Evaluando el impacto de la automatización y la gestión de trazas en la calidad de transformaciones de modelos

Verónica A. Bollati[1], Juan Manuel Vara[2], David Granada[2], Esperanza Marcos[2]
[1]*Conicet - Universidad Tecnológica Nacional, Faculta Regional Resistencia*
Resistencia, Chaco - Argentina
vbollati@gmail.com
[2] *Grupo Kybele, Universidad Rey Juan Carlos*
Móstoles, Madrid – España
(juanmanuel.vara, david.granada, esperanza.marcos)@urjc.es

Abstract

En el contexto de la Ingeniería Dirigida por Modelos (Model-Driven Engineering, MDE), la generación de trazas puede ser automatizada usando las relaciones de trazabilidad implícitas en la transformación del modelo. Además, si las transformaciones de modelos se desarrollan utilizando una aproximación basada en modelos, se hace uso de las ventajas de MDE en términos de desarrollo de software menos costoso, reduciendo la complejidad de la codificación de las mismas. En trabajos previos se ha presentado MeTAGeM-Trace, una herramienta basada en EMF para el desarrollo dirigido por modelos de transformaciones de modelos con soporte a la generación de trazas. Sin embargo, la automatización podría tener un impacto negativo en la calidad de los productos generados. Para evaluar si este es realmente el caso con MeTAGeM-Trace, este trabajo presenta los principales resultados de un experimento en el que se comparar la calidad de transformaciones de modelos desarrolladas de diferentes maneras, incluyendo el uso de MeTAGeM-Trace, con el objetivo de evaluar el impacto de la automatización y la generación de trazas.

1. Introducción

La gestión de la trazabilidad en desarrollos de software implica mantener las relaciones entre los diferentes artefactos de software producidos durante el proceso de desarrollo. Una adecuada gestión de la trazabilidad ayuda a monitorear la evolución de los componentes del sistema y llevar a cabo diferentes actividades de software tales como la evaluación de impacto del cambio, la validación de requisitos, diversas tareas de mantenimiento, etc. [1]. Desafortunadamente, la generación y mantenimiento de las trazas entre los diferentes artefactos de software es una tarea lenta, tediosa y propensa a errores si no se cuenta con un soporte de herramientas [16].

El surgimiento de la Ingeniería Dirigida por Modelos (*Model-Driven Engineering*, MDE) cuyos principios son priorizar el rol de los modelos y aumentar el nivel de automatización en todo el proceso de desarrollo [23], provee un nuevo escenario que influye positivamente en diferentes actividades de la Ingeniería del Software como por ejemplo, la gestión de la trazabilidad [20]. De hecho, MDE ha dado lugar a la aparición de nuevos escenarios en los que la adecuada gestión de la trazabilidad es casi obligatoria, como la sincronización de modelos o el control de los cambios incrementales de modelos [21].

Según Bézivin [3], la clave para fomentar la automatización en proyectos MDE es por lo general un conjunto de transformaciones de modelos que conectan los diferentes modelos involucrados en la propuesta. En pocas palabras, una transformación de modelos define un conjunto de relaciones entre los elementos de los metamodelos origen y destino que deben cumplirse entre los elementos de los modelos conformes a dichos metamodelos [22]. De esta manera, una transformación de modelos contiene información implícita a partir de la cual se pueden obtener los enlaces (trazas) entre dichos elementos. Estos enlaces, pueden ser vistos como instancias de las relaciones definidas a nivel de metamodelo. Si ponemos esta información explícita en el propio modelo de transformación, se podría generar, además de los modelos de destino correspondientes, un modelo adicional que contenga las trazas entre los

elementos de los modelos involucrados en la transformación. Las relaciones de trazas generadas pueden, adicionalmente, ser recolectada en un modelo de trazas para luego ser procesadas utilizando técnicas de MDE, como transformaciones de modelos, emparejamiento de modelos (*model matching*) o fusión de modelos (*model merging*) [2].

En [12], [28] se presentó `MeTAGeM-Trace`, un framework que permite elevar el nivel de abstracción con el que se conciben y desarrollan las transformaciones de modelos con el fin de proporcionar una solución independiente del lenguaje de transformación. Específicamente, `MeTAGeM-Trace` es una herramienta para el desarrollo semiautomático de transformaciones modelo a modelo que soporta la generación de trazas. Esto significa que, además de la automatización de la generación de trazas a partir de la ejecución de las transformaciones de modelos, `MeTAGeM-Trace` también se beneficia de la idea de modelos de transformación, por tanto, aborda uno de los principales problemas, planteado por Selic [21], que dificultan la adopción de MDE: la ausencia de enfoques sistemáticos para la desarrollo de transformaciones de modelos.

Uno de los principales inconvenientes que se pueden encontrar en la automatización de cualquier proceso de desarrollo de software es, que esta automatización podría tener un impacto negativo en la calidad del producto generado. Un ejemplo de esto es el uso de los editores tipo WYSIWYG (*What You See Is What You Get*), que desde su aparición reciben una gran cantidad de críticas por la baja calidad de los sitios webs que generan [24]. Por tanto, es importante verificar el impacto que el proceso de automatización brindado por `MeTAGeM-Trace` tiene en la calidad del producto obtenido, es decir, en el código de la transformación generada.

