ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **14**, Number 1, 2011, 51–66

Automatic Transformations for Integrating Instrument Models across Technological Spaces

Anca Daniela IONITA, Adriana OLTEANU,
Traian IONESCU, Liliana DOBRICA

Department of Automatic Control and Industrial Informatics,
"*Politehnica*" University of Bucharest,
No. 313 Splaiul Independentei, 060042 Bucharest, ROMANIA
E-mail: {anca.ionita, adriana, liliana}@aii.pub.ro, tcion@rnc.ro

Abstract. The paper presents a model driven approach for managing non-homogenous information coming from various sources: sensors, scientific equipment, or databases. Experimental data are considered together with their measurement context, which is modeled, presented and processed based on several technological / modeling spaces, giving more insight for multiple actors: metrologists, instrument evaluators, students, educators. Automatic model transformations have been introduced in order to integrate their work, based on different tools and models.

1. Introduction

The heterogeneity of devices in industry and in real-life situations involves heterogeneity of data and models, so the necessity to integrate them. Moreover, designing complex systems involves multiple actors, who need multiple kinds of modeling and analysis, corresponding to multiple points of view. This leads to the necessity to integrate non-homogeneous data, but also to transform models from certain formalism to another [22], or to compose models [3], views [32] and/or modeling languages [8].

An emerging field having this concern is Computer Automated Multi-Paradigm Modeling (CAMPaM), related to complex system design, which addresses three dimensions for the integration of models: abstraction on multiple levels, multi-formalism

modeling, and meta-modeling – as a way to model the formalisms themselves [25]; the glue between these orthogonal directions is considered to be the automated model transformation.

Model transformations play an important part in model driven development, because they facilitate model manipulation. By establishing relationships between modeling elements, one can generate a model, based on information contained in another model. Model transformation represents a mechanism for solving a large variety of problems, in various phases of the development process, and for various tasks, like modification, creation, adaptation, union, combination, or filtering [5].

Some well-known definitions of model transformations are:

- "a program that mutates one model into another", according to Tratt [30];
- "the process of converting a model into another model of the same system", according to Miller et al. [23];
- "automatic generation of a target model from a source model, according to a transformation description", from the paper of Kleppe et al. [18];
- "automatic generation of one or multiple target models from one or multiple source models, according to a transformation description", as described by Mens and Van Gorp [22], in order to extend the definition given by Kleppe.

Also, the Query-View-Transformation (QVT) standard [27], adopted by OMG, defines three language levels, two for declarative purposes (the core, supporting pattern matching and the relations, for matching complex object patterns) and an operational mappings language for providing imperative implementations .

This paper presents a series of model transformations realized for EquiLAB, a data and model integration environment dedicated to the domain of measurement instruments, including various apparatus, equipment, acquisition or sensor systems. Section 2 describes the modeling spaces adopted in our approach, in order to render contextual models, which describe the settings related to the experiment and to the measuring instruments. As this information is used by multiple actors, like metrologists, tool integrators, evaluators, or training personnel, we define multiple models, written in different modeling languages. In this context, automatic transformations between these models are essential for getting efficiency and increasing portability and quality [21]. Section 3 presents transformations between EquiLAB and two other spaces: XML (eXtensible Markup Language) and GME (Generic Modeling Environment). Section 4 frames EquiLAB transformations into various classification schemes existent in literature.

2. Modeling spaces for instrument integration

2.1. Theoretical Foundations

Increasing the level of abstraction represents a solution for integrating heterogeneous artifacts, like data, software, or models. This principle was applied, for

example, for integrating heterogeneous software artifacts [8], like components, legacy applications, services or COTS (Components Off The Shelf) or even physical devices.

The importance of defining multiple levels of abstraction has been generally recognized starting with the adoption of MDA (Model Driven Architecture) [21], for which Object Management Group (OMG) has specified standard languages like MOF (Meta Object Facility) and UML (Unified Modeling Language). Separating the design of PIM (Platform Independent Model) from that of PSM (Platform Specific Model) allows the migration of the application logic towards PIM, which represents an important support for reuse and interoperation. This approach has been extended from the four MDA levels to an unlimited number of abstraction levels [2] and was applied in multiple technological spaces [19], or modeling spaces [12], like object oriented programming, XML representation, and relational databases.

A similar theory is encountered in domain modeling, by defining a conceptual architectural model valid for all possible applications pertaining to the domain of interest, as performed in product line engineering and generative programming [6]. A domain model defines concepts (classes of elements) and relationships between them. It is formalized through a metamodel, which serves as a language for modeling (programming) applications in the domain of interest. This metamodel defines the abstract syntax of a Domain Specific Language (DSL) [17] expressing the fundamental, high level concepts. The ideas and the terminology are similar to MDA, but belong to the more general area of Model Driven Engineering (MDE) [10]. A domain model is based on the domain ontology, which classifies and inter-relates its elements [9] and/or on the domain variations, meaning the differences between one application and another [15].

