Metamodeling and development of information systems

Metamodelování a vývoj informačních systémů

M. Pícka

Czech University of Agriculture, Prague, Czech Republic

Abstract: Metamodeling is becaming an important part of information systems development. When metamodeling we are working with metamodel which define the syntax and the semantics of models. There are a lot of standards for definition of metamodel e.g. COMMA, GOPRR, MOF. Metamodeling is used for defining and creating of new methodologies, their implementation into CASE and metaCASE tools. Metamodeling is used for manipulation of data and metadata and for optimalization of information's system design with utilization of generic models. The aims of this article are: to discusse fundamental concepts of metamodeling, demonstrate some theoretically and practically important meta-metamodels and to position metamodeling principles in the development of information systems.

Key words: model, metamodel, meta-metamodel, metamodeling, methodology, metaCASE, MOF, COMMA, GOPRR, generic model

Abstrakt: Metamodelování se stává důležitým nástrojem při tvorbě informačních systémů. Při metamodelování pracujeme s metamodelem, který zachycuje syntaxi a sémantiku modelu. Pro definici metamodelu existuje mnoho více či méně rozšířených standardů, jako např. COMMA, GOPRR a MOF. Metamodelování se používá při tvorbě nových metodologií, pří vytváření CASE a metaCASE nástrojů, při manipulaci s daty a metadaty a při optimalizaci návrhu informačního systému pomocí generického modelu. Cílem tohoto článku je ukázat základní koncepty metamodelování, některé teoreticky a prakticky významné meta-metamodely a použití principů metamodelování při tvorbě informačních systémů.

Klíčová slova: model, metamodel, meta-metamodel, metamodelování, metodologie, metaCASE, MOF, COMMA, GOPRR, generický model

FOREWORD

Today, when modern software programming tools are available in the market, it may look, that information systems analysis and development are easier than in the past. This statement might be true, but only when user requirements are still the same and are well defined. But, in the area of agricultural information systems, there is a significant demand for software support of new business processes, the need for data structures being very complex, time dependent etc. This is in contrast with the "more conventional" requirements, as they are usually simpler but typically use more data. This means that the most complicated step in any agricultural information system is to perform system analysis, which must be fast and seamless.

INTRODUCTION INTO META

At present, the new "modern" word (or rather prefix) appears to be *meta-*. We are providing operations with metadata. We have a metaCASE tool driven by the metamodel and we use them for metamodeling new methodologies. The methodology for designing new method is called metamethod. Programs can be generated from metadata. Even we can program (or better metaprogram) in metaprogramming languages.

To understand the meaning of the word, there are encyclopedias and there we see that meta- is a prefix originating from Greek language and it means beyond or after (spatially). The other meaning of this word (fundamental for information systems) – one level of abstraction higher- is that metadata is data defining another data, metamodel is a model describing concepts used in another model (the model of models). On the other hand, a metasystem is not the system composed of other systems. This is a typical misinterpretation of the "meta" concept.

METAMODEL ARCHITECTURE

According to MOF specification, the classical framework for metamodeling is based on an architecture with four metalayers. These layers are conventionally described as follows:

The contibution presented at the international conference Agrarian Perspectives XII (CUA Prague, September 18–19, 2003).

– The information layer is comprised of the data that we wish to describe.
– The model layer is comprised of the metadata that describe data in the information layer. Metadata are informally aggregated as models.
– The metamodel layer is comprised of the descriptions (i.e., meta-metadata) that define the structure and semantics of metadata. Meta-metadata is informally aggregated as metamodels. – A metamodel is an "abstract language" for describing different kinds of data; that is, a language without a concrete syntax or notation.
– The meta-metamodel layer is comprised of the description of the structure and semantics of meta-metadata. In other words, it is the "abstract language" for defining different kinds of metadata.

The classical four layer meta-modeling framework is illustrated in Figure 1. This example shows metadata and the corresponding model for simple composition between an employee and a car (i.e. instances of *Employee* and *Car*) along with its metamodel for describing and hard-wired meta-metamodel. The layer of metamodel consists of meta-metadata that describe the nature of *Class*, *Composition* and *Attribute* and their relationships. The top layer consists of the hard-wired meta-metamodel, that defines the metamodeling constructs (e.g., *meta-Classes* and *meta-Attributes*).

While the example shows only one model and one metamodel, the main aim of having four meta- layers is to support multiple models and metamodels. Just as the class *Car* describes many *Car* instances at the information level, the *Class* metamodel can describe many classes at the model level. Similarly, the meta-metamodel level can describe many other metamodels. These metamodels that represent other kinds of metadata describing other kinds of information.

