Advanced techniques for metamodelling

BOGDAN A. BRUMAR,
Department of Computer Science, Lucian Blaga University of Sibiu, Faculty of Sciences
Str. Dr. Ion Ratiu 5-7, 550012, Sibiu, ROMANIA

EMIL M. POPA,
Department of Computer Science, Lucian Blaga University of Sibiu, Faculty of Sciences
Str. Dr. Ion Ratiu 5-7, 550012, Sibiu, ROMANIA

Abstract: - Due to rapid changing business requirements the complexity in developing applications which deliver business solutions is continually growing. To manage this complexity, environments providing flexible metamodelling capabilities instead of fixed metamodels has shown to be helpful. The main characteristic of such environments is that the formalism of modelling -the metamodel -can be freely defined and therefore be adapted to the problem under consideration. This paper gives an introduction into metamodelling concepts and presents a generic architecture for metamodelling platforms. Three best practice examples from industry projects applying metamodelling concepts in the area of business process modelling for e-business, e-learning, and knowledge management are presented. Finally, an outlook to future developments and research directions in the area of metamodelling is given.

Key-Words: - metamodelling techniques, knowledge-management, modelling approaches, metamodelling architecture, mechanism.

1 Introduction

Due to rapid changing business requirements such as faster time to market, shorter product lifecycles, increased interdependencies between business partners, and tighter integration of the underlying information systems, the complexity in developing applications which deliver business solutions is continually growing. Therefore, the elements of an enterprise are managed more and more model-based.

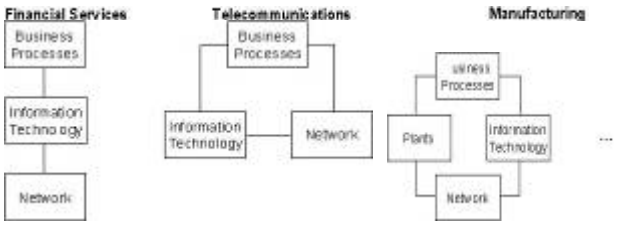

Fig. 1: Branch-specific business architectures

The state-of-the-art in the area of modelling of organisations is based on fixed metamodels. Product models are created by using product modelling environments, process models are created in business process modelling tools and organisational models are realised in personnel management tools. Web service models link these business models to information technology. They are created by using standardised languages and common ontologies. Information technology is modelled in tools supporting notions such as workflow or object-orientation. The models of the company's strategy, goals and the appropriate measurements are described and monitored by using tools supporting management concepts such as Balanced Scorecard.
Additionally, business architectures depend highly on the branches under consideration. E.g. as the network is a supporting element for doing business in financial services or manufacturing, in the telecommunication industries the network is the essential part of the business model (see figure 1). Branch specific solutions can be seen for example in the enterprise resource planning market, where all major manufacturers offer solutions for different lines of businesses. This causes additional requirements for modelling platforms, such as integration mechanisms for different views and aspects under consideration. Other major requirements to an enterprise modelling platform are flexibility, adaptability, and openness, to integrate models based on different modelling paradigms such as decision support models, descriptive models, or predictive models. These requirements have to be fulfilled by environments providing flexible metamodelling capabilities. The main characteristic of such environments is that the formalism of modelling -the metamodel -can be freely defined. Platforms based on metamodelling concepts should support the following topics:

- Engineering the business models & their web services
- Designing and realizing the corresponding information technology
- Evaluating the used corporation resources and assets

This raises research issues on how to design, manage, distribute and use flexible metamodels on a syntactic as well as on a semantic level and how to integrate, run and maintain a metamodelling platform in a corporation's environment.

2 Metamodelling Architecture

Metamodelling platforms should be realised on a component-based, distributable, and scalable architecture. Figure 2 shows a generic architecture for metamodelling platforms.

Model Builder Model Builder Metamodel Builder MechanismViewer ...

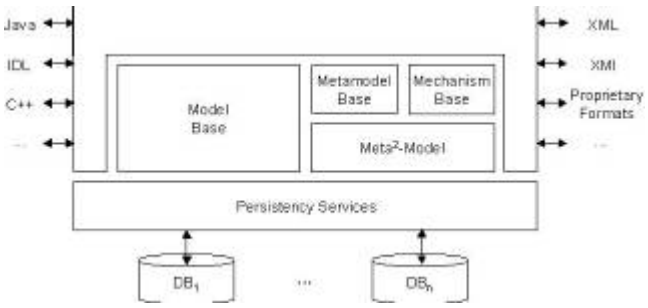

Fig. 2: Generic architecture of metamodelling platforms

The storage of all model and metamodel information is managed by persistency services. These services provide transparency of concrete storage types such as specific databases, files systems etc. Furthermore the persistency services enable the distribution of parts of models and metamodels.

The meta^2-model provides the basic concepts to create metamodels and mechanisms. Typical concepts are "classes", "relations", "attributes", "modeltypes", "scripts" etc. The meta^2-model is the central part of the architecture, as it provides the conceptual foundation and is connected with all other parts.