En este trabajo se presenta el resultado de un experimento, basado en el propuesto por van Amstel et. al en [25], [26], que mide la calidad de las transformaciones que se generan automáticamente con `MeTAGeM-Trace`, y compararlos con la calidad de transformaciones generadas manualmente. En [26], van Amstel et. al, proponen una serie de métricas específicas para las transformaciones de modelos que consideran seis atributos de calidad. Estos atributos de calidad se determinan analizando el trabajo presentado por Boehm et al. [5] y la recomendación de la ISO 9126 [11] y aplicándolas a las transformaciones de modelos. De hecho, según algunos autores como Herbsleb et. al. [9], Humpherey [10] y la ISO [11], la calidad del producto está determinada por la calidad del proceso usado para generarlo. De esta manera, la evaluación de la calidad de las transformaciones producidas por `MeTAGeM-Trace` hace que sea posible de alguna manera validar la calidad del proceso que las genera.

Es importante mencionar, que este trabajo se centra en la calidad de las transformaciones de modelos generadas por `MeTAGeM-Trace` para el lenguaje de transformación ATL [13], ya que, como se verá más adelante, las métricas utilizadas son específicas para este lenguaje.

El trabajo se estructura de la siguiente manera: la sección 2 presenta una breve descripción del proceso seguido para llevar a cabo el experimento que permitió evaluar la calidad de las transformaciones de modelos. En la sección 3 se muestra un resumen de los resultados obtenidos con dicho experimento y que luego son analizados en la sección 4. Por último, en la sección 5 se muestran las conclusiones obtenidas, resumiendo las principales ideas y presentando las líneas de trabajo futuro.

2. Evaluación de la Calidad

Como se ha mencionado previamente, uno de los principales problemas que pueden surgir al momento de la automatización de cualquier proceso de desarrollo de software es la pérdida de calidad en el producto generado. Un ejemplo de esto es el uso de los editores tipo WYSIWYG (*What You See Is What You Get*) que permiten desarrollar sitios web viendo directamente el resultado final, y que desde su aparición, y aún en la actualidad, han sido muy criticados por la baja calidad de los sitios web generados [24].

Por otro lado, la ISO 9126 [11], el estándar para el aseguramiento de la calidad de los productos software, establece que una de las maneras de asegurar la calidad del proceso es evaluando la calidad del producto generado usando dicho proceso. Watts Humphrey [10], uno de los impulsores de la calidad del producto de software, estableció de manera similar en los seis principios de la calidad del software que "*La calidad de un producto es determinada por la calidad del proceso usado para desarrollarlo. Esto implica que para gestionar la calidad del producto, hay que gestionar la calidad del proceso utilizado para desarrollar ese producto. Si un producto de calidad tiene pocos o ningún defecto, significa que el proceso que lo generó tiene pocos o ningún defectos*".

Teniendo esto en mente, y con el objetivo de verificar el impacto que la automatización y la generación de trazas, en particular el proceso propuesto en `MeTAGeM-Trace`, tienen en la calidad de las transformaciones de modelos generadas, hemos considerado apropiado llevar a cabo un experimento basado en el propuesto por van Amstel et. al en [26] para medir la calidad de las transformaciones generadas con `MeTAGeM-Trace`.

Con el fin de mejorar el rigor de esta validación, se han seguido las guias para la realización de experimentos propuestos por Runeson and Höst en [18]. En particular, se ha utilizado el protocolo usado por Pérez-Castillo et. al. en [17] que está basado en la propuesta de Runeson and Höst. A continuación se muestra, de forma resumida, cada uno de los pasos realizados.

2.1. Selección del caso de estudio

Con el objetivo de considerar diferentes niveles de complejidad, hemos seleccionado los siguientes casos de estudios (de menor a mayor complejidad):

- La transformación **Class2Relational**, ampliamente conocida, encontrada en el sitio web de ATL[1], que contiene 114 líneas de código (*Lines of Code*, LOC).
- La transformación **KM32XML**, que permite transformar modelos KM3 a modelos XML, que también puede ser obtenida desde el sitio de ATL (166 LOC).
- La transformación **UML2XMLSchema**, una de las transformaciones de `M2DAT-DB`[2], una herramienta para el desarrollo de esquemas de Bases de Datos modernas [27]. Esta transformación permite transformar modelos conceptuales de datos, representados por diagrama de clases UML, en esquemas XML (460 LOC).

Es importante mencionar que se seleccionaron estos casos de estudios porque sus transformaciones son públicas y pueden ser usadas como referencia y ser comparadas con las producidas con `MeTAGeM-Trace`. Además, cualquier persona interesada puede descargar dichas transformaciones para verificar su complejidad.

Por otra parte, con el fin de llevar a cabo la validación de `MeTAGeM-Trace`, se han considerado cuatro diferentes tipos de implementaciones de dichas transformaciones:

- **Implementación de Referencia.** Implementaciones públicas disponibles en el sitio web de ATL. Estas implementaciones han sido desarrolladas manualmente y no brindan soporte a la generación de trazas.
- **Implementación de Referencia con soporte a trazas.** Dado que las implementaciones de referencia no soportan la generación de trazas, se ha utilizado la herramienta `TraceAdder` [14] para añadir la capacidad de generar trazas a las implementaciones de referencia. En otras palabras, son implementaciones de referencias que han sido alteradas para dar soporte a la generación de trazas.
- **Implementación con `MeTAGeM`.** `MeTAGeM` es un framework para el desarrollo dirigido por modelos de transformaciones de modelos; en particular, `MeTAGeM` permite el desarrollo semi-automático de transformaciones modelo a modelos en dos lenguajes de transformación, ATL and RubyTL [19]. Con el fin de hacer hincapié en el uso de la automatización en el desarrollo de la transformación, hemos considerado implementaciones de transformaciones desarrolladas con `MeTAGeM` [6], [7].
- **Implementación con `MeTAGeM-Trace`**. Por último, hemos comparado las implementaciones anteriores de cada transformación con los que se obtuvieron de forma semiautomática, siguiendo el proceso de MDD propuesto e implementado por `MeTAGeM-Trace`, para producir transformaciones modelo que soporten la generación de trazas.

