Universidad de los Andes

Documento de proyecto de grado

WebPicture: Generador de editores de modelos basados en EMF y Picture

Presentado a:
Departamento de ingeniería de sistemas y computación

Presentado por:
Andrés Felipe Guzmán Bautista.

Asesores:
Mario Eduardo Sánchez Puccini
John Casallas

Bogotá
Colombia
2014

Tabla de contenido

1. Introducción

En la practica de arquitectura empresarial, el uso de modelos y representaciones abstractas de la realidad de una organización desde varios puntos de vista aporta un amplio valor a su ejercicio [1]. Los artefactos utilizados en este ejercicio son creados con el fin de representar y/o modelar el estado o la realidad de una organización [2]. Los artefactos utilizados en la arquitectura empresarial son herramientas formales que se estructuran entorno a la especificación de un metamodelo.

Los metamodelos describen las propiedades estructurales tales como reglas, relaciones y construcción de un modelo. Sin embargo el metamodelo por si solo no es suficiente para construir un modelo. Un modelo se constituye de dos partes la primera es el metamodelo y segundo la especificación de su representación grafica. Pese a que un modelo solo cuenta con una sola especificación estructural (metamodelo), un mismo modelo puede tener diferentes representaciones graficas [3].

En los últimos años la arquitectura empresarial ha evolucionado en un ejercicio basado en el soporte de toma de decisiones basado en hallazgos. En la practica los hallazgos o descubrimientos se presentan al cliente en forma de entregables. Los entregables de arquitectura empresarial consiste en cualquier documento, artefacto o producto tangible que le permita al arquitecto justificar las decisiones tomadas [4].

Los artefactos generados en la arquitectura empresarial consisten principalmente en documentos que contienen generalmente un conjunto de modelos que pueden representar el estado actual o esperado de la organización y justifican aspectos clave de las decisiones tomadas por el arquitecto.

En la actualidad la arquitectura empresarial se ha convertido en una practica formalizada que cuenta con metodologías, herramientas y *frameworks* formales alineados con estándares globales. No obstante los modelos establecidos por *frameworks* como TOGAF entre otros pueden quedarse cortos en algunas circunstancias, en las cuales el arquitecto requiere representar decisiones o aspectos de la realidad sin embargo no le es posible ya que con los metamodelos disponibles pueden no soportar dichos aspectos [5].

Adicionalmente en el ámbito de los proyectos de arquitectura, el gobierno y control de los artefactos generados hacen parte de la complejidad y riesgos con los que debe se debe lidiar. En la practica el arquitecto empresarial debe contar con practicas, habilidades y herramientas que le permitan disminuir la complejidad generada en la creación de artefactos.

Es por esto que surge la necesidad de contar con herramientas que le brinden al arquitecto la capacidad de primero poder representar en modelos aspectos no soportados por los metamodelos estándares y segundo disminuir efectivamente la complejidad generada a partir de la construcción de artefactos.

Un ambiente integrado de arquitectura empresarial consiste en un extenso marco de herramientas concebidas para el diseño, construcción, difusión y análisis de artefactos de arquitectura.

Dicho ambiente cuenta con herramientas entre las cuales se destacan editores especializados, gestores documentales, repositorios de modelos, herramientas para construcción de editores y herramientas de análisis. Para el ejercicio de arquitectura empresarial contar con un ambiente integrado para modelado tiene amplias ventajas desde el ámbito de proyecto de arquitectura, principalmente la reutilización y cocreación de modelos.

En el marco de los proyectos de investigación del grupo de tecnologías de información y construcción de software (TISCw) de la universidad de los Andes, se han presentado distintos acercamientos y avances en la creación de un ambiente integrado de modelado para arquitectura; no obstante los prototipos construidos para distintas funcionalidades se han implementado de manera independiente por lo cual no se cuenta aun con ambiente completamente integrado y todavía no se ha estudiado de manera completa los requerimientos del ambiente en su totalidad.

No obstante no se existe una herramienta que permita al arquitecto crear, modificar y extender modelos; y a la vez controlar los artefactos generados de su ejercicio. En este documento se detalla la construcción de WebPicture una herramienta para la generación de editores ad hoc basado en tecnologías web.

WebPicture toma metamodelos construidos sobre el framework de EMF y se integra con Picture un lenguaje creado para describir de manera formal la representación grafica de un modelo para construir un editor grafico para el modelo especificado.

1.1. Resumen

En la actualidad la arquitectura empresarial se ha convertido en una practica formalizada que cuenta con metodologías, herramientas y *frameworks* formales alineados con estándares globales. No obstante en el desarrollo del ejercicio los metamodelos definidos pueden quedarse cortos en algunas circunstancias.

Es por esto que surge la necesidad de contar con herramientas que le brinden al arquitecto la capacidad de primero poder representar en modelos aspectos no definidos por metamodelos convencionales y segundo disminuir efectivamente la complejidad generada a partir de la construcción de artefactos.

Un ambiente integrado de arquitectura empresarial consiste en un extenso marco de herramientas concebidas para el diseño, construcción, difusión y análisis de artefactos de arquitectura.

En este documento se detalla la construcción de WebPicture una herramienta para la generación de editores ad hoc basado en tecnologías web.

1.2. Objetivos

- Generar un producto flexible, capaz de integrarse fácilmente con los demás componentes del ambiente.
- Crear un generador de editores ad hoc basado en tecnologías web.
- Crear una herramienta capaz de generar editores web utilizando como componentes de entrada metamodelos construidos con EMF y cuya representación grafica sea denotada con el lenguaje Picture.

1.3. Motivación

En la actualidad la arquitectura empresarial se ha convertido en una practica formalizada que cuenta con metodologías, herramientas y *frameworks* formales alineados con estándares globales. No obstante los modelos establecidos por *frameworks* como TOGAF entre otros pueden quedarse cortos en algunas circunstancias, en las cuales el arquitecto requiere representar decisiones o aspectos de la realidad sin embargo no le es posible ya que con los metamodelos disponibles pueden no soportar dichos aspectos.

Existen tecnologías como GMF que permiten crear editores gráficos ad hoc basado en EMF [6]. Pese a ser altamente configurable GMF es complejo de utilizar y requiere primero la instalación de componentes adicionales sobre Eclipse y segundo que el usuario escriba las anotaciones sobre el metamodelo para generar el editor grafico. GMF y EMF son tecnologías ampliamente utilizadas en la industria, sin embargo para utilizarlas se requiere obligatoriamente contar con una instalación de Java.

El grupo TISCw de la universidad de los Andes ya ha realizado distintos aportes para cambiar el paradigma de la generación de editores en GMF. Desde esta perspectiva se propuso entonces la separación del modelo en su metamodelo construido sobre EMF y de su representación grafica.

Es por esto que se propuso la creación de Picture un lenguaje formal y estructurado construido sobre XText. Picture le permite al usuario describir formalmente la representación grafica de los elementos del metamodelo. Sin embargo los prototipos desarrollados aun dependían de la plataforma Eclipse.

WebPicture pretende transformar el proceso de generación de editores primero al tomar la aproximación de la separación del modelo en su metamodelo construido en EMF y su representación grafica descrita en el lenguaje Picture. Segundo agilizar el proceso al tomar el metamodelo y la representación grafica para generar editores web capaces de realizar validaciones ontológicas y lingüísticas.

