en la escalabilidad de las aplicaciones. De esta manera se escogió el enfoque híbrido, solo con la especificación de que es para un equipo pequeño, ya que elimina las debilidades mencionadas anteriormente combinando las mejores prácticas de los enfoques anteriores.

Metodología OpenUP

Una vez seleccionado el enfoque, se seleccionó una metodología acorde con el mismo.

Para la selección de la metodología a través de la cual se desarrolló la investigación, se tuvieron en cuenta los siguientes aspectos:

- Tiempo: según (Pressman, 2005) esta es una de las variables fundamentales para la elección de la metodología, pues es a partir de la medida de la misma, es que se distribuirán las actividades del PDS en el tiempo. El tiempo de desarrollo del sistema es de seis meses.
- Nivel de formalidad: ya que la presente investigación se realizó como un acto de diploma para optar por el título de ingeniero en ciencias informáticas, la misma requiere de la entrega de una serie de artefactos necesarios para cumplimiento de sus objetivos. De esta manera no importa el tamaño del equipo para desarrollar la documentación necesaria para este tipo de trabajo.
- Nivel de complejidad: se realizó un análisis bibliográfico en las tesis de grado localizadas en la base de datos de la universidad. Este arrojó que hasta el momento, en la universidad, no se han realizado trabajos similares. Por lo que para su desarrollo no será posible contar con soluciones en el mismo campo de acción en dicha universidad. Por cuanto se procede a clasificar la investigación con un nivel de complejidad elevado.

A partir de las características descritas se eligió OpenUP como metodología de desarrollo. Esta, como una variante ágil del Proceso Unificado de Desarrollo (RUP por sus siglas en inglés), está pensada según (Pollice y otros, 2003), para equipos de desarrollo pequeños, con tiempo de desarrollo corto y provee las documentaciones necesarias para esta investigación. Dicha metodología adopta como principios colaborar para alinear intereses y para compartir conocimiento. Se centra en balancear las prioridades para maximizar las necesidades de los interesados.

OpenUP estructura el ciclo de vida en 4 fases: Inicio, Elaboración, Construcción y Transición. El ciclo de vida provee la visibilidad y los puntos de decisión tanto para los interesados (stakeholders) como

miembros del equipo, permitiendo una vigilancia efectiva del proceso de desarrollo y facilitando la toma de decisiones apropiadas en cada etapa.

Conclusiones Parciales

A partir de lo previamente visto en el capítulo se concluyó que:

- Las nuevas especificaciones que trae consigo la versión 2.0 del lenguaje ApEM-L no permite su modelado de manera eficiente en las herramientas CASE existentes para UML.
- La elaboración de una herramienta CASE para el modelado de un determinado lenguaje necesita del establecimiento del respectivo metamodelo de dicho lenguaje; así como el establecimiento de las restricciones para este lenguaje.
- Para el correcto modelado con ApEM-L 2.0 será necesario una herramienta que permita la interpretación de su metamodelo, así como la definición de sus estereotipos decorativos.
- A diferencia de las herramientas de modelado, una herramienta basada en metamodelos permite al usuario acceder y modificar las especificaciones del lenguaje lo que permite la construcción de una herramienta para el modelado de un lenguaje de modelado más flexible.
- Con el empleo del desarrollo dirigido por modelos, el ciclo de vida de un proyecto de software, se reduce en tiempo y esfuerzo en el desarrollo del mismo.
- Para el desarrollo de la solución, y teniendo en cuenta el entorno, se seleccionaron un grupo de tecnologías y las herramientas. GMF como framework de desarrollo, Eclipse como IDE, UML 2.0 como lenguaje de modelado, entre otros.

CAPÍTULO II: CARACTERÍSTICAS DE LA COLECCIÓN DE PLUGINS

En el capítulo II se abordó, acerca de las características de las herramientas CASE que desarrollan aplicaciones educativas en la UCI. Además de proponer una solución al problema planteado en la investigación. Se definió también un modelo del dominio, los requisitos funcionales y no funcionales y el diagrama de flujo de la herramienta realizada.

2.1 Características de la actividad de modelado en el desarrollo de aplicaciones educativas en la UCI, mediante herramientas CASE

La UCI, siendo un gran exponente del desarrollo productivo en la industria de software en Cuba, presenta centros de desarrollo dedicados a la creación de aplicaciones educativas. Entre ellos se encuentra: el Centro de Tecnologías para la Formación (FORTES) y Centro de Innovación y Calidad en la Educación (CICE). El desarrollo de aplicaciones educativas en dichos centros se realiza a partir de los estándares definidos para el PDS. El desarrollo de la presente investigación solo centrará el análisis en la actividad de modelado de estos centros; con el objetivo de caracterizar el proceso de generación de diagramas para el desarrollo de aplicaciones educativas en la UCI.

Para el desarrollo de la segunda tarea de la investigación se escogió una muestra de trabajos de diploma desarrollados en estos centros y otros relacionados con el tema, en aras de analizar la actividad de modelado que desarrollaron. Además, se hicieron entrevistas a algunos de los desarrolladores de estos trabajos, como: al Ing. Norlen Rosales Velázquez, desarrollador de una de las tesis de grado analizadas y a la Ing. Mairelis Gari Maribona, analista principal del centro FORTES en la UCI.

Se escogió una muestra compuesta por ciento doce (112) tesis de una población de ciento sesenta (160) trabajos de diploma que hubieran desarrollado aplicaciones educativas a partir del año 2010. Se utilizó la aplicación Stast, recomendada por (Sampieri y otros, 2006), para el cálculo de la muestra, con un error de un 5% y un nivel de confianza de 95%. Después de un análisis de cada elemento de la muestra se llegaron a los siguientes resultados parciales:

- Un 20% de los trabajos usaron como lenguaje de modelado ApEM-L 1.0 y el otro 80% usó UML.
- Un 75% de los trabajos usó como herramienta CASE Visual Paradigm for UML, otro 22% Rational Rose y otro 3% ArgoUML.

A partir de estos resultados se analizaron las herramientas usadas para modelar con los lenguajes mencionados, bajo los indicadores definidos en la investigación para el nivel de completitud de la variable "herramientas CASE para la actividad de modelado":

Tabla 4: Comparación entre las herramientas que modela con ApEM-L y UML

Indicador	Análisis	Valor
Cantidad de componentes de la herramienta que representan con completitud diagramas que modelan la estructura lógica de la presentación del sistema.	Según UML este tipo de diagrama no está definido dentro de sus especificaciones. Por lo que los componentes para su realización bajo UML no existen en las herramientas mencionadas. Los estereotipos pueden ser definidos mediante los mecanismos de extensión de UML en las herramientas referidas anteriormente. Sin embargo las restricciones definidas para estos no son interpretadas por la herramienta, por lo que existe la posibilidad de una incorrecta realización de los diagramas.	Bajo
Cantidad de componentes de la herramienta que representan con completitud diagramas que modelan la estructura lógica de la navegación del sistema.	Según UML este tipo de diagrama no está definido dentro de sus especificaciones. Por lo que los componentes para su realización bajo UML no existen en las herramientas mencionadas. Los estereotipos pueden ser definidos mediante los mecanismos de extensión de UML en las herramientas referidas anteriormente. Sin embargo las restricciones definidas para estos no son interpretadas por la herramienta, por lo que existe la posibilidad de una incorrecta realización de los diagramas.	Bajo
Cantidad de componentes de la herramienta que representan con completitud los diagramas de casos de uso del sistema.	No existe la especificación de usuarios determinados (ej. profesor, estudiante, tutor, etc.) que se benefician de las funcionalidades de carácter educativo del sistema. Los estereotipos pueden ser definidos mediante los mecanismos de extensión de UML en las herramientas mencionadas. Sin embargo las restricciones definidas para estos no son interpretadas por la herramienta, por lo que existe la posibilidad de una incorrecta realización de los diagramas.	Medio
Cantidad de componentes de la herramienta que representan con completitud los diagramas de clases del diseño del sistema bajo el patrón MVC-E.	Este tipo de diagrama es modelado en estas herramientas, por lo que su metamodelo soporta su modelado. En cuanto a su representación mediante estereotipos, estos se pueden mostrar mediante estereotipos descriptivos. Por lo que para su uso es necesario basarse en la descripción de la pragmática del estereotipo con el objetivo de usar e interpretar correctamente el mismo. Por otra parte, las restricciones que se definen con este estereotipo no son interpretadas por las herramientas, por lo que es posible un incorrecto modelado de los mismos.	Medio

La anterior tabla mostró como resultado final, que el nivel de completitud de las herramientas CASE usadas en la UCI para el modelado de aplicaciones educativas es insuficiente. De esta manera, los diagramas resultantes de esta actividad en estas herramientas, no permiten representar todas las características de las aplicaciones que modelan.

También es necesario decir que los lenguajes con los que se modela las aplicaciones educativas presenta una serie de insuficiencias las cuales son analizadas a continuación:

(Herrera, 2014a) Expresa que primeramente UML es un lenguaje de modelado visual de propósito general, por lo que su semántica y sintaxis, en algunos casos, no logra representar las características de un determinado dominio Por ende, los diagramas que se generaron con las herramientas CASE, en las investigaciones donde se usó UML, tampoco logran representar todas las características del dominio de las aplicaciones educativas cubanas.

Por otro lado, ApEM-L 1.0 presenta insuficiencias, como las detectadas por (Herrera, 2014a):

- No todos los artefactos modelan con exactitud todas las características de las aplicaciones, lo que evidencia una insuficiente representatividad en los diagramas resultantes.
- El alcance de estos artefactos resultantes del modelado es limitado debido al uso de notaciones y conceptos de modelado no concebidos por el lenguaje.
- El Diagrama de Estructura de Presentación (DEP) como consecuencia de su estructura actual trae consigo una interpretación del diagrama que no está en correspondencia con su propósito real.
- Tanto el DEP como el Diagrama de Estructura de Navegación (DEN) presentan errores debido al uso incorrecto e insuficiente de los conceptos de modelado establecidos en el lenguaje.

Por estas razones, las herramientas que modelan con estos lenguajes, no representan con exactitud las características reales de las aplicaciones que intentan describir.

En la entrevista realizada a la analista principal de FORTES se concluyó que actualmente en dicho centro se usa como lenguaje de modelado UML. De esta manera los diagramas generados en este centro no logran expresar todas las características del dominio en el que se desarrollan y documentadas en los expedientes de los proyectos.

A partir de todo lo visto en el epígrafe y de la operacionalización de la segunda variable, se concluye que los diagramas generados bajo los lenguajes usados poseen un nivel insuficiente de precisión y representatividad; (Ver Anexo I). Entiéndase por los conceptos mencionados, lo siguiente:

- Precisión: capacidad que debe poseer el modelo de proveer una representación legítima de las características del sistema en interés.
- Representatividad: capacidad del artefacto y sus componentes de representar las características del software según el propósito por el cual fueron creados.

2.2 Modelo conceptual del dominio

En el año 2006 se crea el método de modelado para organizaciones con flujos de información complejos y difusos: DoMet, creado por el Dr.C Febe Angel Ciudad Ricardo y el Ing. Yosnel Herrera Martínez. El resultado de dicho método es un eficiente diagrama conceptual del modelo de dominio. Este utiliza técnicas más actuales de modelado a partir de las teorías existentes que abordan la estructura y organización de la información en los sistemas.(Ciudad y otros, 2006)

En el capítulo anterior se justifica la selección de la metodología escogida para la realización del presente trabajo, la cual es OpenUP. Dicha metodología expresa, según autores como (Santiago y otros, 2013), que aunque sea una metodología con un enfoque ágil, está basada en la metodología RUP y por lo tanto, comparten las mismas prácticas que subyacen por debajo del flujo de trabajo y los roles de OpenUP. Dichas prácticas se exponen en la figura a continuación.

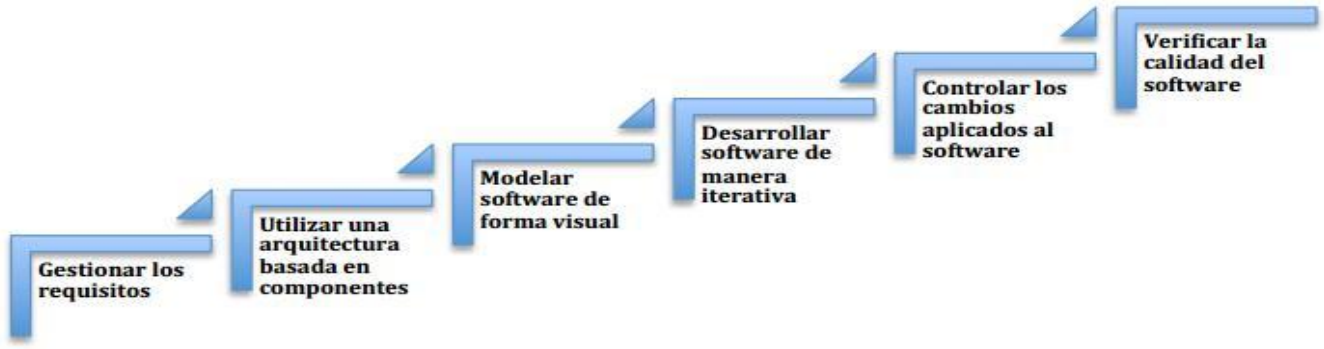

Figura 7: Prácticas de OpenUP. [Tomado de (Santiago y otros, 2013)]

A partir de lo planteado y considerando los principios de la ingeniería de software, se toma la decisión de utilizar como apoyo al método DoMet. Para así construir un modelo conceptual del dominio, a pesar que la metodología seleccionada no realice este paso.

Según (Ciudad y otros, 2006) el método DoMet, cuenta con algunas actividades como:

- Determinación de la estructura organizacional: cuyo artefacto resultante es el organigrama.
- Determinación de los eventos principales que ocurren: donde el artefacto resultante es la descripción de los eventos principales.
- Determinar informaciones que se manejan: los artefactos resultantes son: el registro de informaciones, glosario de conceptos y el registro de objetos.

- Determinar quién o quienes participan y sus roles: da lugar al artefacto registro de roles.
- Definición de clases: como artefacto resultante tiene a las clases del dominio.
- Relacionar clases del dominio: quedando como artefactos resultantes, modelo de dominio y el diagrama de clases del dominio.

A partir de lo planteado, para la realización del diagrama conceptual, se resuelven los siguientes artefactos:

- Objetos reales: herramienta CASE, los tipos de herramientas CASE utilizadas (Visual Paradigm, Rational Rose, y ArgoUML), diagramas, símbolos.
- Se determina un glosario de conceptos con términos como: proceso de modelado, modelo, el metamodelo, el lenguaje de modelado, el lenguaje de modelado de dominio específico, los tipos de lenguajes de modelado usados (UML, UML 2.0, UWE, ApEM-L 1.5, OMMMA-L, ApEM-L 2.0).
- Registro de roles: especialista Informático.
- Clases del dominio: se procedió a convertir las clases del dominio a: los objetos reales, los conceptos y al rol.
- Modelo del dominio: al establecer las relaciones entre las clases se obtiene el diagrama siguiente:

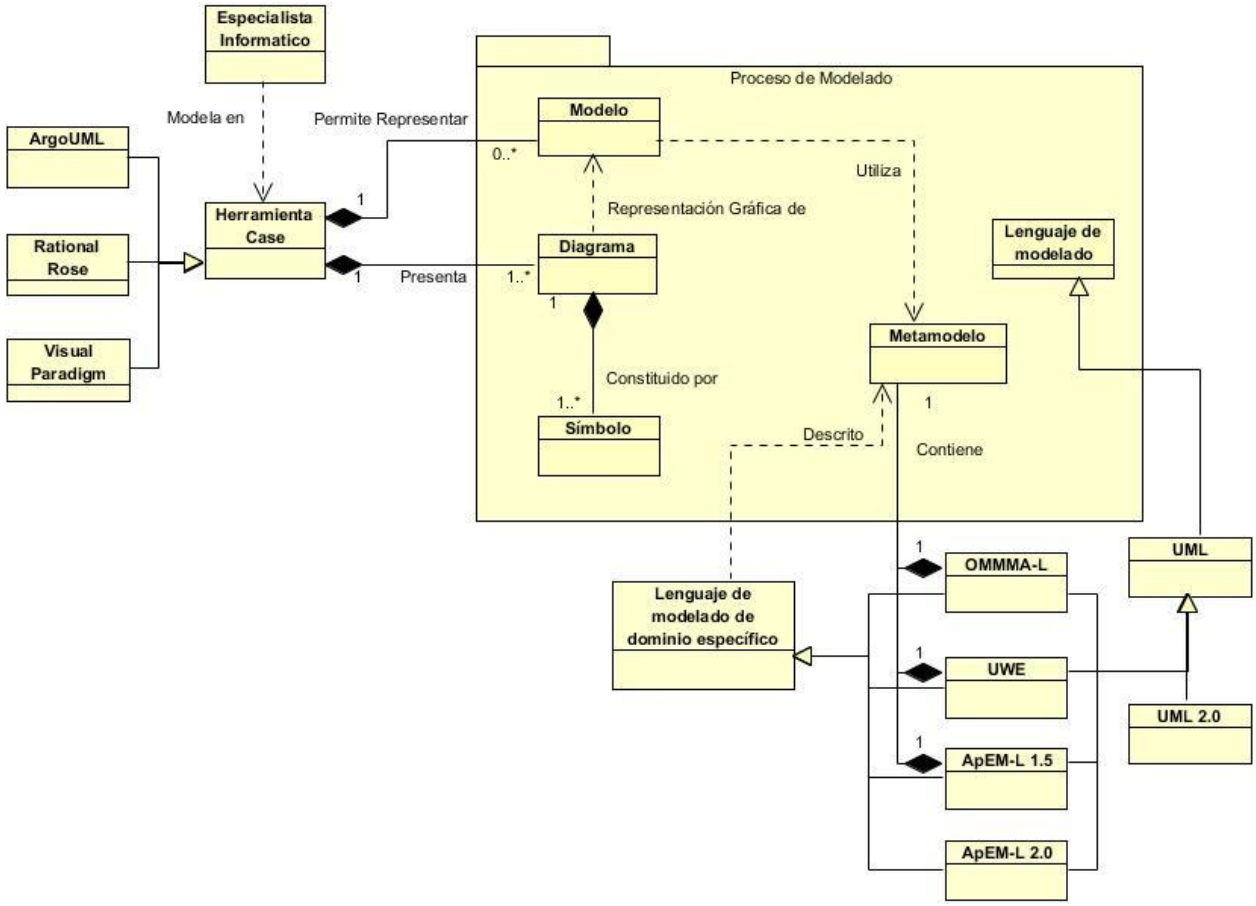

Figura 8: Diagrama Conceptual del Dominio

Definición de los conceptos relacionados con el modelo del dominio:

- Especialista informático: es la persona que realizara la actividad de modelado, apoyándose en la herramienta CASE.
- Herramienta CASE: herramienta de modelado que se utiliza para poder realizar el proceso de generación y representación de diagramas.
- Visual Paradigm: Visual Paradigm for UML es una de las herramientas CASE que se utiliza para realizar la actividad de modelado. Esta herramienta utiliza UML como lenguaje de modelado, además de que permite dibujar todos los tipos de diagramas, generar código desde diagramas y realizar la documentación.

- Rational Rose: herramienta CASE que se utiliza para realizar la actividad de modelado. Esta utiliza el lenguaje de modelado UML, además de que permite dibujar todos los tipos de diagramas, generar código desde diagramas y realizar la documentación.
- ArgoUML: herramienta para modelar sistemas, mediante el cual se realizan diseños en UML. Esta herramienta puede crear la mayoría de los diagramas estándares de UML.
- Diagramas: es la representación gráfica de los modelos, el cual está compuesto por nodos (que son los elementos que denotan objetos, conceptos, etc.) y arcos que serían las relaciones entre estos nodos.
- Símbolos: son aquellos artefactos, que conforman los diagramas como: las relaciones (arcos) y los elementos (nodos).
- Proceso de modelado: el proceso de modelado, es aquel que tiene como objetivo la construcción de un modelo, el cual se representa a partir de diagramas con la ayuda de un lenguaje de modelado.
- Modelos: un modelo es una abstracción del sistema, especificando el sistema de modelado desde cierto punto de vista y en un determinado nivel de abstracción.
- Metamodelos: el metamodelo es un nivel de abstracción del sistema más alto que el modelo (describe al modelo).
- Lenguaje de modelado: sistema que posibilita la representación gráfica de modelos para representar el comportamiento de programas.
- Lenguaje de modelado de dominio específico: es el lenguaje de modelado que está especialmente diseñado para desarrollar software restringido a un dominio determinado.
- UML: es uno de los lenguajes de modelado más utilizados por la industria del software para realizar el modelado de los sistemas, pero a pesar de que presente un gran número de ventajas, no posee la capacidad de detallar todo tipo de dominio, sobre todo aquellos lenguajes que presentan un alto nivel de detalles.
- UML 2.0: la versión 2.0 de UML presenta todas las ventajas de la antigua versión y según (Berkenkötter, 2010) presenta además: una sintaxis nueva, nueva semántica y restricciones e información adicional como la transformación a normas.
- ApEM-L 1.5: ApEM-L 1.5 es un lenguaje de modelado de dominio específico, para el desarrollo de aplicaciones educativas y de multimedia; el cual es una extensión del lenguaje UML.

- UWE: es un lenguaje de modelado de dominio específico, para el desarrollo de aplicaciones web, orientado a objetos; el cual es una extensión del lenguaje UML.
- OMMMA-L: es un lenguaje de modelado de dominio específico, para el desarrollo de las aplicaciones de multimedia; el cual es una extensión del lenguaje UML.
- ApEM-L 2.0: la versión 2.0 del lenguaje ApEM-L es una versión superior ya que gana en representatividad, completitud, y formalidad en el lenguaje. A partir de la definición textual de sus vistas y la agregación de nuevos conceptos.

2.3 Especificación de los Requisitos del sistema

El flujo de trabajo de requisitos ayuda a formar y mantener el acuerdo con los clientes o con los interesados en la aplicación. Proporciona además, a los desarrolladores del sistema una mejor comprensión de los requisitos y define las fronteras del sistema, (Somerville, 2005). A partir de los métodos empíricos realizados como la observación y la entrevista, y de un estudio del mercado, basado en herramientas como Visual Paradigm y GMF se identificaron los siguientes requisitos funcionales y no funcionales.

