Measuring Model Repositories

Éric Vépa, Jean Bézivin, Hugo Bruneliere, Frédéric Jouault

Measuring Model Repositories

Éric Vépa, Jean Bézivin, Hugo Brunelière, Frédéric Jouault
ATLAS (INRIA & LINA)
University of Nantes
2, rue de la Houssinière BP 92208
44322, Nantes, France
{eric.vepa | jbezivin | hugo.bruneliere | f.jouault}@gmail.com

Keywords

Model repository; Model Metrication.

Abstract

We first present a model repository that has been built as part of the open source Eclipse GMT/AM3 project (Generative Modeling Technology/ATLAS MegaModel Management). Several contributed artifacts present in this repository are organized into sets of models of similar nature called zoos. The structure of the repository will be rapidly described. Its content is very rapidly extending, providing a publicly available source of experimental data to evaluate real life sets of model engineering artifacts. As an initial experiment, this paper shows how the elements contained in the AM3 zoos can be measured. Some examples of such measurements are provided for illustration purposes.

1. INTRODUCTION

We first present a model repository that has been built as part of the open source Eclipse GMT/AM3 project (Generative Modeling Technology/ATLAS MegaModel Management). Several contributed artifacts present in this repository are organized into sets of models of similar nature called zoos. The structure of the repository will be rapidly described in section 2. Its content is very rapidly extending, providing a publicly available source of experimental data to evaluate real life sets of model engineering artifacts. As an initial experiment, this paper shows how the elements contained in the AM3 zoos can be measured. Some examples of such measurements are provided for illustrative purposes.

In addition to the description of the AM3 open source model repository initiative, this paper introduces several new research contributions. A classification of abstract models is presented with several advantages allowing dealing similarly with terminal models or metamodels when necessary. This conceptual framework has been followed by the physical implementation of the repository. A set of tools have been built to support the approach.

In [7] we have proposed to use model transformation to check any kind of models. The result of such an operation is a diagnostic model conforming to a diagnostic metamodel. This strong result could be applied to any zoo of models. For example the zoo of KM3 metamodels [10] (i.e. models conforming to KM3) contains several consistent metamodels, that have been used in several practical transformations. It also contains some non consistent models, for example incomplete fragments of metamodels. This is done on purpose, to illustrate that the zoo contains real life artifacts, of various qualities. As a result, it is interesting to apply some automated checks on all the models of a given zoo to get the consistency diagnostic.

In the present contribution we discuss a new and different result. Similarly to the approach presented in [7], we propose here to implement elementary or composite measures on models as transformations. Here again we use the ATL transformation language [6] to practically illustrate the approach. The various measures on models may be expressed as transformations. These transformations are themselves contributed to the library of transformations (the zoo of transformations). It is then possible to measure transformations and even measure transformations.

2. The AM3 Model Repository

The GMT/AM3 project is hosting a repository of models [10], [11]. By model we mean here as well terminal models (like a UML model) or metamodels based on a given metamodel like MOF 2.0. Terrminal models may be based on any kind of metamodel. Figure 1 summarizes the organization of this repository.

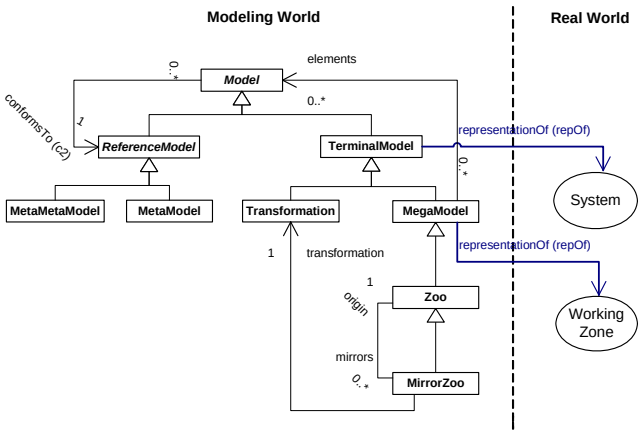

Figure 1. Theoretical aspects of our Global Model Management approach

In MDE (Model Driven Engineering), models are first class entities. Models are organized in three different categories: *terminal models*, *metamodels,* or *metametamodels*. Each model conforms to its reference model, i.e. a metamodel or a metametamodel. It is important to note that the reference model of a metametamodel is always itself. A terminal model is a

representation, that conforms to a given metamodel (its reference one) and that is also a representation of a *real world system*.

