ARTÍCULO SELECCIONADO DEL CONAIISI

Evaluación en MDE: Medición y evaluación de transformaciones a nivel metamodelo.

Ing. Corina Natalia Abdelahad[1]
Ing. Enrique Alfredo Miranda[2]
Ing. Norma Beatriz Pérez[3]
Ing. Daniel Edgardo Riesco[4]

Universidad Nacional de San Luis

[1] Magister en Ingeniería de Software.
E-mail: cabdelah@unsl.edu.ar

[2] Licenciado en Ciencias de la Computación
E-mail: eamiranda@unsl.edu.ar

[3] Magister en Ingeniería de Software
E-mail: nbperez@unsl.edu.ar

[4] Doctor en Ingeniería de Software
E-mail: driesco@unsl.edu.ar

RESUMEN

La Ingeniería Dirigida por Modelos está en auge y junto con ella han surgido diversas propuestas de lenguajes de transformación. QVT Relations, componente de QVT, permite formalizar transformaciones dirigidas por modelos. Debido al aumento de la utilización de estos lenguajes, es importante comenzar a considerar aspectos relacionados al mantenimiento, reutilización y comprensión de las transformaciones, ya que estas desempeñan un papel fundamental proporcionando un mecanismo para expresar el refinamiento de modelos. Este trabajo presenta una estrategia de medición y evaluación de transformaciones QVT a nivel metamodelo, asistiendo al ingeniero de software en el mantenimiento y evolución en las mismas.

ABSTRACT

For several years, Model Driven Engineering has been widely used. Several model transformation languages have been proposed. The QVT Relations, a component of QVT, allows to formalize model-driven transformations. Given the increased utilization of these languages, it is important to consider aspects related to maintenance, reusability and comprehension of the transformations. Indeed, these transformations provide a fundamental mechanism to express the refinement of models. This paper presents a measurement and evaluation strategy of QVT transformations at the metamodel level, which will help software engineers to carry out maintenance and evolution tasks in these transformations.

PALABRAS CLAVES: MDE, Métricas, Transformación de Modelos, QVT.

1. INTRODUCCIÓN

La Ingeniería Dirigida por Modelos (Model-Driven Engineering-MDE) [1] ofrece un escenario ideal para potenciar el papel de la trazabilidad en el desarrollo de software. MDE propone un proceso de desarrollo de software en el cual la clave son los modelos y las transformaciones entre ellos. En este proceso, el software es desarrollado construyendo uno o más modelos, y transformando estos en otros modelos o transformarlos hasta llegar a código ejecutable. La transformación de modelos se refiere al proceso de transformación (relaciones y mapeo) de elementos de un modelo a elementos correspondientes de otro modelo. Los modelos son especificados a través de los metamodelos los cuales definen su sintaxis. Es decir, un metamodelo define el lenguaje con el cual se construyen modelos. Una transformación de modelos toma como entrada un modelo acorde a un determinado metamodelo y produce como salida

otro modelo acorde también a un determinado metamodelo.

Teniendo en cuenta lo expuesto en el párrafo previo, varias organizaciones proponen lenguajes para expresar consultas y definir transformaciones entre modelos. Debido al importante crecimiento en la utilización de dichos lenguajes, es imprescindible comenzar a contemplar tareas de mantenimiento y evolución en dichas transformaciones. En este contexto, las transformaciones poseen dificultades similares vinculadas a las tareas de mantenimiento y evolución en sistemas escritos en lenguajes de programación de propósito general como Java, C, C#, etc.

Por lo general, los desarrolladores de software tradicionales toman acciones correctivas como refactorizaciones o revisiones de código para conservar el código en un estado mantenible. Estos indicadores aún no se hacen presentes en los lenguajes de transformación declarativos como QVT Relations (QVT-R). Sin embargo, algunas investigaciones iniciales usan enfoques similares a métricas para lenguajes de programación funcional como Lisp o Haskell. Teniendo en cuenta que los lenguajes de transformación declarativos son parte de la familia del paradigma funcional, ciertas métricas para este tipo de paradigma pueden ser útiles como punto de partida para la definición de métricas para lenguajes de transformación declarativos [2].

La definición de métricas puede ayudar a medir el mantenimiento, comprensión y reutilización de las transformaciones de modelos. Estas métricas definen como representar y obtener los valores de los atributos. A través de los indicadores se define como se interpretarán los valores de estos atributos y además se especifica e implementa la evaluación de la métrica. Teniendo en mente este objetivo, en el contexto de este trabajo se definen criterios y métricas que pueden ser utilizadas para medir y evaluar la comprensión, mantenimiento y el reuso de las transformaciones relacionales. Además se muestra que la dificultad de comprender, mantener y reutilizar aumenta a medida que las transformaciones crecen y las relaciones individuales se vuelven más complejas. Por otra parte se exhiben dos casos de estudio a los cuales se les aplican las métricas presentadas en este trabajo.

