# Metacase Tools for Multidimensional Development of Information Systems

Iveta Kremenova, Roman Sulovec, and Milan Petrik

University of Zilina

**Preface**

In the area of system Development Company's world-wide face a continuous need to improve their software development practices. Two challenges are encountered especially often: support for a company's evolving business processes and the ability to build better quality software faster. It is clear that information systems are part of business processes. The more the software supports a company's business processes, the more valuable it is. Accordingly, changes in business processes and strategies must be reflected in the supporting software and systems. This is becoming one of the key issues in integrating IT infrastructure development with business development. At present on our department we do Meta CASE tools application in conditions of transforming Slovak Post (concrete metaEDIT+) for analysis and design new systems purposes. Software and non-software systems related for example to new organizational structure creating and development of information systems.

Reengineering is one of the tendencies of management work in 90s. When we say reengineering, we usually mean cardinal revaluation and radical change of business processes for purpose of large improvement of present business results. The key factor in reengineering process is the ability to creative find and application new ways that are provided by information technologies. By reengineering we can understand also more generally, as an ability back to square one start an qualitative process innovation in harmony with needs and possibilities of the information society.

## 1 The kinds of modeling in system design

In system designs within the software engineering throughout development of methodologies and methods supported by CASE systems we can see these ways of modeling:

**Data Modeling** - system consist of the quantity of operated entries (data). The most important thing is selection of main objects (entities) in modeling system. Each entity is characterized by its properties and relations with others data model entities. The best-known technique ER diagrams are used in fixed systems, relatively little changed.

**Process modeling** - the processes (procedures) in modeling system are accentuated. You have to select system processes and I/O information flows, information's, that those processes swap. Besides processes and flows there are designed various data stores (files, dB servers) and external objects. Those are used for systems that fluctuate all the time.

**Object-oriented modeling** - the base is a design of objects by the help of which it is possible to describe all past views on the system (data, process, state). The object is given

by its properties (attributes) and methods (processes), which is possible to realize over this object. Further there is defined a group of occasions, on which that object respond, responses on those occasions as well as relations between particular objects. By the help of so designed objects it is possible to create different types of diagrams describing modeling system (data, process and state diagrams).

**Unifying in modeling** - unification of existing modeling tools. Unified Modeling Language (UML) establishes certain intensity of the standardization in the modeling area. The authors of already existing modeling tools created it and it combines advantages of these tools. UML is a modeling language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non - software systems.

Modeling technique - is the best way of clear description of developing system in actual and future state. The tools, that create various types of models are usually a component part of CASE systems.

## 2 CASE and MetaCASE tools

The term CASE (Computer Aided Software/Systems Engineering) means abreviately a development of an information system or SW engineering supported by the computer. To onset CASE it require a reorganization so the work with the accent on the discipline, systematic approach and the formalization of all the process. Et hoc genus omen is good many times more difficult than to handle some CASE product.

Traditional CASE tools are based on a two-level architecture: system designs are stored into a repository, in which all information's about the designed system and the results of all project steps are stored. Basically it is a database automatically maintained in the consistent state. Repository schema is programmed and compiled into the CASE tool. This "hard-coded" part defines what kind of models can be made and how they can be analyzed. Most importantly, only the tool vendor can modify the method, because it is fixed in the code.

All rules, that can be within the used methodology algorithmic, CASE all the time applied on the stored resultants. At the same time it disallow activities that would make logical or formal mistakes. From CASE systems in the next time we expect:

- Unique standard (normalization) - UML modeling,
- more integration of tools, movement to the object orientation.

### 2.1 MetaCASE tools

MetaCASE tools are based on a three-level architecture. The lowest - model level - is similar to that of CASE tools. It includes system designs as models. The middle level contains a model of the method, i.e. a metamodel. A metamodel includes the concepts, rules, and diagramming notations of a given method. For example, a metamodel of UML specifies concepts like a "class" and an "inheritance", how they are related, and how they are represented. Thus instead of being imbedded in code in the tool, as in a fixed

CASE tool, the method is stored as data in the repository. The use of metamodels has recently become more popular: many method books include metamodels of their method, and several important innovations, such as XML, are metaled-based.

Unlike a CASE tool, a metaCASE tool allows to modify the metamodel. Hence, meta-CASE is based on the flexibility of the method specifications. This is achieved by having a third, highest level that includes the Meta modeling language for specifying methods.

All the three levels are tightly related: a model is based on a metamodel, which in turn is based on a metamodeling language. This dependency structure is similar to that between objects, classes, and metaclasses in some object-oriented programming languages.

The Figure 1 illustrates differences between CASE and metaCASE tools.

