Documento de Proyecto de Grado

Diagramas Web: Aplicación web basada en MDE para la Construcción de Modelos.

Edgar David Sandoval Giraldo

Universidad De Los Andes
Facultad De Ingeniería
Ingeniería de Sistemas y Computación
Bogotá
2014

TABLA DE CONTENIDO

1. Introducción.

Actualmente, la herramienta más poderosa de comunicación de ideas es el modelo. Para la construcción de estos, existen innumerables herramientas para su desarrollo. Unas pocas de estas herramientas están basadas en diferentes plataformas y frameworks que manejan la lógica y la semántica de los modelos de manera particular. Por otro lado, la gran mayoría simplemente se limitan a la construcción de modelos gráficamente (dibujar sus elementos en un lienzo), produciendo modelos semánticamente incorrectos.
Las pocas herramientas que manejan la lógica y semántica de un modelo están basadas en un framework del popular IDE de desarrollo eclipse llamado GMF (Graphical Modeling framework) [8], el cual se basa una herramienta llamada EUGENIA. EUGENIA permite crear dinámicamente un editor de modelos según la especificación de un metamodelo [5], el cual, valida sintácticamente el modelo construido. El problema de este editor es la absoluta dependencia a ese framework, haciéndola pesada y únicamente utilizable en eclipse.

Por tal motivo, se hace necesaria una herramienta de creación y edición de modelos liviana, y capaz de validar la sintaxis de los mismos, utilizando su respectivo metamodelo como referencia. En este documento se describirá, en primera instancia, los elementos utilizados para desarrollar la herramienta, luego los aspectos más importantes que llevan a la herramienta a cumplir sus objetivos, y por último, ejemplos que demuestran el funcionamiento de la misma.

1.1. Resumen.

Debido a la baja cantidad de editores de modelos flexibles y livianos que validan los mismos, se realizó este proyecto, el cual dió origen a Diagramas web. Esta herramienta comparte el potencial que posee el framework de modelos de eclipse (EMF) , Gracot y Eugenia, permitiendo realizar modelos sintácticamente correctos utilizando como referencia un metamodelo, y el desarrollo de múltiples sintaxis gráficas para un solo modelo en particular. La forma en que Diagramas web soluciona estos problemas es aprovechando al máximo las características que ofrecen las tecnologías web html5 y javascript, las cuales, entretejidas en una arquitectura por componentes logran un desempeño óptimo que equivale al de las herramientas tradicionales.

En cuanto al manejo de los modelos, Diagramas web permite persistirlos en su formato nativo (.mpicture) para ser editados en cualquier momento del tiempo, y pueden ser exportados a XMI (formato de modelos de EMF) para asegurar integración con EMF. También aprovecha las librerias gráficas JqueryUI y canvas (propia de html5) para dibujar notaciones gráficas personalizadas, las cuales dependen de un archivo xml denominado 'descriptor de visualización'. Éste define tanto la paleta de elementos dibujables en el editor como las formas de presentación de cada posible instancia de un metamodelo. La forma en la que se define la visualización de cada instancia se realiza combinando 4 formas nativas que ofrece la herramienta (Box2, Textbox, FlowBox,Link) que en conjunto, permiten el desarrollo de las formas de visualización más complejas imaginables. El componente encargado de todo lo

descrito anteriormente es el 'webpict.ui.js', el cual maneja todas las instancias utilizando un árbol, facilitando enormemente la validación de contenencias y el numero de instancias contenidas, de acuerdo con un metamodelo.
Para la validación de los modelos, el componente 'webpict.mmparser.js' realiza el trabajo de evaluar el modelo sintácticamente, a través de un metamodelo ecore del framework EMF que éste carga nativamente. Actualmente se validan 8 reglas lingüísticas y ontológicas básicas, entre las cuales se encuentran el correcto diligenciamiento de tipos de atributo y la correcta contenencia.

Para evaluar el potencial de la herramienta, se realizaron ejemplos con la vista de funciones de ArchiMate y uml, los cuales se validan sintácticamente de manera correcta respecto a un metamodelo, y se dibujan según un descriptor gráfico definido por el usuario.

1.2. Objetivos.

Los objetivos del proyecto de grado son los siguientes:

- Realizar un editor gráfico de modelos totalmente web.
- Utilizar diferentes metamodelos compatibles con ECORE (forma de representar metamodelos en EMF).
- Crear y utilizar diferentes sintaxis gráficas para cada metamodelo.
- Realizar validaciones sintácticas de los modelos respecto a los metamodelos.

1.3. Motivación.

Actualmente la mayoría de herramientas que manejan la estructura de un modelo están basadas en un framework llamado GMF (graphical Modeling Framework). Sobre ese está construido EUGENIA, que permite crear dinámicamente un editor gráfico de modelos según la especificación de un metamodelo [5]. El problema de este editor es que es totalmente dependiente de GMF y Eclipse. Diagramas Web, gracias a su total desarrollo en web/HTML5 Javascript nace para crear modelos bien formados de manera fácil, rápida y sin tener que relizar una instalación.

1.4. Resultados Esperados.

Con este proyecto se busca tener una herramienta de creación rápida y flexible de modelos de cualquier tipo. Los modelos resultado deben ser lógicamente y semánticamente correctos. Además, la definición de los elementos gráficos que representan el modelo, debe hacerse de manera fácil, y produciendo elementos gráficos ricos y personalizados.

2. Marco de Referencia.

Durante la realización de Diagramas Web, se tuvieron en cuenta los conceptos de la ingeniería orientada a modelos y el metamodelo, basados en el framework de modelos de Eclipse EMF. También se presentan algunos editores que cumplen los objetivos de este proyecto, como Eugenia y Gracot.

2.1. La Ingeniería Orientada a Modelos.

Se trata de un paradigma de desarrollo de software que se centra en la creación y explotación de modelos de dominio (es decir, representaciones abstractas de los conocimientos y actividades que rigen un dominio de aplicación particular), más que en conceptos informáticos (o algoritmos). Cuando se refieren a este término suelen asociarlo con complementos de eclipse, más específicamente EMF [1].

El enfoque MDE tiene por objeto aumentar la productividad mediante la maximización de la compatibilidad entre los sistemas (a través de la reutilización de modelos estandarizados).

Diagramas web pretende reforzar este enfoque, creando un editor liviano y flexible, para evitar recurrir únicamente a eclipse como ambiente de desarrollo para este paradigma.

2.2. Eclipse Modeling Framework (EMF).

EMF es un framework de modelado y generación de código para construir herramientas y varias aplicaciones basadas en un modelo de datos estructurado. Desde los modelos especificados en XMI, EMF provee herramientas y los esquemas DOM para desarrollar otros complementos que necesiten desarrollar en MDE [2]. EMF se refleja en un complemento de eclipse, el cual puede ser instalado en este ambiente de desarrollo para producir herramientas más poderosas, como Eugenia y Gracot. Las anteriores herramientas utilizan la especificación de EMF y muchas de sus clases para las funciones que realizan. Diagramas web, debido a que necesita ser independiente de eclipse, solo utilizará las especificaciones de los objetos DOM que produce EMF, en particular, los XMI (modelos emf) y ecore que representan el concepto de metamodelo en un diagrama de tipo UML.

2.3. EL Metamodelo y su relación con los Modelos.

Consiste en un modelo de información para describir modelos. Un metamodelo expresa las características de un modelo en términos de sus atributos, relaciones y entidades [3]. EMF especifica una manera de plasmar este concepto utilizando diagramas de tipo ecore, los cuales son el principal insumo de Diagramas Web para validar y construir modelos.

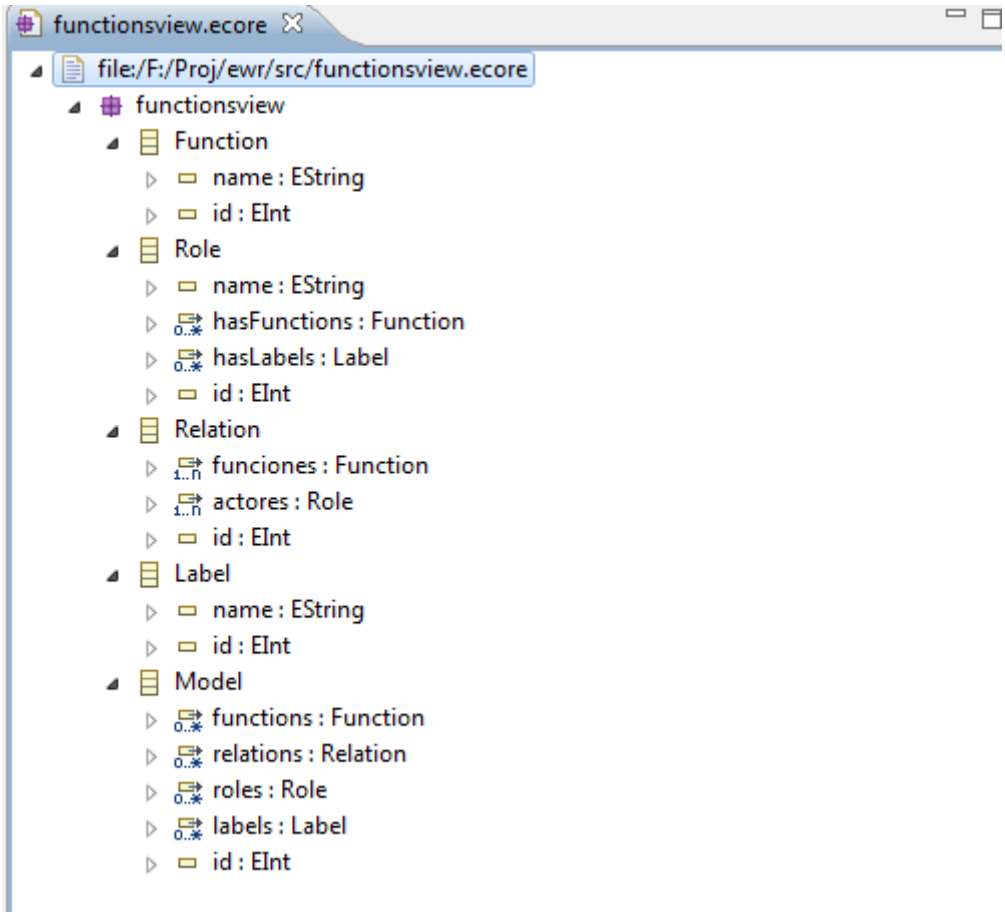


Figura 1: Metamodelo del Ejemplo "Simple Classes" En formato ecore.

2.4. Herramientas Similares.

A continuación se presenta una breve descripción de Gracot y Eugenia, dos herramientas desarrolladas en Eclipse para construir modelos sintácticamente correctos.

2.4.1. Gracot.

GraCot (Graphical Co-Creation Tool) es una herramienta gráfica de edición de modelos para eclipse, que permite editar tanto el modelo como el metamodelo en ambos sentidos. El objetivo de Gracot es detectar y evitar los problemas de conformidad de un modelo respecto a un metamodelo.

Gracot evalúa la conformidad de las reglas ontológicas y lingüísticas del modelo y metamodelo, y puede sugerir múltiples soluciones en caso de que esta conformidad no se alcance y modificar el metamodelo o el modelo respectivamente [4].

Diagramas web utiliza como referencia algunas las reglas lingüísticas y ontológicas que evalúa Gracot para hacer la validación del modelo respecto al metamodelo (mencionadas más adelante) y proporciona una interfaz similar de edición de gráfica de modelos y metamodelos (según la especificación de EMF).

2.4.2. Eugenia y GMF.

Eugenia es un complemento de eclipse que se encarga de generar automáticamente los insumos para construir un editor de modelos GMF basado en un metamodelo ecore simple. El objetivo de este complemento es reducir la barrera de entrada para la generación de editores, debido a que el solo framework GMF es complejo [5].

GMF se trata de un generador de editores gráficos de modelos XMI de eclipse, basados en un metamodelo, basados en unas especificaciones gráficas de representación (conocido como gmfgraph) una paleta de herramientas (gmftool) y el mapeo de herramienta-elemento-figura (gmfmap). El resultado es un complemento de eclipse con un editor visual de modelos, que los valida respecto al metamodelo seleccionado [8].

Diagramas Web debe realizar la misma función de generar diferentes editores según varios metamodelos de entrada al igual que Eugenia. En este caso, el gmfgraph, el gmftool y el gmfmap se mezclan en lo que se llama el descriptor de visualización (.mdescriptor) el cual se realiza manualmente, y la derivación del editor se realiza al cargar éste descriptor y el metamodelo ecore en conjunto, tal como lo hace Eugenia.

3. Detalles de la Herramienta.

A continuación se explicarán los aspectos más importantes que tiene Diagramas Web para cumplir los objetivos del proyecto. Entre estos están su arquitectura, las estructuras utilizadas y sus tecnologías.

3.1. Descripción General.

Diagramas Web es una herramienta de creación y edición de modelos liviana capaz de validar sintácticamente los mismos, utilizando su respectivo metamodelo en formato ecore de EMF como referencia. Para asegurar la flexibilidad, la herramienta está desarrollada en Javascript, JQuery y HTML5, cuya representación gráfica estará basada en un descriptor de visualización y renderizada por el nuevo lienzo nativo html5 Canvas. Este descriptor de visualización desarrolla una notación que permite el diseño de elementos gráficos ricos y personalizados que se asocien a los elementos existentes en el metamodelo correspondiente.

