UNIVERSIDAD NACIONAL DE RIO CUARTO

Trabajo de tesis para la obtención del grado de Licenciatura en Ciencias de la Computación

Título .....

Autores
Jaimez Jacinto
Pereyra Orcasitas Nicolás
...@...

Director de Tesis: *Mg. Marcela Daniele*
Co-Director de Tesis: *Mg. Ariel Gonzalez*

Rio Cuarto, Argentina
mes... 2016

Resumen

El estudio de .

Palabras clave: Desarrollo Dirigido por Modelos, Web Services, QVT.

Agradecimientos

Agradezco a ...

Una dedicatoria muy especial es para ...

List of Figures

5.1 Arquitectura QVT. 8
5.2 Especificación del formato Ecore. 9

Contents

Resumen **i**

Agradecimientos **ii**

Lista de figuras **iii**

1 Introducción **1**
1.1 Contribución 1
1.2 Metas 1
1.3 Organización del Trabajo 1

2 Trabajos Relacionados **2**

3 Nociones Preliminares **3**
3.1 Servicios Web 3
3.2 Desarrollo Dirigido por Modelos - MDD 3

4 Selección de Servicios Web basada en MDD **4**

5 Implementación ... **5**
5.1 Análisis de los Lenguajes de Transformación de Modelos 5
5.2 Lenguaje Query/View/Transformation Relations 7
Características de Eclipse Modeling Framework y Ecore 7
5.3 La Transformación 9

6 Ejemplo de Aplicación **10**

7 Conclusiones y Trabajos Futuros **11**

Apéndices **13**

A Especificación de la Transformación en Query/View/Transformation Relations **14**

Chapter 1

Introducción

.....

1.1 Contribución

La principal contribución de este trabajo es ....

1.2 Metas

Puntualizando el propósito de este trabajo de investigación, se describe a continuación un conjunto de metas a alcanzar.

- Relacionar y vincular los conceptos subyacentes....
- Mostrar que ...
- Presentar un mecanismo que permita ....

1.3 Organización del Trabajo

El trabajo es organizado de la siguiente manera. En el capítulo 2, hacemos mención de los trabajos relacionados con esta investigación. En el capítulo 3, introducimos los distintos formalismos, metodologías, estándares y teoría general que dan la base a la problemática planteada, tales como .... En el capítulo **??**, presentamos el proceso de .... Posteriormente, en el capítulo 6 se exhibe un ejemplo de aplicación de la propuesta. Finalmente, en el capítulo 7 se presentan las conclusiones y trabajos futuros.

Chapter 2

Trabajos Relacionados

....

Chapter 3

Nociones Preliminares

En este capítulo se presentan los principales conceptos referentes a las áreas de base que dan soporte al trabajo, estas son: ...

En la sección **??**, hacemos una introducción a .... En la sección 3.2 se describen brevemente los principios de MDD, en particular la propuesta concreta de la OMG conocida como MDA. Y finalmente en la sección **??**, introducimos la base teórica de ...

3.1 Servicios Web

3.2 Desarrollo Dirigido por Modelos - MDD

Chapter 4

Selección de Servicios Web basada en MDD

Chapter 5

Implementación ...

...

5.1 Análisis de los Lenguajes de Transformación de Modelos

En esta sección se analizan las propuestas existentes con el fin de fundamentar la elección del lenguaje utilizado en el actual trabajo. El cuadro 5.1 es una muestra de sólo algunos de una diversidad de lenguajes de transformación que se pueden encontrar en la actualidad. Se muestra el gran número de propuestas existentes, que evidencia el gran interés que se ha despertado por este aspecto del paradigma del desarrollo de software dirigido por modelos, así como la enorme actividad de investigación surgida alrededor del mismo.

Existe una variedad muy amplia de lenguajes de transformación, donde cada uno posee características deseables al igual que inconvenientes. Los lenguajes específicos del dominio (Domain Specific Language - DSL), como MT, ofrecen la ventaja de aprovechar todo el potencial del lenguaje *host*, sumándole la expresividad y sofisticación del aparato transformacional. No obstante, esta característica también los hace poco compatible con otras herramientas que se basen en los estándares de la OMG (QVT, OCL, MOF, etc). Por otra parte, los lenguajes gráficos como UMLX son sumamente intuitivos pero a la vez limitados por la misma conformación de la simbología del lenguaje. Conjuntamente, la descripción de una transformación demasiada compleja puede resultar engorrosa para su utilización. Además, suelen incurrir en costos computacionales más elevados en comparación con lenguajes textuales, debido especialmente a que su representación es más compleja de procesar por su naturaleza gráfica. Los lenguajes de transformación M2T son bastante útiles para realizar implementaciones específicas de plataforma a partir de un modelo independiente de plataforma. Sin embargo, dejan de lado la transformación entre modelos, parte fundamental del desarrollo de software dirigido por modelos. En cualquier caso, actualmente muchas de las propuestas analizadas se encuentran en etapa de desarrollo, lo cual seguramente