Requisitos Funcionales

Según (Somerville, 2005), los requisitos funcionales (RF) son aquellas capacidades o condiciones que el sistema debe cumplir. Expresan la naturaleza del funcionamiento del sistema, cómo interactúa el sistema con su entorno y cuáles van a ser su estado y funcionamiento. Estos requerimientos deben:

- Estar redactados de tal forma que sean comprensibles para usuarios sin conocimientos técnicos avanzados de Informática.
- Especificar el comportamiento externo del sistema y evitar, en la medida de lo posible, establecer características de su diseño.

A partir de lo planteado, para el desarrollo de la propuesta, se han definido los siguientes requisitos funcionales de acuerdo a las características que presentan los plugins.

Tabla 5: Listado de los requisitos funcionales de la colección plugins a desarrollar.

Código	Descripción de la función
RF 1	Conformar la paleta de herramientas.
RF 2	Dibujar los diagramas del lenguaje ApEM-L 2.0.
RF 2.1	Dibujar el diagrama de Casos de Uso de ApEM-L 2.0.
RF 2.2	Dibujar el diagrama de estructura de presentación.
RF 2.3	Dibujar el diagrama de estructura de navegación.
RF 2.4	Dibujar el diagrama de vista de estática.
RF 3	Establecer la propiedad del elemento dibujado.
RF 4	Gestionar error de modelado.
RF 5	Realizar menú auxiliar para cada elemento.

Descripción de los requisitos funcionales

Conformar la paleta de herramientas: la interfaz de usuario (IU) que se construyó a partir de los plugins debe ser capaz de mostrar una paleta de herramientas. Dicha paleta debe contener para cada diagrama que se desee modelar, los elementos necesarios para su creación.

Dibujar los diagramas del lenguaje ApEM-L 2.0: la colección de plugins debe ser capaz de dibujar en un lienzo, todos los elementos contenidos en la paleta. El elemento que se dibuje debe corresponder a su descripción en (Herrera, 2014). Cada elemento solo debe crearse en el plugin correspondiente a su diagrama.

Establecer la propiedad del elemento dibujado: la colección de plugins debe permitir la visualización de las propiedades de cada uno de los elementos que se creen en la IU. Además debe permitir que las propiedades se modifiquen y se valide su modificación.

Gestionar error de modelado: la colección de plugins debe permitir la correcta validación del diagrama que se cree. Se debe validar que se cumplan todas las restricciones del metamodelo, y en caso de que existan errores, mostrarlos correctamente en la IU.

Realizar menú auxiliar para cada elemento: la colección de plugins debe permitir la creación de un elemento, a partir de menús auxiliares que se muestren dinámicamente mientras se modela en el lienzo.

Requisitos No Funcionales

Con apoyo de (Somerville, 2005) se puede decir que los requisitos no funcionales (RNF) son aquellas propiedades o cualidades que debe estar presente en el sistema. Estas propiedades o cualidades se refieren a las características que hacen al producto atractivo, usable, rápido y/o confiable. Normalmente estos están vinculados a los requisitos funcionales. Es decir, una vez que se conoce lo que el sistema debe hacer, se puede determinar cómo ha de comportarse y qué cualidades o propiedades debe tener. En la siguiente tabla se muestran los RNF.

Tabla 6: Listado de los requisitos no funcionales de la colección plugins a desarrollar

Código	*Tipo*	*Descripción*
RNF 1	***Usabilidad***	Facilidad de uso por parte de los usuarios: El sistema debe presentar una interfaz que permita la fácil interacción con el mismo y llegar de manera rápida y efectiva a la información buscada. Debe ser una interfaz de manejo cómodo.
RNF2	***Usabilidad***	Menús: la solución propuesta debe presentar una serie de menús tanto laterales como en forma de barra de iconos flotantes e internos que permitan el acceso rápido a la información por parte de los usuarios, aprovechando así las potencialidades de estas estructuras.
RNF 3	***Software cliente***	Se debe tener la herramienta Eclipse, y el Graphic Modeling Framework (GMF) instalados.
RNF 4	***Software cliente***	Se debe tener la máquina virtual de Java Open JDK de la versión 6 o alguna versión superior, instalada.
RNF 5	***Hardware cliente***	Un mínimo de 385 MB de espacio en disco.
RNF 6	***Hardware cliente***	Microprocesador Intel Pentium IIII con 1, 0 GHz o superior.
RNF 7	***Hardware cliente***	Mínimo 512 MB de RAM (Random Access Memory, por sus siglas en inglés), pero se recomienda 1,0 GB.
RNF 8	***Portabilidad***	Los plugins una vez integrados a la herramienta Eclipse, podrán ser instalados y ejecutados en diferentes sistemas operativos, por ser Eclipse una herramienta Multiplataforma.
RNF 10	***Restricciones del diseño y la implementación.***	Se hace uso de la herramienta Eclipse y del framework GMF.
RNF 11	***Restricciones del diseño y la implementación.***	El lenguaje de programación que será usado para la implementación es Java.

2.4 Propuesta de solución

Sobre la base de lo que se plateó en los fundamentos referentes al trabajo con GMF, para la solución, se planteó la construcción de una colección de plugins que permitiera el modelado de cada uno de los diagramas modificados para ApEM-L 2.0. Dicha colección dará apoyo a la generación de diagramas en la actividad de modelado de los proyectos productivos encargados del desarrollo de aplicaciones educativas en la UCI, para el lenguaje ApEM-L 2.0. Para el desarrollo de la propuesta se hizo uso de las herramientas mencionadas en el capítulo anterior. En cada uno de los plugins de la colección se definió el metamodelo de diagrama a realizar, además de sus reglas y estereotipos.

Diagrama de flujo

La presente investigación utiliza como método de modelado el de Elementos orientados al flujo, dado que proporciona un conocimiento adicional de los requisitos y del flujo del sistema, pues se describen las transformaciones que sufre la información de entrada al sistema y la salida que genera el procesamiento. Para ello, se utiliza el diagrama de flujo, que muestra la manera en que una entrada se transforma en una salida conforme los objetos de datos se mueven a través del sistema. Este diagrama según (FUNDIBEQ, 2011) es una representación gráfica de la secuencia de pasos que se realizan para obtener cierto resultado. Al igual que el diagrama conceptual expuesto anteriormente, el diagrama de flujo no forma parte de las actividades de la metodología seleccionada. Aun así, se confecciona para la predicción de pasos a realizar en la elaboración del software. El diagrama de flujo debe expresar fielmente la actividad en estudio. Este tiene dos principales características:

- Capacidad de Comunicación: la cual permite conocer los conocimientos individuales sobre un proceso y facilita la mejor comprensión global del mismo;
- Claridad: es la característica que proporciona la información sobre los procesos de forma clara, ordenada y concisa.

Descripción general

El flujo general de la información en el sistema comienza con dos eventos principales. Primeramente mediante el evento on mouse move. Este es el que se produce cuando se desplaza por la superficie del

lienzo el puntero del mouse, luego de haber seleccionado un elemento en la paleta de herramientas. Posteriormente se espera el evento on mouse click, que es el que se produce cuando se hace click con el mouse luego de producido el evento on mouse move.

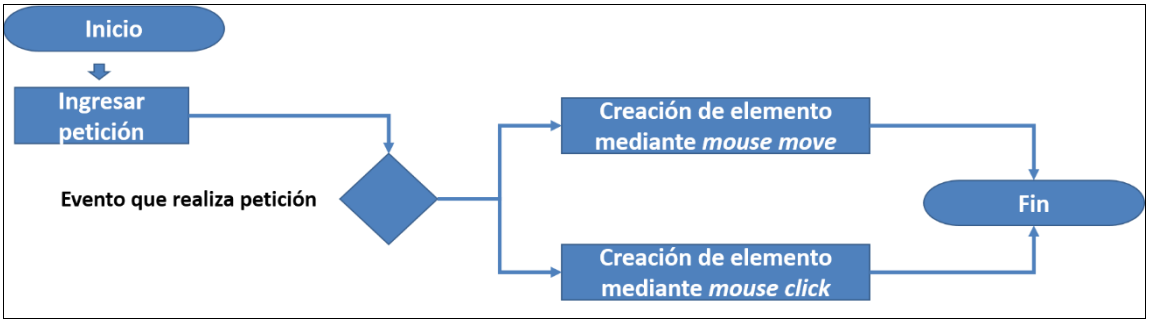

Figura 9: Diagrama de flujo general para la creación de un elemento

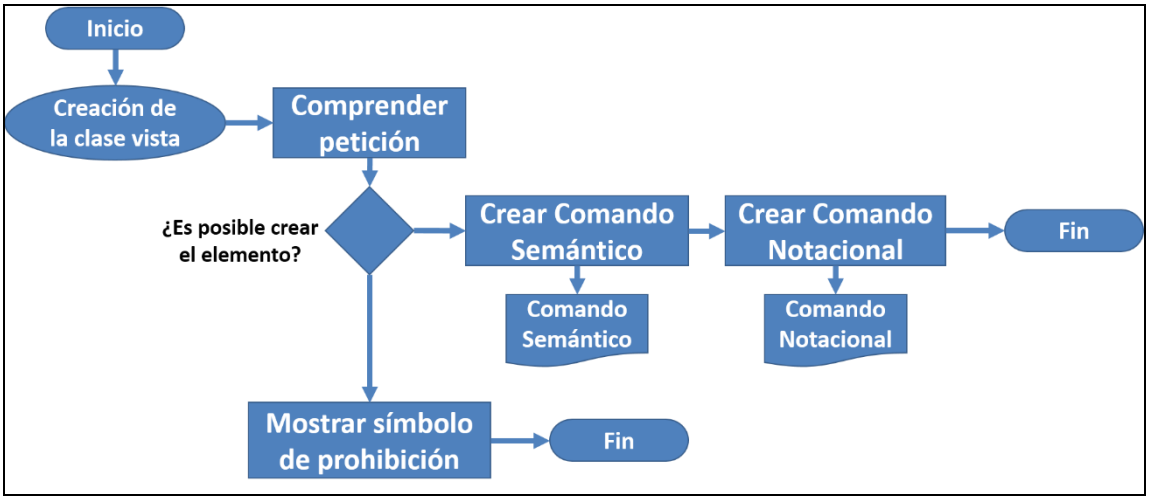

Figura 10: Diagrama de flujo para actividad del evento on mouse move

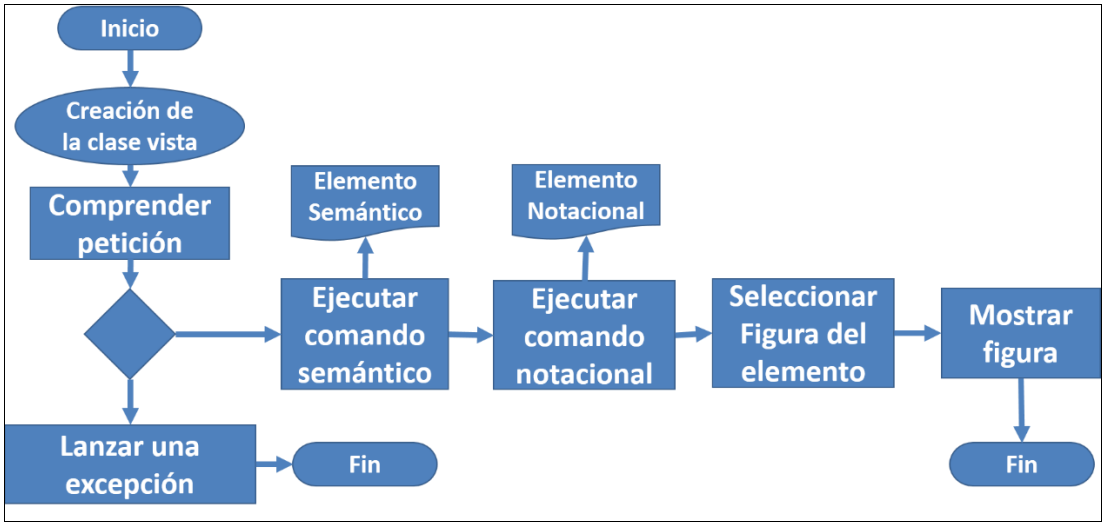

Figura 11: Diagrama de flujo para actividad del evento on mouse click.

Conclusiones Parciales

En el presente capitulo se concluyó que:

- Tras el análisis de las herramientas CASE usadas en la UCI para el modelado de aplicaciones educativas, se pudo concluir que el nivel de completitud de las mismas es insuficiente. De esta manera, los diagramas resultantes de este proceso, en estas herramientas, no logran definir todas las características de las aplicaciones que modelan.
- Gracias análisis del entorno organizacional, se esclarece y visualiza que no existe una herramienta que cuente con un metamodelo para el lenguaje ApEM-L 2.0. Esta situación hace que imposible su modelado, por lo que se necesita una herramienta que lo modele.
- Debido a un estudio de mercado y un análisis con el cliente, se lograron definir 8 requisitos funcionales y 11 no funcionales. A partir de los mismos se construyó la propuesta de solución.

CAPÍTULO III: CONSTRUCCIÓN DE LA SOLUCIÓN PROPUESTA

El presente capítulo abordó como se desarrolló la aplicación. A continuación se mostrará la forma en fue construido el sistema y que patrones son guía para su confección. Así como los diagramas de clases del diseño, los diagramas de secuencia y los diagramas de componentes utilizados para el desarrollo de la herramienta.

3.1 Patrón arquitectónico: Modelo-Vista-Controlador

GMF es la herramienta utilizada para la solución del presente trabajo. Para definir la arquitectura de la solución propuesta, resulta fundamental regirse por los elementos arquitectónicos definidos en dicha herramienta. Esta, según la aplicación de ayuda de Eclipse, propone una arquitectura basada en el patrón arquitectónico modelo vista controlador (MVC), a través del cual es posible separar el modelado del dominio, la presentación y las acciones realizadas por el usuario, en tres clases diferentes, según (Pavón, 2009).

- **Modelo:** Es el que administra el comportamiento del dominio de la aplicación, empleando para ello las acciones contenidas en el framework. Estas acciones están asociadas a los elementos del modelo y contemplan la creación de estereotipos, atributos, operaciones, clases y sus relaciones, entre otras.
- **Vista:** Es la responsable de la visualización de la información, a través de los diferentes tipos de diagramas que contiene el lenguaje ApEM-L y de las interfaces mostradas al especialista informático. Esto se evidencia en el módulo para el diseño, del diagrama de flujo propuesto, en el capítulo anterior.
- **Controlador:** Es el responsable de interpretar los eventos producidos por el usuario, informando al modelo y/o a la vista para que cambien según resulte apropiado. Ejemplos de esto son las acciones implementadas en el módulo controlador de elementos del diagrama de flujo propuesto, el cual captura los elementos del modelo (Modelo) y a partir de ellos realizan acciones sobre los diagramas (Vista) y viceversa.

3.2 Diseño y construcción del sistema

Metamodelo del sistema a realizar

Para la construcción del metamodelo, se toma como apoyo el metamodelo propuesto por (Herrera, 2014b), para la creación de los nuevos diagramas del lenguaje ApEM-L 2.0: DEP (Diagrama Entidad Relación), el DEN (Diagrama Entidad Navegación), el DCU (Diagrama de Casos de Uso) y DVE (Diagrama de Vista Estática).

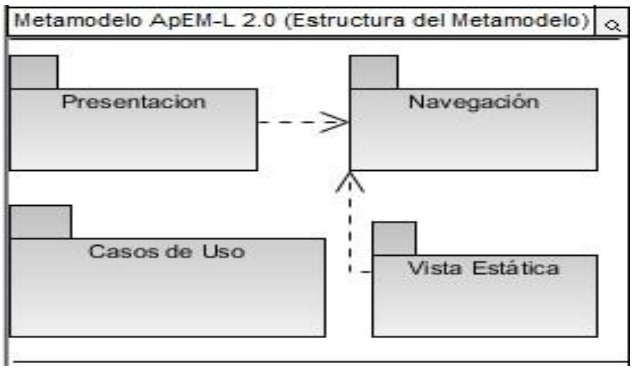

Figura 12: Estructura del metamodelo de ApEM-L 2.0 [Tomado de(Herrera, 2014a)]

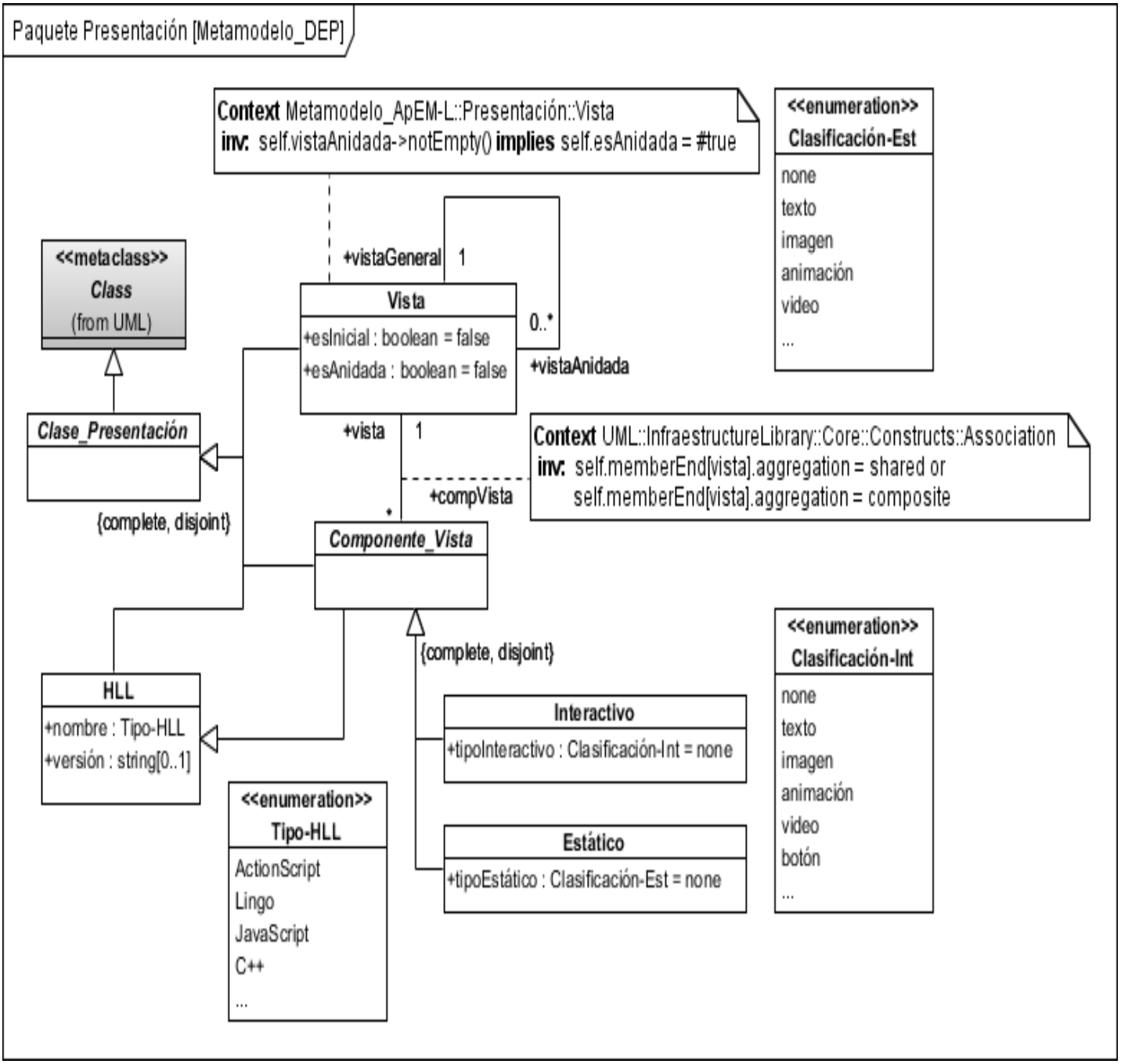

Figura 13: Metamodelo DEP de ApEM-L correspondiente al paquete presentación [Tomado de (Herrera, 2014a)]

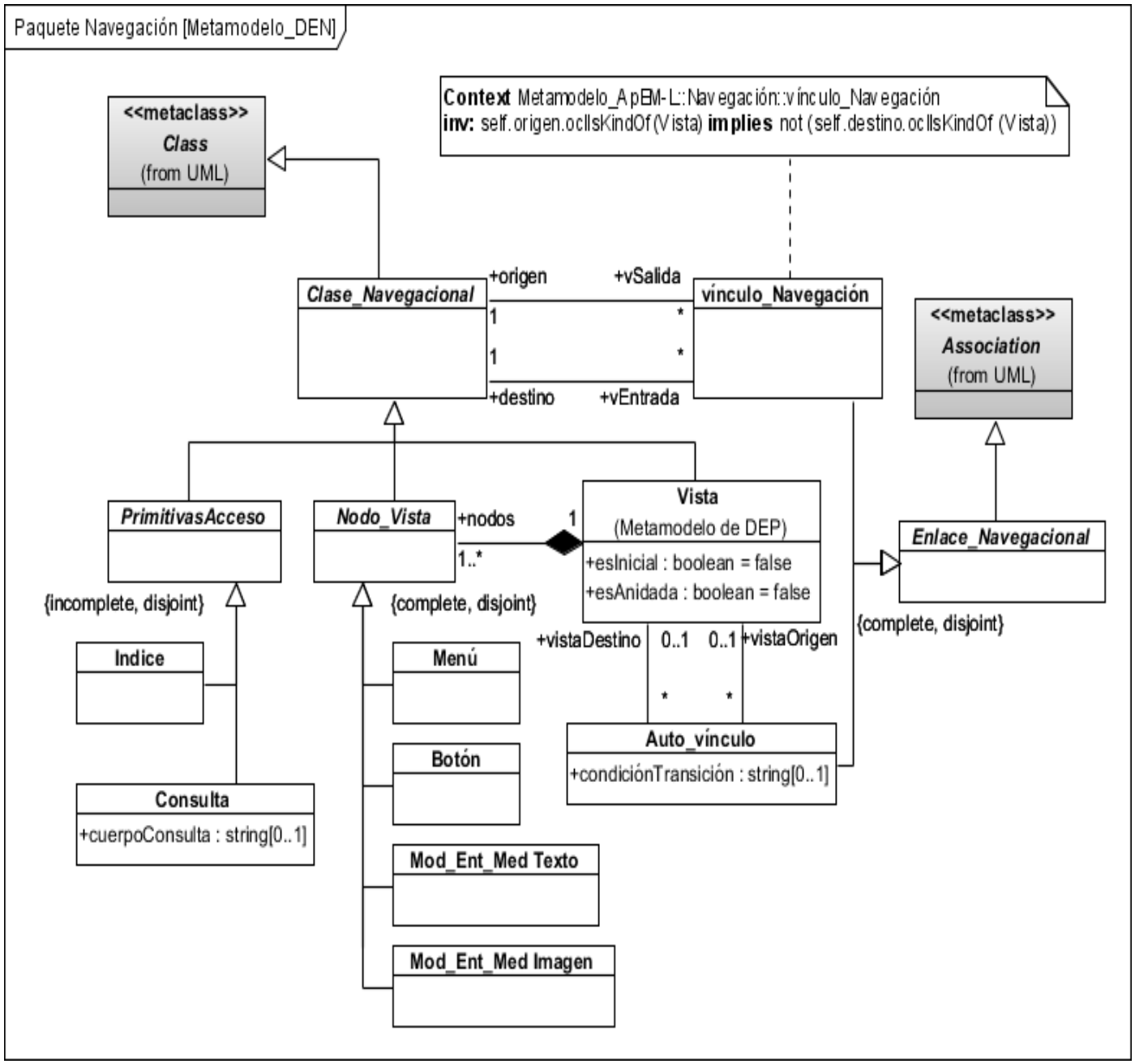

Figura 14: Metamodelo del DEN de ApEM-L correspondiente al paquete Presentación. [Tomado de (Herrera, 2014a)]