Como lo hemos mencionado previamente, la validación presentada en este trabajo se ha realizado utilizando casos de estudios de ATL, dado que ATL provee más escenarios, documentación e información en general que cualquier otro lenguaje de transformación, en particular que ETL, que es el otro lenguaje soportado por `MeTAGeM-Trace`. Por lo tanto, fue más fácil encontrar soluciones de referencias para compararlas con las generadas con `MeTAGeM-Trace`. Sin embargo, el proceso de validación también puede ser aplicado a las transformaciones ETL [15].

2.2. Diseño

Para el desarrollo del caso de estudio nos hemos basado en la propuesta de van Amstel et. al. [26] que establece cómo una serie de métricas definidas explícitamente para el lenguaje ATL influyen positiva o negativamente en seis características de calidad que describen la calidad interna de un modelo de transformación: comprensibilidad, modificabilidad, concisión, integridad, consistencia y reutilización. Con el objetivo de establecer la relación entre las métricas y las características de calidad, los autores analizan el índice de correlación de Kendall entre dichos valores usando para ello el test de correlación de rangos no-paramétricos.

Los valores recogidos fueron comparados para obtener un indicador general de la calidad de cada transformación de modelos. Para esto, hemos utilizado una heurística que nos permite cuantificar la transformación desarrollada con respecto a la característica de calidad, lo que nos ayuda a determinar el nivel de calidad de una transformación respecto a los demás, dicha heurística es la siguiente:

Sea n el número de métricas, p el número de características y k el número de transformaciones cuyas características queremos estimar. (1)

Sea $X \in [-1,1]^{n \times p}$ la matriz que contiene los (2)

[1] http://www.eclipse.org/m2m/atl/basicExamplesPatterns/
[2] http://m2datdb.wordpress.com/

coeficientes de correlación de Kendall para cada par métrica y característica.

Sea $Y \in \mathbb{R}^{n \times k}$ la matriz que contiene las métricas para cada transformación. (3)

El objetivo es estimar la matriz $\sim Z \in [0,1]^{p \times k}$ con las características de cada transformación.

De esta manera, se puede calcular Z simplemente como la media ponderada de las métricas correspondientes, en la que los pesos están dados por los coeficientes de correlación.

Tabla 1. Matriz X.

Valor significado Nro. métricas influenciadas por atributo	90,00%											
	Compren.		Modific.		Integridad		Consist.		Reusa.		Concis.	
Métrica	C.C.	Sig.	C.C.	Sig.	C.C.	Sig.	C.C.	Sig.	C.C.	Sig.	C.C.	Sig.
# Elem por outp pattern	-0,375	0,026	-0,215	0,202	-0,228	0,180	0,124	0,472	-0,146	0,389	-0,122	0,474
# Calls resolveTemp()	-0,352	0,049	-0,358	0,045	-0,159	0,380	-0,088	0,632	-0,236	0,189	-0,179	0,323
# Calls resolveTemp x rule	-0,326	0,068	-0,306	0,087	-0,106	0,558	-0,061	0,741	-0,236	0,189	-0,153	0,399
# Ouput models	-0,407	0,029	-0,407	0,029	-0,391	0,038	-0,122	0,524	-0,345	0,066	-0,218	0,249
Called rule fan-in	-0,407	0,029	-0,407	0,029	-0,391	0,038	-0,122	0,524	-0,345	0,066	-0,218	0,249
# Unused helpers	-0,407	0,029	-0,407	0,029	-0,391	0,038	-0,122	0,524	-0,345	0,066	-0,218	0,249
# Input models	-0,407	0,029	-0,407	0,029	-0,391	0,038	-0,122	0,524	-0,345	0,066	-0,218	0,249
Unit fan-in	-0,407	0,029	-0,407	0,029	-0,391	0,038	-0,122	0,524	-0,345	0,066	-0,218	0,249
Unit fan-out	-0,407	0,029	-0,407	0,029	-0,391	0,038	-0,122	0,524	-0,345	0,066	-0,218	0,249
# Param per called rule	-0,407	0,029	-0,407	0,029	-0,391	0,038	-0,122	0,524	-0,345	0,066	-0,218	0,249
# Unused param x called rule	-0,407	0,029	-0,407	0,029	-0,391	0,038	-0,122	0,524	-0,345	0,066	-0,218	0,249
# Units	-0,407	0,029	-0,407	0,029	-0,391	0,038	-0,122	0,524	-0,345	0,066	-0,218	0,249
# Times a unit is imported	-0,318	0,088	-0,138	0,459	-0,379	0,045	0,086	0,654	-0,084	0,656	-0,247	0,192
Lazy rule fan-in	-0,356	0,037	-0,143	0,404	-0,397	0,021	0,005	0,976	-0,021	0,905	-0,062	0,719
# Imported units	-0,252	0,164	-0,080	0,661	-0,323	0,078	0,110	0,554	-0,027	0,883	-0,223	0,225
Helper cyclomatic complexity	-0,248	0,142	-0,154	0,364	-0,357	0,037	-0,026	0,882	-0,175	0,304	0,126	0,461
Rule fan-out	-0,223	0,175	-0,124	0,453	-0,333	0,046	-0,157	0,351	-0,235	0,157	0,024	0,885
# Direct copies	0,227	0,197	0,040	0,822	0,322	0,070	-0,059	0,745	-0,125	0,478	-0,196	0,271
Helper fan-out	0,020	0,907	0,183	0,278	-0,105	0,537	0,302	0,081	0,264	0,120	0,136	0,426
# Transform rules	0,024	0,885	-0,086	0,604	-0,082	0,623	-0,364	0,031	-0,273	0,099	-0,092	0,582
# Unus call rules	-0,128	0,482	-0,263	0,149	-0,158	0,389	-0,308	0,098	-0,365	0,046	-0,347	0,060
# Called rules	-0,128	0,482	-0,263	0,149	-0,158	0,389	-0,308	0,098	-0,365	0,046	-0,347	0,060
# Rules per input pattern	-0,109	0,507	-0,114	0,489	-0,130	0,434	0,029	0,861	-0,014	0,931	0,315	0,059
# Var per helper	0,006	0,972	0,063	0,727	0,013	0,944	-0,111	0,550	0,178	0,328	0,328	0,074
# Unus input patt elem	-0,032	0,854	0,059	0,737	-0,033	0,854	-0,056	0,758	0,033	0,854	0,297	0,097
# Rules with filter	-0,005	0,977	-0,038	0,818	-0,005	0,977	-0,049	0,771	-0,129	0,435	-0,402	0,016
# Non-lazy matc rules	0,014	0,931	0,000	1,000	0,034	0,839	-0,029	0,861	-0,129	0,435	-0,383	0,022

