Marco de Trabajo basado en MDA para la Medición Genérica del Software

Beatriz Mora, Félix García,
Francisco Ruiz, Mario Piattini
Dep. de Tecnología y Sistemas de Información
Escuela Superior de Informática de Ciudad Real
Universidad de Castilla-La Mancha
{Beatriz.Mora | Felix.Garcia | Francisco.RuizG |
Mario.Piattini}@uclm.es

Artur Boronat, Abel Gómez,
José Á. Carsí, Isidro Ramos
Dep. de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
{aboronat | agomez | pcarsi | iramos}@dsic.upv.es

Resumen

Actualmente, con el objetivo de obtener productos software de calidad es necesario llevar a cabo una buena gestión de los procesos software donde la medición de los procesos se convierte en un factor fundamental. Debido a la gran variedad de entidades candidatas para la medición, se considera necesario un marco consistente para integrar la medición de los distintos tipos de entidades. En este trabajo se presenta la propuesta de un entorno genérico para la medición de cualquier entidad software a partir de los metamodelos que las representan. Partiendo de un metamodelo común de medición, y mediante transformaciones QVT, se puede llevar a cabo la medición de un modelo de dominio cualquiera. En el trabajo se explica como se ha llevado a cabo la propuesta: por un lado, se ha trabajado con la herramienta MOMENT, que proporciona el soporte necesario para la gestión automática de modelos de acuerdo a MDE y a la arquitectura MDA, por otro lado, se ha adaptado FMESP a MDA. FMESP es un marco de trabajo para la integración del modelado y de la medición de procesos software, que sirve de base conceptual y tecnológica para su mejora. Además, se muestran las etapas a seguir para conseguir la medición genérica basada en MDA, y un caso de ejemplo en el dominio de bases de datos relacionales.

Palabras clave: Medición, MDA, MOMENT.

1. Introducción

La actual necesidad de la industria del software por mejorar su competitividad fuerza a la búsqueda de la mejora continua de sus procesos. Para conseguirlo, es necesaria una gestión exitosa de dichos procesos [10], lo que implica su correcta definición, ejecución, medición, control y mejora. Entre estas fases del ciclo de vida de los procesos destaca la medición, que ayuda a controlar los errores y carencias dentro del desarrollo y mantenimiento del software facilitando la toma de decisiones. De hecho, la medición se ha convertido en un aspecto fundamental de la Ingeniería del Software [8].

Los procesos software constituyen la base a partir de la cual se realiza el trabajo dentro de una organización software. Dichos procesos se aplican en la práctica en forma de proyectos. Como resultado de la ejecución de proyectos concretos se obtienen productos. Por lo tanto, para facilitar y promover la mejora continua de sus procesos, las empresas requieren llevar a cabo la medición del software de manera efectiva y consistente. Esto implica la necesidad de una disciplina para la medición y análisis de datos [5] y la definición, recopilación y análisis de medidas sobre el propio proceso, los proyectos y los productos software.

La gran variedad de tipos de entidades y atributos que son candidatos para la medición motiva la necesidad de disponer de modelos de medición homogéneos, que puedan gestionarse por las empresas de la misma forma, independientemente de cual sea la entidad a

medir. Esto implica la necesidad de una referencia consistente y adecuada para la definición de sus modelos de medición del software así como el soporte tecnológico necesario para integrar la medición de los diferentes tipos de entidades.

Con el fin de satisfacer las necesidades expuestas, es muy útil considerar el paradigma MDE (*Model-Driven Engineering*) [3], de especial importancia en la actualidad. La filosofía de este enfoque consiste en que el desarrollo de software es dirigido por los modelos, siendo éstos los principales artefactos de los procesos de Ingeniería del Software. La arquitectura MDA (*Model Driven Architecture*) [22] y sus estándares relacionados (MOF [20], QVT [24], OCL [23] y XMI [21]) proporcionan la base conceptual y tecnológica necesaria para llevar a cabo las ideas de dicho paradigma. De acuerdo al estándar QVT, el proceso de desarrollo del software se puede considerar como una serie de transformaciones de modelos, a partir de un nivel de abstracción alto hasta un nivel más específico. En el nivel más abstracto se pueden ver los requisitos, y en el nivel más específico estaría el código que implementa la aplicación. En conclusión, la principal importancia a la formalización de los modelos y las transformaciones entre ellos es que facilita la automatización del proceso de desarrollo de software.