El trabajo se estructura de la siguiente manera: la sección 2 describe los trabajos relacionados con la temática que se presenta en el artículo; la sección 3 introduce los lenguajes de transformación, en particular QVT; la contribución de este artículo se presenta en la sección 4; en la sección 5 se exponen dos casos de estudio a los cuales se les han aplicado las métricas definidas en la sección 4 y finalmente, en la sección 6 se presentan las conclusiones y los trabajos futuros.

2. TRABAJOS RELACIONADOS

Son numerosas y variadas las investigaciones, en el estado del arte, que se centran en medición de la calidad. En este contexto, se han definido nuevas métricas a través de técnicas y estrategias que permiten llevar adelante un proceso de medición más específico. La mayoría de los estudios que definen métricas cubren un amplio conjunto de paradigmas y lenguajes de programación como: lenguajes OO, procedurales, lógicos, funcionales, entre otros. Sin embargo, son relativamente pocos los trabajos fuertemente relacionados con la temática presentada en este artículo. A continuación se mencionan los trabajos más relevantes en este campo y que han motivado el desarrollo de este trabajo.

Luis Reynoso et al. [3] en su artículo persiguen dos objetivos principales: (i) exponen un conjunto de métricas para medir las propiedades estructurales de las expresiones OCL (Object Constraint Language) y (ii) proponen un método de validación teórico para dichas métricas a fin de asegurar la correctitud de las mediciones empleadas.

Mohagheghi y Dehlen [4] presentan una estrategia de 7 pasos sobre cómo definir un framework de calidad que se adapte a MDE, integrando ingeniería de calidad con evaluación de la calidad. A modo de ejemplo, aplican el framework sobre la calidad de transformaciones.

Van Amstel et al. [5] presentan las primeras investigaciones sobre calidad en transformaciones de modelos. Con el transcurso de los años, este grupo de investigación propuso diferentes enfoques centrados en transformación de modelos [6], utilizando lenguajes como ASF + SDF [7] y ATL [8].

Lucia Kapova et al. [2] proponen un conjunto inicial de indicadores de calidad que permiten evaluar las transformaciones escritas en QVT-R. Estos indicadores se aplicaron a diversas transformaciones con el objeto de mostrar cómo juzgar la mantenibilidad de la transformación. Una característica a destacar del enfoque propuesto es la extracción automática de métricas.

Los trabajos expuestos, en los párrafos anteriores, evalúan distintos aspectos de calidad definiendo un conjunto inicial de métricas y/o atributos.

Sin embargo, estos carecen de una especificación detallada de la estrategia de medición y evaluación de atributos. La mayoría de los casos de estudios propuestos no han sido aplicados a transformaciones definidas en un entorno de desarrollo industrial. Además dichos casos de estudio se exhiben subespecificados, aspecto que dificulta la interpretación y comprensión de la aplicabilidad de los enfoques propuestos.

En este trabajo, se propone la definición de un conjunto de métricas y atributos para la medición y evaluación de las transformaciones de modelos en lenguajes declarativos como QVT-R. Se empleó un framework basado en el trabajo de Olsina et al. [9]. Para la evaluación experimental, a diferencia de otras investigaciones, se muestran dos casos de estudio relevantes. Un caso de estudio fue implementado por uno de los autores de este trabajo [10] el cual fue diseñado y desarrollado para la industria. El otro caso de estudio fue presentado en [11] para resolver los problemas surgidos en el estudio de los mecanismos de señales de las células (pathway) en el campo de la bioinformática.

3. LENGUAJES DE TRANSFORMACIÓN

Como se mencionó anteriormente, MDE impulsa la utilización de modelos como artefactos principales a ser construidos y mantenidos. Cualquier especificación puede ser expresada con modelos, en consecuencia un proceso de desarrollo de software se convierte en un proceso de refinamiento y transformación entre modelos de manera tal que el nivel de abstracción vaya decrementando hasta llegar al código ejecutable. La definición de transformaciones de modelos requiere la aplicación de lenguajes específicos. Estos lenguajes deberían tener una base formal, y al menos un metamodelo que describa su sintaxis abstracta [12]. Un metamodelo describe el conjunto de modelos admisibles, como se mencionó anteriormente.

Los lenguajes de transformación híbridos combinan distintas características que proveen los lenguajes declarativos y los imperativos. Los lenguajes declarativos poseen la ventaja de ser razonados matemáticamente y por lo general se limitan a escenarios donde los metamodelos de entrada y de salida poseen una estructura similar. Si bien los lenguajes imperativos permiten manejar un mayor número de escenarios, poseen un nivel de abstracción menor. Esto deriva en un mayor esfuerzo por parte del usuario para resolver cuestiones como la trazabilidad.

