Evaluación de Lenguajes de Transformación para el Desarrollo de Software Dirigido por Modelos

Juan C. Herrera[1-2], Francis Losavio[2], Alfredo Matteo[2]

[1]Universidad Bolivariana de Venezuela
Programa de Formación de Grado Informática para la Gestión Social
Los Chaguaramos. Caracas, Venezuela
jchr1982@yahoo.com
[2]Universidad Central de Venezuela
Facultad de Ciencias - Escuela de Computación - Laboratorio MoST
Los Chaguaramos. Caracas, Venezuela
{flosav,almatteo}@cantv.net

Abstract. In the approach of model driven software development the use model transformation language plays an essential role. So, as the OMG (Object Management Group) through the specification MOF/QVT (Query / Views / Transformations) defines a set of guidelines that should be satisfied by the transformation language used. In recent years many languages have been defined and many works on the evaluation and/or comparison of these languages have been published. However, in each of these comparisons are evaluated aspects under a single perspective or dimension, such as scenarios of use, technological space, patron or mechanisms of execution among others. In this work is carried out an assessment of transformation language that brings together the criteria and characteristics under a certain dimension. These dimensions have been identified from work carried out on comparisons of transformation language. The evaluation process done is based on the methodology Desmet with the approach of analysis of characteristics.

Keywords: Model driven software development, model transformation language, MOF/QVT.

1 Introducción

Es evidente que la evolución de los sistemas de software es una tarea cotidiana para los desarrolladores, debido a los continuos cambios que las organizaciones experimentan, adicionalmente, la creciente complejidad de estos y a su vez los rápidos avances tecnológicos en las plataformas que la implementan hacen difícil su evolución, que conlleva al incremento en los recursos que se destinan para ello. Como alternativa para facilitar la evolución de los sistemas de software surge el desarrollo de software dirigido por modelos (DSDM). Hecho evidenciado en Pérez et al. [24] al indicar que a pesar del incremento en el nivel de abstracción en el área del desarrollo de software en los últimos años, aún quedan problemas por resolver, tales como la creciente complejidad de los sistemas y el surgimiento de nuevas plataformas. En consecuencia el DSDM surge como paradigma que combina los siguientes elementos:

lenguajes específicos de dominio (DSL) que formalizan la estructura de la aplicación, el comportamiento y los requisitos en un dominio particular; descritos mediante metamodelos; y motores de transformación que analizan los modelos y generan el código fuente u otros modelos alternativos, acorde a reglas de transformación, permitiendo separar el conocimiento del dominio del negocio (modelo conceptual) de la plataforma tecnológica (modelo de implementación), así se reducirán los esfuerzos a la hora de evolucionar los sistemas.

El éxito del DSDM requiere de adecuados lenguajes de modelado y entre ellos lenguajes de transformación de modelos (LTM) para escribir las reglas de transformación. En tal sentido Mens, Czarnecki, Van Gorp [20] han identificado requisitos de calidad que deben poseer estos lenguajes, tales como usabilidad, escalabilidad y conforme a estándares. Además, marcos teóricos como el modelo de características discutido en [4] y la taxonomía de transformaciones de modelos presentada en [20] en alguna medida permiten comparar si la técnica de transformación utilizada por el LTM se acopla al propuesto por el OMG en la guía MDA [22], u otros enfoques utilizados en la práctica [5].

Muchos LTM son utilizados para especificar las definiciones de transformación, en tal sentido, Kurtev [17] indica que aunque un lenguaje de propósito general (GPL) puede ser utilizado para escribir un programa que transforme un modelo en otro, la tendencia en la comunidad DSDM es la de utilizar DSL para escribir las definiciones de transformación. Esto se evidencia por el surgimiento de varias propuestas en respuesta a la solicitud realizada por el OMG [21] para la definición de un LTM bajo el estándar denominado QVT**,** entre ellas las más significativas están la del grupo QVT Merge [25] **y** QVT Partners [26]. Este hecho es confirmado por Czarnecki [7] al indicar que las transformaciones de modelos requieren de DSL para su definición.