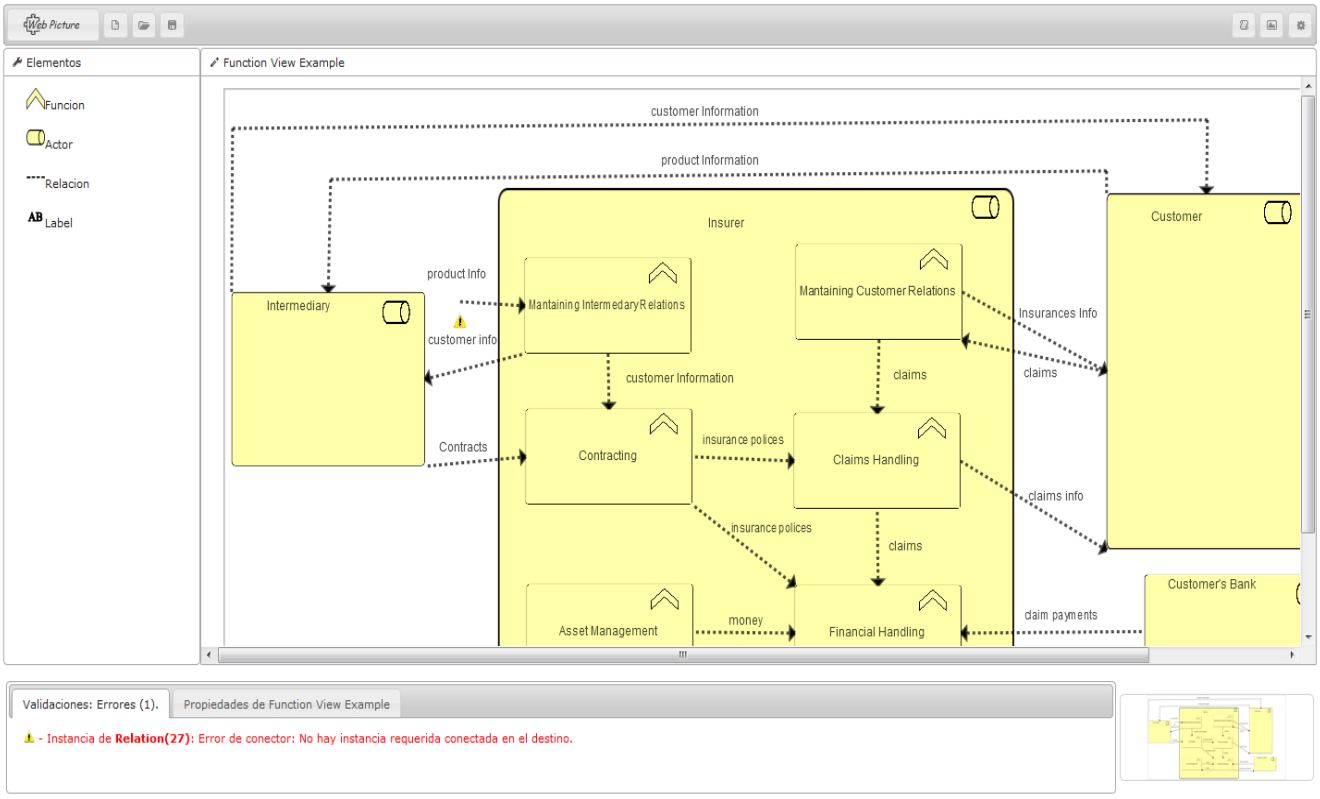


Figura 2: Vista de Diagramas Web.

3.2. Tecnologías Utilizadas.

En la herramienta se utilizaron en conjunto las tecnologías web html5, javascript y Jquery. A continuación se describen más en detalle.

3.2.1 Html 5.

Esta tecnología es la quinta revisión importante del lenguaje básico de la World Wide Web HTML [6]. Diagramas Web utiliza el nuevo tag introducido por esta tecnología llamado 'canvas'. Este tag permite que en el navegador web se puedan renderizar gráficos en tiempo real en un marco de dibujo, el cual es definido por este tag. Su funcionamiento es similar al marco de dibujo que ofrece la librería Java2D, solo que la definición de las figuras en su interior se realiza utilizando JavaScript.

Esta tecnología permite la edición visual del modelo, así como la selección, edición y eliminación visual de los elementos del mismo de forma nativa, que, a diferencia de la mayoría de editores de diagramas por web, lo hacen utilizando complementos de terceros (p.ej: adobe flash player).

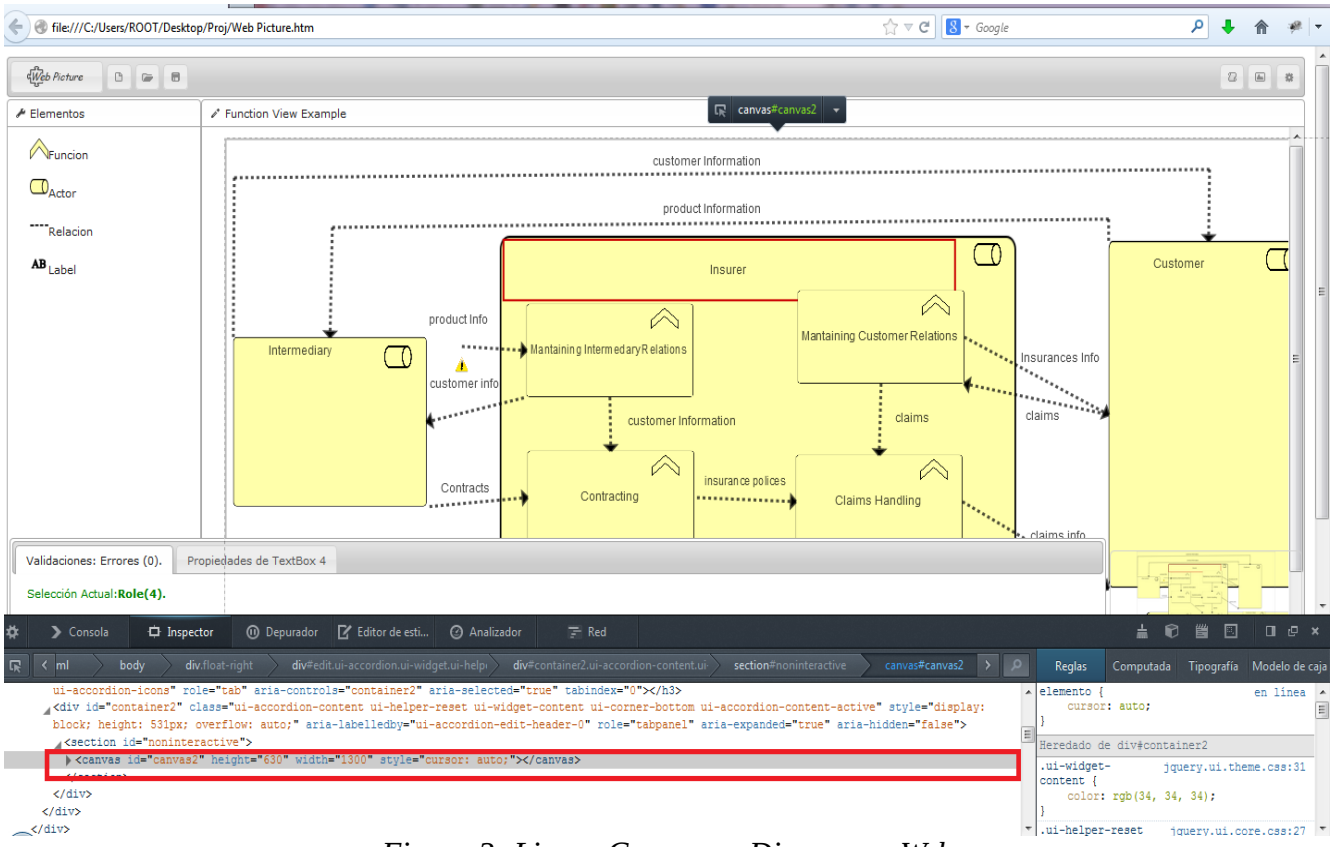


Figura 3: Lienzo Canvas en Diagramas Web

3.2.2. JavaScript.

JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Se diseñó con una sintaxis similar al C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo Java y JavaScript no están relacionados y tienen semánticas y propósitos diferentes.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM) [7].

Diagramas Web está implementado en este lenguaje en su totalidad, y el procesamiento que se realiza a partir de las especificaciones de los archivo de eclipse, se realiza nativamente con este lenguaje sin depender de complementos externos. Lo anterior hace que Diagramas Web pueda utilizar la especificación de metamodelo y XMI de EMF sin necesidad de instalar el ambiente de desarrollo que los soporta.

3.3. Arquitectura de la Herramienta.

A continuación se explicará el contexto de la herramienta, y la forma que utiliza para dibujar los modelos y validarlos.

3.3.1. Contexto de la Herramienta.

Para el funcionamiento de Diagramas Web, Se requiere de entrada un metamodelo en formato ecore, y un descriptor de visualización cuyo formato es propio de la herramienta (el cual se explicará más adelante). Como salida, se obtienen modelos sintácticamente correctos en formato también definido por la herramienta.

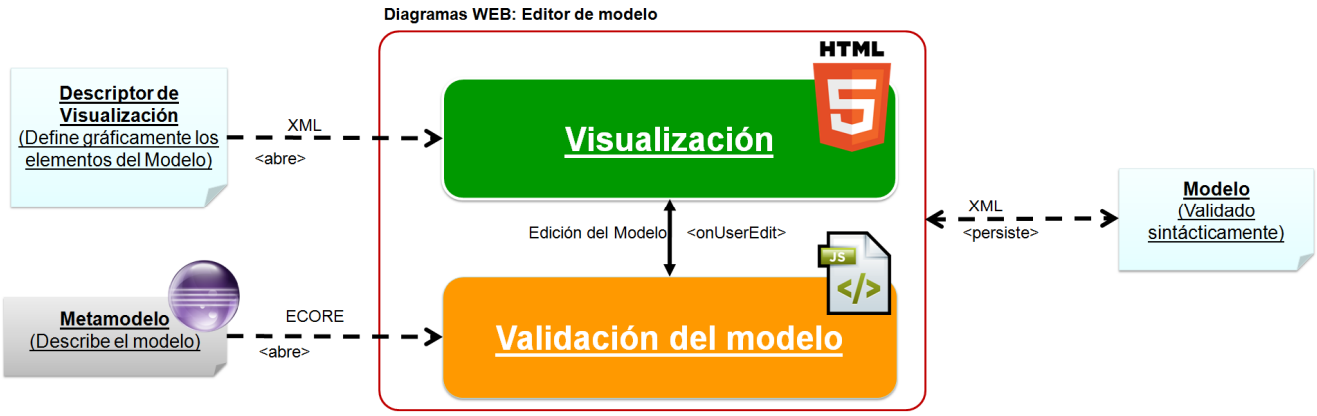


Figura 4: Contexto de la herramienta.

Adicional al modelo de salida, es posible exportar el modelo a formato XMI de EMF, para aprovechar las características ofrecidas por las herramientas basadas en esta tecnología que existen actualmente para eclipse.

3.3.2. Estructuras de datos más importantes y sus ventajas.

La herramienta utiliza dos estructuras de datos importantes: Para el modelo visual, se utiliza un grafo, cuyos vertices son las instancias de los elementos del metamodelo, y los arcos las referencias posibles de dicha instancia (dadas por el metamodelo). Las propiedades de un grafo permiten representar cualquier tipo de modelo sin perder información, lo cual hace de la herramienta altamente flexible en cuanto a los posibles tipos de modelo que se quieren representar, y facilita enormemente la tarea de validación. Entiéndase por modelo visual el conjunto de instancias que contienen la información de sus atributos y su visualización.

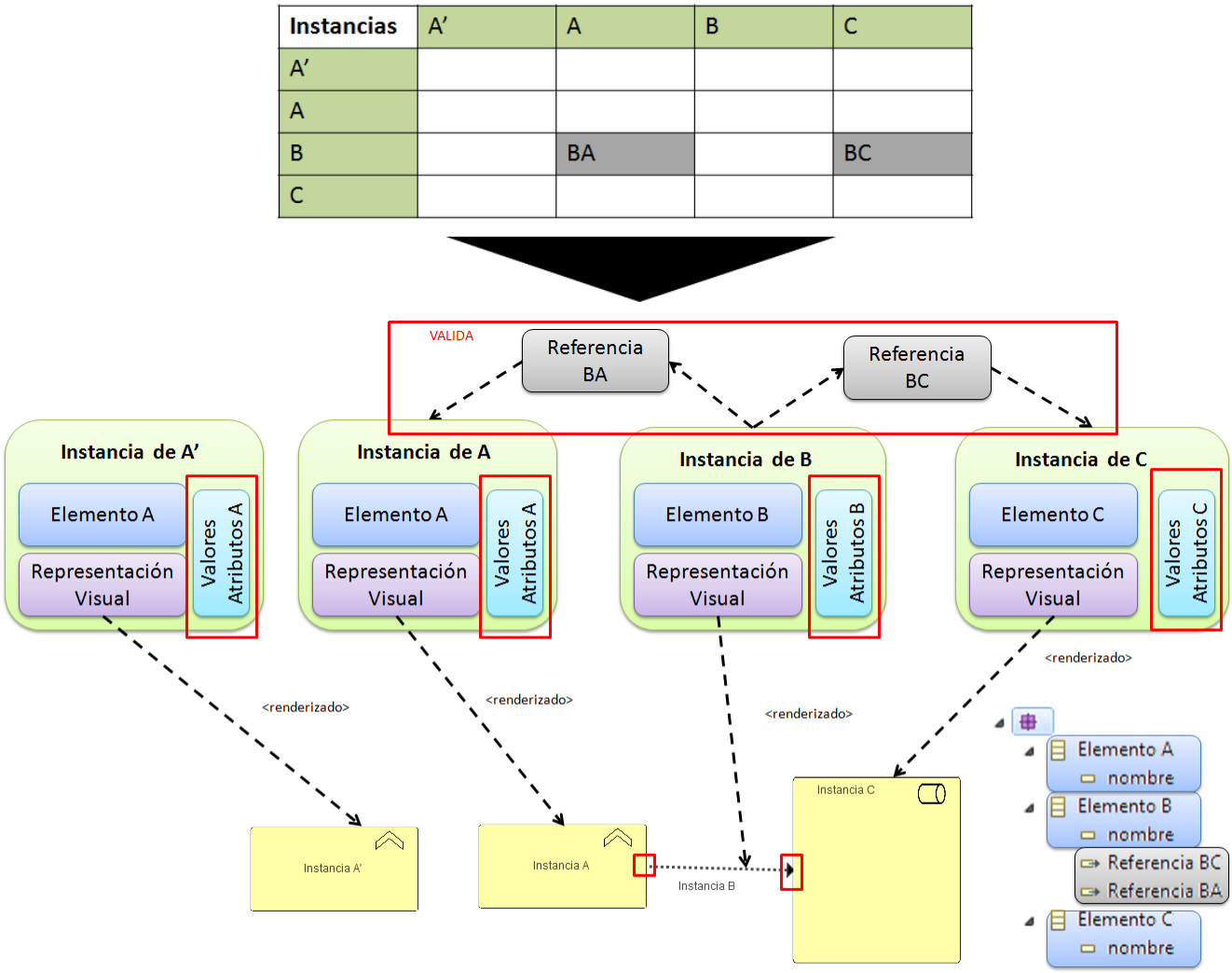


Figura 5: Funcionamiento de la estructura que soporta el Modelo Visual.