Transformations and megamodels are examples of terminal models with particular properties. A transformation is a model that can be processed (with a virtual machine for example) in order to build a target model from a source model, both of them conforming to a metamodel that can be different.

The following definitions, related to the core concept of *model*, have been taken from [11] before being reformulated:

Definition 1. A technical space [15] [13] is a model management framework, belonging to the "modeling world", with a set of tools that operate on the models definable within the framework.

Definition 2. A system is a delimited part of the world (the "real world") considered as a set of elements in interaction. It can be represented in terminal models.

Definition 3. A model is a representation of a given system. For each question of a given set of questions, the model will provide exactly the same answer that the system would have provided in answering the same question.

Definition 4. A terminal model (M1) is a model such that its reference model is a metamodel, i.e. it conforms to its reference metamodel. It is a representation of a "real world" system.

Definition 5. A metamodel (M2) is a model such that its reference model is a metametamodel, i.e. it conforms to its reference metametamodel.

Definition 6. A metametamodel (M3) is a model that is its own reference model, i.e. it conforms to itself.

Now that we have clearly defined the general concept of "model" and all its main related concepts, we must specify the new concepts we add in our approach in order to deal with the general problems of GMM:

Definition 7. A working zone is a delimited part of the world (the "real world") consisting of MDE resources.

Definition 8. A transformation is a terminal model that defines a transformation from a model $M1_A$, conforming to a source metamodel $M2_S$, to a model $M1_B$ conforming to a target metamodel $M2_T$. Its reference model is a transformation metamodel (like the metamodel of the ATL language for example [6]).

Definition 9. A megamodel is a terminal model such that all its elements are models (i.e. all kinds of modeling artifacts and modeling tools like terminal models, metamodels, metametamodels…). It is a representation of a "real world" working zone. We consider this concept as the core of our GMM approach but also as a central part of the "modeling in the large" principle discussed in [12].

Definition 10. A zoo is a megamodel such that all models that compose it have the same metamodel (i.e. the same reference model). The kind of modeling artifact that can be found in a zoo may vary (as an example, the "Atlantic zoo" and the "ATL transformation zoo" which are located in [1] are zoos, respectively of metamodels and of transformations). Alternatively, a zoo may be considered as a view on a megamodel. This view may be implemented as a transformation for example. A zoo may have several mirrors, each of them having this zoo as original zoo.

Definition 11. A mirror zoo is a zoo that has been automatically generated, by the execution of a given transformation, from a specified original zoo. There are several types of events that can be the triggers of the generation, or regeneration, of a mirror zoo (as an example, when a modification occurs in the original zoo…).

A preliminary original description of the AM3 project in general terms (its presentation, its motivations, its use cases, and its global objectives) has already been given in [12]. However the content of the AM3 project has much evolved during the recent period. In fact, this project currently offers to Eclipse community's users some general documentations, three functional free-downloadable plugins and two zoos (with several mirrors for one of the two zoos).

As previously mentioned, the AM3 project already provides many MDE resources which are mainly at this time, metamodels and transformations. In fact, three zoos and several mirrors are currently freely available in the AM3 project:

Zoo 1. The Atlantic Zoo: It is composed of metamodels expressed in KM3 (Kernel MetaMetaModel) format [10]. This zoo has several mirrors that are auto-generated from it by using model transformations in ATL (the number of mirrors is variable and evolving, with new mirror zoos forthcoming like one in the Prolog language):

