

Object-Oriented Conceptual Modeling and Simulation of Health Care Processes

Radek Hřebík^(✉), Vojtěch Merunka, Zuzana Kosejková,
and Pavel Kupka

Faculty of Nuclear Sciences and Physical Engineering,
Department of Software Engineering, Czech Technical University in Prague,
115 19 Břehová 7, Prague, Czech Republic
{radek.hrebik, vmerunka, knoflice, kupka.pavel}@gmail.com

Abstract. This paper presents our own approach to modeling, visualization and simulation of biology-related processes. It presents the own BORM (Business and Object Relation Modeling) approach, which is an application of the object-oriented paradigm and finite-state machines. The first part of this paper discusses the motivation for the new approach and explains its theoretical foundation. The second part presents our practical experience with BORM as the method enabling necessary capture of requirement and verification activities for the analysis and design of information systems. The real example of the largest Prague Hospital is used. The BORM method has been used in the last 17 years on various projects in the Czech Republic and Central-European region.

Keywords: BORM · Requirement engineering · Process modeling and simulation · Biology-related processes · Sociotechnical processes · Object-oriented approach · Finite-state machines

1 Introduction

Business process models show and animate (when they are simulated) the collaboration of more participants within the solved system. Such approach is needed for simulation, validation and verification of the real problems [1]. The issue is stressed in specific areas of technical systems analysis and design, e.g. in area of agriculture, landscape management and also, as it is used in this paper, in the visualization of law-based processes of country planning. A very important purpose of such business model is to create and simulate an interconnected complex system where local actors, citizens, regional government, various interested organizations and partners and other participants mutually communicate [3, 7]. In addition to that, business process models are also the foundation of subsequent system modeling activities of software engineering, organizational design and management consulting. Typical way of performing these activities is to start directly with drawing process diagrams just during the initial interviews. But in this paper, we present the idea, that for better modeling, we need to use a specific textual technique, which helps us to recognize, define and refine our initial set of business process participants and their properties before the graphical business process model is assembled.

2 Motivation

Expected output of the business process modeling and simulation activities is information or data in a form that can be directly used as an input for implementation of the system in the spirit of software engineering and organizational modeling and management consulting. However, this is not the easy case; there are following issues described by Ilgen and Hulin [7] and Aalst [1]:

Oversimplification - while trying to at least finish business and organizational model we are forced to simplify the problem being modeled, and

Inability - some important details cannot be recorded because of the poorly used method.

A perennial problem with the development of business systems is the existing communication gap between analysts and domain experts; each live in their own well defined and complex cultures. This gap is represented in the constant failure of simulation model designers to fully capture the requirements of any proposed business system. In our experience, gathered during the last ten years working on major projects, not all system requirements are known at the start of the project, and the customer expects that their discovery and refinement will form part of the project [7]. This problem is complicated further, since the function of any major system developed has a significant impact on the organizational and management structure of the company or organization where the system is being implemented.

3 Borm Approach

3.1 Method Basics

Business Object Relation Modeling (BORM) is an approach to both process modeling and the subsequent development of information systems [8, 9]. It provides an approach that facilitates the description of how real business systems evolve, change and behave. 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, databases and distributed environments. Subsequently, it has been realized that this method has significant potential in business process modeling and other related business issues. BORM has been used in last 17 years (1998–2015) for a number of business consulting and software engineering projects including the health care processes IS, as a tool for business process reengineering in the electricity supply, gas supply industry and telecommunication network management, several business process simulation projects in various areas of modeling and simulation with subsequent IS development as well as in organizational modeling and simulation of regional management project concerning the analysis of the legislation and local officials' knowledge such as living situations, law, country planning etc.

Process approach and object orientation are the pillars of the BORM method. It is the application of principles that are successful in the field of modeling and software. The basis of the object approach is the notion that each action must have an object that

executes it; or, vice versa; that each object must have some activity in a conceptual model. It is impermissible to have an action without an object, or an object without an action. This is BORM interpretation of the Model-driven architecture (MDA) approach [10].

Any modeling and simulation tool and any diagramming technique used at this kind of business projects should be comprehensible to the stakeholders, many of whom are not software engineering literate [5]. Moreover, these diagrams must not deform or inadequately simplify requirement information. It is our experience that the correct mapping of the problem into the model and subsequent visualization and possible simulation is very hard task with standard diagramming techniques. We believe that the business community needs a simple yet expressive tool for process modeling; able to play an equivalent role to that played by Entity-Relation Diagrams, Data-Flows Diagrams or Flow-Charts over the past decades. One of the strengths of these diagrams was that they contained only a limited set of concepts (about 5) and were comprehensible by problem domain experts after few minutes of study. Unfortunately UML approach (as well as BPMN) lost this power of simplicity and clarity [17].