El campo de la medición software puede beneficiarse de la nueva filosofía MDE, proporcionando la integración y el soporte necesario a la automatización de la medición de las diversas entidades del proceso software. Esto implica: la definición de modelos de medición de manera homogénea y consistente a partir de un metamodelo adecuado; la definición de medidas genéricas que puedan aplicarse a cualquier modelo; y el soporte necesario para calcular de forma automática las medidas definidas, almacenar de forma homogénea los resultados y facilitar la toma de decisiones mediante el análisis de los mismos. Por tanto, se habla de homogeneidad en el sentido de que existe una manera general de definir y medir cualquier tipo de entidad o artefacto software. Estos aspectos constituyen el principal interés del presente trabajo, en el que se ilustra la aplicación de los principios, normas y herramientas de MDA al campo de la medición del software. El objetivo es desarrollar un entorno genérico para la definición de modelos de medición *bien formados respecto a*[1] un metamodelo común, y para la medición de cualquier entidad software en base a un metamodelo de dominio. Para llevar a cabo la propuesta, se ha trabajado con el entorno MOMENT [15], que proporciona el soporte necesario para la gestión automática de modelos de acuerdo a MDE y MDA.

El resto de este artículo está organizado de la siguiente forma: En el siguiente apartado se muestran los trabajos relacionados. Después se presenta la arquitectura conceptual y el metamodelo de medición del software. En el cuarto apartado se describe la adaptación realizada del marco FMESP[12] a MDA, para lo cual se indican el entorno tecnológico empleado, la adaptación propiamente dicha de la parte de medición de FMESP a MDA, y se muestra el procedimiento de uso de dicha adaptación. A continuación se presenta un caso de ejemplo en el dominio de bases de datos relacionales. Por último, se presentan algunas conclusiones y trabajos futuros.

2. Trabajos Relacionados

Muchas publicaciones mencionan las herramientas que dan soporte y automatización a la medición como importantes factores de éxito en los esfuerzos de la medición del software [18], proporcionando entornos de trabajo y aproximaciones generales [17], o dando arquitecturas de soluciones más específicas [16].

Como consecuencia, en la bibliografía existe una gran variedad de herramientas que dan soporte a la creación, control y análisis de métricas software. Una lista extensa puede consultarse en [6]. Por su parte, Auer examina en [2] distintas herramientas de medición del software en entornos heterogéneos, como son: MetricFlame, MetricCenter, Estimate Profesional, CostXPert y ProjectConsole.

También se pueden encontrar en la bibliografía algunas propuestas en las que se

[1] A lo largo del trabajo se habla de la relación existente entre modelo y metamodelo. Las expresiones que se utilizan en el trabajo son: un modelo "*está bien formado respecto a*" o "*está definido mediante*" un metamodelo. Otras expresiones como un modelo "*es conforme a*" o "*es instancia de*" un metamodelo no se utilizan porque se consideran menos precisas.

aborda la medición de software más integrada y menos específica que en las herramientas anteriores. En [26] se propone MMR, que es una herramienta basada en el modelo CMMI para la evaluación de procesos software. Otras herramientas similares pueden consultarse en [19], [14], y [28]. Sin embargo, muchas de estas herramientas están restringidas a dominios o modelos de evaluación de calidad específicos, lo que reduce su generalidad y alcance.

Para abordar la medición genérica, en [11-13] y como parte del entorno FMESP (*Framework for the Modeling and Evaluation of Software Processes*), se propone un marco de trabajo basado en la arquitectura conceptual de metadatos del estándar MOF. Esta propuesta está soportada por la herramienta GenMetric, que permite definir modelos de medición software y calcular las medidas definidas en dichos modelos sobre cualquier entidad software. Sin embargo, se ha considerado interesante adaptar FMESP al paradigma MDE para explotar los beneficios que puede aportar a la medición del software. Esto implica: i) el refinamiento del metamodelo de medición del software definido en la propuesta anterior; y ii) la adaptación y extensión de GenMETRIC hacia un entorno que permita de una forma intuitiva tanto la definición de modelos de medición software que puedan aplicarse sobre cualquier tipo de entidad software como el cálculo de las medidas definidas, todo ello en el contexto de modelos y transformaciones de modelos de la arquitectura MDA.

3. Medición de Software basada en MDA: Arquitectura Conceptual

Tal y como se ha comentado en el apartado anterior, ante la necesidad de disponer de un entorno genérico y homogéneo para la medición del software en trabajos previos [11-13] se propone una arquitectura conceptual y una herramienta para la integración de la medición del software. A continuación, se describen brevemente las principales características de dicha propuesta. Una descripción más detallada puede consultarse en [13].

3.1. Arquitectura Conceptual

La propuesta presentada sirve para dar el soporte necesario para gestionar y representar el conocimiento referente a la medición de cualquier entidad software de una forma integrada y consistente. Para ello, los distintos elementos de la arquitectura se organizan de acuerdo a los siguientes niveles de abstracción, siguiendo la línea del estándar MOF.