In this paper, we apply the abstraction principle with the purpose of uniformly treating a large variety of measurement devices; the integration is twofold: that of the measured data and that of the acquisition context models described with different formalisms.

2.2. Description of EquiLAB approach

The idea of EquiLAB is to integrate data coming from various sources and to make them available in unitary formats for being further processed. Unlike major integration approaches, this one is not based on data models only, but it creates a framework for defining the context of measurements, such as to carry out repeatable scientific experiments and to compare results on rigorous basis. Thus, a measurement is characterized through the following elements:

- the measured object (sample, probe, distributed system, configuration of sensor network);
- the settings performed before acquiring data, regarding the instrument or the experimental setup;
- the measurement results, which are data representing amounts of various physical properties.

The objectives of our approach are:

– to represent the measurement context by creating a metamodel for the measurement domain, presented in [14] and defining a conform model for each measurement instrument;

– to integrate data coming from apparatus or more complex equipment, as well as other experimental or industrial settings [16];

– to offer a framework for evaluating various instruments, in order to perform more acquisitions;

– to give a unitary educational support for trainers and university teachers, for the presentation of a computerized laboratory [26].

The data integration application is based on the interpretation of EquiLAB models created with a Web editor, conforming to a metamodel that abstracts the data and the settings of various instruments; an example for modeling a digital multimeter is presented in Fig. 1. The measurement instruments are modeled using groups of parameters like: InstrumentSetup, Data, Measurement, Experiment and Warnings. Each parameter is characterized by its name, type and measurement unit and is saved in a database.

Thus, one creates a modeling space accessible to the metrologists' background, because they are not accustomed with graph based design, but can simply fill in forms containing specific parameters and their measurement units. Moreover, the environment supports the integration with other modeling spaces (presented in the subsection 2.3) more appropriate for actors like trainers, tool integrators or equipment evaluators.

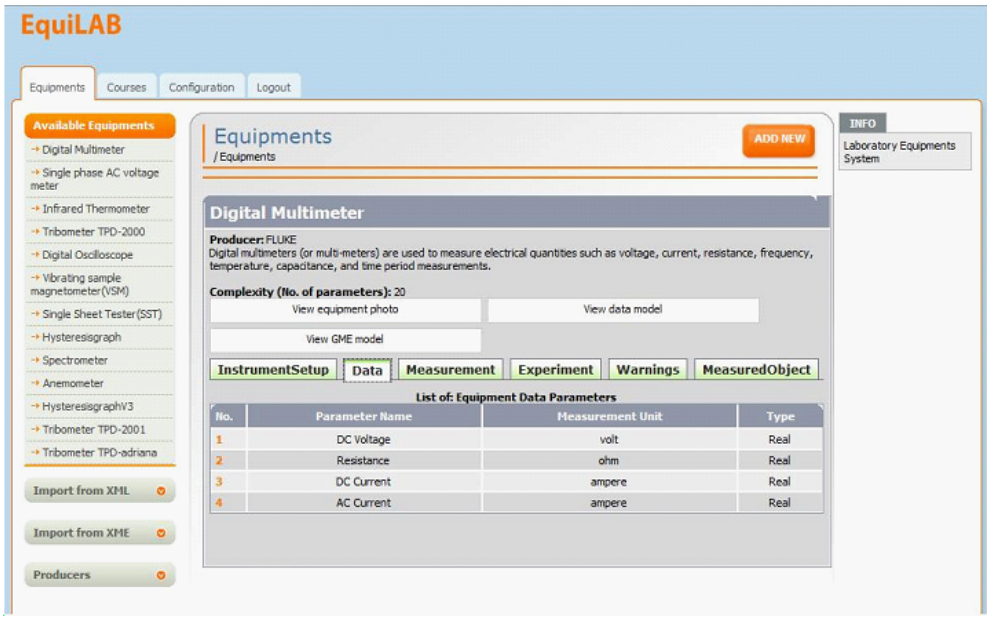

Fig. 1. EquiLAB model for a digital multimeter.

2.3. Presentation of EquiLAB Modeling Spaces

EquiLAB was initially conceived as a Web application that supports the integration of data based on instrument models conforming to a specific metamodel and stored in XML. Later, one added conceptual models represented with two other languages:

- the general modeling language UML;
- a DSL for the Measurement Domain, realized with GME (Generic Modeling Environment).

Instead of re-implementing everything, we delegated some functionalities to other modeling spaces, like UML and GME, which already have a strong tool support, including open source solutions. Therefore, our approach is based on inter-formalisms model-to-model transformations, which allow us to take advantage of the possibilities of modeling, visualization and interpretation offered by existent environments. Thus, besides data integration facilities, EquiLAB becomes an integrated modeling environment.

