Formalization of Two-Hemisphere Model Driven Approach in the Framework of MDA

Natalya Pavlova*
natalya.pavlova@ctco.lv

Oksana Nikiforova*
ivasiuta@egle.cs.rtu.lv

Abstract: The Model Driven Architecture (MDA) is a framework built under supervision of the Object Modeling Group (OMG). The MDA separates certain key models of systems and brings a consistent structure to these models, which are structured explicitly into Computation Independent Model (CIM), Platform Independent Model (PIM), Platform Specific Model (PSM), which then is used for code generating. The problem is that there are tools to generate a code from PSM, but there are no tools to generate PSM from PIM, and formal PIM construction. Decision of the problem is in the construction of complete and consistent PIM to transform it to PSM automatically. The paper provides an effort to make an existing approach for construction of CIM, which is Two Hemisphere Model Driven, usable for further transition from CIM to PIM in more formal way, let formal transformations to PSM and code would be possible.

Key Words: Model Driven Architecture, Two Hemisphere Model Driven approach, Computation Independent Model, Platform Independent Model, formal transformation

1. Introduction

The Model Driven Architecture (MDA), built under supervision of the Object Modeling Group (OMG), separates the system business aspects from the system implementation aspects on a specific technology platform [1]. The MDA separates certain key models of systems, brings a consistent structure to these models and states the transformation abilities between these models. The models are structured explicitly into Computation Independent Models (CIM), Platform Independent Model (PIM) and Platform Specific Models (PSM) [2]. There are tools to generate a code from PSM, but are no tools to generate a PSM from PIM. It is a serious problem, and this problem decision could be in precise PIM construction to transform it to PSM automatically. Also there has to be rules for PIM checking if it defines all problem domain concepts in the correct way.

For realization of MDA principles and transformations the Two Hemisphere Model Driven (2HMD) approach is selected. Initial version of the 2HMD approach was proposed in [3], where the general framework for formalization of object oriented software development had been discussed and its application for driving school's software development had been demonstrated. The current version of the approach [4, 5] supports formal construction of CIM and semi-formal model transformation from problem and application domain into design and implementation [6]. The goal of the research is to make transformations from PIM to PSM in the framework of MDA more formal.

*Riga Technical University, Faculty of Computer Science and Information Technology, Institute of Applied Computer Systems, Department of Applied Computer Science, Meza 1/3, LV-1048, Riga, Latvia

The section 2 describes MDA in whole, it models and declared transformation, and discuss some abilities and eliminations of transformation solutions in the framework of MDA, which were analyzed in details in [7, 8]. The current state of 2HMD and proposed abilities to formalize the approach is presented in the section 3. The section 4 presents an application case of formalized in the paper 2HMD approach for abstract example of Hotel room reservation.

2. Transformation abilities in the framework of MDA

MDA is one of the most exiting innovations in information system development. First time MDA idea was published in 2001 [1]. The main principles are dividing system specification from platform independent specification, thus raise the abstraction level on which information systems are developed. MDA framework implies system development based on modeling, neither on programming activities. System development is divided into 3 stages according to the level of abstraction. The first one is Computation Independent model development (CIM), the second is Platform Independent model (PIM) development and after that Platform Specific model (PSM) construction.[2] The circuit of transformations in the framework of MDA is demonstrated in Figure 1 [8].

Figure 1. Declaration of transformation in the framework of MDA

The purpose of CIM is to represent real world system. Programming concepts are not considered at this abstraction level. PIM is describing that part of software specification, which is close to code, but is independent of platform specific features. PIM is representing a system in the way that will remain unchanged on any programming platform. Nevertheless PIM usually is accommodated to specific architecture style. PSM consist of all the information in the PIM and platform specific details are adding to it. According to circuit in Figure 1 the transformation PSM -> code is already well known and supported with wide range of special tools. PSM model is enough formal to perform automatically transformations.

There are a set of different approaches for the construction of PIM, some of them are shown in Figure 2.