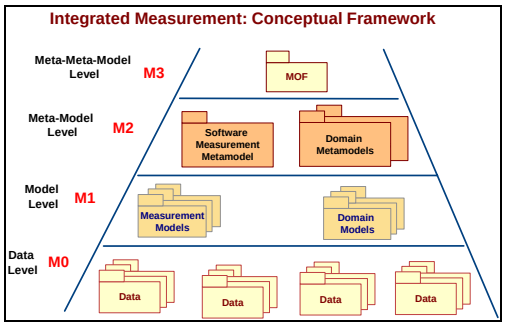

Figura 1. Marco de trabajo conceptual para gestionar la medición del Software

Como se observa en la Figura 1, la arquitectura se ha organizado en los siguientes niveles de metadatos:

- Nivel de Meta-Metamodelo (M3): en el que se encuentra un lenguaje abstracto para definir metamodelos. Este lenguaje es MOF.
- Nivel de Metamodelo (M2): en este nivel, de acuerdo al marco de trabajo se incluyen: el metamodelo de medición, que define los modelos de medición específicos (descrito en el apartado 3.2) y los metamodelos de dominio, que representan a los tipos de entidades candidatas para la medición (por ejemplo, el metamodelo UML para modelos de sistemas OO).
- Nivel de Modelo (M1): en este nivel se incluyen los modelos específicos definidos mediante los metamodelos de M2, pueden ser de dos tipos: los modelos de Medición, definidos mediante el metamodelo de medición; y los modelos de dominio, definidos mediante los correspondientes metamodelos de dominio.

3.2. Metamodelo de Medición del Software

Ante la diversidad terminológica existente en el campo de la medición software, como paso previo a la creación de la primera versión del metamodelo de Medición fue necesario desarrollar una ontología [11]. Dicha ontología permitió establecer y aclarar los elementos involucrados mediante la identificación de todos los conceptos, las definiciones precisas de todos los términos, y la aclaración de las relaciones entre ellos. Basándose en los conceptos y relaciones de esta ontología, se construyó el metamodelo de medición. La Figura 2 muestra en un diagrama de clases UML los principales constructores (o elementos) del Metamodelo de Medición del Software, que constituyen el núcleo de dicho metamodelo [9].

