7th WSEAS Int. Conf. on SOFTWARE ENGINEERING, PARALLEL and DISTRIBUTED SYSTEMS (SEPADS '08),
University of Cambridge, UK, Feb 20-22, 2008

Domain Models for Laboratory Integration

ANCA DANIELA IONITA
Computers and Industrial Informatics Department
University "Politehnica" of Bucharest
Spl. Independentei 313, 060042, Bucharest
ROMANIA

Abstract: - Laboratory equipment and software have a large degree of non-homogeneity and laboratory workflows are often based on manual, error-prone activities. The current trend is to realize integrated laboratories, supporting tool interoperation and data integration. A solution for hiding the non-homogeneity of tools and data is to raise the integration problem at a higher level of abstraction. The paper studies the domains that generally pertain to a laboratory performing measurements and tests, such as to obtain integration by composing their domain models. A data integration domain is presented here, whose design is based on the interpretation of specific, user-defined, laboratory models.

Key-Words: software design, domain modelling, model driven engineering, metamodels, data integration, laboratory integration, interoperation

1 Introduction

In laboratories performing tests and measurements, one works with text editors, data bases, spreadsheet editors, software embedded in measurement devices. In certain cases, like in accredited test laboratories [1], the workflow is well defined, but information is often passed from one program to another by humans. This also happens in research laboratories, where one uses new, up-to-date tools and also non-standard programs developed in-house. Laboratory integration, intended to introduce automate transfer of information and commands, deals with data integration and tool interoperation.

The existent systems for integrated laboratories are focused on the automation and integration of data acquisition from various devices; both raw data and those resulted from a primary processing are taken into account. This integration may be done either under a new, specific program, or under an existent LIMS (Laboratory Integrated Management System) (http://www.limsource.com).

Even if many solutions are available for integrating laboratories, the problem gets complicated if the devices and tools come from different producers, conform to different standards, or are not conceived for interoperation. The choice of a laboratory integration technique has to take into account three major criteria:

A) Program non-homogeneity – on one hand, most measurement tools produce specific, non-standard formats of data, incompatible with advanced processing tools; on the other hand, laboratories use a large variety of tools, making the task of direct inter-tool interoperation very complicated;

B) Code reuse - to minimize the interoperation effort, it is desirable to wrap the tools into software artefacts that support an easier interoperation*;* in this way, the complex code included in these wrappers may be reused;

C) Predicted adaptations - laboratory requirements may vary over time; LIMS may need to evolve, such as to support new tools and processes.

After studying the state of the art of tool interoperation (see [2]) one concludes that techniques based on models are expected to introduce a more flexible and efficient integration of laboratories. The idea is to avoid the direct communication between tools, as well as the direct inter-change of data. Instead, one should build a framework, customized on the basis of the laboratory model, such as to allow the interoperation of various tools, through the composition (synchronization) of the models that characterize them at an abstract level.

Our work started with the identification of integration and interoperation needs for the whole scale of laboratory systems; the current laboratory integration techniques just cover a small part of it (see chapter 2.1); this lead us to the use of a model-based approach (chapter 2.2), supporting the entire spectrum of non-homogenous tools, based on the composition of domain models (see [3], [4]). As the area is rather large, we have focused till now on the interoperation between measurement and processing tools; for facilitating it, a Data Integration domain

was developed, based on a metamodel intended to be general for any type of laboratory (see chapter 3.2). This domain is driven in respect with laboratory specific models and constitutes a preparation for a model-based interoperation between the programs embedded in the measurement (testing) devices and other tools for performing an advanced processing.

2 Domain Modelling for Integrated Laboratories

2.1 Integrated Laboratory Domains

For determining the needs of an integrated laboratory, the guidelines regarding the use of computers in accredited laboratories [1] have been studied. This lead to the identification of the following architectural elements (components, tools, packages etc.) required for performing the laboratory activities (see Figure 1):

- Measurement (containing data acquisition and processing),
- Accounting (related to payment facilities) and
- Activity Scheduler (dedicated for managing the laboratory workflow).