3.2 Combination of the OOP and FSM

Currently there is not a ‘standard solution’ to the problem of gathering and representing knowledge. That is reason why we developed and successfully used our own UML-based BORM process diagramming technique [17] and our own way to start object-oriented business system analysis recommended by Taylor [16] and together with Scheldbauer [13] prefer this approach before the semantically different BPMN [6, 15].

BORM innovation is based on the reuse of old thoughts from the beginning of 1990s regarding the description of object properties and behaviour using finite state machines (FSM). The first work expressing the possible merge of Object-Oriented Paradigm (OOP) and FSM was the book by Shaller and Melor [12]. One of the best books speaking about the applicability of OOP to the business modeling was written by Taylor [16]. These works together with our practical experience is the reason to believe that the business requirement modeling and simulation and software modeling could be unified on the platform of OOP and FSM.

The object-oriented approach has its origins in the researching of operating systems, graphic user interfaces, and particularly in programming languages, that took place in the 1970s. It differs from other software engineering approaches by incorporating non-traditional ways of thinking into the field of informatics. We look at systems by abstracting the real world in the same way as in ontological, philosophical streams. The basic element is an object that describes data structures and their behavior. In other modeling approaches, data and behavior are described separately, and, to a certain extent, independently. OOP has been and still is explained in many books (in [14], for example), but we think that this one by Rubin and Goldberg [11] is written by OOP pioneers and belong to the best.

In the field of theoretical informatics, the theory of automata is a study of abstract automaton and the problems they can solve. An automaton is a mathematical model for a device that reacts to its surroundings, gets input, and provides output. Automaton

can be configured in a way that the output from one of them becomes input for another. An automaton's behavior is defined by a combination of its inner structure and its newly - accepted input. The automata theory is a basis for language and translation theory, and for system behavior descriptions. Its usage for modeling and simulation in software engineering activities has been described by Shlaer and Mellor in [12] and many newer publications. The idea of automata also inspired behavioral aspects of the UML.

3.3 Modeling Cards

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 are modified modeling cards from the Object Behavior Analysis (OBA) being firstly published in [11]. Original OBA is an only text-based method and used a large set of form sheets, textual lists and tables for storing and manipulating the information being processed. Modeling cards are structured texts, various lists and tables and so-called modeling cards (textual forms).

In BORM, we do not start directly by drawing the process diagrams. Process diagrams are the subsequent refined visual representation of the information collected by the modeling cards.

1. *Modeling card of a scenario* clarifies the entire process contours, process participants, necessary legislation, documents etc. (see example in Fig. 1)
2. *Modeling card of a participant* is a textual description of some role in a process. It has similar structure as scenario card, but seen from the different perspective of particular participant (e.g. process actor). Participant modeling cards are subsequently refined into several FSM. (see examples in Figs. 2 and 3)

Business process diagrams in BORM, or Object-Relationship Diagrams (ORD), are visual representation of processes and objects inside of processes obtained by modeling

Patient in the hospital - scenario	
<i>initiation</i>	<i>participants</i>
Patient needs to stay in the hospital.	Patient Doctor Medical records Insurance account Release report
<i>actions</i>	
Receiving the patient, his inclusion on the bed, procedures and transfers.	
<i>result</i>	
Patient is released from the hospital or the patient's death.	

