

# Medición del Software mediante Transformaciones QVT

Beatriz Mora, Francisco Ruiz, Félix García, Mario Piattini

Dep. de Tecnología y Sistemas de Información  
Escuela Superior de Informática de Ciudad Real  
Universidad de Castilla-La Mancha

{Beatriz.Mora | Francisco.RuizG | Felix.Garcia | Mario.Piattini}@uclm.es

## Resumen

En este trabajo se presenta el empleo de transformaciones QVT como mecanismo para ejecutar modelos de medición de software de forma que se obtengan automáticamente los resultados de las mediciones. Este mecanismo se está empleando en el desarrollo de varios componentes software que forman parte de FMESP, un framework general para la medición aplicable a cualquier tipo de propiedad de cualquier tipo de dominio software: esquemas de bases de datos, diagramas de actividad UML, código fuente en Java, documentos de pruebas, realización de proyectos y procesos, etc. Esta concepción genérica es posible de abordar gracias al aprovechamiento del potencial de abstracción del paradigma MDE. En detalle, se presentan las transformaciones QVT del componente FMESP-Measurement, que permiten ejecutar automáticamente modelos de medición de software y obtener otros modelos ampliados con los resultados de las mediciones. Además, se detalla la manera en la que se obtiene el modelo QVT-Relations de forma automática y transparente al usuario.

**Palabras clave:** Medición Software, MDA, Transformaciones QVT, Modelos QVT-Relations.

## 1. Introducción

Con el objetivo de conseguir una mejora continua en los procesos de la industria del software, es necesario gestionar correctamente las fases del ciclo de vida de dichos procesos [6]: definición, ejecución, medición, control y mejora. Es por ello, que la medición se ha convertido en un aspecto fundamental en la Ingeniería del Software [5].

La base organizacional en base a la cual se realiza el trabajo dentro de una organización que desarrolla o mantiene software, son los procesos software; si bien, en la práctica, dichos procesos se concretan y encuadran dentro de proyectos acotados en el tiempo y el espacio. El resultado de la ejecución de estos proyectos concretos son los productos software. En consecuencia, para que una organización obtenga mejores productos software es necesario que mejore, entre otros, sus procesos de ingeniería del software. Para ello se requiere llevar a cabo la medición del software de manera efectiva y consistente, siendo necesaria una disciplina para la medición y análisis de datos [4] y la definición, recopilación y análisis de medidas sobre el propio proceso, los proyectos y los productos software.

Los tipos de artefactos y propiedades candidatos para llevar a cabo una medición software son de una gran variedad. Esta heterogeneidad motiva la necesidad de disponer de modelos de medición genéricos que permitan diseñarlos, gestionarlos y ejecutarlos de una manera única común, independientemente de cuales sean las entidades software y propiedades a medir y de las formas de medirlas. Para este objetivo es muy útil considerar el paradigma MDE (*Model-Driven Engineering*) [2], de forma que los modelos de medición software (SMM, software measurement models) se convierten en los principales artefactos del proceso de medición.

La arquitectura MDA (*Model Driven Architecture*) [16] y sus estándares relacionados (MOF, QVT, OCL y XMI) proporcionan la base conceptual y tecnológica necesaria para llevar a la práctica estas ideas.

En [15] se presenta el marco de trabajo FMESP-Measurement para el modelado y mejora basada en la medición de los procesos software. El

trabajo aquí presentado se encuadra dentro de dicho marco de trabajo. FMESP-Measurement, utiliza las transformaciones de modelos para lograr la automatización de la ejecución de los SMMs. Esto implica, en primer lugar, la definición de los SMMs de manera homogénea y consistente a partir de un metamodelo de medición universal adecuado. En segundo lugar, es necesario actuar a nivel del metamodelo de forma que la definición de las medidas y las formas de medir sea genérica para que puedan aplicarse a cualquier modelo de cualquier dominio software. En tercer lugar, se trata de conseguir el cálculo de forma automática de las medidas definidas, almacenar de forma homogénea los resultados y facilitar la toma de decisiones mediante el análisis de los mismos.