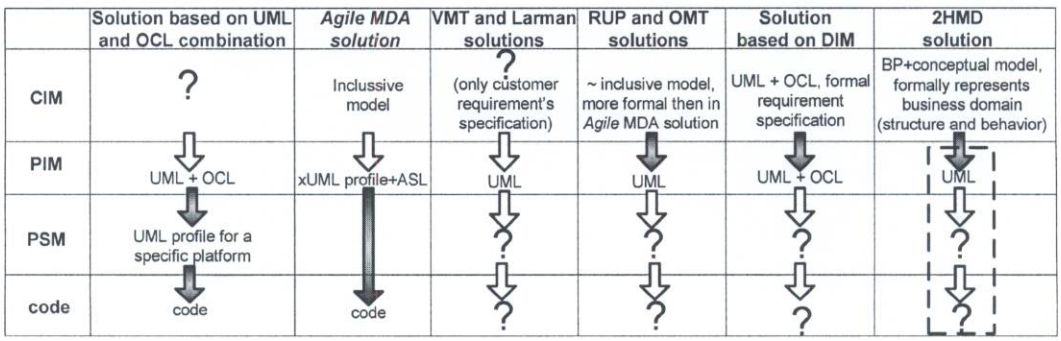

Figure 2. Realization levels of transformations in the framework of MDA in some advanced solutions

The most popular approaches are described in [2, 5, 9-14, 16]. All of them were analyzed by authors in [4–8] according to transformation abilities from CIM to PIM and its further transformations to PSM and code. Unfortunately, these solutions do not completely correspond to MDA main principles. Figure 2 shows the results of the research presented in

[7, 8], where each solution was described in detail, and information transfer during it was analyzed: methods were compared by the criteria, based on the main activities have to be performed during CIM, PIM and PSM construction and transformation from one into another. Arrows in Figure 2 shows necessary transitions have to become in the framework of MDA. Painted out arrows mark transformations, which could be performed completely or at least partly automatically. Transparent arrows indicate transformations, which are not clearly defined, and could be performed only by human intellect involving. Question-marks means, that the place of MDA framework does not defined. The current paper is the next research step in the solution of formalization problem in the framework of MDA. Authors are trying to make the 2HMD approach more formal in the transition from PIM to PSM and further code generation (the field is defined by dotted line in Figure 2).

3. Formalization of the 2HMD approach

The 2HMD approach [5] (see Figure 3) may be considered as a version of business process model driven approach. Business process modeling describes business processes on the high level of abstraction and such a representation is consistent, formal and complete. Two-Hemisphere model driven approach proposes use of business process modeling to represent systems in the platform independent manner and describe, how to transform business process models into UML models to make platform dependent system representation.

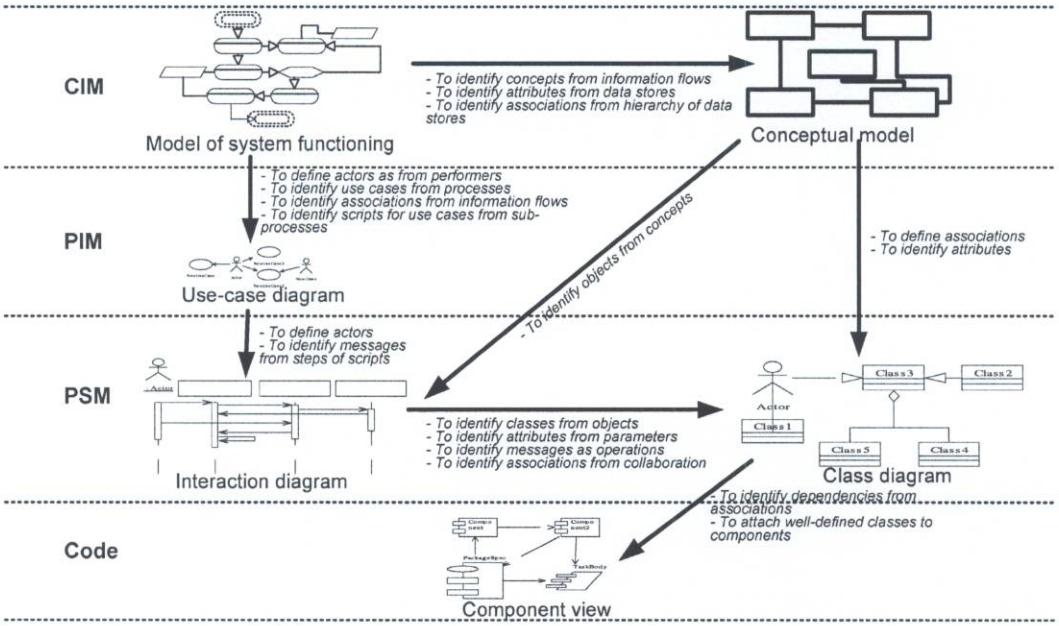

Figure 3. Model transformations in 2HMD approach

The 2HMD approach addresses the construction of information about problem domain by use of two interrelated models at computation independent model level, namely, the business process model and the conceptual model. The conceptual model is used in parallel with business process model to cross-examine software developers understanding of problem and platform independent models. Transformation abilities of models in the framework of MDA are illustrated in Figure 3.