Fig. 1. Patient in hospital - scenario example

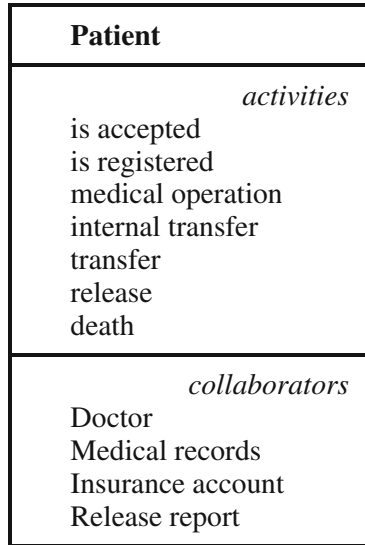


Fig. 2. Patient

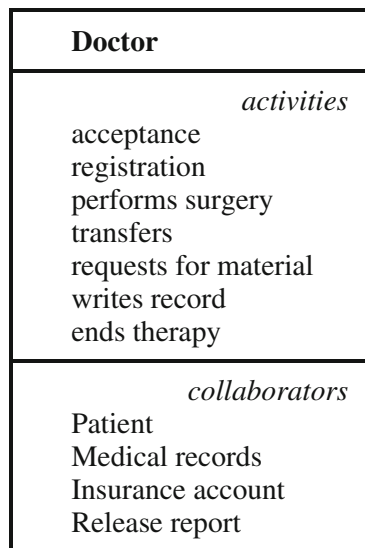


Fig. 3. Doctor

cards technique. Process diagram consists of participants, their states and transitions and their mutual communications. Each participant is composed of a set of states, activities and transitions (communications). Formally, it is a Mealy-type FSM. [12] Conceptual link within one participant can be considered as a transition between states, it contains no data, because it is only behavioral concept. On the other side communication between more participants may contain the data and therefore can be considered as data flows between activities of these participants making together some concrete process. Therefore a whole process diagram can be seen as a set of several finite state machines where each FSM represents just one participant.

If we consider the basic concept of the FSM, each i -th participant will be represented as a unique entity, defined as 5-tuple $P_i(S_i, I_i, \delta_i, s_i^0, s_i^e)$, where:

- S_i is a finite non-empty set of states which the participant may be in it.
- I_i is a finite non-empty set of all possible inputs
- δ_i represents the activities carried out, i.e. transitions between states, $\delta_i : I_i \times S_i \rightarrow S_i$.
- s_i^0 is the initial state of the process, $s_i^0 \in S_i$
- s_i^e is the final state of the process, $s_i^e \in S_i$

The participant starts from the state s_i^0 and according user input I_i and actual state transfers itself into a next state. In case the participant ends in the state s_i^e we say that P_i accepts user word from input I_i^* . We allow reading an empty symbol X so the participant can continue to the next state even without any user input.

So far we have considered a model with only one participant. Let's extend our model by N not communicating participants $P_1 \dots P_N$ simulated together. Let's suppose that user input symbols differs for each participant.

$$\bigcap_{i=1}^N I_i \subseteq \{\varepsilon\} \quad (1)$$

We can define a finite state machine P_Σ , which will be composition of the partial automata representing individual participants. That machine will simulate all participants together.

$$S = S_1 \times \dots \times S_N, \quad s_0 = (s_1^0, s_2^0, \dots, s_N^0), \quad s_e = (s_1^e, s_2^e, \dots, s_N^e) \quad (2)$$

$$I = \bigcup_{i=1}^N I_i \quad (3)$$

Therefore inner states of P_Σ are tuples of length N composed of individual participant's inner states. Same can be said about start state and end state. We suppose that user input symbols are different for each automata thus we can easily define action function for compound FSM with the help of individual action function X .

$$\delta : I \times S \rightarrow S \quad (4)$$

$$\delta(i, s_1, \dots, s_N) = (s_1, \dots, s_{j-1}, s'_j, s_{j+1}, \dots, s_N) \mid \exists j \in \hat{N}, \delta_j(i, s_j) = s'_j \quad (5)$$

If we didn't need to capture the participants communications, we considered above for a full description. So far we are using finite automata to describe participants of the process and introduced a model, which is made up of one finite state machine composed of sub-machines. Each sub-machine corresponds to a just one participant in the process. Business processes are not made up of isolated participants. They also contain communication between participants and there can be also communications across processes. Communication is realized by sending or receiving messages. Messages are sent during the execution of activities, i.e. within participants Activities. Communication model used in BORM process diagrams assumes that communication

operation is atomic and is done just between two participants [9]. In case any listening participant is not in the appropriate listening state the action which will lead to sending the message cannot be performed. Thus the communication can be viewed as an action which can be performed only when each participant is in the appropriate state and leads to transition of all communicating participants to the new states. Unfortunately this approach using only basic concept of finite state machine has not been enough. There is concept of communicating finite state machines (CFSM) which allows describe that kind of communications. The presence of communications requires a significant change from the above definition of the participants using basic concept of FSM, but allows us to describe the process diagram as a whole. The following process description is based on the basic concept of a finite automaton, but it enhances the part of the model of communicating finite state machines that are necessary to capture the mutual communication participants.