Por consiguiente, el establecer criterios adecuados para evaluar los LTM en el dominio del DSDM para la definición de transformaciones es prioritario para la comunidad de desarrolladores, por lo cual esta investigación identifica, selecciona, unifica, agrupa y reubica las características más relevantes a través de la definición de siete dimensiones vinculadas con la formalización, organización, patrón de ejecución, escenarios de uso, espacio tecnológico, aspectos no funcionales y herramienta. Esto es justificado por Van Gorp y Mens [29] al indicar que es conveniente establecer un conjunto de criterios concretos y adecuados que caractericen los LTM que ayude al desarrollador en la elección de un LTM, acorde a necesidades específicas.

Para darle formalidad, la metodología DESMET [16] es utilizada como mecanismo para evaluar los LTM en el dominio de la Ingeniería de Modelos y con ello esbozar las fortalezas y debilidades, así como los alcances y/o limitaciones.

La investigación se ha estructurado en tres secciones; la sección 1 trata la presente introducción. La sección 2 establece la metodología de evaluación y esboza los resultados de la evaluación. La sección 3 presenta el análisis de los resultados con las conclusiones y por último se listan las referencias bibliográficas.

2 Metodología de Evaluación

En esta investigación, se sigue el enfoque sistémico, propuesto por Kitchenham [16] denominado DESMET, esta metodología ayuda a un evaluador en una organización a

planificar y ejecutar un ejercicio de evaluación que es imparcial y confiable con el objetivo de identificar el método de evaluación más adecuado que se adapte a un contexto especifico. La metodología identifica varios métodos de evaluación y los caracteriza estableciendo criterios técnicos y prácticos para seleccionar el método más adecuado. Para la evaluación de los LTM, DESMET propone un método de evaluación denominado análisis de características por proyección; que consiste en una evaluación cualitativa y estructurada que permite caracterizar los LTM para poder compararlos, y con ello facilitar la toma decisión a la hora de elegir el lenguaje que se adecue a las necesidades del desarrollador. Así mismo para organizar la evaluación se utilizó el enfoque de filtrado o selección que se caracteriza por ser realizada por una sola persona, responsable de todo el proceso, donde la información obtenida está basada principalmente en la documentación y se utiliza frecuentemente como una primera etapa en un ejercicio de evaluación más complejo que implica la reducción a uno o dos posibles candidatos que pueden ser evaluados posteriormente con más detalle. A continuación se describe en detalle el método utilizado en este trabajo.

2.1 Método de Análisis de Características por Proyección

La implementación del método fue organizada en seis pasos, su especificación se indica a continuación, según Kitchenham [16]:

2.1.1 Seleccionar un conjunto de lenguajes candidatos para evaluar.

De acuerdo al estudio de los trabajos recopilados en la investigación, se han identificado veintisiete (27) LTM. Debido a la gran cantidad de lenguajes, se hace necesario elegir cuales son los más relevantes a fin de reducir el conjunto de lenguajes a evaluar. Para la selección de los LTM se utilizaron los siguientes criterios: el primer criterio fue ubicar los LTM según la técnica de implementación utilizada en el contexto del DSDM descrito por Czarnecki y Helsen [5]:

- Manipulación directa: permite la representación interna del modelo y APIs para manipularlo;
- Dirigido por la estructura: crea la estructura jerárquica del modelo, luego establece los atributos y referencias en él;
- Operacional: extiende el formalismo de un metamodelo mediante librerías/frameworks para expresar cálculos/operaciones;
- Relacional: establece relaciones entre elementos origen y destino utilizando restricciones;
- Transformación de grafos: representa gráficamente elementos del modelo facilitando la especificación y comprensión de las transformaciones;
- Híbrido: combina técnicas anteriores;
- Meta-programación: utiliza un DSL embebido en un lenguaje procedimental.

El segundo criterio fue seleccionar al menos uno y máximo dos LTM representativos por cada técnica en base al crecimiento, aceptación que han tenido en la comunidad DSDM, la cual es reflejada en función de la cantidad trabajos

publicados y la utilización de estos en la solución de problemas a través de herramientas que soportan a los lenguajes.