- **Mirror 1.** The AtlantEcore Zoo: It contains metamodels expressed in EMF **XMI 2.0**, conforming to **Ecore**.
- **Mirror 2.** The Atlantic MOF/MDR Zoo: It contains metamodels expressed in MDR **XMI 1.2**, conforming to **MOF 1.4**.
- **Mirror 3.** The Atlantic UML Zoo: It contains UML - class diagram's representations of metamodels expressed in MDR **XMI 1.2**, conforming to **UML**, that are compatible with the Poseidon UML CASE tool.
- **Mirror 4**. The Atlantic Raster Zoo: It contains graphical representations of metamodels expressed in **PNG** bitmaps.
- **Mirror 5**. The Atlantic SQL DDL Zoo: It contains metamodels' representations expressed in **SQL DDL** (Data Definition Language), conforming to **SQL**. They have been tested with the MySQL DBMS.
- **Mirror 6**. The Atlantic Microsoft DSL Tools Zoo: It contains "domain models" expressed in the **DSL Tools** specific XML format (".dsldm" files). These files are usable under Visual Studio 2005 with the Visual Studio 2005 SDK (including the DSL Tools) [8].
- **Mirror 7**. The Atlantic Microsoft Visual Basic Zoo: It contains metamodels expressed in Visual Basic source code for Visual Studio 2005.
- **Mirror 8**. The Atlantic XAMS Zoo: It contains metamodels expressed as abstract machines in XAMS which is an open source compiler for Abstract State Machines (ASMs) [4].
- **Mirror 9**. The Atlantic AsmL Zoo: It contains metamodels expressed as abstract state machines in the Microsoft Abstract State Machine Language [5].

- **Mirror 10**. The Atlantic GME Zoo: It contains metamodels expressed in the Generic Modeling Environment (GME) format.

Zoo 2. The ATL Transformation Zoo: It is composed of model transformation expressed in ATL (ATLAS Transformation Language) format [6].

Zoo 3. The AMW Zoo: It is composed of weaving metamodels and extensions of weaving metamodels expressed in KM3 format. For the moment, it provides the AMW weaving core metamodel and some already defined basic extensions metamodels.

All files contained in these zoos or mirror zoos are directly downloadable in the "Zoos" page of the AM3 Project. Several other mirrors of the Atlantic Zoo, as well as mirrors of the ATL Transformation Zoo, may be created and added to the list of available zoos in this project. The MDE artifacts (internal ones or imported external ones) may also be stored locally or geographically distributed.

However new zoos are currently being built in various formalisms, for example zoos of terminal models. These may be for example UML 2.0 models, but also other models expressed in XMI and conforming to any other metamodel than UML. Several of these terminal models will be available soon from the AM3 federation of zoos.

3. MEASURING MODELS

With several hundreds of models directly contributed or generated in the AM3 zoos, it is now possible to perform several investigations on this experimental material. For example measuring the various models may bring interesting insight into the consideration of these models. Furthermore comparing similar measures applied to models of various origins may bring interesting observations. The initial metrication campaign is outlined below. It applies mainly to the various metamodels. The idea is to collect measurement data from metamodels, to compare them, establish producer profile, etc. For doing this, metrics on metamodels must be defined. A set of existing metrics on class diagram can be reused on metamodels.

3.1 Metamodel for representing measurement data

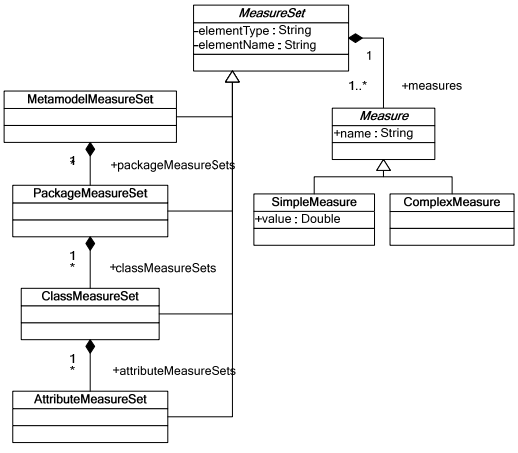

Figure 2: Measure metamodel

The previous metamodel offer the possibility of organizing sets of measures on different model elements (like metamodel, package, class or attribute). A set of measures owns a type (the model element concerned) and a name (from the model element). These two attributes are needed for the tabular representation of measurement data.

About the measures, they own a name and can be simple or complex. A simple measure also owns a double value and is used for metrics like TNC (Total Number of Classes) or DIT (Depth Inheritance Tree). More complex measures have not yet been implemented.