Business-process diagram representing a particular process can be defined as a finite set of participants.

$$BP = \{P^i\} \quad (6)$$

Each participant then can be described as an ordered 6-tuple

$$P^i = (S^i, -M^i, +M^i, f^i, g^i, s_1^i)$$

where:

- S^i is a finite set of all possible states which the participant may be in it
- $-M^i$ is a finite set of all outgoing messages
- $+M^i$ is a finite set of all received messages
- f^i represents the activities carried out, i.e. transition between states. The transition function can be defined as $f^i : S^i \times +M^i \rightarrow S^i$
- g^i is output function that can be defined as $g^i : S^i \times +M^i \rightarrow -M^i$
- s_1^i is the initial state of participant P^i , if $s_1^i \in S^i$

Without loss of generality, we assume that the participant will not send the message to itself

$$-M^i \cap +M^i = \emptyset \quad (7)$$

The set of all messages P^i participant will be the union of the set of all outgoing and incoming messages

$$M^i = -M \cup +M \quad (8)$$

Without loss of generality, assume that within the entire communications of a system is just one identical m_i . Each message has only one recipient and one sender.

$$\forall m_1 \in M^1, m_2 \in M^1 : m_1 \neq m_2 \quad (9)$$

The set of all messages M^i consists of an ordered triple $\langle \sigma^i, in^i, out^i \rangle$

$$M^i = \{ \langle \sigma^i, in^i, out^i \rangle \} = \{ m \}^i, \quad (10)$$

where:

σ^i is representing the transition

in^i, out^i are data

Each message has its sender and recipient

$$\forall P^i : \bigcup_i -M^i = \bigcup_i +M^i \quad (11)$$

We can define functions $data(P^i)$ and $in(m^i)$ for recipient where

$$data(P^i); in(m^i) = in^i; m^i = \langle \sigma^i, in^i, out^i \rangle \quad (13)$$

$$in(m^i) = in(\langle \sigma^i, in^i, out^i \rangle) = m^i \quad (14)$$

Analogously we define functions $data(P^j)$ and $out(m^i)$ for sender:

$$data(P^j); out(m^i) = out^i; m^i = \langle \sigma^i, in^i, out^i \rangle \quad (15)$$

$$out(m^i) = out(\langle \sigma^i, in^i, out^i \rangle) = m^i \quad (16)$$

Although the exchange of messages carried out from the perspective of BORM semantics at the same time, if we apply the theory of finite automata, we must distinguish the state before the transition and the new state after transition. Sent or received message at time $t+1$ will depend on the state at time t .

Data interchange between participants we can define for recipient as:

$$data^{t+1}(P^i) = data^t(P^i) \cup in(m^{ij}) \wedge in(m^{ij}) \subseteq data^t(P^j) \quad (17)$$

And for sender:

$$data^{t+1}(P^j) = data^t(P^j) \cup out(m^{ij}) \wedge out(m^{ij}) \subseteq data^t(P^i) \quad (18)$$

Using the formal description based on the theory of finite state machines we have defined process participant which can communicate with other participants by sending messages. Process diagram consists of a set of participants obtaining partial composition of finite state machines into a single comprehensive machine. Such a complex machine will represent a whole process diagram. Chance composing machines, thereby reducing the complexity of the resulting process has been known for decades [12]. In detail, the issue of composition, minimization and generalization of finite state

machines deals such as [3], which describes the specific algorithms for their composition.

Based on the derived definition of communicating participants, we can describe any process diagram in BORM as a finite state machine. This composed machine will consist of a set of finite state machines each of which will represent just one participant. If we apply the algorithm for the composition of finite state machines described in [12] on a set of participants, we get

$$FSM^{BP} = (\mathcal{A}, \mathcal{E}, \hat{E}, \varphi, \gamma, \sigma_1) = \{P^i\} = \{(S^i, -M^i, +M^i, f^i, g^i, s^i)\} \text{ where:}$$

The transition function φ for the composed finite state machine obtained using the transition functions of each automaton represents participants.

The output function γ for the composed finite state machine obtained using the transition functions of each automaton represents participants.

As we have shown above with we can be able to describe any process model in BORM using FSM theory. The practical impact is the ability to use all the theoretical assumptions and practices that are known from the theory of FSM for process models in BORM. Therefore it can verify that all states of participants of the process are reachable and that all activities performed. It is also possible to automatically identify the state in which could lead to deadlock the process and evaluate the consistency of the model. Formal description also opens up opportunities for better implementation of the method BORM in CASE tools, especially in the construction process simulators.