Finalmente, en la tabla 1 se muestran los resultados del proceso de selección, indicando en la columna 2, cuales son los diez (10) LTM que serán evaluados.

Tabla 1. LTM a ser evaluados agrupados según la técnica de implementación utilizada.

Técnica de implementación		LTM a ser evaluados	
1	Manipulación Directa	1	JMI [13]
2	Dirigido por la Estructura	2	OptimalJ [9]
3	Operacional	3	KERMETA [30]
		8	SmartQVT [8]
4	Relacional	4	KMTL [2]
		7	MOMENT [27]
5	Transformación de Grafos	5	GReAT [1]
6	Hibrido	6	ATL [3]
7	Meta-Programación	9	RubyTL [6]
8	XSLT	10	UMT [12]

2.1.2 Definir las propiedades o características para la evaluación del conjunto de LTM seleccionados.

Una vez identificado los LTM a evaluar, se definen las características que se utilizan para realizar la evaluación. Para ello se realizan las siguientes actividades:

- Recolección de información: Esta actividad involucra el estudio de las características relevantes en los LTM obtenidas de la literatura relacionada con las técnicas y lenguajes utilizados para la transformación de modelos, incluyendo especificaciones formales como QVT [23], para la definición de los LTM. Para la selección de los criterios o propiedades de evaluación que conforman la propuesta se realizó un compilado de los criterios utilizados en las investigaciones que sirvieron de antecedentes en este trabajo: [1], [3], [9], [10], [12], [15], [18], [19], [20], [23], [27], [28], [29].

- Identificación de dimensiones: Al examinar diversos trabajos donde existen comparaciones, análisis de características y propuestas de requisitos funcionales y no funcionales en los LTM, se han identificado que existen diferentes ámbitos o dimensiones en las cuales se sitúan estos trabajos. Así por ejemplo existen trabajos tales como [10], [23] en los cuales el análisis realizado se fundamenta en lo relativo a como los lenguajes son definidos; también existen otros trabajos como [15], [28] que solo consideran como están organizados estos lenguajes; Finalmente se han identificado siete dimensiones vinculadas con la formalización, organización, ejecución, escenarios de uso, espacio tecnológico, aspectos no funcionales del lenguaje y herramienta; cada una representando una macro categoría para la evaluación de los LTM. En la figura 1 se muestra el modelo conceptual de un LTM que indica que un lenguaje es implementado a través de una o varias técnicas y que tiene asociado unas características que

poseen atributos que sirven como indicadores para evaluar el lenguaje y que cada característica tiene dos propiedades, la primera, indica la dimensión a la que pertenece la característica; la segunda, indica si la característica es obligatoria.

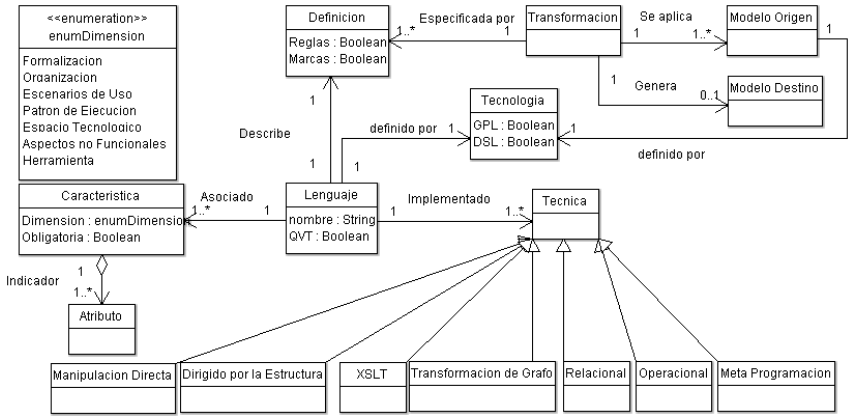

Fig. 1. Modelo conceptual sobre un LTM

A continuación se describe el significado de cada una de las dimensiones:

- **Formalización del Lenguaje:** Agrupa las características y/o requisitos que un LTM debe satisfacer para proveer un soporte formal para la especificación y/o descripción de las transformaciones de modelo.
- **Organización del Lenguaje:** Agrupa las formas o maneras en la cual los constructos, sentencias, reglas del lenguaje pueden ser organizadas.
- **Patrones de Ejecución:** Agrupa los mecanismos que dispone un LTM para que pueda ofrecer sus funcionalidades.
- **Escenarios de Uso:** Agrupa los escenarios en que pueden aplicarse los LTM. Desde el punto de vista de lo que el lenguaje puede hacer.
- **Espacio Tecnológico:** Dimensión que utiliza una abstracción del concepto de espacio tecnológico definido en [14] como un contexto de trabajo con un conjunto de conceptos relacionados, es decir, agrupa el conjunto de tecnologías basadas en estándares y/o plataformas asociadas a los LTM.
- **Aspectos no Funcionales:** Agrupa las características de calidad deseables en los LTM desde el punto de vista de los usuarios finales.
- **Herramienta:** Agrupa las características de las herramientas que implementan los LTM y que son de gran importancia al momento de elegir un lenguaje.

- Distribución de características: Inicialmente, para cada uno de los trabajos examinados se sitúan las características identificadas por los autores con respecto a las dimensiones previamente definidas. Posteriormente, se procede a la aplicación de un proceso cognitivo implementado a través del uso de referencias cruzadas entre las características en su correspondiente dimensión; donde los siguientes criterios fueron utilizados:

- Unificación: unifica características cuya semánticas son consideradas iguales, incluye agrupar y unificar atributos;
- Conversión: transforma características en atributos y las agrupa dentro de una característica nueva o existente.
- Incorporación: anexa nuevas características, producto de la investigación;
- Reubicación: existen trabajos que abarcan diferentes dimensiones, por lo cual algunas características deben ser reubicadas.

Finalmente, se generó una única tabla, para cada una de las siete (7) dimensiones. Sin embargo, por limitaciones de espacio, solo se muestran dos de ellas; ver tabla 2 y 3.

Table 2. Características para la evaluación de los LTM en la dimensión contexto de uso.

Características	Atributos	Descripción
Traducción*	Migración	Un modelo es transformado en otro en el mismo nivel de la abstracción.
	Síntesis	Un modelo se transforma en otro a un nivel inferior de abstracción.
	Ingeniería inversa	Un modelo es transformado en otro a un nivel superior de abstracción.
	Endógeno	Todos los modelos son conforme al mismo metamodelo.
	Exógeno	Múltiples metamodelos son utilizados.
	Reemplaza	Modelo de entrada sí se convierte en modelo de salida.
	Crea	Modelo de entrada: solo lectura. Modelo de salida: se genera.
Interpretación	Normalización	Un modelo es transformada por reducirlo a un sub lenguaje.
	Refactorización	Un modelo es reestructurado, mejorando su diseño.
	Corrección	Un modelo es modificado a fin de fijar un error.
	Adaptación	Un modelo es modificado a fin de adaptarse a nuevos requerimientos.
Sincronización*	Parcial	Solo aplica a determinadas partes o elementos de un modelo.
	Comprobación	Verifica que los modelos estén relacionados de una forma determinada.
	Doble sentido	Genera el modelo a partir del código fuente (LTM) y viceversa.
	Vistas	Un modelo completamente derivado de otro modelo. Un solo sentido.
Orden superior		Permite aplicar una transformación a otra transformación (modelo).

Table 3. Características para la evaluación de los LTM en la dimensión organización.