3.2 Measurement data on a metamodel

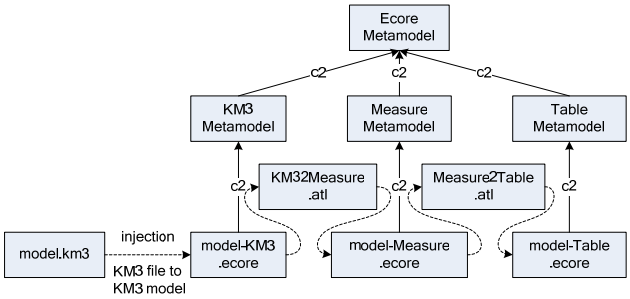

Figure 3: Collecting measurement data from one KM3 metamodel of the zoo

A first step is to collect measurement data on one metamodel of the KM3 zoo. We do this with ATL transformations.

First a metamodel is selected from the zoo in KM3 format. We inject this model to an Ecore model and use it as the input model of the first transformation. The first transformation input and output metamodels handlers are KM3 and Measure. The collection of the measurement data is done by running the transformation *KM32Measure*.

We obtain an output model of measures with the hierarchy and model elements from the Measure metamodel. The metrics used in the transformation will be explained in an upcoming section.

To have a better vision of the resulting measures we performed another transformation. The metamodels handlers of this transformation are Measure and Table. The resulting model is a generic tabular representation (A table contains rows and rows contains cells).

3.3 Measurement data on the entire zoo of KM3 metamodels

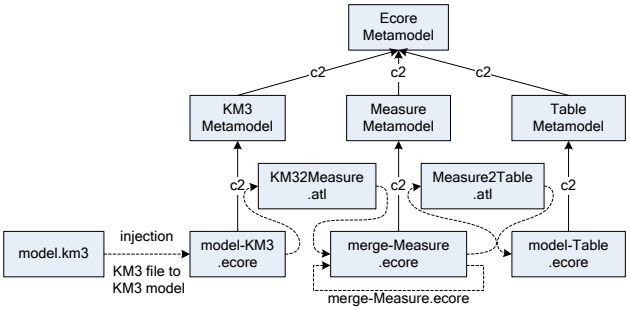

Figure 4: Collecting measurement data from the entire zoo of KM3 metamodel

We are now able to collect measurement data for each metamodel of the zoo. But we also want to have a global vision of the measures performed on the entire zoo.

To do this, we have to consider only the MetamodelMeasureSet elements and merge all the measures from all metamodels.

This step is done with an ANT script which iterates on the zoo and run a transformation which refines the model of measure of one metamodel and merges it with the measures from the previous metamodels of the zoo.

The metamodels handlers for this refining and merging transformation are both Measure. We have two inputs, the model generated by *KM32Measure* and a second model for storing the measurement data from the entire zoo. For the first iteration, the second model is empty. But next, it is passed as the second model for the next iteration. It is a way to refine and merge data with one transformation and using an ANT script.

3.4 Final measurement data representations

Other transformations have been developed from the model of table containing measurement data from one metamodel or for the entire KM3 zoo. The main output formats are HTML and SVG.

We can obtain an HTML file containing tables, by performing a *Table2HTML* transformation and an extraction.

The same method is possible for SVG, with two possible representations. Bar chart and pie chart with the transformations *Table2SVGBarChart* and *Table2SVGPieChart*, followed by an extraction to a SVG file.

3.5 Metrics implemented

These metrics are primitives metrics needed for the next metrics:

- Total Number of Packages

  TNP is the total number of packages in the metamodel.

- Total Number of Classes

  TNC is the total number of classes in a package or the metamodel.

- Total Number of Attributes

  TNA is the total number of attributes in a class, package or the metamodel.

As we said, these metrics are metrics from UML class diagram. But we can reuse those which do not concern methods, visibility and overwriting. KM3 metamodel contains packages which contains classes and enumerations. And a class can contain attributes and references (composition or association relationships).