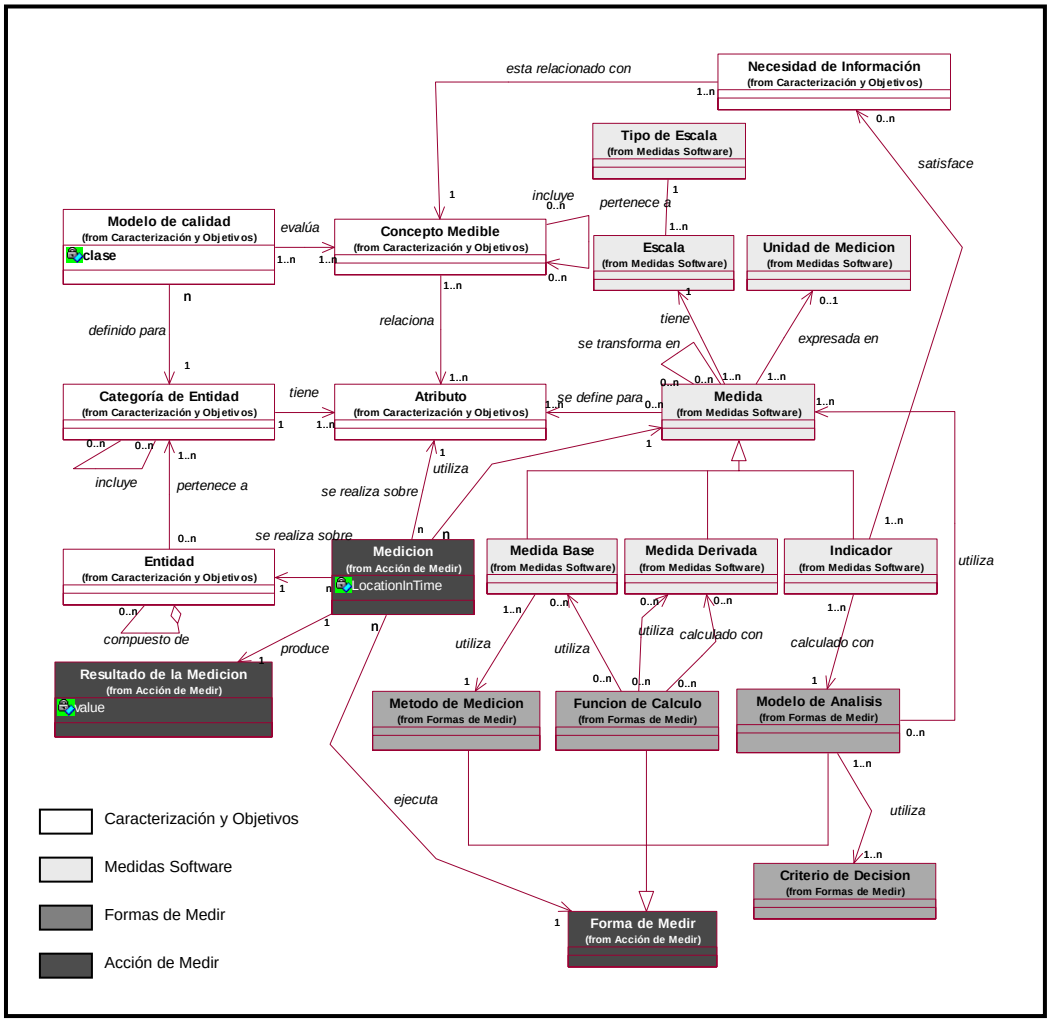

Figura 2. Metamodelo de Medición del Software

El metamodelo de Medición del Software está organizado en 4 paquetes principales (ver Figura 2) que se describen brevemente (para más detalle véase [9, 11]):

- **Caracterización y Objetivos de la medición del Software:** incluye los constructores necesarios para la definición de una vista fundamental del proceso de medición, basada en establecer los objetivos y en identificar y caracterizar los elementos necesarios para cumplir con dichos objetivos.
- **Medidas Software**: incluye los elementos necesarios para definir las características generales de las medidas.
- **Formas de medir**: incluye los elementos de modelado necesarios para la representación de la formas de medir de los diferentes tipos de medidas.
- **Acción de Medir**: incluye los constructores necesarios que permiten representar la forma de llevar a cabo el proceso de medición.

4. Adaptación de FMESP-Measurement a MDA

En este apartado se explican los detalles de la manera en que se ha realizado la adaptación del entorno FMESP a MDA.

4.1. Entorno Tecnológico

La herramienta utilizada en este trabajo ha sido el entorno de gestión de modelos MOMENT. MOMENT [4] es una herramienta que permite definir modelos como especificaciones algebraicas de una teoría expresada en lógica ecuacional condicional. Está basada en una aproximación híbrida entre una herramienta formal (Maude [1]) y un entorno industrial de modelado (Eclipse Modelling Framework, EMF [7]).

Desde un punto de vista funcional, MOMENT tiene dos componentes: ejecución de consultas OCL (MOMENT-OCL) y transformaciones QVT (MOMENT-QVT).

MOMENT-OCL [5] permite ejecutar consultas e invariantes OCL sobre un modelo definido mediante su metamodelo y almacenado en xmi. Incluye un editor (*OCLEditor*) que facilita la codificación de invariantes y consultas OCL que se van ejecutar sobre el modelo.

En este trabajo se ha empleado esta funcionalidad para comprobar y validar las consultas OCL utilizadas en las transformaciones QVT. Los resultados de la consultas OCL se muestran por pantalla (véase Figura 3).

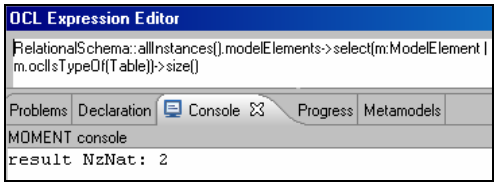

Figura 3. Ejemplo de ejecución de consulta OCL con MOMENT

Por otro lado, MOMENT-QVT [27] permite usar el lenguaje QVT Relations [24] para la definición de relaciones de equivalencia y transformaciones. El motor de transformaciones está integrado en esta herramienta, y permite obtener un modelo de salida a partir de uno o más modelos de entrada, siempre y cuando todos los modelos tengan su correspondiente metamodelo.

Tal y como se explica en el apartado 4.2, las transformaciones QVT son un elemento fundamental de esta propuesta.

Para ejecutar en MOMENT una transformación QVT es necesaria, bien la especificación textual de la transformación (codificada mediante el editor *Textual QVT Editor* y almacenada con extensión .qvtext), o bien, su equivalente modelo QVT Relation (almacenado con extensión .qvt). A continuación se muestran dos representaciones de ejemplo: una especificación textual (Figura 4), y su equivalente modelo QVT Relation (Figura 5).