Table 1. Technological Spaces for the Measurement Domain

Technological/ Modeling Spaces	**Meta-metamodel (M3 level)**	**Metamodel (M2 level)**	**Model (M1 level)**
EquiLAB	UML	EquiLAB metamodel	Web EquiLAB model
MDA	MOF	UML metamodel	UML model
XML	EBNF	Schema or DTD	XML Document
GME	MetaGME Paradigm	Instruments Paradigm	Instruments Model

Thus, instrument integration is approached with a combination of technological spaces, used for implementation or for modeling purposes, all of them having the following 4 abstraction levels: the real-world, its representation through a model, the metamodel (or the modeling language) and a meta-metamodel. As summarized in Table 1, the approach integrates:

- EquiLAB – with a specific metamodel described in UML, allowing the definition of instrument models through a Web editor based on tags for each class of modeling elements; this space is dedicated to domain experts, like metrologists working in an industrial or an academic laboratory;
- MDA – based on MOF meta-metamodel [27] and on UML [28]; it has a large variety of free or proprietary editors and it is well known in academic communities, so it is a space introduced for educational purposes; the object oriented conceptual models - expressed in a standard, general modeling language - can fasten the understanding of new instruments;

– XML – based on XML documents that conform to an XML schema or to its precursor, the DTD language (Document Type Definition) [29]; at M3 level there is EBNF (Extended Backus–Naur Form) - the metasyntax, context-free grammar, describing computer programming languages; XML is portable, standardized and it is often used for integration purposes. In our case, the language utilization is twofold: i) for the persistency of instrument models and ii) for allowing programmers to assure interoperability with other tools, pertaining to various technological spaces;

– MetaGME – is based on MetaGME paradigm, allowing one to define specific metamodels and generating the correspondent model editors [20, 24]; the *Instruments* paradigm was especially, defined for the purpose of analyzing various aspects of the instruments and of comparing them according to various criteria, in order to evaluate the complexity of the instrument and of its embedded software; thus, the tool evaluator can work at his or her appropriate level of abstraction; the environment generates editors, which can be configured with a suggestive concrete syntax.

MDA and MetaGME were chosen for giving supplementary facilities for graphical modeling and for evaluating these models, while taking advantage of the existent tools. Consequently, MDA and MetaGME can effectively be called modeling spaces, as in the approach of Gašević et al. [12]. XML is a space particularly concerned with model persistence and, besides its legibility, was also introduced for implementation reasons, and for facilitating transformations, as further described in the next section. So, it is also used as a technological space, in the sense of Bezivin's approach [3].

For a better integration of these spaces, one needs to automatically transform the models situated on M2 level from one technological space to another, as illustrated in Fig. 2. In the following section, the paper presents mappings and examples for three of them: EquiLAB, MetaGME and XML, for integrating the work of metrologists, evaluators and programmers who integrate tools.

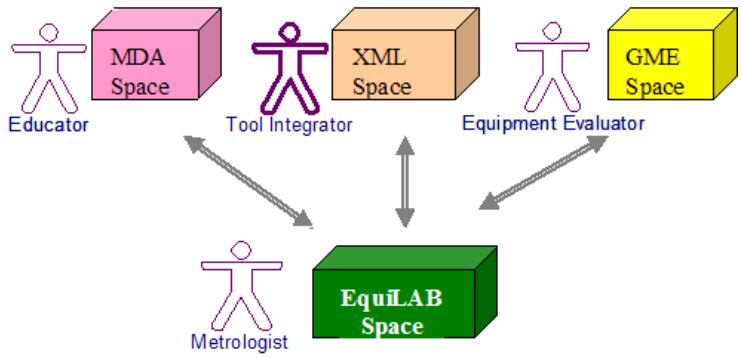

Fig. 2. Integration of modeling spaces used for the domain of measurement instruments.

3. Transformations between EquiLAB, XML and GME models

This section presents the transformations introduced for integrating EquiLAB, XML and MetaGME modeling spaces (see Fig. 3.). The MetaGME – XML transformation is supported by the GME tool, in both directions. The graphical GME model can be exported / imported to /from a file with .xme extension, containing information related to the modeling elements correspondent to a specific modeling paradigm (in our case *Instruments*).

The EquiLAB – XML transformation (T1) was developed based on the correspondences between the EquiLAB metamodel and an XML schema defined for his purpose. It is an inter-formalism transformation.

The MetaGME – EquiLAB transformation is performed sequentially. Given a graphical model of an instrument, realized in GME, it is exported into an .xme file, pertaining to the XML space. Then, it is transformed into an .xml file conforming to our schema, applying an intra-formalism transformation inside the XML space (T2). Subsequently, it is imported into EquiLAB , allowing one to visualize, for the same instrument, another model, having the style close to a spreadsheet organized by groups of parameters.

