

BORM – Business Object Relation Modeling

Vojtěch Merunka
Dept. of Information Engineering
Czech University
of Agriculture in Prague
merunka@pef.czu.cz

Jiří Polák
Management
and Consulting Group
Deloitte&Touche Prague
jpolak@deloitteCE.com

Louis Rivas García
Dept. of Information Engineering
Czech University
of Agriculture in Prague
garcia@pef.czu.cz

Abstract

In this paper, we present BORM - Business Object Relation Modeling, some of its tools and methods. BORM is the member of so-called domain specific methodologies and provides a methodology developed to capture user requirements and present them in a way, that according from clients suggests is very effective. BORM is well applicable in areas, where software application requirements are not completely specified at the begin of the development process, but can be interviewed, modeled, refined and validated as the collection of processes with mutually collaborating objects.

Introduction - Object Oriented Systems Development

"Μεθοδολόγια" is originally a Greek term that means "study of methods". However, the term methodology is pragmatically well established within the field of information systems to mean the same as method. In this paper, it is assumed that "methodology" has the same meaning as the word "method".

Developing software systems is a complex activity charged with many difficulties for software engineers as they endeavor to ensure, that there the right system is built. It means that the software developers must fully understand the user domain and moreover to convey this understanding of the domain to the user. The problem is to find a common language for the developers to express their understanding of the problem space that is both sufficiently rich for the developers to fully articulate their ideas while also being comprehensible to users from all business domain areas of discourse. [26]

Use-Case [3] has become a well accepted part of object oriented analysis and in many cases has proved a useful mechanism for communication between developers and domain experts. Eeeles and Sims [20] define a business process as a collection consisting of a number of elements; activities, transitions, states and decisions. They too state that the UML activity-diagrams can be used as the modeling tool in capturing business processes and user requirements.

Are there Some Problems with the Actual Object Oriented Design Methodologies?

The first and we think the major problem with object oriented methodologies arises in the initial stages of system development cycle. [8,27] The initial stage of any object oriented design methodologies should be concerned with the construction of an initial object model, often called an

essential object or conceptual model built out of an set of objects known as essential objects. This model should be constructed with the active participation of the stakeholders, in order to ensure that the correct system is being developed. Consequently, any tools or diagrams used at this stage should be meaningful to the stakeholders, many of which are not “computer engineering literate”. Hence, any diagrams used should be easily comprehensible to all, regardless of their systems experience.

The representation techniques used in most existing methodologies, often involves constructing a diagram using a diagrammatic language which provides quite complex concepts. For example the models used in OMT [13] or UML [14,15] can use quantifiers, link classes, external relations, aggregations, etc. Many of these concepts of essential requirements for any software implementation are too “software code oriented” to be useful in capturing primary system information. Moreover, such diagrams are often conceptually rich and difficult for the customer and other “non computer” people to fully understand. There is of course no compulsion to use these features but in our experience, software designers will often use all the facilities provided without any consideration as to their appropriateness. However, their existence shows that these diagrammatic techniques were developed with the later stages of software development in mind and their use in the early stages of software development was often added on as an after thought.

This approach is in marked contrast to the “old good” Structured Development Methodologies. If we compare standard Entity-Relation Diagram (ERD) [16,17] with object-class diagram used in OMT or UML, we find that ERD only uses three basic, yet powerful concepts. Object-class diagram, on the other hand, generally uses about twenty different concepts, spread over a number of different levels of abstraction.

Recently, it is true there have appeared some add-ons to a number of CASE tools to facilitate the stages of initial analysis. This is a move in the right direction, but there is still considerable progress to be made. Adding Use-Case Analysis [3] to OOA and OOD methods provides these extensions.