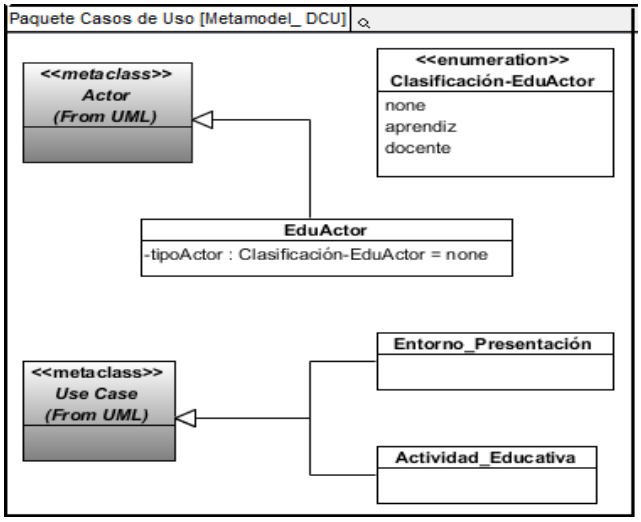

Figura 15: Metamodelo del DCU de ApEM-L correspondiente al paquete de Casos de Uso. [Tomado de (Herrera, 2014a)]

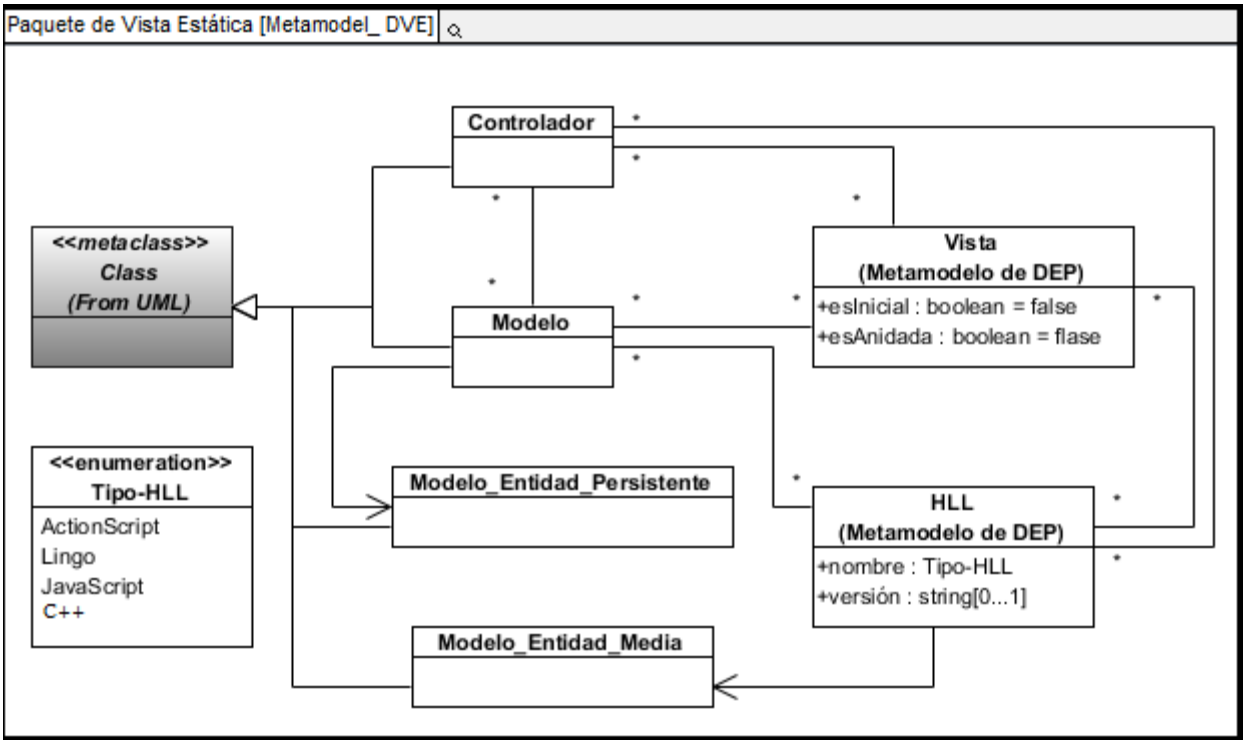

Figura 16: Metamodelo del DVE de ApEM-L correspondiente al paquete de Vista Estática. [Tomado de (Herrera, 2014a)]

Construcción del editor del modelo del sistema

Con el apoyo de autores como (Montenegro y otros, 2011a) y del tablero de herramientas (dashboard) de la figura 16 de la herramienta GMF, se realizó la construcción del editor del modelo del sistema. Para el desarrollo de la investigación es necesario construir un editor del modelo para cada diagrama a implementar en la solución.

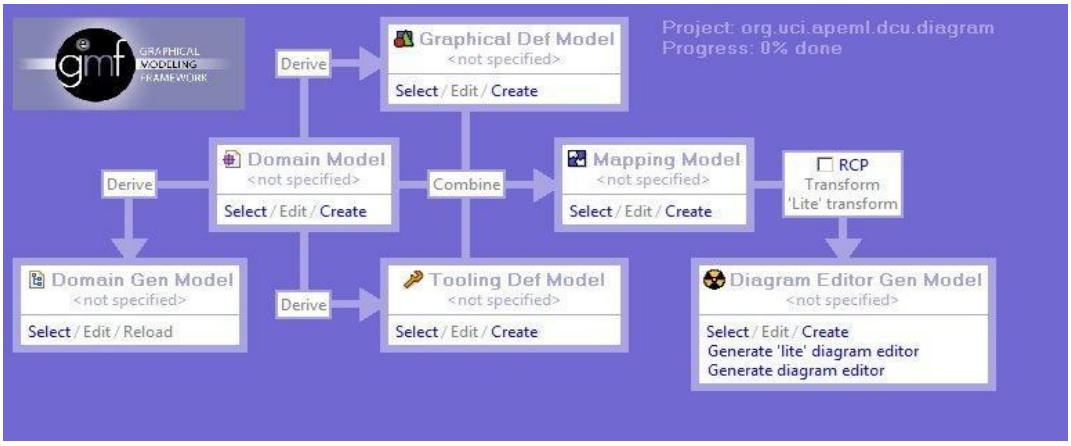

Figura 17: Tablero de Herramientas de GMF (Dashboard) [Tomado del IDE Eclipse].

3.3 Diagrama de clases del diseño

A continuación solo se muestra una pequeña parte del diagrama completo, de forma general y resumida, dado el tamaño de los mismos. (Ver Anexo II). Se muestran algunas de las clases que realizan la funcionalidad principal de construir diagramas. El paquete exhibido llamado “metamodelo”, es el correspondiente según el plugin al que pertenezca, los cuales fueron expuestos en el epígrafe anterior.

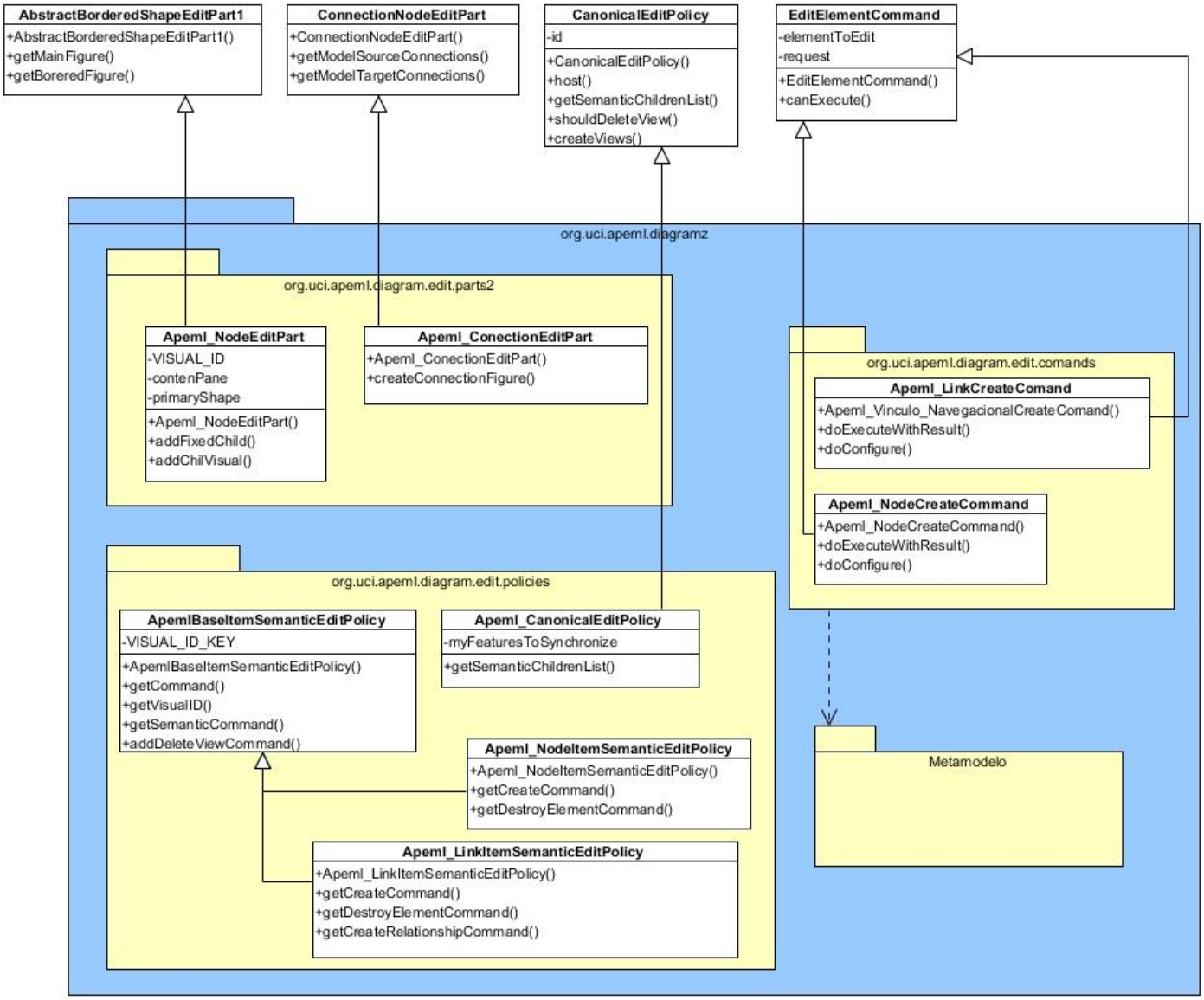

Figura 18: Diagrama de clases para la funcionalidad de "crear elemento"

Descripción de clases del sistema

A partir del diagrama de clases expuesto, se procederá para realizar la descripción de las clases que se muestran.

Tabla 7: Descripción de las clases de la solución propuesta

Clase	Descripción
AbstractBorderedShapeEditPart	Clase del sistema que tiene como función la contención de figuras delimitadas (que tengan fronteras). Maneja la adición de los elementos de frontera con un localizador por defecto
ConnectionNodeEditPart	Clase del sistema que es la conexión especializada que instala un EditPolicy.GRAPHICAL_NODE_ROLE (clase responsable de crear y reconectar las conexiones gráficamente) y también implementa INodeEditPart (interfaz que define las piezas que se pueden conectar)
CanonicalEditPolicy	Clase del sistema donde se va a crear, si es necesario, elementos de notación para todos los elementos semánticos insertados en el elemento de acogida, o eliminar el elemento de notación para el elemento semántico eliminando el elemento de acogida.
EditElementCommand	Clase del sistema abstracta para los comandos que modifican los elementos del modelo.
Apeml_NodeEditPart	Clase controladora para la visualización de la notación y el elemento semántico. Esta clase representa a todas las clases que representan a los elementos delimitados de los diagramas. Como los elementos: Apeml_HLLEditPart, Apeml_VistaEditPart, Apeml_Vista2EditPart, Apeml_InteractivoEditPart (que representa a los elementos: video, imagen, texto, animación, botón), Apeml_EstáticoEditPart (que representa a los elementos: video, imagen, texto, animación). Del diagrama DEP. En el diagrama DEN contendrá los elementos como: Apeml_VistaEditPart, Apeml_Vista2EditPart, Apeml_ConsultaEditPart, Apeml_IndiceEditPart, Apeml_BotonEditPart, Apeml_MenuEditPart. En el DCU se evidenciará: Apeml_Actor_EducativoEditPart, Apeml_Actividad_EducativaEditPart, Apeml_Entorno_PresentaciónEditPart. En el DVE contendrá los elementos de: Apeml_VistaEditPart, Apeml_ControladorEditPart, Apeml_ModeloEditPart, Apeml_HLLEditPart, Apeml_Modelo_Entidad_PersistenteEditPart, Apeml_Modelo_Entidad_MediaEditPart.
Apeml_LinkEditPart	Clase controladora para la visualización de la notación y el elemento semántico. Esta clase representa las conexiones entre los elementos de los diagramas. Como por ejemplo: Apeml_AutoVinculo_NavegacionalEditPart, Apeml_Vinculo_NavegacionalEditPart.
ApemlBaseItemSemanticEditPolicy	Clase cuya función es manejar la creación y actualización de los elementos del modelo semántico.
Apeml_NodeItemSemanticEditPolicy	Clase que extiende de la clase ApemlBaseItemSemanticEditPolicy. Esta clase representa los elementos delimitados de los diagramas. Como los elementos: Apeml_HLLItemSemantiEditPolicy, Apeml_VistaItemSemantiEditPolicy, Apeml_Vista2ItemSemantiEditPolicy, Apeml_InteractivoItemSemantiEditPolicy (que representa a los

Clase	Descripción
	elementos: video, imagen, texto, animación, botón), Apeml_EstáticoItemSemantiEditPolicy (que representa a los elementos: video, imagen, texto, animación). Del diagrama DEP. En el diagrama DEN se almacenan los elementos: Apeml_VistaItemSemantiEditPolicy, Apeml_Vista2ItemSemantiEditPolicy, Apeml_ConsultaItemSemantiEditPolicy, Apeml_IndiceItemSemantiEditPolicy, Apeml_BotonItemSemantiEditPolicy, Apeml_MenuItemSemantiEditPolicy. En el DCU se almacenan los elementos: Apeml_Actor_EducativoItemSemantiEditPolicy, Apeml_Actividad_EducativaItemSemantiEditPolicy, Apeml_Entorno_PresentaciónItemSemantiEditPolicy. En el DVE se almacenan los elementos de: Apeml_VistaItemSemantiEditPolicy, Apeml_ControladorItemSemantiEditPolicy, Apeml_ModeloItemSemantiEditPolicy, Apeml_HLLItemSemantiEditPolicy, Apeml_Modelo_Entidad_PersistenteItemSemantiEditPolicy, Apeml_Modelo_Entidad_MediaItemSemantiEditPolicy.
Apeml_LinkItemSemanticEditPolicy	Esta clase representa las conexiones entre los elementos de los diagramas, como: Apeml_AutoVinculo_NavegacionalItemSemanticEditPolicy, Apeml_Vinculo_NavegacionalItemSemantiEditPolicy.
Apeml_NodeCanonicalEditPolicy	Clase que extiende de CanonicalEditPolicy heredando sus funcionalidades. Esta clase representa a todas las clases que representan un elemento delimitado como: Apeml_HLLEditPolicy, Apeml_VistaEditPolicy, Apeml_Vista2EditPolicy, Apeml_InteractivoEditPolicy (que representa a los elementos: video, imagen, texto, animación, botón), Apeml_EstáticoEditPolicy (que representa a los elementos: video, imagen, texto, animación). Del diagrama DEP. En el diagrama DEN se almacenan los elementos: Apeml_VistaEditPolicy, Apeml_Vista2EditPolicy, Apeml_ConsultaEditPolicy, Apeml_IndiceEditPolicy, Apeml_BotonEditPolicy, Apeml_MenuEditPolicy. En el DCU se almacenan los elementos: Apeml_Actor_EducativoEditPolicy, Apeml_Actividad_EducativaEditPolicy, Apeml_Entorno_PresentaciónEditPolicy. En el DVE se almacenan los elementos: Apeml_VistaEditPolicy, Apeml_ControladorEditPolicy, Apeml_ModeloEditPolicy, Apeml_HLLEditPolicy, Apeml_Modelo_Entidad_PersistenteEditPolicy,

Clase	Descripción
	Apeml_Modelo_Entidad_MediaEditPolicy.
Apeml_NodeCreateCommand	Clase que extiende de la clase del sistema EditElementCommand, heredando así sus funcionalidades. Esta clase representa a todas las clases que representan un elemento delimitado como: Apeml_HLLCreateCommand, Apeml_VistaCreateCommand, Apeml_Vista2CreateCommand, Apeml_InteractivoCreateCommand (que representa a los elementos: video, imagen, texto, animación, botón), Apeml_EstáticoCreateCommand (que representa a los elementos: video, imagen, texto, animación). Del diagrama DEP. En el diagrama DEN se almacenan los elementos: Apeml_VistaCreateCommand, Apeml_Vista2CreateCommand, Apeml_ConsultaCreateCommand, Apeml_IndiceCreateCommand, Apeml_BotonCreateCommand, Apeml_MenuCreateCommand. En el DCU se almacenan los elementos: Apeml_Actor_EducativoCreateCommand, Apeml_Actividad_EducativaCreateCommand, Apeml_Entorno_PresentaciónCreateCommand. En el DVE se almacenan los elementos: Apeml_VistaCreateCommand, Apeml_ControladorCreateCommand, Apeml_ModeloCreateCommand, Apeml_HLLCreateCommand, Apeml_Modelo_Entidad_PersistentCreateCommand, Apeml_Modelo_Entidad_CreateCommand.
Apeml_LinkCreateCommand	Clase que extiende de la clase del sistema EditElementCommand, heredando así sus funcionalidades. Esta clase representa a las clases que representan relaciones como: Apeml_AutoVinculo_NavegacionalCreateCommand, Apeml_Vinculo_NavegacionalCreateCommand

3.4 Diagrama de secuencia del sistema

Un flujo de información fundamental de los plugins que se desarrollaron, es el de creación de un elemento en el lienzo de desarrollo. Para ello, el sistema reacciona a dos eventos principales, el evento on mouse move (movimiento del mouse), y el evento on mouse click (realización de click con el mouse). En el primer evento son creados los comandos para la creación de los elementos, y con el segundo evento se ejecutan los comandos del evento anterior. En los siguientes diagramas se muestra el flujo de acciones que ocurren en el sistema al realizar la funcionalidad de dibujar un elemento, mediante los eventos mencionados.

1. Creación de un elemento con el evento on mouse move.

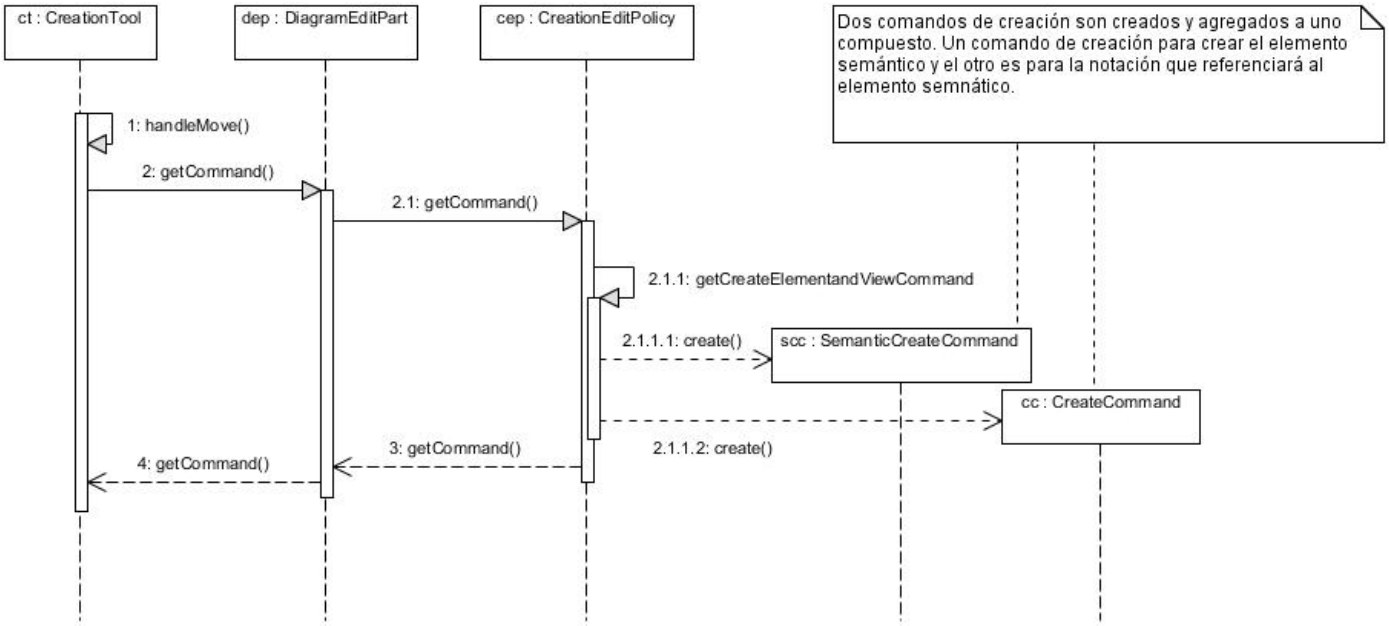

Figura 19: Diagrama de secuencia para la creación de un elemento con el evento on mouse move.