En este trabajo se explican las transformaciones QVT integradas en FMESP-Measurement para conseguir los fines citados. Para ello, el resto del documento está organizado de la siguiente forma: primero se presentan otros trabajos relacionados (sección segunda) y se presentan las características de FMESP-Measurement (sección tercera). En la sección cuarta se explican las transformaciones y en la sección siguiente se presenta un caso de ejemplo. Se concluye con las principales conclusiones y trabajos en desarrollo o futuros.

## 2. Trabajos relacionados

Se pueden encontrar numerosas publicaciones que hablan de herramientas como importantes factores de éxito en los esfuerzos de medición del software [13], proporcionando entornos de trabajo y aproximaciones generales [12], o dando arquitecturas de soluciones más específicas [11]. En [4] se incluye una lista de herramientas que dan soporte a la creación, control y análisis de métricas software. Por otro lado, en [1] se examinan distintas herramientas de medición del software en entornos heterogéneos, como son: MetricFlame, MetricCenter, Estimate Profesional, CostXPert y ProjectConsole.

También se pueden encontrar algunas propuestas en las que se aborda la medición de software más integrada y menos específica que en los casos anteriores. Así, en [18] se propone la herramienta MMR, basada en el modelo CMMI para la evaluación de procesos software, y en [10,

14, 19] se pueden consultar otras herramientas similares. Sin embargo, estas herramientas están restringidas a dominios o modelos de evaluación de calidad específicos, lo que reduce su generalidad y el alcance de su aplicación.

Para abordar la medición software de manera realmente global, en [7-9] se presenta FMESP (Framework for the Modeling and Evaluation of Software Processes), un marco de trabajo basado en la arquitectura conceptual de metadatos del estándar MOF, que incluye una ontología y un metamodelo de medición software. Esta propuesta incluye la herramienta GenMETRIC para definir modelos de medición software y calcular las medidas definidas en dichos modelos sobre cualquier entidad software. Sin embargo, con posterioridad se ha considerado de interés adaptar FMESP al paradigma MDE [15] para explotar los beneficios que puede aportar a la medición del software, refinando el metamodelo de medición del software definido en la propuesta anterior; y adaptando y extendiendo GenMETRIC hacia un entorno que permita la definición de modelos de medición software y el cálculo de las medidas definidas, todo ello en el contexto de modelos y transformaciones de modelos de la arquitectura MDA. Esta última propuesta, llamada FMESP-Measurement, es el punto de partida de este trabajo, centrado en la definición de la transformación QVT requerida para llevar a cabo mediciones con FMESP-Measurement.

Con posterioridad a la publicación de la ontología y el metamodelo de la medición software como partes de FMESP, OMG ha considerado la oportunidad y necesidad de disponer de un metamodelo de medición software estandarizado. Con tal fin ha sido presentada una “*request for proposals*” y han comenzado los trabajos [17].

## 3. Medición basada en MDA: FMESP-Measurement

Como se ha comentado, FMESP-Measurement es el marco en el que se encuadra este trabajo. Por ello, se explican brevemente las características y elementos de dicho framework y el procedimiento para usarlo (para más detalle véase [15]).

Los elementos clave de FMESP-Measurement están encuadrados en los niveles correspondientes de la arquitectura MOF (Figura 1). Esta

arquitectura está diseñada para que el resultado de una medición se obtenga mediante una transformación QVT, en la cual los modelos de entrada son un SMM y un modelo de dominio (ambos basados en sus correspondientes metamodelos); y el modelo de salida es el SMM extendido con los resultados de la medición.

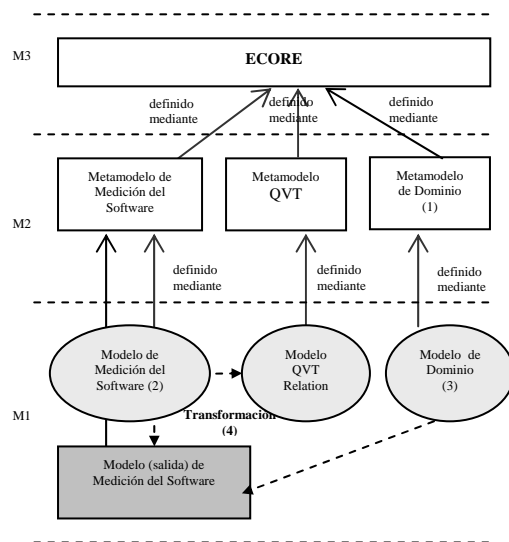


Figura 1. Elementos de FMESP-Measurement

Las etapas que se deben seguir para poder medir de forma automática utilizando FMESP-Measurement son cuatro (en la Figura 1 se muestra el número de etapa entre paréntesis):