Use Case modeling, which in the foundation of most object-oriented development methods is often insufficient by itself to fully support the depths required for initial system modeling. Fowler [18] highlights some deficiencies in the use-case approach, suggesting that use-case diagrams if they are to convey all the necessary information need supplementation by sequence and activity diagrams. These modifications to use case analysis would result in a number of different diagrams that are used initially to define the interaction between any proposed system and any users. BORM, on the other hand, uses a single diagram that embodies the same information and, therefore is clear for the user to form a complete understanding of the interaction of the various system components.

In many development methodologies, the diagrams used do not distinguish between those concepts from the users' area of discourse (which we refer to as basic concepts) and those added to implement an object-oriented software design. We believe any initial diagram should support only basic concepts; any derived concepts should be left for later in the modeling process.

Additionally, diagrams should not overload users with complicated graphical symbols that relate to concepts defined at various levels of abstraction.

A further desirable aspect of initial system diagram is that they should not only be applicable for software development, but should also support managers with the development of decision-making models, business process reengineering and analysis of management changes, irrespective of whether the outcome requires any software implementation.

The second problem that we find with most development methodologies is that they require a number of different diagrams to fully describe the system. Each diagram is used to model one aspect of the system.

The final problem we find with most Object Oriented development methodologies is that the graphical notations are used throughout the entire development lifecycle. BORM does not use the same object concepts in all the phases of the system development process; rather there is a logical progression of concepts, usually from more abstract to more concrete representations. Moreover, it is expected that individual attributes and links between objects will change during the process of system development. In the BORM approach, such transformations of individual concepts are the matter of specific techniques used to move between the different phases of the system development.

The BORM Approach

BORM – Business Object Relation Modeling was originally developed in 1993 and was intended to provide seamless support for the building of object oriented software systems based on pure object-oriented languages and environments such as Smalltalk and object databases. Subsequently, it has been realized that this method has significant potential in business process modeling and other related business and user requirement issues. The initial work on BORM was carried out under the support of the Czech Academic link program of the British Council, as part of the VAPPIENS research project; further development has been carried out with the support of Deloitte & Touche Czech Republic. Recent developments of this method have been extensively tested within business-oriented projects. [2]

Business Object Relation Modeling (BORM) is a new approach to both process modeling and the subsequent development of information systems. It provides an approach that facilitates the description of how real business systems evolve, change, and behave. It has been used successfully, to develop a number of information systems in various areas of business activity. (18 BORM applications include Health Care, Electricity and Gas Distribution, Radio and TV broadcasting and Regional Government.)

The BORM approach is based on each object having three independent attributes called dimensions. These are data, behavior, and history (a composition of states and transitions. i.e. the object lifecycle).

BORM is fundamentally an object oriented development methodology [1] but differs from other such methodologies. In BORM, the extent of knowledge required to understand an object and use it effectively in the design process evolves throughout the development process in a clear, precise and consistent manner. Initially, objects are defined as **business objects**, where only knowledge of their activities, relationships, and intercommunications is required. [19] Business objects during the design process are changed, via a set of clearly defined and consistent techniques, into **conceptual objects**.

During the implementation phase conceptual object evolve in a similar structured and controlled manner into **software objects**. Thus at each stage of the development process, BORM requires that the degree of knowledge about an object is only what is required to enable the development process to proceed.

Thus, business consultants can participate in the development process without the need to understand software concepts that are the expertise of the software developer and vice versa. Participants in the development process are required to contribute with knowledge solely from their area of expertise.

BORM has strong theoretical background. The method uses theory of finite-state machines and forms the modeled process as a composition of more Mealy-type automata. [1,21] The method can be used with CASE Tool MetaEdit®, in which has been implemented via its meta-model description. [22] The implementation of BORM in other CASE tools is also in progress.