1.4. Resultados esperados

- Crear una herramienta capaz de generar editores web utilizando Picture para describir la representación grafica y metamodelos construidos sobre EMF.
- Generar editores capaces de realizar validaciones lingüísticas y ontológicas sobre los modelos generados.

- Crear una herramienta que soporte una interfaz web y requiera configuraciones simples.
- Generar una herramienta capaz de integrarse fácilmente con los componentes del ambiente integrado de arquitectura empresarial.

1.5. Antecedentes

- Greta
  Greta consiste en un framework construido sobre una arquitectura basada en Eclipse capaz de soportar operaciones de composición sobre modelos. Asimismo capaz de soportar operaciones de composición sobre elementos gráficos [7].
- Web Picture
  Proyecto de grado de Edgar Sandoval, consiste en un editor web de modelos, capaz de permitir al usuario cargar metamodelos en formato .ecore y archivos .picture (contiene las relaciones entre elementos gráficos definidos y elementos del metamodelo) para construir y editar modelos.

- Picture Maker
  Proyecto de grado de Oscar Martínez, consiste en un editor web de modelos capaz de realizar validación de modelos. Este editor permite la edición de elementos gráficos del modelo (archivos .picture) y asimismo genera el correspondiente código HTML para visualizar el modelo.

2. Marcos de referencia

2.1. Modelos y metamodelos

Un modelo es la representación abstracta y simplificada de una realidad o un sistema (SuS) [8]. Los metamodelos son una representación abstracta de un modelo en si. De acuerdo con la especificación de OMG y MOF *un metamodelo es un modelo que define el lenguaje y la estructura para construir un modelo* [9]. Un metamodelo define las propiedades estructurales, reglas, objetos y relaciones para la construcción de un modelo.

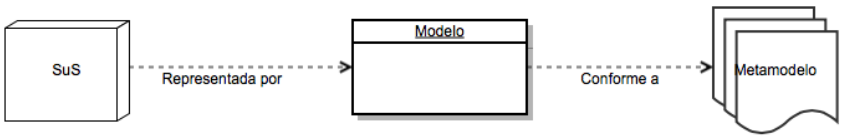

Figura 1: Estructura básica de los modelos y metamodelos

En la *Figura 1* se muestra la relación de conformidad entre un modelo y su metamodelo, entre un metamodelo se relacionan con los metamodelos. Por definición los modelos tienen una relación de conformidad a la especificación de un metamodelo definido, dicha relación de conformidad tiene dos partes, conformidad lingüística y conformidad ontológica.

La conformidad lingüística consiste implica que el modelo cumpla con la especificación lingüística del metamodelo. Por su parte la conformidad ontológica implica que el modelo cumpla las propiedades estructurales definidas por el metamodelo [9].

2.1.1. Conformidad

Por definición los modelos son conformes lingüística y ontológicamente conformes a la especificación de un metamodelo.

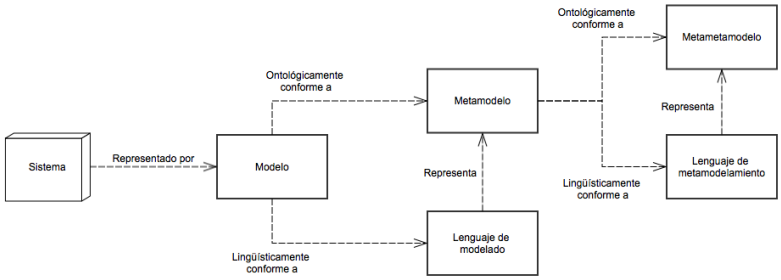

Figura 2: Modelos y conformidad [1]

La conformidad ontológica implica que los modelos son estructuralmente conformes a la definición de un metamodelo. Al mismo tiempo los modelos son lingüísticamente conformes a un lenguaje de modelado que representa al metamodelo. De la misma forma al ser un modelo per se, el metamodelo debe cumplir con las mismas propiedades de conformidad que un modelo.

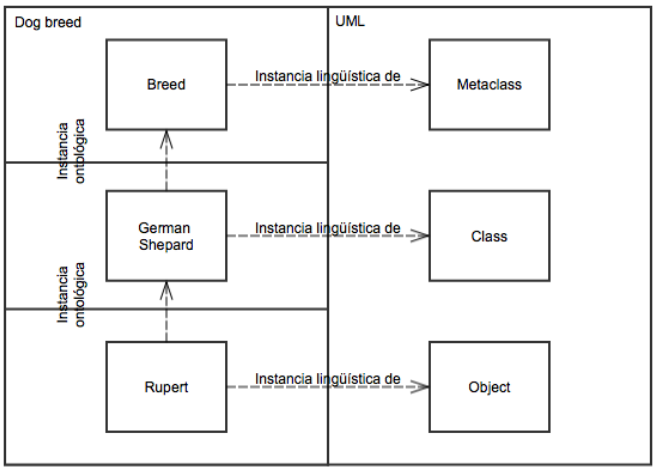

Figura 3: Conformidad lingüística y ontológica [2]

[1] Adaptado de Genova G. What is a metamodel: the OMG's metamodeling infrastructure. In: Modeling and metamodeling in model driven development, Varsovia, Polonia, 2009

[2] Adaptado de Gašević D., Djurić D, Devedzic V. Model driven engineering. In: Model driven engineering and ontology development capitulo 4.

Para explicar de manera practica la conformidad ontológica y lingüística de un modelo se propone el ejemplo de la *Figura 2*. Suponga que el lenguaje de modelamiento para este ejemplo es UML. En el modelo *Dog breed* el objeto UML *Rupert* es una instancia ontológica de la clase *German Shepard* y es ontológicamente conforme con esta. *German Shepard* es una instancia ontológica de la metaclase *Breed* y a su vez es ontológicamente conforme con esta. Adicionalmente en este ejemplo los elementos del modelo *Dog breed* son lingüísticamente conformes a la especificación del metamodelo de UML.

2.2. Model driven engineering (MDE)

MDE consiste en una aproximación al desarrollo de software en la cual se sugiere primero desarrollar un modelo del sistema para luego transformar dicho modelo en ejecutables [9]. Los modelos juegan un papel clave en MDE ya que son el insumo clave para el proceso de transformación y generación de artefactos.

La idea básica detrás MDE consiste en utilizar modelos en diferentes niveles de abstracción y transformar dichos modelos en artefactos de software. No obstante para realizar el proceso de transformación se requieren lenguajes de dominio especifico (DSL) para especificar la notación de los modelos y herramientas capaces de realizar la transformación y generar los artefactos.

Los DSL son lenguajes especializados para describir modelos, estos lenguajes pueden ser gráficos y/o textuales. Por otra parte las herramientas responsables de hacer la transformación de modelos en artefactos de software requieren de modelos representados con un DSL que puedan consumir y transformar en código [10].

MDE tiene como premisa disminuir la complejidad del proceso de desarrollo, minimizar los errores potenciales sobre el código y aumentar la productividad.

2.2.1. Eclipse Modeling Framework (EMF)

Eclipse Modeling Framework o EMF es un conjunto de herramientas desarrolladas por *Eclipse Foundation*. EMF brinda un framework de desarrollo orientado a MDE que permite crear modelos y transformarlos en código. Por medio de un modelo especificado en XMI/Ecore EMF puede generar el código Java base del modelo y así mismo generar un conjunto de editores básicos para manipular el modelo [11].