2. Creación de un elemento con el evento on mouse click. Se observará en dos fases. Primero la creación del elemento semántico y la notación, y luego la creación del EditPart (controlador del elemento) y su respectiva figura.

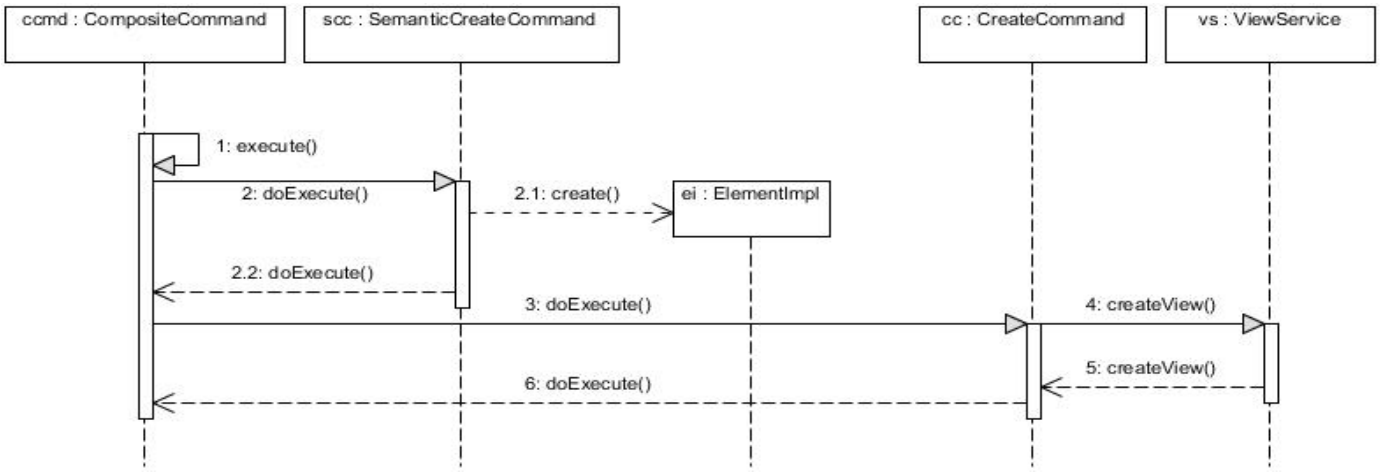

Figura 20: Diagrama de secuencia para la creación de un elemento con el evento on mouse click (Fase 1)

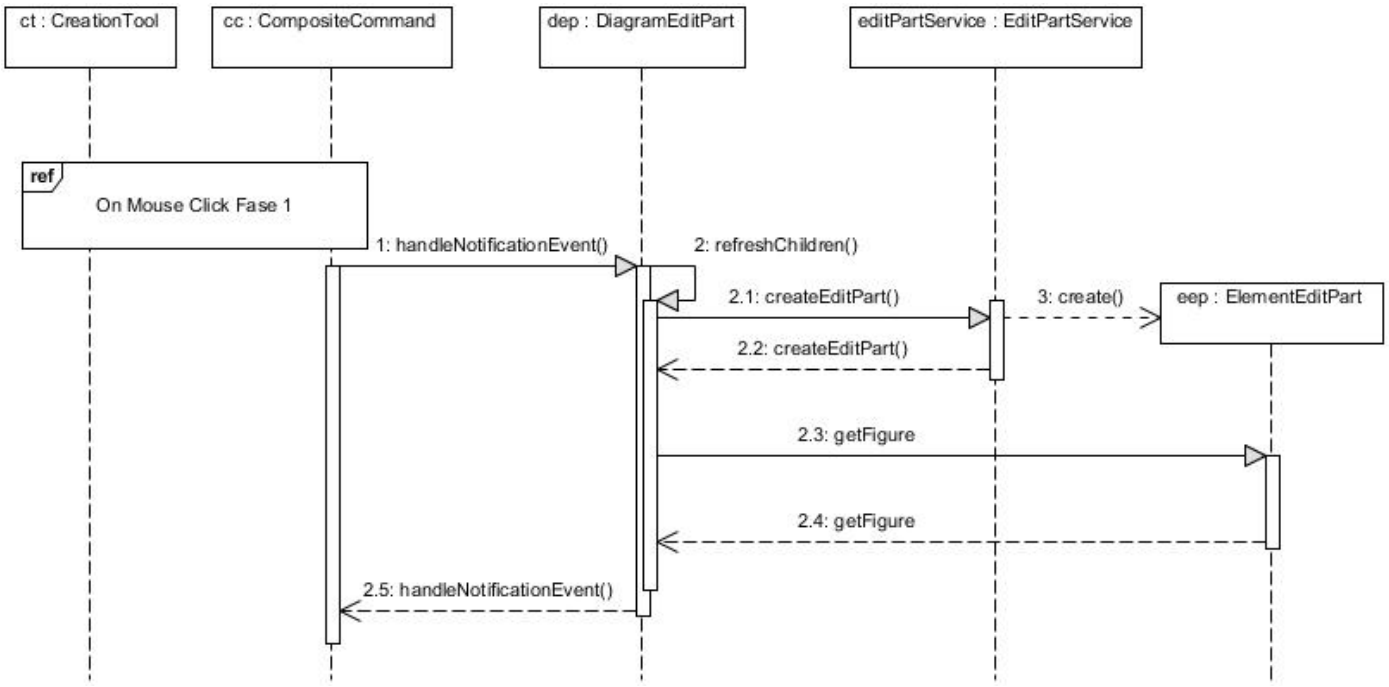

Figura 21: Diagrama de secuencia para la creación de un elemento con el evento on mouse click (Fase 2)

3.5 Diagrama de componentes del sistema

El presente diagrama de componentes describe cada uno de los elementos asociados al diseño de clases descrito anteriormente, así como la relación de dependencia entre los elementos que lo integran.
El componente org.uci.apeml.diagram.jar representa la extensión para la herramienta Graphical Modeling Framework encargado de realizar las funciones de dibujar los diagramas para el lenguaje ApEM-L 2.0. Mientras que el plugin org.uci.apeml.jar es una extensión que contiene al metamodelo con sus reglas y restricciones. Lo que permite a org.uci.apeml.diagram.jar realizar sus funcionalidades.

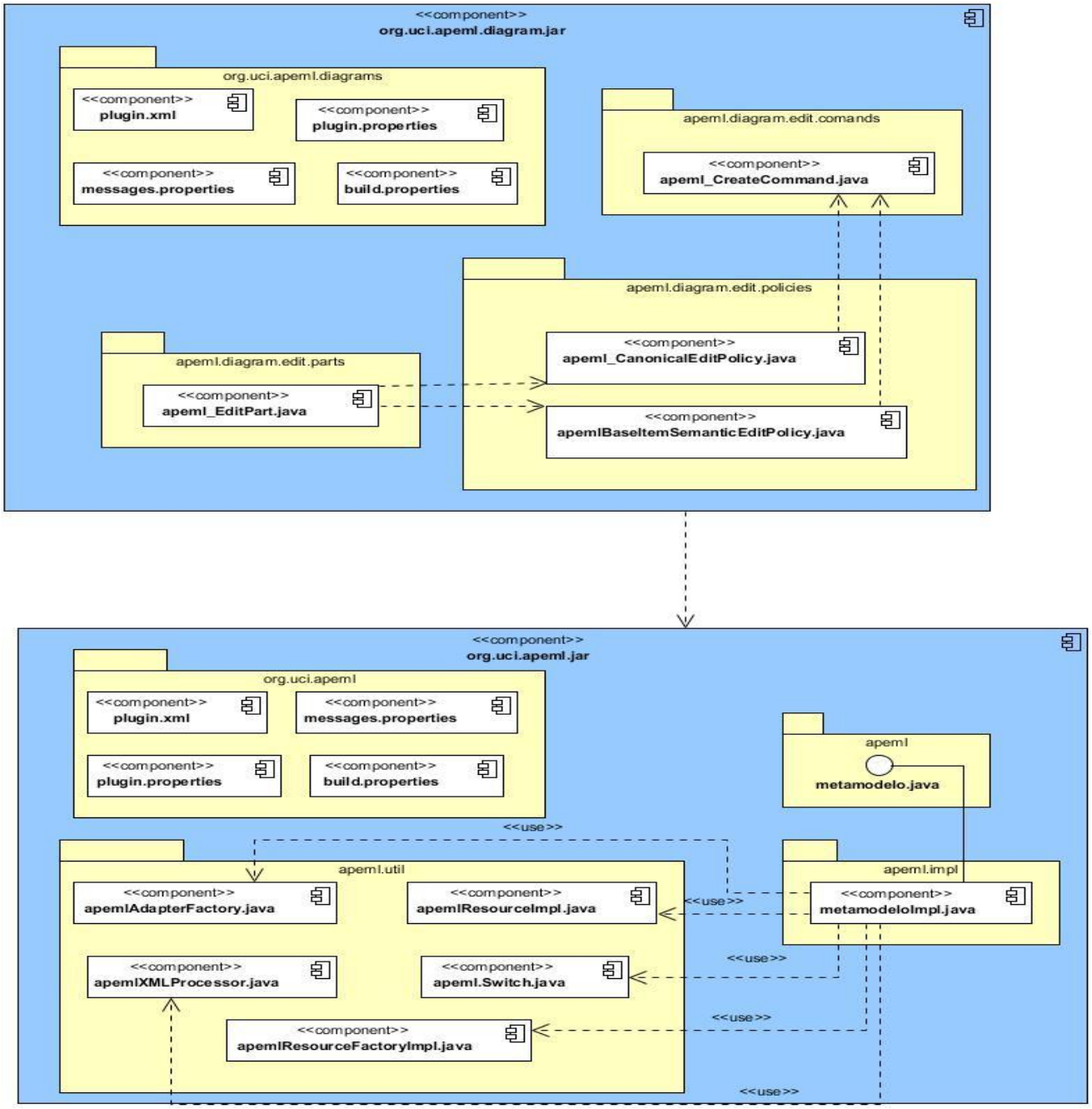

Figura 22: Diagrama de componentes del plugin para el diagrama de vista estática.

3.6 Pruebas de Software

Una vez realizada la implementación del software, se procede a la realización de las pruebas del mismo para corregir posibles errores. Según (Pressman, 2005) existen dos enfoques fundamentales a la hora de realizar una prueba de software. Estos son denominados pruebas de caja negra y pruebas de caja blanca. Para el desarrollo de la investigación se escogió el enfoque de caja blanca ya que realiza un extremado análisis del procedimiento que se implementó, lo que permitió generar casos de prueba para bucles y estructuras condicionales.

Como estrategia de prueba se ejecutaron casos de pruebas de forma manual y automática, para garantizar la veracidad de los resultados obtenidos. Esto permitió que finalmente se compararan los resultados obtenidos para cada caso.

Camino básico

Para la realización manual de las pruebas se usó la técnica de camino básico. Dicho método permitirá definir un conjunto básico de rutas de ejecución, y comprobar el funcionamiento de cada una ellas. Esto permitirá validar que todas las sentencias del procedimiento se ejecuten al menos una vez y de manera correcta.

Según (Pressman, 2005) los pasos a realizar para aplicar esta técnica son:

- Representar el programa en un grafo de flujo: se utiliza para representar el flujo de control lógico de un programa.
- Calcular la complejidad ciclomática: el valor calculado define el número de rutas independientes en el conjunto básico de un programa, y proporciona un límite superior para el número de pruebas que deben aplicarse, lo cual asegura que todas las instrucciones se hayan ejecutado por lo menos una vez.
- Determinar el conjunto básico de rutas independientes: es cualquier ruta del programa que ingresa por lo menos un nuevo conjunto de instrucciones de procesamiento o una nueva condición.
- Derivar los casos de prueba que fuerzan la ejecución de cada camino.

A continuación se muestra el caso de prueba, guiado por los pasos anteriores, al método *getCreateElementAndViewCommand*. El mismo se localiza en la clase controladora *CreationEditpolicy* del plugin para el diagrama de vista estática perteneciente a la colección de plugins. El objetivo dicho método es crear los comandos necesarios para dibujar cualquier elemento de este tipo de diagrama. La clase y el

método mencionado están representados en el diagrama de secuencia (Figura 19) del presente capítulo.

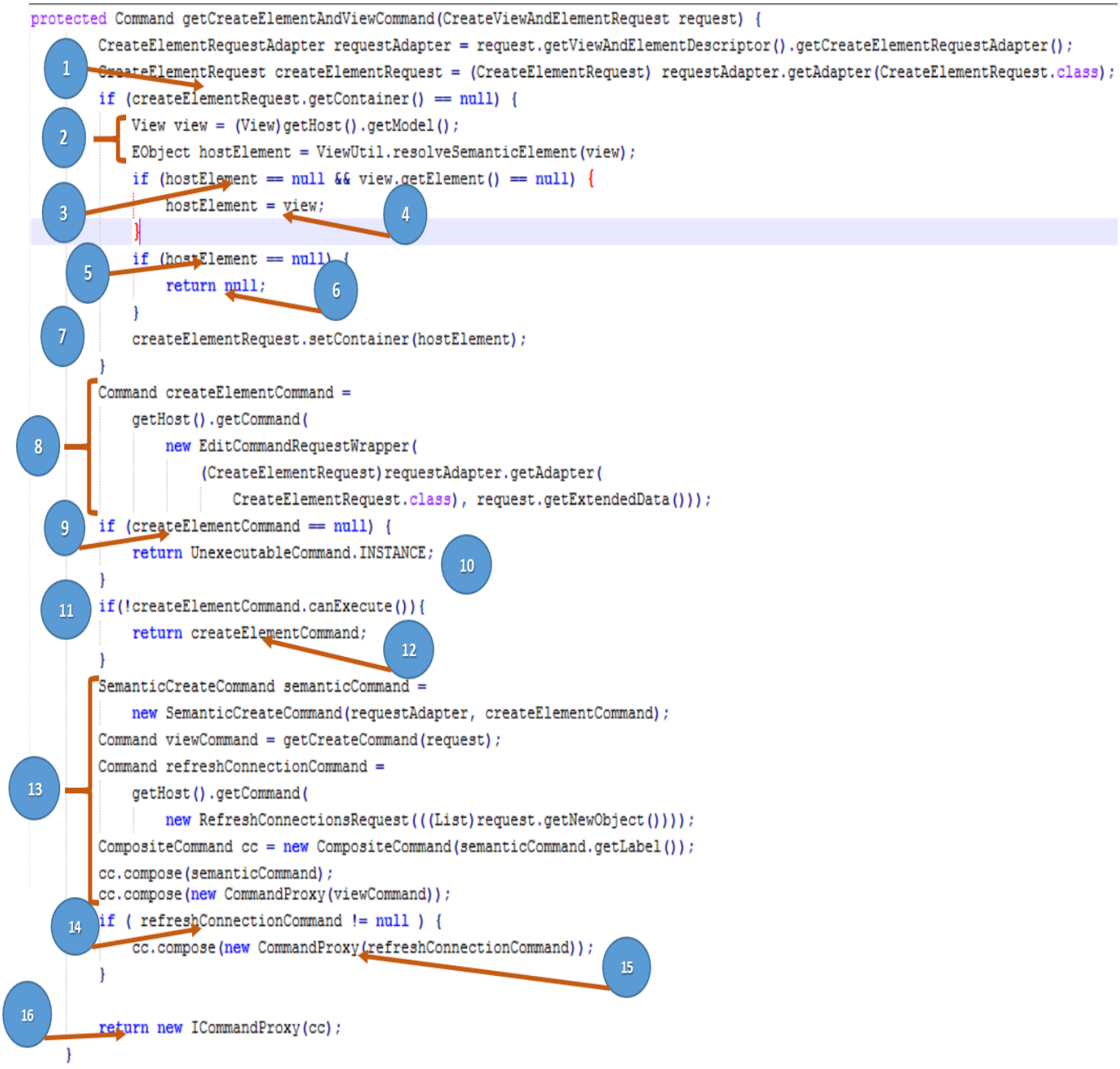

Figura 23: Código fuente del método getCreateElementAndViewCommand de la clase CreationEditpolicy.

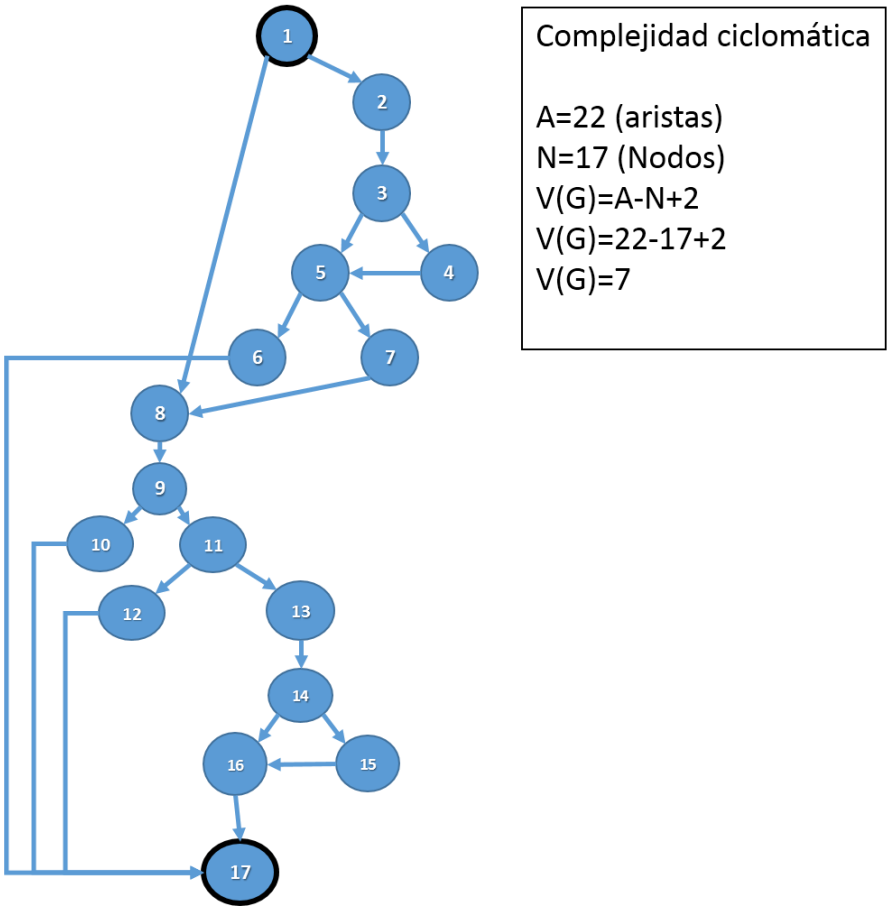

Figura 24: Gráfica de flujo y Complejidad ciclomática correspondiente al método getCreateElementAndViewCommand.

Como es posible observar, en la figura 24, se añade un nuevo nodo con el número 17. El mismo se crea con el objetivo de agrupar todos los métodos de retorno del algoritmo.

1-8-9-10-17
1-8-9-11-12-17
1-8-9-11-13-14-16-17
1-8-9-11-13-14-15-16-17
1-2-3-5-7-8-9-11-13-14-16-17
1-2-3-5-6-17
1-2-3-4-5-6-17

Figura 25: Rutas linealmente independientes correspondientes al método getCreateElementAndViewCommand.

Casos de prueba diseñados:

Caso de prueba del camino 1:

Descripción de entrada: se recibe como parámetro una petición (Request) para la creación de una clase Vista, con errores en la misma.

Resultado Esperado: UnexecutableCommand.INSTANCE

Objetivo: se garantiza que el elemento creado en el flujo 8 no sea de tipo null. Esto significaría que el elemento no se creó correctamente.

Caso de prueba del camino 2:

Descripción de entrada: se recibe como parámetro una petición para la creación de una relación entre una clase Vista y una Modelo Entidad Persistente. Dicha petición no necesita de su completamiento, pues la relación estará contenida dentro del paquete principal.

Resultado Esperado: null.

Objetivo: Se garantiza la comprobación de la condición de creación para el comando de un elemento para el diagrama.

Caso de prueba del camino 3:

Descripción de entrada: se recibe como parámetro una petición, para la creación de una clase Vista. Dicha petición no necesita de su completamiento, pues la clase estará contenida dentro del paquete principal.

Resultado Esperado: Un comando compuesto (composite comand) con uno para crea la clase Vista semánticamente y otro para crear la clase Vista visualmente.

Objetivo: Se garantiza la comprobación de los pasos para crear un comando compuesto de creación.

Caso de prueba del camino 4:

Descripción de entrada: se recibe como parámetro una petición, para la creación de una clase Vista que poseerá conexiones con una clase Modelo. Dicha petición no necesita de su completamiento, pues la clase estará contenida dentro del paquete principal.

Resultado Esperado: Un comando compuesto con uno para crea la clase Vista semánticamente y otro para crear la clase Vista visualmente. Además estará compuesto por otro extra para refrescar las conexiones en la notación.

Objetivo: Se garantiza la comprobación de la condición de creación de un comando para refrescar las conexiones a un elemento.

Caso de prueba del camino 5:

Descripción de entrada: se recibe como parámetro una petición, para la creación de una clase Vista, con la especificación semántica, pero no visual, del paquete al que pertenece.

Resultado Esperado: Un comando compuesto con uno para crea la clase Vista semánticamente y otro para crear la clase Vista visualmente.

Objetivo: Se garantiza la ejecución de los pasos para asignarle un elemento contenedor a un elemento.

Caso de prueba del camino 6:

Descripción de entrada: se recibe como parámetro una petición, para la creación de una clase Vista, sin la especificación semántica, ni visual, del paquete al que pertenece.

Resultado Esperado: null.

Objetivo: Se garantiza la comprobación de la condición existencia de un elemento contenedor para el elemento que se desea crear.

Caso de prueba del camino 7:

Descripción de entrada: se recibe como parámetro una petición, para la creación de una clase Vista, sin la especificación semántica, pero si visual, del paquete al que pertenece.

Resultado Esperado: null.

Objetivo: Se garantiza la comprobación de la condición existencia de un elemento contenedor para el elemento que se desea crear. Además se comprueban los pasos para la obtención de un elemento semántico contenedor.