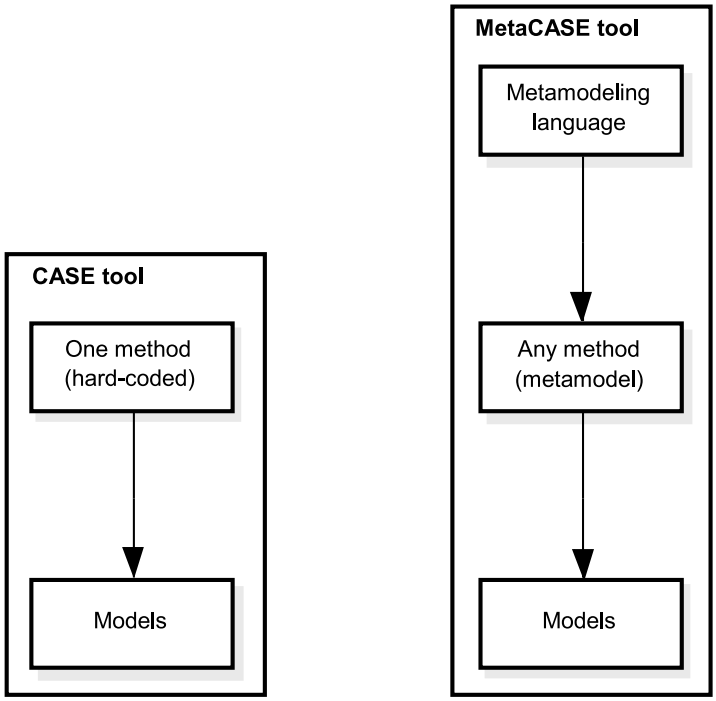


Figure 1. CASE tools versus metaCASE tools

### 2.2 Necessary functionality for MetaCase Tools

In addition to the three-level architecture, a metaCASE tool must provide functions that support both method development and implementation of the CASE functionality. These necessary functions are described in the following:

**Definition of metamodels.** The first obvious requirement is that a metaCASE tool can specify the concepts, rules, and symbols of individual modeling techniques as well as their interconnection rules. Moreover, the method definitions should be as complete as possible, and should be relatively easy and fast to make in comparison with programming a CASE tool from scratch. Let's quantify 'easy and fast' method definition by saying that to learn the tool and create a metamodel for Use Case diagrams should not take a newcomer longer than a few hours.

**Creation of modeling tools.** Based on the method definitions, a metaCASE tool must provide all necessary modeling tools to support designing with that method. These include different kinds of editors, toolbars, dialogs, online help, etc. The creation of these tools should be automatic, based on the metamodels.

**Creation of repository.** Both models and metamodels must be stored into a multi-user repository and be accessible for developers. Other repository administration tools should be available as well.

**Definition for model reporting.** In addition to the model editing and storing, a metaCASE tool should allow the definition of various model analyses, checking, code generation and model documentation reports. It must be noted, that although a good metamodel includes all the rules of the method, checking them can not be fully automatic: some rules can be checked only after models are considered complete.

**Metamodel management.** MetaCASE tool should provide functionality for metamodel management similar to the CASE tools functions for the model management. This includes browsers, documentation tools, libraries for metamodels, and setting of access rights for method specifications or for their parts.

**Management of method updates.** MetaCASE tool should provide transformation rules between method versions (e.g. UML 1.2 to UML 1.3, or to your own extended UML) and also update designs (semi-) automatically to correspond to the new method version. This function usually necessitates an integrated metaCASE tool and CASE tool.

Interchange format for method definitions (and models). The resulting CASE tool should provide importing and exporting of both models and metamodels. Importing should be incremental: previously imported data from the same exporter should be updated automatically, rather than creating duplicates.

## 3 Object modelling methodize and methods

Object modeling - on whichever real system we can understand as an assembly of objects, among which can some relations exist. The goal of OM: in the phase of the analysis to find relevant objects, describe them by some way, and mostly integrate them to the group of all objects, find regular correlative relations. We look for borders of system and watch system relations with the environs.

Then in next phases (design and implementation) we just elaborate particular objects, defines more relations among them, and we try to consolidate created object models to get

only the most important set of basic objects constituting a heart of the system. These objects constituting a ”business logic” than can be connected e.g. with SW objects providing for the communication with the user or objects stored in the data base.

If we scan today's way of SW systems making, we discover, that just the logic object layer is often absent. The reason is of course the missing problem analysis and the analysis problem area as an entity.

### 3.1 History of object oriented modeling-OOM

About OM was more intensive spoken in late 80s, i.e. in the time when object approach was enforced. We can talk about three generations of the object-oriented modeling.