2.2.2. Ecore

Ecore es el formato estándar aceptado por EMF para la creación de modelos. La sintaxis del lenguaje esta basada en XMI. Por otra parte los elementos estructurales del lenguaje provienen de elementos definidos de MOF. A continuación se describen los elementos básicos del lenguaje utilizados para describir la estructura de cualquier modelo [9].

- EClass: Se utiliza para representar una clase, este tiene un nombre y puede tener atributos (EAttributes).

- EAttribute: Es utilizado para representar un atributo, tiene un nombre y un tipo de dato.
- EReference: Se utiliza para representar una relación entre dos clases del modelo.
- EDataType: Se utiliza para representar un tipo de dato que puede pertenecer al conjunto de primitivas definidas o datos representados como objetos (Ej. Java.util.Date)

2.3. XText

XText es una herramienta diseñada para la creación de lenguajes textuales de dominio especifico (DSL textuales). XText provee un conjunto de herramientas que a partir de la especificación de una gramática permiten generar artefactos para crear editores textuales para el lenguaje [12]. A continuación se describen los artefactos generados por Xtext a partir de la especificación de una gramática [13]:

- Metamodelo del lenguaje.
- Código generado a partir del metamodelo creado.
- Herramientas de validación del lenguaje (*Lexer* y *parser*)
- Editor textual del lenguaje.

XText es una herramienta sumamente poderosa que el agiliza el proceso de creación de lenguajes de dominio especifico y es capaz de generar herramientas de edición. Las herramientas generadas permiten ejecutar un editor basado en Eclipse validar la sintaxis escrita por con respecto al metamodelo del lenguaje.

2.4. Picture

Picture es un lenguaje de dominio especifico creado para describir la representación grafica de un modelo de forma independiente a la especificación del metamodelo [14]. A continuación se presenta el metamodelo de la versión actual del lenguaje.

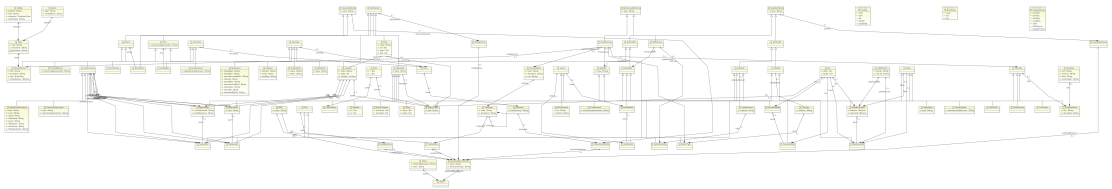

Figura 4: Metamodelo de Picture

Picture fue modelado y construido utilizando el framework de XText por lo que los elementos que se describan utilizando el lenguaje pueden ser extraídos a partir del metamodelo y ser validados utilizando las herramientas del framework.

3. Acerca de WebPicture

WebPicture es una herramienta web diseñada para generar editores de modelos de arquitectura empresarial ad hoc. El objetivo de WebPicture es dar flexibilidad al ejercicio de arquitectura empresarial en el ámbito del modelamiento y al mismo tiempo mantener la formalidad al brindar al arquitecto la posibilidad de crear sus propios modelos.

WebPicture toma la especificación del metamodelo en formato *Ecore* y su representación grafica descrita en el lenguaje *Picture*. Estos elementos son directamente especificados por el usuario para generar un editor grafico. La herramienta genera editores construidos sobre tecnologías web, ligeros, fáciles de utilizar y compatibles con la mayoría de los navegadores web disponibles.

El proceso de generación de editores tiene tres partes. La primera consiste en la validación de la representación grafica con respecto al metamodelo del lenguaje y su validación con el metamodelo cargado. La segunda consiste en la transformación del metamodelo y su representación grafica en elementos web y la generación de reglas estructurales del modelo. Por ultimo el proceso reúne todos los elementos generados en un script web que puede ser reutilizado para la creación de distintos modelos.

3.1. Arquitectura de la herramienta

WebPicture es una aplicación web distribuida en tres capas. Una capa de presentación, una capa de lógica y una de datos. En la *Figura 5* se muestra la arquitectura por capas de la herramienta.

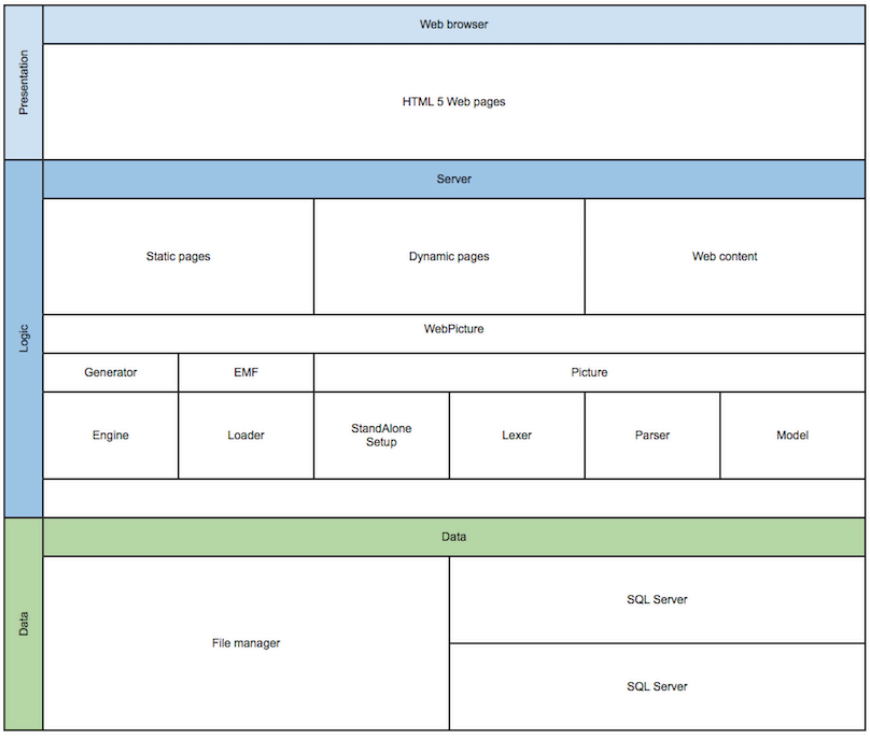

Figura 5: Diagrama de arquitectura por capas de la herramienta

La capa de presentación contiene las paginas web en HTML que el usuario utiliza para interactuar con la aplicación.

La capa de lógica agrupa los componentes responsables de conectarse con el usuario vía HTML y los componentes responsables de manejar la lógica detrás de la validación de elementos y la generación de editores.

Por ultimo la capa de datos agrupa un manejador de archivos, un manejador transaccional y un servidor de base de datos.

3.1.1. Capa de presentación

La capa de presentación contiene las paginas web utilizadas por el cliente para interactuar con la aplicación dichas paginas están construidas con tecnología HTML 5. En esta capa se muestra y recoge información utilizando contenedores web dinámicos y/o estáticos.

3.1.2. Capa de lógica

La capa de lógica agrupa los componentes responsables de conectarse con el usuario vía HTML.