Esta estructura de datos se maneja en el módulo de visualización, lo cual agiliza el proceso de validación ya que el modelo se valida cada vez que el usuario modifique alguna propiedad del modelo. Lo señalado en recuadros rojos, indica la sintaxis que se valida a partir del metamodelo. Para agilizar el renderizado gráfico, debido a la complejidad de recorrido del

grafo, se recurre al uso de una lista auxiliar ordenada que respete las formas de visualizar contenencias y conectores. La herramienta prioriza el dibujado de conectores sobre todas las cosas, y se ayuda de las referencias disponibles para ordenar los otros elementos (o bien, decide si dibujar a partir del grafo si la jerarquia de elementos es compleja).

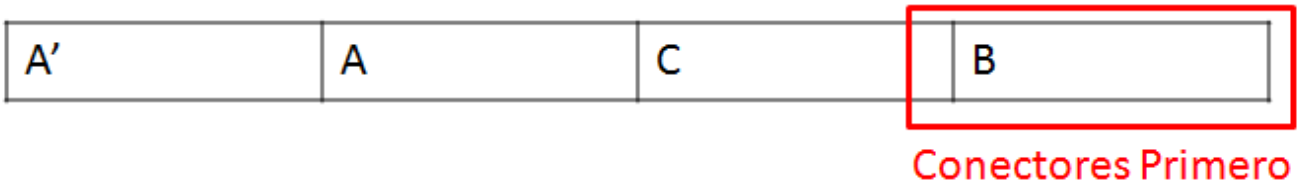


Figura 6: Ordenamiento de la Lista Auxiliar.

Para el metamodelo, se utilizaron listas, discriminando referencias y elementos. Las referencias tienen un origen y un destino que están asociados a la lista de elementos del metamodelo. Así mismo, sobre estas se ejecutan las funciones de validación principales, y arrojan un resultado dependiendo de los ítems de entrada. En el caso de la validación de referencias, entran los elementos de origen y destino, y se busca si esa referencia existe en la lista de referencias. En cuanto a los atributos, se valida si el tipo de valor corresponde al que indica el metamodelo ecore, dando como entrada la instancia de un elemento. Otras validaciones se deben realizar manualmente desde la construcción del descriptor de visualización. Las reglas que indican específicamente lo que se valida serán mencionadas más adelante.

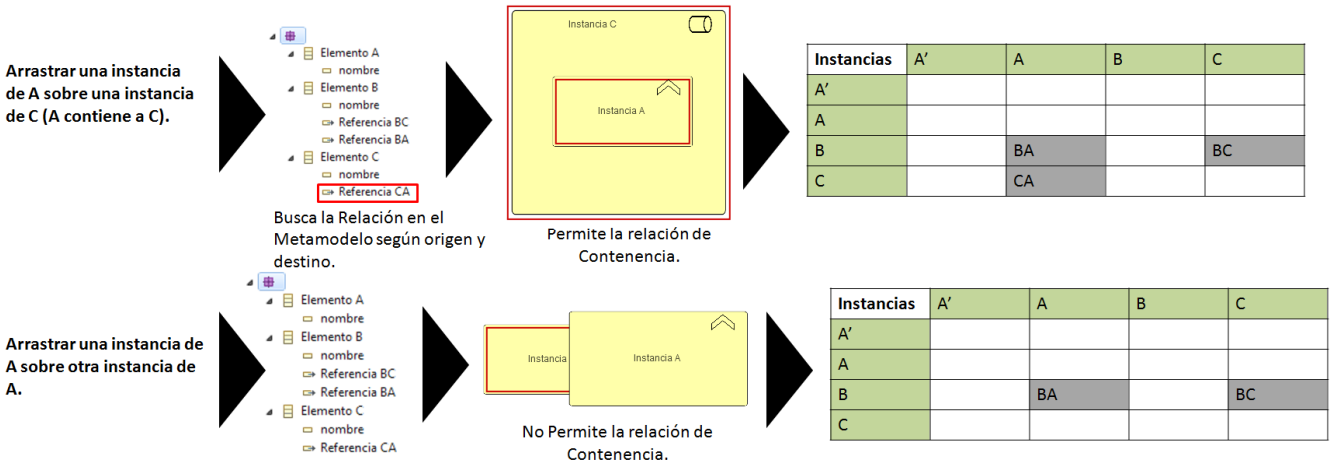


Figura 7: Ejemplo de validación según la interacción del usuario con el modelo.

Adicionalmente, en la búsqueda de referencias válidas se evalúa la multiplicidad, lo cual hace que se genere una advertencia al usuario si el límite de instancias permitido no es suficiente , o si se sobrepasa.

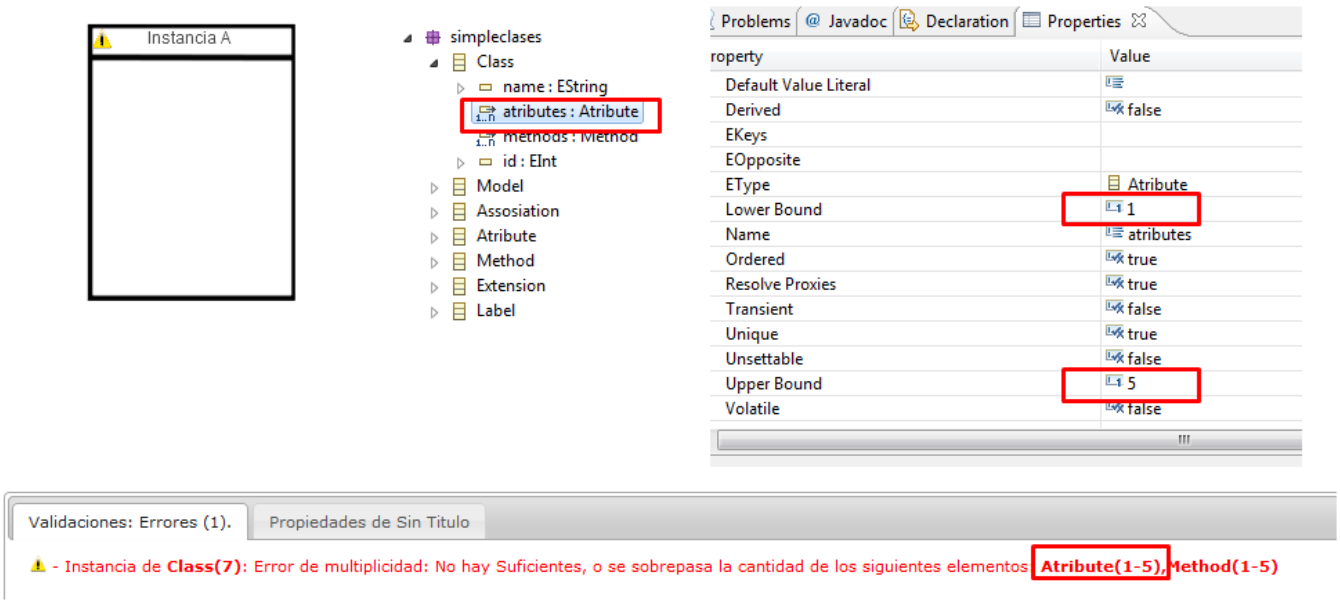


Figura 8: Ejemplo de validación de multiplicidad.

3.3.3. Representación gráfica de los modelos.

La representación gráfica de los modelos se basa esencialmente en el descriptor de visualización. El descriptor de visualización consiste en un archivo XML creado manualmente que contiene la definición de las figuras que representan un elemento de un metamodelo y la paleta disponible para definir los elementos. Este archivo posee la siguiente estructura:

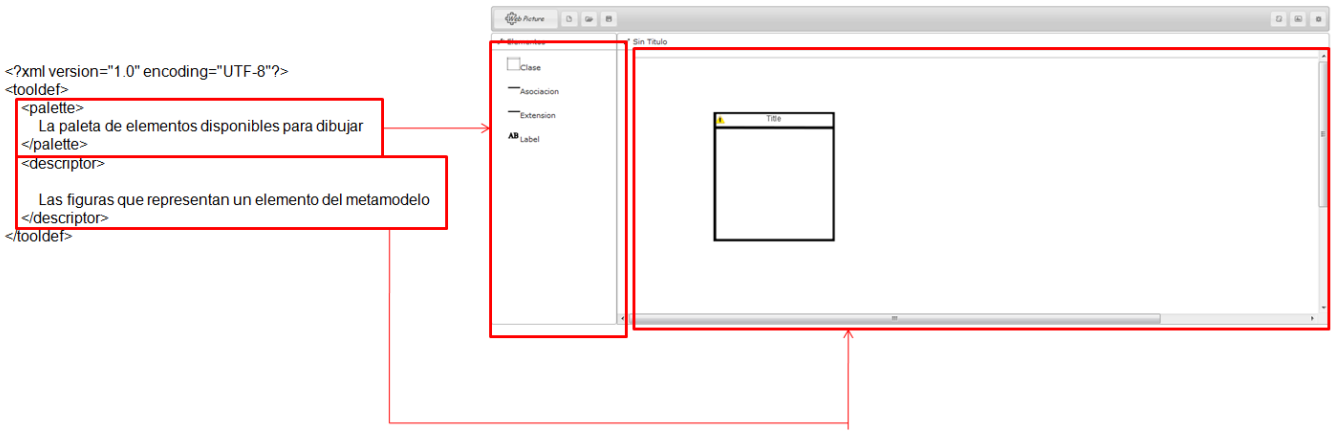


Figura 9: Estructura del descriptor de visualización.

La paleta se define por una serie de tags con nombre "Tool", que posee las siguientes propiedades:

- **elementId:** El Id de la figura raíz existente en <descriptor> que esta herramienta va a dibujar.
- **thumbnail:** La imagen de muestra de la herramienta. Se recomienda utilizar imágenes relativas al directorio "shapes" de la herramienta.
- **name:** Texto que aparece en frente de la herramienta.

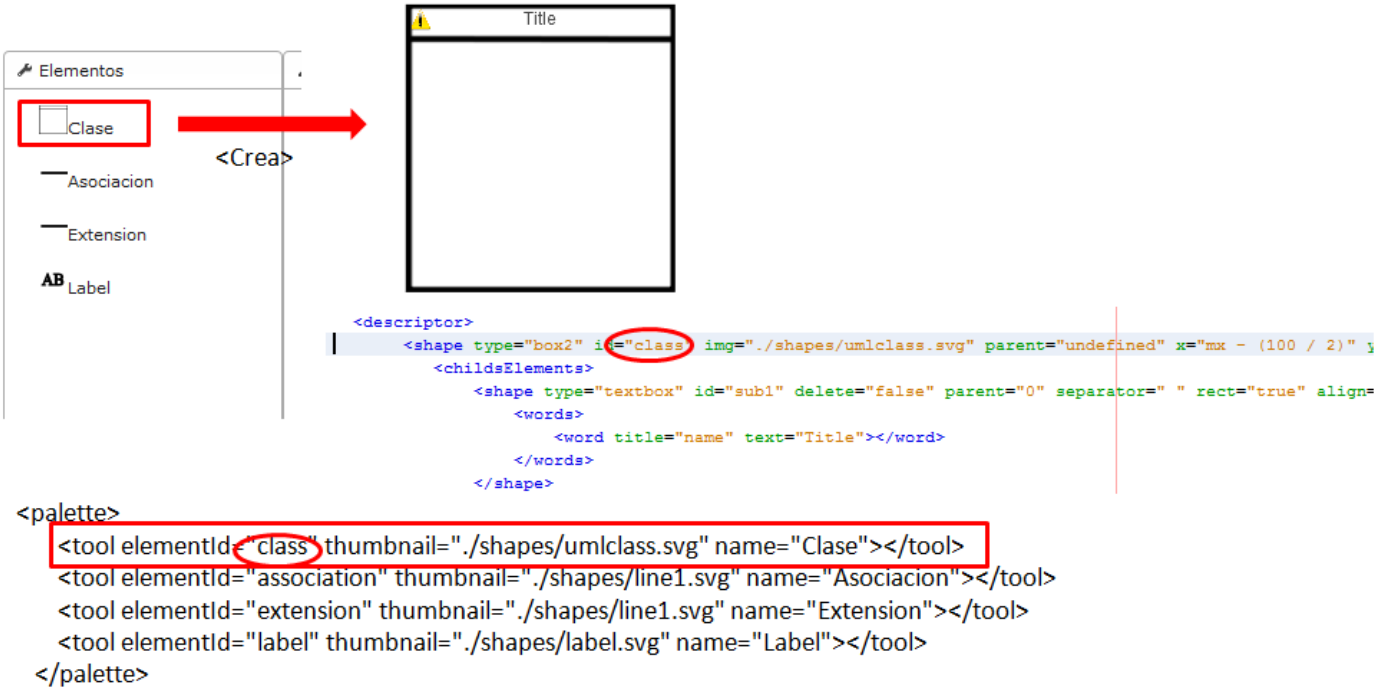


Figura 10: Estructura de la Paleta.