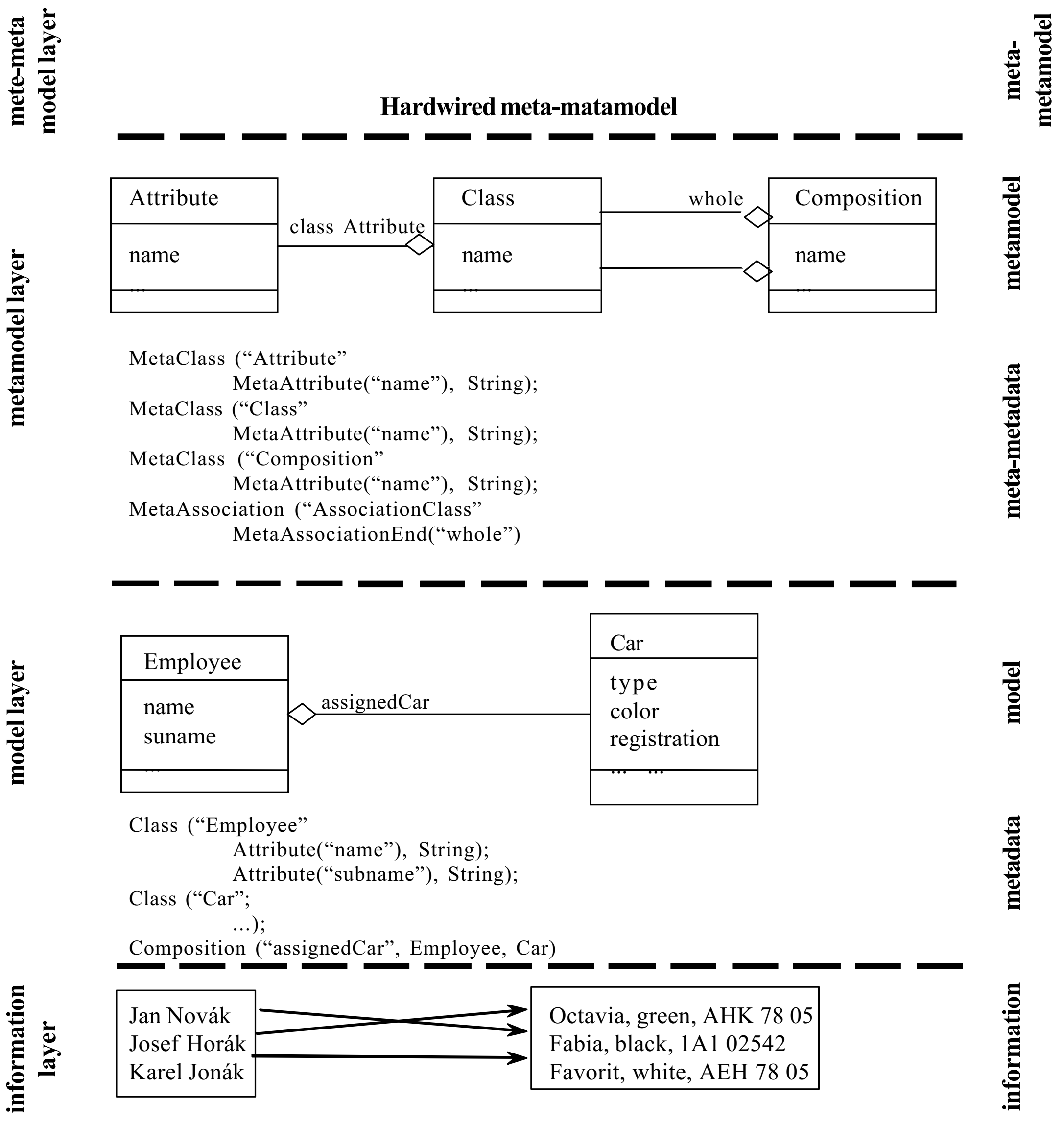

Figure 1. Four layer metadata architecture

The classical four layer metadata architecture has a number of advantages over simple modeling approaches. If the framework is designed appropriately:
– it can support any kind of model and modeling paradigm imaginable,
– it can allow different kinds of metadata to be related,
– it can allow metamodels and new kinds of metadata to be added incrementally, and
– it can support the interchange of arbitrary metadata (models) and meta-metadata (metamodels) between parties that use the same meta-metamodel.