Research laboratories also have separate tools for *advanced processing*, so the architecture should be enriched with elements dedicated to this purpose.

Current integration techniques regard the Measurement component from Fig. 1, while others are generally left aside. There are other issues, like controlling instrument calibration, controlling tests [5], defining and managing standard procedures, which are not treated in an integrated way, or are entirely treated manually. Moreover, in research laboratories, new measurement and processing tools are supposed to be introduced rather often. They should also be integrated in LIMS and moreover, the flux of information should be controlled by computerized procedures.

A global solution of an integrated laboratory may be developed on a component framework, with the possibility to add new measurement devices, by wrapping them into components conforming to the framework model [6]. A possible architecture is described in Fig. 1. The advantage may be that such frameworks offer support for services like persistence or security.

However, it is highly probable that the elements identified above are not simple components conforming to the same component model, as assumed above. In the real world, there is not a single component for such an element, but there are several tools with different natures, which have been developed independently, with no awareness of each other and often without the intention to interoperate ever. The activity scheduler may use spreadsheet tools or project management tools; accounting is also based on data bases, text editors and spreadsheets, while their archiving implies a data base, a document manager or a version control system. Last but not least, measurement equipment involves a large variety of software for acquisition and primary processing. One can notice that in Fig. 1 there are not actually 4 components, but 4 complex domains [7], each one with its specific tools, which have been acquired on the market or developed in-house.

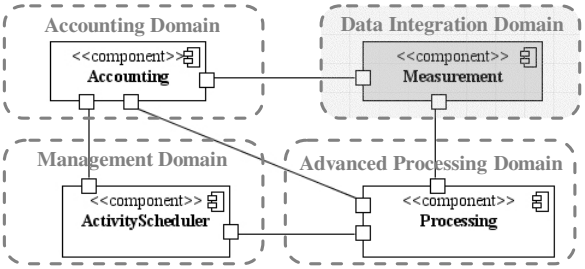

Fig. 1 Component Diagram for an Integrated Laboratory

2.2 Towards a Model Based Approach

Which technology may be appropriate to treat the complexity of an integrated laboratory and to deal with the lack of homogeneity? There are several ideas that arise from the current literature and practice and may be useful in this situation.

Idea 1 - Separate the platform independent elements. A lesson learned from MDA (Model Driven Architecture) is the definition of several layers of abstraction [8]. The separate design for PIM (Platform Independent Model) and PSM (Platform Specific Model) allows the interoperation logic to migrate at the level of PIM.

Idea 2 - Define application specificities through models. Product Line Engineering [9] analyzes domains from two points of view: identifying the commonalities (which may be supported by a common architecture) and identifying the variable points that may appear from one application to another - pertaining to the same domain (family). A possible way to define these

variations is by defining application specific models [10].

Idea 3 - Generate applications by defining their models. Getting one step further, generative programming aims to automatically generate such applications, by defining the variation points through textual or graphical specifications, which are actually done with domain specific languages, corresponding to the domain model [11].

Idea 4 - Define interoperation at a high level of abstraction. An essential aspect of interoperation, introduced by modelling, is that its logic may be explicitly expressed with the abstract model and eventually modified in the same way. This is the situation from Fig. 2, where the model coordinates the integration of laboratory tools, which may also be libraries, components or legacy software. The laboratory tools do not communicate directly, they do not know each other and their interoperation is coordinated by a model that may be easily changed, without modifying the whole integrator component.