La definición de las figuras se realiza en la sección <descriptor> del xml. Una figura se define por el tag "shape". El <shape> raíz corresponde a la figura básica principal de representación de un elemento. Sus hijas, ubicadas en <childsElements> son otras <shape> que enriquecen a la principal. Las tres <shape> posibles en ese tag son Box2, FlowBox y Textbox. Solo es posible definir como raíz un shape de tipo Box2, y las demás como sus hijas. A continuación se muestran las propiedades básicas de definición de <shape>.

- **id:** El Identificador único de la figura (debe ser único incluso si es hija).
- **type:** El tipo de figura nativa que se va a definir. los valores posibles son box2,textbox,flowbox o link. Este atributo determina los otros posibles atributos del tag.
- **element:** Nombre del elemento del metamodelo al que hace referencia. Debe existir en el metamodelo.

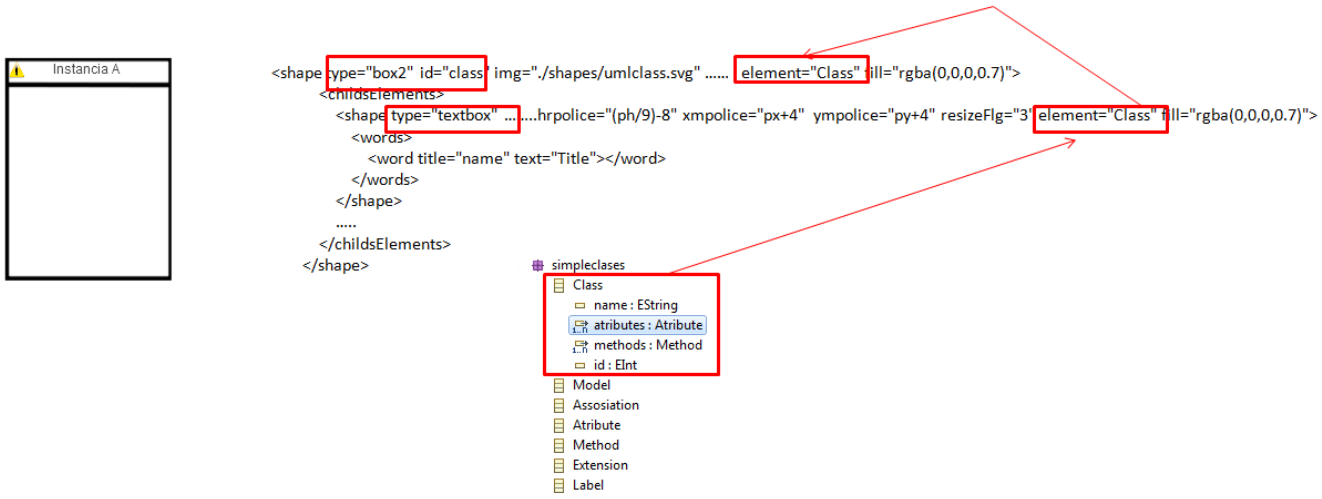


Figura 11: Estructura de la figura.

Cada figura <shape> básica posee propiedades de ubicación x, y, y tamaño w (ancho), h (alto), además de otras propiedades que hacen de cada figura única. El esquema de combinación de figuras básicas permite al usuario una alta flexibilidad a la hora de definir una forma de visualizar una instancia.
Una característica importante para la combinación de figuras es la alineación de estas respecto a un cambio de tamaño del padre. Por tal motivo, Diagramas web proporciona el uso de diez variables globales en la definición de posición y tamaño de las figuras nativas .

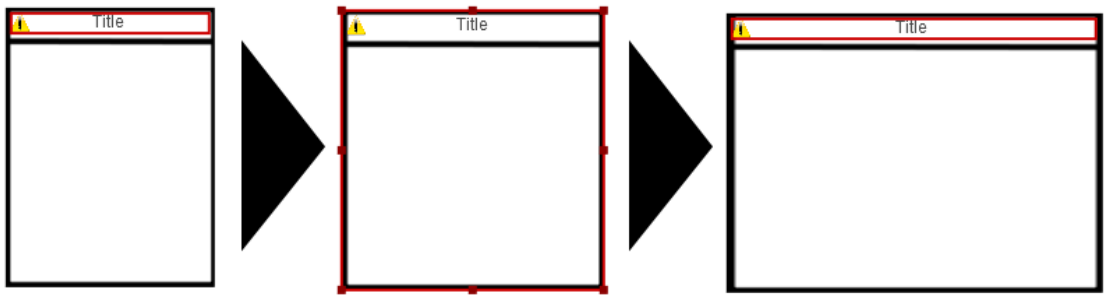
- **mx:** Obtiene la posición en x del cursor en el lienzo canvas.
- **my:** Obtiene la posición en y del cursor en el lienzo canvas.
- **sx:** Obtiene la posición en x de la figura básica que lo invoca.
- **sy:** Obtiene la posición en y de la figura básica que lo invoca.
- **sw :** Obtiene el ancho de la figura básica que lo invoca.
- **sy:** Obtiene el alto de la figura básica que lo invoca.
- **px:** Obtiene la posición en x de la figura básica padre que lo contiene.
- **py:** Obtiene la posición en y de la figura básica padre que lo contiene.
- **pw:** Obtiene el ancho de la figura básica padre que lo contiene.
- **ph:** Obtiene el alto de la figura básica padre que lo contiene.

Estas variables se pueden utilizar en las propiedades de x , y, w , h (disponibles en el Box2, FlowBox y Text Box) que definen la posición inicial de la figura al ser creada. Para manejar los eventos de cambio de tamaño y posición respecto al padre que los contiene, se manejan las siguiente propiedades adicionales.

- **xmpolice:** Indica la posición de la figura en x cuando su padre es movido/ cambiado de tamaño.
- **xypolice:** Indica la posición de la figura en y cuando su padre es movido/ cambiado de

tamaño.

- **wrpolice:** Indica el ancho de la figura cuando su padre es movido/ cambiado de tamaño.
- **hrpolice:** Indica el alto de la figura cuando su padre es movido/ cambiado de tamaño.




```
<shape type="box2" id="class" img="./shapes/umlclass.svg" parent="undefined" x="mx - (100 / 2)" y="my - (100 / 2)" w="150" h="200"
wrpolice="sw" hrpolice="sh" xmpolice="sx" ympolice="sy" resizeFlg="0" element="Class" fill="rgba(0,0,0,0.7)">
        <childsElements>
          <shape type="textbox" id="sub1" delete="false" parent="0" separator=" " rect="true" align="CENTER" font="12px Arial"
forecolor="false" img="null" x="px+4" y="py+4" w="pw-8" h="(ph/9)-8" wrpolice="pw-8" hrpolice="(ph/9)-8" xmpolice="px+4" ympolice="py+4"
resizeFlg="3" element="Class" fill="rgba(0,0,0,0.7)">
            <words>
              <word title="name" text="Title"></word>
            </words>
          </shape>
```

Figura 10: Uso de las variables globales según el descriptor de visualización.

A continuación se explican las figuras nativas posibles y sus propiedades adicionales.

Box2

El Box 2 es la figura más básica del editor. Es un rectángulo, y es la única que permite interacción directa con el usuario (puede ser cambiada de tamaño y arrastrada). Posee las siguientes propiedades adicionales.

- **img:** La imagen fomrato svg de fondo. Se recomienda utilizar imágenes relativas al directorio "shapes" de la herramienta.
- **resizeflg:** 0 si la figura se puede cambiar de tamaño por el usuario, 1 si no se desea.
- **fill:** color de relleno de la figura en formato rgba(x,x,x,x).
- **parent:** debe ser siempre 0.

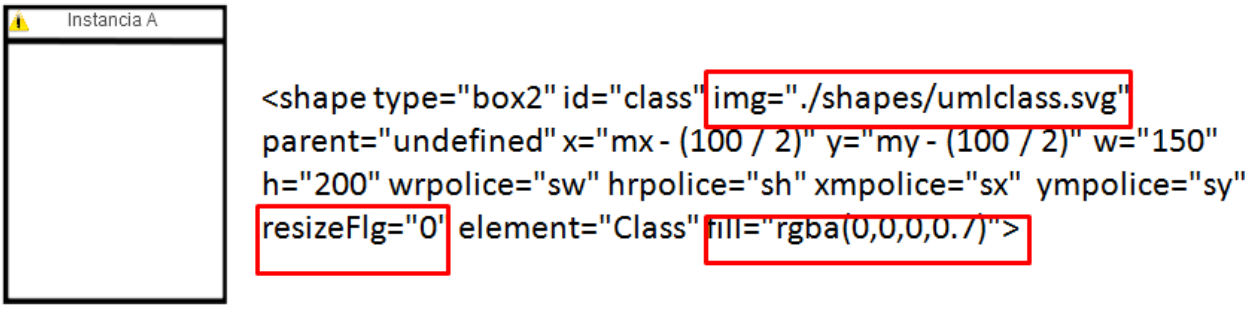


Figura 11: Ejemplo definición Box2.

Flow Box

El FlowBox es un rectángulo que cuya posición y tamaño son relativas a un Box2 que lo contenga como padre. Adicionalmente, éste puede contener textbox en su interior con un orden en particular, y pueden ser añadidos por un menú contextual siempre y cuando existan textbox en la raíz del descriptor, que representen elementos con referencias de contenencias válidas de este como padre. Las propiedades adicionales que presenta esta figura son las siguientes.

- **img:** La imagen formato svg de fondo. Se recomienda utilizar imágenes relativas al directorio "shapes" de la herramienta.
- **stmode:** 0 el orden de los textbox que se añaden es horizontal, 1 si el orden es vertical.
- **fill:** color de relleno de la figura en formato rgba(x,x,x,x).
- **parent:** debe ser siempre 0.

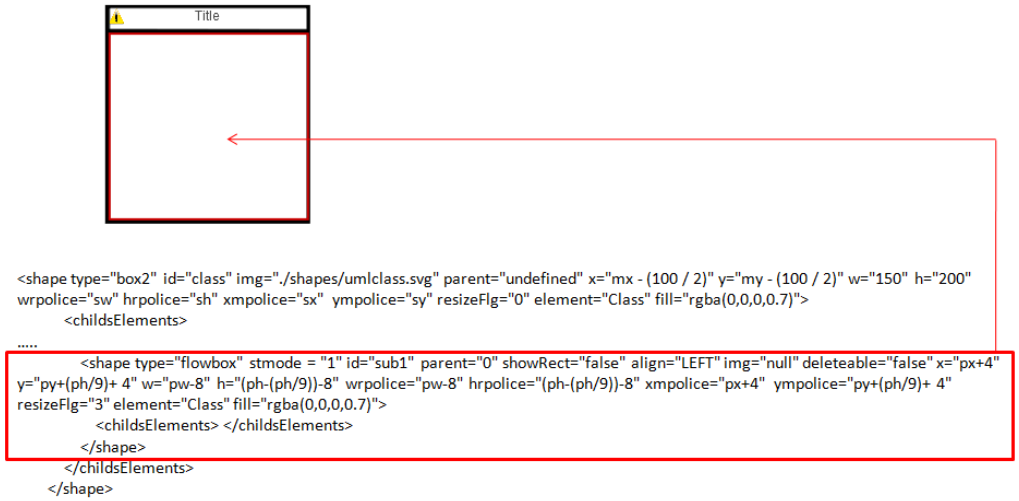


Figura 12: Ejemplo definición FlowBox.

De estas maneras se pueden utilizar los FlowBox para contenidos de imagen estática o texto.

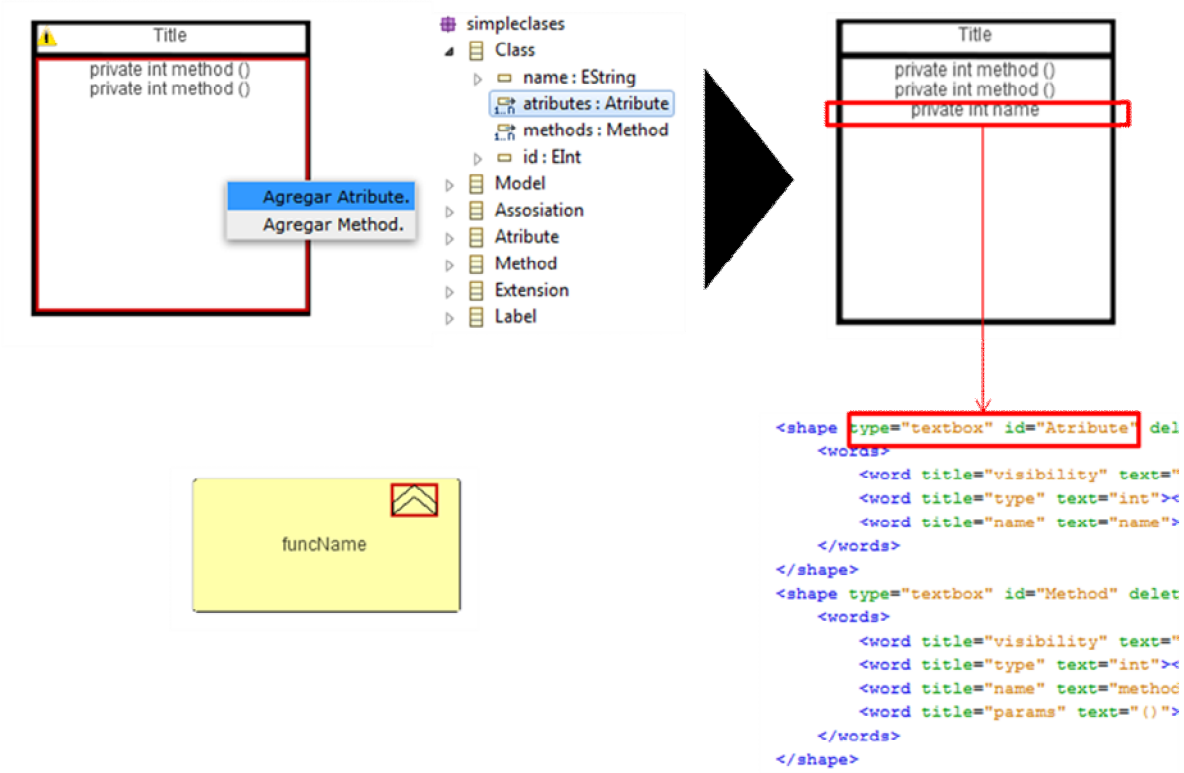


Figura 13: Formas con FlowBox.