Características	Atributos	Descripción
Modularidad a nivel de reglas*	Composición	Permite el desarrollo de transformaciones mediante módulos.
	Combinarse	Permite combinar módulos entre sí para producir otros módulos.
Modularidad a nivel de transformación	Encadenarse	Permite encadenar varias transformaciones consecutivamente.
	Composición	Permite componer transformaciones para formar una nueva.
	Combinarse	Permite combinar varias transformaciones.
Definición de Condiciones de ejecución*	Precondición	La aplicación de las reglas es condicionada a algún contexto.
	Determinístico	Un algoritmo dirige el orden de ejecución de las reglas.
	Iterativo	Provee estructuras de iteración de reglas, o mecanismos de recursión.
	No deterministico	No hay orden de ejecución predefinido entre reglas.
	Interactivo	Permite al usuario especificar la estrategia de ejecución de la regla.
Mecanismos de integración y reuso	Librerías	Importando librerías de transformación.
	Parametrizadas	Permite especializar transformaciones utilizando parámetros.
	Deshabilitar	Permite restringir el conjunto de objetos al cual aplica una regla.
	Reemplazo	Permite la sobreescritura de reglas.
	Transformación	Permite reutilizar partes grandes de una transformación.
	Reglas	Permite construir nuevas reglas a partir de reglas existentes.

Correspondencia de elementos*	Colección	Permite explícitamente iterar sobre los elementos de una colección.
	Simple	Expresado en términos de un simple elemento.
Definición de elementos	Por elementos	Múltiples elementos destino pueden ser definidos en una sola regla.
	Por reglas	Múltiples reglas proveen valores de propiedad al mismo objeto destino.
Transformaciones Intermedias		Permite la definición de transformaciones intermedias, apoyando así las transformaciones multipaso.

2.1.3 Priorizar las características por cada dimensión tomando en cuenta las necesidades o requisitos del usuario.

Cada característica está acompañada por un grado de estimación de importancia. Un buen LTM es aquel que incluye las características más importantes para los usuarios. Su importancia puede ser estimada al considerar si esta es obligatoria o deseable. Para valorar una característica la siguiente escala ordinal es considerada: Obligatorio, Deseable; la asignación de la prioridad está estrechamente vinculada con los requisitos obligatorios y opcionales, según las propiedades deseables establecidas en los trabajos analizados. Las características obligatorias se identifican con un *.

2.1.4 Convenir sobre un sistema de puntuación que puede aplicarse a todas las características.

La evaluación utiliza la escala nominal que permite evaluar características simples, donde se detecta la presencia o ausencia de la característica en el LTM evaluado. La presencia es identificada por el signo (+), la ausencia es identificada por no tener signo. Cada característica tiene sus atributos de medición que actúan como indicadores que determinan si el lenguaje presenta la característica. El valor final será obtenido tomando en cuenta los siguientes criterios: Primero, por la sumatoria de todas las características que satisfaga el lenguaje; segundo, por la sumatoria de todos los atributos presentes; tercero, se verifica que todas las características obligatorias estén presentes en el lenguaje. En consecuencia, el LTM con mayor puntuación es aquel que posee el mayor número de características, si hay dos lenguajes con igual cantidad de características satisfechas, se tomará en consideración el número de atributos presentes, indicando que la característica está mejor implementada.

Un LTM que satisfaga todas las características obligatorias, entonces satisface la dimensión que las agrupa. Por otro lado, es inaceptable un LTM que no posea características obligatorias.

2.1.5 Llevar a cabo la evaluación para determinar que tan bien los LTM satisfacen los criterios previamente establecidos.

La evaluación es llevada a cabo al analizar los ejemplos propuestos en cada uno de los trabajos identificados que describen las características y funcionalidad del lenguaje además de esbozar las fortalezas y debilidades presentes, contrastándolas con las características identificadas y agrupadas según la dimensión a la cual pertenecen, ver tablas 2, 3.

2.1.6 Analizar e interpretar los resultados para su presentación.

Los resultados de la evaluación son presentados en forma de tabla de dos ejes, ver tablas 4, 5; en el eje vertical están las características (C) y sus atributos (A) por dimensión, ver tabla 2, 3 para una descripción detallada; y en el eje horizontal están los 10 LTM evaluados (L), ver tabla 1. Esto permitirá al evaluador comparar los LTM desde la perspectiva de la dimensión de su interés. El observar los totales, mostrará el nivel de satisfacción de cada LTM acorde a la dimensión analizada. Ejemplo, el valor 2/6 del lenguaje 2, indica que el LTM satisface dos características y seis atributos.