4 The Project

Our project was the analysis and subsequent reorganization process in one of the largest hospital complexes in the Czech Republic - General University Hospital in Prague. Our task was to map all processes related to the system of patient care and to connect this bio-technological process-based system with management and financial systems of the hospital.

Together, we identified 37 different types of participants in 16 different top-level processes. These process models have had an average of four participants for each process and the total number of states and activities (transitions) was around 70 for each process after its decomposition.

The resulting analysis was the basis for business-process re-engineering. After subsequent organizational change, this matter was also a basis for a selection procedure for the requirement description for a management information system in the hospital.

There is an example of BORM business process diagram at Fig. 4. It shows a general process of a patient in hospital. There are four participating subjects in this process: Patient, Doctor, Medical record and Insurance account. The small rectangles within these subjects are their states. The ovals within the same subjects are their activities, which are conceivable as the transitions between the states as well. This example shows mutually connected the FSM of a Patient and the FSM of a Doctor. The

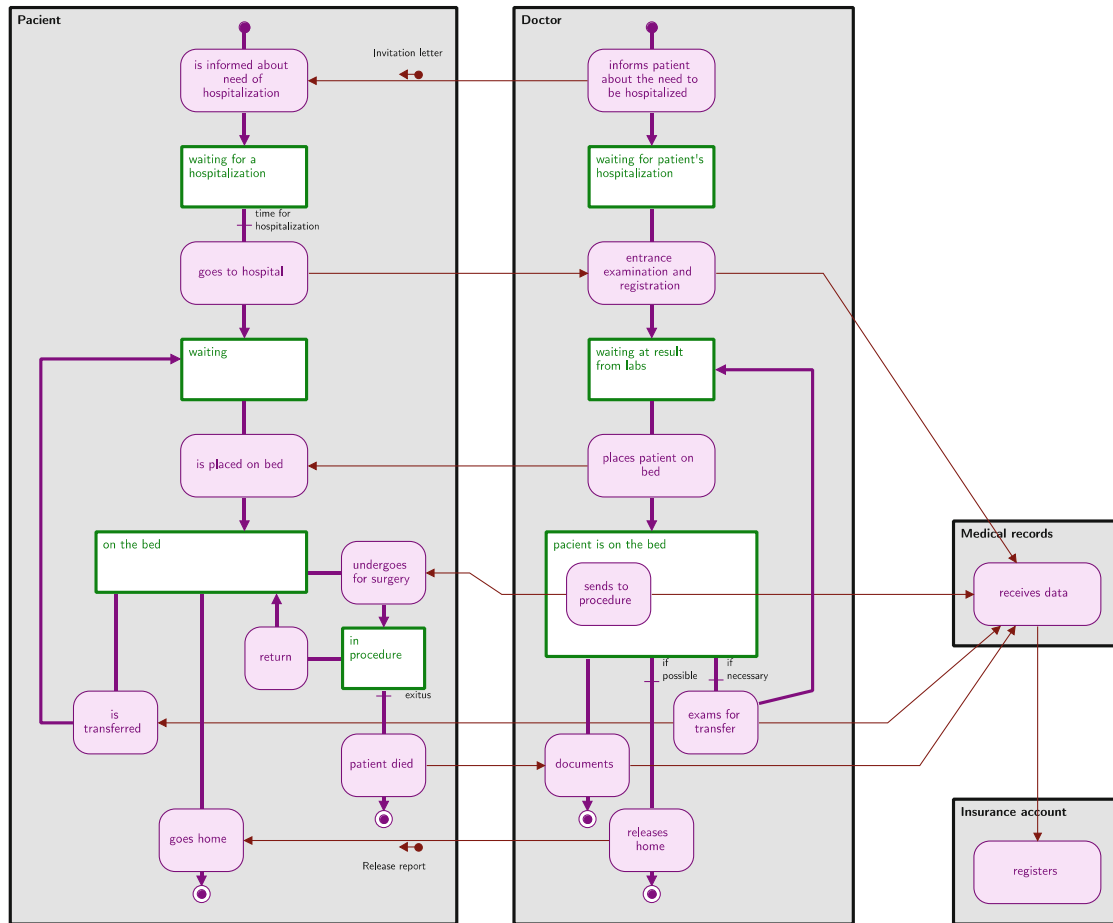


Fig. 4. Hospital care general process example (simplified)

thin arrows between these subjects are the mutual communications, and the very small arrows are the data flows being performed during these communications. If a line is crossed, it contains a condition that makes its force.