En esta capa también se encuentran los componentes de validar los elementos cargados por el usuario. Esta capa cuenta con tres componentes clave. Primero el componente *WebPicture Engine* se encarga de orquestar el proceso de generación de editores y validar los elementos cargados por el usuario. Segundo el componente *EMF Loader* es responsable de cargar el metamodelo y transformarlo en una estructura de datos intermedia. Por ultimo el componente *Picture StandAlone Setup* se ocupa de cargar y validar la implementación del lenguaje Picture cargada por el usuario.

En esta capa se hacen las transformaciones de información para generar los elementos gráficos y las reglas del editor.

3.1.3. Capa de datos

En la capa de datos se agrupan tres componentes responsables de la persistencia de la información de la herramienta. El primer componente *FileManager* es el modulo responsable de administrar y almacenar la información real de los editores generados (scrips, recursos, contenido…) en el servidor. El segundo componente *SQL Sever* corresponde a un servidor de base de datos donde se almacena metainformación de los editores creados. Finalmente el componente *SQLManager* se encarga de conectarse la base de datos para extraer o insertar información de esta.

3.2. Puntos de vista y modelos de arquitectura

A continuación se presentan distintos puntos de vista y modelos con el objetivo de explicar la arquitectura implementada.

3.2.1. Vista de contexto

En la vista de contexto presentada en la *Figura 6* se muestra como el usuario interactúa con la herramienta y como interactúan los componentes de las capa de lógica y la capa de datos.

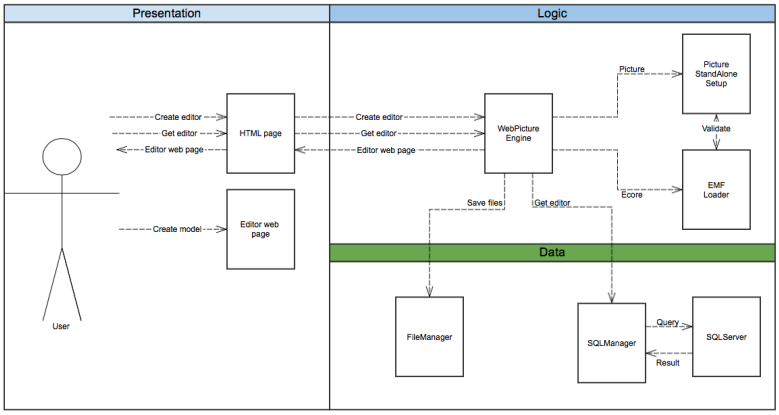

Figura 6: Vista de contexto de la herramienta

En esta vista se especifica como el usuario interactúa para crear un editor o utilizar un editor existente.

En el caso de que el usuario desee crear un nuevo editor debe cargar los archivos que describen el metamodelo (Ecore) y la representación grafica (Picture), estos archivos deben pasar por un proceso de validación y transformación para generar un editor.

En el caso de que el usuario desee utilizar un editor existente, la herramienta provee mecanismos para recuperar el editor.

Así mismo en el momento que el usuario obtiene el editor por cualquiera de los dos mecanismos el servidor deja de interactuar con el cliente y en este momento el editor es ejecutado en el navegador del cliente.

3.2.2. Vista de despliegue

En la vista de despliegue de la Figura 7 se muestra la localización de los nodos de ejecución de la herramienta. En esta vista los distintos componentes se ejecutan en tres dispositivos, una maquina cliente que cuenta con un navegador web, una maquina servidor y un servidor de base de datos.

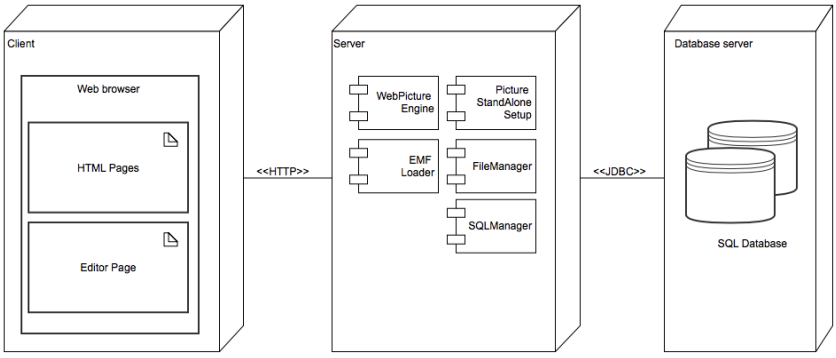

Figura 7: Vista de despliegue de la herramienta

La maquina del cliente visualiza y ejecuta las paginas web generadas en el servidor, esta se comunica con el servidor vía protocolo HTTP. Desde este nodo se envían las peticiones y reciben del servidor.

Por su parte en la maquina servidor se realizan las validaciones y la transformación de el metamodelo y la representación grafica para generar un editor web. Los editores web generados se son guardados en el sistema de archivos del servidor.

En el servidor de base de datos se almacena la metainformación de los editores generados. Este nodo se comunica con el servidor vía protocolo JDBC[3]. Este nodo permite realizar consultas SQL para manipular la metainformación de los editores.

[3] What is JDBC: http://publib.boulder.ibm.com/infocenter/rbhelp/v6r3/index.jsp?topic=%2Fcom.ibm.redbrick.doc6.3%2Fciacg%2Fciacg33.htm