METAMODELING INSTRUMENTS

Metamodeling methods define a framework for metamodeling, which consists of the definition of the meta-metamodel and metamodeling language that is used for defining derived metamodels. For the purpose of metamodeling (and defining metamodels), there were a lot of approaches defined by information and software engineers – COMMA, GOPRR, MOF, OPRR, CoCoA, NIAM, COOM etc. There are three metamodel frameworks characterized in the following paragraps.

COMMA

Project COMMA (Common Object Methodology Metamodel Architecture) tried to find common core of all object oriented methodologies and then represented the concepts of these methodologies in a common meta-metamodel (see Henderson-Sellers, Bulthuis 1998). On the basis of this meta-metamodel, the metamodels of common object oriented methodologies were built.

COMMA usies these basic concepts:
– Concept – expresses an entity. It has got name and attributes.
– Inheritance – expresses a relation of generalization-specialization between concepts.
– Association – expresses a relation between two concepts and this relation is undirected.
– Aggregation – is a special kind of association and expressed relation of whole part.
– Role – can be used, when concept assume characteristics of other concept. Role is temporary and concept is able to have more roles.

The main outcome of COMMA project is a simple (but powerful) object oriented metamodeling framework. The outcomes of this project are mostly theoretical, because there is no connection to the CASE tools.

GOPRR

Metamodeling framework GOPRR (Graph-Object-Property-Role-Relationship) was developed as a form of ER model modified for metamodeling (Kelly 1997). The main aim of GOPRR is to develop a quick and easy to use metamodeling framework and then to implement it into a CAME (Computer Aided Method Engineering) tool.

Primary concepts of GOPRR are:
– Graph – represents diagram. It contains a number of other non-properties: object, roles and relationships.
– Object – represents the entity which has an existence in its own right.
– Property – describes graph, object, role or relationship.
– Relationship – represents connection among two or more objects.
– Role – exists between object and relationship.

These are to be viewed in an object-oriented fashion: every element has an existence in its own right, independent of the other element. In addition to these primary concepts, GOPRR also uses some secondary concepts, many of which are better considered as simple data structures: sets, collections and bindings. These do not have existence in their own right (i.e. they exist only if contained within the primary element).

GOPRR is intended to be easy to use, implemented in and thus supported by a metaCASE environment (e.g. MetaEdit+ – see Web of MetaCase company), and to work the same for both metamodels and models. Two others guidelining ideas of GOPRR have been object-orientation and reuse.

OMG standards

Standards of OMG (Object Management Group) are based on four level metadata architecture (see Metamodel architecture). Meta-metamodel layer is defined by standard MOF (Meta Object Facility – see MOF Specification). OMG defines several metamodel standards – all of them are based on MOF:
– the UML metamodel (Unified Modeling Language) – standard of object modeling language (see OMG Unified Modeling Language Specification),
– the IDL (Interface Definition Language) – standard description object interfaces of distributed objects based on CORBA standard and their mapping into programming languages,
– the CWM metamodel (Common Warehouse Metamodel) – standard defining architecture of data warehouses etc.

Data among metamodels based on MOF can be exchanged by the XMI (XML Metadata Interchange) standard.

The MOF metadata architecture has a few important features that distinguish it from the earlier metamodeling architectures:
1. The MOF Model (the MOF core meta-metamodel) is object-oriented, with metamodeling constructs that are aligned with UML object modeling constructs.
2. The meta-levels in the MOF metadata architecture are not fixed. While there are typically 4 meta- levels, there could be more or less than this, depending on how MOF is deployed. Indeed, the MOF specification does not require there being discrete meta-levels at all at the

implementation level. MOF meta-levels are purely a convention for understanding the relationships between different kinds of data and metadata.
3. A model (in the broad sense of a collection of metadata) is not necessarily limited to one meta- level.
4. The MOF Model is self-describing. In other words, the MOF Model is formally defined using its own metamodeling constructs.

MOF metamodeling is primarily about defining information models for metadata. The MOF uses an object modeling framework that is essentially a subset of the UML core. In a nutshell, the 4 main modeling concepts are:
1. Classes, which model MOF metaobjects.
2. Associations, which model binary relationships between metaobjects.
3. DataTypes, which model other data (e.g., primitive types, external types, etc.).
4. Packages, which modularize the models.

The UML metamodel may by considered as an instance of MOF meta-metamodel. The UML metamodel and MOF meta-metamodel are based on the same principles and differ only in a few things (MOF is simpler – e .g. association in MOF is only binary etc). The result is that mapping between MOF meta-metamodel and UML core metamodel is straightforward. For example, class in UML diagrams is defined as an instance of class *Class* form UML metamodel. Analogous to this, *Class* from UML metamodel is defined as an instance of class *Class* from MOF meta-metamodel.

