Medidas para un Lenguaje Visual basado en Transformaciones de Grafos (eMotions)

Manuel F. Bertoa, Fernando López y Antonio Vallecillo

GISUM/Atenea Research Group. Universidad de Málaga (Spain).
{bertoa,fernando,av}@lcc.uma.es

Abstract. El presente trabajo presenta un conjunto de medidas para un lenguaje visual eMotions, cuyo objetivo es valorar la calidad de los modelos construidos con este lenguaje. Estas métricas analizan tres factores relevantes: 1) Medidas de tamaño del modelo y de las reglas; 2) Acoplamiento entre reglas; y, 3) Complejidad de las expresiones OCL utilizadas. En este trabajo preliminar presentamos la suite de métricas propuestas quedando como trabajo futuro la definición de indicadores y la validación de las mismas. Las medidas básicas o simples, se extraen mediante un conjunto de transformaciones ATL (Atlas Transformation Language), generando unos archivos con formato separados por comas (csv) que alimentan una base datos. Las medidas derivadas, se calculan sobre los datos almacenados en esta base de datos.

1 Introducción

Los lenguajes visuales de dominio específico (LVDE) [1] han comenzado a proliferar tan pronto como las prácticas de Desarrollo de Software Dirigido por Modelos (DSDM) están siendo adoptadas por muchas empresas y organizaciones. Este tipo de lenguajes permiten especificar modelos del sistema a un nivel de abstracción adecuado y usando notaciones que resultan familiares a los expertos del dominio.

Actualmente existen diferentes tipos de LVDE, dependiendo fundamentalmente de cómo especifican el comportamiento de los sistemas que modelan [10]. Una de las formas más usuales de describir el comportamiento de los modelos es mediante transformaciones de grafos. En este enfoque, los modelos son grafos, y las acciones se modelan mediante reglas, que especifican el *antes* y el *después* de cada acción. Esta forma es bastante intuitiva y se utiliza en muchos de los lenguajes de programación visuales para niños como por ejemplo Webcast [12].

Aunque modelar con este tipo de lenguajes no es complejo, los modelos producidos han de estar sujetos a controles y medidas de calidad que permitan evaluarlos, como sucede con el resto de los artefactos que forman parte de la Ingeniería del Software Dirigida por Modelos. De hecho, nuestra experiencia muestra que las formas en las que dos personas modelan un mismo sistema usando un LVDE suelen diferir notablemente. Y cada forma es mejor dependiendo de un aspecto de calidad concreto: usabilidad, mantenibilidad, eficiencia, etc.

Si bien es cierto que existen algunos trabajos sobre la evaluación de la calidad de lenguajes visuales de programación (p.ej. [11]), por un lado suelen ser muy generales

y por otro no contemplan los lenguajes de modelado sino solo los de programación. Hasta donde llega nuestro conocimiento no existen trabajos específicos que propongan métricas para la evaluación de los lenguajes visuales de modelado basado en transformaciones de grafos aunque sí existen propuestas de métricas para ATL o para expresiones OCL que se comentan en el apartado 6 de Trabajos relacionados. Por ello, el presente trabajo presenta un conjunto de medidas para este tipo de lenguajes, cuyo objetivo es valorar la calidad (y especialmente la usabilidad, mantenibilidad y rendimiento) de los modelos construidos con ellos. Estas métricas analizan tres factores relevantes: 1) Medidas de tamaño del modelo y de las reglas; 2) Acoplamiento entre reglas; y, 3) Complejidad de las expresiones OCL utilizadas. Las medidas de tamaño nos dan una indicación de la cantidad de información que presenta el modelo y nos permitirá relativizar otras medidas o indicadores. Las medidas de acoplamiento son una indicación de la complejidad del modelo, derivada de la dependencia entre las reglas, que hará más difícil su compresión y utilización. Las medidas de complejidad de las expresiones ofrecen una valoración de la complejidad añadida que puede tener un modelo según el tipo de expresiones OCL que utilice. En los siguientes apartados vamos a comentar brevemente las medidas propuestas para cada uno de estos aspectos. Por concreción, en este trabajo nos centraremos en un lenguaje concreto, *e-Motions*.