3.2.3. Vista de información

3.2.3.1. Modelo estático de datos

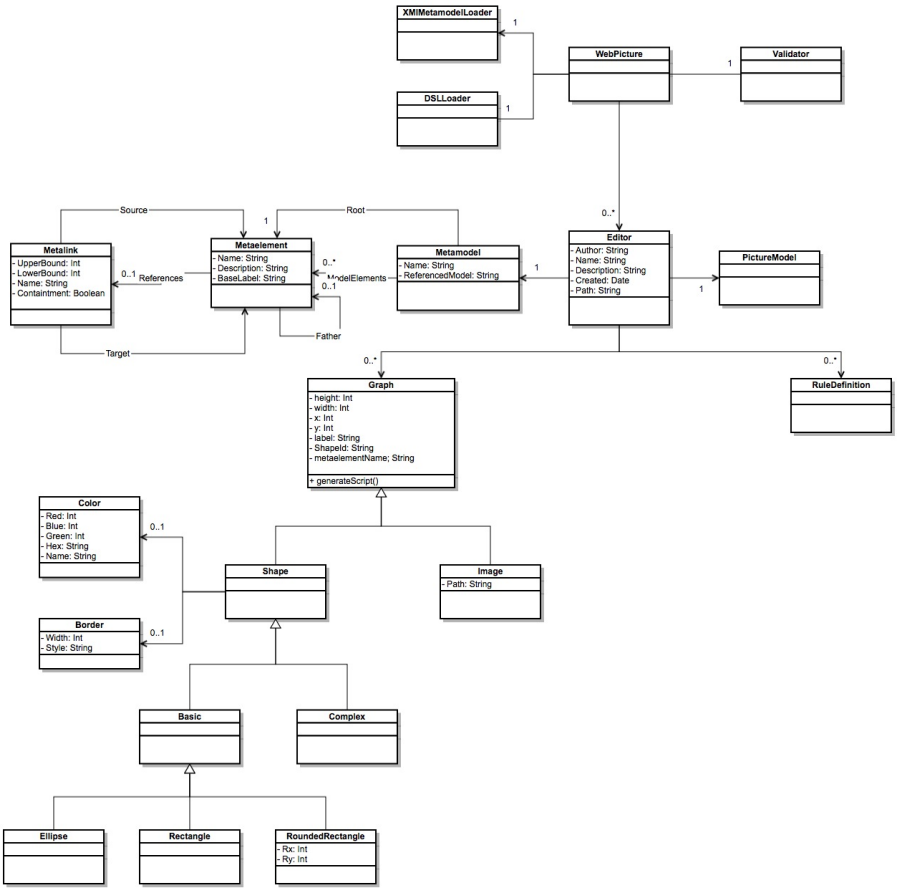

Figura 8: Modelo estático de datos

Entidad	Descripción
WebPicture	Corresponde a la entidad principal del modelo de datos, es la responsable de administrar los editores generados
Validator	Corresponde a la entidad responsable de validar elementos del metamodelo con la especificación de picture
XMIMetamodelLoader	Corresponde a la entidad responsable de cargar al sistema el metamodelo
DSLLoader	Corresponde a la entidad responsable de cargar la implementación del lenguaje y transformarlo en un modelo
Editor	Representa a un editor web generado
Metamodel	Representa al metamodelo cargado por el usuario
Metaelement	Representa un elemento del metamodelo cargado por el usuario
Metalink	Representa una relación entre dos elementos del metamodelo cargado
PictureModel	Representa el modelo del lenguaje Picture obtenido de la especificación del usuario
RuleDefinition	Representa una regla estructural generada
Graph	Representa un elemento grafico del modelo
Image	Representa un elemento grafico con una imagen de fondo

Shape	Representa una forma básica
Color	Representa un color de una forma básica
Border	Representa el borde de una forma básica
Complex	Representa una forma compleja
Basic	Representa una forma geométrica básica
Ellipse	Representa una forma circular
Rectangle	Representa una forma rectangular
RoundedRectangle	Representa una forma rectangular con bordes biselados

Tabla 1: Catalogo de entidades de WebPicture

3.2.3.2. Modelo de flujo de información

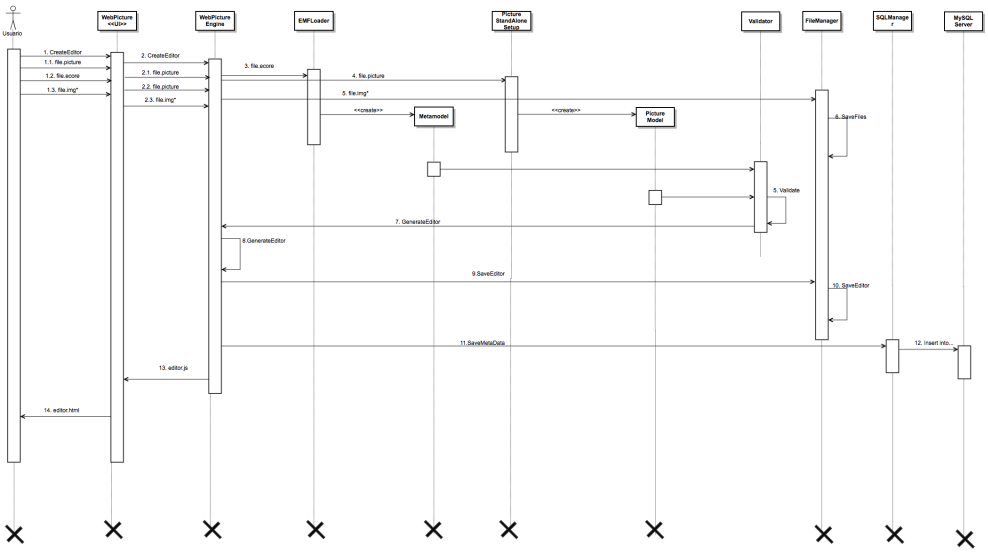

Figura 9: Diagrama de secuencia creación de editor

3.2.3.3. Estructura de la base de datos SQL

En esta sección se presenta la estructura de las tablas en la base de datos SQL (*Figura 10*). Se proponen dos Tablas una en la que se persista la información de los editores generados y una segunda en la cual se persiste la información de los diagramas creados.

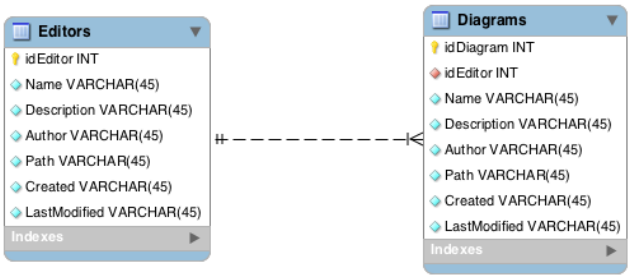

Figura 10: Estructura de las tablas en la base de datos

3.3. Tecnologías utilizadas

3.3.1. Java

Java es un lenguaje de programación orientado a objetos capaz de operar en distintas plataformas. Los componentes de lógica de la herramienta están implementados en Java.

3.3.2. HTML 5

HTML es el estándar global para la descripción de documentos web [15]. HTML 5 es la ultima versión del lenguaje. En esta versión el lenguaje ofrece nuevas capacidades. WebPicture utiliza como mecanismo de presentación interfaces construidas sobre HTML 5. En especial WebPicture aprovecha las capacidades graficas que ofrece el lenguaje para generar editores gráficos utilizando elementos de JavaScript y CSS3.

3.3.3. JavaScript

JavaScript es el estándar utilizado en para el desarrollo de aplicaciones web dinámicas. JavaScript es capaz de integrarse directamente sobre contendedores desarrollados sobre HTML [16].

WebPicture utiliza el estándar JavaScript para la construir las partes interactivas de la plataforma e implementar las funcionalidades del editor generado.

La herramienta usa específicamente la librería Joint JS para crear los elementos gráficos de los editores generados.

3.3.3.1. Joint JS

Joint JS es una librería JavaScript diseñada para la creación y visualización de diagramas sobre HTML 5 [17].

WebPicture utiliza los elementos provistos por la API de Joint JS bajo los términos de uso provistos por Mozilla Public Licence 2.0.

3.3.3.2. SVG

SVG (Scalable vector graphics) consiste en lenguaje marcado basado en XML para describir elementos gráficos en dos dimensiones [18]. Los elementos gráficos provistos por la librería se basan en el estándar SVG.

3.3.4. CSS3

CSS es un estándar para definir el esquema para mostrar elementos HTML en pantalla. HTML especifica la estructura de los elementos y CSS especifica como se ven. CSS3 es el ultimo estándar de CSS [19].

WebPicture esta basado en los principios de diseño de CSS de Pure CSS bajo los términos de uso provistos por Yahoo BSD Licence.

3.3.5. EMF

EMF brinda un framework de desarrollo orientado a MDE que permite crear modelos y transformarlos en código. WebPicture utiliza el las herramientas provistas por EMF para acceder a los elementos del metamodelo cargado por el usuario para obtener información de este y generar artefactos intermedios para generar el editor.

3.3.6. XText – Picture

XText es una herramienta diseñada para la creación de lenguajes textuales de dominio especifico (DSL textuales). La herramienta utiliza las herramientas provistas por XText y EMF para validar la estructura del archivo Picture y obtener información del modelo del lenguaje.

3.3.7. Apache Tomcat

Apache Tomcat es un contenedor de aplicaciones web open source. El prototipo de aplicación web de la herramienta corre sobre este contenedor.