The BORM method provides a consistent set of concepts, with concise graphical representations, which are used at each stage in developing the process model. Moreover, these concepts facilitate a coherent model regardless of any subsequent use of the process model. Thus, the developer is easily able to use their model in a wide range of follow-on activities. (For example, Advanced BPR, Data Mining for Information System or in Information System Design)

BORM does not use the same object concepts in all the phases of the system development process; rather there is a logical progression of concepts, usually from more abstract to more concrete representations. Moreover, it is expected that individual attributes and links between objects will change during the process of system development. Such transformations of individual concepts are the matter of specific techniques used to move between the different phases of the system development. (Look at figure 1)

In all phases of BORM development process, it is possible for each concept to have some of the following:

1. *A Set of superior concepts* from which it could be derived by an appropriate technique
2. *A Set of inferior concepts* which could be derived from it by an appropriate technique
3. *Validity Range* - The phases (of the development process) where it is appropriate. There are also phases where superior and inferior concepts should be used instead.
4. *A Set of techniques and rules*, which guide the step-by-step transformation and the concept revisions between the system development phases, which are:
 - a) *Object Behavior Analysis*; a technique for transforming the initial informal problem description into the first object-oriented representation,
 - b) *Behavioral Constraints*; a set of determining rules which describes the set of possible transformation of the initial model into more detailed form with precisely specified object hierarchies like inheritance, dependency, aggregation etc.,
 - c) *Applying Patterns Technique* which helps to synthesize the analysis object model by including of object patterns,
 - d) *Set of Structural Transformations* (Class Refactoring, Hierarchies Conversions and Substitutions) which are aimed at the final transformation of the detailed design model into

a form acceptable in the implementation environment (programming language, database, user interface, operating system, solution of legacy problem, ...).