1. **Añadir el metamodelo de dominio:** el dominio que engloba los tipos de artefactos y propiedades que se quieren medir ha de ser definido mediante su correspondiente metamodelo. Ejemplo: esquemas relacionales.
2. **Crear el modelo de medición:** en base al metamodelo de medición integrado de partida en la herramienta. Ejemplo: qué medir y cómo medirlo en un esquema relacional.
3. **Creación del modelo de dominio:** el modelo de dominio definido a partir del metamodelo de dominio representa el modelo concreto sobre el cuál se va a aplicar la medición. Ejemplo: un esquema de una base de datos relacional concreta.

4. **Ejecución de la medición:** se realiza mediante una transformación QVT, obteniéndose como salida el modelo de medición de entrada pero ampliado con los resultados de la medición.

#### 4. Transformación QVT-Relations

El modelo QVT-Relation (ver Figura 1), es el modelo de la transformación QVT necesaria para la ejecución automática de los SMMs. Este modelo se obtiene automáticamente como salida de otra transformación QVT previa, mostrada en la Figura 2. Como se aprecia en dicha figura, este modelo QVT-Relation, que podríamos llamar extendido o final, se obtiene mediante una transformación QVT tomando como modelos de entrada el modelo QVT-Relation básico o inicial (el basado directamente en el metamodelo QVT, véase Figura 3) y el modelo de Medición del Software previamente definido.

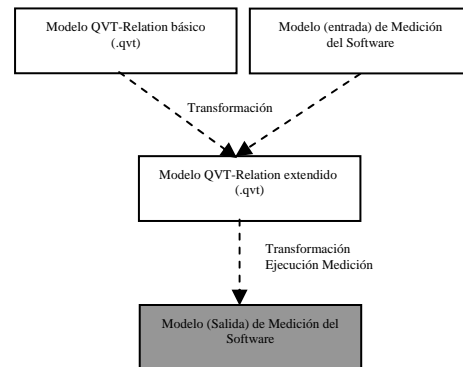


Figura 2. Transformación del modelo QVT-Relation

El Modelo QVT-Relation extendido amplía el modelo QVT-Relation básico de entrada con los siguientes aspectos:

1. **Transformation Model:** para obtener el modelo QVT-Relation extendido, se necesita especificar los modelos de entrada. En este caso, los modelos de entrada son dos: el modelo de medición y el modelo del dominio. Puesto que el modelo de medición es siempre el mismo y por tanto ya está definido en el modelo QVT-Relation básico, sólo habría que indicar el modelo del dominio, señalando los atributos

*name* y *uri* del modelo. Esta información se toma a partir del modelo de Medición que contiene toda la información relativa a la medición.

2. **Relation Domain:** para llevar a cabo la transformación es necesario indicar los *checkonly domain* de la transformación. En este caso se tienen dos *checkonly domain*, uno por cada modelo de entrada: el modelo de dominio y el modelo de medición. Sólo hay que indicar el *checkonly domain* del modelo de dominio ya que el del modelo de medición ya está definido en el modelo QVT-Relation básico, puesto que es siempre el mismo.

3. **Function:** función que contiene las consultas OCL necesarias para la ejecución de la medición. Dichas consultas OCL son la implementación de las “formas de medir abstractas” que están definidas a nivel de metamodelo.

En el modelo QVT-Relation básico estos tres elementos están vacíos (elementos sombreados en la Figura 3). Por ello, es necesario llevar a cabo la transformación de la Figura 2 cuya salida es el Modelo QVT-Relation extendido con dichos elementos.