QVT es un lenguaje estándar para la transformación de modelos definido por la OMG [13]. Su especificación depende de los estándares OCL 2.0 y MOF 2.0. Con QVT es posible definir transformaciones genéricas entre metamodelos, de esta manera cualquier instancia de un metamodelo puede ser transformado en una instancia de otro metamodelo. La especificación QVT cuenta con una naturaleza híbrida, es decir declarativa e imperativa. QVT provee un lenguaje imperativo llamado Operational Mappings y uno declarativo llamado QVT-R. En este lenguaje es necesario declarar parámetros en una transformación para manejar los modelos involucrados en la misma. Estos parámetros están tipados sobre los metamodelos apropiados.

Una transformación en QVT-R especifica un grupo de relaciones que los elementos de los modelos involucrados deben satisfacer. Una transformación puede tener cualquier número de parámetros de entrada y salida llamados dominios. Una relación define reglas de transformación, y esa relación implica la existencia de clases para cada uno de sus dominios. Las relaciones pueden contener cláusulas *when* y *where*. Estas cláusulas pueden contener cualquier expresión OCL además de las expresiones de invocación de la relación. Una transformación puede contener dos tipos de relaciones: relaciones *top-level* y relaciones *no-top-level*. La ejecución de una transformación requiere que todas sus relaciones *top-level* se cumplan, mientras que las relaciones *no-top-level* se deben cumplir sólo cuando son invocadas directa o indirectamente a través de la cláusula where de otra relación. Un tipo de modelo está definido por un metamodelo y un conjunto opcional de expresiones de condición. El lector interesado puede consultar estos conceptos en la especificación QVT [14].

Es importante destacar que la introducción de estos conceptos (lenguajes de transformación y conceptos propios de QVT-R) son las bases, en que los autores de este trabajo se apoyaron, para definir métricas para transformaciones QVT-R.

4. UNA ESTRATEGIA PARA MEDIR Y EVALUAR TRANSFORMACIONES

Como bien se induce en apartados previos, el proceso involucrado en medir los posibles cambios y mejoras en una transformación no ha sido presentado en ningún trabajo hasta el momento. Se considera que la definición de métricas y posterior evaluación permitirá mejorar el proceso de

desarrollo mediante distintos tipos de tareas relacionadas al mantenimiento, reuso y comprensión.

Para especificar el conjunto de métricas se ha utilizado un enfoque similar al propuesto por Olsina et al. [9]. En dicho trabajo, los autores proponen modelos y frameworks de calidad en conjunto con estrategias de evaluación que apuntan a un enfoque integrado para medir y evaluar diferentes áreas de las nuevas aplicaciones que se utilizan en la web. Por otra parte, en este trabajo se propone en primera instancia definir un conjunto de métricas y sus correspondientes indicadores y posteriormente se plantea enriquecer la estrategia empleando un método de evaluación multicriterio propuesto por Dujmovic et al. [15,16].

En general, al momento de medir y evaluar cualquier tipo de artefacto (ya sea, un programa, una técnica, un proyecto, etc.) de forma sistemática, es necesario definir una estrategia que contemple un conjunto de principios, métodos, técnicas y herramientas; que permitan especificar, definir y recolectar métricas e indicadores y sus respectivos valores [9]. Además, con el objetivo de llevar a cabo el análisis y el proceso de toma de decisión, es necesario asegurar que las medidas (etapa de especificación de métricas, apartado 4.2) y los valores de los indicadores (etapa de evaluación, apartado 4.3) sean repetibles y comparables entre todos los demás en el proyecto. Por lo tanto, es mandatorio guardar no solo los datos de la medición, sino también ciertos metadatos como procedimiento de medición, escala, tipo de escala, modelo de indicador elemental, niveles de aceptabilidad, entre otros.

Una métrica es la especificación de un proceso de medición que trasforma un atributo de una entidad en una medida. Un indicador elemental es la especificación de un proceso de evaluación, el cual recibe como entrada los valores correspondiente a una métrica en particular y produce un valor indicador (es decir, información contextual). Sin embargo en los distintos trabajos relacionados a la temática, en general, se le ha dado poca relevancia a la especificación de este tipo de conceptos.

En las siguientes subsecciones se presentan los pasos de la estrategia para medición y evaluación de transformaciones especificadas en QVT-R, desde el punto de vista del mantenimiento [17].

4.1. DEFINICIÓN DE ATRIBUTOS

Por lo general, en la estrategia de medición y evaluación se debe definir un conjunto de atributos (o requerimientos) agrupados en categorías, subcategorías, etc. De esta manera se obtiene una estructura de árbol que pasa a ser el elemento principal del proceso de medición y evaluación.