Señalaremos también que en la especificación de cualquier modelo siguiendo este enfoque es preciso definir dos partes bien diferenciadas: el modelo estructural y el de comportamiento. El primero define las entidades y relaciones entre ellas, usando un metamodelo. El segundo define el comportamiento de las entidades descritas en el metamodelo estructural, mediante un conjunto de reglas. En este trabajo nos centramos en la parte de comportamiento, ya que para el metamodelo que define la estructura de los modelos se pueden aplicar las medidas existentes en la literatura o las medidas clásicas de la programación orientada a objetos como por ejemplo las de Chidamber and Kemerer [2] o MOOD [3].

2 Un Lenguaje visual de Transformaciones de grafos

eMotions [4] es un lenguaje de la familia de LVDEs basados en transformaciones de grafos. Cuenta además con una herramienta que permite especificar, simular y analizar sistemas en tiempo real de forma precisa e intuitiva. *eMotions* ha sido desarrollado para la plataforma Eclipse. El comportamiento dinámico de un Lenguaje de Modelado de Dominio Específico (LMDE) se especifica mediante transformaciones *in-place*.

Las transformaciones *in-place* están formadas por un conjunto de reglas, cada una de las cuales representa una posible acción del sistema. Estas reglas son de la forma $l:[NAC]^* \times LHS \rightarrow RHS$, donde *l* es el nombre de la regla y LHS (left-hand side), RHS (right-hand side) y NAC (negative application conditions) son patrones de modelos que representan ciertos estados del sistema. El conjunto de plantillas formadas por LHS y NAC son las precondiciones para que la regla sea aplicada, mientras que RHS representa el resultado de que se lleve a cabo la acción. Una regla

se aplica si en el modelo existe una coincidencia con el patrón LHS y no aparece ninguna con los posibles NAC que tenga dicha regla. En el caso de que aparezcan varios patrones coincidentes con LHS se selecciona uno aleatoriamente y se aplica la regla sobre él, de modo que dicho patrón es sustituido por una instancia del modelo representado en RHS. Las reglas se aplican en un orden no determinístico hasta que ninguna es aplicable, aunque este comportamiento puede ser modificado mediante algunos mecanismos de control de ejecución como por ejemplo estrategias [5].

eMotions hace uso de una extensión de transformaciones de modelo *in-place* que incluye un modelo de comportamiento temporal y un mecanismo para declarar propiedades de las propias acciones representadas por las reglas, tratándose éstas como objetos de primera instancia. Es importante resaltar que *eMotions* soporta el uso de expresiones OCL a través de mOdCL [6], el cual implementa y da semántica a OCL en Maude [5]. Además de un entorno gráfico de modelado de comportamiento, *eMotions* permite generar de forma automatizada (mediante transformaciones implementadas en ATL[7]) las correspondientes especificaciones en Maude del modelo diseñado. Maude se utiliza como una notación formal para dar una semántica precisa a las especificaciones de *eMotions* [8]. De este modo, se podrán realizar tareas de análisis formal y simulación a partir de las especificaciones [9].

Contamos con dos tipos de reglas para especificar el comportamiento de las acciones en el tiempo: reglas *Atomic* y reglas *Ongoing*. Las reglas *Atomic* representan acciones atómicas, con una duración específica y además pueden ser periódicas. Las reglas *Ongoing* representan acciones que se realizan de forma continuada en el tiempo mientras continúen cumpliéndose las precondiciones (que se cumpla LHS y no se cumplan los NACs). Para ambos tipos de regla puede establecerse el intervalo de tiempo en que deben aplicarse.

eMotions facilita la representación de forma explícita de las acciones que representan las reglas. Incluyendo un objeto denominado `Action Execution` podemos especificar un tipo de acción concreta y el conjunto de objetos involucrados en la misma. Este mecanismo es útil para comprobar si un objeto participa en una acción o si una acción ya ha sido ejecutada.