Después de culminado el proceso de prueba de manera manual, se usó de la herramienta EclEmma para la realización de las pruebas automáticas. Para la realización de dichas pruebas se usaron los mismos casos de prueba mencionados anteriormente. En estas pruebas el código quedó marcado de la siguiente manera:

- En color verde, las líneas de código que han sido ejecutadas.
- En color rojo, las líneas de código que no han sido ejecutadas.

A continuación se presenta en la figura 26 y en la figura 27 los resultados de los casos pruebas 1 y 5 respectivamente.

```
protected Command getCreateElementAndViewCommand(CreateViewAndElementRequest request) {
        CreateElementRequestAdapter requestAdapter = request.getViewAndElementDescriptor()
        .getCreateElementRequestAdapter();
        CreateElementRequest createElementRequest = (CreateElementRequest) requestAdapter
        .getAdapter(CreateElementRequest.class);
        if (createElementRequest.getContainer() == null) {
            View view = (View)getHost().getModel();
            EObject hostElement = ViewUtil.resolveSemanticElement(view);
            if (hostElement == null && view.getElement() == null) {
                hostElement = view;          }
            if (hostElement == null) {
                return null;             }
            createElementRequest.setContainer(hostElement);
        }
        Command createElementCommand =
            getHost().getCommand(
                new EditCommandRequestWrapper(
                    (CreateElementRequest)requestAdapter.getAdapter(
                        CreateElementRequest.class), request.getExtendedData()));
            return UnexecutableCommand.INSTANCE;
        }
        if(!createElementCommand.canExecute()){ return null;
        }
        SemanticCreateCommand semanticCommand =
            new SemanticCreateCommand(requestAdapter, createElementCommand);
        Command viewCommand = getCreateCommand(request);
        Command refreshConnectionCommand =
            getHost().getCommand(
                new RefreshConnectionsRequest(((List)request.getNewObject())));
        CompositeCommand cc = new CompositeCommand(semanticCommand.getLabel());
        cc.compose(semanticCommand);
        cc.compose(new CommandProxy(viewCommand));
        if ( refreshConnectionCommand != null ) {
            cc.compose(new CommandProxy(refreshConnectionCommand));
        }
        return new ICommandProxy(cc);
    }
```

Figura 26: Resultado del caso de prueba 1 con EclEmma.

```
protected Command getCreateElementAndViewCommand(CreateViewAndElementRequest request) {
        CreateElementRequestAdapter requestAdapter = request.getViewAndElementDescriptor()
        .getCreateElementRequestAdapter();
        CreateElementRequest createElementRequest = (CreateElementRequest) requestAdapter
        .getAdapter(CreateElementRequest.class);
        if (createElementRequest.getContainer() == null) {
            View view = (View)getHost().getModel();
            EObject hostElement = ViewUtil.resolveSemanticElement(view);
            if (hostElement == null && view.getElement() == null) {
                hostElement = view;          }
            if (hostElement == null) {
                return null;             }
            createElementRequest.setContainer(hostElement);
        }
        Command createElementCommand =
            getHost().getCommand(
                new EditCommandRequestWrapper(
                    (CreateElementRequest)requestAdapter.getAdapter(
                        CreateElementRequest.class), request.getExtendedData()));
        if (createElementCommand == null) {
            return UnexecutableCommand.INSTANCE;
        }
        if(!createElementCommand.canExecute()){ return null;
        }
        SemanticCreateCommand semanticCommand =
            new SemanticCreateCommand(requestAdapter, createElementCommand);
        Command viewCommand = getCreateCommand(request);
        Command refreshConnectionCommand =
            getHost().getCommand(
                new RefreshConnectionsRequest(((List)request.getNewObject())));
        CompositeCommand cc = new CompositeCommand(semanticCommand.getLabel());
        cc.compose(semanticCommand);
        cc.compose(new CommandProxy(viewCommand));
        if ( refreshConnectionCommand != null ) {
            cc.compose(new CommandProxy(refreshConnectionCommand));
        }
        return new ICommandProxy(cc);
    }
```

Figura 27: Resultado del caso de prueba 5 con EclEmma

Conclusiones Parciales

Teniendo en cuenta lo abordado en el capítulo se concluyó que:

- El patrón arquitectónico MVC cumple con las características que permiten en la solución propuesta la distribución de las responsabilidades de procesamiento, almacenamiento y las actividades de

entradas y salidas. Este permitió separar las responsabilidades de las clases para el procesamiento de la información necesaria en la gestión de los diagrama a generar por los plugins.

- La aplicación de la técnica camino básico y el uso de la herramienta EclEmma, permitió comprobar los resultados respecto a los casos de pruebas. Esta comprobación arrojó resultados favorables para la investigación.
- En el desarrollo de la solución se necesitó de la construcción de un plugin para cada uno de los diagramas descritos en la nueva versión de ApEM-L. Dichos plugins necesitaran del IDE Eclipse para su funcionamiento.

CAPÍTULO IV: VALIDACIÓN DE LA PROPUESTA

El presente capítulo está dedicado a evaluar el cumplimiento de la hipótesis. Para alcanzar dicha meta se hace necesario probar si la herramienta resultante cumple con ApEM-L 2.0. Se procederá a realizar una comparación de los resultados de esta comprobación, con los valores iniciales.
Para la comprobación realizada de la herramienta, se definió el caso de estudio empleado por (Ciudad, 2007) con la aplicación multimedia “A Jugar”.

4.1 Modelado de la aplicación multimedia “A Jugar” con el lenguaje ApEM-L 1.0, UML 2.0 y ApEM-L 2.0

La aplicación “A Jugar” es un entorno el cual se desarrollaba en la antigua facultad 9 hoy facultad 6 de la UCI. El objetivo fundamental del proyecto, era el desarrollo de un producto educativo interactivo para desarrollar el aprendizaje de los niños en edades preescolares.
Para la realización del modelado de la aplicación antes mencionada, el autor (Ciudad, 2007) realiza el trabajo para la presentación y la tarea 1 del nivel 4 del producto. Parte del modelado contó con diagramas como: el diagrama de casos de uso; el diagrama de clases (modelo conceptual); el diagrama de estructura presentación y el diagrama de estructura navegación, los cuales se realizaban en cualquier herramienta de modelado que soportara el lenguaje UML.

Modelado del diagrama de Casos de uso

- Diagrama de casos de uso con ApEM-L 1.0, el cual no sufre transformación alguna del él ya existente de UML. Realizado con la herramienta Rational Rose

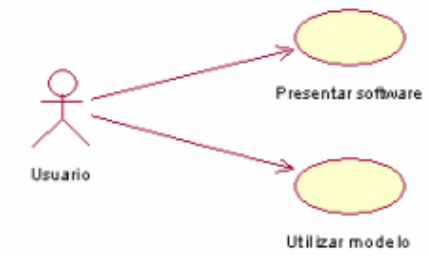

Figura 28: Diagrama de casos de uso seleccionado del proyecto: "A jugar". [Tomado de (Ciudad, 2007)]

Para una mejor comprensión se incorporará la descripción textual de los casos de uso utilizada según (Ciudad, 2007). (Ver Anexo III).

- Diagrama de casos de uso con ApEM-L 2.0, el cual sufre notables transformaciones (en los estereotipos) al establecer el modelado en ApEM-L 2.0, las cuales la nueva herramienta facilita en su representación. Realizado con la solución propuesta en la investigación.

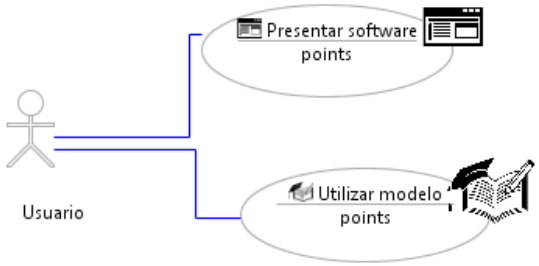

Figura 29: Diagrama de casos de uso modelado con ApEM-L 2.0 en la propuesta de solución.

Modelado del Diagrama clases (Modelo Conceptual)

- Diagrama de Clases con ApEM-L 1.0: incorpora nuevos elementos representados con la decoración de estereotipos de UML Realizado con Rational Rose.

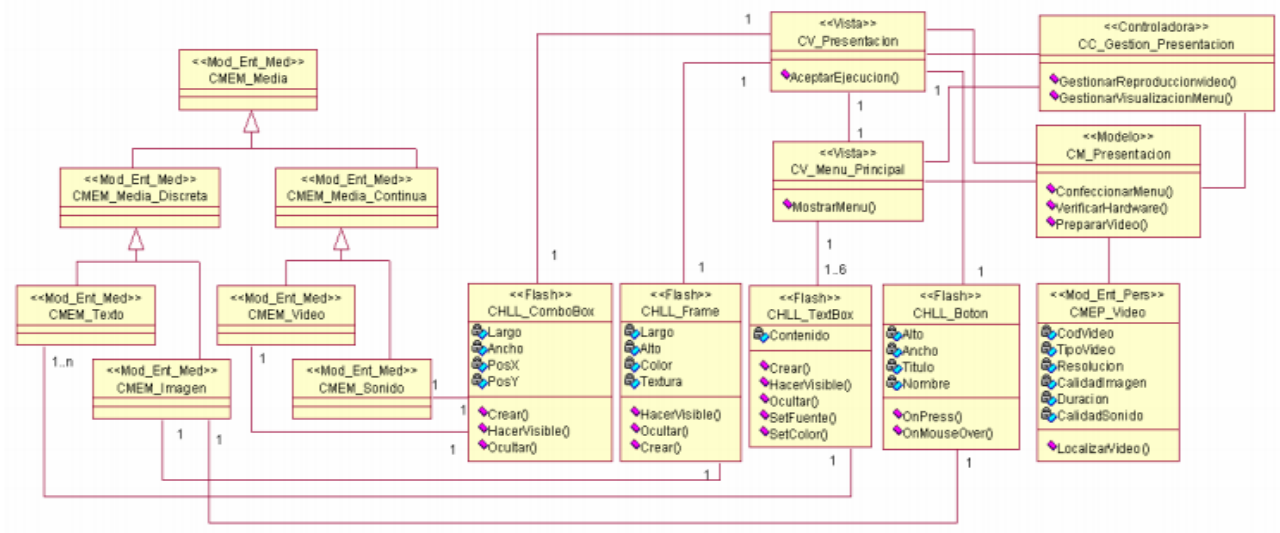

Figura 30: Diagrama de clases (Modelo Conceptual) del caso de uso: PresentarSoftware. [Tomado de (Ciudad, 2007)]

- Diagrama de Clases con UML 2.0: sin la especificación de los elementos del diagrama anterior. Realizado con Visual Paradigm for UML

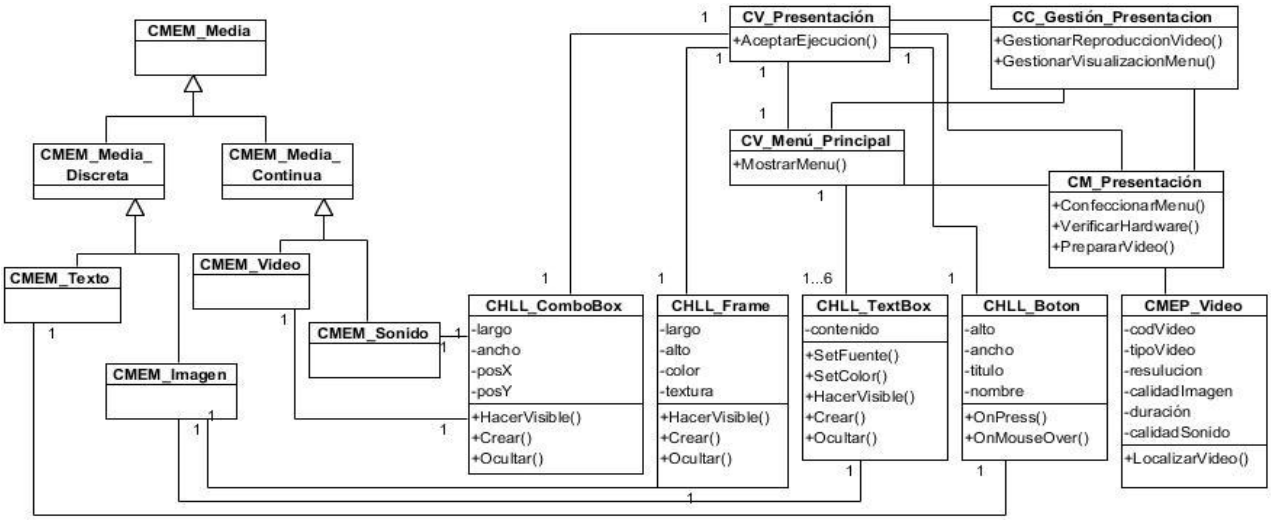

Figura 31: Diagrama de clases (Modelo Conceptual) del caso de uso: Presentar Software modelado con UML 2.0

- Diagrama de clases con ApEM-L 2.0: incorpora nuevos estereotipos restrictivos y fue creado según la especificación de su metamodelo. Realizado la solución propuesta en la investigación.

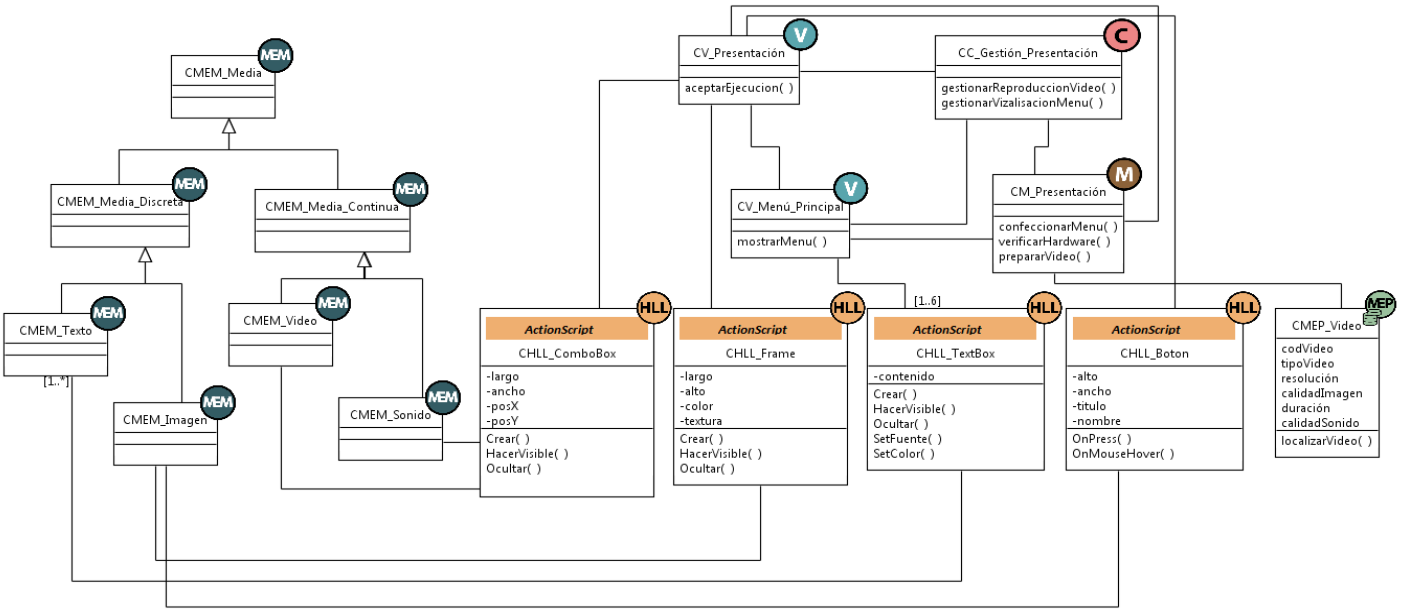

Figura 32: Diagrama de clases (Modelo Conceptual) del caso de uso: Presentar Software modelado con la propuesta de solución

Modelado del Diagrama de estructura de navegación

- Diagrama de estructura de navegación, el cual representa al caso de uso: presentar software.

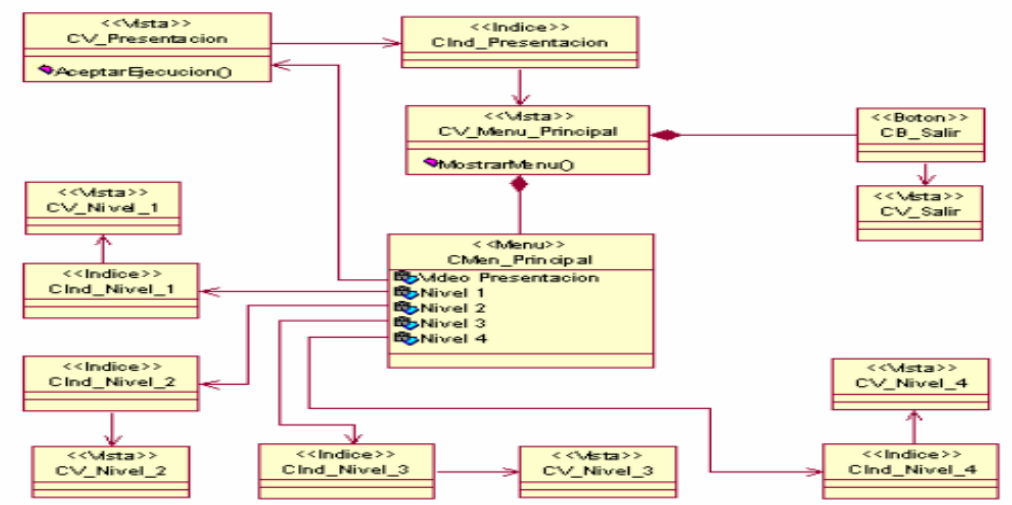

Figura 33: Diagrama de estructura de navegación del caso de uso: Presentar software. [Tomado de (Ciudad, 2007)]

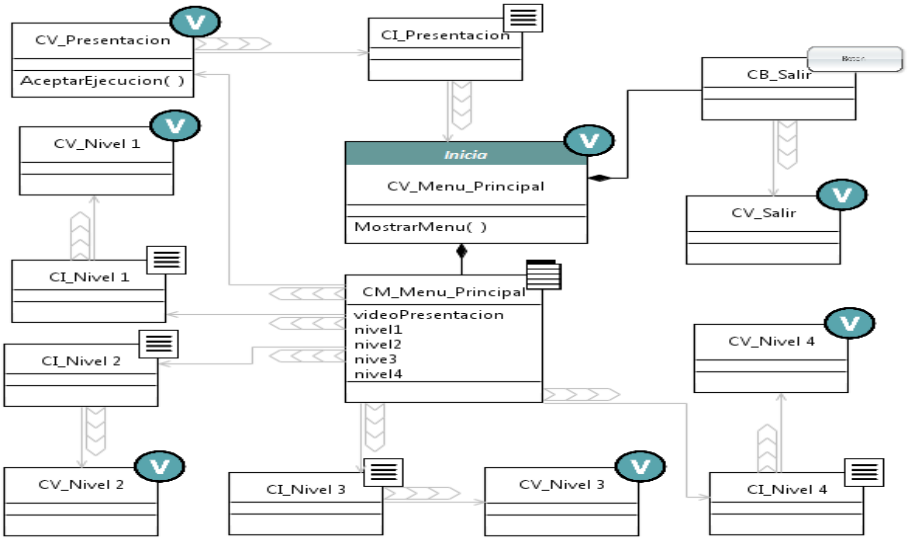

Figura 34: Diagrama de estructura de navegación del caso de uso: Presentar software. Modelado con la propuesta de solución

Modelado del Diagrama de estructura presentación

- Diagrama de estructura de presentación, el cual representa al caso de uso: presentar software.