- Depth Inheritance Tree

  DEFINITION. The Depth of inheritance of a class is the DIT metric for a class. In cases involving multiple inheritances, the DIT will be the maximum length from the node to the root of the tree.

  GOAL. DIT is a measure of how many ancestor classes can potentially affect this class. This metric was proposed as a measure of class complexity, design complexity and potential reuse. It is based on the idea that the deeper a class is in the hierarchy, the greater the number of attributes and relationships it is likely to inherit.

- Number of Children

  DEFINITION. The Number of Children (NOC) is the number of immediate subclasses subordinated to a class in the class hierarchy.

  GOAL. This is a measure of how many subclasses are going to inherit the attributes and relationships of the parent class.

- Number of Attributes

  DEFINITION. The Number of Attributes (NA) is defined as the total number of attributes in a class.

  GOAL. The number of attributes available to the class and not its instances affects the size of a class.

- Number of Attributes Inherited

  DEFINITION. The Number of Attributes Inherited metric (NAI) is defined as the total number of attributes inherited by a subclass.

  GOAL. This metric looks at the quality of the classes use of inheritance. It examines superclass-subclass inheritance relationships

- Attribute Inheritance Factor

  DEFINITION. The Attribute Inheritance Factor (AIF) is defined as a quotient between the sum of inherited attributes in all classes of the system under consideration and the total number of available attributes (locally defined plus inherited) for all classes:

  $$AIF = \sum Ai(Ci) / \sum Aa(Ci)$$

  Where: $Aa(Ci) = Ad(Ci) + Ai(Ci)$ = attributes available in Ci (those that can be manipulated in association with Ci), $Ad(Ci) = An(Ci) + Ao(Ci)$ = attributes defined in class Ci (those declared in Ci), An(Ci) = new attributes in class Ci, Ai(Ci) = attributes inherited in class Ci (those inherited in Ci)

  GOAL. AIF is defined as a measure of inheritance, and therefore a measure of the level of reuse.

3.6 Sample results

This section provides some sample results obtains with ATL transformations with SVG as the target metamodel. Many other result shapes, for example Kiviat diagram could be obtained similarly. By changing the target metamodel, for example XHTML or Excel, we could obtain tabular presentation as previously mentioned.

The length limitation of this paper does not allow illustrating the variety of information that may be extracted in various tabular forms from metamodels. However the first results of this measurement campaign show how easy it is to build a library of reusable measurement transformations.

The two following colour pictures are not of excellent printing quality, but have been included in the paper to illustrate how various model metrics may be obtained in real time, as a result of a simple ATL transformation.

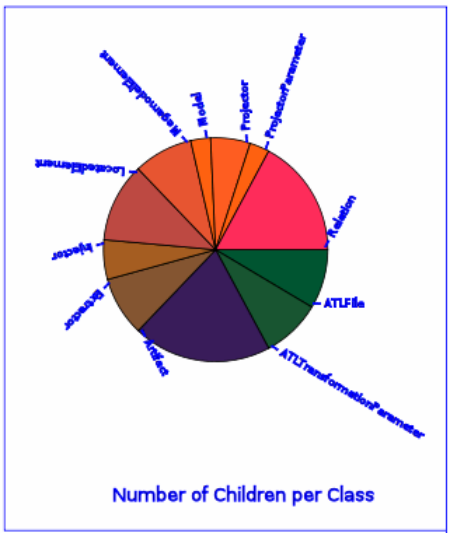

Figure 5: Pie chart with a sector for each value of the metric Number of Children for each class

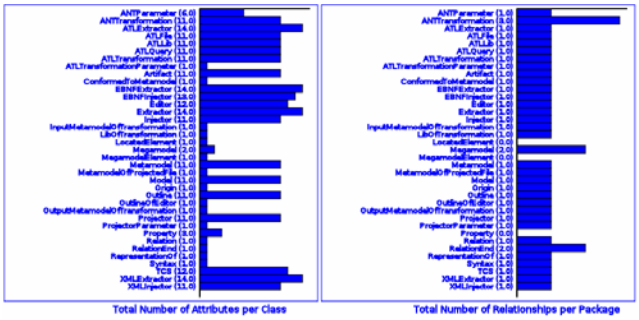

Figure 6: Bar chart with the metric Total Number of Attributes per Class and the second with the metric Total Number of Relationships per Package

4. CONCLUSIONS

We are working in an area where we need experimental data to perform research investigations. This has been the initial goal of the AM3 project: setting up an open source repository for models of various types. We have found the Eclipse community and support to be of great help in this. The repository of models is now evolving in different directions and available to the model engineering research community for various experiments.