En este trabajo en particular se ha definido un árbol de requerimientos con el principal objetivo de medir y evaluar transformaciones en lenguaje QVT-R, desde el punto de vista del mantenimiento de software [18]. Dicho árbol está compuesto de 3 categorías de más alto nivel las cuales, a su vez, contienen atributos atómicos. Las categorías son *Tamaño*, *Complejidad* y *Reutilización*. En la categoría *Tamaño* se ha definido 3 atributos que pretenden medir la dificultad que conlleva la comprensión y mantenimiento de una transformación, desde el punto de vista del tamaño que presente la misma. Con los atributos *Tamaño de las Relaciones* y *Relaciones de Gran Tamaño* se pretende relevar la dimensión de las componentes más importantes que presenta una transformación, en el caso del lenguaje QVT-R, las relaciones. Otro atributo de esta categoría es *Relaciones Pequeñas*; teniendo en cuenta lo mencionado previamente, un gran número de relaciones pequeñas también incrementa la dificultad al momento de desarrollar las actividades correspondientes al mantenimiento y/o evolución de software.

Por otra parte, también se ha incluido la categoría *Complejidad*. La misma está compuesta de atributos que pretenden medir desde ciertos aspectos sintácticos, la complejidad de una transformación. Un claro ejemplo de estos atributos son *Cláusulas when*, *Cláusulas where* y *Relaciones con OCL*; es decir, si la mayoría de las relaciones poseen dichas cláusulas y/o expresiones OCL, entonces la complejidad para llevar a cabo actividades de mantenimiento y evolución se ve claramente incrementada. Lo mismo sucede si la transformación presenta un alto grado de anidamientos, este aspecto es medido por el atributo *Relaciones Anidadas*. Si la transformación utiliza muchos metamodelos (*Cantidad de Metamodelos*), claramente, esto amplía el espectro de conocimiento que debe poseer el ingeniero de software para poder realizar cambios en el código.

La última categoría se denomina *Reutilización*, la misma posee dos atributos que pretenden reflejar cuan reutilizable es el código que compone la transformación*: Metamodelos Estándares* y *Contexto Abarcado por cada Dominio*. El primero indica si en la transformación se ha utilizado algún metamodelo no estándar, es decir que no ha sido especificado por alguna organización de relevancia internacional (OMG, IEEE, ACM, etc.). El atributo

Contexto Abarcado por cada Dominio pretende establecer el contexto necesario que debe tener el ingeniero de software con respecto a los metamodelos utilizados, a la hora de realizar tareas de mantenimiento y evolución de software. Claramente, si el contexto es grande, esto hace a la transformación menos reutilizable y más difícil de mantener, ya que determinadas tareas implicadas en el mantenimiento y evolución (como la comprensión) consumen más tiempo y esfuerzo.

4.2. ESPECIFICACIÓN DE MÉTRICAS

Una vez definido el árbol de atributos, es necesario establecer las métricas que permitirán cuantificar dichos atributos. Para diseñar una métrica, se debe definir el método de medición y el procedimiento de cómputo, en conjunto con la escala en donde se refleja la medición.

Una *medición* produce una *medida*, es decir, el proceso de medición se define como la actividad que usa una especificación de métrica con el objetivo de producir un valor, resultado de dicha medición; mientras que, una *medida* es el numero o categoría asignada a un atributo de una entidad por medio de una medición [9]. Teniendo esto en cuenta, para diseñar una métrica directa, es necesario especificar claramente un procedimiento de medición y su correspondiente escala. El tipo de la escala depende de la naturaleza de las relaciones entre los valores de la misma. Entre los tipos de escalas más utilizadas en Ingeniería de Software se encuentran: *nominal*, *ordinal*, *intervalos*, *porcentaje*, *absolutas*. Esto es importante, ya que cada tipo de escala, determina el uso de operaciones matemáticas y técnicas estadísticas adecuadas para analizar los datos.

Es importante destacar que a algunas métricas definidas en [17] se le han efectuado leves ajustes con motivos de mejorar la efectividad en las mediciones. Además se ha redefinido la métrica *Relaciones Anidadas* con el principal objetivo de perfeccionar la medición de dicho atributo del árbol.

Por motivos de extensión de este artículo se ha especificado la métrica *Relaciones Anidadas* (Categoría: *Complejidad*). Para más detalles, es posible consultar el reporte interno generado a partir de esta investigación [19] y el trabajo de Abdelahad et al. [17].

ATRIBUTO: Relaciones Anidadas.

DEFINCIÓN: Esta métrica proporciona información sobre la magnitud de los anidamientos respecto de las invocaciones entre relaciones. De manera similar a determinados lenguajes de programación de propósito general, el anidamiento en las construcciones sintácticas de los programas introducen dificultades a la hora de tratar de comprender los mismos. Para esto se construye una estructura con forma arbolada (en ciertas ocasiones puede ser una foresta) extraída de forma estática donde se representan las invocaciones entre relaciones. En ciertas ocasiones, es necesario replicar los nodos ya que sino la estructura podría incluir ciclos. Una vez extraída dicha estructura, se ejecuta el algoritmo 1 que permite computar el grado de anidamiento de una transformación (GAR). La métrica está basada en que el anidamiento de invocaciones entre las relaciones dificulta el entendimiento de las transformaciones, y por ende, las tareas de mantenimiento y evolución.