Table 4. Resultados de la evaluación de los LTM para la dimensión escenarios de uso.

C	A	L									
		1	2	3	4	5	6	7	8	9	10
1*	1						+		+		+
	2		+								+
	3		+								+
	4	+		+	+	+	+	+	+	+	
	5					+	+	+	+		
	6						+				
	7		+		+	+	+				
2	1										
	2			+			+				
	3										
	4							+	+		
3*	1	+	+	+	+	+	+	+	+	+	+
	2		+					+	+		
	3										
	4		+					+			
4	1										
Total		2/2	2/6	3/3	2/3	2/4	3/7	3/6	3/6	2/2	2/4

C = característica, A = atributo/indicador, L = lenguaje, + = satisface, * = requisito obligatorio

Table 5. Resultados de la evaluación de los LTM para la dimensión organización.

C	A	L									
		1	2	3	4	5	6	7	8	9	10
1*	1	+	+	+	+	+		+	+	+	+
	2									+	
2	1										
	2			+				+		+	
	3										
3*	1	+		+	+	+		+			
	2	+	+	+	+	+	+	+	+	+	+
	3					+					
	4					+	+				
	5										
4	1	+		+			+			+	
	2		+								
	3										

	4			+						+	
	5			+							
	6			+			+		+		
5*	1	+		+	+			+	+	+	
	2	+	+	+	+	+	+	+	+	+	+
6	1					+				+	
	2	+								+	
7	1			+						+	
Total		5/7	4/4	6/11	3/5	4/7	3/5	4/6	4/5	7/11	3/3

C = característica, A = atributo/indicador, L = lenguaje, + = satisface, * = requisito obligatorio

3 Conclusiones

Al analizar las tablas se observa que MOMENT(7) y SmartQVT(8) satisfacen en gran medida la dimensión “escenarios de uso” con un total de 3 puntos y una distribución homogénea de atributos satisfechos, lo cual indica su versatilidad; seguido muy de cerca por ATL(6); puede observarse que todos los LTM poseen todas las características obligatorias; sin embargo se evidencia que en algunos lenguajes estas características están mejor implementadas; como es el caso de la característica 1 “Traducción” donde ATL es la que tiene la mayoría de las funcionalidades, es decir, de 7 atributos posee 5. Por otra parte RubyTL(9) tiene la mayor puntuación en la dimensión “organización”, indicando su gran flexibilidad, seguido de cerca por Kermeta(3). Lo importante es que las dimensiones permiten visualizar de forma rápida las fortalezas y debilidades en cada LTM, facilitando la toma de decisión en la elección del LTM acorde a las necesidades del desarrollador.

Referencias

1. Agrawal, A. GReAT: A Metamodel Based Model Transformation Language, Institute for Software Integrated Systems (ISIS), Vanderbilt University. (2003).
2. Akehurst, D. H., Howells, W. G., McDonald-Maier, K. D., Kent Model Transformation Language, University of Kent at Canterbury, UK (2005).
3. Bézivin, J., Dupé, G., Jouault, F., Pitette, G., Rougui, J. E., First experiments with the ATL model transformation language: Transforming XSLT into Xquery, Atlas Group, INRIA and IRIN University of Nantes (2003).
4. Czarnecki, K., Helsen, S. Classification of model transformation approaches. In Proceedings of the 2nd OOPSLA Workshop on Generative Technique in the Context of the Model Driven Architecture (2003).
5. Czarnecki, K., Helsen, S. Feature-based survey of model transformation approaches. IBM Systems Journal, Vol. 45, No 3 (2006).
6. Cuadrado, J. S., Molina, J. J. G., Tortosa, M. M., Rubytl: Un Lenguaje de Transformación de Modelos Extensible, XV JISBD 2006, Barcelona (2006).
7. Czarnecki K., Overview of Generative Software Development (2004).
8. Dupe, G., Belaunde, M., SmartQVT, France Telecom, http://smartqvt.elibel.tm.fr/ (2007).
9. García, J. M., Rodríguez, J., Menárguez, M., Ortín, M.J., Sánchez, J., Un estudio