Una descripción más amplia del lenguaje *eMotions* se realiza en el informe técnico [17]; otros recursos sobre *eMotions* están disponibles en la web de nuestro grupo en el enlace http://atenea.lcc.uma.es/index.php/Main_Page/Resources/E-motions.

3 Medidas de tamaño

Las medidas de tamaño están orientadas a medir la cantidad de los distintos elementos que aparecen en el modelo de comportamiento. Estas medidas pueden servir como una primera aproximación a la complejidad del modelo y para poder relativizar otras medidas al comparar distintos modelos o distintas soluciones de modelado. Las medidas base de tamaño cuentan el número de elementos de cada categoría que pueden aparecer en el modelo, se resumen en las siguientes tablas.

Es importante recordar que una regla tiene una parte izquierda (LHS), una parte derecha (RHS) y puede tener cero o varios NACs. Cada una de estas partes tiene

objetos, condiciones, etc. Por tanto, al nivel más bajo medimos cuántos elementos hay en la parte izquierda, cuántos en la derecha y cuántos en los NACS. Vamos a detallarlo para los objetos, siendo el enfoque similar para el resto de elementos. A nivel de regla mediremos la suma de objetos de su parte izquierda, de su parte derecha y de sus posibles NACs. Por último, al nivel del modelo podemos agregar todos los objetos de todas las reglas y medir el número de objetos de todo el modelo (ver Tabla 2).

Tabla 1. Medidas base para Objetos

Abreviatura	Nombre	Descripción
NObj	Nº Objetos del modelo.	Suma de los objetos en todas las reglas (r_i) del modelo. $NObj = \Sigma (NObj(r_i))$
$NObj(r_i)$	Nº de Objetos en la regla r_i.	Será la suma de los objetos en su parte izquierda, derecha y NACs. $NObjLHS(r_i) + NObjRHS(r_i) + NObjNAC(r_i)$
$NObjLHS(r_i)$	Nº Objetos en LHS	Nº Objetos en la parte izquierda de la regla r_i
$NObjRHS(r_i)$	Nº Objetos en RHS	Nº Objetos en la parte derecha de la regla r_i
$NObjNAC(r_i)$	Nº Objetos en las NACs	Nº Objetos en las posibles NAC de la regla r_i. Si tiene más de una NAC, esta medida es la suma de objetos en todas sus NACs.
NObjLHS	Nº total de objetos en LHS	$\Sigma (NObjLHS(r_i))$
NObjRHS	Nº total de objetos en RHS	$\Sigma (NObjRHS(r_i))$
NObjNAC	Nº total de objetos en NACs	$\Sigma (NObjNAC(r_i))$

Al igual que para los objetos, las medidas anteriores se realizan para todos los elementos que pueden aparecer en una regla: objetos, atributos (slots), acciones, enlaces (links), condiciones y *object role*.

Por último, otro aspecto que puede incrementar la complejidad del modelo es el número de objetos modificados en las reglas, bien porque se crean, se destruyen o se modifican sus atributos (slots). Esta medida también la hacemos a nivel de cada regla y para todo el modelo. En la Tabla 2, resumimos las medidas base que se proponen a nivel de modelo, que serán una agregación de dichas medidas a nivel de regla o inferior.