Nótese que para añadir instancias desde el menú contextual, es necesario que el atributo id del textbox ubicado en la raíz sea igual al nombre del elemento del metamodelo (el mismo valor del atributo "element" del textbox) que se va a contener.

Text Box

El textbox es la única figura nativa que soporta texto. Éste se encarga de representar los atributos de los elementos. El texto que contiene la figura se puede dividr por palabras que pueden contener diferentes atributos, separados por un "token" que se define en las propiedades adicionales. Las propiedades son las siguientes.

- **img:** La imagen formato svg de fondo. Se recomienda utilizar imágenes relativas al directorio "shapes" de la herramienta.
- **delete:** Detemina si la figura es eliminable por el usuario (true o false).
- **separator:** detemina el "token" de separación de las palabras del texto con diferentes atributos.
- **rect:** determina si el cuadro se bordea con un rectángulo del color de la propiedad fill.
- **align:** CENTER,LEFT,RIGHT. Alineación del texto en el rectángulo de la figura.
- **forecolor:** Color de fuente. Si no es definido (poner undefined) se utiliza el definido en fill.
- **resizeflg:** debe ser siempre 3.
- **parent:** debe ser siempre 0.

- **font:** tamaño y tipo de fuente. Ejemplo: 12px Arial.
- **stmode:** 0 el orden de los textbox que se añaden es horizontal, 1 si el orden es vertical.
- **fill:** color de relleno de la figura en formato rgba(x,x,x,x).

Esta figura posee en su interior el tag <words>. Dentro de éste es necesario definir al menos un tag <word>, que corresponde al mapeo atributo,valor del elemento al que representa dicho textbox. El tag <word> posee las siguientes propiedades.

- **title:** Nombre del atributo, que debe existir en el elemento definido en "element" del textbox.
- **text:** Valor inicial del atributo.

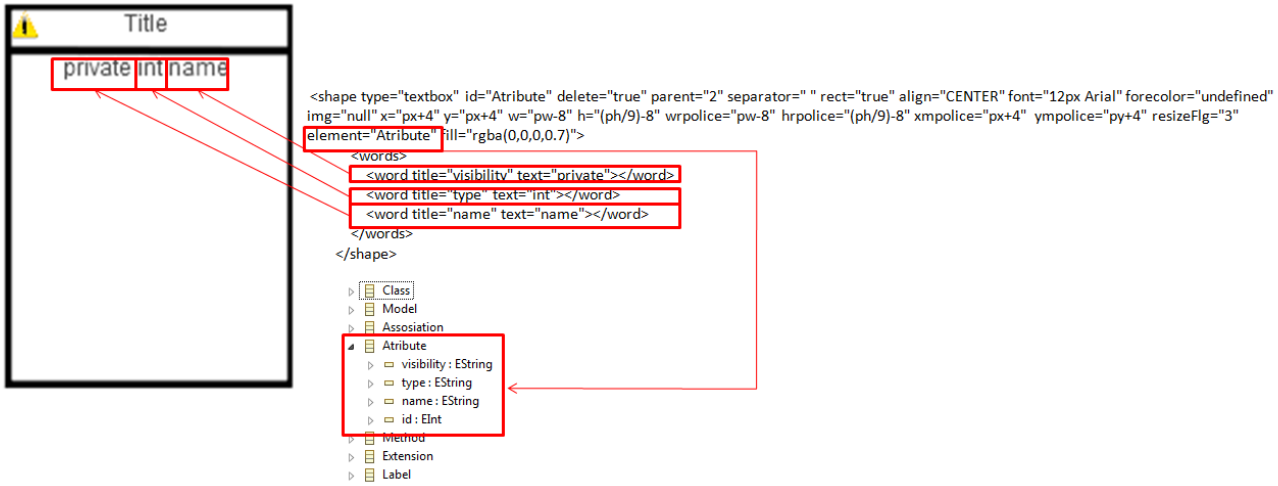


Figura 14: Definición de TextBox.

Nótese que el separador en la imagen anterior es un espacio.

Link

El Link es la única figura básica para realizar conectores. Esta posee la propiedad de conectarse únicamente con Box2 es sus extremos. Adicionalemente, puede doblarse hasta cinco veces. La propiedades adicionales que posee son las siguientes.

- **p1Rel - p2Rel:** Nombres de las referencias (separadas por comas y existentes en el metamodelo) que el conector tendrá en cuenta al momento de conectarse a otras instancias.
- **w:** determina el ancho de la línea.
- **arrowStyle:** Estilo de las puntas de la flecha. Valores de 1 a 10. 0 sin puntas.
- **arrowSize:** Tamaño de las cabezas de la flecha.
- **arrowColor:** Relleno de las cabezas de la flecha. poner "no" indica que las cabeceras se dibujan con una flecha sin relleno.
- **fill:** color de relleno de la línea rgba(x,x,x,x).

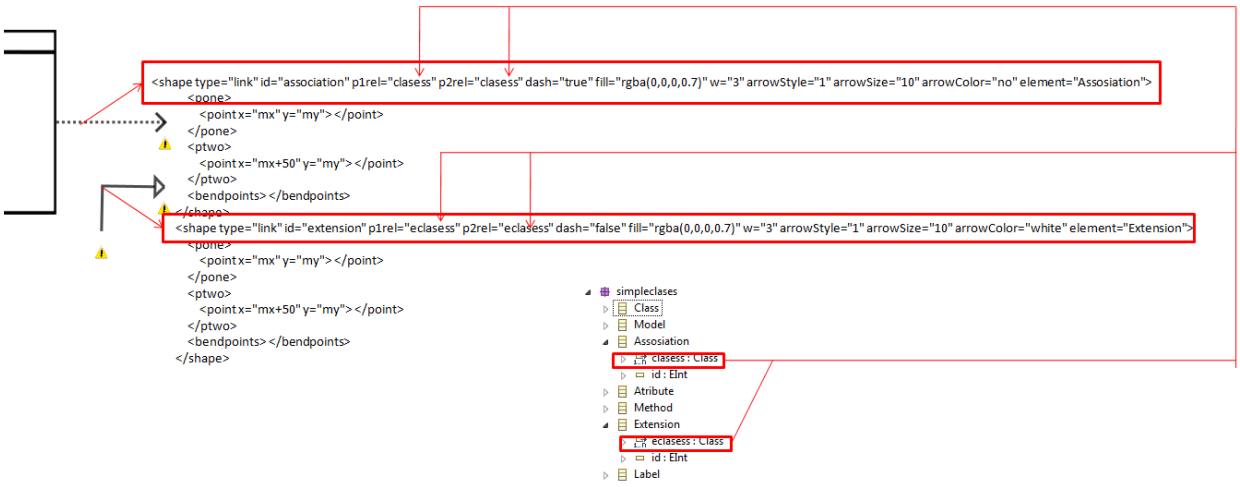


Figura 14: Definición de Link.

Nótese que en el interior de la figura se encuentran los tags <pone>, <ptwo> y <bendpoints>. Dentro de los tags <pone> y <ptwo> se definen los puntos iniciales al agregar el conector al modelo (tag <point x="posición en x" y="posición en y">) puede utilizar únicamente la variable mx y my. El tag <bendpoins> dejarlo en blanco.

3.3.4. La función Exportar a XMI.

Esta función del editor permite que los modelos creados en diagramas web sean utilizables en los frameworks de ingeniería de modelos de eclipse. Para esto, los elementos de los metamodelos ecore deben tener un atributo de nombre id, y un elemento de nombre "Model" que tenga referencias a todos los elementos del metamodelo.

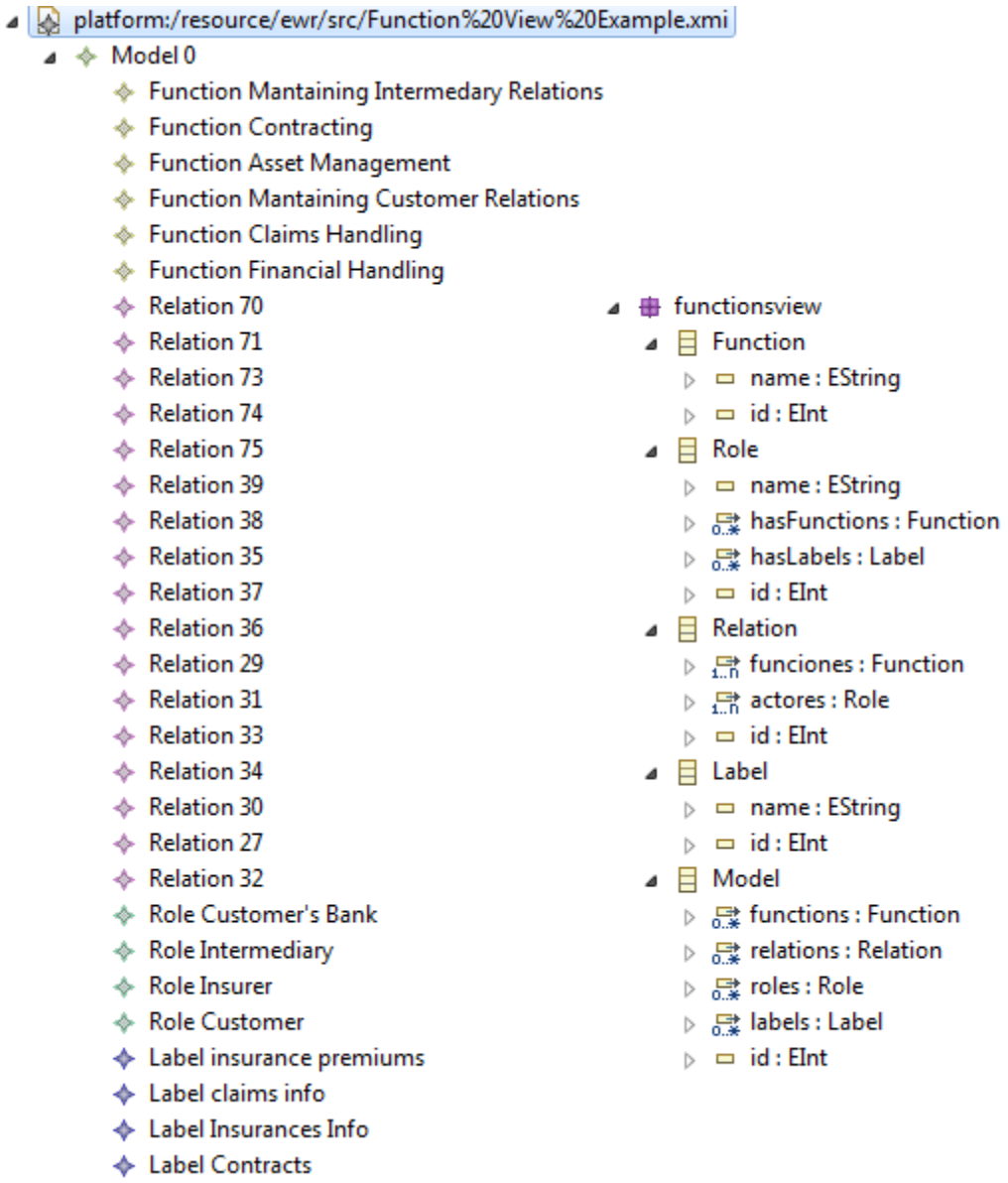


Figura 26: Ejemplo de Salida de XMI y ecore de origen.

3.4 Validaciones Disponibles.

Las reglas de validación lingüísticas y ontológicas [9] en las que se basa la herramienta para evaluar la sintaxis de los modelos se muestran en las siguientes tablas.

Rule	***Problem Detected***
MC-LR-1	The value of MetamodelURI is not provided.
MC-LR-2	No instance matches the root EClass in the domain MM.
EC-LR-1	An Element instance has not value to typeName.
EC-LR-2	The value of typeName in one Element instance has blanks.
EC-LR-3	The Element instance does not have an owner instance.
AC-LR-1	The Attribute instance does not have any value in typeName.
AC-LR-2	The typeName in one Attribute instance has blanks.
AC-LR-3	The typeName in one Attribute instance starts or ends with comma.
AC-LR-4	There are several Attribute instances with the same value typeName belongs to the same Element instance.
AC-LR-5	The value of one Attribute instance does not correspond with the type.
CRC-LR-1	The ContainmentRelation instance does not have any value in typeName.
CRC-LR-2	The typeName in one ContaintmentRelation instance has blanks.
CRC-LR-3	Two ContainmentRelation instances associate two Element instances with opposite direction.
CrRC-LR-1	The CrossRelation instance does not have any value in typeName.
CrRC-LR-2	The typeName in one CrossRelation instance has blanks.