MÉTRICA INDIRECTA: Grado de Anidamiento de las Relaciones (*GAR*).

Algoritmo 1: Algoritmo para estimar el Grado de Anidamiento entre Relaciones

```
   Input: fir/*Foresta de Invocación a Relaciones*/
   Output: gar /*Grado de Anidamiento entre Relaciones*/
1  puntaje :=0;
2  foreach arbol in fir do
3  |  fordeach nodo in arbol do
4  |  |     ponderacion := PONDERACION (nodo);
5  |  |     puntaje := puntaje + (ponderacion * NUMERO HIJOS(nodo));
   |  end
   end
6  gar := puntaje / NUMERO NODOS INTERNOS(fir)
```

ESCALA NUMÉRICA: Continua. TIPO DE VALOR: Real. TIPO DE ESCALA: Proporción. MÉTRICAS Y ESTRUCTURAS RELACIONADAS: 1) Foresta de Invocación a Relaciones (FIR). 2) Ponderación de cada nodo en el árbol (función PONDERACIÓN).

OBJETIVO: Determinar el grado de profundidad y esparcimiento de los anidamientos de las invocaciones entre las relaciones.

MÉTODO DE CÁLCULO: Especificado en Algoritmo 1.

1) ESTRUCTURA: Foresta de Invocación a Relaciones (*FIR*).

OBJETIVO: contiene las dependencias de invocaciones a relaciones en forma de foresta. Dicha foresta puede contener uno o más árboles de acuerdo a cada transformación.

ESPECIFICACIÓN: se construye el árbol (o la foresta) partiendo de la/las relaciones *top-level* siguiendo las invocaciones entre relaciones. Para que se construya un árbol (o foresta), puede ser necesario replicar los nombres de las relaciones (de otra manera, podrían existir ciclos).

2) ATRIBUTO: Ponderación de cada nodo en el árbol (función *PONDERACIÓN*).

MÉTRICA DIRECTA: Ponderación de nodo.

OBJETIVO: Asignar un valor a un nodo considerando la profundidad del mismo en el árbol.

ESPECIFICACIÓN: la función *PONDERACIÓN* calcula la ponderación del nodo actual sumando los siguientes valores: i) altura del nodo en el árbol + 1 y ii) *ponderación* del nodo padre.

ESCALA NUMÉRICA: Discreta. *TIPO DE VALOR:* Entero. *TIPO DE ESCALA:* Absoluta.

4.3. EVALUACIÓN

Claramente, todo procedimiento de medición, es seguido de uno o varios procesos de evaluación. De manera análoga, las métricas definidas en los apartados precedentes serán útiles si se definen los indicadores adecuados en el proceso de evaluación. Un *indicador* permite especificar como calcular e interpretar los atributos especificados en el árbol en la Figura 1. En general se utilizan dos tipos de indicadores: i) indicadores elementales e ii) indicadores parciales/globales. Los primeros evalúan los requerimientos de más bajo nivel, es decir, los atributos. Cada indicador elemental posee un modelo elemental que proporciona una función que mapea las medidas obtenidas a partir de la métrica (el dominio) a la escala del indicador (el rango).

La nueva escala se interpreta utilizando criterios de decisión, también conocidos como *Niveles de Aceptabilidad*, los cuales ayudan a analizar el nivel de satisfacción alcanzado por cada requerimiento elemental; es decir, por cada atributo. Dichos niveles aportan tanta semántica como los mismos indicadores, ya que permiten determinar los grados de satisfacción del atributo bajo análisis. Para este caso de estudio, se han establecido 3 criterios de aceptabilidad en general: i) *Satisfactorio*; ii) *Marginal*; iii) *Insatisfactorio*; por cada uno de estos se debe especificar un intervalo en el rango [0,100] que permite determinar en que criterio de evaluación se encuentra el valor del indicador elemental. Es relevante remarcar que en esta investigación en particular, se ha tomado un enfoque orientado al mantenimiento. Por lo tanto, si la evaluación de un atributo resulta en un valor que se encuentra en el intervalo asociado al criterio *Satisfactorio*, desde el enfoque evaluado en este trabajo, esto implica que las tareas de mantenimiento no comprenderán grandes esfuerzos tomando en cuenta dicho atributo. Por el contrario, de manera análoga, si resulta en un valor en el intervalo asociado al criterio *Insatisfactorio*, eso implica que las tareas de mantenimiento conllevarán grandes esfuerzos. En esta instancia del trabajo, los intervalos de aceptabilidad para cada indicador han sido definidos considerando distintos criterios de investigadores entendidos en la temática.