Un modelo QVT Relation está definido mediante el metamodelo QVT, y puede obtenerse mediante un parseo de la especificación textual. También puede obtenerse mediante una transformación QVT.

```
top relation PackageToSchema
{
    packageName: String;

    checkonly domain ecoreDomain p:EPackage {
        name=packageName
    };

    enforce domain rdbmsDomain s:Schema {
        name=packageName
    };
}

top relation ClassToTable
{
    className: String;

    checkonly domain ecoreDomain c: EClass {
```

Figura 4. Parte de la Especificación textual de una transformación UML2RDBMS

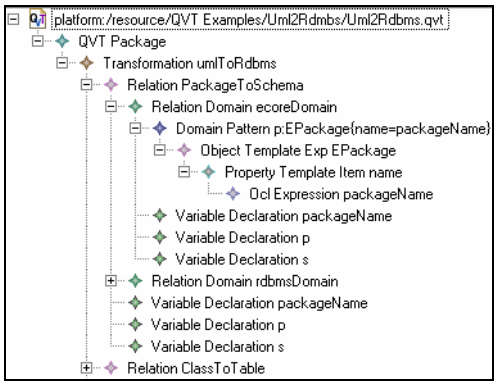

Figura 5. Modelo QVT Relation de una transformación UML2RDBMS

En esta propuesta se ha optado por trabajar con el modelo QVT Relation en lugar de la especificación textual.

Con respecto a la estructura de la plataforma EMF, es de interés resaltar que sigue la cultura de metamodelado presentada en el estándar MOF [20] con la diferencia de que en el nivel M3 se utiliza Ecore[2]. Puesto que Ecore está basado en MOF, la arquitectura de EMF es válida para la propuesta aquí presentada (véase Figura 1).

[2] Ecore es un lenguaje común basado en EMOF, que es parte de la especificación MOF 2.0 usado para la definición de metamodelos

4.2. Adaptación de FMESP-Measurement

En la Figura 6 se muestran los elementos clave para la adaptación de FMESP-Measurement, encuadrados en los niveles correspondientes de la arquitectura MOF. Como se puede observar, el metamodelo QVT (nivel M2) y el Modelo QVT Relation (M1) son los añadidos con respecto a la Figura 1, es decir, para la adaptación de FMESP-Measurement a MDA.

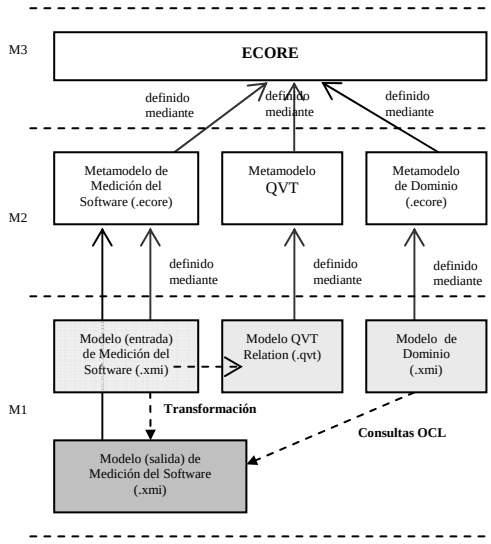

Figura 6. Elementos de la Adaptación de FMESP-Measurement a MDA

Con FMESP-Measurement, el resultado de la medición se obtiene mediante una transformación QVT, donde los modelos de entrada, son dos: un modelo de Medición del software y un modelo de dominio; y el modelo de salida es el modelo de Medición del software con los resultados obtenidos. Esta transformación se obtiene a partir del modelo QVT Relation (véase Figura 6).

A continuación se explican brevemente alguno de los elementos de la figura anterior:

- Metamodelo de Medición del Software (presentado en el apartado 3.2): metamodelo integrado en el marco de trabajo.
- Modelo QVT Relation: este modelo se obtiene mediante una transformación del modelo de de Medición. Contienen la información necesaria para llevar a cabo la transformación QVT de la propuesta FMESP-Measurement.

- Modelo (salida) de Medición del software: es el modelo resultante de la transformación definida en el modelo QVT Relation.

4.3. Procedimiento de Uso

A continuación se explican las etapas que debe llevar a cabo el usuario para poder medir usando la herramienta propuesta:

1. **Incorporación del metamodelo de dominio:** la medición se va a realizar sobre un dominio en concreto. Este dominio ha de ser definido mediante su correspondiente metamodelo (situado en el nivel M2 y bien formado respecto del meta-metamodelo Ecore). Ejemplos de dominios son: esquemas relacionales, diagramas de clases UML, diagramas de flujos de datos, código fuente en Java, manuales de usuario, etc.
2. **Creación del modelo de medición:** en base al metamodelo de medición integrado de base en el framework FMESP-Measurement, se crea el correspondiente modelo de medición, sobre el que se va a realizar la medición. Al ser un modelo de entrada, en este momento el modelo de medición no incluye la parte de los resultados, es decir, el paquete *Acción de Medir*.
3. **Creación del modelo de dominio:** a partir del metamodelo de dominio (incorporado en la etapa 1) se define el modelo de dominio, que representa "*lo que se va a medir*", es decir, el modelo sobre el cuál se va a aplicar la medición.
4. **Ejecución de la medición:** la ejecución de la medición se realiza mediante una transformación QVT, en la que partiendo de los dos modelos de entrada (modelo de medición y modelo de dominio) creados en las etapas 2 y 3 respectivamente, se obtiene un modelo de salida que es el modelo de medición pero ampliado con los resultados de la medición (paquete *Acción de Medir*). Estos resultados se calculan mediante consultas OCL realizadas sobre el modelo de dominio (véase Figura 7).