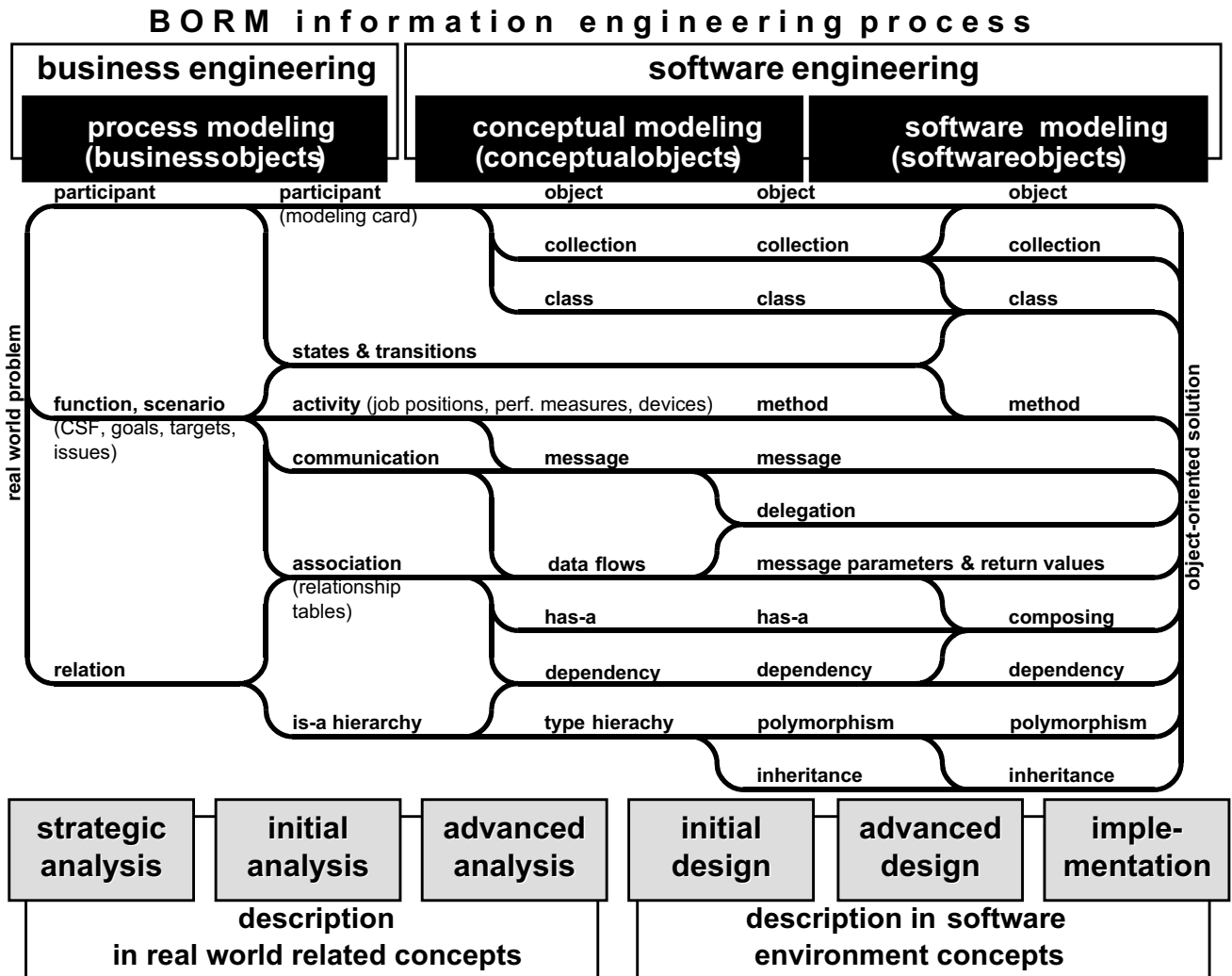


figure 1

OBA in BORM

The BORM development methodology starts from an informal problem specification and provides both methods and techniques, to enable this informal specification to be transformed into an initial set of interacting objects. The main technique used here is modified form of Object Behavior Analysis (OBA). [1,24]. BORM modified Object Behavioral Analysis (OBA) is a step-by-step iterative approach to analysis. It is a text-based method and uses a large set of form sheets, modeling cards and tables for storing and manipulating the information being processed. The result of any OBA procedure is a description of a model for the analyzed problem, expressed in natural language terms. OBA serves as an introductory knowledge acquirement technique of the BORM and the outputs of OBA are essential for the subsequent creation of the initial object relation diagrams (ORD).

After the completion of OBA, in model exists a large amount of structured information sufficient for the construction of the initial conceptual models. In addition, this structured information can be tested, refined, presented, and discussed with experts from the user domain. Hence, the OBA approach contrasts with the object diagramming out of "thin air" approach of many development methodologies. The OBA tools help to cross the conceptual gap (representational mismatch) between a description of a real world problem and the syntax and semantics of object-oriented modeling techniques.

OBA in BORM consists of five stages. Each of these has its appropriate set of tools such as tables, forms or modeling cards. Each OBA step may consist of several substeps. The five stages of OBA in BORM are presented in the following table:

| Step # | Action | Result |
|--------|---|---|
| 1 | Understand the application, perform interviews, identify the system processes | Initial behaviors of the system as the list of recognized system processes and scenarios |
| 2 | Derive initial objects in the system using behavioral perspective | Objects and object behaviors as the collection of modeling cards |
| 3 | Start classifying objects | Secondary recognized objects and all objects classified with behaviors and visible properties |
| 4 | Identify relationships among objects | Object associations and communications miscellaneous expressed via tables |
| 5 | Model and evaluate object processes | Object lifecycles and interactions |

The OBA approach is compatible with an iterative (spiral) approach used in object-oriented development. [25]. It also enables the easy identification of object mechanisms such as inter-object communications, required object methods and state changes of the object over time. This is very important because of the majority of problem descriptions provided for information systems development can be expressed in behavioral definitions and terminology. This is the case of business process specifications, database transaction requirements, and user-interface functionality expectations, etc.

Process Modeling in BORM.