Por otra parte, los *indicadores parciales/globales* evalúan requerimientos de mediano o alto nivel, es decir, características o subcaracterísticas (por ejemplo, en el árbol de la Figura 1, la característica *Complejidad*). Para esto es posible utilizar algún modelo de agregación, como por ejemplo LSP (Logic Scoring of Preference) [15, 16]. Un indicador global, representa el grado de satisfacción global del objeto de estudio con respecto a los requerimientos medidos y evaluados. En esta investigación en particular, sólo se utilizan indicadores elementales, es decir, sólo se evalúan los atributos del árbol de requerimientos, para posteriormente llevar a cabo un análisis holístico de los indicadores.

A continuación se muestra el indicador elemental correspondiente a la métrica *Relaciones Anidadas*, definida en el apartado 4.2.

ATRIBUTO: Relaciones Anidadas.

INDICADOR ELEMENTAL:

NOMBRE: Nivel de Desempeño en Grado de Anidamiento en las Relaciones (*D_GAR*).

MOI

$$D\_GAR = \begin{cases} 100 - GAR & \text{GAR} \leq 100 \\ 0 & \text{GAR} \geq 100 \end{cases}$$

ESPECIFICACIÓN: el mapeo es determinado por la función 1: (1)

donde *GAR* es la métrica indirecta definida en el apartado anterior.

CRITERIOS DE DECISIÓN:

CRITERIO	*1:* Insatisfactorio	*2:* Marginal	*3:* Satisfactorio;
RANGO	$0 \leq D\_GAR \leq 60$;	$60 < D\_GAR \leq 80$	$80 < D\_GAR \leq 100$;

ESCALA NUMÉRICA: Continua. TIPO DE VALOR: Real. TIPO DE ESCALA: Proporción.

5. CASO DE ESTUDIO

Para mostrar la aplicabilidad del enfoque desarrollado en las secciones previas, en este apartado se presentan dos casos de estudio en donde se evalúan dos transformaciones. Por un lado, una transformación definida para la empresa especializada en optimización multiobjetivo ESTECO [20] (CE1). Dicha transformación permite la conversión de muchos flujos de trabajo ingenieriles definidos en el formato propietario de ESTECO al estándar de procesos de negocio BPMN2 [10]. Por otro lado, una transformación definida en el campo de la bioinformática [11] (CE2). El objetivo de la misma es automatizar el análisis y simulación de mecanismos de señales de las células (pathways). Un pathway es un conjunto de reacciones químicas que se producen en el interior de una célula tras la recepción de un estímulo. Dicha transformación representa los datos biológicos, que frecuentemente se hace manualmente, en una Red de Petri Coloreada, logrando de esta manera ejecutar dicho modelo en herramientas de simulación.

Tabla 1. Indicadores Elementales para la transformación (valores expresados en %)

Característica / Atributo	I.E CE1	I.E CE2
Tamaño		
Tamaño de las Relaciones	21	38
Relaciones Grandes	78	91
Relaciones Pequeñas	86	100
Complejidad		
Cláusulas When	61	55
Cláusulas Where	81	55
Cantidad de Metamodelos	14	90
Relaciones con OCL	100	18
Relaciones Anidadas	71	87
Reutilización		
Metamodelos Estándares	0	0
Contexto Abarcado por cada Dominio	10	30

De acuerdo a lo establecido en la sección 4, para este estudio, se han utilizado 3 intervalos de aceptabilidad en una escala de porcentaje para los indicadores elementales definidos en el apartado 4.3. Los valores pueden caer en los intervalos: i) Insatisfactorio, indicado con color negro; ii) Marginal, en color blanco; o iii) Satisfactorio determinado por el color gris.

La Tabla 1 muestra los resultados de la evaluación para los casos de estudio. Como puede observarse en relación a la categoría Tamaño se han utilizado 3 métricas: el indicador para el atributo *Tamaño de las Relaciones* posee un valor de 21% y 38% respectivamente. En el CE1 el promedio del tamaño de las relaciones supera el valor que se ha considerado para determinar que una relación es calificada como grande. Esto dificulta el mantenimiento, ya sea correctivo, adaptativo o perfectivo, debido a que si las relaciones poseen muchas líneas de código, el esfuerzo para realizar este tipo de tareas se ve incrementado. Si observamos el resultado con respecto al atributo *Relaciones Grandes*, se puede percibir que en ambos casos las transformaciones no poseen un gran número de relaciones extensas. De manera análoga al atributo *Relaciones Grandes*, ambas transformaciones poseen pocas *Relaciones Pequeñas*. Esto reduce las posibilidades que existan muchas invocaciones a otras relaciones.