Figura 15: Validaciones Lingüísticas

Rule	***Problem Detected***
MC-OR-1	The domain MM has several root EClasses.
MC-OR-2	The value of the attribute MetamodelURI does not coincide with domain MM URI.
EC-OR-1	The typeName of one Element instance does not match with any EClass name.
EC-OR-2	The Element instance has several owner instances.
EC-OR-3	There are several Element instances that match with the root EClass in domain MM.
EC-OR-4	The Element instance does not have associated the Attribute instances required.
EC-OR-6	The Element instance does not have associated the CrossRelation instances required.
AC-OR-1	The typeName of one Attribute instance does not match with any EAttribute name of the corresponding EClass.
AC-OR-2	The type in one Attribute instance does not match with the corresponding Type.
AC-OR-3	The attribute value in one Attribute instance is not defined.
AC-OR-4	The quantity of values in the attribute value for one Attribute instance is lower than the lower bound in the domain MM or greater than the upper bound in the domain MM.

CRC-OR-1	The typeName in one ContainmentRelation instance does not match with any EReference name of the corresponding EClass.
CRC-OR-2	The quantity of ContainmentRelation instances belong to one Element instance is lower than the lower bound in the domain MM or greater than the upper bound in the domain MM.
CrRC-OR-1	The typeName in one CrossRelation instance does not match with any EReference name of the corresponding EClass.
CrRC-OR-2	The quantity of CrossRelation instances belong to one Element instance is lower than the lower bound in the domain MM or greater than the upper bound in the domain MM.

Figura 16: Validaciones Ontologicas

La herramienta valida las reglas anteriores de la siguiente manera.

Regla	Forma de Validar	Validada?
Lingüísticas		
MC-LR-1	No aplica, no es el validador de EMF.	No
EC-LR-1	Validado con el Descriptor grafico del Metamodelo (cuando es creado).	Si
EC-LR-2		Si
AC-LR-1		Si
AC-LR-2		Si
AC-LR-3		Si
AC-LR-5		Si
CRC-LR-1		Si
CRC-LR-2		Si
CRC-LR-3	No Contemplado en versión preliminar.	No
CrRC-LR-1	Validado con el Descriptor grafico del Metamodelo (cuando es creado).	Si
CrRC-LR-2		SI
Ontológicas		
MC-OR-1	No aplica- No Probada.	No
MC-OR-2	No aplica, no es el validador de EMF.	No
MC-OR-3	Validado cuando se crea el descriptor de figuras del metamodelo.	Si
EC-OR-1		Si
EC-OR-2		Si
EC-OR-3	No aplica, no es el validador de EMF.	No
EC-OR-4	Validado cuando se crea el descriptor de figuras del metamodelo.	Si
EC-OR-5		Si
EC-OR-6		Si
EC-OR-7	No aplica para la versión preliminar del software.	No
AC-OR-1	Validado por el descriptor de figuras del metamodelo.	Si
AC-OR-2		Si
AC-OR-3		Si

AC-OR-4	No aplican restricciones de longitud del valor en la versión preliminar del software.	No
AC-OR-5	Validado por el descriptor de figuras del metamodelo.	Si
CRC-OR-1		Si
CRC-OR-2	La aplicación web maneja las relaciones de manera distinta. En la versión preliminar se realizan las siguientes validaciones: De contenencia, upper limit y lower limit, y de conectores creados como elementos (EClass con Ereference origen y destino).	Si
CRRC-OR-1	Validado con el descriptor de figuras del metamodelo.	Si
CRRC-OR-2	La aplicación web maneja las relaciones de manera distinta. En la versión preliminar se realizan las siguientes validaciones: De contenencia, upper limit y lower limit, y de conectores creados como elementos (EClass con Ereference origen y destino).	Si

Figura 17: Validaciones realizadas por la aplicación.

Las validaciones de primera jerarquía[9] son validadas por el descriptor de la figura (cuando este se construye) lo cual hace que todas las validaciones programáticas, no sean dependientes.

4. Ejemplos.

Mediante el uso de la aplicación web se pueden generar diferentes tipos de editores gráficos para la creación de modelos de diferentes dominios. A continuación se presentaran dos ejemplos mediante los cuales se ilustra la creación de diferentes modelos.

4.1. Vista de Funciones de ArchiMate.

Este ejemplo se realizó con base en el caso de estudio “Archisurance” de “The Open Group”. A continuación se presenta un diagrama de ArchiMate utilizando los elementos de la capa de negocio para hacer un modelo que represente de un punto de vista funcional del negocio.

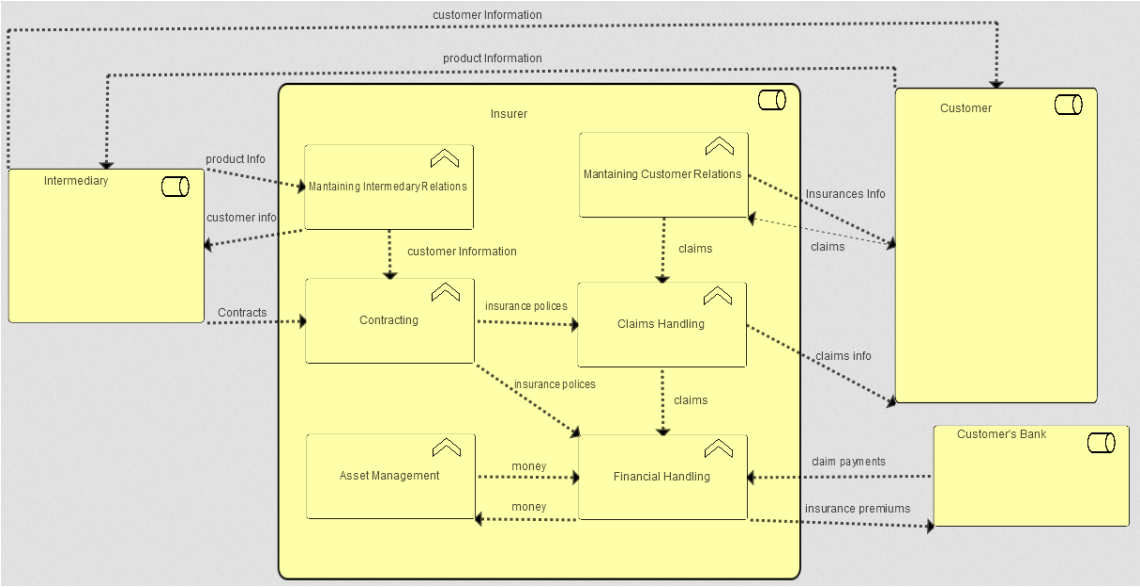


Figura 18: Modelo de ejemplo ArchiMate.

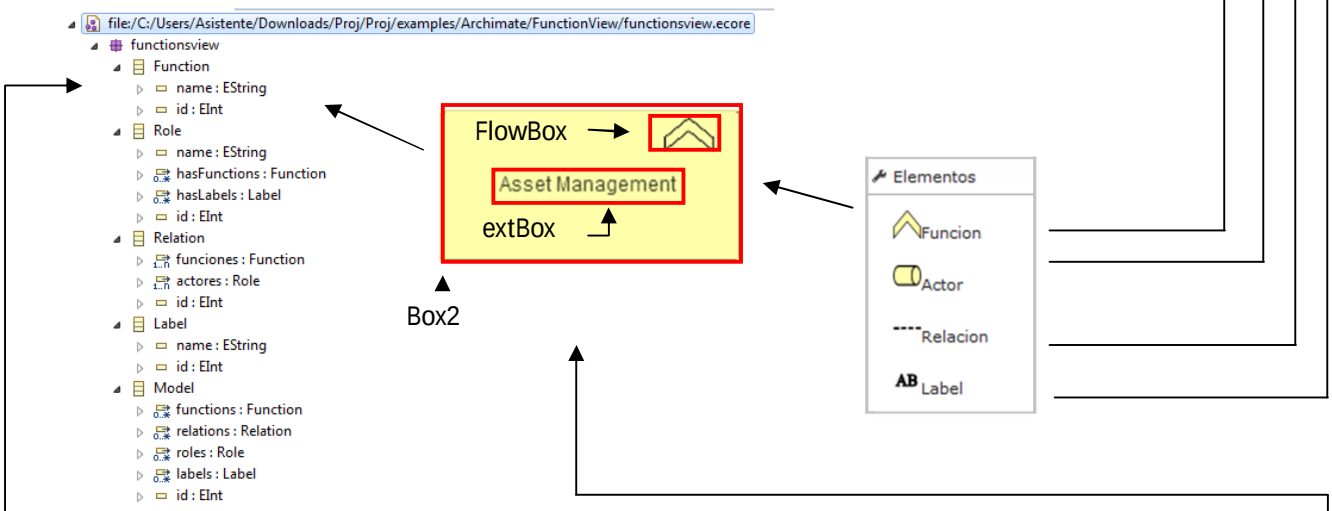
Así mismo, a partir de este modelo se pretende mostrar la capacidad de la aplicación para la creación de relaciones entre diferentes elementos, contenencias, conectores y representación de imágenes vectoriales (formato SVG).

En el descriptor (archivo.picture) se especifican los elementos que tendrá en este caso la paleta de la aplicación. Asi mismo, cada elemento de la paleta se encontrará asociado a un elemento del metamodelo (archivo.ecore) y representado por uno de los cuatro tipos de representaciones visuales con los que se cuenta: Box2, FlowBox, TextBox o Link respectivamente especificadas en el descriptor.

Definición de la paleta en el descriptor (Archivo .picture)

```
<tooldef>
    <palette>
        <tool elementId="function" thumbnail="./shapes/function.svg" name="Funcion"></tool>
        <tool elementId="actor" thumbnail="./shapes/actor.svg" name="Actor"></tool>
        <tool elementId="link" thumbnail="./shapes/line2.svg" name="Relacion"></tool>
        <tool elementId="label" thumbnail="./shapes/label.svg" name="Label"></tool>
    </palette>
```

Metamodelo (.ecore)



Definición de los elementos graficos en el descriptor (Archivo .picture)

```
<shape type="box2" id="function" img="./shapes/box.svg" parent="undefined" x="mx - (100 / 2)" y="my - (100 / 2)" w="180"
h="90" wrpolice="sw" hrpolice="sh" xmpolice="sx"  ympolice="sy" resizeFlg="0" element="Function" fill="rgba(0,0,0,0.7)">
    <childsElements>
        <shape type="textbox" id="sub1" delete="false" parent="0" separator=" " rect="true" align="CENTER" font="12px Arial"
        forecolor="false" img="null" x="px+4" y="py+4" w="pw-8" h="ph-8" wrpolice="pw-8" hrpolice="ph-8" xmpolice="px+4"  ympolice="py+4"
        resizeFlg="3" element="Function" fill="rgba(0,0,0,0.7)">
            <words>
                <word title="name" text="funcName"></word>
            </words>
        </shape>
        <shape type="flowbox" stmode = "1" id="sub1" parent="0" showRect="false" align="LEFT" img="./shapes/function.svg" deleteable="fal
            <childsElements> </childsElements>
        </shape>
    </childsElements>
</shape>
```

Figura 19: Definición de elementos ejemplo ArchiMate

En este ejemplo se puede evidenciar la capacidad de la aplicación para representar las contenencias de elementos. En este caso, la entidad Actor puede contener uno o varios Labels y Funciones.

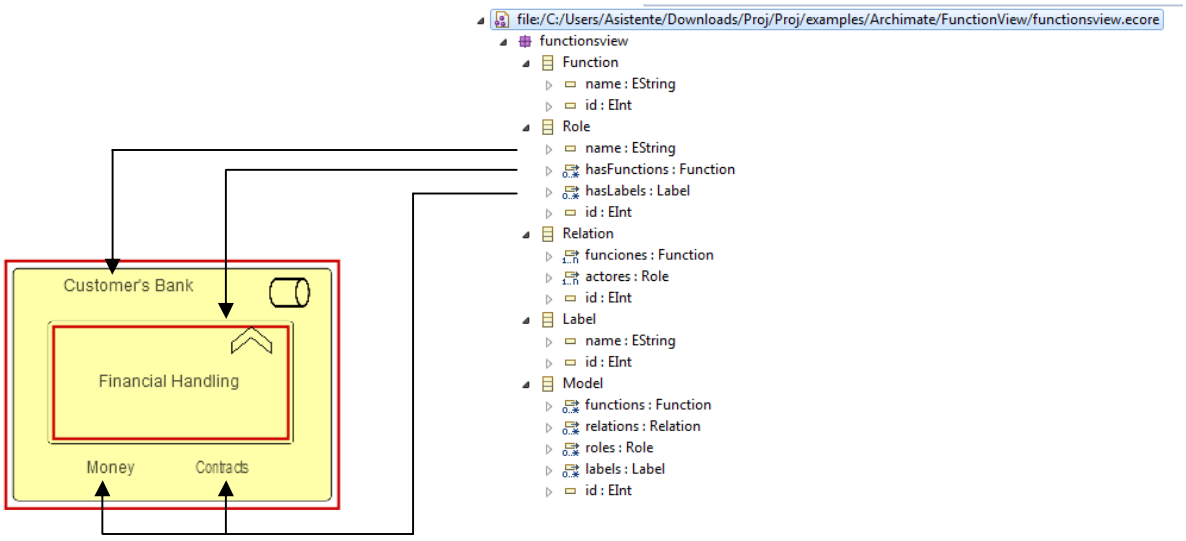


Figura 20: Ejemplo de contenencia ArchiMate.