Tabla 2. Medidas Base de Tamaño del modelo

Abreviatura	Descripción	Relaciones
NRTot	Nº de reglas (totales) en el modelo	Las reglas pueden ser atómicas u ongoing. NRTot = NRAtm + NROng
NRAtm	Nº de reglas atómicas	
NROng	Nº de reglas ongoing	
NHlp	Nº de Helpers del modelo.	Los helpers se definen a nivel del modelo, fuera de cualquier regla.
NVbl	Nº de variables del modelo	$\Sigma (NVbl(r_i))$

NNAC	Nº de NACs en el modelo. Una regla puede tener 0 o varios NACs.	Σ (NNAC(r_i)) siendo r cada una de las reglas del modelo
NObj	Nº Objetos del modelo. Suma de los objetos en todas las reglas (r_i) del modelo.	Σ (NObj(r_i))
NLnk	Nº de enlaces (link) en el modelo	Σ (NLink(r_i))
NCnd	Nº de condiciones en el modelo	Σ (NCond(r_i))
NAct	Nº de acciones del modelo	Σ (NAcc(r_i))
NObjRol	Nº Object Role del Modelo	Σ (NObjRol(r_i))
NSlt	Nº de Slots (atributos) de todos los objetos que aparecen en el modelo. Siendo los slots de cada objeto que aparece en cada regla del modelo.	Σ (NSlot(r_i)), siendo NSlot(r_i) = Σ (NSlot(O_j)) donde O_j es el objeto j de la regla r_i
NObjCrea	Nº de objetos creados	Objetos que no están en el LHS pero aparecen en RHS
NObjDest	Nº de objetos destruidos	Objetos que están en LHS pero no aparecen en RHS

A partir de estas medidas podemos ir definiendo y validando diversas medidas derivadas o indicadores. Entre otros hemos definidos los que aparecen en la Tabla 3.

Tabla 3. Medidas derivadas de Tamaño

Abreviatura	Descripción	Relaciones
ILRObj	Índice L-R de Objetos	NObjLHS / (NObjLHS + NObjRHS)
ILRLnk	Índice L-R de Links	NLnkLHS / (NLnkLHS + NLnkRHS)
ILRAct	Índice L-R de Acciones	NAccLHS / (NAccLHS + NAccRHS)
ISlt	Índice de Slots por objeto	NSlt / NObj
IHelp	Índice de helpers por reglas	NHlp /NRT
IVbl	Índice de variables por reglas	NVbl /NRT
IObj	Índice de objetos por reglas	NObj /NRT
INac	Índice de NACs por reglas	NNac /NRT
ILnk	Índice de links por reglas	NLnk /NRT
ICnd	Índice de condiciones por reglas	NCnd /NRT
IAct	Índice de acciones por reglas	NAcc /NRT
IObjRol	Índice de object roles por reglas	NObjRol /NRT
ICrea	Índice de Creación	NObjC /NRT
IDest	Índice de destrucción	NObjD / NRT

4 Medidas de Acoplamiento

En modelos complejos, una característica que añade una alta complejidad es la utilización en una regla de referencias a reglas del modelo. Este concepto es cercano al de acoplamiento en otros ámbitos de la informática que en estos tipos de modelos

se da con algunas características específicas. De forma no rigurosa, entendemos que una regla presenta acoplamiento cuando referencia o es referenciada por otra regla. Dada las características de este lenguaje, estas referencias pueden aparecer de tres formas diferentes dentro una regla: (1) La regla interrumpe a otra; (2) La regla depende de otra; y, (3) La regla excluye a otra.

Evidentemente, este acoplamiento puede ir en dos direcciones y nos referiremos al Acoplamiento Eferente (hacia fuera) y Acoplamiento Aferente (que llega o hacia dentro) y definimos las medidas de acoplamiento entre reglas que detallamos a continuación. Hay que hacer notar que en modelos simples es posible que este acoplamiento no exista siendo todas sus reglas independientes.