One of the major issues of all analysis techniques in information engineering is to capture the intelligible description of processes in the modeled problem. This knowledge must be fulfilled quickly and effectively and the result must be comprehensible to domain experts for possible improvements during interviews, reuse in other parts problem etc. [5,12] This is the major problem of classical object-oriented methods, because they appear for large number of domain experts only as tools for visualizing software code.

In BORM, the Object Relationship Diagram¹ (ORD) shows the progress of the objects participating in some processes through its relevant activities and states. This is very simple diagram, which combines state-transition, interaction and sequence modeling of business objects. In figure 2 is a simple and funny example of ORD describing the business process in a restaurant. (Large rectangles mean objects participating in the process, small rectangles are states of objects, ovals are activities of particular objects. Activities can perform transitions between states of objects. Arrows among activities are communications/messages with optional data flows)

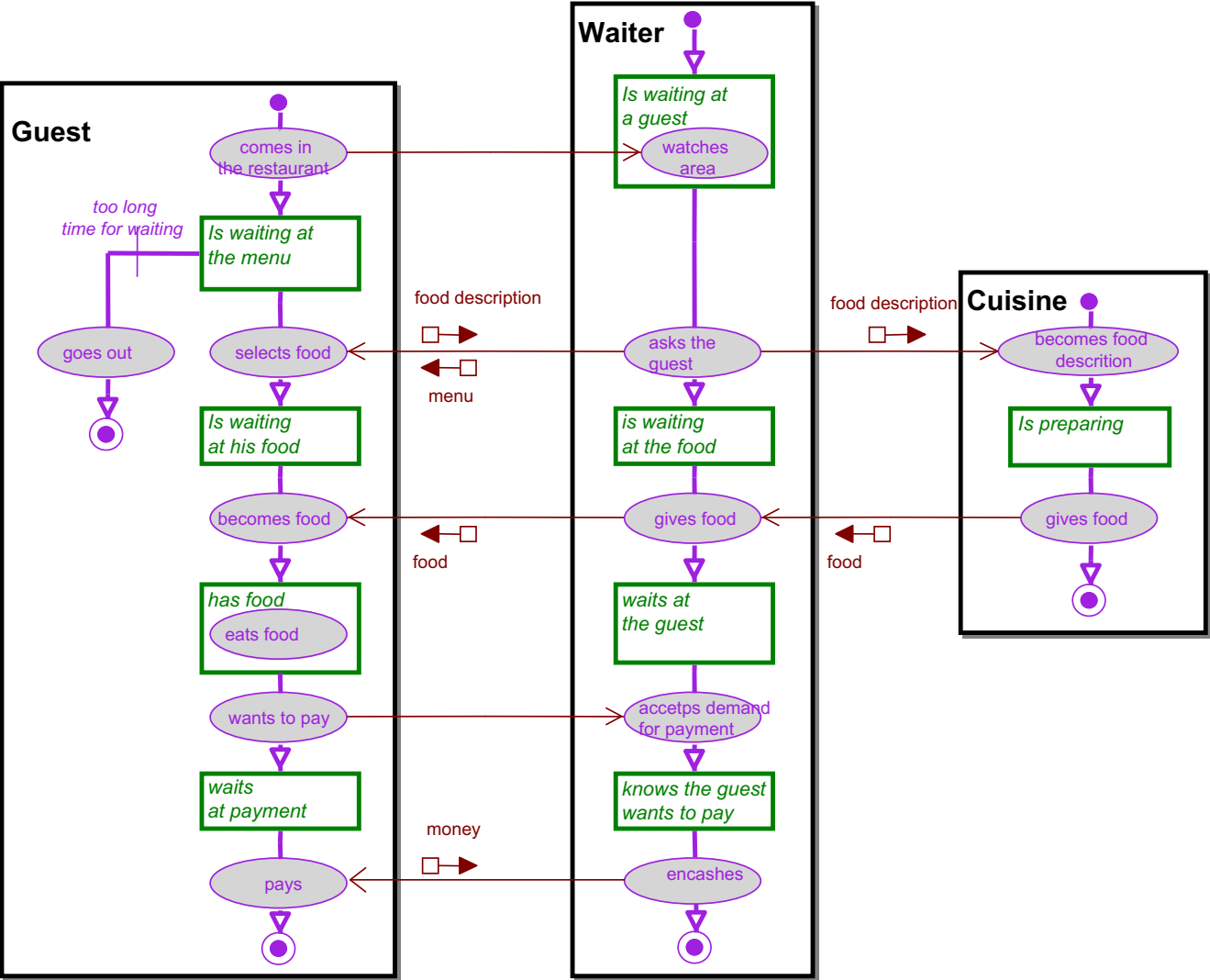


figure 2

¹ UML-like diagrams are used for subsequent conceptual and software stage of BORM.

The Advantages of BORM Approach

BORM stresses the process modeling, which has proved to be beneficial in software development - “process approach makes faster and better analysis of problem being solved” – this say our software developers having experience with OMT and UML method on similar projects, which were created in BORM.

In our experience, stakeholders from the problem domain are able to understand the BORM approach very quickly - normally a one-hour introduction at the start of analysis is enough. A consulting team of Deloitte Touche Prague and our university colleagues has worked for the past 5 years using the BORM system as well as ARIS and Rational's Objectory/Unified method. They have found BORM to be in average 3-4 times faster in carrying out the analysis phase.

The methodology is easily acceptable to domain experts, analysis consultants, and developers. (Because BORM is based on a step-by-step transformation of the model and in each phase only a limited and consistent subset of BORM concepts are used.)