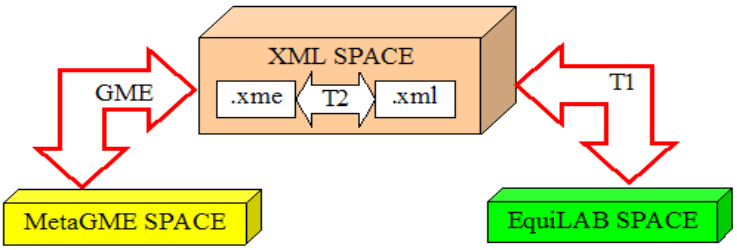

Fig. 3. Transformations between EquiLAB, XML and GME modeling spaces.

The transformations of models between EquiLAB, XML and MetaGME spaces are bidirectional. It is also possible to parse the other way around, and to transform a model defined by the metrologists in EquiLAB, into a graphical GME model, more suggestive for the tool evaluator.

3.1. Mappings at the metamodel level

Automatic model transformators can lower the time and cost of various related activities and, for being able to write them, the explicit knowledge of metamodels is very important. The transformations introduced between EquiLAB, XML and MetaGME spaces are horizontal, so it is necessary to establish correspondences between the metamodels to whom they conform, situated at MDA M2 level of abstraction. According to Iacob's classification of transformation design patterns [13], we use a mapping pattern, for making straightforward transformations between modeling elements having "more or less" the same semantics, even if expressed if different languages.

Figure 4 shows some examples of mappings between modeling elements of the three spaces under discussion, representing similar, but not identical concepts. Some of them have got almost the same names, like the XML element called "parameter", mapped on the class "Parameter" from EquiLAB and on the atom similarly called from the MetaGME space. Still, the parameter descriptions are different, according to the necessities of each modeling space. In EquiLAB, they consist of their name, measurement unit and type (real, integer, string, or a user defined enumeration). In GME, the parameter is also characterized by the physical property to which it corresponds, the dimension (scalar or vectorial) and an implicit value – elements important for the evaluation of the measurement instrument.

The names of other correspondent elements are completely different, *e.g.*:

- "MeasurementItem" – from EquiLAB space – having an aggregation relationship with the class "Parameter";
- "group" – from XML space – representing a complex type that contains a name, an identificator and a sequence of parameters;
- "ParameterGroup" – from GME space – which is a model containing multiple "Parameter" atoms.

A fragment of code for our T2 transformator is given in Appendix 1.

However, our transformators are not exclusively based on mappings. As MetaGME and EquiLAB hierarchies are not identical, the transformations towards / from XML models are based on a flattening pattern [13]; the xml schema was conceived as for an interchange language, so T1 and T2 also include refinement / abstraction patterns related to some details, like the description of "Parameter".

3.3. Example of generated models for a multimeter

A multimeter is an electronic instrument dedicated to the measurement of voltage, current and resistance. Even if it is a simple apparatus, various models, coming from different producers, may differ by the range and the resolution of these physical properties, the measuring frequency range, the transmitter resistance, etc. Usually, this information comes in natural language, which makes comparisons between equipment models more difficult, especially if they are more complex and deal with many physical properties, some of them representing measured data, others representing settings performed on the instrument or for the specific measurement experiment.

An example of multimeter model expressed in EquiLAB space was given in Fig. 1. It is situated at MDA M1 level of abstraction and groups the information in several tables, accessible through multiple tabs. The fragment of the correspondent model in XML space, obtained after T1 transformation, is given in Fig. 5. Also, a fragment of the GME model is represented in Fig. 6; it was obtained with a chain of automatic transformations: T1 from EquiLAB to .xml model, T2 from .xml to .xme model, and then GME import.