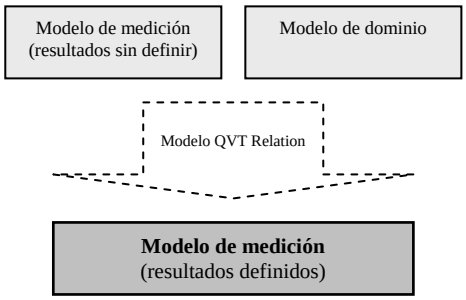

Figura 7. Ejecución de la medición

5. Caso de Ejemplo: Medición en el Dominio de Bases de Datos Relacionales

Para ilustrar la aplicación de la propuesta, se considera el ejemplo de la medición de esquemas relacionales. Por simplicidad se consideran los elementos mostrados en la Figura 8.

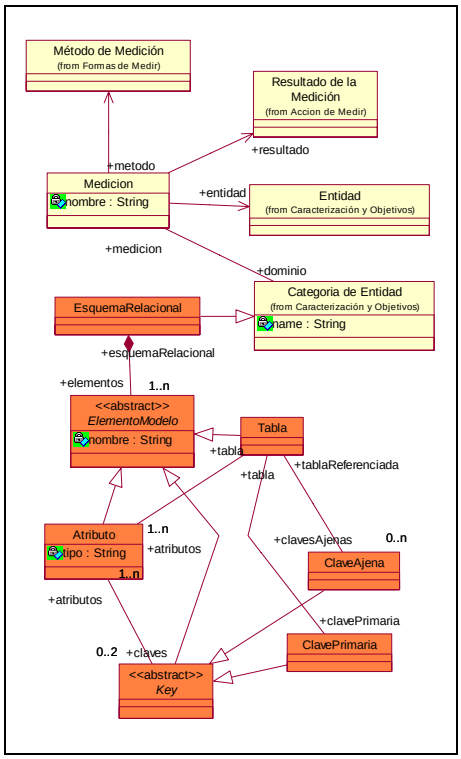

Figura 8. Metamodelo de Medición y de Dominio

Para la realización del caso de ejemplo, los elementos del metamodelo de medición más significativos son: *método de medición*, *entidad* a la que se va a aplicar el método de medición y *resultado* de la medición.

Por otro lado, se necesita el metamodelo de dominio, en este caso, se toma el metamodelo de e*squemas relacionales*. Ambos metamodelos son independientes físicamente (véase Figura 8), aunque están relacionados lógicamente. En la Figura 8 se ha distinguido el metamodelo de medición con color claro y el metamodelo de dominio con color oscuro.

5.1. Ejemplo del Método de Medición Contar Elementos

En este apartado, se explica cómo se realiza la medición "*contar el número de tablas"* de un esquema relacional. Tal y como se ha presentado en el apartado 4.3, la medición se realizaría en las siguientes cuatro etapas:

1. Incorporación del metamodelo de esquemas relacionales (partes de color oscuro en la Figura 8).
2. Creación del modelo de medición de esquemas relacionales. Para la forma de medir "contar elementos de tipo tabla", los valores para *Entidad* y *Método de Medición* son Tabla y Contar, respectivamente. Todavía no se incluye el *Resultado de la Medición.*
3. En base al metamodelo del paso 1 se crea el modelo del esquema relacional sobre el cuál se va a aplicar la medición. En este caso, el esquema relacional contiene información de una escuela universitaria. El esquema está compuesto de 5 tablas, con sus correspondientes atributos, claves primarias y ajenas (véase Figura 9).

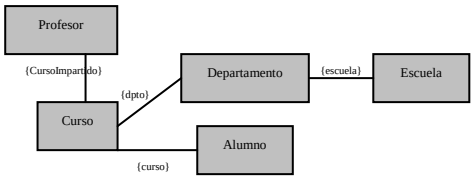

Figura 9. Ejemplo de modelo de dominio

4. Para ejecutar la medición, se utilizan como entradas el modelo de medición (etapa 2), el modelo de dominio (etapa 3); y el modelo QVT Relation (la obtención del modelo QVT Relation es automática y transparente al usuario, véase apartado 4.2). El modelo de salida que se consigue (véase Figura 10) contiene los resultados de medición definidos, es decir, el atributo *Resultado de la Medición* definido. En este caso con valor 5 (número de tablas).

```
<?xml version="1.0" encoding="ASCII"?>
<medicion:ModeloMedicion
    xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:medicion="http://procesoMedicion/medicion"
    nombreModelo="ModeloMedicion1">
  <mediciones name="Relaciones"
    metodo="Contar" elemento="Tabla"
    resultado="5"/>
</medicion:ModeloMedicion>
```

Figura 10. Resultado de la medición

La medición se lleva a cabo de forma automática gracias al modelo QVT Relation definido automáticamente mediante una transformación QVT del modelo de medición. A título informativo, en la Figura 11 y en la Figura 12 se muestra la especificación QVT equivalente al modelo QVT Relation. En dicha especificación QVT se pueden distinguir dos partes: una general (Figura 11) y otra más específica del modelo de medición (Figura 12).