Por último, utilizamos escalas ~ Z de modo que cada elemento varía de 0 a 1, donde, un valor cercano a 1 se considera aceptable.

En la Tabla 1 se muestra la matriz X que contiene los coeficientes de correlación de Kendall para cada par métrica y atributo de calidad. Los coeficientes de correlación (cc) indican la fuerza y la dirección de la correlación, es decir, un cc positiva significa que hay una relación positiva entre el atributo de calidad y la métrica, y una cc negativa implica una relación negativa.

Por otra parte, el valor de significación (sig) indica la probabilidad de que no exista una correlación entre el atributo de calidad y la métrica. En otras palabras, la probabilidad de que exista una relación causal entre el atributo de calidad y la métrica. En la tabla, se muestran las correlaciones significativas (fondo gris), la aceptación de los niveles de significación de 0,10.

En la Tabla 2 se muestra la matriz Y, en el que podemos ver los valores obtenidos para cada una de las transformaciones de modelo analizadas para cada una de las métricas.

Table 2. Matriz Y.

	Class2Relational				KM32XML				UML2XMLW			
Metric	(1)	(2)	(3)	(4)	(1)	(2)	(3)	(4)	(1)	(2)	(3)	(4)
# Elements per output pattern	1,83	2,83	1,83	10,44	1,80	2,80	1,80	10,84	2,62	3,62	0,00	15,35
# Calls to resolveTemp()	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	7,00	7,00	6,00	7,00
# Calls to resolveTemp per rule	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,54	0,54	0,00	0,41
# Ouput models	1,00	2,00	1,00	2,00	1,00	2,00	1,00	2,00	1,00	2,00	2,00	2,00
Called rule fan-in	0,00	0,00	0,00	0,66	0,67	0,00	0,00	0,67	0,00	0,00	0,00	0,75
# Unused helpers	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	3,00	4,00	4,00	3,00
# Input models	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	2,00	2,00	2,00	2,00
Unit fan-in	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Unit fan-out	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
# Parameters per called rule	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
# Unused parameters per called rule	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
# Units	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
# Times a unit is imported	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Lazy rule fan-in	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
# Imported units	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Helper cyclomatic complexity	1,00	1,00	1,00	1,80	4,00	4,00	4,00	2,66	1,50	1,50	0,00	1,55
Rule fan-out	0,42	0,42	0,42	4,89	0,15	0,15	0,15	4,65	2,04	1,92	0,00	7,71
# Direct copies	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Helper fan-out	0,00	0,00	0,00	0,10	1,00	1,00	1,00	0,42	0,48	0,48	0,00	0,42
# Transformation rules	6,00	6,00	7,00	9,00	10,00	10,00	10,00	13,00	13,00	13,00	15,00	17,00
# Unused called rules	0,00	0,00	0,00	1,00	0,00	0,00	0,00	1,00	0,00	0,00	1,00	1,00
# Called rules	0,00	0,00	0,00	3,00	0,00	0,00	0,00	3,00	0,00	0,00	4,00	4,00
# Rules per input pattern	2,00	2,00	2,00	2,00	1,00	1,00	1,00	1,00	2,60	2,60	0,00	2,60
# Variables per helper	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,04	0,04	0,00	0,03
# Unused input pattern elements	0,00	0,00	0,00	0,00	2,00	0,00	2,00	0,00	0,00	0,00	0,00	0,00
# Rules with filter	4,00	4,00	4,00	4,00	0,00	0,00	0,00	0,00	11,00	11,00	9,00	11,00
# Non-lazy mat rules	6,00	6,00	7,00	6,00	10,00	10,00	8,00	10,00	13,00	13,00	11,00	13,00

Por último, aplicando la fórmula presentada previamente podemos obtener la matriz Z, simplemente como la media ponderada de los indicadores correspondientes, en la que los pesos están dados por los coeficientes de correlación (Tabla 3).