Lenguaje/Implementación	**Breve Descripción**
ATL (Atlas Transformation Language)	Lenguaje de transformación de modelos y herramienta para Eclipse desarrollada por el Atlas Group (INRIA).
BOTL(Basic Object-Oriented Transformation Language)	Propuesta gráfica de la Universidad Técnica de München.
ETL (Epsilon Transformacion Language)	Lenguaje definido sobre la plataforma Epsilon, plataforma desarrollada como un conjunto de plug-ins (editores, asistentes, pantallas de configuración, etc.) sobre Eclipse. Dicha plataforma presenta el lenguaje de metamodelado independiente Epsilon Object Language que se basa en OCL, y puede ser utilizado como lenguaje de gestión de modelos o como infraestructura a extender para producir nuevos lenguajes específicos de dominio, como el citado Epsilon Transformation Language (ETL). Además de ETL, Epsilon soporta a Epsilon Comparison Language (ECL) y a Epsilon Merging Language (EML), orientados a la comparación y composición de modelos, respectivamente.
M2T (Model to Text project)	Se focaliza en la generación de artefactos textuales a partir de modelos. Consta de un conjunto de herramientas desarrolladas para la plataforma Eclipse.
MTF (Model Transformation Framework)	Es un conjunto de herramientas desarrollado por IBM que permiten hacer comparaciones, comprobar la consistencia y ejecutar transformaciones entre modelos EMF.
MTL (Model Transformation Language)	Lenguaje de transformación de modelos y herramienta en Eclipse desarrollada por el grupo Triskell.
QVT (Query/View/Transformation)	Especificación estándar de OMG. Está basado en MOF (Meta Object Facility) para lenguajes de transformación en MDA. Implementaciones: QVT−Operational: QVT Operacional de Borland SmartQVT Eclipse M2M QVT-Core: OptimalJ (Descontinuado) QVT-Relations: MediniQVT Eclipse M2M Declarative QVT ModelMorf
RubyTL	Lenguaje de transformación híbrido definido como un lenguaje específico del dominio embebido en el lenguaje de programación Ruby. Diseñado como un lenguaje extensible en el que un mecanismo de plugins permite añadir nuevas características al núcleo de características básicas.

Table 5.1: Lenguajes de transformación de modelos.

permitirá observar, en el futuro, mejoras en cada una de ellas.

QVT y los basados en éste, como por ejemplo ATL, tienen la ventaja de ceñirse a un estándar, lo cual los hace mas compatibles con otras tecnologías y comprensibles dada la formalidad de su especificación. Además, múltiples herramientas se han desarrollado que dan soporte a su implementación, entre ellas MEDINIQVT, la cual es una de las más robustas en la actualidad. Bajo las consideraciones mencionadas y otras comparativas efectuadas el lenguaje seleccionado para el actual trabajo es QVT.

5.2 Lenguaje Query/View/Transformation Relations

La OMG ha definido el estándar Query/View/Transformation (QVT) [**?**] para utilizar en los procesos de desarrollo de software guiados por modelos. QVT consta de tres partes: consulta, vista y transformaciones. El componente de consultas de QVT toma como entrada un modelo y selecciona elementos específicos del mismo. Para la resolución de las consultas y restricciones sobre los modelos, se propone el uso de una versión extendida de OCL 2.0 [**?**]. La parte de vista es una proyección realizada sobre un modelo y es creada mediante una transformación. Una vista puede verse como el resultado de una consulta sobre un modelo, ofreciendo como resultado un punto de vista de éste, restringiéndolo de acuerdo a alguna condición impuesta. En esta sección se presenta el componente de transformaciones de QVT que tiene como objetivo definir transformaciones. Estas transformaciones describen relaciones entre un metamodelo fuente F y un metamodelo objetivo O. Ambos metamodelos deben estar especificados con el estándar Meta-Object Facility (MOF) [1]. Una vez definida la transformación en QVT, ésta puede ejecutarse para obtener un modelo objetivo, que es una instancia del metamodelo O a partir de un modelo fuente, el cual es una instancia del metamodelo F.