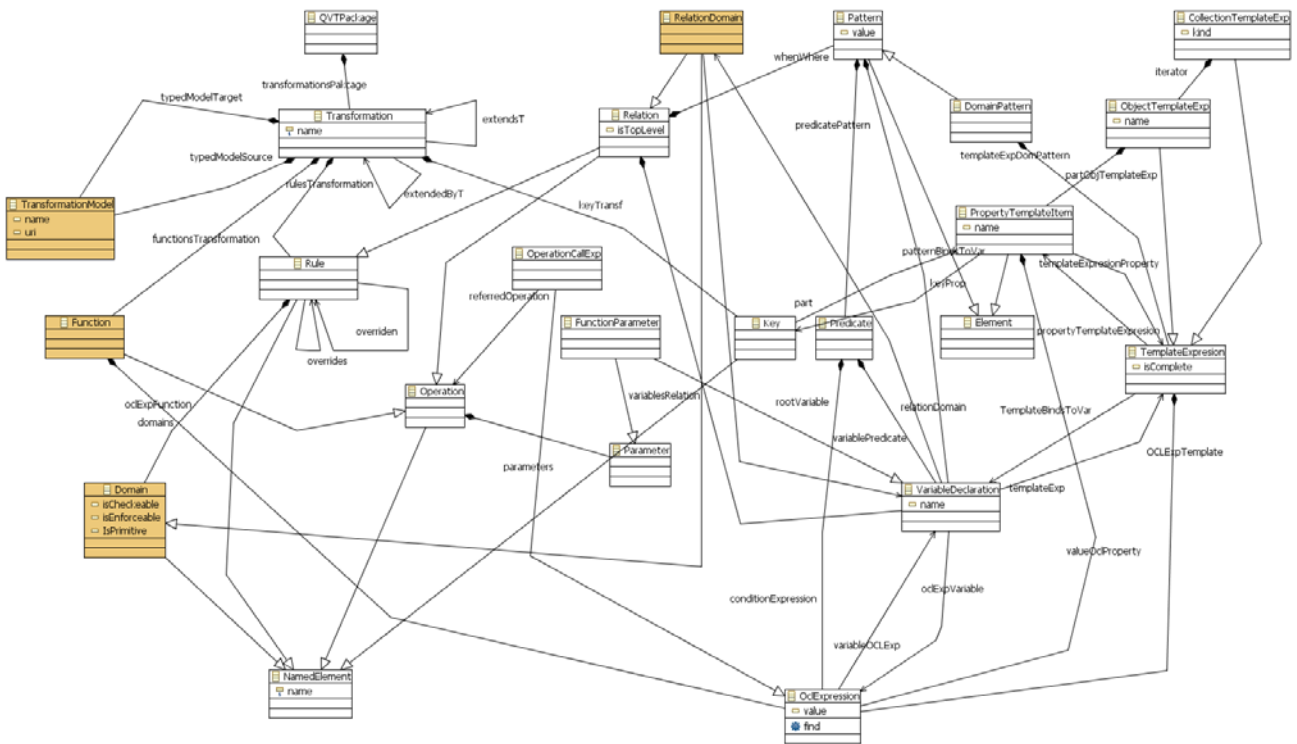


Figura 3. Parte del Metamodelo QVT

## 5. Caso de ejemplo: forma de medir “contar elementos”

Para ilustrar la ejecución de mediciones software mediante transformaciones QVT se considera el caso de la medición de esquemas relacionales. Como método de medición se ha elegido “contar elementos de tipo tabla”, que será una instanciación del método abstracto “contar elementos de tipo X”. En la Figura 4 se muestran los metamodelos de medición y de dominio (esquemas relacionales) simplificados, por razones didácticas.