4.1 Acoplamiento Eferente (hacia fuera) – AE

Estará compuesto por tres medidas según hemos comentado anteriormente: RIe, Reglas interrumpidas por una regla; RDe, Reglas de las que depende una regla; y REe, Reglas que excluye una regla. De forma global, el AE será la suma ponderada de estas tres medidas:

$$AE = w_i RIe + w_d RDe + w_e REe$$

4.2 Acoplamiento Aferente (que llega) – AA

Similarmente al anterior estará compuesto por tres medidas: RIa, Reglas que interrumpen a esta regla; RDa, Reglas que dependen de esta regla; y, REa, Reglas que excluyen a esta regla. Por tanto, AA será la suma ponderada de estas tres medidas:

$$AA = w_i RIa + w_d RDa + w_e REa \quad (1)$$

4.3 Inestabilidad de las reglas - In

Vamos a definir la Inestabilidad de una regla como un ratio entre el acoplamiento eferente y el aferente según la siguiente fórmula: In = AE / (AE + AA). Esta medida nos permite clasificar las reglas en cuatro categorías:

1. **Regla Independiente**, no referencia ni es referenciada por nadie, estas reglas deben cumplir que AE + AA = 0, y por tanto su Inestabilidad quedaría indefinida.
2. **Regla Eferente**, son reglas que referencia o dependen de otras reglas pero ninguna regla la referencia a ella. Por tanto cumplen la condición siguiente: AE > 0 y AA = 0 => In = 1.
3. **Regla Aferente**, son reglas que no referencian a ninguna regla externa pero son referenciadas por otras reglas, cumplirán que: AE = 0 y AA > 0 => In = 0.
4. **Regla Biacoplada**, son reglas que referencias a otras reglas y que además son referenciadas por otras reglas. Por tanto cumplen la siguiente condición: AE > 0 y AA>0 => 0 < In < 1.

Los tres tipos de reglas no independientes se pueden agregar como **Reglas Acopladas,** que serán las que cumplen: AE + AA > 0.

4.4 Medidas de Acoplamiento entre reglas

A partir de estas definiciones podemos definir un conjunto de medidas derivadas de acoplamiento entre reglas que resumimos en la Tabla 4.

Tabla 4. Medidas de acoplamiento entre reglas

Abreviatura	Descripción	Relaciones
NRInd	Nº Reglas Independientes	Cada r cumple: AE + AA = 0
NRCoup	Nº Reglas Acopladas	Cada r cumple: AE + AA > 0
PRCoup	Porcentaje Reglas Acopladas	NRCoup / (NRCoup+NRInd)
NREf	Nº Reglas Eferentes	Cada r cumple: IN=1
NRAf	Nº Reglas Aferentes	Cada r cumple: IN=0
NRBi	Nº Reglas Biacopladas	Cada r cumple: 0 < IN < 1
IEfC	Índice de Acoplamiento Eferente	NREf / NRCoup
IAfC	Índice de Acoplamiento Aferente	NRAf / NRCoup
IBiC	Índice de Acoplamiento Biacoplado	NRBi / NRCoup
MRIns	Inestabilidad media de las Reglas	Promedio(In) de las reglas acopladas
ModelIns	Inestabilidad del Modelo	Σ (In) de las reglas acopladas

5 Medidas de Complejidad de las expresiones OCL

Las expresiones OCL pueden aparecer en distintas partes del modelo: condiciones, helpers, reglas, atributos de los objetos. Estas expresiones añaden complejidad y, por tanto, dificultad de comprensión del modelo y del procesamiento del mismo. Es por ello, que hemos considerado una parte importante de la valoración de la calidad del modelo evaluar de alguna forma la utilización de las expresiones OCL atendiendo a la propia complejidad de las expresiones.

Actualmente, simplemente clasificamos cada expresión según un tipo o grado de complejidad en función de los operadores que utiliza. En futuros trabajos esperamos asignar una escala de valores más compleja y detallada a cada expresión OCL.

De esta forma, además del nº total de expresiones OCL (no triviales) en el modelo, podemos tener una primera indicación de la complejidad que estas expresiones añaden al modelo. No es lo mismo, tener una expresión que realice una simple operación aritmética que una expresión que utilice un operador *forAll* que realizará una operación para cada elemento de una colección.