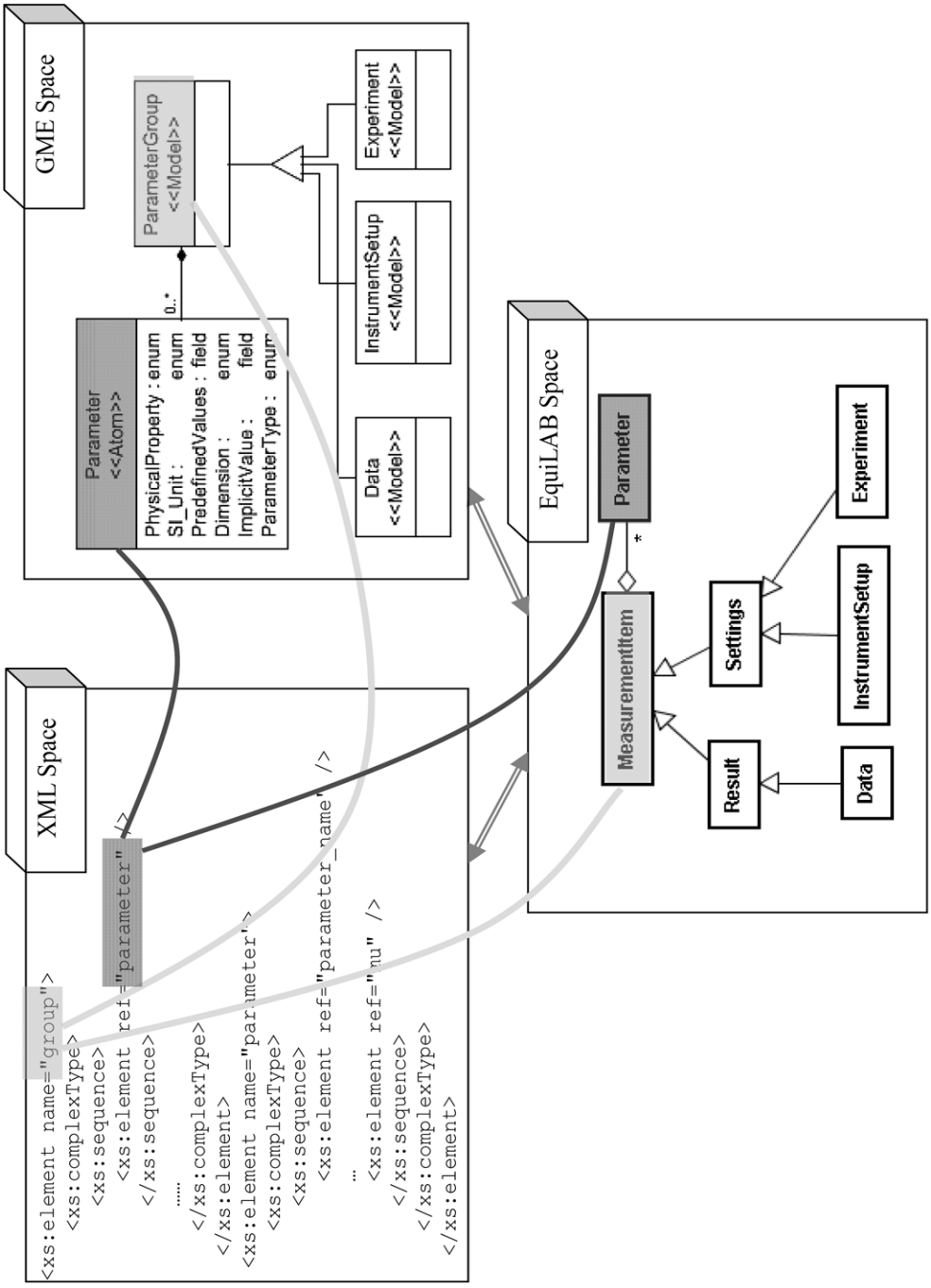

Fig. 4. Mappings between EquiLAB, XML and GME metamodels.

```
- <equipment>
  <name>Digital Multimeter</name>
  <producer>FLUKE</producer>
  <description>Digital multimeters are used to measure electrical
quantities such as voltage, current, resistance, frequency </description>
- <parameters>
- <group id="2" name="Data">
- <parameter>
  <parameter_name>DC Voltage</parameter_name>
  <parameter_data_type>Real</parameter_data_type>
  <parameter_type>Base Parameter Type</parameter_type>
  <mu>volt</mu>
  </parameter>
- <parameter>
  <parameter_name>Resistance</parameter_name>
  <parameter_data_type>Real</parameter_data_type>
  <parameter_type>Base Parameter Type</parameter_type>
  <mu>ohm</mu>
  </parameter>
- <parameter>
  <parameter_name>DC Current</parameter_name>
  <parameter_data_type>Real</parameter_data_type>
  <parameter_type>Base Parameter Type</parameter_type>
  <mu>ampere</mu>
```

Fig. 5. Fragment of the XML Model for a Multimeter.

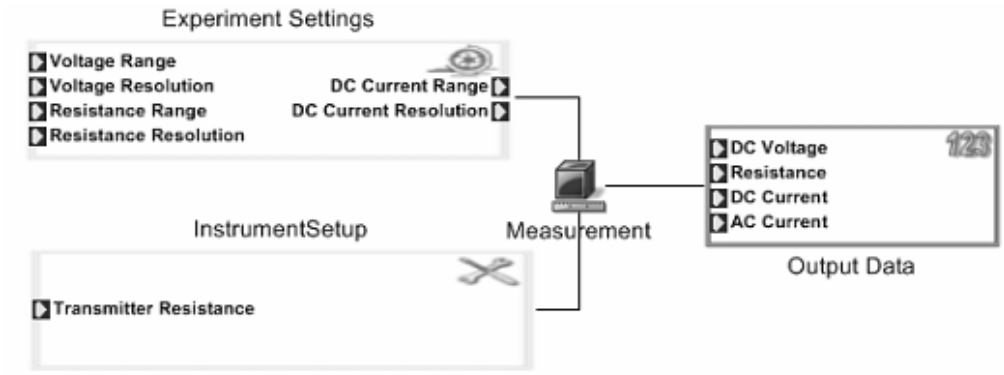

Fig. 6. Fragment of the GME Model for a Multimeter.