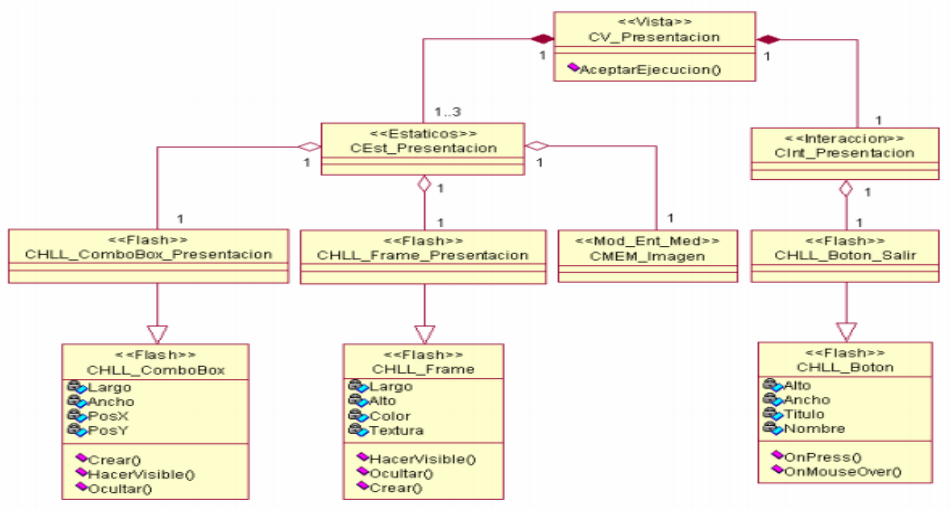

Figura 35: Diagrama de estructura de presentación de la interfaz de usuario Presentación [Tomado de (Ciudad, 2007)]

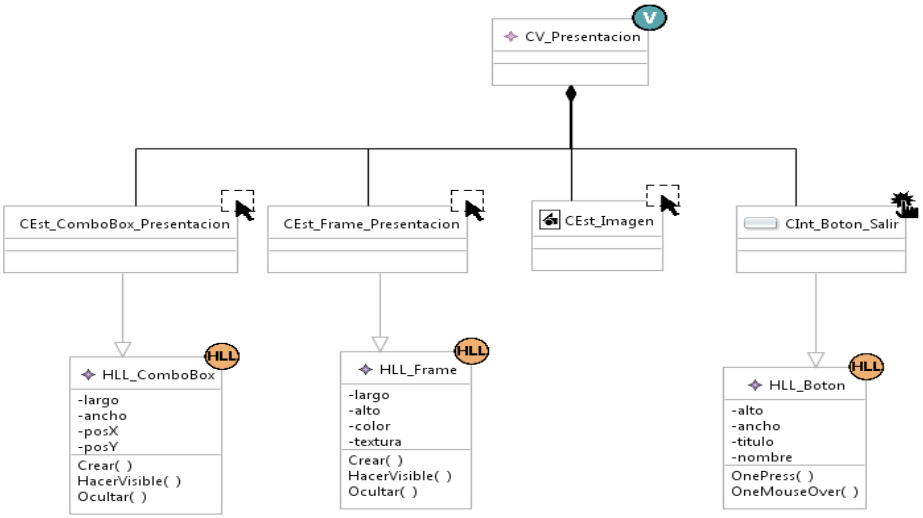

Figura 36: Diagrama de estructura de presentación de la interfaz de usuario Presentación, modelado en la propuesta de solución

4.2 Resultado de los valores alcanzados

Primeramente se aplicó la operacionalización a los diagramas de la primera versión del lenguaje en cuanto a la dimensión precisión llegando a los siguientes resultados:

Tabla 8: Resultado de la operacionalización de la dimensión precisión a ApEM-L 1.0

Indicador	Análisis	Resultado
Cantidad de componentes de los diagramas que modelan con precisión la estructura lógica de la presentación del sistema.	En este indicador se analizó el DEP, donde los componentes estereotipados de <<Estáticos>> e <<Interacción>> traen consigo una incorrecta interpretación, pues la agregación que se realiza en estos casos da a entender que existe una cantidad de elementos incorrectos. Nótese que una correcta interpretación de este diagrama es que existen hasta tres componentes estáticos, y cada uno agrega un Combo Box, un Frame y una imagen, en un total de hasta 6 elementos, lo cual no es lo que se quiere dar a entender.	Bajo
Cantidad de componentes de los diagramas que modelan con precisión estructura lógica de la navegación del sistema.	Para este indicador se analizó el DEN, en el cual no existe una representación del inicio de la navegación del sistema, lo cual disminuye la precisión del diagrama, ya que no se define como comenzar a leer el mismo. En la versión 1.5 del lenguaje esta característica se representa con el uso de un actor, sin embargo esto no es pertinente, pues existen herramientas que no permiten el uso de elementos “actor” en un diagrama de clases	Medio
Cantidad de componentes del diagrama de casos de uso modelado con ApEM-L que reflejan con precisión las características del sistema que representan.	Este diagrama según (Ciudad, 2013) es innecesario, al igual que sus descripciones, debido al uso del DEP en los artefactos del sistema.	Bajo
Cantidad de componentes de los diagramas que modelan con precisión la estructura lógica de la vista estática del sistema.	Para este indicador se analizó el diagrama de clases del diseño, en el cual no se detectaron deficiencias en cuanto a su precisión.	Alto

Posteriormente se realizó la operacionalización a la dimensión representatividad para los mismos diagramas quedando los siguientes resultados:

Tabla 9: Resultado de la operacionalización de la dimensión representatividad a ApEM-L 1.0

Indicador	Análisis	Resultado

Cantidad de componentes del DEP que representan al concepto del dominio al que están asociados.	Los componentes de este diagrama se representan a partir del uso de los estereotipos y mecanismos de extensión de UML. Sin embargo la visualización de estos componentes tiende a confundir su interpretación, lo que afecta la representatividad.	Bajo
Cantidad de componentes del DEN que representan al concepto del dominio al que están asociado.	Los componentes de este diagrama se representan a partir del uso de los estereotipos y mecanismos de extensión de UML. Sin embargo la visualización de estos componentes tiende a confundir su interpretación, lo que afecta la representatividad.	Bajo
Cantidad de componentes del diagrama de casos de uso modelado con ApEM-L que representan al concepto del dominio al que están asociados.	Los componentes de este diagrama se representan a partir del uso de los estereotipos y mecanismos de extensión de UML. Sin embargo la visualización de estos componentes tiende a confundir su interpretación, lo que afecta la representatividad.	Bajo
Cantidad de componentes del DVE que sirven para representar al concepto del dominio al que están asociado.	Los componentes de este diagrama se representan a partir del uso de los estereotipos y mecanismos de extensión de UML. Sin embargo la visualización de estos componentes tiende a confundir su interpretación, lo que afecta la representatividad.	Bajo

Luego de la aplicación de la operacionalización a los diagramas generados bajo la versión 1.0 del lenguaje ApEM-L, se volvió a realizar para los diagramas generados con la colección de plugins de Eclipse. Dicha operacionalización quedó de la siguiente manera:

Tabla 10: Resultado de la operacionalización de la dimensión precisión a ApEM-L 2.0

Indicador	Análisis	Resultado
Cantidad de componentes de los diagramas que modelan con precisión la estructura lógica de la presentación del sistema.	Las clases que representaban los componentes de la IU, dejan de ser agregaciones de las clases que eran estereotipadas con <<Estático>> e <<Interactivo>>, para entonces relacionarse directamente con la clase que los engloba (la vista). De esta forma se elimina las interpretaciones inconsistentes producto del significado de la multiplicidad entre las clases como concepto del UML.	Alto
Cantidad de componentes de los diagramas que modelan con precisión estructura lógica de la	Se realizó una representación, en forma de estereotipo decorativo, de cuál será la clase que comenzará la navegación del sistema. Por lo que la limitación del diagrama anterior se elimina.	Alto

navegación del sistema.		
Cantidad de componentes del diagrama de casos de uso modelado con ApEM-L que reflejan con precisión las características del sistema que representan.	Se realizó una representación mediante estereotipos decorativos de actores que se benefician de casos de uso específicos del sistema, los cuales son también representados mediante estereotipos decorativos.	Alto
Cantidad de componentes de los diagramas que modelan con precisión la estructura lógica de la vista estática del sistema.	A este diagrama no se le realizaron modificaciones en cuanto a su precisión, pues de la versión 1.0 no existieron inconvenientes.	Alto

Tabla 11: Resultado de la operacionalización de la dimensión representatividad de ApEM-L 2.0

Indicador	Análisis	Resultado
Cantidad de componentes del DEP que representan al concepto del dominio al que están asociados.	Los diagramas generados presentan una representación visual, la cual es representada a partir de estereotipos decorativos creados en la versión 2.0 del lenguaje (Vista, HLL, Interactivo y Estático). Por lo que el resultado ofrece interpretaciones correctas del diagrama a partir de su interpretación visual.	Alto
Cantidad de componentes del DEN que representan al concepto del dominio al que están asociado.	Los diagramas generados presentan una representación visual, la cual es representada a partir de estereotipos decorativos creados en la versión 2.0 del lenguaje (Vista, Índice, Consulta, Menú, Botón, Enlace de navegación y Auto Navegación). Por lo que el resultado ofrece interpretaciones correctas del diagrama a partir de su interpretación visual.	Alto
Cantidad de componentes del diagrama de casos de uso modelado con ApEM-L que representan al concepto del dominio al que están asociados.	Los diagramas generados presentan una representación visual, la cual es representada a partir de estereotipos decorativos creados en la versión 2.0 del lenguaje (Actor Educativo, Entorno de Presentación y Actividad Educativa). Por lo que el resultado ofrece interpretaciones correctas del diagrama a partir de su interpretación visual.	Alto
Cantidad de componentes del DVE que sirven para representar al concepto del dominio al que están asociado.	Los componentes de este diagrama se representan a partir del uso de los estereotipos oficiales creados para versión 2.0 de ApEM-L.	Alto

Luego se realizó la operacionalización a los diagramas creados con UML en la herramienta Visual Paradigm for UML, arrojando los siguientes resultados:

Tabla 12 Resultado de la operacionalización de la dimensión precisión de UML 2.0

Indicador	Análisis	Resultado
Cantidad de componentes de los diagramas que modelan con precisión la estructura lógica de la presentación del sistema.	Según (Larman, 2007) este tipo de diagrama no existe dentro de los existentes para UML, por lo que este indicador obtiene su menor valor.	Bajo
Cantidad de componentes de los diagramas que modelan con precisión estructura lógica de la navegación del sistema.	Según (Larman, 2007) este tipo de diagrama no existe dentro de los existentes para UML, por lo que este indicador obtiene su menor valor.	Bajo
Cantidad de componentes del diagrama de casos de uso modelado con ApEM-L que reflejan con precisión las características del sistema que representan.	Se aplican los mismos señalamientos que para la versión 1.0 de ApEM-L.	Bajo
Cantidad de componentes de los diagramas que modelan con precisión la estructura lógica de la vista estática del sistema.	No se representa de manera visual, las responsabilidades que tendrá cada una de las clases modeladas. Solo se puede deducir a partir de los nombres de las mismas. Por lo que sus componentes no describen suficientemente el caso de uso que modelan.	Bajo

Tabla 13 Resultado de la operacionalización de la dimensión representatividad para UML 2.0

Indicador	Análisis	Resultado
Cantidad de componentes del DEP que representan al concepto del dominio al que están asociados.	Según (Larman, 2007) este tipo de diagrama no existe dentro de los existentes para UML, por lo que este indicador obtiene su menor valor	Bajo
Cantidad de componentes del DEN que representan al concepto del dominio al que están asociado.	Según (Larman, 2007) este tipo de diagrama no existe dentro de los existentes para UML, por lo que este indicador obtiene su menor valor	Bajo

Cantidad de componentes del diagrama de casos de uso modelado con ApEM-L que representan al concepto del dominio al que están asociados.	Se aplican los mismos señalamientos que para la versión 1.0 de ApEM-L.	Bajo
Cantidad de componentes del DVE que sirven para representar al concepto del dominio al que están asociado.	Los componentes de este diagrama se representan a partir del uso de los estereotipos y mecanismos de extensión de UML. Sin embargo la visualización de estos componentes tiende a confundir su interpretación, lo que afecta la representatividad.	Bajo

Para una mejor comprensión, se tomó apoyo de las siguientes gráficas las cuales dan una muestra visual del comportamiento de las variables.

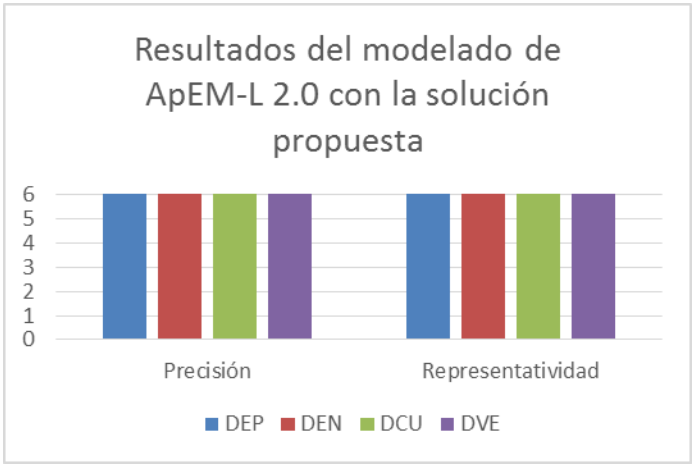

Figura 37: Resultado del modelado con ApEM-L 2.0

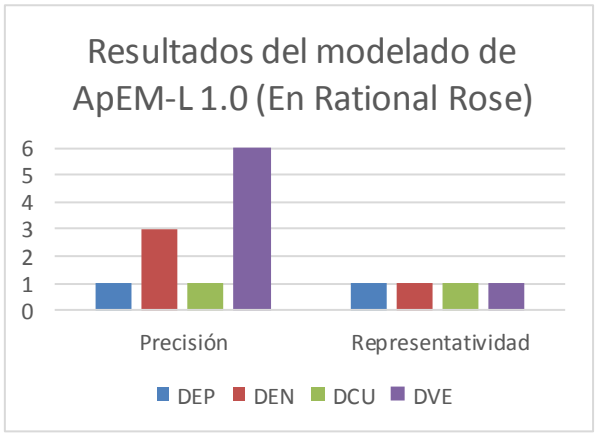

Figura 38: Resultado del modelado con ApEM-L 1.0

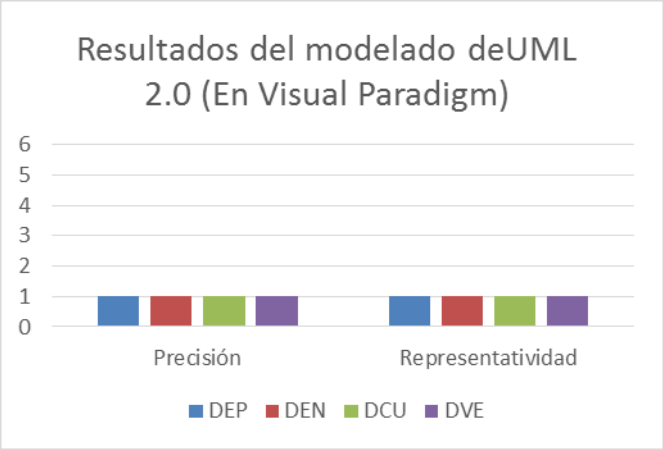

Figura 39: Resultado del modelado con UML 2.0

Con el auxilio del análisis realizado en el capítulo II de la presente investigación, acerca del comportamiento de la dimensión completitud de la variable independiente: Herramienta CASE, para el modelado de aplicaciones educativas en la UCI. Gracias al caso de estudio establecido, mediante el cual se evalúo la completitud de la solución propuesta, se procedió a realizar el siguiente análisis:

Completitud de la herramienta: Colección de plugins para el modelado de diagramas de ApEM-L 2.0.

Tabla 14: Resultado de la operacionalización de la dimensión completitud de la propuesta de solución

Indicador	Análisis	Valor
Cantidad de componentes de la herramienta que representan con completitud diagramas que modelan la estructura lógica de la presentación del sistema.	Las especificaciones de este tipo de diagrama se realizaron bajo las descripciones formales de la versión 2.0 de ApEM-L. Estas especificaciones fueron realizadas por (Herrera, 2014a), con el objetivo de *«modelar la estructura lógica de la capa de presentación de las aplicaciones educativas; específicamente la estructura de las IU, sus componentes y las características interactivas de estos últimos».* Así la herramienta soporta el metamodelo definido para este tipo de diagramas. En cuanto a los estereotipos, se usó la descripción de los estereotipos restrictivos y redefinitorios, así como las restricciones creadas para ApEML-L 2.0 por el mismo autor.	Alto

Cantidad de componentes de la herramienta que representan con completitud diagramas que modelan la estructura lógica de la navegación del sistema.	Las especificaciones de este tipo de diagrama se realizaron bajo las descripciones formales de la versión 2.0 de ApEM-L. Estas especificaciones fueron realizadas por Herrera (2014), con el objetivo de *«el objetivo de representar gráficamente la estructura navegacional de la aplicación educativa; es decir, todos aquellos componentes que intervienen o influyen en la navegación del sistema».* Así la herramienta soporta el metamodelo definido para este tipo de diagramas. En cuanto a los estereotipos, se usó la descripción de los estereotipos restrictivos y redefinitorios, así como las restricciones creadas para ApEML-L 2.0 por el mismo autor.	Alto
Cantidad de componentes de la herramienta que representan con completitud los diagramas de casos de uso del sistema.	Las especificaciones de este tipo de diagrama se realizaron bajo las descripciones formales de la versión 2.0 de ApEM-L. En dichas especificaciones se ofrecen elementos para representar usuarios determinados (ej. profesor, estudiante, tutor, etc.) que se benefician de las funcionalidades de carácter educativo. Además se presentan extensiones de los conceptos de caso de uso, para representar casos de uso específicos de carácter educativo. De esta manera la herramienta soporta el metamodelo definido para este tipo de diagramas. En cuanto a los estereotipos, se usó la descripción de los estereotipos restrictivos y redefinitorios, así como las restricciones creadas para ApEML-L 2.0 por el mismo autor.	Alto
Cantidad de componentes de la herramienta que representan con completitud los diagramas de clases del diseño del sistema.	Las especificaciones de este tipo de diagrama se realizaron bajo las descripciones formales de la versión 2.0 de ApEM-L. En esta versión se ofrece la descripción del metamodelo basado en el patrón MVC-E. Se hace uso de los estereotipos restrictivos y redefinitorios, así como las restricciones creadas para ApEML-L 2.0 por el autor mencionado anteriormente.	Alto

A continuación se muestra gráficamente el comportamiento de la dimensión completitud de la variable independiente, a partir de la operacionalización realizada.

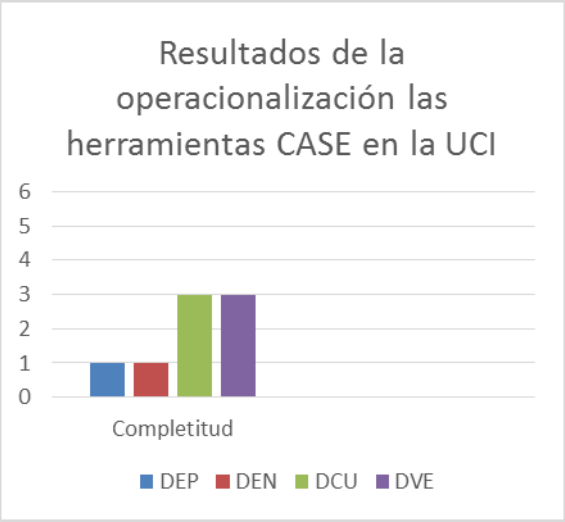

Figura 40: Resultado del modelado con las herramientas CASE en la UCI

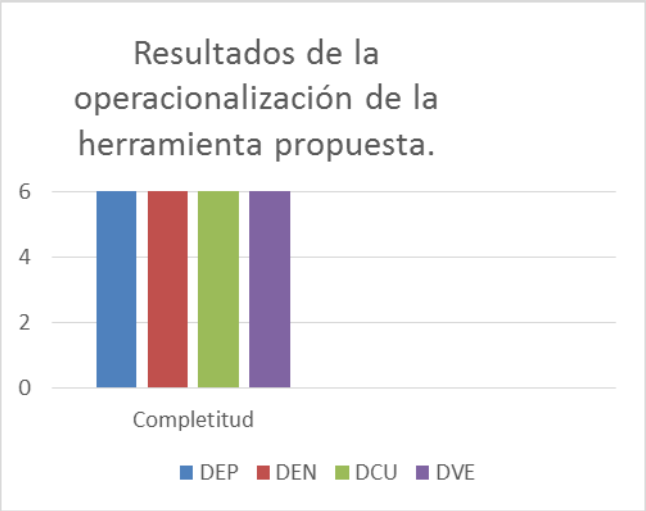

Figura 41: Resultado del modelado con la solución propuesta

Conclusiones Parciales

Una vez realizada la comparación entre las variables resultantes y las variables analizadas al principio de la investigación se concluye que:

- Se produjo un aumento en los valores de los indicadores de las dimensiones: precisión y representatividad de la variable dependiente y la completitud de la variable independiente tras el uso de la solución propuesta.
- Los resultados obtenidos en los indicadores se pueden extrapolar a los proyectos productivos para aplicaciones, si se hace uso de la solución propuesta en la investigación.
- Se cumple la hipótesis planteada en la investigación con un nivel satisfactorio, pues los valores de los indicadores alcanzan su valor máximo.

Conclusiones Finales

Llegado a este punto de la investigación se arriba a las siguientes conclusiones:

- El estudio de las herramientas CASE existentes arrojó, que ninguna de las mismas soporta el modelado con ApEM-L 2.0. Por lo que la actividad de modelado de aplicaciones educativas, mediante herramientas CASE no se puede realizar con el uso de este lenguaje. Esto hace necesario la propuesta de un generador de diagramas para una herramienta CASE que soporte ApEM-L 2.0.
- El estudio de los generadores de diagramas de herramientas CASE para el modelado de dominio específico demostró que, para la construcción de los mismo, es necesario la definición de su metamodelo, sus reglas semánticas y estereotipos. Por lo que se propuso para la implementación de la solución el uso del GMF para eclipse, ya que permite la definición de los elementos anteriormente mencionados.
- Para el desarrollo de la investigación se usó como metodología OpenUP, ya que se ajustó a las características del entorno donde se desarrolló la solución. Se usó el patrón arquitectónico MVC, que permitió separar las responsabilidades de las clases, para el procesamiento de la información necesaria en la gestión de los diagramas.
- El experimento realizado con el caso de estudio propuesto permitió comprobar el cumplimiento de la hipótesis. Lo cual demuestra el cumplimiento del objetivo de la investigación.

Recomendaciones

Una vez concluida la investigación se recomienda:

- Enriquecer el generador de diagramas construido a partir de la realización de los diagramas correspondientes al lenguaje ApEM-L 2.0, en otras de las vistas del lenguaje.
- Sugerir a la dirección del centro FORTES la utilización del generador de diagrama para el modelo de aplicaciones educativas, lo que permitirá obtener nuevas necesidades y nuevas funcionalidades a tomar en cuenta para una nueva versión de la aplicación.

Referencias bibliográficas

- **AMATRIAIN, X. y ARUMI, P**. Frameworks Generate Domain-Specific Languages: A Case Study in the Multimedia Domain. 2011, vol. 37, nº 4, p. 544 - 558. Disponible en: http://doi.ieeecomputersociety.org/10.1109/TSE.2010.48. . ISSN ISSN : 0098-5589
- **BERKENKÖTTER, K**. Using UML 2.0 in Real-Time Development. En *Bremen Alemania,* 2010.
- **CIUDAD, F. A**. *ApEM-L Lenguaje de modelado orientado a objetos.* La Habana: 2013, Disponible en: http://revista.jovenclub.cu/?p=1084.
- ---. *ApEM-L como una nueva solución a la modelación de aplicaciones educativas en la UCI.* Tesis en opción al título de Máster en Informática Aplicada, Universidad de las Ciencias Informáticas, 2007.
- ---. ENFOQUES DE LA INGENIERÍA DE SOFTWARE. En *La Habana,* 2011.
- ---. Utilización del Patrón Modelo – Vista – Controlador (MVC) en el diseño de software educativos. En *TALLER DE SOFTWARE EDUCATIVO E HIPERMEDIA*, 2006.
- CIUDAD, F. A. y HERRERA, Y. DoMet COMO PROPUESTA PARA LA MODELACIÓN DE ENTORNOS ORGANIZACIONALES COMPLEJOS Y DIFUSOS. *UCIENCIA*, 2006, nº
- ---. FUNDAMENTOS ACTUALES DE LA INGENIERÍA DEL SOFTWARE PARA LAS APLICACIONES EDUCATIVAS. *UCIENCIA*, 2012, nº
- **CRUZ, E. D. L. y GUZMÁN, D**. *Propuesta de herramienta CASE para los proyectos del Centro de Desarrollo de Informática Industrial.* Trabajo de diploma para optar por el título de ingeniero informático, Universidad de las Ciencias Informáticas (UCI), 2010.
- **DREA**. *Diccionario de la Real Academia Española* Madrid, España: de 2014]. Disponible en: http://www.rae.es/obras-academicas/diccionarios/diccionario-de-la-lengua-espanola.
- **ENGELS, G. y SAUER, S.** Extending UML for Modeling of Multimedia Applications. En *Tokyo. 13 Sep 1999-16 Sep 1999 1999*. p. 80-87.
- **FUNDIBEQ.** *Diagrama de Flujo* Disponible en: http://www.fundibeq.org/opencms/export/sites/default/PWF/downloads/gallery/methodology/tools/diagrama_de_flujo.pdf.
- **GONZALES, J.** *El discreto encanto del metamodelo de UML.* México: 1998, Disponible en: http://www.zeusconsult.com.mx/DISCRET.pdf.