3. Extracción de Datos

La Tabla 4 muestra la matriz ~ Z que resulta del proceso de verificación de calidad, en la que es posible observar los valores producidos por las métricas para cada una de las características de calidad evaluadas para las diferentes implementaciones en cada uno de los casos de estudio: implementación de referencia (1); implementación de referencia con soporte a trazas (2); implementación con `MeTAGeM` (3) e implementación con `MeTAGeM-Trace` (4). Además, se muestra el valor de cada característica de calidad, junto con el valor global de calidad para cada transformación (calculado simplemente como la media aritmética de los valores anteriores).

Tabla 3. Matriz Z.

Filas afectadas de la MatrizX			5-18	6-16	8-22	23-26	8-16 y 24-26	25-31	
Métrica			Comprens.	Modific.	Integridad	Consist.	Reusab.	Concis.	CALIDAD
Class2Relational	1	B	-0,35	-0,28	-0,33	-1,20	-1,18	-0,68	-0,67
	2	C	-0,50	-0,38	-0,43	-1,20	-1,29	-0,68	-0,75
	3	D	-0,35	-0,28	-0,33	-1,40	-1,32	-0,68	-0,73
	4	E	-1,08	-0,44	-0,65	-4,21	-3,50	-2,58	-2,08
KM32XML	1	F	-0,40	-0,35	-0,50	-1,58	-1,81	0,01	-0,77
	2	G	-0,50	-0,38	-0,53	-1,58	-1,85	-0,21	-0,84
	3	H	-0,35	-0,28	-0,44	-1,58	-1,74	0,01	-0,73
	4	I	-1,11	-0,44	-0,68	-4,88	-4,06	-2,10	-2,21
UML2XMLW	1	J	-1,20	-1,28	-0,78	-2,40	-2,60	-1,28	-1,59
	2	K	-1,42	-1,46	-0,97	-2,40	-2,83	-1,28	-1,73
	3	L	-1,07	-1,34	-0,86	-6,07	-5,26	-2,98	-2,93
	4	M	-2,21	-1,43	-1,12	-6,29	-5,51	-3,65	-3,37
Promedio			-0,88	-0,69	-0,64	-2,90	-2,75	-1,34	

Con el objetivo de evaluar cuantitativamente el impacto de agregar el soporte a la generación de trazas hemos comparado los valores obtenidos para las transformaciones de referencia (1) con los valores obtenidos por las transformaciones de referencia con soporte a trazas (2); de la misma manera, se comparan los valores obtenidos por la transformación generada con `MeTAGeM` (3) con los valores obtenidos por la transformación generada con `MeTAGeM-Trace` (4).

A modo de ejemplo, en el primer caso, la implementación de referencia de Class2Relational obtiene un valor de 1.00 para la característica de Comprensibilidad, mientras que la implementación de referencia con soporte a trazas obtiene el valor de 0.92. Es decir, la comprensibilidad de la transformación Class2Relational con soporte a traza disminuye en un 7,80% con respecto a la implementación de referencia.

En el segundo caso, la implementación Class2Relational generada con `MeTAGeM` obtiene un valor de 1.00 para la misma característica, mientras que la implementación generada con `MeTAGeM-Trace` obtiene el valor de 0.61. Es decir, la comprensibilidad de la transformación Class2Relational desarrollada con `MeTAGeM-Trace` disminuye en un 38.96% con respecto a la transformación desarrollada con `MeTAGeM`. A primera vista, se puede decir que el soporte a trazas tiene una influencia negativa sobre la característica de comprensibilidad; en la siguiente sección se dará más detalles sobre esta afirmación.

Como se puede observar, el resto de los atributos de calidad analizados tienen un comportamiento similar: el soporte a trazas tiene un impacto negativo en la calidad de las transformaciones de modelos. Otra observación relevante es que las distancias (en términos de calidad) entre las diferentes implementaciones de la misma transformación se hacen más grandes a medida que

aumenta la complejidad de dicha transformación. Por ejemplo, la implementación de UML2XML desarrollada con `MeTAGeM` obtiene el valor de 0.61 para la característica de comprensibilidad, mientras que la implementación desarrollada con `MeTAGeM-Trace` tiene el valor de 0.00. Esto es, la comprensibilidad de la transformaciones UML2XML desarrollada con `MeTAGeM-Trace` disminuye un 61.29% con respecto a la implementación de la misma transformación desarrollada con `MeTAGeM`.

En la última fila se muestran los datos resumidos, indicándo los valores promedios, las diferencias de cada característica y el indicador global.