4. Our position in existent transformation classifications

We presented before the transformations from our domain specific syntax and semantics, EquiLAB, to the textual XML and the graph-based GME formalisms. This section discusses how these transformations fit into existent classifications of model transformations. First, we have to outline that there is no consensus in this

domain and there have been multiple attempts to define the taxonomy of model transformations [7][22][5][25].

We shall analyze our T1 and T2 transformations according to 5 criteria regarding the elements that are modified during the transformation, summarized by Biehl [5]:

– level of abstraction;
– metamodel;
– technological spaces;
– number of models;
– content kind.

The model transformation can change the *level of abstraction* between source and destination models, determining the quantity of details presented in the model. A transformation preserving the level of abstraction is called horizontal, and one modifying it, by refinement or abstraction, is called vertical [22]. Well-known examples are: (i) refactoring, as a horizontal transformation [11], and (ii) refinement, as a vertical transformation [33][1]. Our T1 and T2 transformations are horizontal, because the models are both situated at M1 MDA level.

From the point of view of *source and destination metamodels*, there are (i) endogenous and (ii) exogenous model transformations [22]. The former is a transformation between models expressed in the same language. Usually, while producing the destination model, one only modifies specific parts of the source one. This kind of transformation is called rephrasing transformation in [31]. The latter is a transformation between models written in different languages, but preserving the same level of abstraction. It is also called translation transformation [31]. We deal with exogenous transformations in our approach: T1 between EquiLAB and XML, and T2 between models written in XML with different schemas.

Model transformation can go across several technical and *technological spaces* [4] [22] like MDA, MetaGME and XML, presented for our application in **??**. The utility is that one can use tools existent in different spaces, which complement each other, and pass smoothly from one space to another. More precisely, transformation T2 is performed inside XML space, and T1 is between EquiLAB and XML spaces. The transformation between MetaGME and XML spaces is supported by the GME environment. These transformations across technological spaces are similar to transformations into another formalism, identified by Mosterman in multiparadigm modeling, along with two other kinds of model transformations, for execution and for optimization [25].

Another criterion of analysis is *the number of source and destination models* involved in the transformation [22]. From this point of view, the minimalist approach is the in-place transformation [7], where a single model represents the source and the destination also; it is based on the modification of specific parts of the source model. However, most transformations conform to the classical definition given by Kleppe et al. [18] and involve distinct models for the source and the destination, which is also our case. The assumption is that the destination model is empty and it will contain elements that will be explicitly generated through the transformation. More generally, it is possible to have multiple sources and / or multiple destinations, which reference each other. Our approach could also be adapted such as to consider a source model

pertaining to one of the modeling spaces and to perform a composite transformation (T1 + T2) such as to generate two destination models, pertaining to the other two spaces.

Regarding the *content type*, Czarnecki et al. introduce the two major categories [7]: model-to-model and model-to-code (with the model-to-text generalization, if the destination content is not written in an executable language). From this point of view, T1 is actually a model-to-text transformation, as the destination is expressed in XML. Also, a model-to-model transformation from EquiLAB to MetaGME is actually composed of a chain of transformation: T1 + T2 + GME import / export. To realize this, we combine a "relational approach", with a non-executable specification of the mapping rules between modeling elements pertaining to different technological spaces and a "direct-manipulation approach".

5. Conclusions

The paper showed how metamodels can help for managing heterogeneous data coming from various sources, like scientific equipment, measurement instrumentation and acquisition systems, supporting embedded software working with different data formats. In this context, our scope is twofold: to integrate data from different instruments and, for a given instrument, to integrate models that describe it in multiple modeling spaces, dedicated to various actors, like metrologists, educators, evaluators and developers. The purpose is to capture not only data models, but also information characterizing their context, so to obtain repeatable scientific experiments.

The automatic transformations between models pertaining to XML, MetaGME and EquiLAB spaces are essential for the efficiency of migrating between various environments and tools. They are horizontal exogenous transformations, performed at M1 level in MDA, corresponding to the classical definition of generating a destination model from a source one and crossing several technological spaces. Even if the final purpose is to obtain model-to-model transformations, they also contain intermediate steps whose type is actually model-to-text.

The implementation of transformations was performed for well defined and stable specifications at the metamodel level, involving the XML schema, the EquiLAB metamodel and the GME paradigm for instruments. An open issue is to assure the evolution of transformations according with the evolution of these metamodels; it is an important future challenge, along with model and metamodel evolution, including their version control.

Acknowledgements. The work on model transformations was supported by CNCSIS - UEFISCSU, Romania, project number PNII – IDEI 1238/2008.

Appendix 1: Fragment of code for T2 transformation

The transformation of .xml models to .xme ones, which can be imported in GME, is accomplished through a database realized with MySQL. The database plays an interchanging role, because it stores data in a uniform way, so they can be further exported to other formats.