Al momento de evaluar la Complejidad de las transformaciones, se pueden analizar desde el punto de vista de ciertos atributos: en este caso en particular, por ejemplo, la cláusula *where* defi-

ne la condición que deben cumplir todos los elementos que intervienen en la relación, y tiene el poder de restringir cualquiera de las variables en la relación y sus dominios. En esta cláusula se pueden encontrar la mayor cantidad de invocaciones a otras relaciones, es por ello que mientras más relaciones posean esta cláusula más difícil será comprender la transformación. En el CE1 se puede observar un 81%, es decir, posee un nivel Satisfactorio. El CE2 posee un 55%, es decir que su nivel de aceptabilidad es Marginal. Por otra lado, es evidente que mientras más metamodelos se utilicen en una transformación, más compleja será, ya que se deben conocer todos los metamodelos que participan en la misma. En los casos de estudio este indicador elemental resulta en 14% y 90% respectivamente. En el primer caso el porcentaje es 14% porque la transformación utiliza 6 metamodelos, en el otro caso el valor es 90% porque sólo utiliza 2 metamodelos. Teniendo en cuenta que la especificación de QVT cuenta con el estándar OCL, las transformaciones suelen poseer expresiones OCL. Si bien OCL ayuda a la hora de construir una transformación, hay que tener en cuenta que mientras más relaciones posean este tipo de expresiones la complejidad aumentará. En el CE1 no se ha encontrado expresiones OCL, por tal motivo el indicador elemental es del 100%. En el CE2 el indicador elemental es de 18%, es decir que el 82% de las relaciones utilizan OCL, esto hace que dicha transformación sea más compleja de mantener. De manera análoga a los lenguajes de programación de propósito general, mientras más anidamientos (*Relaciones Anidadas*) presente la transformación, más difícil de comprender será. En el CE1 se puede observar que el nivel de aceptabilidad es de 71%, es decir Marginal. En el CE2 el nivel de aceptabilidad es de 87%. Es decir, que la transformación no posee un alto grado de anidamiento, lo que facilitará su comprensión.

Desde el punto de vista de la Reutilización se han analizado dos aspectos: es claro que si todos los metamodelos son estándares facilitará la reutilización de la transformación completa o partes de la misma. Por el contrario, si la transformación utiliza al menos un metamodelo que no es estándar, el ingeniero de software puede verse en diferentes tipos de dificultades, como por ejemplo falta de documentación, asistencia por parte de expertos, entre otras. Ambos casos de estudio poseen metamodelos no estándares, es por ello que el nivel de aceptabilidad de los indicadores son Insatisfactorios. Por otra parte, para determinar el contexto abarcado por cada dominio se han considerado todas las clases que participan en una transformación y que pertenecen al dominio origen y dominio destino. Si en la transformación se utilizan todas las clases de los metamodelos que participan en la transformación, más difícil será poder reutilizar la misma. En ambos casos de estudio analizados los indicadores elementales han resultado en Insatisfactorio ya que se utilizan un alto porcentaje de clases sobre el total contenido en los dominios.

6. CONCLUSIONES Y TRABAJOS FUTUROS

Sin lugar a dudas, una de las tareas más complejas y que más tiempo consume en el ciclo de vida de una aplicación es la de mantenimiento [18]. La tendencia al uso de los lenguajes de transformación, indica que este tipo de "programas" no es la excepción en comparación con los lenguajes de propósito general. Las tareas comprendidas dentro del contexto de mantenimiento y evolución de software pueden llegar a ser diversas de acuerdo al objetivo de dicha actividad. En este sentido, es posible que el mantenimiento conlleve corrección de errores, reestructuración, mejora en desempeño, portabilidad o cualquier otro atributo de calidad, entre otros.

En este artículo se definió un conjunto de métricas que posibilitan la medición y evaluación de transformaciones a nivel metamodelo con el objetivo de asistir al ingeniero de software a analizar y estimar esfuerzos para llevar a cabo tareas de mantenimiento y evolución en las mismas.

En primer lugar se definió un conjunto de atributos agrupados en tres categorías diferentes: Tamaño, Complejidad y Reutilización [17]. Luego, por cada atributo: i) se especificó una métrica que permite obtener un valor (medida) del mismo y ii) se definió un indicador elemental que permite obtener información contextual respecto de la medida, es decir, permite evaluar el resultado de medir dicho atributo.

Para probar el enfoque propuesto en este trabajo se emplearon dos casos de estudio. Por un lado una transformación la cual permite la conversión de muchos flujos de trabajo ingenieriles definidos en el formato propietario de ESTECO [20] al estándar de procesos de negocio BPMN2 [10].

Por otro lado, una transformación definida en el campo de la bioinformática [11] la cual representa datos biológicos, que frecuentemente se hace manualmente, en una Red de Petri Coloreada, logrando así ejecutar dicho modelo en herramientas de simulación.

Es posible afirmar que las métricas y los indicadores propuestos brindan información valiosa asistiendo al ingeniero de software a llevar a cabo tareas de mantenimiento en transformaciones a nivel metamodelo. Por medio de estas métricas, como se ha analizado en los casos de estudio, es posible estimar costos de mantenimiento, identificar aspectos específicos a mejorar en las transformaciones, estrategias de refactorización, entre muchos otros aspectos relativos al mantenimiento y evolución de las transformaciones.