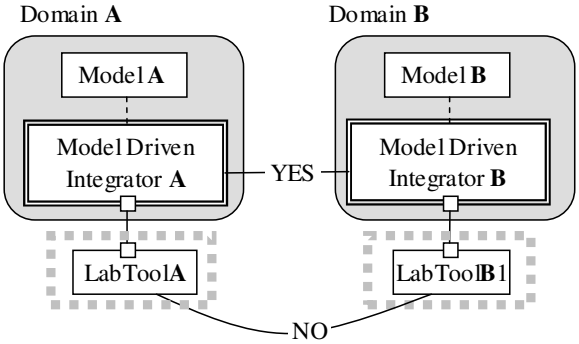

Fig. 2 Integration with model driven coordination

Laboratory integration may be performed by composing its domains. A possible solution is to establish relationships, while guarding the domain models and most of the code of the Model Driven Integrators untouched. The relationships may be written apart, using AOP (Aspect Oriented Programming) [3]. A separate specification of the interaction templates for domain collaboration may be encountered in [4].

3 Design of a Data Integration Domain

3.1 Software architecture

The work for laboratory integration based on the above presented approach has been first focused on the Data Integration domain, such as to prepare its composition with the Advanced Processing domain (see Fig. 3). The general idea was to keep the tools at a lower level and to allow the interoperation between model driven integrators. Let us discuss in detail how this architecture has assimilated the above presented ideas.

Applying idea 1. For *separating the platform independent elements*, the Integrator has two parts:

- the Tool Parsers, which are specific to the platform and may not be reused; they transform the data from various formats - specific to the measurement tools - to a common xml format, according to a schema correspondent to the laboratory model;
- the metamodel Interpreter, implemented in Java, which is platform independent and has been introduced to facilitate reuse; it performs data integration, storing, visualization and export, in accordance with the laboratory Measurement model and based on the general metamodel (domain model).

The parsers have to be written apart for any new tool that has to be integrated; the tools exemplified in Fig. 3 are specific for a magnetic material laboratory, for which this approach has been applied; they were presented in detail in [12].

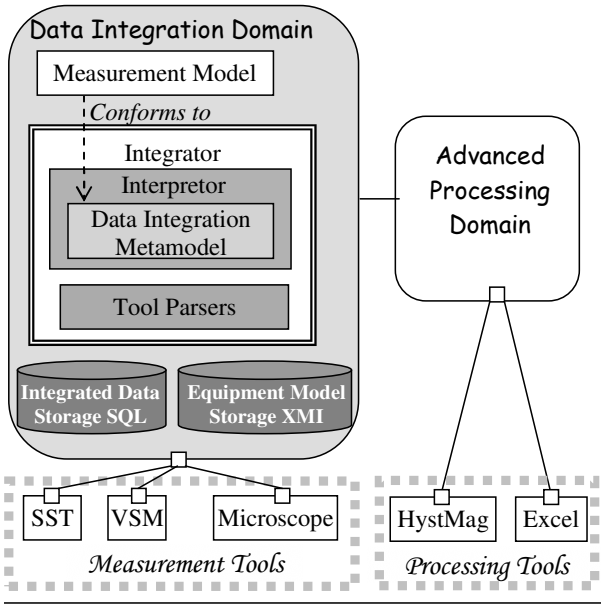

Fig. 3 Data Integration Domain

Applying idea 2. For *defining application specificities through models,* the Data Integration metamodel (presented in detail in the next section) is

not specific for the magnetic material laboratory (as the tools are) but is intended to grasp concepts and relationships that are common for any measurement laboratory.

Applying idea 3. Each laboratory has its specificities (which generate the necessity for tool specificities). *For generating applications by defining their models,* their variable part is captured by defining a Measurement model, conforming to the general metamodel. After editing the model, in respect with the specific measurement tools, it is just interpreted according to the metamodel; no supplementary coding is necessary.

Applying idea 4. This architecture has been conceived in order to *define integration at a high level of abstraction* and to allow an easy composition of the Data Integration domain with other laboratory domains, like Advanced Processing, or Activity Scheduler, which have to be studied and developed in the future.