The metamodel base contains all information about the metamodels currently managed by the modeling platform. Changes in the metamodel base are delegated to the model base accordingly, to keep the models and their corresponding metamodels consistent.

The mechanism base contains information about functionalities to be applied to models and metamodels. These functionalities can be either stored directly in the mechanism base or outside of the metamodelling platform. If they are stored outside, the mechanism base holds only information how to find the appropriate mechanisms e.g. by using external name services.

The model base contains all models based on the metamodels. The model base communicates with the metamodel base to track metamodel changes and to forward them to the corresponding models.

Access services provide file-based and online interfaces to the different types of bases. According to access rights the appropriate information from the bases can be queried or even changed.

On top of the access services, different viewer and builder components support the usage and maintenance of the metamodelling platform such as model builder, metamodel builder, and mechanism builder.

3 Syntax, Semantics, Notation, and Mechanisms

A (graphical) modelling language is described by its syntax, semantics, and notation.

The *syntax* describes the elements and rules for creating models and is described by a grammar. For modelling languages two major approaches exist to describe their syntax: graph grammars [2] or metamodels [1]. Often, UML class diagrams are used to describe the metamodel of the syntax. For syntactical rules, which cannot be fully expressed by class diagrams, additional constraint languages are used such as OCL or AdoScript.

The *semantics* describes the meaning of a modelling language and consists of a semantic domain and the semantic mapping. The semantic domain describes the meaning by using ontologies, mathematical expressions etc. The semantic mapping connects the syntactical constructs with their meaning defined in the semantic domain ("semantic schema"). To formulate semantic definitions approaches such as denotational semantics, operational semantics, axiomatic semantics or algebraic semantics are used [1]. Sometimes, only (informal) textual descriptions are used to define the semantics, e.g. in the definition of the UML.

The *notation* describes the visualisation of a modelling language. Static approaches define the symbols for visualizing the syntactical constructs e.g. using pixel-based graphics or vector graphics,

but they do not consider the state of the modelling constructs during modelling. Dynamic approaches consider the model state by splitting the notation in a representation part and a control part. The representation part maps to the static approach. The control part defines rules to query the model state and to influence the representation depending on the model state [3].

Mechanisms provide the functionality to use and evaluate the models built by using the modelling language. Mechanisms can be classified into generic, specific, and hybrid. Generic mechanisms are implemented on the meta^2-model, so they can be used for all metamodels based on the meta^2-model. Specific mechanisms are implemented for a particular metamodel. Hybrid mechanisms are implemented on the meta^2-model, but are adapted to particular metamodels, e.g. to improve usability [4].

Considering the components of metamodelling platforms shown in figure 2, different roles in administering and using such platforms can be distinguished.

The method engineer is responsible for a consistent and properly defined modelling method. Additional to his technical skills, the method engineer often has professional skills in an application domain. Application domains can be divided into verticals such as financial services, telecommunications, public administration, and manufacturing and horizontals such as business process modelling, application development, workflow management, and knowledge management.

The language engineer defines the modelling language. He is responsible for an adequate definition of the syntax, semantics, and notation.

The process engineer is responsible for the definition of the modelling procedure. Often the process engineer is an expert in applying modelling languages and has considerable experiences in project management and project execution.

The tool engineer configures the mechanisms of a metamodelling platform for particular metamodels. If additional mechanisms are needed, he is the responsible for implementing these mechanisms.

The infrastructure engineer provides the necessary IT infrastructure to run a metamodelling platform and to integrate the platform into existing infrastructures.

The method user applies the modelling method by using the platform. He creates models by using the modelling language, following the modelling procedure and applying the available mechanisms.

4 Metamodelling Approaches

There exist various metamodelling approaches, different in richness of concepts and ranging from conceptual proposals to already implemented products. In the following, some of them will be illustrated briefly.

Atkinson proposes a modelling hierarchy aligned with the MOF hierarchy [5]. The focus is modelling in the area of distributed object systems. Atkinson stresses the dichotomy of “class” and “instance” which occurs changing the language level and proposes requirements for metamodelling approaches.

Frank proposes within his *MEMO* approach (“multi perspective enterprise modeling”) a three level modelling hierarchy. Based on this hierarchy a modelling framework with the same named is suggested [6].

The *Resource Description Framework (RDF)* provides a modelling hierarchy for semantic networks. The foundation of RDF is build by three object types (“resource”, “property” and “statement”) for representing named properties and property values.

The *CASE Data Interchange Format (CDIF)* is based on a four level model architecture [7]. CDIF is a standard designed for the exchange of CASE models between tools of different tool providers. CDIF is not be further developed but major parts of the concepts influence the design of other metamodelling approaches such as the Meta Object Facility (MOF).

The *MOF* is a infrastructure for managing meta information [8]. Conceptually, MOF can be divided into two major parts: (a) the definition and maintenance of meta information based on a four level modelling hierarchy and (b) specifications of interfaces to access the metainformation within a distributed environment.