Import from .xml models to the database

```
$original_file_name = $path_parts['filename'];
$original_file_extension = $path_parts['extension'];
$uploadfile = "../import/" . $original_file_name . "." .
$original_file_extension;
...
Validate schema
$xmlSchema = 'schema/import_xml.xsd';
//Validate the XML file against the schema
if (!$xmlDoc->schemaValidate($xmlSchema))
...
$name = mysql_real_escape_string($xmlDoc->getElementsByTagName("name")
->item(0)->nodeValue);
...
Insert the data into the tables:
$query = "insert into t_eqp_equipments (prd_number, eqp_name, eqp_description,
eqp_website, eqp_photo, eqp_modelphoto, eqp_gmemodelphoto, usr_number,
eqp_creationdate, eqp_modifieddate)
values (" . $prd_number . ",'" . $name . "','" . $description . "','" .
$website . "','','','," . $_SESSION['usr_number'] . ",SYSDATE(),SYSDATE())";
...
```

Export from the database to .xme models

```
Select the data from database:
$query = "select epr.*, prt_name, prg.prg_number, prg_name,
prg_gmegroupname, top_name, mu_name
from  t_epr_equipmentparameters epr
     join t_prt_parametertype prt on epr.prt_number = prt.prt_number
     join t_prg_parametergroups prg on prg.prg_number = epr.prg_number
     join t_top_typeofparameters top on top.top_number = prt.top_number
     join t_mu_measurementunit mu on mu.mu_number = epr.mu_number
          where eqp_number = " . $eqpno . "
          order by prg_name, prt_name";
...
Building the model of the GME file, XME:
$xml_output .= chr(9) . '<model id="' . $ModelId . '" kind="' .
fctParseIllegalCharacters($row['prg_gmegroupname'])  . '" role="' .
fctParseIllegalCharacters($row['prg_gmegroupname'])  . '" guid="{' .
fctGetGUID() . '}" relid="0x1" childrelidcntr="0x4">' . chr(13);
$xml_output .= chr(9) . '<name>' . fctParseIllegalCharacters($row["prg_name"]).
'</name>' . chr(13);
$xml_output .= chr(9) . '<regnode name="PartRegs" status="undefined">
      <value></value>
      <regnode name="DataModel" status="undefined">
      <value></value>
      <regnode name="Position" isopaque="yes">
      <value>252,91</value>
      </regnode>
```

```
      </regnode>
...
$ConnectionsOutput .= '<connection id="' . fctGetItemID() .
'" kind="Connection"
role="Connection" guid="{' . fctGetGUID() . '}" relid="0x13">
      <name>Connection</name>
      <regnode name="autorouterPref" isopaque="yes">
      <value>Ew</value>
      </regnode>
      <connpoint role="dst" target="' . $ModelId . '"/>
      <connpoint role="src" target="' . $MeasurementId . '"/>
      </connection>' . chr(13);
...
$xml_output .= chr(9) . chr(9) . '<model id="' . $MeasurementId .
'" kind="Measurement" role="Measurement" guid="{4a563bbd-ba6e-4e50-ae5b-
a9f2a8520bce}" relid="0x9" childrelidcntr="0x0">
       <name>Measurement</name>
       <regnode name="PartRegs" status="undefined">
       <value></value>
       <regnode name="MeasurementFramework" status="undefined">
       <value></value>
       <regnode name="Position" isopaque="yes">
       <value>475,286</value>
       </regnode>
       </regnode>
       </regnode>
       </model>' . chr(13);
$xml_output .= chr(9) . chr(9) . $ConnectionsOutput;
...
```

References

[1] Back R. J., Von Wright J., *Refinement Calculus*, Springer Verlag, 1998.

[2] Batory D., *Multi-Level Models in MDD, Product-Lines & Metaprogr.*, IBM Systems Journal, vol. **45**, no. 3, 2006.

[3] Bezivin J., Bouzitouna S., Didonet Del Fabro M., Gervais M. P., Jouault F., Kolovos D., Kurtey I., Paige R. F., *A Canonical Scheme for Model Composition*, *Model Driven Architecture – Foundations and Applications*, Lecture Notes in Computer Science, vol. **4066**, pp. 346–360, 2006.

[4] Bezivin J., Jouault F., Valduriez P., *On the need for megamodels*, *Proceedings of Workshop on Best Practices for Model-Driven Software Development at the 19th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications*, Vancouver, British Columbia, Canada, October 2004. Available: `http://www.softmetaware.com/oopsla2004/mdsd-workshop.html`

[5] Biehl M., *State of the Art in Model Transformation Technology*, July 2010, Available at: `http://www.md.kth.se/~biehl/files/papers/mt.pdf`

[6] Czarnecki K., *Overview of Generative Programming*, UPP'04 – Mont Saint-Michel, France, 2004.