Tabla 4. Matriz ~ Z

Transf.		Comprens.		Modifi.		Integridad.		Consis.		Reusab.		Concis.		CALIDAD	
Class2Relacional	1	1,00		1,00		1,00		1,00		1,00		0,81		0,97	
	2	0,92	*-7,80%*	0,92	*-7,96%*	0,88	*-12,23%*	1,00	*0,00%*	0,97	*-2,56%*	0,81	*0,00%*	0,92	*-5,09%*
	3	1,00		1,00		1,00		0,96		0,97		0,81		0,96	
	4	0,61	*-38,96%*	0,87	*-13,21%*	0,59	*-40,65%*	0,41	*-55,20%*	0,46	*-50,41%*	0,29	*-51,79%*	0,54	*-41,70%*
KM32XML	1	0,97		0,95		0,79		0,93		0,85		1,00		0,91	
	2	0,92	*-5,08%*	0,92	*-2,63%*	0,75	*-4,03%*	0,93	*0,00%*	0,84	*-0,85%*	0,94	*-5,81%*	0,88	*-3,07%*
	3	1,00		1,00		0,87		0,93		0,87		1,00		0,94	
	4	0,59	*-40,61%*	0,87	*-13,29%*	0,56	*-30,98%*	0,28	*-64,87%*	0,33	*-53,68%*	0,42	*-57,60%*	0,51	*-43,51%*
UML2XML	1	0,54		0,16		0,43		0,76		0,67		0,65		0,54	
	2	0,42	*-11,87%*	0,00	*-15,92%*	0,19	*-24,02%*	0,76	*0,00%*	0,62	*-5,13%*	0,65	*0,00%*	0,44	*-9,49%*
	3	0,61		0,10		0,33		0,04		0,06		0,18		0,22	
	4	0,00	*-61,29%*	0,03	*-7,48%*	0,00	*-32,81%*	0,00	*-4,36%*	0,00	*-5,84%*	0,00	*-18,23%*	0,00	*-21,67%*
Prom.		0,72	*-27,60%*	0,65	*-10,08%*	0,62	*-24,12%*	0,67	*-20,74%*	0,64	*-19,75%*	0,63	*-22,24%*	0,65	*-20,75%*

La Tabla 5 muestra el valor promedio obtenido para cada característica de calidad por cada grupo de implementaciones, y un indicador general de la calidad de las transformaciones producidas, calculado como el promedio de las características de calidad. A continuación hará un análisis de los resultados obtenidos.

Tabla 5. Valores promedios para cada una de las características de calidad

	Promedio						
	Comp.	Mod.	Int.	Cons.	Reus.	Conc.	**Prom Calidad**
1	0,84	0,70	0,74	0,90	0,84	0,82	0,81
2	0,76	0,61	0,60	0,90	0,81	0,80	0,75
3	0,87	0,70	0,73	0,64	0,63	0,66	0,71
4	0,40	0,59	0,38	0,23	0,27	0,24	0,35

Por último, en la Fig. 1 se muestran los resultados obtenidos por medio de un gráfico de barras.

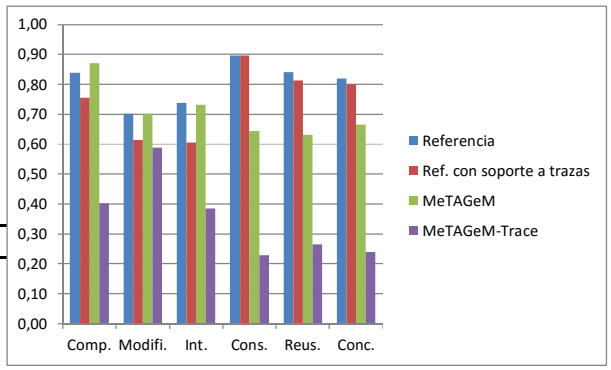

Fig. 1. Valores promedios para cada una de las características de calidad

4. Discusión

Con respecto al impacto que tiene brindar el soporte a la generación de trazas en los modelos de transformación y de acuerdo a lo mostrado en la Tabla 4, se puede concluir que proveer el soporte a la generación automática de trazas, en general, tiene un impacto negativo en la calidad de las transformaciones.

De hecho, si comparamos las dos implementaciones manuales, es decir, implementaciones de referencia (1) y las implementaciones de referencia enriquecidas con `TraceAdder` (2), se observa una pérdida de calidad para los tres casos de estudios.

De la misma manera, si comparamos las dos transformaciones desarrolladas de forma semi-automática, esto es, las implementaciones con MeTAGeM (3) y MeTAGeM-Trace (4) la pérdida de calidad es mucho más evidente.

Estas evidencias están, de hecho, alineadas con la intuición inicial, ya que el soporte a la generación de trazas implica líneas de código adicional con el fin de poner en práctica la maquinaria que va a generar dichas trazas. Las transformaciones más grandes son afectadas en mayor medida (KM32XML y UML2XMLSchema): cuantas más reglas de mapeo tenga la transformación original, más reglas deberán ser añadidas para enriquecer la transformación. Sería necesario un análisis en profundidad, incluyendo más casos de estudios, a fin de obtener mayor precisión en los valores que evidencian la pérdida de calidad al brindar soporte a la generación de trazas.

En cuanto a la automatización, se obtienen valores similares en las implementaciones de referencia (1), desarrollada manualmente, y la llevada a cabo de manera semi-automática con MeTAGeM (3), la diferencia es de menos el 3%. Incluso en el caso de la transformación KM32XML, la implementación desarrollada con MeTAGeM obtiene un indicador de calidad mejor (0.94) que la desarrollada de forma manual. Esto nos permitiría afirmar que la automatización no tiene una influencia significativa en la calidad de las transformaciones de modelos.