The *General Modeling Environment (GME)* is based on a four level modeling architecture. In [9] general metamodelling requirements and a approach of model integrated computing (MIC) is proposed.

The *MétaGen* system distinguish a “user metamodel” and a “implementation metamodel”. Based on transformation rules between these metamodels, the system development should be more aligned with the requirements definition [10].

Kühn et al. propose a four level modelling hierarchy [4].

Another commercial product is the metaCASE tool *MetaEdit+* from MetaCase Consulting. MetaEdit+ is a configurable CASE tool, based on a metamodelling approach.

5 Conclusion

Metamodelling concepts and metamodelling platforms are getting more and more an *integral part of business engineering* strategies and approaches. Prominent examples are the international standards UML and MOF, which are both based on a four level metamodelling approach [8]. In addition, this trend is underpinned by metamodelling products already available such as ADONIS or MetaEdit+ [3]. The *major advantages* from out experiences using flexible metamodel approaches instead of approaches using fixed metamodels are considerable savings in time and costs in application development, increased quality of delivered solutions, and enhanced acceptance because of directly mapping the domain under consideration.

Nevertheless, metamodelling is still a very challenging field for innovative future developments and essential research activities. Some of the developments and research directions we are expecting are:

- *Integration and interoperability:* The integration of heterogeneous systems to interoperable systems is part of enterprise application integration (EAI) efforts. In addition to technical integrations, the systems have to be integrated on a semantically level. Coordinated metamodels, integration of ontologies, and enterprise model integration (EMI) give rise to further research.
- *Semantic Web:* The vision stated by Berners-Lee [11] aims at developing languages for expressing information in the WWW in a machine understandable form. Currently, most information in the Web is for human consumption. Promising efforts such as RDF are based on metamodelling concepts.
- *Model-driven Business Engineering:* Managing organisations and developing large enterprise applications causes complex interdependencies between different parts of organisations and applications. Often these parts are managed and realized by using different technologies and, if any used, different modeling environments. Chaining models for business, development and evaluation ("straight through business engineering") to measure and control business decisions based on operational data generated by business applications is of vital research interest.
- *Combination of modelling paradigms:* Modelling paradigms used in the IS development field are mostly descriptive. Other paradigms such as decision support models and predictive models are often used focusing on the business domain. We expect strong interest in combining these approaches by metamodelling to form new possibilities in enterprise management and development.

Language Engineering: The definition of "good" modelling languages and their implementation in helpful software support still need a lot of experience and knowledge. To capture these experiences, patterns could be an appropriate formalism. E.g. the current definition of semantics of modelling languages is either informal, and therefore often error prone and not directly understandable by machines, or formal, i.e. very time-consuming and expensive. In this area we are expecting improvements by interdisciplinary research.

References:

[1] Geisler, R.; Klar, M.; Pons, C.: *Dimensions and Dichotomy in Metamodeling*. Technical Report 98-5, Technical University Berlin (1998). http://tfs.cs.tu-berlin.de/~espress/doc/own/GKP98.ps

[2] Revault, N. M., Sahraoui, H.A., Blain, G., Perrot: *A Metamodeling Technique: The MétaGen System.* In: Proceeding of the Tools Europe'95 Conference (1995)

[3] BOC: ADONIS 3.6 Manual IV: *Method Definition and Administration.* BOC Information Technologies Consulting GmbH, http://www.boc-eu.com

[4] Kühn, H., Junginger, S., Bayer, F.: *How Business Models Influence the Development of E-Business Applications.* In: Standford-Smith, B., Kidd, P. (eds.): Proceedings of the eBusiness and eWork Conference 2000. IOS Press (2000)

[5] Atkinson, C.: *Metamodeling for Distributed Object Environments*. In: Proceedings of the 1st International EDOC Workshop (1997)

[6] Frank, U.: *Multi-Perspective Enterprise Modeling (MEMO) -Conceptual Framework and Modeling Languages*. In: Proceedings of the 35th Hawaii International Conference on System Sciences, Hawaii (2002)

[7] Electronic Industries Association-CDIF Technical Commitee: *CDIF CASE Data Interchange Format -Overview*. EIA-IS-106 (1994)

[8] Object Management Group: *OMG Unified Modeling Language Specification*, Version 1.4 (2001).http://www.omg.org/cgi-bin/doc/?formal/01-09-67.pdf

[9] Rozenberg, G. (ed.): *Handbook of Graph Grammars and Computing by Graph Transformation -Volume 1: Foundations*. World Scientific Pub. Co. (1997)

[10] Pan, J. Z., Horrocks, I.: *Metamodeling Architecture of Web Ontology Languages*. In: Proceedings of the Semantic Web Working Symposium (2001)

[11] Berners-Lee, T.: *Semantic Web Road Map*. W3C Design Issues. http://www.w3c.org/DesignIssues/Semantic.html