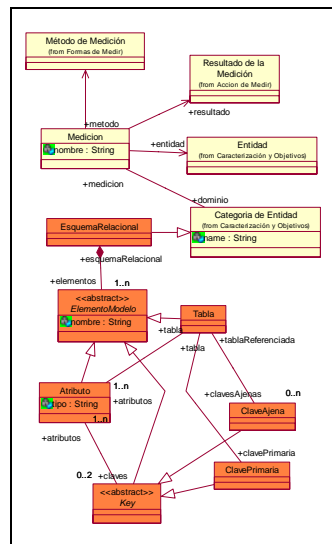


Figura 4. Ejemplo de Metamodelos de Medición y de Dominio

Para llevar a cabo la medición, se realizan los cuatro pasos siguientes:

1. Añadir el metamodelo de esquemas relacionales (partes de color oscuro en la Figura 4)
2. Crear el modelo de medición de esquemas relacionales. Para el método de medición “contar elementos de tipo tabla” *Entidad* se asocia con *Tabla*. Todavía no se incluye el *Resultado de la Medición*.
3. Añadir el modelo del esquema relacional sobre el cuál se va a aplicar la medición. En este

ejemplo, el esquema relacional contiene información relativa a un centro universitario (ver Figura 5). En la figura se representan el conjunto de tablas con sus correspondientes atributos, claves primarias (en sombreado y negrita) y claves ajenas (en subrayado y cursiva).

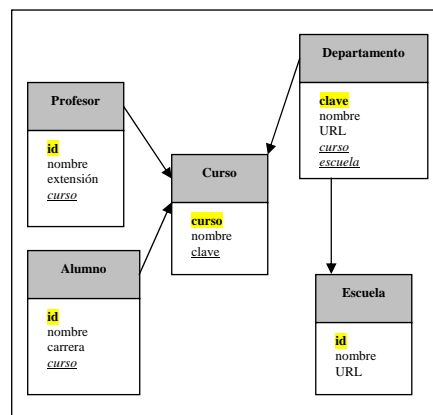


Figura 5. Ejemplo de modelo de dominio

4. Para ejecutar la medición, se realiza la transformación QVT tomando como entradas el modelo de medición (paso 2), el modelo de dominio (paso 3) y el modelo QVT Relation extendido. El modelo de salida que se consigue contiene los resultados de medición definidos. En el caso de ejemplo habría una instancia de medición asociada a la instancia “contar elementos de tipo Tabla” de método de medición y asociada a una instancia de “resultado de medición” con valor 5 (número de tablas) en su atributo resultado. La Figura 6 muestra un fragmento del código XMI resultante.

```
<?xml version="1.0" encoding="ASCII"?>
<medicacion:ModeloMedicacion
  xmi:version="2.0"
  xmlns:xmi="http://www.omg.org/XMI"
  xmlns:medicacion="http://procesoMedicacion/medicacion"
  nombreModelo="ModeloMedicacion1">
  <mediciones name="Relaciones"
    metodo="Contar" elemento="Tabla"
    resultado="5"/>
</medicacion:ModeloMedicacion>
```

Figura 6. Resultado de la Medición

Obtenido el modelo QVT-Relation extendido, ya sería posible realizar la transformación que

ejecuta el modelo de medición para obtener su versión ampliada con los resultados de la medición.

En el caso de ejemplo, los elementos del metamodelo QVT-Relations extendido que amplían el modelo básico, tal como se ha comentado en el apartado 4, son:

1. Transformation Model: el modelo de entrada es el modelo del dominio de esquemas relacionales.
2. Relation Domain: se indica la parte del *checkonly domain* del modelo de esquemas relacionales (ver Figura 7).
3. Function: función que contiene las consultas OCL necesarias para la ejecución de las mediciones, en este caso para implementar el método de medición “contar elementos de tipo X”, cuando X es Tabla (ver Figura 8).

```

checkonly domain DominioRelacional
  srcEsquemaRelacional : EsquemaRelacional {
    name = nombreEsquemaRelacional
  };
enforce domain dominioMedicion srcModeloMedicion1:
  ModeloMedicion {

    nombreModelo = nombreModelo1,
    mediciones = dstMedicion1 : Medicion {
      nombre = nombreEsquemaRelacional,
      metodo = metodo1,
      elemento = elemento1,
      resultado = medicion(srcEsquemaRelacional,
        metodo, elemento) //CONSULTA OCL
    }
  };

```

Figura 7. Elemento “Relation Domain” del modelo QVT-Relation extendido

```

enforce domain dominioMedicion srcModeloMedicion1:
  ModeloMedicion {

    nombreModelo = nombreModelo1,
    mediciones = dstMedicion1 : Medicion {
      nombre = nombreEsquemaRelacional,
      resultado = medicion(srcEsquemaRelacional)
    }
  };
function medicion(
  esquemaRelacional: EsquemaRelacional) : Integer
{
  relationalSchema.modelElements->
  select (m:ModelElement |
    m.oclIsTypeOf(Table))->size()
}