Por desgracia, la comparación entre las dos implementaciones que brindan soporte a la generación de trazas: implementación de referencia enriquecida con TraceAdder (2) y las implementaciones realizadas con MeTAGeM-Trace (4) no producen el mismo resultado ya que se detecta una pérdida de calidad importante.

La razón de esto es porque la transformación enriquecida con TraceAdder produce trazas a nivel de objeto, mientras que la implementación con MeTAGeM-Trace produce trazas también a nivel propiedades que contiene mayor información. Por ejemplo, con MeTAGeM-Trace se registra el tipo de operación a partir de la cual se deriva la traza. Esto implica mucha más líneas de código adicionales. Además, TraceAdder fue diseñado específicamente para enriquecer transformaciones ATL mientras que MeTAGeM-Trace está destinado a ser tan genérico como sea posible, en la actualidad permite enriquecer transformaciones ATL y EpsilonTL. La solución implementada con MeTAGeM-Trace podría no ser la más óptima para el desarrollo de transformaciones ATL.

Dado todo esto, se podría plantear desarrollar una versión menos ambiciosa de MeTAGeM-Trace, diseñada específicamente para brindar soporte a la generación de trazas para transformaciones ATL, lo que con seguridad permitiría obtener mejores valores para los indicadores de calidad.

Teniendo esto en mente, y desde nuestro punto de vista, la pérdida de calidad de MeTAGeM-Trace es insignificante si se compara con los beneficios de obtener el código que implementa una transformación modelo con soporte a la generación de trazas forma semiautomática.

5. Conclusiones

En este trabajo se presentan los principales resultados de un experimento llevado a cabo con el objetivo de evaluar el impacto de la automatización y el soporte a la generación de trazas en la calidad de las transformaciones de modelos.

Este experimento se ha llevado a cabo tomando como base la propuesta de van Amstel et. al. [26], que establece cómo una serie de métricas que se definen explícitamente para el lenguaje ATL tiene un efecto positivo o negativo sobre seis características de calidad que describen la calidad interna de un modelo de transformación: comprensibilidad, modificabilidad, concisión, integridad, consistencia y reutilización.

En particular, nos hemos centrado en la evaluación de MeTAGeM-Trace, un framework que permite el modelado de forma automática de transformaciones de modelos con soporte a la generación de trazas. Parte del experimento consistió en comparar los valores obtenidos en las métricas con el uso de MeTAGeM-Trace con los obtenidos mediante el uso de otras propuestas en diferentes casos de estudio.

Como lo hemos planteado en la sección de Discusión, una de las principales conclusiones a las que hemos arribado después de llevar a cabo el experimento es que algunas intuiciones iniciales se han confirmado: tanto la automatización como el soporte a la generación de trazas tienen un impacto negativo en la calidad de las transformaciones de modelos. En particular, el impacto del soporte a la generación de trazas es mucho más relevante.

Sin embargo, todavía creemos que este impacto es asumible si se consideran las ventajas aportadas por estas dos características. Como el hecho de que, la automatización (en realidad la automatización basada en modelos) y la trazabilidad se reconocen como áreas cruciales de la Ingeniería de la Calidad del Software [8].

Del mismo modo, un análisis exhaustivo de los datos recogidos puede producir algunos resultados más útiles. Al final no es más que una cuestión de jugar con estos datos; la parte más difícil que es la recolección de datos se ha hecho ya.

También se plantea ampliar el experimento para incluir otros lenguajes de transformación, tales como ETL [15]. Para esto, será necesario adaptar las métricas

propuestas por van Amstel et. al. [26] con el objetivo de incluir estos lenguajes de transformación. También tenemos la intención de llevar a cabo el mismo experimento con casos de estudios más complejos que, por ejemplo, incluyan más de un modelo de origen y generen más de un modelo de destino.

Consideramos que ahora que MDE ha alcanzado cierto nivel de madurez [21], ha llegado el momento de que los adeptos a MDE empiecen a considerar a los problemas de calidad en el desarrollo de propuestas basadas en modelos, que hasta ahora se han descuidado en favor de mostrar que MDE se podría utilizar con eficacia.

6. Agradecimientos

Esta investigación ha sido parcialmente financiada por el Gobierno de la Comunidad de Madrid bajo el proyecto SICOMORo-CM (S2013/ICE-3006), por el proyecto ELASTIC (TIN2014-52938-C2-1-R), financiado por el Ministerio de Ciencia e Innovación del Gobierno de España y por el Grupo de Excelencia Investigadora en Service Science, Management and Engineering-GES2ME (Ref. Orgánica 30VCPIGI05) co-financiado por la Universidad Rey Juan Carlos y el Banco Santander y co-financiado por el CONICET y la Universidad Tecnológica Nacional.

7. Referencias

[1]. N. Aizenbud-Reshef, B. T. Nolan, J. Rubin, and Y. Shaham-Gafni. "Model traceability", *IBM Systems Journal*, 45(3):515-526, 2006.

[2]. P. A. Bernstein and S. Melnik. "Model management 2.0: manipulating richer mappings" *in Proc. 2007 ACM SIGMOD international conference on Management of data, SIGMOD '07*, pages 1-12, New York, NY, USA, 2007. ACM.

[3]. Bézivin, J. "In search of a Basic Principle for Model Driven Engineering". *Novatica/Upgrade*, V(2), 21-24.