3.2 Metamodel for the Data Integration Domain

The laboratory information, coming from varied equipment, may be modelled with the following concepts: Sample, EquipmentSetup, Experiment, ExperimentalData, ExperimentalCurve, pertaining to our metamodel. Each of them is characterised by certain parameters, which may differ in respect with the equipment type (see Fig. 4). These parameters are grasped into types that may be defined for each of the above mentioned concepts, with the editor presented in Fig. 5.

The methodology used for integration is based on the idea to store the parameters separately from their values. The parameters constitute the model of the equipment, which is defined once, at the beginning, using the model editor.

The implementation takes advantage of the EMF (Eclipse Modeling Framework) facilities for data integration, using EMF resources for creating the storage, based on XMI (XML Metadata Interchange) files. The parameter values for various samples, experiments, experimental data etc. are stored in an SQL database, any time a new measurement is performed with that equipment. In this way, the mechanism for storing the values is the same for any equipment, because it interprets the equipment model. In order to introduce new equipment, everything one has to do is to define its parameters with the model editor and to program some parsers, for converting the data from their particular format to the one conform to the metamodel.

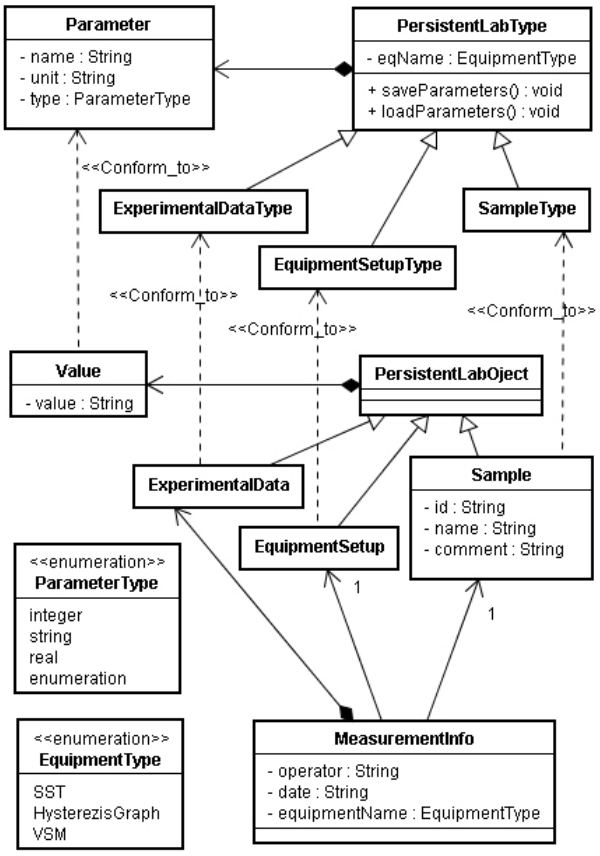

Fig. 4 Laboratory Data Integration Metamodel

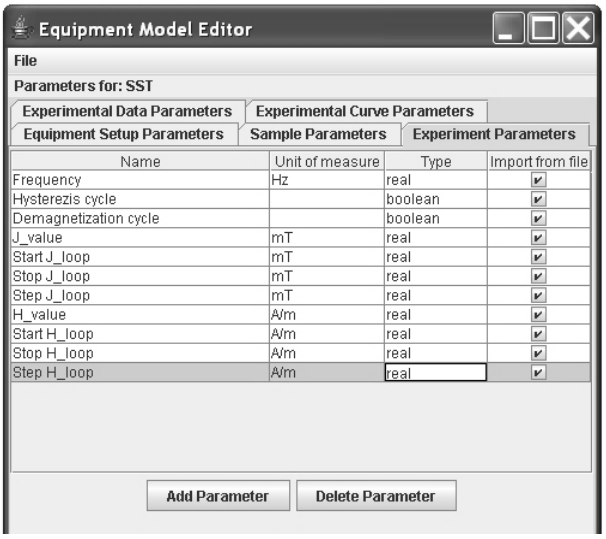

Fig. 5 Equipment Model Editor

This approach may be compared with MEMOPS framework [13], based on UML (Unified Modeling Language) for defining data models and tempting to be general for any scientific data modelling. However, the system presented here is based on different technologies; it uses models for integrating any kind of equipment, but also for improving the data accessibility to other programs for advanced processing (specialized software for electromagnetic field computation, like Hystmag, or spreadsheet tools like Excel – see Fig. 3).