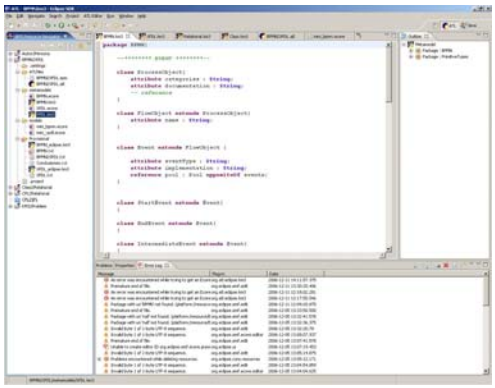

Figura 11. Parte genérica del código de la especificación QVT

En la parte general (Figura 11) se puede observar como la definición de los resultados es una llamada al método medición (representado en la Figura 12), que es una consulta OCL sobre el modelo de dominio. La consulta OCL es

específica en el sentido de que hay que conocer los elementos medibles y los métodos de medición para elaborar la consulta OCL.

```
function medicion(
    esquemaRelacional: EsquemaRelacional,
    metodo: String,
    elemento : String) : Integer
{
    if (metodo = 'contar') then
        if(elemento = 'Tabla') then
            relationalSchema.modelElements->
            select(m:ModelElement |
                m.oclIsTypeOf(Table))->size()
        else
            if(elemento = 'Atributo') then
                relationalSchema.modelElements->
                select(m:ModelElement |
                m.oclIsTypeOf(Atributo))->size()
            else
```

Figura 12. Parte específica del código de la especificación QVT

El proceso de transformación que permite obtener el modelo QVT Relation hace de la propuesta FMESP-Measurement una alternativa completamente genérica que le evita al usuario toda responsabilidad en cuanto a la escritura de la especificación de la transformación MDA para la medición.

6. Conclusiones y Trabajos Futuros

En este artículo se ha presentado un entorno genérico para la definición de modelos de medición basados en un metamodelo común y para la medición de cualquier entidad software a partir de los metamodelos que las representan. Siguiendo el enfoque de MDA, y partiendo del metamodelo de Medición (universal), es posible llevar a cabo la medición de cualquier dominio mediante transformaciones QVT, proceso que es completamente transparente al usuario.

Con el framework FMESP-Measurement, es posible medir cualquier entidad software sin más que disponer de los correspondientes metamodelos. La tarea del usuario consiste, únicamente, en la elección del metamodelo de dominio (según lo que se quiere medir en cada momento) y la definición de los modelos de entrada: modelo de dominio y de medición. El metamodelo de medición está integrado de base en el entorno.

El marco conceptual de la propuesta respeta completamente la filosofía MDA.

De cara al futuro, un trabajo importante es la realización de un plugin integrado en Eclipse que facilite al usuario la entrada de datos y el proceso de medición. Dicho plugin deberá servir para crear modelos de medición (de entrada) de manera fácil e intuitiva.

Adicionalmente, se pretende llevar a cabo diversas experiencias para validar esta propuesta con distintos dominios siguiendo las normas de estandarización adecuadas. En particular, se tendrá en cuenta la propuesta de la OMG, Software Metrics Meta-Model (SMM) [25], que actualmente está en fase de desarrollo.

Agradecimientos

Este trabajo ha sido parcialmente financiado por los proyectos ENIGMAS (Junta de Comunidades de Castilla-La Mancha, PBI-05-058), ESFINGE (Ministerio de Educación y Ciencia, TIN2006-15175-C05-05), COMPETISOFT (Programa Iberoamericano de Ciencia y Tecnología para el Desarrollo, 506AC0287) y META (Ministerio de Educación y Ciencia, TIN2006-15175-C05-01).

Referencias

[1] *The Maude System*, Department of Computer Science, University of Illinois, Urbana-Champaign.

[2] Auer, M., Graser, B. y Biffl, S.; *A Survey on the Fitness of Commercial Software Metric Tools for Service in Heterogeneous Environments: Common Pitfalls* Ninth International Software Metrics Symposium. (Metrics '03). (2003). pp.144.

[3] Bezivin, J., Jouault, F. y Touzet, D.; *Principles, standards and tools for model engineering*, 10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'2005). (2005). pp.28-29.