The main advantages of OMG standards are wide acceptance of them, coverage of many domains and cohesion among these standards. The main disadvantage is the complexity of these standards.

APPLICATION OF METAMODELING PRINCIPLES

Metamodeling takes part in definition and creating new methodologies, in the implementation methodologies at CASE tools, for structuring of repositories, system integration, generation of program code from model, generation the reports and model checks. Understanding a metamodel can be used for flexible design of information system by generic models.

Applications of metamodels for definition methodologies

Methodologies are defined by metamodels. Each methodology is defined by its own metamodel. It can be formally or informally described. Informal descriptions of metamodels are included e.g. in books written about methodologies. For example, in each of book about UML, we can read that representation of class is a rectangle with three areas. In these areas, there are the name, attributes and methods of the class. We can read that association is a relationship between classes and it can be drawn by a firm line. These examples are a very informal description of part of metamodel concerning with classes and associations. For superior understanding of metamodel or computer aid of modeling, we need a formalized definition of metamodel (see OMG Unified Modeling Language Specification).

Metamodeling is the base of new methodologies development (ME – Method Engineering). The new methodology can be defined entirely new or as an extension of an existing methodology (see Pícka 2003a). It provides appliances for finding the common characteristics of methodologies, their standardizations and possibility of their mutual interoperation.

CASE and metaCASE tools

One of the first applications of metamodel were CASE tools. Methodologies are defined by metamodel and CASE tools have to implement it. The other application of metamodels is formats for exchanging data among CASE tools – e.g. older CDIF (Common Data Interchange Format) and XML based XMI.

Later, the metaCASE tools are arising. The metaCASE (or sometimes called CAME) tools allow to define own metamodel (based on hardwired meta-metamodel) and thereby define own methodology.

There is a schema of metaCASE tool in Figure 2. The main parts are:
– Graphics core + GUI – they provide proper painting and interaction with the user.
– Hard-wired meta-metamodel – it is an interpreter of metamodeling language.
– Modeling core – through the use of modeling core, an appropriate model is built.
– Meta-modeling core – through the use of it, the metamodel is programmed in a language that is defined by the meta-metamodel. This part is the main difference compared to the classical CASE tools. It provides the ability to program one’s own methodologies.
– Model – data of model typically placed in a database.
– Metamodel – data of metamodel (i.e. meta-metadata) typically placed in a database.

MetaCASE tools are sometimes called metamodel-driven software, because by change of the metamodel, we can change their behavior.

Metamodel and manipulations with data and metadata

Understanding the metamodel of data enables the possibility of understanding the structure of data and manipulating with them. Typical examples are the repository standards for store metadata. One of the first applications of metamodel is concerning the repository standards (e.g. PCTE or ISO standard IRDS – Information Resource Dictionary Standard). If we have data in the standardized

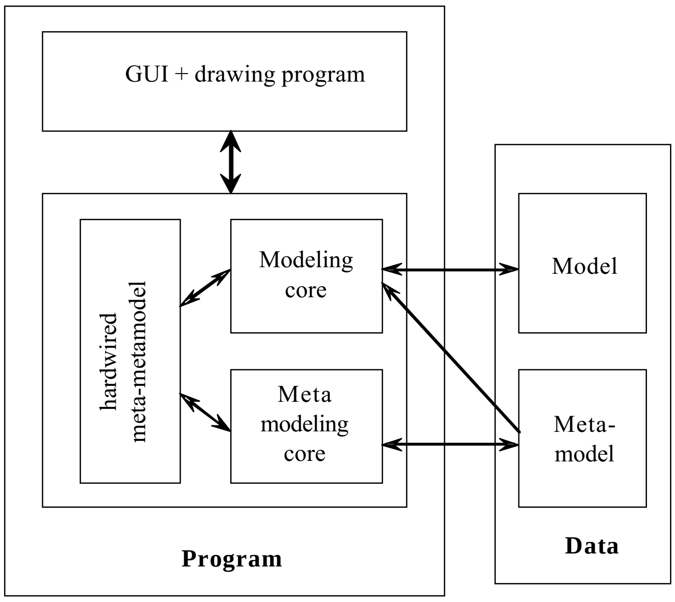

Figure 2. Schema of MetaCASE tool

repository (in the database), it is easy to manipulate with them. Typical examples of processing that data are consistency checking, metrics measure, generation of reports and programs.

Metamodeling admits characterization of different data structures of more systems in the same ways that provide an ability to compose them to the superior system. This system is able to work with data of its subsystems and to aid their integration. By contrast, the metamodel provide a good abstraction of the system and help us to divide it to the subsystems and thus help us to produce complex projects.