3.3.8. MySQL Server

MySQL Server es una especificación open source para operar servidores de bases de datos. WebPicture utiliza como mecanismo de persistencia el software y las interfaces para guardar metainformación de los editores construidos.

3.4. Generación de editores

A nivel del servidor el proceso de generación de editores comienza cuando se obtienen los archivos del metamodelo, la representación grafica y los recursos adicionales (imágenes de cualquier formato). Estos archivos deben pasar primero por un proceso de validación con el objetivo de verificar su estructura y completitud.

Una vez se ha validado la estructura de los archivos se hace un proceso de transformación para extraer la información de los archivos y modelar el editor web. El proceso tiene dos partes, una es la transformación del metamodelo cargado en un metamodelo intermedio (*Figura 11*). Este representa la estructura del metamodelo, contiene las características graficas descritas por la representación y a partir del cual se genera el código JavaScript del editor generado con los elementos gráficos y las reglas estructurales que debe seguir el editor. La segunda parte consta en la transformación del contenido del archivo Picture en un modelo basado en XMI. Este modelo contiene la misma información del Picture. No obstante este modelo tiene un formato estructurado del cual se puede extraer fácilmente la información contenida. Esta transformación se realiza utilizando las herramientas provistas por XText.

A continuación se debe revisar primero que el archivo Picture sea la representación correspondiente al metamodelo y que se cuente con la totalidad de los archivos adicionales requeridos descritos en el Picture. En esta etapa se completa el metamodelo intermedio que se genero al inicio, al tomar información del archivo Picture.

Una vez el metamodelo intermedio este completo se puede generar el código JavaScript para los elementos gráficos y las reglas estructurales del editor. Por ultimo una vez se ha generado el código del editor este se guarda en un directorio del servidor y se envía vía HTTP el código del editor generado.

En las siguientes secciones se mostrara en detalle cada uno de los pasos para la generación del código del editor.

3.4.1. Validación inicial

El proceso de validación inicial tiene tres pasos. El primer paso consiste en validar el contenido del archivo con la representación grafica con respecto a la definición de la gramática. El segundo paso consiste en revisar la estructura del archivo del metamodelo. Si en alguno de estos pasos de verificación se encuentra alguna inconsistencia el proceso de creación de editores se revierte y se notifica al usuario sobre el error.

3.4.1.1. Validación Picture

En este paso se hace una validación del contenido del archivo Picture cargado con respecto a la especificación de la gramática definida para el lenguaje. En este paso se genera un modelo creado a partir de dicho archivo del cual se puede primero obtener una lista de errores con respecto a la gramática y segundo se pueden extraer información.

3.4.1.2. Validación Ecore

En este paso se verifica la estructura de los elementos del metamodelo. Se debe validar primero que la estructura del archivo sea correcta y segundo se debe validar que no existan elementos con nombres repetidos. En este paso se genera el esqueleto del modelo intermedio que representa el metamodelo y del cual se generara el código base del editor.

3.4.2. Validación Picture – Ecore

A continuación se debe validar que el contenido del archivo Picture corresponda con los elementos descritos en el metamodelo y que se cuenta con todos los archivos adicionales descritos en el Picture. Se comienza por extraer y validar el elemento raíz del modelo. Luego se extraen y se valida que los elementos gráficos descritos en el Picture se encuentren en el metamodelo. En esta etapa se completa el metamodelo

intermedio que se genero y a partir de este se puede generar el código JavaScript del editor web.

3.4.3. Transformaciones

En esta sección se detallan las transformaciones que hace la herramienta. Estas transformaciones facilitan la extracción de información y la generación de código JavaScript.

3.4.3.1. Generación del modelo intermedio

El archivo Ecore cargado al servidor contiene la descripción de la estructura del metamodelo. A partir de este archivo se podría generar un modelo ejecutable con las entidades y relaciones descritas en este. Sin embargo algunos elementos del metamodelo no son relevantes al momento de generar un modelo grafico por lo cual con el objetivo de reducir la complejidad y optimizar el proceso se decidió generar un modelo intermedio generado a partir de los elementos descritos en el metamodelo.

La estructura del modelo intermedio se construye a partir de la especificación descrita por las entidades (EClass) y las referencias (EReferences) descritas en el metamodelo. A continuación en la *Figura 11* se presenta la especificación del metamodelo intermedio propuesto.

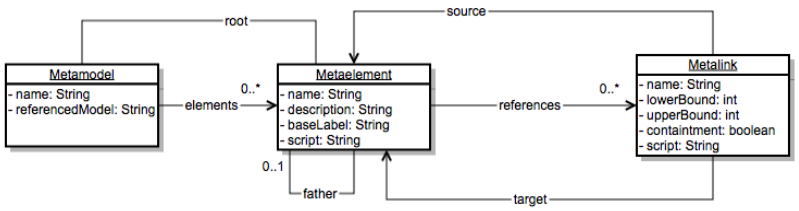

Figura 11: Especificación del metamodelo intermedio

3.4.3.2. Generación del modelo del lenguaje

Al iniciar el proceso de creación de editores el usuario carga un archivo textual escrito en lenguaje Picture. Inicialmente este archivo no cuenta con un formato estructurado del cual sea posible extraer información fácilmente. XText permite generar un modelo basado en XMI a partir de la definición de una gramática y de un archivo fuente con escrito con la especificación de la gramática. Este modelo contiene la misma información descrita en el archivo Picture.

3.4.3.3. Procesamiento de elementos gráficos

Al iniciar el proceso el usuario debe cargar los recursos gráficos adicionales descritos en el archivo Picture. Cuando se carga el grupo de imágenes estas quedan almacenadas en un directorio especifico del servidor. Al hacer la validación entre el

archivo Picture y el metamodelo se valida que en servidor existan todos los recursos descritos en el Picture. En el caso de que exista algún error o algún archivo falte se revierte el proceso y se notifica al usuario del error.

3.4.4. Generación del editor

En la etapa de validación del metamodelo, la representación grafica y los recursos adicionales se completa la estructura del metamodelo intermedio con la información extraída del modelo del lenguaje.

3.4.4.1. Generación de elementos gráficos

A partir de cada uno de los elementos del metamodelo intermedio es posible generar código JavaScript para dibujar los elementos en la paleta del editor y dibujar los elementos sobre el canvas.

3.4.4.2. Generación de reglas estructurales

Joint JS permite especificar las relaciones que pueden existir entre los elementos dibujados en el canvas. Así mismo puede restringir conexión entre dos elementos del canvas si no se permite. Estas reglas se especifican a partir de la estructura del metamodelo, para esto es necesario tomar la estructura del modelo específicamente desde las relaciones entre sus elementos y traducir dicha relación en una regla de conexión en JavaScript.

Una vez el metamodelo intermedio esta completo es posible generar el código JavaScript para cada uno de los elementos gráficos y las reglas estructurales que debe seguir el editor. Este código se guarda en el servidor y se envía el código del editor generado al navegador del cliente por medio de HTTP.

4. Validaciones disponibles

En los documentos publicados por el grupo TICSw [20] se describen una serie de problemas ontológicos y lingüísticos encontrados a partir de las validaciones generadas desde la aproximación de separar el metamodelo de su representación grafica. A continuación en las *Tabla 2 y Tabla 3* se presentan los problemas encontrados y las soluciones propuestas desde el enfoque de la herramienta.