comparativo de dos herramientas MDA OptimalJ y ArcStyler, Departamento de Informática y Sistemas, Universidad de Murcia (2004).
10. Gardner, T., Griffin, C., Koehler, J., Hauser, R., A review of OMG MOF 2.0 Query/Views/ Transformations Submissions and Recommendations towards the final Standard (2003).
11. Gerber, A., Lawley, M., Raymond, K., Steel, J., Wood, A., Transformation: The missing link of MDA , Graph Transformation: First International Conference, ICGT (2002), http://archive.dstc.edu.au/AU/staff/kerry-raymond/missing-link.pdf.
12. Grønmo, R., Oldevik, J., An Empirical Study of the UML Model Transformation Tool (UMT), In 1st Int. Conf. Interoperability of Enterprise Software and Applications (2005) http://interopesa05.unige.ch/INTEROP/Proceedings/Industrial/IND2_Gronmo.pdf.
13. Java™ Metadata Interface (JMI) Specification, Java Community Process, http://www.jcp.org/, Version 1.0, Final Specification (07-June-2002).
14. Kurtev, I., Bézivin, J., Aksit, M., Technological Spaces: An Initial Appraisal, In: CoopIS'02 DOA, http://www.sciences.univ-nantes.fr/lina/atl/www/papers/PositionPaperKurtev.pdf.
15. Kurtev, I., Berg, K. V., Jouault, F., Evaluation of rule-based modularization in model transformation languages illustrated with ATL, In: ACM Symposium on Applied Computing (SAC) (2006).
16. Kitchenham, B. DESMET: A method for evaluating Software Engineering methods and tools. Department of Computer Science, University of Keele, Staffordshire (1996).
17. Kurtev, I. , Adaptability of Model Transformations, Phd thesis, University of Twente. (2005) url: http://wwwhome.cs.utwente.nl/~kurtev/files/thesis.pdf.
18. Langlois, B., Farcet, N., THALES recommendations for the final OMG standard on Query / Views / Transformations, THALES Research & Technology France (2003).
19. Mens, T., Czarnecki, K., Gorp, P. V., A Taxonomy of Model Transformations (2005). http://drops.dagstuhl.de/opus/volltexte/2005/11/pdf/04101.SWM2.Paper.pdf.
20. Mens, T., Czarnecki, K., Van Gorp, P. A Taxonomy Of Model Transformations (2005).
21. Object Management Group, Request for Proposal: MOF 2.0 Query/Views/Transformations RFP (2002). http://www.omg.org/docs/ad/02-04-10.pdf.
22. Object Management Group. (2003). Model driven architecture (MDA) Guide Version 1.0.1 http://www.omg.org/docs/omg/03-06-01.pdf.
23. Object Management Group, Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, Final Adopted Specification (2007).
24. Pérez, J. M., Ruiz, F., Piattini, M. Model Driven Engineering Aplicado a Business Process Management (2007), www.esi.uclm.es:8080/tsi/informes/UCLM-TSI-002.pdf.
25. QVT Merge, Revised submission for MOF 2.0 Query/View/Transformation RFP (2005).
26. QVT Partners, Initial submission for MOF 2.0 Query/Views/Transformations RFP, http://qvtp.org/ Version 1.1 (2003/08/18).
27. Queralt, P., Hoyos, L., Boronat, A., Carsí, J. Ramos, A. I., Un Motor de Transformación de Modelos con Soporte para el Lenguaje QVT Relations, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, España (2006).
28. Sendall, S., Kozaczynski, W., Model Transformation – the Heart and Soul of Model-Driven Software Development (2003).
29. Van Gorp, P., Mens, T., and workgroup participants, A Taxonomy of Model Transformation and its Application to Graph Transformation Technology (2004) http://kathrin.dagstuhl.de/05161/.
30. Vignaga, A., Transformación de un Modelo de Dominio y Diagramas de Comunicación en un Diagrama de Clases de Diseño, Dpto. Computación, Universidad de Chile (2006).