Una forma de clasificar las expresiones de forma simple y objetiva es según los operadores que aparezcan en ella, asignándole un tipo según el operador más complejo que aparezca en cada una de ellas. Los operadores y su clasificación según su complejidad aparecen en la Tabla 5.

Hay muchas expresiones en los modelos (por ejemplo, al asignar un valor a un atributo) que son literales y que no aportan complejidad al modelo. Estas expresiones

triviales las hemos clasificado como de Tipo 0 y no las tendremos en cuenta en los indicadores de complejidad de las expresiones OCL.

Tabla 5. Tipos de operadores OCL de acuerdo a la complejidad que implican

Tipo 0 Literal o Trivial	
Expresión literal	sin operadores
Tipo 1 Básica	
Aritméticos	+, -, *, /
Funciones básicas	abs, floor, div, mod, max, min, …
Funciones para cadenas	Size, concat, substring, toInteger, toReal, …
Operadores de comparación	< , >, <=, >=, <>, =
Tipo 2. Condicional	
Expresión condicional	If-then-else
Tipo 3. Colección	
Queries	size, isEmpty, notEmpty, count, sum, product
Tipo 4. Iterativa	
Expresión iterativa	forAll , exists, iterate, collect, select, reject,
Colecciones	includes, excludes, includesAll, excludesAll, union, intersection, including, excluding, symmetricDifference
Cambios de tipo	flatten, asSet, asBag, asSequence, asOrderedSet
Queries	isUnique, any, one
Tipo aumentado (+1)	
Operadores lógicos	and, or, xor, not, implies

Tipo 1. Básica. Son fundamentalmente expresiones para tipos básicos (Integer, Real, Boolean, String) con operadores aritméticos, funciones básicas o para cadenas y con operadores de comparación.

Tipo 2. Condicionales (IfExp). Tienen un operador if-then-else

Tipo 3. Colecciones. (Sets, OrderedSets, Bags, Sequences). Implican tipos de colecciones y operaciones genéricas de colecciones o específicas de cada tipo concreto de colección, todas tendrán un operador ->.

Tipo 4. Iterativa (Todos los elementos). Una expresión iterativa (IterateExp) es una expresión que se evalúa para cada elemento de una colección.

Tipo aumentado (Operadores Lógicos). De alguna forma queremos representar la complejidad añadida de tener expresiones con operadores lógicos, ya que realmente lo que suelen representar estos operadores es la agregación de varias expresiones. Por ello, si aparece al menos un operador lógico, incrementamos en uno el Tipo de complejidad de esa expresión OCL.

5.1 Medidas de Complejidad OCL del Modelo

A nivel del modelo, hemos propuesto tres indicadores de la complejidad OCL del modelo (MOclC):

(1) Tamaño (*size*) de MOCLC (SMOclC): mide el tamaño ponderado de las expresiones OCL dentro del modelo.

Sea *NTi* el número de expresiones OCL de Tipo *i* que aparecen en el modelo y *Wti*, el coeficiente de ponderación de complejidad de las expresiones OCL de Tipo *i*. Definimos como SMOclC, al tamaño de las expresiones OCL del modelo según la siguiente expresión:

$$SMOclC = \Sigma(Wti * NTi)$$

Los valores de los coeficientes utilizados son los siguientes:

Expresión	Tipo 1	Tipo 2	Tipo 3	Tipo 4	Tipo 5
	0,04	0,16	0,36	0,64	1,00

O, expresado de otra forma:

$$SMOclC = 0{,}04 \cdot NT_1 + 0{,}16 \cdot NT_2 + 0{,}36 \cdot NT_3 + 0{,}64 \cdot NT_4 + 1{,}00 \cdot NT_5$$

(2) Ratio de MoclC (RMoclC). Para dar una medida de la complejidad relativa, definiremos una ratio, dividiendo SMOclC por el número total de las expresiones en el modelo. Definimos el Ratio de la Complejidad Ocl del Modelo como

$$RMOclC = SMOclC / \Sigma(NTi)$$

(3) Nivel de complejidad Ocl (OclCL). Por último, vamos a proponer un indicador global que tenga en cuenta el tamaño (SMOclC) y el Tipo de complejidad del modelo (RMOclC), así definimos el nivel de complejidad Ocl (OclCL) como:

$$OclCL = SMOclC * RMOclC$$

Para mostrar cómo se comportan estos indicadores, hemos realizado unos ejemplos para 60 y 30 expresiones con diferentes distribuciones en cuanto al número de expresiones de cada tipo que reflejamos en la tabla 6.