[4]. Bézivin, J., Büttner, F., Gogolla, M., Jouault, F., Kurtev, I., & Lindow, A. "Model Transformations? Transformation Models!", *in Proc 9th International Conference on Model Driven Engineering Languages and Systems, MoDELS 2006*, Genève, Italy.

[5]. Boehm, B.W., Brown, J.R., Kaspar, H., Lipow, M., Macleod, G.J., Merrit, M.J. Characteristics of Software Quality. North-Holland. 1978

[6]. Bollati, V. A. "MeTAGeM: a framework for Model-Driven Development of Model Transformations". Ph. D. Thesis. Rey Juan Carlos Univ.. http://www.kybele.etsii.urjc.es/members/vbollati/Thesis/. 2011.

[7]. Bollati, V. A., Vara, J. M., Jimenez, A., Marcos, E., "Applying MDE to the (semi-) automatic development of model transformations", *Information and Software Technology*, vol 55 (5), pp. 699-718, ISSN 0950-5849.

[8]. Breu, R., & Kuntzmann-combelles, A. "New Perspectives on Software Quality". *IEEE Software*, *34*(1), 32–38. (2014)

[9]. Herbsleb, J., Zubrow, D., Goldenson, D., Hayes, W., & Paulk, M. (1997), "Software quality and the Capability Maturity Model", *Commun. ACM*, vol. 40 (6), pp. 30-40.

[10]. Humpherey, W.S (2005), "Acquiring Quality Software", *CrossTalk, The Journal of Defense Software Engineering*. Vol. 18, Nº.12, December 2005.

[11]. ISO (International Standards Organization for Standardization) ISO 9126. Retrieved, February 20, 2013, from:http://www.issco.unige.ch/en/research/projects/ewg96/node13.html

[12]. Jiménez, Á. "Integrating traceability management in a framework for MDD of model transformations". PhD thesis, Rey Juan Carlos Univ. (2012).

[13]. Jouault, F., Allilaire, F., Bézivin, J. & Kurtev, I. „ATL: A model transformation tool". *Sci. Comput. Program.* 72, 1-2 (June 2008), 31-39.

[14]. F. Jouault (2005), "Loosely coupled traceability for ATL", *in Proc. 1st European Conference on Model-Driven Architecture Foundations and Applications (ECMDA-FA'05)*, Nuremberg, Germany, 2005.

[15]. D. Kolovos, R. Paige, and F. Polack. "The Epsilon Transformation Language". *Theory and Practice of Model Transformations*, volume 5063 of LNCS, pages 46{60. Springer Berlin / Heidelberg, 2008.

[16]. P. Mäder, O. Gotel, and I. Philippow. "Enabling automated traceability maintenance through the upkeep of traceability relations". *Model Driven Architecture - Foundations and Applications*, volume 5562 of LNCS, pages 174-189. Springer Berlin / Heidelberg, 2009.

[17]. R. Pérez-Castillo, J.A. Cruz-Lemus, I. García-Rodríguez de Guzmán, M. Piattini, "A Family of Case Studies on Business Process Mining Using MARBLE", *Journal of Systems and Software* 85(6) (2012) 1370-1385.

[18]. P.Runeson, M. Höst, "Guidelines for Conducting and Reporting Case Study Research" *Software Engineering, Empirical Sotware Engineering*, 14(2), (2009) 131-164.

[19]. J. Sánchez Cuadrado, J. García Molina, M. Menarguez Tortosa, "RubyTL: a practical, extensible transformation language", *in Proc: European Conference on Model Driven Architecture – Foundations and Applications*, Bilbao, Spain, 2006.

[20]. I. Santiago, A. Jiménez, J. M. Vara, V. De Castro, V. A. Bollati, and E. Marcos. "Model-Driven Engineering as a new landscape for traceability management: A systematic literature review". *Inf. Softw. Technol.*, 54(12):1340{1356, Dec. 2012.

[21]. B. Selic. "What will it take? a view on adoption of model-based methods in practice". *Software and Systems Modeling*, pages 1-14, 2012.

[22]. Sendall, S. & Kozaczynski, W. (2003), "Model transformation: the heart and soul of model-driven software development", *Software, IEEE,* vol. 20 (5), pp. 42-45.

[23]. D. Schmidt. “Model-Driven Engineering”. *IEEE Computer*, 39(2):25-31, 2006.

[24]. Spiesser, J., Kitchen, L. “Optimization of HTML automatically generated by WYSIWYG programs”, *in Proc: 13th International Conference on World Wide Web*, pp. 355–364. WWW '04. ACM, New York, NY (New York, NY, Estados Unidos, 17-20 May 2004)

[25]. van Amstel, M. & van den Brand, M. (2010), "Quality assessment of ATL model transformations using metrics", in *2nd International Workshop on Model Transformation with ATL (MtATL2010)*, Malaga, Spain, 2010.

[26]. van Amstel, M. & van den Brand, M. (2011), "Using Metrics for Assessing the Quality of ATL Model Transformations", in *3rd International Workshop on Model Transformation with ATL (MtATL2011)*, Zürich, Switzerland, 2011, pp. 20-34.

[27]. Vara, J.M, (2009). “M2DAT: a Technical Solution for Model-Driven Development of Web. Information Systems”. PhD Thesis. Rey Juan Carlos Univ., November 2009.

[28]. Vara, J.M., Bollati, V.A., Jimenez, A. Marcos, E. “Dealing with traceability in the MDD of model transformations”. *IEEE Transactions on Software Engineering*. 40(6) (June 2014).