Linguistic rules validations				
Rule	**Linguistic problem detected in the model**	**Supported**	**Picture language solution**	**WebPicture solution**
MC-LR-1	The value of the attribute MetamodelURI must be provided with the URI of the domain metamodel	Yes	Metamodel URI provided in the picture language file	Provided by picture language parser
EC-LR-1	The Element instance does not have any value in the attribute typeName	Yes	Element type name provided in the picture language file	Provided by picture language parser
EC-LR-2	The value of the attribute typeName in one Element instance has blanks	Yes	Restricted by language structure	Provided by picture language parser
AC-LR-1	The Attribute instance does not have any value in the attribute typeName	Yes	Restricted by language structure	Provided by picture language parser
AC-LR-2	The value of the attribute typeName in one Attribute instance has blanks	Yes	Restricted by language structure	Provided by picture language parser
AC-LR-3	The value of the attribute typeName in one Attribute instance starts or ends with comma	Yes	Restricted by language structure	Provided by picture language parser
AC-LR-5	The value of the attribute typeName in one Attribute instance does not correspond with the type specified in the attribute type	Yes	No automatic support	Provided by initial ecore and picture validation
CRC-LR-1	The ContaintmentRelation instance does not have any value in the attribute typeName	Yes	No automatic support	Provided by initial ecore validation
CRC-LR-2	The value of the attribute typeName in one ContaintmentRelation instance has blanks	Yes	No automatic support	Provided by initial ecore validation
CRC-LR-3	Two ContaintmentRelation instances associate two Element instances with opposite direction	Yes	No automatic support	Provided by initial ecore validation
CrRC-LR-1	The CrossRelation instance does not have any value in the attribute typeName	Yes	No automatic support	Provided by initial ecore validation
CrRC-LR-2	The value of the attribute typeName in one CrossRelation instance has blanks	Yes	No automatic support	Provided by initial ecore validation

Tabla 2: Validaciones lingüísticas disponibles

Ontological rules validations				
Rule	**Ontological problem detected in the model**	**Supported**	**Picture language solution**	**WebPicture solution**
MC-OR-1	The domain metamodel has several root EClasses	Yes	Restricted by language structure	Provided by picture language parser
MC-OR-2	The value of the attribute MetamodelURI is different from the URI of the current domain metamode	Yes	No automatic support	Provided by initial ecore and picture validation
MC-OR-3	The model does not have any instance conforms to the root EClass in the domain metamodel	Yes	No automatic support	Provided in the intermediate metamodel validation
EC-OR-1	The value of the attribute typeName in one Element instance does not match with any EClass name in the domain metamodel	Yes	No automatic support	Provided by initial ecore and picture validation
EC-OR-2	The Element instance has several owner instance	Yes	No automatic support	Not used
EC-OR-3	There are several Element instances that match with the root EClass in the domain model	Yes	No automatic support	Provided by initial ecore and picture validation
EC-OR-4	The Element instance does not have associated the Attribute instances required	Yes	No automatic support	Not used
EC-OR-5	The Element instance does not have associated the ContaintmentRelation instances required	Yes	No automatic support	Provided in the intermediate metamodel construction
EC-OR-6	The Element instance does not have associated the CrossRelation instances required	Yes	No automatic support	Not used
EC-OR-7	The Element instance does not have an owner instance	Yes	No automatic support	Not used
AC-OR-1	The value of the attribute typeName in one Attribute instance does not match with any EAttribute name of the correspondent EClass in the domain metamodel	Yes	No automatic support	The editor generator does not review attributes
AC-OR-2	The value of the attribute type in one Attribute instance does not match with the EType in the correspondent EAttribute in the domain metamodel	Yes	No automatic support	The editor generator does not review attributes
AC-OR-3	The value of the attribute typeName in one Attribute instance is not defined, but the correspondent EAttribute in the domain metamodel has a default value	Yes	No automatic support	The editor generator does not review attributes
AC-OR-4	The quantity of values specified in the attribute value for one Attribute instance is lower than the lower bound in the domain metamodel or greater than the upper bound in the domain metamodel	No	No automatic support	The generated editor can not validate cardinality
AC-OR-5	There are several Attribute instances with the same value in the attribute typeName that belongs to the same Element instance	Yes	No automatic support	The editor generator does not review attributes
CRC-OR-1	The value of the attribute typeName in one ContaintmentRelation instance does not match with any EReference name of the correspondent EClass in the domain metamodel	Yes	No automatic support	Provided by initial ecore and picture validation
CRC-OR-2	The quantity of ContaintmentRelation instances belong to one Element in- stance is lower than the lower bound in the domain metamodel or greater than the upper bound in the domain metamodel	No	No automatic support	The generated editor can not validate cardinality
CrRC-OR-1	The value of the attribute typeName in one CrossRelation instance does not match with any EReference name of the correspondent EClass in the domain metamodel	Yes	No automatic support	Provided by initial ecore and picture validation
CrRC-OR-2	The quantity of CrossRelation instances belong to one Element instance is lower than the lower bound in the domain metamodel or greater than the upper bound in the domain metamodel	No	No automatic support	The generated editor can not validate cardinality

Tabla 3: Validaciones ontológicas disponibles

5. Detalles de la herramienta

En esta sección se describen los requerimientos soportados por la herramienta y las limitaciones encontradas.

5.1. Requerimientos soportados

A continuación se presentan los requerimientos funcionales de la herramienta.

5.1.1. Requerimientos funcionales

- Cargar un metamodelo.
- Cargar un metamodelo.
- Validar el contenido de un archivo Picture.
- Validar el contenido de un archivo Ecore.
- Generar un editor web a partir de un archivo Ecore y Picture.
- Persistir información de los editores y diagramas.
- Reutilizar editores web ya creados.

5.1.2. Requerimientos no funcionales

- Flexibilidad:
  La herramienta es flexible y puede integrarse fácilmente a otros servicios.
- Facilidad de modificación:
  La herramienta puede ser fácilmente modificada y su funcionalidad puede ser extendida fácilmente.
- Persistencia en base de datos:
  La información de los editores y diagramas creados es persistida en base de datos.
- Presentación web:
  La información de los editores y diagramas creados es persistida en base de datos.
- Recuperación ante fallas:
  La información de la herramienta puede ser restaurada con facilidad ya que se mantiene una base de datos con la metainformación de los editores y diagramas; y de manera independiente se mantienen los archivos físicos de los editores generados.

5.2. Limitaciones

5.2.1. Limitaciones de uso

En el proceso de desarrollo de la herramienta no se consiguió poder dibujar las relaciones según la descripción del archivo Picture por lo cual estas solo pueden dibujarse de una sola manera por defecto.

5.2.2. Limitaciones de desempeño

Se encontró que la herramienta tiene problemas de desempeño en la etapa de compilación y despliegue en servidor. Por lo cual es necesario en la configuración del servidor aumentar el tiempo de inicio. No obstante esto solo ocurre durante el despliegue en el servidor.

6. Conclusiones

Existen herramientas alternativas a WebPicture que utilizan tecnologías basadas en modelos EMF para generar editores gráficos tal como es el caso del framework de herramientas provisto por GMF. No obstante estas herramientas requieren extensas configuraciones y solo pueden ser utilizadas localmente en Eclipse. WebPicture utiliza una aproximación basada en modelos EMF en la cual se desacopla el metamodelo de la representación grafica de un modelo.