Tabla 6. Distribución de frecuencias de los operadores

Nº Expresiones OCL					
Tipo 1	0	12	30	6	0
Tipo 2	0	12	20	12	0
Tipo 3	10	12	10	24	5
Tipo 4	20	12	0	12	10
Tipo 5	30	12	0	6	15
Resultados					
Σ NTi	20	40	20	40	20
SMOCLC	46.40	26.40	8.00	24.48	23.20
RMOCLC	0.77	0.44	0.13	0.41	0.77
CL	35.88	11.62	1.07	9.99	17.94

6 Trabajos relacionados

En el ámbito de las transformaciones de modelos, principalmente con ATL, exsiten algunos trabajos similares a nuestra propuesta, es interesante las propuestas de van Amstel et al. [13] proponiendo métricas para las transformaciones de modelos o el informe técnico de Vignaga [14] donde propone 81 medias para transformaciones ATL ampliando la propuesta anterior de van Amstel con medidas de grano más fino.

Respecto a medición de expresiones OCL, hay trabajos como Cabot y Teniente [15] donde evalúan la complejidad de las expresiones OCL con un algoritmo que da una horquilla de valores en función de las instancias de las clases involucradas en las expresiones. En nuestro caso, nos interesaba más conseguir un valor concreto y una aproximación más simple de la posible complejidad de la expresión evaluada de forma estática. Cómo trabajo futuro tenemos previsto estudiar la posible integración de esta propuesta en nuestra herramienta. Otro trabajo que propone medidas para OCL es el de Reynoso et al. [16] donde como parte inicial de su trabajo proponen un conjunto de medidas para las expresiones OCL utilizadas en modelos UML/OCL.

7 Conclusiones y trabajos futuros

Actualmente, las medidas base se obtienen mediante transformaciones ATL que generan un conjunto de ficheros separados por comas (csv) que posteriormente se cargan en una base de datos MySql. Una vez en la base de datos, estas medidas se pueden procesar fácilmente para obtener el conjunto de medidas derivadas que se proponen en este trabajo u otras que se vean de interés en el futuro. El próximo trabajo a realizar en este sentido es obtener una interfaz más amigable y una automatización de los pasos de obtención y carga de las medidas base.

Por otro lado, y quizás el paso más importante, es validar las medidas propuestas demostrando que miden algunas características de calidad de los modelos de calidad del software existentes o de algún modelo de calidad que se cree específico para los lenguajes de transformaciones. En este sentido, podíamos haber empezado proponiendo un modelo de calidad y, posteriormente, definir medidas para las características que aparezcan en dicho modelo de calidad, es decir, hacer una definición de arriba abajo. Sin embargo, hemos decido empezar por analizar "qué" podemos medir en los modelos generados con eMotions y hacer una propuesta de abajo arriba. A veces, es fácil proponer características de calidad que luego quedan vacías de contenido al ser casi imposible medir adecuadamente algo que pueda dar una valoración de esas características. Ejemplo de este problema aparece hasta en los propios estándares de calidad de ISO. Es por ello que hemos decido ver qué medidas son reales y posibles y, en siguientes pasos, analizar qué características de calidad están relacionados o pueden ser evaluadas con las medidas que proponemos.