Otro aspecto importante representado en este ejemplo son las distintas validaciones que puede presentar el modelo. Por ejemplo, la representación visual de un link en el descriptor vista como una relación en el metamodelo siempre debe tener un elemento asociado tanto de origen como de destino. Si no cumple con dichos requisitos, la aplicación mostrara al usuario un mensaje de advertencia visual y un mensaje en rojo en la parte de validaciones del modelo.

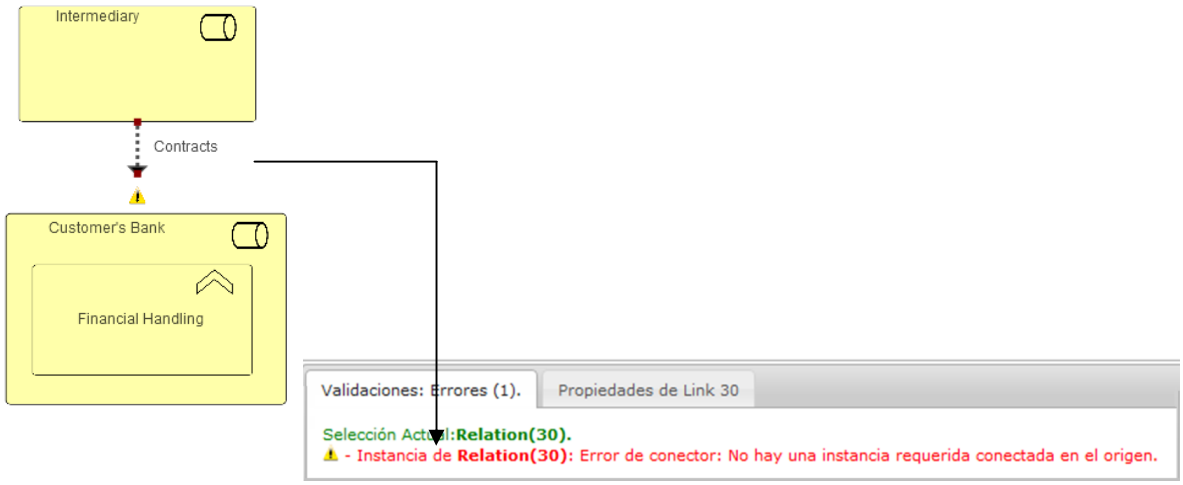


Figura 21: Ejemplo de Validaciones ArchiMate.

4.2. Clases UML Simple.

Se creó un editor de modelos UML simple, en el cual se tienen los elementos básicos del lenguaje de modelado tales como las clases, atributos, asociaciones, extensiones y los labels para representar las características de dichos elementos.

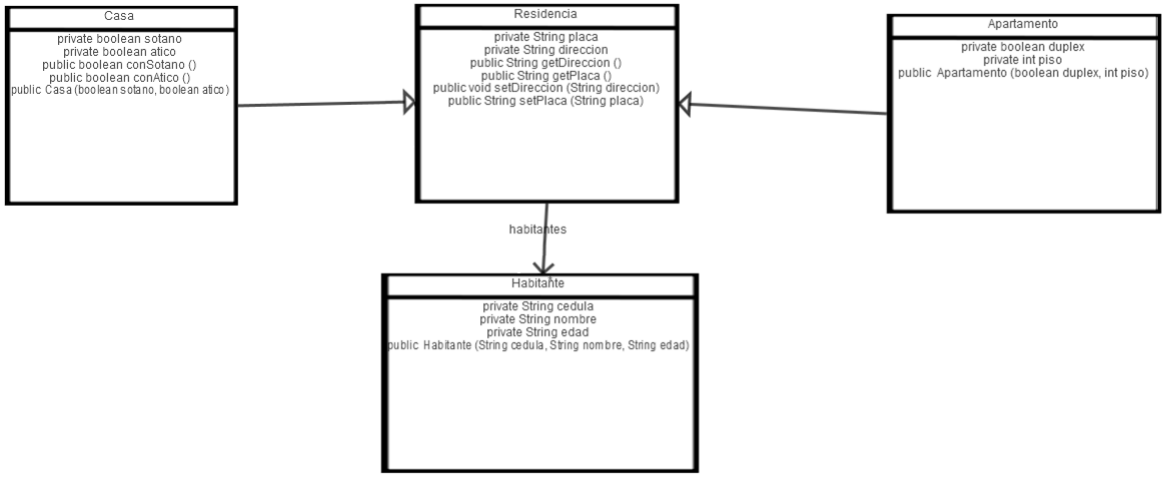


Figura 22: Modelo de ejemplo UML.

A partir de este modelo se pretende mostrar la capacidad de la aplicación para generar la representación visual de diferentes tipos de figuras entre las cuales podemos encontrar los Box2, que en este caso harían referencia a la entidad o clase que se está representando, los flowbox que se encargan de agrupar los atributos, métodos y características de cada una de las clases y los TextBox mediante los cuales se especifica el texto asociado a las características de cada elemento.

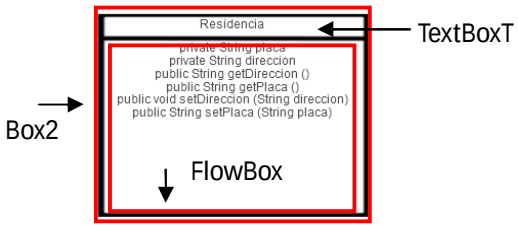


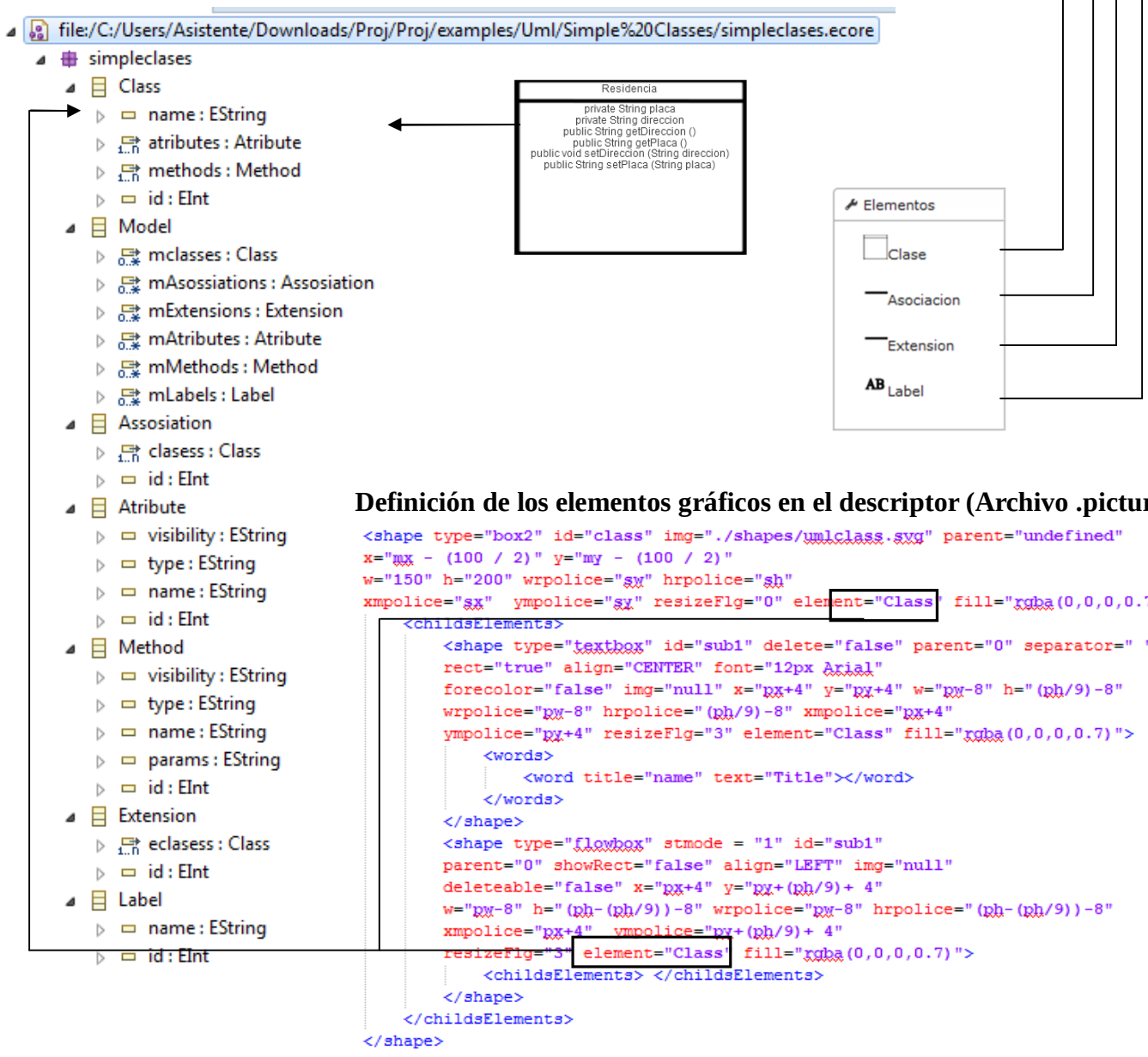
Figura 23: Representación visual de elementos ejemplo UML.

De igual manera mediante este ejemplo se muestra la capacidad que tiene la aplicación para la realización de validaciones dentro de un modelo, siguiendo los parámetros establecidos. En el caso de que dichos parámetros no se cumplan la aplicación muestra un mensaje de advertencia al usuario indicando el problema asociado.

Definición de la paleta en el descriptor (Archivo .picture)

```
<tooldef>
    <palette>
        <tool elementId="class" thumbnail="./shapes/umlclass.svg" name="Clase"></tool>
        <tool elementId="association" thumbnail="./shapes/line1.svg" name="Asociacion"></tool>
        <tool elementId="extension" thumbnail="./shapes/line1.svg" name="Extension"></tool>
        <tool elementId="label" thumbnail="./shapes/label.svg" name="Label"></tool>
    </palette>
```

Metamodelo (.ecore)



Definición de los elementos gráficos en el descriptor (Archivo .picture)

```
<shape type="box2" id="class" img="./shapes/umlclass.svg" parent="undefined"
x="mx - (100 / 2)" y="my - (100 / 2)"
w="150" h="200" wrpolice="sw" hrpolice="sh"
xmpolice="sx"  ympolice="sy" resizeFlg="0" element="Class" fill="rgba(0,0,0,0.7)">
    <childsElements>
        <shape type="textbox" id="sub1" delete="false" parent="0" separator=" "
        rect="true" align="CENTER" font="12px Arial"
        forecolor="false" img="null" x="px+4" y="py+4" w="pw-8" h="(ph/9)-8"
        wrpolice="pw-8" hrpolice="(ph/9)-8" xmpolice="px+4"
        ympolice="py+4" resizeFlg="3" element="Class" fill="rgba(0,0,0,0.7)">
            <words>
                <word title="name" text="Title"></word>
            </words>
        </shape>
        <shape type="flowbox" stmode = "1" id="sub1"
        parent="0" showRect="false" align="LEFT" img="null"
        deleteable="false" x="px+4" y="py+(ph/9)+ 4"
        w="pw-8" h="(ph-(ph/9))-8" wrpolice="pw-8" hrpolice="(ph-(ph/9))-8"
        xmpolice="px+4"  ympolice="py+(ph/9)+ 4"
        resizeFlg="3" element="Class" fill="rgba(0,0,0,0.7)">
            <childsElements> </childsElements>
        </shape>
    </childsElements>
</shape>
```

Figura 24: Definición de elementos ejemplo UML.

De la misma manera que en el ejemplo de ArchiMate, en este caso se presentarían las validaciones al modelo correspondientes y se mostrarían mensajes de advertencia y/o error en el caso de que no se cumpla una restricción establecida.

En este caso por ejemplo, si en el modelo se coloca una clase vacía se mostraría un error de multiplicidad debido a que en el metamodelo se especifico que una clase tendría mínimo un atributo y un método.

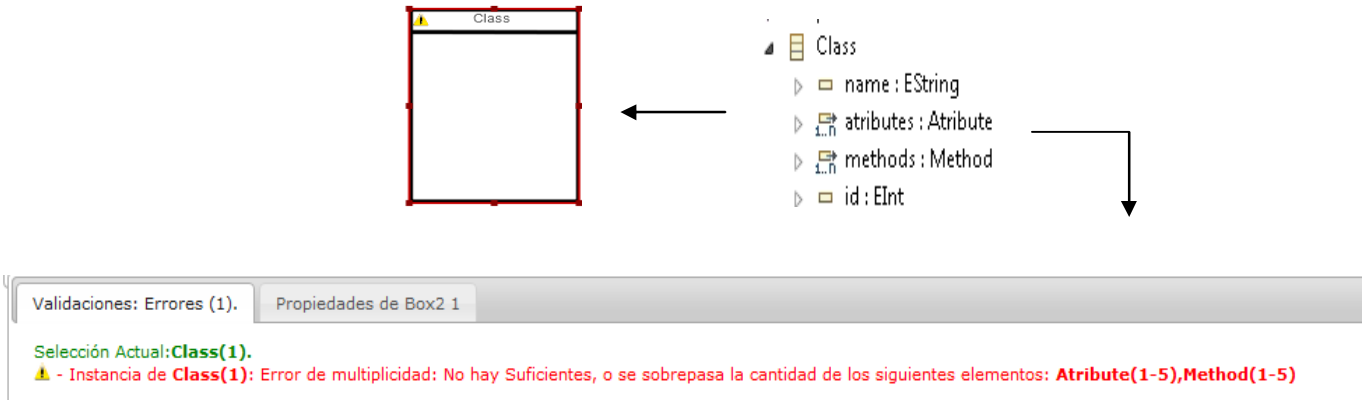


Figura 25: Ejemplo de Validaciones ArchiMate.