The generic model

For project optimization, the generic model can be used (see Polák, Merunka, Carda 2003)). Generic model is a model that combined the features of model and metamodel (see Figure 3).

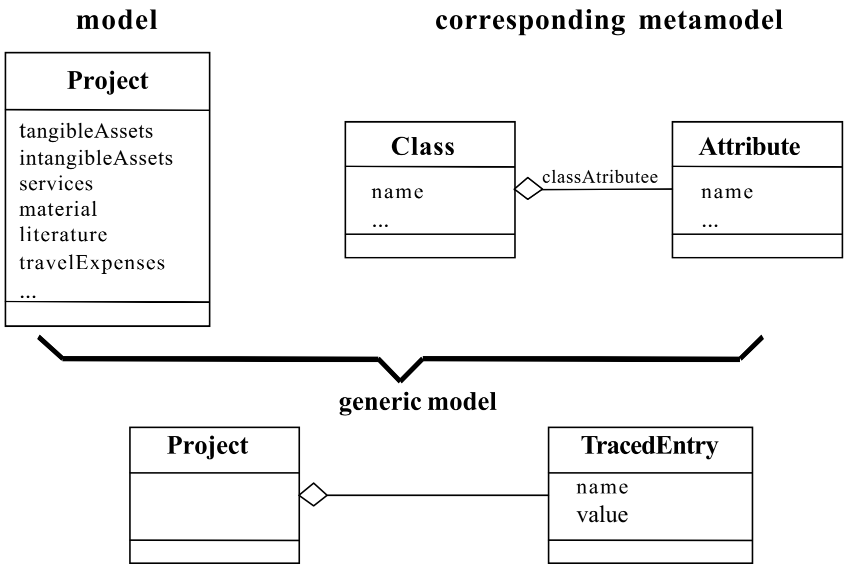

Figure 3. The generic model

There is a class *Project* in the figure that is a part of "standard" design of information system. Instances of this class have got a lot of attributes concerning the project finances (tangibleAssets, intangibleAssets, material, literature, travelExpenses etc). This design is functional but there will be problems when these attributes do not cover all of our needs (e.g. we need to cover information about domestic and foreign travel expenses separately). The other disadvantage of this design is that many of these attributes cannot be used (i.e. they have a zero value) and we need additional attributes for covering our new needs. A graceful solution is creation of class *TracedEntry* and its composition with class *Project*. This model has got the features of metamodel. Structure of the metamodel exemplifies as a pattern of modeling.

Importance of generic models is in creation of easily modifiable software that can fulfill the needs that are not known during the analysis.

CONCLUSION

Metamodeling is used for still more problem solutions at present . The most important cases for metamodeling are developing new methodologies (Method Engineering) and their computer support, providing interoperability of methodologies, system and data integration, model checks, generation of software from the model. For these cases, there exist many metamodeling frameworks . OMG metamodeling standards (such as MOF or UML metamodel) seem to be very perspective.

REFERENCES

Community site for metamodeling and semantic modeling: http://www.metamodel.com

Henderson-Sellers B., Bulthuis A. (1998): Object-Oriented Metamethods. Springer-Verlag New York; ISBN 0-387-98257-4.

Kelly S. (1997): GOPRR Metamodel. Apendix of doctor thesis Towards a Comprehensive MetaCASE and CAME Environment: Conceptual, Architectural, Functional and Usability Advances in Metaedit+, Ph.D. Thesis. Jyväskylä University.

Meta Object Facility (MOF) Specification, version 1.4. Available at http://www.omg.org.

OMG Unified Modeling Language Specification, version 1.5. Available at http://www.omg.org.

Pícka M. (2003a): Metamodel BORMu jako rozšíření metamodelu UML. Objekty 2003. Sborník konference Objekty 2003, Ostrava; ISBN 80-248-0274-0.

Pícka M. (2003b): Metamodelování v praxi. Sborník prací z mezinárodní vědecké konference Agrární perspektivy XII, díl 2., Praha; ISBN 80-213-1056-1.

Polák J., Merunka V., Carda A. (2003): Umění systémového návrhu. Grada, Praha; ISBN 80-847-0424-2.

Web site of MetaCase company: http://www.metacase.com.

Arrived on 8th December 2003

Contact address:

Ing. Marek Pícka, Česká zemědělská univerzita v Praze, Kamýcká 129, 165 21 Praha 6-Suchdol, Česká republika
e-mail: picka@pef.czu.cz