We have proved that any verification or measurement of a model may be expressed as a transformation in ATL, a QVT-like model transformation language developed at INRIA. This paper has reported on the first metrication campaign performed with ATL on the AM3 zoo of metamodels. This campaign will be followed by many others and has already allowed a deeper understanding of several issues. Among them for example is the issue of metamodel dependent and independent expression of measures.

5. ACKNOWLEDGEMENTS

We would like to thank all the members of the ATLAS team for their help in this ongoing work, and specially Patrick Valduriez, Ivan Kurtev and Guillaume Hillairet. This work has been partially supported by ModelPlex, an IST European project.

6. REFERENCES

[1] **AM3 "ATLAS MegaModel Management"** official site (**Eclipse/GMT** subproject): http://www.eclipse.org/gmt/am3/

[2] **AMW "ATLAS Model Weaver"** official site (**Eclipse/GMT** subproject): http://www.eclipse.org/gmt/amw

[3] **ANT**, **The Apache Ant Project**: http://ant.apache.org/

[4] **ASM (Abstract State Machine)**, A Formal Method for Specification and Verification: http://www.eecs.umich.edu/gasm/

[5] **AsmL (Microsoft Abstract State Machine Language)** official site: http://research.microsoft.com/fse/asml/

[6] **ATL "ATLAS Transformation Language"** official site (**Eclipse/GMT** subproject): http://www.eclipse.org/gmt/atl/

[7] Bézivin, J, and Jouault, F : **Using ATL for Checking Models**. In: Proceedings of the International Workshop on Graph and Model Transformation (GraMoT), Tallinn, Estonia. 2005

[8] Bézivin, J., Hillairet, G., Jouault, F., Kurtev, I., and Piers, W., **Bridging the MS/DSL Tools and the Eclipse Modeling Framework**. In: Proceedings of the International Workshop on Software Factories at OOPSLA 2005, San Diego, California, USA.

[9] Bézivin, J., Jouault, F., and Valduriez, P., **On the Need for Megamodels**. In: Proceedings of the OOPSLA/GPCE: Best Practices for Model-Driven Software Development workshop, 19th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications.

[10] Bézivin, J., Jouault, F., **KM3: a DSL for Metamodel Specification**. In: Proceedings of 8th IFIP International Conference on Formal Methods for Open Object-Based Distributed Systems, Bologna, Italy.

[11] Bézivin, J., Jouault, F., Kurtev, I., and Valduriez, P., **Model-Based DSL Frameworks**. Submitted for Publication.

[12] Bézivin, J., Jouault, F., Rosenthal, P., and Valduriez, P., **Modeling in the Large and Modeling in the Small**. In: Proceedings of the European MDA Workshops: Foundations and Applications, MDAFA 2003 and MDAFA 2004, LNCS 3599, edited by Uwe Aßmann, Mehmet Aksit, Arend Rensink. Springer-Verlag GmbH, pages 33-46.

[13] Bézivin, J., Kurtev, I., **Model-based Technology Integration with the Technical Space Concept**. In: Metainformatics Symposium 2005, Esbjerg, Denmark, November 2005, to be published in LNCS volume

[14] **DSL Tools** for **Visual Studio 2005**, **Microsoft** official site: http://msdn.microsoft.com/vstudio/dsltools/default.aspx

[15] Kurtev, I., Bézivin, J., Aksit, M., **Technical Spaces: An Initial Appraisal**. CoopIS, DOA'2002 Federated Conferences, Industrial track, Irvine, 2002 http://www.sciences.univ-nantes.fr/lina/atl/publications/

[16] Genero M., Piattini M. and Calero C., **Early measures for UML class diagrams.** L'Objet. Volume 6 – No. 4/2000