BORM has been used enthusiastically by Smalltalk and Java programmers and by non-relational object database programmers (Gemstone, ArtBase). One feature of BORM software modeling, for example, that they find attractive is the way it exploits collection concept not just classes and that these collections are seamlessly integrated into the development environment. (Compare with multi-objects in UML). BORM works also with more object hierarchies (polymorphism, is-a, dependency, ...) than other methods which usually provide only object inheritance. These features provide a much richer tool for pure object-oriented programming.

Conclusion

Business systems developed for real companies often have a higher level of complexity, which makes development much more difficult. Consequently, it is essential (from the software developer's viewpoint) to improve the initial phases of software development. This is why visual modeling tools must not serve only for visualization of application code, but must provide useful tools for capturing, analyzing and validating knowledge about user requirements as well.

Until recently, it was correctly assumed that conceptual modeling tools and techniques were used through all stages of project development, from the initial phase to the eventual implementation. However, we feel that currently the position of conceptual modeling is migrating towards the implementation phase, because of the evolution of software development tools. The analysis is now being performed using newly developed techniques and “business” objects modeling tools.

In a domain-specific visual modeling languages, the models are made up of elements that are related to entities of the domain world, not the code world. Such modeling language is able to express the domain abstractions and semantics. Domain-specific modeling also provides an advantage for expert developers. Rather than have them help others in fire-fighting problems in basic development tasks, or move them to a new area and lose their expertise, they can be put to work on a problem they will find interesting and rewarding, and which will best leverage their expertise. [23]

BORM approach has been used successfully in a number of projects, both large and small, in various areas of user requirement information engineering. All of these successful projects required, in their initial stages, the construction of a business process model. Thus, BORM has a proven track record for object-oriented modeling well understood by both users and developers.

Authors would like to acknowledge the assistance of Dr. Roger P. Knott from Loughborough University, the co-author of BORM and people at Deloitte&Touche Prague BORM consulting team.