Al desacoplar el metamodelo y la representación grafica de un modelo es posible construir una herramienta más flexible, estable y fácil de utilizar. WebPicture utiliza esta aproximación por lo cual requiere de dos herramientas fundamentales para generar editores.

La primera es la especificación de un metamodelo, para esto utiliza metamodelos construidos con el framework de EMF. En este se pueden especificar metamodelos de manera rápida y estructurada.

No obstante la herramienta aun requiere un mecanismo efectivo para describir la representación grafica del modelo. Para esto se creo Picture un lenguaje estructurado, que puede ser escrito fácilmente por el usuario. Picture describe la representación grafica para los elementos y relaciones de un modelo.

A pesar que Picture no cuenta con su propio compilador las herramientas del framework de XText permiten generar rápidamente un compilador de escritorio para el lenguaje. De la misma manera es posible extraer las herramientas básicas del compilador y utilizarlas en un contexto web. De esta manera WebPicture puede validar el lenguaje y extraer información para generar un editor web.

WebPicture toma un metamodelo, la descripción de su representación grafica (expresada en lenguaje Picture) y recursos de imagen adicionales para generar un editor web de modelos. Para esto la herramienta realiza una serie de validaciones y transformaciones para generar el código de un editor. El código de un editor puede ser utilizado y reutilizado para la creación de distintos modelos ya que el código solo se genera una vez y este es conservado en el servidor.

WebPicture aprovecha las tecnologías provistas en la definición de HTML 5; y utiliza principalmente el concepto de HTML canvas y su integración con polígonos SVG. A partir de esto es posible generar un editor que no solamente es capaz de pintar elementos sobre un canvas sino también un editor capaz de hacer validaciones estructurales sobre modelos. Estas validaciones son construidas de la estructura del metamodelo y se generan a partir de una serie de transformaciones.

WebPicture es una herramienta que aprovecha las ventajas de tecnologías basado en modelos EMF. No obstante WebPicture lleva estas ventajas hacia un contexto web y consigue disminuir al mínimo el número de configuraciones para el usuario final.

La herramienta tiene potencial enorme en el marco de una suite completa de modelamiento de arquitectura empresarial desde el punto de vista de que es una herramienta liviana, fácil de utilizar y permite generar rápidamente editores ad-hoc con capacidades de validación. Finalmente podría implementarse un esquema de gobierno de artefactos arquitectura con conceptos como usuarios y roles bajo un esquema de proyectos.

7. Mejoras sugeridas

7.1. Mejoras sugeridas para el lenguaje Picture

Picture como lenguaje es una herramienta útil para estructurar la representación grafica de un modelo. No obstante durante el proceso de desarrollo se descubrieron problemas con respecto al lenguaje.

- El editor que se genera a partir de la gramática en XText es sensible a las tabulaciones, espacios y saltos de línea.
- Las comas funcionan como palabras reservadas la gramática.
- Las instrucciones no tienen un carácter para separar instrucciones.
- Las librerías que WebPicture pueden dibujar un archivo señalado por una ruta hacia un archivo SVG sino que reconoce el SVG Path.
- El lenguaje no cuenta con una documentación actualizada, la documentación disponible corresponde solo a las primeras versiones del lenguaje.
- Los Wizards incorporados no describen efectivamente las acciones que debe hacer el editor en el caso esperado e igualmente en los Wizards de modificación de atributos no se describe el atributo a modificar.

De acuerdo a los problemas encontrados se sugiere:

- Reestructurar el lenguaje, volverlo más flexible con el objetivo de hacer que el lenguaje no sea sensible a espacios y tabulaciones.
- Cambiar como se utilizan las comas.
- Incorporar un marcador de cambio de línea.
- En el caso de utilizar archivos SVG reemplazar la ruta del archivo por un SVG path.
- Actualizar la documentación del lenguaje y mantenerla actualizada ante eventuales cambios.
- Incorporar las reglas a los Wizards y hacer que las instrucciones incorporadas sean definidas por una nueva extensión del lenguaje que hable de acciones en los elementos ante eventos de usuario.

7.2. Mejoras para la siguiente versión

El producto final puede mejorar en distintos aspectos para una próxima implementación a continuación se describen las mejoras potenciales.

- Integrar un generador/editor de Picture incorporado que permita editar la representación grafica de un modelo.
- Mejorar aspectos de la interfaz grafica.
- Incorporar un esquema de gobierno de artefactos y control de usuarios en el sistema.
- Incorporar un esquema de trabajo concurrente para múltiples usuarios en un mismo editor.

8. Bibliografía

1. Naranjo D., Sánchez M., Villalobos J. PRIMROSe - A Tool for Enterprise Architecture Analysis and Diagnosis. In: 16th International Conference on Enterprise Information Systems, Lisboa, Portugal, 2014.
2. The Open group. TOGAF 9.1: Architecture content framework: Architectural artifacts, Estados Unidos, 2011.
3. Fill H., Karagiannis D. On the conceptualisation of modeling methods using the ADOxx metamodelling platform. In: Enterprise modelling Information Systems Architectures Vol 8, No 1, Viena, Austria, 2013.
4. MEGA. Enterprise architecture overview. Estados Unidos, 2014.
5. The Open group. TOGAF 9.1: Architectural artifacts by ADM phase, Estados Unidos, 2011.
6. Kolovos D., EuGENia: GMF for mortals. Nueva York, Estados Unidos, 2014.
7. Melo I., Sánchez M., Villalobos J. Composing Graphical Representations of Composed Models. In: International Workshop on The Globalization of Domain Specific Languages (GlobalDSL) at European Conference on Object-Oriented Programming (ECOOP2013). Montpellier, Francia, 2013.
8. Genova G. What is a metamodel: the OMG's metamodeling infrastructure. In: Modeling and metamodeling in model driven development. Varsovia, Polonia, 2009.
9. Gašević D., Djurić D, Devedzic V. Model driven engineering. In: Model driven engineering and ontology development. Nueva York, Estados Unidos, 2009.
10. Schmidt D. Model driven engineering. In: Computer Vanderbilt University. Nashville Tennessee, Estados Unidos, 2006.
11. Eclipse Foundation, Eclipse Modeling Framework (EMF). Ottawa, Ontario, Canadá, 2014.
12. Eclipse Foundation, Textual Modeling Framework (TMF). Ottawa, Ontario, Canadá, 2014.
13. Efftinge S, Völter M. oAW XText: A framework for textual DSLs. Alemania, 2006.
14. Melo I. EnAr-Picture, Bogotá, Colombia, 2013.
15. W3 Schools. What is HTML?. 2014.
16. Chmiel M. Javascript. *Salem Press Encyclopedia*. September 2013.
17. Client IO. Joint JS. 2014.
18. Mozilla Developer Network. SVG. 2014.
19. W3 Schools. CSS3 Introduction. 2014.
20. Gómez P., Sánchez M., Villalobos J. An approach to the co-creation of models and metamodels. In: Journal of object technology. 2014.

9. Agradecimientos

Agradezco a todos aquellos profesores de la universidad de los Andes que me enseñaron, me inspiraron y me transmitieron su pasión por lo que hacen y hoy a punto de terminar mi carrera me hacen confirmar que la decisión de escoger ingeniería de sistemas y computación fue la correcta. Pero a la persona que más debo agradecer es a mi mama, el motor de mi vida, mi inspiración, mi mejor amiga y confidente.