```

Figura 8. Elemento “Function” del modelo QVT-Relation extendido

## 6. Conclusiones y Trabajos Futuros

En este trabajo se ha presentado un mecanismo basado en transformaciones QVT para poder automatizar las mediciones de cualquier tipo de propiedad de cualquier tipo de dominio o artefacto software. Estas transformaciones QVT se emplean en el marco FMESP-Measurement, que permite la definición de modelos de medición, basados en un metamodelo de medición universal y en un modelo de dominio software, y la ejecución automática de dichos modelos. El resultado de dicha ejecución es un nuevo modelo de medición extendido con las instancias y valores de las correspondientes mediciones.

La ejecución de los modelos de medición mediante transformaciones QVT es transparente para el usuario. Para conseguir dicha transparencia ha sido técnicamente necesario disponer de un motor de transformaciones QVT/OCL. Por esta razón, los componentes de FMESP-Measurement están siendo implementados sobre MOMENT (MOdel management) [3].

De esta manera, 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 (que lo normal es que sea estándar y ya exista) y de medición. El resto (obtención del modelo de la transformación QVT-Relations a partir del modelo de medición) es realizado de forma automática y transparente para el usuario.

Un trabajo importante de cara a facilitar el uso es la realización de un plugin integrado en ECLIPSE que permita la definición de modelos de medición software mediante un lenguaje gráfico basado en estereotipos de UML.

Adicionalmente, se piensa incorporar el futuro “Metamodelo de Métricas Software” que está en fase de elaboración por el “OMG Architecture Driven Modernization Task Force” [17], en cuyo “request for proposal” han sido incluidos como referencias la ontología SMO [7].

## 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) y COMPETISOFT (Programa Iberoamericano de Ciencia y Tecnología para el Desarrollo, 506AC0287).

## Referencias

- [1] Auer, M., Graser, B. y Biffi, 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.
- [2] 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.
- [3] Boronat, A., Carsí, J. Á. y Ramos, I.; *Algebraic Specification of a Model Transformation Engine*, LNCS. Fundamental Approaches to Software Engineering (FASE'06). ETAPS'06. Vienna (Austria). (2006).
- [4] 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.
- [5] Fenton, N. y Pfleeger, S. L.; *Software Metrics: A Rigorous & Practical Approach, Second Edition*, (1997). p.
- [6] Florac, W. A., Carleton, A. D. y Barnard, J.; *Statistical Process Control: Analyzing a Space Shuttle Onboard Software Process*, IEEE Software. 17 (4). (2000).
- [7] 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
- [8] 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
- [9] 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).
- [10] Harrison, W.; *A flexible method for maintaining software metrics data: a universal metrics repository*, Journal of Systems and Software 72. (2004). pp.225-234
- [11] Jokikyynty, T. y Lassenius, C.; *Using the internet to communicate software metrics in a large organization*, Proceedings of GlobeCom'99. (1999).
- [12] Kempkens, R., Rösch, P., Scott, L. y Zettel, J.; *Instrumenting Measurement Programs with Tools*, PROFES 2000. Oulu, Finland. (2000).
- [13] 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).
- [14] Lavazza, L. y Agostini, A.; *Automated Measurement of UML Models: an open toolset approach*, Object Technology. 4(4). (2005). pp.115-134.
- [15] Mora, B., García, F., Ruiz, F., Piattini, M., Boronat, A., Gómez, A., Carsí, J. Á. y Ramos, I.; *Marco de Trabajo basado en MDA para la Medición Genérica del Software*, JISBD. Zaragoza. (2007).
- [16] OMG, *Model-Driven Architecture (MDA) Guide Version 1.0.1*, Object Management Group. (2003).
- [17] OMG, *Architecture-Driven Modernization (ADM): Software Metrics Meta-Model (SMM)*. OMG Document: dmtf/2007-03-02, Object Management Group. (2007).
- [18] 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.
- [19] 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.