La especificación de QVT 1.0 [**?**] tiene una naturaleza híbrida declarativa/imperativa, donde la parte declarativa se divide en una arquitectura de dos niveles (ver figura 5.1) constituida por un metamodelo y un lenguaje llamado *Relations* que sigue un paradigma declarativo, el cual soporta concordancia de patrones (Pattern Matching) de objetos complejos y plantillas de creación de objetos, y un metamodelo y un lenguaje llamado *Core*, definido usando mínimas extensiones de MOF y OCL. En este trabajo se hace uso del lenguaje *Relations* que permite especificar relaciones entre metamodelos MOF y de la herramienta para la definición de las transformaciones MediniQVT [2]. Esta herramienta implementa la especificación QVT/Relations en un poderoso motor QVT. Está diseñada para transformaciones de modelos permitiendo un rápido desarrollo, mantenimiento y particularización de reglas de transformación de procesos específicos. Su interfaz está basada en Eclipse y utiliza Eclipse Modeling Framework (EMF) [3] para la representación de los modelos. Además, posee un editor con asistente de código y un depurador de relaciones. MediniQVT exige que los metamodelos (y modelos) sean escritos en una variante simplificada del estándar MOF, llamada Ecore [4], el cual ha sido definido en el EMF.

Características de Eclipse Modeling Framework y Ecore

El lenguaje Ecore es el utilizado por el Eclipse Modeling Framework (EMF). Los metamodelos y modelos usados por EMF se representan con documentos XML. Existen herramientas que tratan automáticamente estos metamodelos y modelos. Por ejemplo, EMF genera código automáticamente para editores de modelos para un lenguaje de modelado, definido con un metamodelo en Ecore. Otro ejemplo es el plug-in de Eclipse para transformar este tipo de modelos, con la definición de transformaciones QVT. Los metamodelos de Ecore son XML, sin embargo EMF posee un editor gráfico de modelos Ecore que facilita su creación.

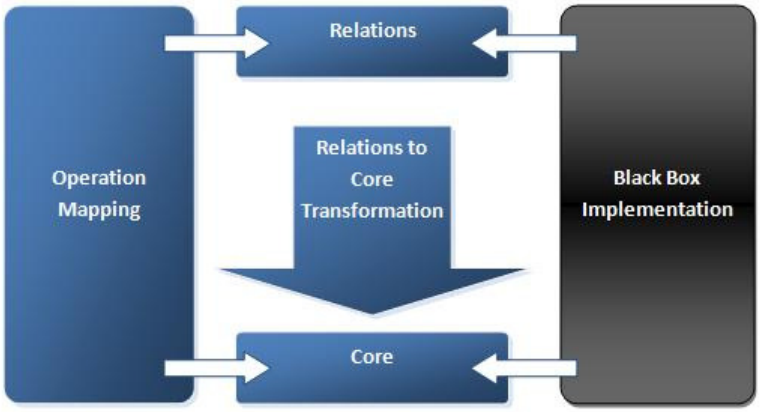

Figure 5.1: Arquitectura QVT.

Las características y elementos del lenguaje de modelado Ecore son los siguientes: el elemento aglutinador (raíz) es el paquete `EPackage`, que contiene físicamente a sus elementos (especificación de `containment`) y, a su vez, éstos pueden estar contenidos en otros paquetes. Hay una fábrica (`EFactory`) por paquete que permite la creación de los elementos del modelo. Las construcciones que describen a un conjunto de elementos (instancias) son clasificadores (`EClassifiers`): `EClass` y `EDataType`.

Ecore especifica las características de las clases (`EClass`), sus características estructurales (`EStructuralFeatures`), atributos (`EAttributes`), operaciones y relaciones (herencia, referencia (`EReference`)). Las `EClasses` tienen superclases y están compuestas por características estructurales (`EStructuralFeatures`: `EReference` y `EAttribute`). Tanto las `EReferences` como los `EAttributes` pueden estar dotados de multiplicidad. Los `EDataTypes` modelan tipos básicos o indivisibles del modelo de datos, y los `EReferences` pueden estar contenidos o ser referencias (punteros). Las `Operations` modelan operaciones de la interfaz (aunque no se provee implementación para ellas). Todos los elementos heredan de `ENamedElement` (tienen nombre), y de `EModelElement` (elemento del modelo). Además, todo elemento del modelo puede tener asociadas anotaciones (`EAnnotation`): pares nombre/valor para especificaciones extra (por ejemplo, restricciones OCL o cadenas de documentación). La figura 5.2, muestra la especificación del estándar Ecore.