[4] Boronat, A., Carsí, J. Á. y Ramos, I.; *An Algebraic Baseline for Automatic Transformations in MDA*, Workshop Software Evolution Through Transformations: Model-baased vs. Implementation-level Solutions (SEETra'04),2nd International Conference on Graph Transformation (ICGT2004), ENTCS. Roma (italia).

[5] Brown, M. y Dennis, G.; *Measurement and Analysis: What Can and Does Go Wrong?*, 10th IEEE International Symposium on Software Metrics (METRICS'04). (2004). pp.131-138.

[6] Dumke, R. R. y Grigoleit, H.; *Efficiency of CAMEtools in software quality assurance*, Software Quality Journal 6(2). (1997). pp.157-169.

[7] Eclipse, *EMF (Eclipse Modelling Framework)*,

[8] Fenton, N. y Pfleeger, S. L., *Software Metrics: A Rigorous & Practical Approach, Second Edition*, (1997). p.

[9] Ferreira, F., García, F., Ruiz, F., Bertoa, M. F., Calero, C., Vallecillo, A., Piattini, M. y Mora, B.; *Medición del Software: Ontología y Metamodelo*, Informe Técnico UCLM-TSI-001. (2006).

[10] Florac, W. A., Carleton, A. D. y Barnard, J.; *Statistical Process Control: Analyzing a Space Shuttle Onboard Software Process*, IEEE Software. 17 (4). (2000).

[11] García, F., Bertoa, M. F., Calero, C., Vallecillo, A., Ruíz, F., Piattini, M. y Genero, M.; *Towards a consistent terminology for software measurement*, Information and Software Technology 48. (2006). pp.631-644

[12] García, F., Piattini, M., Ruiz, F., Canfora, G. y Visaggio, C. A.; *FMESP: Framework for the modeling and evaluation of software processes*, Journal of Systems Architecture - Agile Methodologies for Software Production 52. (2006). pp.627-639

[13] García, F., Serrano, M., Cruz-Lemus, J., Ruiz, F. y Piattini, M.; *Managing Software Process Measurement: A Metamodel-Based Approach*, Information Sciences, In press. (2007).

[14] Harrison, W.; *A flexible method for maintaining software metrics data: a universal metrics repository*, Journal of Systems and Software 72. (2004). pp.225-234

[15] ISSI, G., Grupo ISSI: MOMENT (A framework for Model ManagemENT).

[16] Jokikyyny, T. y Lassenius, C.; *Using the internet to comunicate software metrics in a large organization*, Proceedings of GlobeCom'99. (1999).

[17] Kempkens, R., Rösch, P., Scott, L. y Zettel, J.; *Instrumenting Measurement Programs with Tools*, PROFES 2000. Oulu, Finland. (2000).

[18] Komi-Sirviö, S., Parviainen, P. y Ronkainen, J.; *Measurement Automation: Methodological Background and Practical Solutions-A Multiple Case Study*, Seventh International Software Metrics Symposium (METRICS'01). London. (2001).

[19] Lavazza, L. y Agostini, A.; *Automated Measurement of UML Models: an open toolset approach*, Object Technology. 4(4). (2005). pp.115-134.

[20] OMG, *Meta Object Facility (MOF) Specification; version 1.4*, Object Management Group. (2002).

[21] OMG, *XML Metadata Interchange (XMI) Specification. OMG Document formal/00-11-02*, Object Management Group. (2002).

[22] OMG, *Model-Driven Architecture (MDA) Guide Version 1.0.1*, Object Management Group. (2003).

[23] OMG, *OCL 2.0 - OMG Final Adopted Specification*, Object Management Group. (2003).

[24] OMG, *Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification*, Object Management Group. (2005).

[25] OMG, *Architecture-Driven Modernization (ADM): Software Metrics Meta-Model (SMM). OMG Document: dmtf/2007-03-02*, Object Management Group. (2007).

[26] Palza, E., Fuhrman, C. y Abran, A.; *Establishing a Generic and Multidimensional Measurement Repository in CMMI context* 28th Annual NASA Goddard Software Engineering Workshop (SEW'03). Greenbelt (Maryland, USA). (2003). pp.12-22.

[27] Queralt, P., Hoyos, L., Boronat, A., Carsí, J. Á. y Ramos, I.; *Un motor de transformación de modelos con soporte para el lenguaje QVT relations*, Desarrollo de Software Dirigido por Modelos - DSDM'06 (Junto a JISBD'06). Sitges, Spain. (2006).

[28] Scotto, M., Sillitti, A., Succi, G. y Vernazza, T.; *A relational approach to software metrics*, Proceedings of the 2004 ACM symposium on Applied computing (SAC'2004). Nicosia, Cyprus. (2004). pp.1536-1540.