En este sentido hemos empezado a realizar una serie de experimentos, donde a partir de una misma propuesta varios sujetos generan modelos que simulan lo mismo pero con distintas soluciones. También estamos analizando un conjunto de simulaciones que nuestro grupo ha realizado para evaluar si alguna de las medidas resalta alguna propiedad relevante de estos modelos. Otro paso que tenemos previsto realizar es generalizar las medidas propuestas para el entorno de los lenguajes visuales, sin centrarnos en uno en concreto.

Agradecimientos. Este trabajo ha sido financiado por los siguientes proyectos de investigación: MDD-MERTS (TIN2008-03107) y MOVIS (P07-TIC-03184).

8 Bibliografía

[1] De Lara, J, Vangheluwe, H.: Defining visual notations and their manipulation through meta-modelling and graph transformation, *Journal of Visual Languages & Computing*, 15(3-4):309-330, June-August 2004

[2] Chidamber, S. R. and Kemerer, C. F., A Metrics Suite for Object Oriented Design, *IEEE Transactions on Software Engineering*, vol. 20, 1994.

[3] Abreu, F. B. Melo, W., Evaluating the Impact of OO Design on Software Quality, Third International Software Metrics Symposium, Berlin, 1996.

[4] Rivera, J.E., Durán, F., Vallecillo, A.: A graphical approach for modeling time-dependent behavior of DSLs. In: Proc. of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'09), IEEE Computer Society (September 2009) 51–55.

[5] Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: *All About Maude – A High-Performance Logical Framework*. Volume 4350 of LNCS. Springer (2007)

[6] Roldán, M., Durán, F.: Representing UML models in mOdCL. Available at http://maude.lcc.uma.es/mOdCL (2008)

[7] Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: ATL: A model transformation tool. *Science of Computer Programming* 72(1-2) (2008) 31–39

[8] Rivera, J.E., Durán, F., Vallecillo, A.: On the behavioral semantics of real-time domain specific visual languages. In Ölveczky, P.C., ed.: Proc. of the 8th International Workshop on Rewriting Logic and its Applications (WRLA'10). LNCS, Springer (March 2010)

[9] Rivera, J.E., Vallecillo, A., Durán, F.: Formal specification and analysis of domain specific languages using Maude. *Simulation: Transactions of the Society for Modeling and Simulation International* 85(11/12) (2009) 778–792

[10] Clark, T., Sammut, P., Willans, J. *Applied Metamodelling*. Ceteva, 2 edition, 2004. http://itcentre.tvu.ac.uk/~clark/docs/Applied%20Metamodelling%20%28Second%20Edition%29.pdf

[11] Green T. R. G.a and Petre M.: Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework. *Journal of Visual Languages & Computing* 7(2) (1996) 131-174

[12] Stagecast http://www.stagecast.com/

[13] M.F. van Amstel, C.F.J. Lange, M.G.J. van den Brand. Metrics for Analyzing the Quality of Model Transformations. In Informal Pre-proceedings of the Seventh Belgian-Netherlands Software Evolution Workshop (BENEVOL 2008), pages 36-37, Eindhoven, The Netherlands, December 2008.

[14] A. Vignaga. Metrics for Measuring ATL Model Transformations. Technical Report TR/DCC-2009-6, Computer Science Department, Universidad de Chile, 2009.

[15] Jordi Cabot y Ernest Teniente. A metric for measuring the complexity of OCL expressions. Model Size Metrics Workshop (co-located with MODELS 2006).

[16] Luis Reynoso, Marcela Genero, Mario Piattini, y Esperanza Manso. Assessing the Impact of Coupling on the Understandability and Modifiability of OCL Expressions within UML/OCL Combined Models. In Proceedings of the 11th IEEE International Software Metrics Symposium (METRICS '05). 2005.

[17] Jose E. Rivera, Francisco Durán and Antonio Vallecillo. e-Motions: A Graphical Approach for Modeling Time-Dependent Behavior of Domain Specific Languages. Technical report available at: http://atenea.lcc.uma.es/images/a/a4/E-Motions.pdf