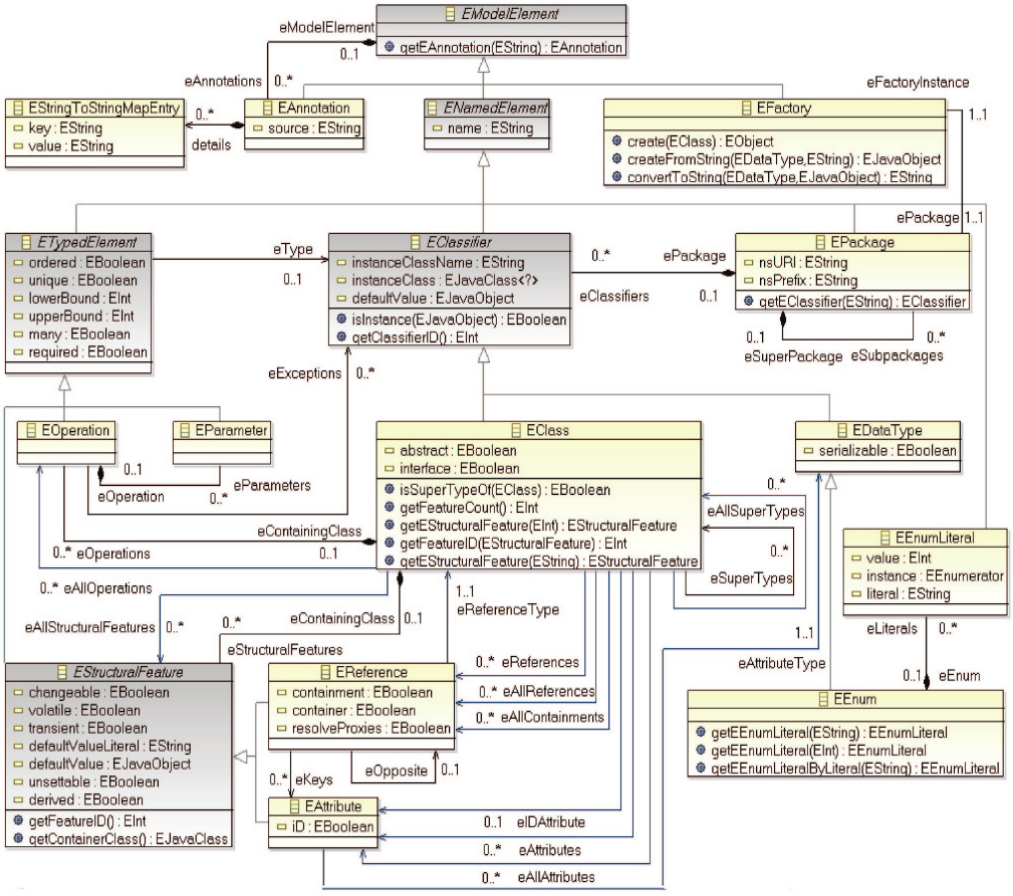

Figure 5.2: Especificación del formato Ecore.

5.3 La Transformación

... hablar de la definiciones de las relaciones. Mostrar las partes de codigo mas relevantes de las relaciones.

Chapter 6

Ejemplo de Aplicación

Chapter 7

Conclusiones y Trabajos Futuros

....

Bibliography

[1] *OMG Meta Object Facility (MOF) Core Specification, Version 2.4.1*, Object Management Group Std., Rev. 2.4.1, 2011. [Online]. Available: http://www.omg.org/spec/MOF/2.4.1

[2] *Medini QVT*, ikv++ Technologies, Last access: May 2015. [Online]. Available: http://www.ikv.de/

[3] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *EMF: Eclipse Modeling Framework 2.0*, 2nd ed. Addison-Wesley Professional, 2009.

[4] *Ecore metamodel specification*, Object Management Group, Last access: May 2015. [Online]. Available: http://download.eclipse.org/modeling/emf/emf/javadoc/2.9.0/org/eclipse/emf/ecore/package-summary.html

Apéndices

Appendix A

Especificación de la Transformación en Query/View/Transformation Relations

Código completo, en QVT Relation, de la transformación de máquinas de estados con elementos opcionales a máquinas de estados concretas.

```
-- Inicio de la Transformación
```