Para cumplir esta validación, se agregaría en tal caso uno o varios métodos según las necesidades del usuario. Estas restricciones se encuentran asociadas a las validaciones ontológicas y linguisticas establecidas en la sección 3.4. En este caso las validaciones asociadas a esta advertencia serian las siguientes: EC-OR-5 y
CRC-OR-2

Así como estos ejemplos anteriormente mencionados, la aplicación permite la creación de modelos de diferentes tipos recibiendo únicamente como parámetros, el metamodelo .ecore asociado a partir del cual queremos que nuestro modelo sea conforme y un archivo .picture en el cual se describen los elementos gráficos de la paleta del editor los cuales permitirán representar gráficamente los elementos en el modelo.

5. Conclusiones.

- Para crear diferentes editores con Diagramas web, basta con cambiar los archivos que se reciben como parámetros de entrada (el metamodelo y el descriptor) y especificándolos de acuerdo al tipo de modelo que se desea crear.

- Para el diseño de la aplicación se utilizó javascript y html debido a que son tecnologías de alta flexibilidad, que desligan el desarrollo de la Ingeniería de modelos de los framework de eclipse.

- El diseño del descriptor de visualización se realizó utilizando un esquema xml y estableciendo una separación de los conceptos de la paleta y la definición de los elementos asociados a esta. Esta abstracción es la misma que maneja GMF, pero con una sintaxis más entendible y genérica.

- En los metamodelos (.ecore) siempre es necesario crear un elemento EClass que represente el modelo para poder crear una instancia dinámica a partir de la cual se realice el tipo de modelo deseado.

- Esta herramienta únicamente valida los modelos sintácticamente. La semántica aún no se ha definido.

- Para el correcto funcionamiento de la aplicación fue necesario consultar una serie de validaciones ontológicas y lingüísticas las cuales se tienen en cuenta de manera significativa en el momento de verificar la validez del modelo.

- La función de exportar a XMI permite a los modelos creados en la herramienta ser utilizados en software destinados únicamente en eclipse.

6. Bibliografía.

[1] Ingeniería orientada a modelos. Modificado y Tomado de http://es.wikipedia.org/wiki/Ingenier%C3%ADa_orientada_a_modelos el 14 de Enero del 2014.

[2] Eclipse Modeling Framework. Modificado y Tomado de http://www.eclipse.org/modeling/emf/ el 14 de Enero del 2014.

[3] Modelos y Metamodelos. Modificado y Tomado de http://sensei.lsi.uned.es/~miguel/tesis/node27.html el 14 de Enero del 2014.

[4] GraCoT . Modificado y Tomado de http://backus1.uniandes.edu.co/~enar/dokuwiki/doku.php?id=gracot#graphical_co-creation_tool el 14 de Enero del 2014.

[5] Eugenia. Modificado y Tomado de http://www.eclipse.org/epsilon/doc/eugenia/ el 14 de Enero del 2014.

[6] Html5. Modificado y Tomado de http://es.wikipedia.org/wiki/HTML5 el 14 de Enero del 2014.

[7] Java Script. Tomado de http://es.wikipedia.org/wiki/JavaScript el 14 de Enero del 2014.

[8] Graphical Modeling Project. Tomado de http://www.eclipse.org/modeling/gmp/ el 14 de Enero del 2014.

[9] Gómez Paola, Sánchez Mario,Florez Hector, Villalobos Jorge. *Co-Creation of Models and Metamodels for Enterprise Architecture Projects*, Recuperado el 20 de Septiembre del 2013.

[10] *Tutorial de BizzDesign*, Recuperado de http://sistemas.uniandes.edu.co/~isis2403/dokuwiki/lib/exe/fetch.php?media=monitorias:monitoriaarchimate.pdf el 20 de Septiembre del 2013.

7. Agradecimientos.

- A Paola Gomez por orientarme en el desarrollo de las reglas de validación de la herramienta.
- A Mayerli Romero Díaz por la colaboración en correcciones del documento y elaboración de imágenes ilustrativas.
- A Mario Sánchez Puccini por su continua retroalimentación para el buen desarrollo del proyecto.

8. Anexos.

Descriptor de Visualización Ejemplo ArchiMate.

```
<?xml version="1.0" encoding="UTF-8"?>
<tooldef>
   <palette>
      <tool elementId="function" thumbnail="./shapes/function.svg" name="Funcion"></tool>
      <tool elementId="actor" thumbnail="./shapes/actor.svg" name="Actor"></tool>

      <tool elementId="link" thumbnail="./shapes/line2.svg" name="Relacion"></tool>

      <tool elementId="label" thumbnail="./shapes/label.svg" name="Label"></tool>

   </palette>
   <descriptor>
    <shape type="box2" id="label" img="./shapes/blank.svg" parent="undefined" x="mx - (50 / 2)" y="my - (30 / 2)" w="50" h="30" wrpolice="sw" hrpolice="sh" xmpolice="sx"  ympolice="sy" resizeFlg="0" element="Label" fill="rgba(0,0,0,0.7)">
         <childsElements>
            <shape type="textbox" id="sub1" delete="false" parent="0" separator=" " rect="true" align="CENTER" font="12px Arial" forecolor="false" img="null" x="px+4" y="py+4" w="pw-8" h="ph-8" wrpolice="pw-8" hrpolice="ph-8" xmpolice="px+4" ympolice="py+4" resizeFlg="3" element="Label" fill="rgba(0,0,0,0.7)">
               <words>
                  <word title="name" text="label"></word>
               </words>
            </shape>

         </childsElements>
      </shape>
      <shape type="box2" id="function" img="./shapes/box.svg" parent="undefined" x="mx - (100 / 2)" y="my - (100 / 2)" w="180" h="90" wrpolice="sw" hrpolice="sh" xmpolice="sx"  ympolice="sy" resizeFlg="0" element="Function" fill="rgba(0,0,0,0.7)">
         <childsElements>
            <shape type="textbox" id="sub1" delete="false" parent="0" separator=" " rect="true" align="CENTER" font="12px Arial" forecolor="false" img="null" x="px+4" y="py+4" w="pw-8" h="ph-8" wrpolice="pw-8" hrpolice="ph-8" xmpolice="px+4" ympolice="py+4" resizeFlg="3" element="Function" fill="rgba(0,0,0,0.7)">
               <words>
                  <word title="name" text="funcName"></word>
               </words>
            </shape>
            <shape type="flowbox" stmode = "1" id="sub1" parent="0" showRect="false" align="LEFT"
```

```
img="./shapes/function.svg" deleteable="false" x="px+(pw-50)+4" y="py+ 4" w="30" h="20" wrpolice="30" hrpolice="20"
xmpolice="px+(pw-50)+4"  ympolice="py+ 4" resizeFlg="3" element="Function" fill="rgba(0,0,0,0.7)">
                <childsElements> </childsElements>
            </shape>
        </childsElements>
    </shape>

    <shape type="box2" id="actor" img="./shapes/box.svg" parent="undefined" x="mx - (100 / 2)" y="my - (100 / 2)" w="180"
h="200" wrpolice="sw" hrpolice="sh" xmpolice="sx"  ympolice="sy" resizeFlg="0" element="Role" fill="rgba(0,0,0,0.7)">
        <childsElements>
            <shape type="textbox" id="sub1" delete="false" parent="0" separator=" " rect="true" align="CENTER" font="12px
Arial" forecolor="false" img="null" x="px+4" y="py+4" w="pw- 70" h="ph/9" wrpolice="pw-70" hrpolice="ph/9" xmpolice="px+4"
ympolice="py+4" resizeFlg="3" element="Role" fill="rgba(0,0,0,0.7)">
                <words>
                    <word title="name" text="roleName"></word>
                </words>
            </shape>
            <shape type="flowbox" stmode = "1" id="sub1" parent="0" showRect="false" align="LEFT" img="./shapes/actor.svg"
deleteable="false" x="px+(pw-50)+4" y="py+ 4" w="30" h="30" wrpolice="30" hrpolice="30" xmpolice="px+(pw-50)+4"
ympolice="py+ 4" resizeFlg="3" element="Role" fill="rgba(0,0,0,0.7)">
                <childsElements> </childsElements>
            </shape>
        </childsElements>
    </shape>
    <shape type="link" id="link" p1rel="funciones,actores" p2rel="funciones,actores" dash="true" fill="rgba(0,0,0,0.7)" w="3"
arrowStyle="1" arrowSize="5" arrowColor="black" element="Relation">
        <pone>
            <point x="mx" y="my"> </point>
        </pone>
        <ptwo>
            <point x="mx+50" y="my"> </point>
        </ptwo>
        <bendpoints> </bendpoints>
    </shape>


  </descriptor>
</tooldef>
```

Descriptor de Visualización Ejemplo SimpleClasses.

```
<?xml version="1.0" encoding="UTF-8"?>
<tooldef>
   <palette>
      <tool elementId="class" thumbnail="./shapes/umlclass.svg" name="Clase"></tool>
      <tool elementId="association" thumbnail="./shapes/line1.svg" name="Asociacion"></tool>
      <tool elementId="extension" thumbnail="./shapes/line1.svg" name="Extension"></tool>
      <tool elementId="label" thumbnail="./shapes/label.svg" name="Label"></tool>
   </palette>
   <descriptor>


      <shape type="box2" id="class" img="./shapes/umlclass.svg" parent="undefined" x="mx - (100 / 2)" y="my - (100 / 2)"
w="150" h="200" wrpolice="sw" hrpolice="sh" xmpolice="sx"  ympolice="sy" resizeFlg="0" element="Class"
fill="rgba(0,0,0,0.7)">
         <childsElements>
            <shape type="textbox" id="sub1" delete="false" parent="0" separator=" " rect="true" align="CENTER" font="12px
Arial" forecolor="false" img="null" x="px+4" y="py+4" w="pw-8" h="(ph/9)-8" wrpolice="pw-8" hrpolice="(ph/9)-8"
xmpolice="px+4"  ympolice="py+4" resizeFlg="3" element="Class" fill="rgba(0,0,0,0.7)">
               <words>
                  <word title="name" text="Title"></word>
               </words>
            </shape>
            <shape type="flowbox" stmode = "1" id="sub1" parent="0" showRect="false" align="LEFT" img="null"
deleteable="false" x="px+4" y="py+(ph/9)+ 4" w="pw-8" h="(ph-(ph/9))-8" wrpolice="pw-8" hrpolice="(ph-(ph/9))-8"
xmpolice="px+4"  ympolice="py+(ph/9)+ 4" resizeFlg="3" element="Class" fill="rgba(0,0,0,0.7)">
               <childsElements> </childsElements>
            </shape>
         </childsElements>
      </shape>
      <shape type="link" id="association" p1rel="clasess" p2rel="clasess" dash="false" fill="rgba(0,0,0,0.7)" w="3"
arrowStyle="1" arrowSize="10" arrowColor="no" element="Assosiation">
         <pone>
            <point x="mx" y="my"> </point>
         </pone>
         <ptwo>
            <point x="mx+50" y="my"> </point>
         </ptwo>
         <bendpoints> </bendpoints>
      </shape>
```

```
<shape type="link" id="extension" p1rel="eclasess" p2rel="eclasess" dash="false" fill="rgba(0,0,0,0.7)" w="3" arrowStyle="1" arrowSize="10" arrowColor="white" element="Extension">
    <pone>
        <point x="mx" y="my"> </point>
    </pone>
    <ptwo>
        <point x="mx+50" y="my"> </point>
    </ptwo>
    <bendpoints> </bendpoints>
</shape>
<shape type="textbox" id="Atribute" delete="true" parent="2" separator=" " rect="true" align="CENTER" font="12px Arial" forecolor="undefined" img="null" x="px+4" y="px+4" w="pw-8" h="(ph/9)-8" wrpolice="pw-8" hrpolice="(ph/9)-8" xmpolice="px+4"  ympolice="py+4" resizeFlg="3" element="Atribute" fill="rgba(0,0,0,0.7)">
    <words>
        <word title="visibility" text="private"></word>
        <word title="type" text="int"></word>
        <word title="name" text="name"></word>
    </words>
</shape>
<shape type="textbox" id="Method" delete="true" parent="2" separator=" " rect="true" align="CENTER" font="12px Arial" forecolor="undefined" img="null" x="px+4" y="px+4" w="pw-8" h="(ph/9)-8" wrpolice="pw-8" hrpolice="(ph/9)-8" xmpolice="px+4"  ympolice="py+4" resizeFlg="3" element="Method" fill="rgba(0,0,0,0.7)">
    <words>
        <word title="visibility" text="private"></word>
        <word title="type" text="int"></word>
        <word title="name" text="method"></word>
        <word title="params" text="()"></word>
    </words>
</shape>
<shape type="box2" id="label" img="./shapes/blank.svg" parent="undefined" x="mx - (50 / 2)" y="my - (30 / 2)" w="50" h="30" wrpolice="sw" hrpolice="sh" xmpolice="sx"  ympolice="sy" resizeFlg="0" element="Label" fill="rgba(0,0,0,0.7)">
    <childsElements>
        <shape type="textbox" id="sub1" delete="false" parent="0" separator=" " rect="true" align="CENTER" font="12px Arial" forecolor="false" img="null" x="px+4" y="py+4" w="pw-8" h="ph-8" wrpolice="pw-8" hrpolice="ph-8" xmpolice="px+4" ympolice="py+4" resizeFlg="3" element="Label" fill="rgba(0,0,0,0.7)">
            <words>
                <word title="name" text="label"></word>
            </words>
        </shape>

    </childsElements>
</shape>
```

```
    </descriptor>
</tooldef>
```