**The first** object modeling generation- late 80s - Booch, Shlaer/Mellor.

**Second-generation** - Object Modeling Technique (OMT) - Coud/Yourdon, Martin/Odell. This is known e.g. by using the Use Case modeling as the heart of Objectory/OOSE methods. In this epoch OMG - Object Management Group - a corporation of hundreds of firms in the object oriented technology market started up.

**The third** generation is characteristic with the object oriented analysis and design. In this time also the UML - Unified Modeling Language (Grady Booch, Ivar Jacobson, James Raumbaugh) started up.

**Usage of OOM:**

System analysis,

Design and implementation

Business modeling methods

Business processing methods

Dynamic and static modeling

Methodic OOM make use of the set of tools, it include e.g. 8 different diagram types.

In the next part this article relate to some current usage of OOM.

**Business Object Notation**

A young and easy methods the first time presented by K. Walden, J.M. Nelson in 1994. It is markedly influenced by the object language Eiffel, that is because of its simplicity, consistent design and strong means of expressions used in the first place when creating systems with the safety and reliability as the key attributes.

Because of its simplicity BON don‘ t contain some means, that are required as a necessary component whatever methods. BON contains relatively slight tools for the dynamic of the system BON with Coad/Yourdon are suitable for object modeling teaching.

**Business Object Relation Modeling**

BORM - is more oriented to the BPR, BM then to the SW making. Partly it issue from the Coad/Yourdon methodizes, from which it takes up the layer model and OBA (Open Business Analysis) method. In the recent time some tools of this methodize was implemented to the MetaEdit+ system.

### 3.2 MetaEDIT+

On the department of the communications we nowadays deal the possibilities of the implementation of metaCASE on Slovak Post conditions. Hence this article writes about the product MetaEdit+.

MetaEdit+ support the largest set of the methods on the market. There are four main groups of methods:

Business modeling (e.g. Value Chains and Value Systems, Business Systems Planning - BSP),

Planning and management of system architectures,

Structured methods (Structured Analysis and Design),

Object - oriented methods (UML, OMT, OOD, OOSD, OOSA).

MetaEdit+ contain a set of predefined methods, but it is possible also to create your own methods, expand existing methods and use them instead of predefined methods or together with them. ME+ support the jump from structured methods to object oriented methods, enable connect and reuse data within different methods, maintaining the information flow among them.

**There are 4 main categories of tools in MetaEdit+:**

**Editors,** which are tools for viewing, editing and managing method specific information about system development projects. The editors are the Diagram Editor, Matrix Editor, and Table Editor.

**Environment management tools** include a Launcher for managing the tools of the environment as well as repository browsers for browsing information stored in the Object Repository.

**Reporting and code generation tools** for accessing information in the repository, and transforming that either to various reports or checking, or to program code. MetaEdit+ includes some predefined reports as well as a tool called Report Browser which allows users to make their own reports.

**Method development tools**, which forms the metaCASE part of the environment allowing users to modify the environment, its methods, graphical symbols, dialogs, and reports.

**Appendix**

Object analysis and design methodizes and methods are in the latest time relatively accomplished means of IS development. In last years step-by-step Business Processing activities that relate to IS development are getting to the foreground. Also in this area can object technologies be useful. If we talk about OOAD methods, then there is visible constantly more reception to implementations of tools suitable for this specific field. In the ideal case it is possible to use methods and tools of only one methods as a main mean of the business modeling as well as an efficient way of IS designing, that adequate support new working practices.

Object technologies have always wider application. And for all that next generations methods and methodizes should shoot this fact.

The sketchy methodizes can be used not only in project management but concretely in transformation works, e.g. in reengineering of the organizational structure of the organization, in new management forms implementation, in the removing of entity multiplicity within the organizational structure of the organization, and in building of the full area IS.

Without building of a modern economic net and a management supported by IT, any project management model either process decomposition in organization and management don‘ t give us results we expect. Only accurate and timely information's, that we have in each situation available, enable us make good decisions. And it is possible to have those information's as early as solving projects, and in analysis and design phases.

## References

[1] Kremenova I."Decomposition Processes in Post with units Information Technology", Habilitation work, EDIS ZU, Zilina

[2] Kremenova I."Automatic information systems. Practical exercises", ES ZU, ISBN80-7100-540-1, 1998

[3] Smil P. "Development methodical and methods objective analyses a design", Software newspapers, January 2000, number 1, year XI

[4] Kremenova I. "Automatic information systems", Part I. (Post, Telecom, Post bank, PNS), ES VSDS 1996, ISBN80-7100-364-6

[5] www.metacaxe.com/cases.html

[6] www.metacaxe.com

[7] www.metacase.com/methods/