[7] Czarnecki K., Helsen S., *Classification of Model Transformation Approaches*, *OOPSLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture.*

[8] Estublier J., Vega G., Ionita A. D., *Composing Domain-Specific Languages for Wide-Scope Software Engineering Applications*, Lecture Notes in Computer Science, vol. **3713**, pp. 69–83, ISSN 0302-9743, Springer-Verlag, 2005.

[9] Falbo R. A., Guizzardi G., Duarte K. C., *An ontological approach to domain engineering*, *Proc. of the* 14^{th} *Int. Conf. on Software Eng. and Knowledge Eng.*, Ischia, Italy, pp. 351–358, 2002.

[10] Favre J. M., *Towards a Basic Theory to Model Driven Engineering*, WiSME 2004.

[11] Fowler M., *Refactoring: Improving the Design of Existing Programs*, Addison-Wesley, 1999.

[12] Gasevic D., Djuric D., Devedzic V., *Model Driven Archit. and Ontology Development*, Springer-Verlag, 2006.

[13] Iacob M. E., Steen M. W. A., Heerink L., *Reusable model transformation patterns*, *Enterprise Distributed Object Computing Workshops, International Conference on*, vol. **0**, pp. 1.

[14] Ionita A. D., *Domain Models for Laboratory Integration*, *Proc. Of the 7th WSEAS Int. Conf.on Software Engineering, Parallel and Distributed Systems (SEPADS '08)*, University of Cambridge, UK, Feb. 20–22, 2008, ISSN: 1790-5117, pp. 119–123.

[15] Ionita A. D., Estublier J., Levenque Th., Hguyen T., *Bi-dim.Comp. with DSLs*, e-Informatica Journal, **3**, 1, 2009.

[16] Ionita V., Ionita A. D., *Architecture for Integrating Data Obtained by Advanced Characterization of Magnetic Materials*, Romanian Journal of Materials, ISSN 1583-3186, **38** (1), pp. 69–75, 2008.

[17] Kelly S., Tolvanen J. P., *Domain-Specific Modeling*, Wiley-IEEE Computer Society Press, 2008.

[18] Kleppe A., Warmer J., Bast W., *MDA Explained: The Model Driven Architecture: Practice and Promise*, Addison-Wesley, 200345, no. 3, 2006.

[19] Kurtey I., Bezivin J., Aksit M., *Technological Spaces: an Initial Appraisal*, *Proc. of Confederated Conf. CoopIS, DOA and ODBASE, Industrial Track*, Irvive, CA, 2002.

[20] Ledeczi A., Maroti M., Bakay A., Karsai G., Garrett J., Thomason C., Nordstrom G., Spinkle J., Volgyesi P., *The Generic Modeling Environment*, *Workshop on Intelligence Signal Processing*, 2001.

[21] Mellor S., Scott K. S., Uhl A., Weise D., *MDA Distilled: Principles of Model-driven Archiecture*, Addison-Wesley, 2004.

[22] Mens T., Van Gorp P., *A taxonomy of model transformation*, Electr. Notes Theor. Comput. Sci, vol. **152**, pp. 125–142, 2006.

[23] Miller J., Mukerji J., *MDA guide version 1.0*, May 2003.

[24] Molnar Z., Balasubramanian D., Ledeczi Á., *An Introduction to the Generic Modeling Environment.* Available at: `http://www.dsmforum.org/events/MDD-TIF07/GME.2.pdf`

[25] Mosterman P. J., Vangheluwe H., *An Introduction to Computer Automated Multi-Paradigm Modeling,* 2004. Available at: http://www.thesimguy.com/GC/papers/scsgc-01.pdf

[26] Olteanu A., Ionita A. D., Ionescu T., *Leveraging Open Source E-learning Systems with Web 2.0 and Knowledge Structures*, (accepted and will be published in) Scientific Bulletin, University *"Politehnica"* of Bucharest.

[27] OMG, *MOF 2.0 query / view / transformation*, OMG, Tech. Rep., December 2009. Available at: http://www.omg.org/spec/QVT

[28] OMG UML Modeling Language (OMG UML). Available at: http://www.omg.org/spec/UML/2.3/

[29] OMG XML Specification. Available at: http://www.omg.org/technology/xml/

[30] Tratt L., *Model transformations and tool integration*, Software and Systems Modeling, vol. **4**, no. 2, pp. 112–122, May 2005. Available at: http://dx.doi.org/10.1007/s10270-004-0070-1

[31] Visser E., *A survey of rewriting strategies in program transformation systems*, *1st International Workshop on Reduction Strategies in Rewriting and Programming*, November 2001.

[32] Yie A., Casallas R., Deridder D., Wagelaar D., *A practical approach to multi-modeling views composition*, *Proc. Of the 3rd Int. Workshop on Multi-Paradigm Modeling, MPM 2009, Electronic Communications of the EASST*, vol. **21**, 2009.

[33] Wirth N., *Program development by stepwise refinement*, Comm. ACM 14, pp. 221–227, 1971.