- **HERRERA, Y.** Tesis en opción al título de Máster en Informática Aplicada, Universidad de las Ciencias Informáticas UCI, 2014a.
- ---. Tesis en opción al título de Máster en Informática Aplicada. 2014b
- **LARMAN, C.** *UML y Patrones, Introducción al Análisis y Diseño Orientado a Objetos.* Traducido por: Valle, B. M.; 2da Edición ed.: Pearson Pretenci Hall, 2007.
- **LOPEZ; SILVA, C***., et al. Extending ArgoUML for real time UML.* St Thomas, US Virgin Island: 2004,
- **LOPEZ, P. G.; LÁZARO, J. A. G***., et al.* Herramientas CASE ¿Cómo incorporarlas con éxito en nuestra organización? *Universidad de Castilla-La Mancha.*, 1999, nº
- **LLANES, L. y ISLA., Y. L**. *Metamodelado para la construcción de software en entornos libre.* Trabajo de Diploma para optar por el título de Ingeniero Informático., Universidad de las Ciencias Informáticas UCI, 2008.
- **MONTENEGRO; GAONA, P. A.,** *et al. APLICACIÓN DE INGENIERÍA DIRIGIDA POR MODELOS (MDA), PARA LA CONSTRUCCIÓN DE UNA HERRAMIENTA DE MODELADO DE DOMINIO ESPECÍFICO (DSM) Y LA CREACIÓN DE MÓDULOS EN SISTEMAS DE GESTIÓN DE APRENDIZAJE (LMS) INDEPENDIENTES DE LA PLATAFORMA.* [Científico]. Colombia: publicado el: Oct 2011 de 2011a, última actualización: Oct 2011. vol. 78, ISBN ISSN 0012-7353.
- **MONTENEGRO, C. E.; ALONSO, P***., et al. Herramienta de modelado de dominio específico (DSM).* [Científico]. Bogotá, Colombia: 2011b, ISBN ISSN0124 2253.
- **MUÑOZ, L. y RICARDO., Y.** *Introducción de ApEM-L 1.0 en proyectos productivos de la UCI. Valoración de resultados.* Trabajo de diploma para optar por el título de ingeniero informático., Universidad de las Ciencias Informáticas UCI, 2008.
- **OMG**. *Diagram Definition.* 01-07-2012 de 2012
- **PAVÓN, J.** Estructura de las Aplicaciones Orientadas a Objetos El patrón Modelo-Vista-Controlador (MVC). En *Facultad de Informática UCM,* 2009.
- **PEDRAJAS, A. P**. *APLICACIONES DE LAS TECNOLOGÍAS DE LA INFORMACIÓN Y DE LA COMUNICIÓN EN LA EDUCACIÓN CIENTÍFICA.* Córdoba, España: 2005, ISBN ISSN 1697-011X.
- **POLLICE, G.; AUGUSTINE, L.***, et al. Software Development for Small Projects. A RUP-centric approach.* Boston, USA: Addison-Wesley Professional, 2003. The Addison-Wesley Object Technology Series. ISBN ISBN-13: 978-0321199508.

- **PONS, C.; GIANDINI, R.*, et al.*** *Desarrollo de Software Dirigido por Modelos.* Buenos Aires, Argentina: Editorial de la Universidad de la Plata, 2010. ISBN ISBN 978-950-34-0630-4.
- **PORTILLO, J. B.** *Entorno Multidisciplinar para el Desarrollo de Sistemas de Control Distribuido.* Tesis Doctoral, U.P.V. / E.H.U, 2004.
- **PRESSMAN, R. S**. *Ingeniería del Software. Un enfoque práctico.* 6ta Edición. ed.: Mc Graw Hill, 2005. ISBN ISBN: 9789701054734.
- **ROJAS, T.; PEREZ, M.*, et al*.** Modelo de decisión para soportar la selección de una herramienta CASE. *Universidad Simón Bolívar. Caracas - Venezuela*, 2008, nº
- **RUMBAUGH, J.; JACOBSON, I.*, et al*.** *EL LENGUAJE UNIFIADO DE MODELADO. MANUAL DE REFERNCIA. .* 2DA EDICIÓN ed. Madrid: Pearson, 2007. ISBN ISBN: 978-84-7829-087-1.
- **SAMPIERI, R.; COLLADO, C**.*, et al. Metodología de la investigación.* Iztapalapa Mexico D.F: McGRAW HILL ITERAMERICANA, 2006. ISBN ISBN: 970-10-3632-8.
- **SANTIAGO, R.; CECILIA, H.*, et al*.** APLICACIÓN DE LA METODOLOGIA OPENUP EN EL DESARROLLO DEL SISTEMA DE DIFUSIÓN DE GESTIÓN DEL CONOCIMIENTO DE LA ESPE. 2013, nº
- **SOMERVILLE, I**. *Ingeniería de Software.* 7ma EDICION ed. Madrid España: Pearson Addison Wesley, 2005. ISBN ISBN: 84-7829-074-5.
- **ZAPATA, C. M.; ARANGO, F. J**.*, et al. Estudio comparativo de las herramientas MetaCASE bajo consistencia y refinamiento.* [Científico]. Colombia: Redalyc (Red de Revistas Científicas de América Latina,El Caribe, España y Portugal) 2007, vol. 43,

I

Anexo I: Definición operacional de las variables de la investigación

Variable Dependiente: diagramas resultantes de la actividad de modelado.

Dimensiones:

- Precisión: capacidad que debe poseer el modelo de proveer una representación genuina de las características del sistema en interés.
- Representatividad: capacidad del artefacto y sus componentes de representar las características del software según el propósito por el cual fueron creados.

Tabla 15: Definición operacional de la variable dependiente de la investigación

Dimensión	Indicador	Escala de valor
Precisión	Cantidad de componentes de los diagramas que modelan con precisión la estructura lógica de la presentación del sistema.	Alto: Entre el 70% y el 100% de los componentes, reflejan con mayor precisión el dominio que representan. Medio: Entre el 70% y el 100% de los componentes, reflejan con mediana precisión el dominio que representan. Bajo: Entre el 70% y el 100% de los componentes, reflejan con poca precisión el dominio que representan. Nulo: Entre el 70% y el 100% de los componentes del DEP, no reflejan con precisión el dominio que representan.
	Cantidad de componentes de los diagramas que modelación precisión la estructura lógica de la navegación del sistema.	Alto: Entre el 70% y el 100% de los componentes del DEN, reflejan con mayor precisión el dominio que representan. Medio: Entre el 70% y el 100% de los componentes del DEN, reflejan con mediana precisión el dominio que representan. Bajo: Entre el 70% y el 100% de los componentes del DEN, reflejan con poca precisión el dominio que representan. Nulo: Entre el 70% y el 100% de los componentes del DEN, no reflejan con precisión el dominio que representan.
	Cantidad de componentes del diagrama de casos de uso modelado con ApEM-L reflejan con precisión la característica del sistema que representan.	Alto: Entre el 70% y el 100% de los componentes del diagrama de casos de uso que se modela con ApEM-L, reflejan con mayor precisión el dominio que representan. Medio: Entre el 70% y el 100% de los componentes del diagrama de casos de uso que se modela con ApEM-L, reflejan con mediana precisión el dominio que representan. Bajo: Entre el 70% y el 100% de los componentes del diagrama de casos de uso que se modela con ApEM-L, reflejan con poca precisión el dominio que representan. Nulo: Entre el 70% y el 100% de los componentes del diagrama de casos de uso que se modela con ApEM-L, no reflejan con precisión el dominio que representan.
	Cantidad de componentes	Alto: Entre el 70% y el 100% de los componentes del DVE,

Dimensión	Indicador	Escala de valor
	de los diagramas que modelan con precisión la estructura lógica de la vista estática del sistema.	reflejan con mayor precisión el dominio que representan. Medio: Entre el 70% y el 100% de los componentes del DVE, reflejan con mediana precisión el dominio que representan. Bajo: Entre el 70% y el 100% de los componentes del DVE, reflejan con poca precisión el dominio que representan. Nulo: Entre el 70% y el 100% de los componentes del DVE, no reflejan con precisión el dominio que representan.
Representatividad	Cantidad de componentes del DEP que representan al concepto del dominio al que están asociados.	Alto: Entre el 70% y el 100% de los componentes del DEP, representar al concepto del dominio al que están asociados. Medio: Entre el 69% y el 30% de los componentes del DEP, sirven bastante para representar al concepto del dominio al que están asociados. Bajo: Entre el 29% y el 10% de los componentes del DEP, sirven para representar al concepto del dominio al que están asociados. Nulo: Menos del 10% de los componentes del DEP, sirven para representar al concepto del dominio al que están asociados.
	Cantidad de componentes del DEN que representan al concepto del dominio al que están asociado.	Alto: Entre el 70% y el 100% de los componentes del DEN, representar al concepto del dominio al que están asociados. Medio: Entre el 69% y el 30% de los componentes del DEN, sirven bastante para representar al concepto del dominio al que están asociados. Bajo: Entre el 29% y el 10% de los componentes del DEN, sirven para representar al concepto del dominio al que están asociados. Nulo: Menos del 10% de los componentes del DEN, sirven para representar al concepto del dominio al que están asociados.
	Cantidad de componentes del diagrama de casos de uso modelado con ApEM-L que representan al concepto del dominio al que están asociados.	Alto: Entre el 70% y el 100% de los componentes del diagrama de casos de uso modelado con ApEM-L, representar al concepto del dominio al que están asociados. Medio: Entre el 69% y el 30% de los componentes del diagrama de casos de uso modelado con ApEM-L, sirven bastante para representar al concepto del dominio al que están asociados.

Dimensión	Indicador	Escala de valor
		Bajo: Entre el 29% y el 10% de los componentes del diagrama de casos de uso modelado con ApEM-L, sirven para representar al concepto del dominio al que están asociados. Nulo: Menos del 10% de los componentes del diagrama de casos de uso modelado con ApEM-L, sirven para representar al concepto del dominio al que están asociados.
	Cantidad de componentes del DVE que sirven para representar al concepto del dominio al que están asociado.	Alto: Entre el 70% y el 100% de los componentes del DVE, representar al concepto del dominio al que están asociados. Medio: Entre el 69% y el 30% de los componentes del DVE, sirven bastante para representar al concepto del dominio al que están asociados. Bajo: Entre el 29% y el 10% de los componentes del DVE, sirven para representar al concepto del dominio al que están asociados. Nulo: Menos del 10% de los componentes del DVE, sirven para representar al concepto del dominio al que están asociados.

Variable Independiente: herramientas CASE en la actividad de modelado de aplicaciones educativas.

Dimensiones:

- Completitud:

Capacidad de la herramienta de cubrir o soportar a través de su metamodelo todas las necesidades de modelado de aplicaciones educativas y de multimedia cubanas.

Capacidad de los estereotipos del lenguaje, que modela la herramienta, de representar las características de aplicaciones educativas y de multimedia cubanas.

Capacidad de las restricciones del lenguaje, que modela la herramienta, de representar las características de aplicaciones educativas y de multimedia cubanas.

Tabla 16: Definición operacional de la variable independiente de la investigación

Dimensión	Indicador	Escala de valor
Completitud	Cantidad de componentes de la herramienta (metamodelo, estereotipo y restricciones) que representan con completitud diagramas que	Alto: todos los componentes cubre las necesidades de las aplicaciones Medio: 2/3 de los componentes cubre las

	modelan la estructura lógica de la presentación del sistema.	necesidades de las aplicaciones Bajo: 1/3 de los componentes cubre las necesidades de las aplicaciones Nulo: No existe la definición de los componentes para el dominio.
	Cantidad de componentes de la herramienta (metamodelo, estereotipo y restricciones que representan con completitud diagramas que modelan la estructura lógica de la navegación del sistema.	Alto: todos los componentes cubre las necesidades de las aplicaciones Medio: 2/3 de los componentes cubre las necesidades de las aplicaciones Bajo: 1/3 de los componentes cubre las necesidades de las aplicaciones Nulo: No existe la definición de los componentes para el dominio.
	Cantidad de componentes de la herramienta (metamodelo, estereotipo y restricciones que representan con completitud los diagramas de casos de uso del sistema.	Alto: todos los componentes cubre las necesidades de las aplicaciones Medio: 2/3 de los componentes cubre las necesidades de las aplicaciones Bajo: 1/3 de los componentes cubre las necesidades de las aplicaciones Nulo: No existe la definición de los componentes para el dominio.
	Cantidad de componentes de la herramienta (metamodelo, estereotipo y restricciones) que representan con completitud diagramas que modelan la estructura lógica de la vista estática del sistema.	Alto: todos los componentes cubre las necesidades de las aplicaciones Medio: 2/3 de los componentes cubre las necesidades de las aplicaciones Bajo: 1/3 de los componentes cubre las necesidades de las aplicaciones Nulo: No existe la definición de los componentes para el dominio.

l

Anexo II: Diagramas de clases del diseño de cada plugin

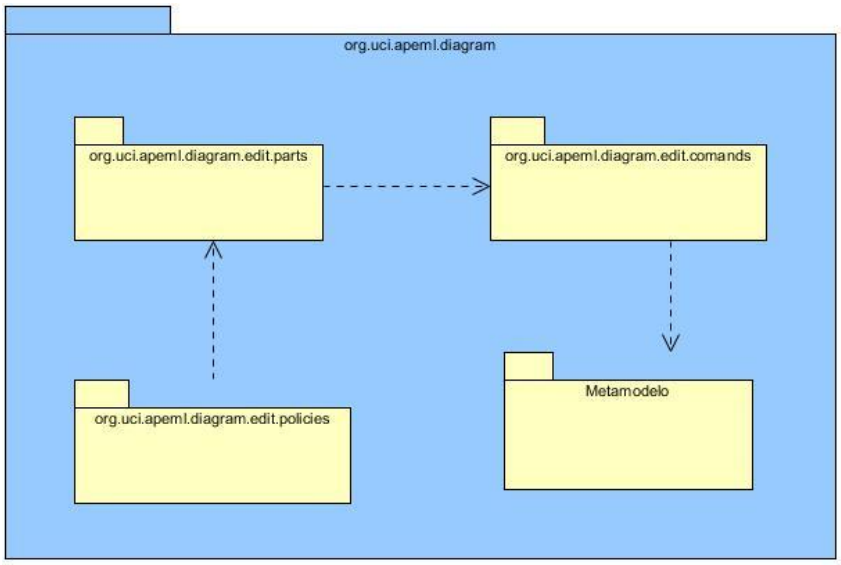

Figura 42: Estructura de la funcionalidad Dibujar diagramas.

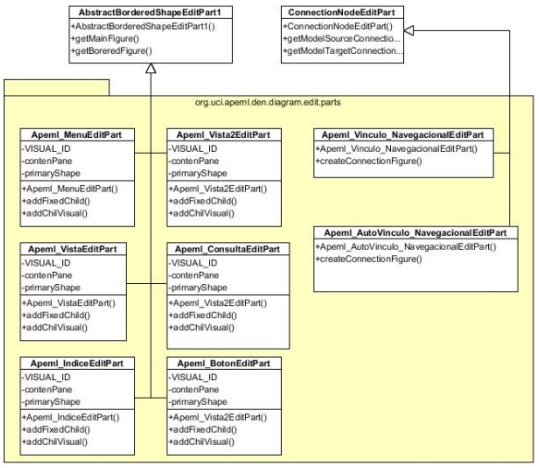

Figura 43: Paquete DEN.editpart**.**

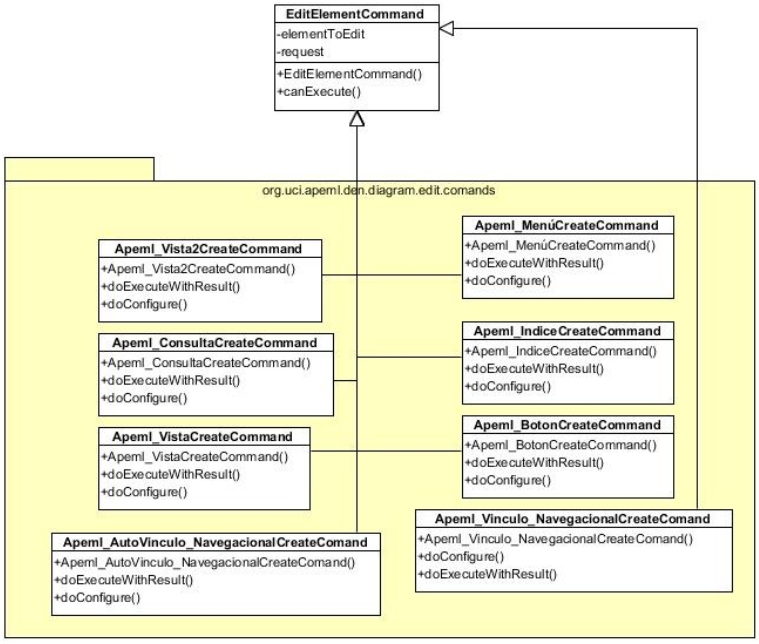

Figura 44: Paquete DEN.editcomands

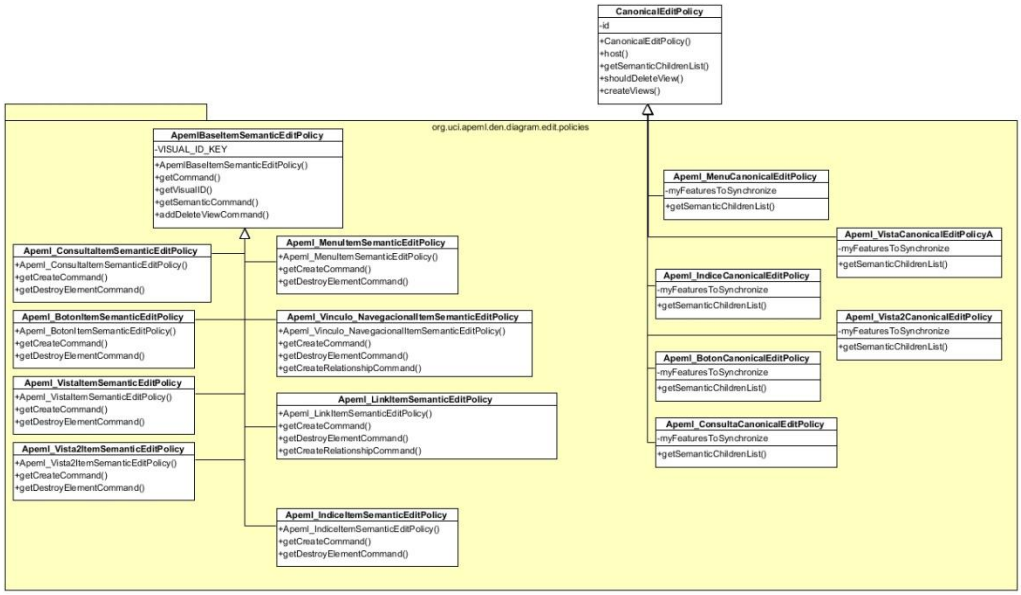

Figura 45: Paquete DEN.editpolicies

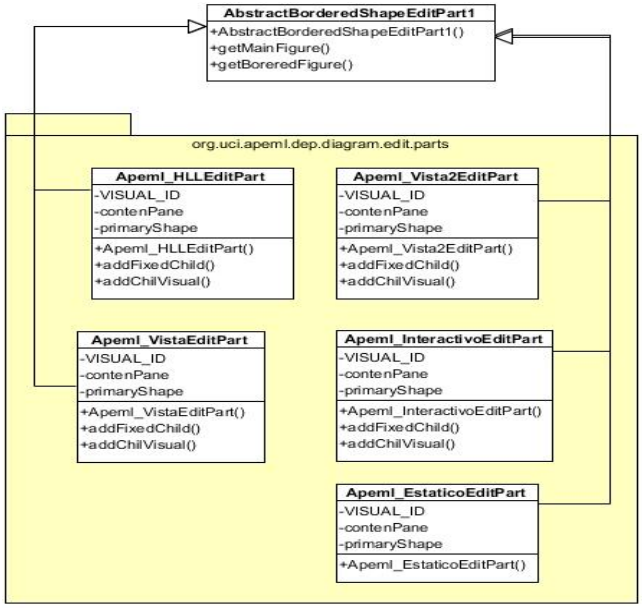

Figura 46*:* Paquete DEP.editpart.

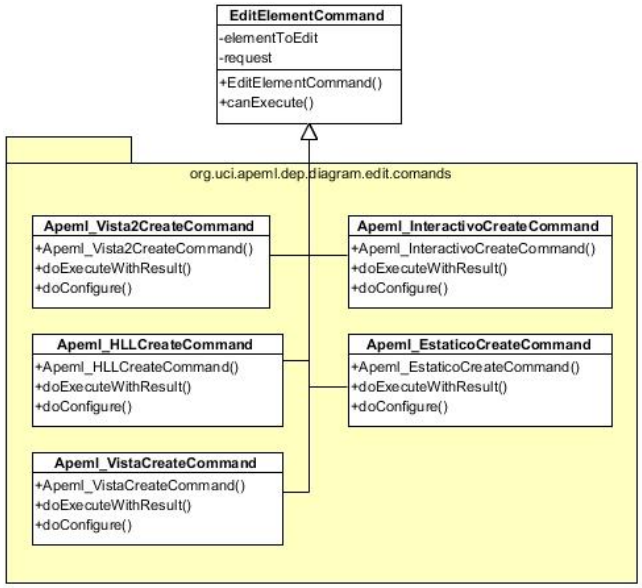

Figura 47: Paquete DEP.editcomands.