4 Conclusion

The paper analysed the spectrum of laboratory software, such as to identify its domains. A model based approach has been proposed for taking over the challenges imposed by the existence of non-homogenous tools, the necessity to reuse the code and the predicted acquisition of new equipment.

A domain for data integration, based on the laboratory measurement tools was presented. A data integration metamodel, which is general for any laboratory, is defined and used for interpreting models conforming to it and grasping the laboratory specificities. This approach facilitates the integration of new measurement tools, by simply defining new types with the model editor and, if necessary, programming the parsers for converting the acquired data.

This domain is designed for a further composition with other laboratory domains, as the one for advanced processing, opening the way for a large scale integration based on models

Acknowledgments

This study was supported by A-37/2007 CNCSIS grant and by CEEX 324/2006 MATHYS ANCS (AMCSIT) contract.

References:

[1] R. Barker, B. Wichmann, Guidance for accredited laboratories on the use of computers, *Accreditation and Quality Assurance*, Springer-Verlag, Vol. 5, No.7, 2000, pp. 287-288

[2] E. Kapsammer, T. Reiter, W. Schwinger, Model-Based Tool Integration - State of the Art and Future Perspectives, *Proceedings of the 3rd Int. Conference on Cybernetics and Information Technologies, Systems and Applications* (CITSA 2006), Orlando, USA, 20-23 July 2006

[3] J. Estublier, A.D. Ionita, G. Vega, Relationships for Domain Reuse and Composition, *Journal of Research and Practice in Information Technology*, Vol. 8, No. 4, 2006, pp. 288-301

[4] A. Occello, O. Casile, A.Dery-Pinna, M. Riveill, Making DomainSpecific Models Collaborate, *Proc. Of the 7th OOPSLA Workshop on Domain Specific Modeling*, Montréal, Canada, 21-22 Oct. 2007

[5] R. Pavlis, Information Integration. A View Toward the Future of Instrument Interfacing, *Scientific Comp. & Instrumentation*, March 2003

[6] L. Bass, P. Clements, R. Kazman, *Software Architecture in Practice*, Addison-Wesley, 2003

[7] M. Simos, Organization Domain Modeling and OO Analysis and Design: Distinctions, Integration, New Directions, *STJA'97 Conf. Proc.,* Technische Universität Ilemenau, Thüringen, 1997, pp. 126-132

[8] J. Bezivin, From Object Composition to Model Transformation with the MDA, *Proceedings of TOOLS*, USA, Santa Barbara, August 2001

[9] D. Weiss, C-T-R Lai, *Software Product-Line Engineering: A Family Based Software Development Process*, Addison Wesley Professional, 1999

[10] A. D. Ionita, J. Estublier, G. Vega, Variations in Model-Based Composition of Domains, in *Proc. of Software and Services Variab. Management Workshop – Concepts, Models and Tools*, Helsinki, Finland, April 2007, pp. 87-93

[11] K. Czarnecki, Overview of Generative Programming, *Proc. Of International Workshop on Unconventional Programming Paradigms UPP'04*, Mont Saint-Michel, France, 2004

[12] V. Ionita, A.D. Ionita, Model-Based Software for Integrated Magnetic Material Laboratory, *Proc. Of Int. Symp. on Electromagnetic Fields in Mechatronics, Electrical and Electronic Eng.*, Prague, Czech Republic, September 13-15, 2007

[13] R.H. Fogh et al., A framework for scientific data modeling and automated software development, *Bioinformatics*, Vol. 21 No. 8, 2004, pp. 1678-1684