Como trabajos futuros se propone direccionar la investigación hacia las siguientes temáticas: i) someter los "componentes" de la estrategia de evaluación a un proceso de mejora continua; es decir, mejorar de manera continua el árbol de atributos, las métricas y los indicadores elementales propuestos, mediante interacción con expertos en la temática; ii) expandir el árbol de atributos abarcando otras características que reflejen aspectos del mantenimiento no contemplados en esta instancia de la investigación; iii) incorporar a la etapa de evaluación, un método de agregación multicriterio para obtener valores a nivel de características [15, 16]; iv) proponer estrategias de extracción automática de métricas como sugiere Kapova et al. [2] en conjunto con la integración en un entorno de desarrollo como Eclipse o Netbeans.

REFERENCIAS

[1] FAVRE, Jean-Marie, ESTUBLIER, Jacky and BLAY, Mireille. Beyond MDA: Model Driven Engineering (L'Ingénierie Dirigée par les Modèles: au-délà du MDA). (2006). Edition Hezmes-Lavoisier. ISBN 2-7462-1213-7.
[2] KAPOVÁ, Lucia, GOLDSCHMIDT, Thomas, BECKER, Steffen and HENSS, Jörg. Evaluating maintainability with code metrics for model-to-model transformations. (2010). En Research into Practice–Reality and Gaps. Springer Berlin Heidelberg: 151-166.
[3] REYNOSO, Luis, GENERO, Marcela and PIATTINI, Mario. Measuring ocl expressions: An approach based on cognitive techniques. (2005). Metrics for Software Conceptual Models.
[4] MOHAGHEGHI, Parastoo and DEHLEN, Vegard. Developing a quality framework for model-driven engineering. (2008). In Models in Software Engineering. Springer Berlin Heidelberg: 275-286.
[5] VAN AMSTEL, Marcel, LANGE, Christian F . and VAN DEN BRAND, Mark. Using metrics for assessing the quality of ASF+ SDF model transformations. (2009), In Theory and Practice of Model Transformations. Springer: 239– 248.
[6] VAN AMSTEL, Marcel, BOSEMS, Stecen, KURTEV, Ivan and PIRES, Luis. Performance in model transformations: experiments with ATL and QVT. (2011). In Theory and Practice of Model Transformations. Springer: 198–212.
[7] VAN DEURSEN, Arie, HEERING, Jan, KLINT, Paul. Language Prototyping: An Algebraic Specification Approach. (1996). Vol. V. World Scientific.
[8] JOUAULT, Frédéric and KURTEV, Ivan. Transforming models with ATL. (2006). En satellite events at the MoDELS 2005 Conference. Springer Berlin Heidelberg: 128-138.
[9] OLSINA, Luis, LEW, Philip , DIESER, Alexander and RIVERA, Belen. Updating Quality Models for Evaluating New Generation Web Applications. (2012). Journal of Web Engineering, 11(3): 209–246.
[10] ABDELAHAD, Corina. Transformación de Workflows Científicos a Modelos de Negocio en BPMN2 utilizando QVT. (2013) Master's thesis, UNSL.
[11] GÓMEZ LLANA, Abel. Recuperación, transformación y simulación de datos biológicos mediante ingeniería dirigida por modelos. Transformación Transpath2Cpn para MediniQVT. Universidad Politécnica de Valencia. Technical report.
[12] GIANDINI, Roxana S., PONS, Claudia y PÉREZ, Gabriela. A two-level formal semantics for the QVT language. (2009). CIbSE vol. 9: 73-86.
[13] OMG.Modeling and meta data specifications. http://www.omg.org/spec, 2015.
[14] QVT. http://www.omg.org/spec/QVT/1.1, 2014.
[15] DUJMOVIĆ, Jozo and KADASTER, Metin. A technique and tool for software evaluation. (2002). Evolution, vol. 374: 246.
[16] MIRANDA, Enrique, BERON, Mario, MONTEJANO, German, VARANDA PEREIRA, Maria and HENRIQUES, Pedro. NESSy: a New Evaluator for Software Development Tools. (2013). In 2nd Symposium on Languages, Applications and Technologies, volume 29 of OpenAccess Series in Informatics(OASIcs), Dagstuhl,-German. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik: 21–37.
[17] ABDELAHAD, Corina, MIRANDA Enrique, y PEREZ, Norma . Medición y Evaluación de Transformaciones a Nivel Metamodelo: un Enfoque Orientado al Mantenimiento. (2014). Congreso Nacional de Ingeniería Informática y Sistemas de Información – CoNaIISI: 1083-1094.
[18] BENNETT, Keith H. and RAJLICH, Václav T. Software maintenance and evolution: a roadmap. (2000). En Proceedings of the Conference on the Future of Software Engineering. ACM: 73-87.
[19] ABDELAHAD, Corina, MIRANDA, Enrique y PEREZ, Norma. Medición y Evaluación de una Transformación en QVT- R para ESTECO. (2015). Technical report.
[20] ESTECO. http://www.esteco.com, 2015.