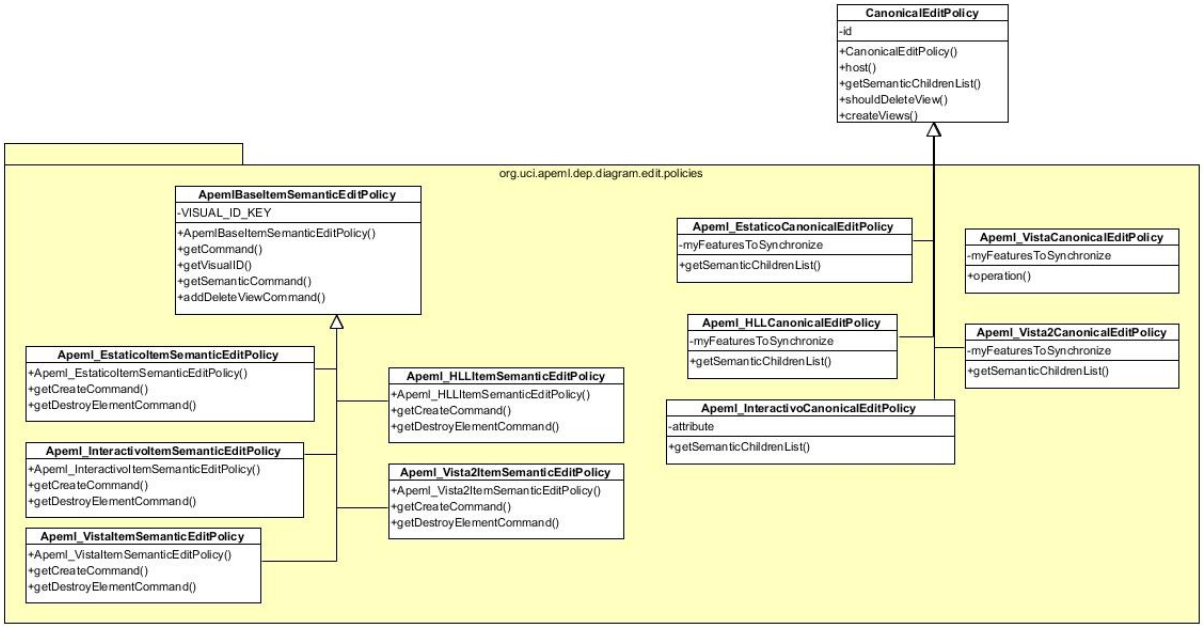

Figura 48: Paquete DEP.editpolicies.

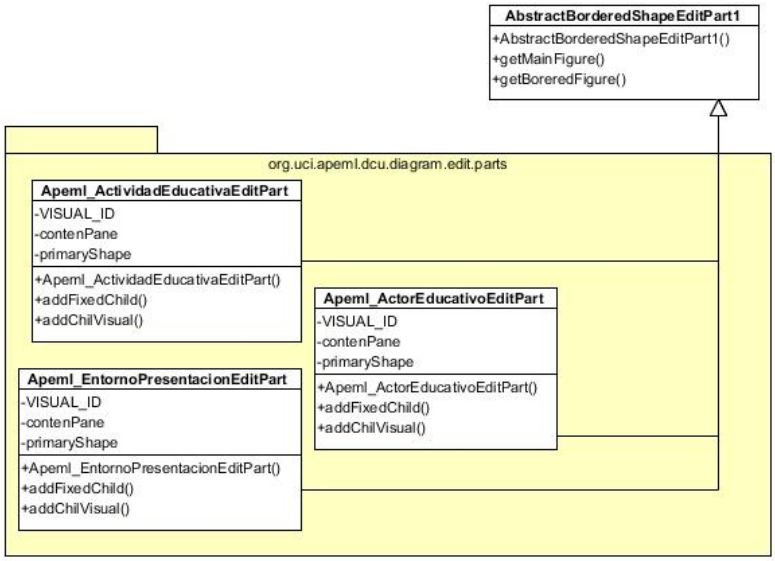

Figura 49: Paquete DCU.editpart.

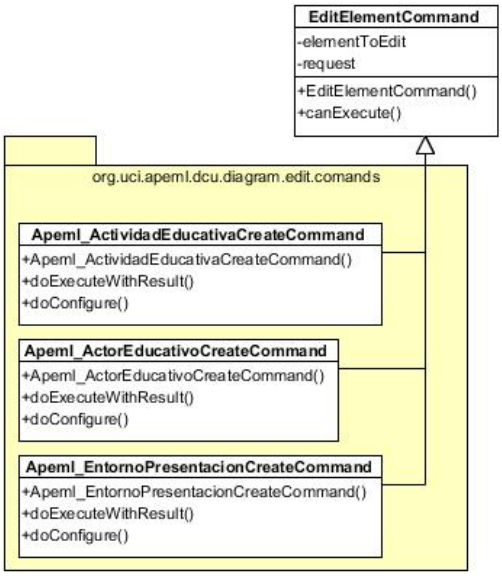

Figura 50: Paquete DCU.editcomand**.**

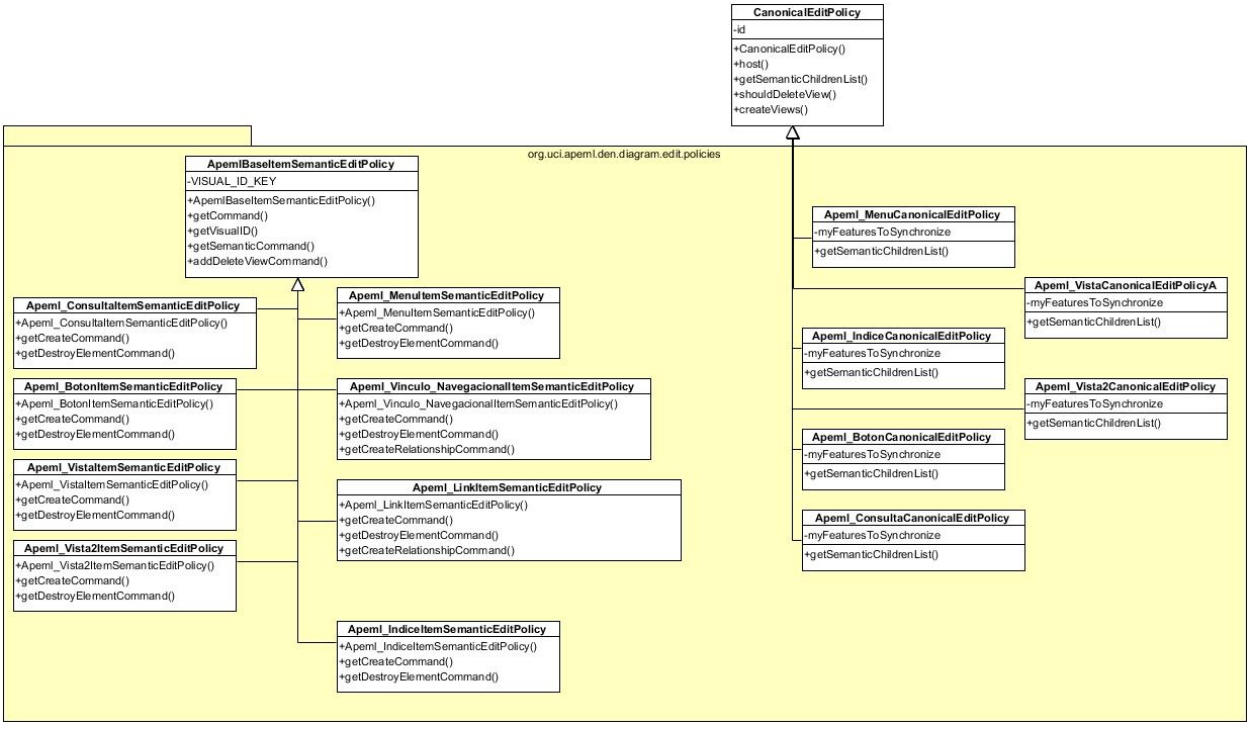

Figura 51: Paquete DCU.editpolicies.

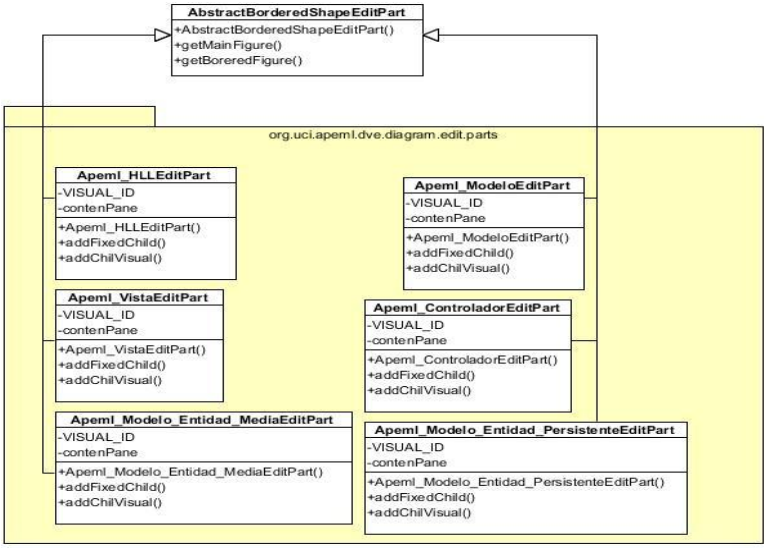

Figura 52: Paquete DVE.editpart

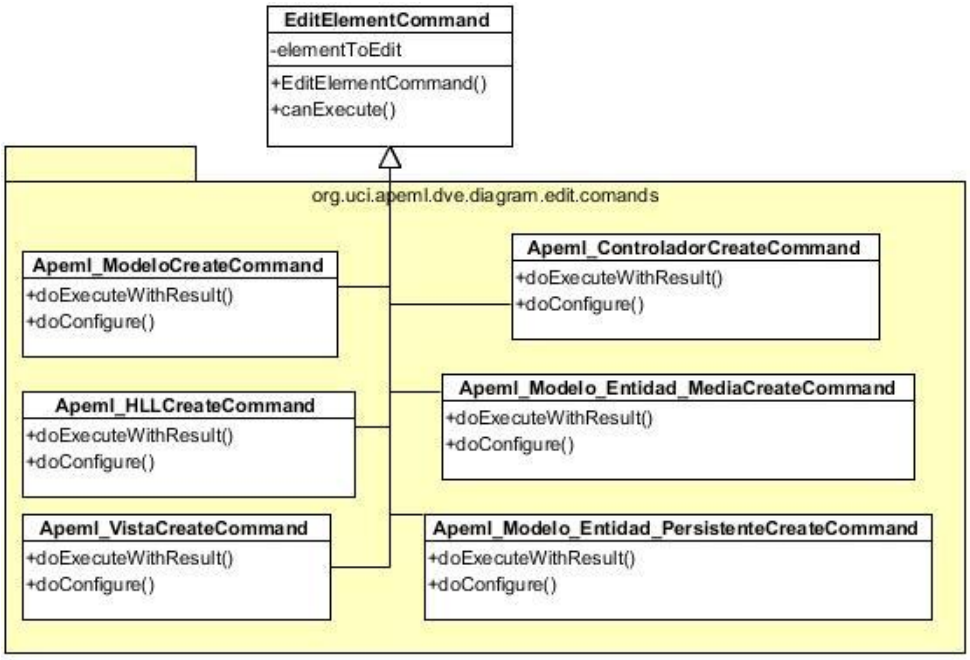

Figura 53: Paquete DVE.editcomand

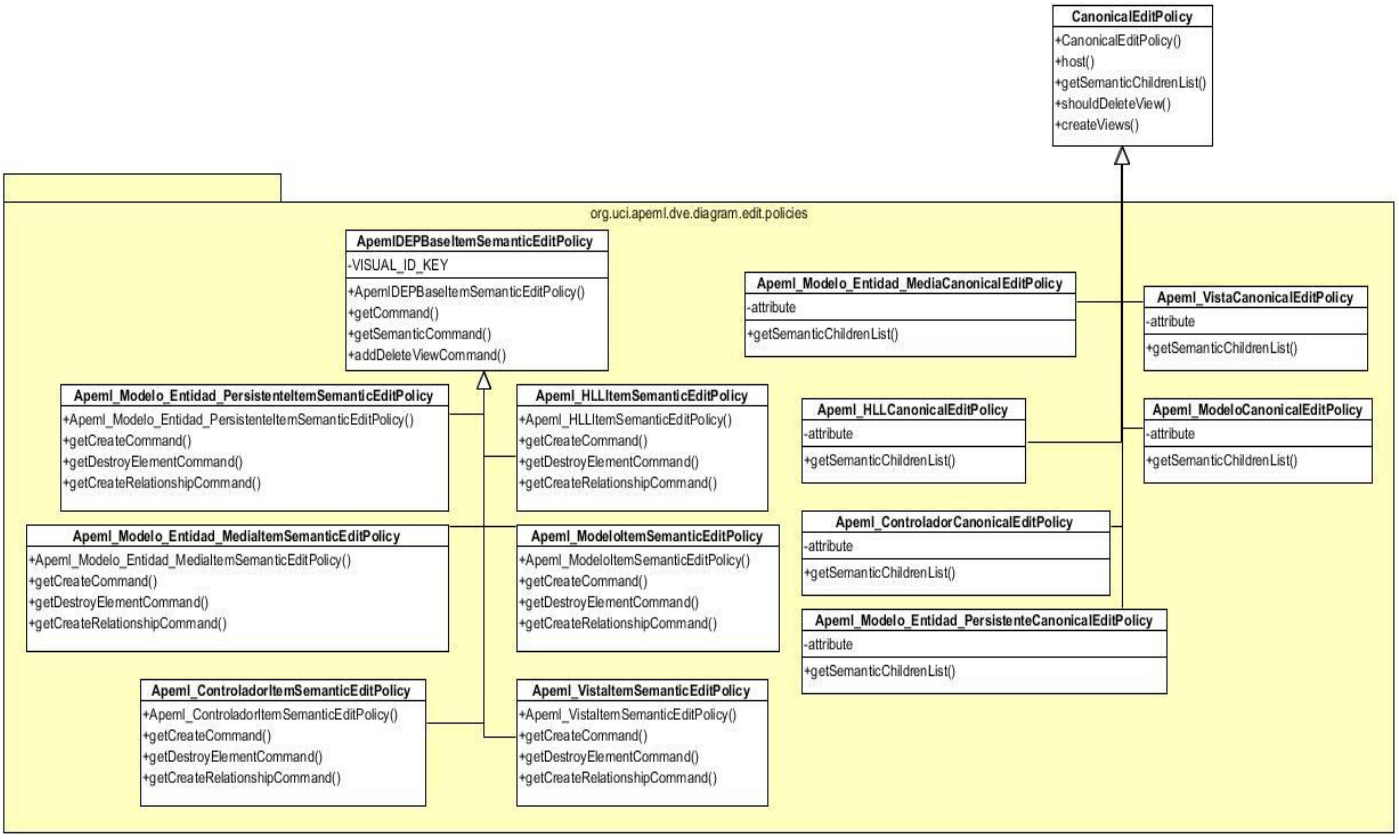

Figura 54: Paquete DVE.editpolicies

Anexo III Descripción de los casos de usos Presentar Software y Utilizar Modelo realizada por (Ciudad, 2007)

Tabla 17: Descripción Textual del Caso de Uso: Presentar Software

Descripción Textual del Caso de Uso: *Presentar Software*			
Actores del caso de uso	Usuario (inicia)		
Propósito	El caso de uso tiene como propósito permitir al usuario ejecutar la aplicación y visualizar la presentación de la misma**.**		
Resumen	El caso de uso se inicia cuando el usuario pulsa sobre el ejecutable del programa, se verifica que no esté ejecutándose otra instancia del mismo, se verifican los requerimientos de hardware necesarios, se visualiza el video de presentación y culmina el caso de uso cuando se presenta el menú principal.		
Casos de uso asociados			
Referencias	R1, R2, R3		
Precondiciones.	• Otra instancia del caso de uso no puede estarse ejecutando		
Poscondiciones	• Menú principal de la aplicación mostrado.		
Curso Normal de los Eventos			
Acciones del Actor		**Respuesta del Sistema**	
1. El usuario pulsa el fichero ejecutable del software educativo.		1.1. El sistema verifica que no se esté ejecutando otra instancia del caso del caso de uso. 1.2. Si la respuesta es negativa, se verifican los requerimientos no funcionales de video, memoria y velocidad de procesamiento. 1.3. Si la verificación es positiva se muestra el video de presentación de la aplicación. 1.4. Al concluir el video de presentación, se muestra el menú principal de la aplicación concluyendo el caso de uso.	
Cursos Alternos de los Eventos			
Acción	**Curso Alterno**		
1.2	Si se está ejecutando otra instancia del caso de uso, se le muestra un mensaje de error al usuario con la información y se culmina el caso de uso		
1.3	Si la verificación no es satisfactoria se le presenta un mensaje de error al usuario, informando que la estación no tiene los requerimientos mínimos para la ejecución del software y se culmina el caso de uso.		
Prioridad	Crítica		
Mejoras			
Medias a utilizar	**Tipo de Media**	**Descripción**	**Estado**

	Imagen	Fondo del menú principal	Existente
		Icono representativo de cada opción del menú.	En construcción
		Iconos representativos de las opciones estándares en la aplicación	En construcción
	Video o Animación	Video de presentación del software	En construcción
	Sonido	Sonido de pequeña duración para cuando se pase por encima de las opciones del menú principal.	En localización
		Sonido para cuando se seleccione una opción del menú principal	En localización
	Texto	Textos de las opciones	En construcción
		Texto de bienvenida al software	En construcción
Elementos pedagógicos	No puede ser posible que el usuario de la aplicación tenga la posibilidad de abandonar el video de presentación, debido a que en él se explican las características principales de las mismas y su forma de funcionamiento.		

Tabla 18: Descripción Textual del Caso de Uso: Utilizar Modelo

Descripción Textual del Caso de Uso: *Utilizar Modelo.*	
Actores del caso de uso	Usuario (inicia)
Propósito	El caso de uso tiene como propósito permitir al usuario desarrollar la tarea número 1 del nivel 4 del software donde el objetivo fundamental es hacer click y arrastrar el mouse.
Resumen	El caso de uso se inicia cuando el usuario selecciona el nivel 4, y dentro de este la tarea No 1 y dentro de esta el Ejercicio 1, luego el usuario se enfrentará a un conjunto de indicaciones donde las acciones principales son selección con el click y arrastrar los objetos hasta un punto determinado, donde si están correctos es satisfactorio y culmina el caso de uso.
Casos de uso asociados	
Referencias	R5, R6, R7, R8.
Precondiciones.	• Niveles previos vencidos. • El usuario ha seleccionado Nivel 4, Tarea 1, Ejercicio 1.
Poscondiciones	• El usuario ha avanzado hacia la tarea siguiente la cual es mostrada en pantalla.
Curso Normal de los Eventos	

Acciones del Actor	Respuesta del Sistema
1. El usuario selecciona la opción Nivel 4. 2. El usuario selecciona la Tarea 1. 3. El usuario selecciona el Ejercicio 1. 4. El usuario hace click sobre uno de los elementos según la carta de orientación. 5. El usuario arrastra el objeto hasta el ómnibus y suelta el botón del ratón. 6. El usuario vuelve a la acción 4 para seleccionar nuevamente un objeto según la carta de orientación.	1.1. El sistema muestra un listado con las tareas de este nivel. 2.1. El sistema muestra el listado de los ejercicios disponibles 3.1. Aparece en el borde superior derecho de la pantalla la mascota. 3.2. Se muestra en la parte inferior derecha la carta de orientación. 3.3. Se muestra el árbol en la parte inferior izquierda, la fuente en el centro y el banco en la parte superior derecha. 4.1. Se verifica que el objeto seleccionado corresponda con el establecido por la carta de orientación. 4.2. Si corresponde, el objeto se ilumina. 5.1. Se verifica que el objeto haya llegado hasta el ómnibus. 5.2. Si llegó el objeto al ómnibus, este se ilumina. 5.3. Se verifica que queden objetos a seleccionar por el usuario. 5.4. Si quedan objetos por seleccionar el usuario pasa a la acción 6. 6.1. Se repiten las acciones del sistema desde la 4.1 hasta la 5.3

Cursos Alternos de los Eventos

Acción	Curso Alterno
4.2	Si no corresponde el objeto con el establecido en la carta de orientación, los elementos palidecen y la mascota vuelve a atrás nuevamente, retornando el caso de uso a la

	acción 3.3		
5.2	Si el objeto no ha llegado al ómnibus, quiere decir que el usuario no completó la tarea, por lo que este vuelve a su posición original y el caso de uso a la tarea 3.3		
5.4	Si no quedan objetos por seleccionar, se visualiza la animación del ómnibus hacia el campismo, y la opción en el borde inferior derecho de seguir al próximo ejercicio; culminando de esta forma el caso de uso.		
Prioridad	Secundaria		
Mejoras			
Medias a utilizar	**Tipo de Media**	**Descripción**	**Estado**
	Imagen	Carta de orientación (contará con 3 elementos: árbol, fuente y banco)	En construcción
		Ómnibus.	Existente
		Árbol	Existente
		Banco	Existente
		Fuente	Existente
		Campismo	Existente
	Video o Animación	Video del ómnibus hacia el campismo.	En construcción
	Sonido	Parlamento de la mascota con la explicación inicial del ejercicio	En construcción
		Sonido grave que identifique error en la acción cometida	En construcción
	Texto	Orientaciones para la carta de guía del ejercicio	En construcción
Elementos pedagógicos	Tener en cuenta que el ómnibus correcto es el más cercano al banco.		

Glosario de términos

Clase: Descriptor de un conjunto de objetos que comparten los mismos atributos., operaciones, métodos, relaciones y comportamientos. Una clase representa un concepto dentro del sistema que se está modelando.

Componente: Una parte física reemplazable de un sistema que empaqueta su implementación, y es conforme a un conjunto de interfaces a las que proporciona su realización.

Metamodelo: Un metamodelo es un modelo que define el lenguaje para expresar un modelo.

Meta-metamodelo: Se refiere al modelo que define el lenguaje para expresar y describir un metamodelo.

Metodologías: Se refiere a los métodos de investigación en una ciencia. Se entiende como la parte del proceso de investigación que permite sistematizar los métodos y las técnicas para llevarla a cabo.

Multiplataforma: Término usado para referirse a la característica de aplicaciones de ejecutarse en diversas plataformas o sistemas operativos

Paquete: Término que denota un mecanismo de propósito general para organizar en grupos los elementos. Se pueden anidar dentro de otros paquetes, y en él pueden aparecer tanto elementos del modelo como diagramas.

Software: Es un término genérico que designa al conjunto de programas de distintos tipos (sistema operativo y aplicaciones diversas) que hacen posible operar con el ordenador.