References

- [1] Knott, Roger P.; Merunka, Vojtěch; Polák, Jiří: *Process Modeling for Object Oriented Analysis using BORM Object Behavioral Analysis*, in Proceedings of Fourth International Conference on Requirements Engineering, Chicago 2000.
- [2] Merunka, Vojtěch; Polák, Jiří: *Experience of Substantial and Detailed Process Mapping as First Phase of Reengineering Requirement Definition in a Gas Distribution Company*, in Proceedings of Fifth International Conference on Requirements Engineering, Toronto 2001.
- [3] Jacobson Ivar: *Object-Oriented Software Engineering - A Use Case Driven Approach*, 1992, Addison Wesley ISBN 0-201-54435-0
- [4] Fowler M, *UML Distilled: Applying the Standard Object Modeling Language*, Addison Wesley, Reading Mass, 1997
- [5] Taylor, D., A. *Business Engineering with Object Technology*, John Wiley 1995.
- [6] Darnton, G., Darnton, M. *Business Process Analysis*, International Thomson Publishing 1997
- [7] P. Eeles P., Sims O., *Building Business Objects*, John Wiley & Sons, Inc., New York, 1998.
- [8] Cotterrell Mike, Hughes Bob: *Software Project Management*, 1995, Thomson Computer Press ISBN 1-850-32190-6
- [9] Cantor Murray, *Object-Oriented Project Management with UML*, 1998, J. Wiley and Sons, ISBN: 0-471-25303-0
- [10] Royce, Walker, *Software Project Management: A Unified Framework*, 1998, Addison Wesley, ISBN:- 0-201-30958-0
- [11] Davis Alan: *Software Requiements - Objects, Functions and States*, 1993, Prentice Hall ISBN 0-13-562174-7
- [12] Kotonya, G and Sommerville, Ian, *Requirements Engineering: Processes and Techniques*, 1999, J. Wiley and Sons,
- [13] Rumbaugh James, Blaha Michael, Premerlani William, Eddy Frederic, Lorensen William: *Object-Oriented Modelling and Design*, 1991, Prentice Hall, ISBN 0-13-630054-5
- [14] Booch Grady, Rumbaugh James and Jacobson Ivar, *The Unified Modeling Language User Guide*, 1998, Addison-Wesley, ISBN: 0-201-57168-4
- [15] Rumbaugh James, Jacobson Ivar, and Booch Grady, *The Unified Modeling Language Reference Manual*, 1999, Addison-Wesley, ISBN: 0-201-30998-X

- [16] Carteret Carina, Vidgen Richard.: *Data Modelling for Information Systems*, 1995, Pitman Publishing ISBN 0-273-60262-4
- [17] Date C.J., *An Introduction to Database Systems (6th Edition)*, 1995, Addison-Wesley, ISBN: 0-201-82458-2
- [18] Fowler, Martin with Scott Kendall, *UML Distilled (2nd Edition)*, 1999, Addison-Wesley ISBN: 0-201-65783-X
- [19] Satzinger J. W. and Orvik T. U.: *The Object-Oriented Approach - Concepts, Modeling and System Development*, Boyd&Fraser 1996.
- [20] Eeles P., Sims O.: *Building Business Objects*, John Wiley & Sons, Inc., New York, 1998.
- [21] Mellor Stephen, Shlaer Sally: *Object Lifecycles: Modeling the World in States*, Yourdon Press, 1992
- [22] *MetaEdit CASE Tool* (program and documentation), Metacase Ltd., Finland, <http://www.metacase.com>
- [23] *Domain Specific Methodology – 10 Times Faster Than UML*. Metacase Ltd., Finland, <http://www.metacase.com>
- [24] Goldberg A., Rubin K. S.: *Succeeding with Objects - Decision Frameworks for Project Management*, Addison Wesley, Reading Mass, 1995.
- [25] Boehm, B. W.: *Software Engineering Economics*, Prentice-Hall, Englewood Cliffs, 1981
- [26] Ambler S.: *Process Patterns – Building Large-Scale Systems Using Object Technology*, SIGS Books 2000, ISBN 0-521-64568-9
- [27] Ambler S.: *More Process Patterns – Delivering Large-Scale Systems Using Object Technology*, SIGS Books 2000, ISBN 0-521-65262-6