5 Conclusion

Based on our long-year experience, we decided to apply the BORM method [8, 9]. Business Object Relation Modeling (BORM) is a complex method for systems analysis and design that utilizes an object oriented paradigm in combination with business process modeling. It originated at the Loughborough University, UK in 1993. Successful BORM experience has been reported and published ever since, mostly for orchestration-intensive systems of several socio-technical projects related to the work-flow systems, industrial business processes, governmental processes, self-administration and law. BORM provides process workflow formalization that is similar to BPMN [6], however it is theoretically better founded. The basic idea of BORM approach is the combination of the object-oriented paradigm and theory of finite-state machines (theory of automata) [12]. BORM process decomposition is based

on a set of more mutually communicating Mealy-type automata representing independent process participants in their states and activities.

5.1 Future Work

We are building the project coordination support system based on the BORM orchestration. The system will be built on a client-server architecture. The server will be hosting BORM diagrams, users and other data, while a thin web client application will provide a portal solution where participants will be able to log in and operate in the workflow. Individual process diagrams are graphically animated during the simulation. Individual users have different access rights from the full process control to simple viewing. The portal will provide the following features to users according to their specific role and rights:

1. General process overview (diagrams)
2. Activities timing overview (calendar)
3. Tasks inbox
4. Process instances spawning
5. Graphical simulation
6. Audit trail (log)
7. Document storage

Acknowledgments. The authors would like to acknowledge the support of the research grant SGS14/209/OHK4/3T/14.

References

1. van der Aalst, W.M.: Business process simulation revisited. In: Barjis, J. (ed.) EOMAS 2010. LNBP, vol. 63, pp. 1–14. Springer, Heidelberg (2010)
2. Allweyer, T.: BPMN 2.0, Books on Demand GmbH, Norderstedt (2010). ISBN 978–3-8391-4985-0
3. Barjis, J.: Developing executable models of business systems. In: Proceedings of the ICEIS - International Conference on Enterprise Information Systems, pp. 5–13. INSTICC Press (2007)
4. Eriksson, H., Penker, M.: Business Modeling with UML. Wiley, New York (2000). ISBN 0-471-29551-5
5. Figl, K., Laue, R.: Cognitive complexity in business process modeling. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 452–466. Springer, Heidelberg (2011)
6. Grosskopf, A., Decker, G., Weske, M.: Business Process Modeling Using BPMN. Meghan Kiffer Press, Cambridge (2006). ISBN 978-0-929652-26-9
7. Igen, D., Hulin, C.L.: Computational Modeling of Behavior in Organizations - The Third Scientific Discipline. American Psychological Association, Washington DC (2000). ISBN 1-55798-639-8
8. Knott, R.P., Merunka, V., Polak, J.: The BORM methodology: a third-generation fully object-oriented methodology. *Knowl.-Based Syst.* **16**, 77–89 (2003). ISSN 0950-7051

9. Knott, R.P., Merunka, V., Polak, J.: Process modeling for object oriented analysis using BORM object behavioral analysis. In: Proceedings of 4th International Conference on Requirements Engineering, N.p., 2000, pp. 7–16. IEEE Xplore. Web (2000)
10. MDA – The Model Driven Architecture, OMG – The Object Management Group. <http://www.omg.org>
11. Rubin, K., Goldberg, A.: Object behavioral analysis. Commun. ACM - Special Issue on Analysis and Modeling in Software Development CACM **35**(9)(1992)
12. Shlaer, S., Mellor, S.: Object Lifecycles: Modeling the World in States. Yourdon Press, Upper Saddle River (1992). 0136299407
13. Scheldbauer, M.: The Art of Business Process Modeling - The business Analyst Guide to Process Modeling with UML and BPMN. Cartris Group, Sudbury MA (2010). ISBN 1-450-54166-6
14. Schach, S.: Object-Oriented Software Engineering. McGraw Hill, Singapore (2008). ISBN 978-007-125941-5
15. Silver, B.: BPMN Method and Style: with BPMN Implementer's Guide. A Structured Approach for Business Process Modeling and Implementation Using BPMN 2.0, 2nd edn. Cody-Cassidy Press, Aptos (2011)
16. Taylor, D.A.: Business Engineering with Object Technology. John Wiley, New York (1995). ISBN 0-471-04521-7
17. The UML standard, OMG – The Object Management Group, <http://www.omg.org>, ISO/IEC 19501