Real-world classes relevant to the problem domain and their relationships are presented in the conceptual model. The notational conventions of the business process diagram give a possibility to identify concepts by analyzing all the data stores in the diagram. Processes to be performed by software system become use-cases in the use-case model, performers of related processes become actors in the use-case model, and scenarios for realization of use-cases may be defined by decompositions of business processes (sub-processes) corresponding to the use-cases. Interaction diagram for each use-case is based on its realization scenario (or sequence of sub-processes). Appropriate interacting objects are extracted from the conceptual model. The class diagram is based on the conceptual model and is formed according to information in the interaction diagram. The class diagram here is already a structure of a software application and contains only those classes, whose objects interact during the use-case realization. [3].

In this research authors try to formalize the 2HMD approach by introducing some additional steps of transformations and excluding some ambiguous elements. Scheme of model transformations modified is shown on Figure 4, where the changes in 2HMD approach are highlighted with grey background.

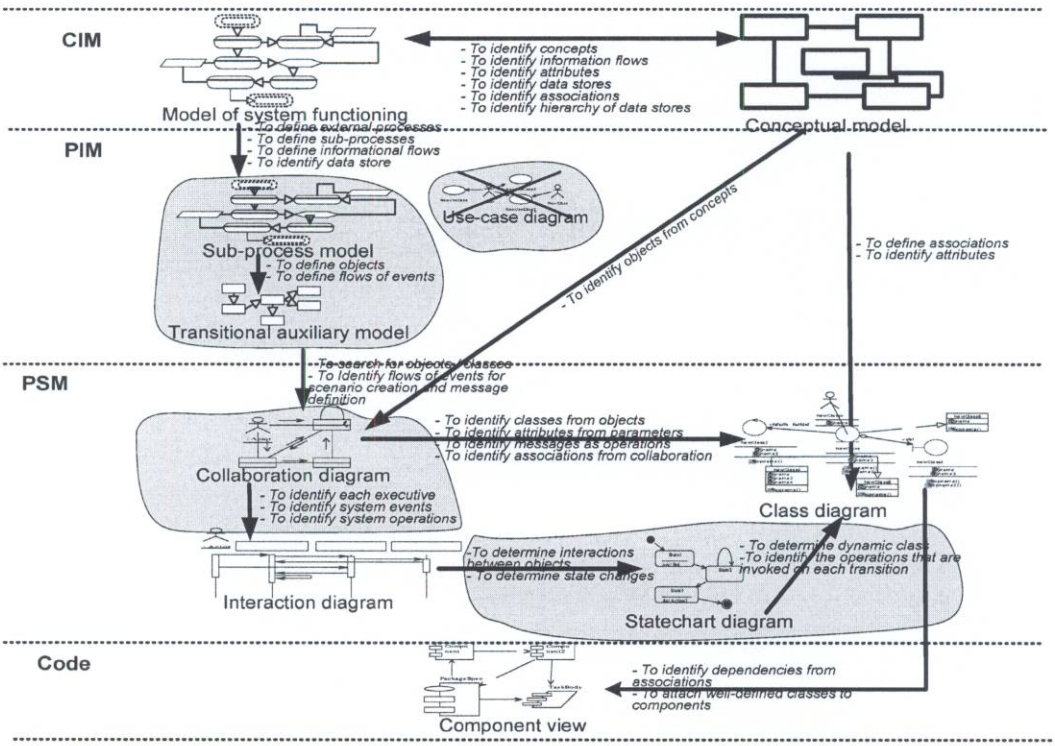

Figure 4. Transformation abilities in the framework of MDA in modified 2HMD approach

First major change is removing of use case diagram from modeling process. The reason of use case removing is problems, which could be aroused by it. These problems are discussed in different publications. The most mentioned problems are the following: a set of use cases does not provide a developer with all of the information about the client's requirements; on account of simplicity of use cases analysts don't have to work very hard to understand the basic Use Case concepts, because of it quality of developed use case diagram falls; the concrete methods how use cases should be selected from user requirements are missed.

In the original 2HMD approach PIM level of the system is presented as use case diagram derived from business process diagram, so far information loss is possible and ambiguous

information including can appear. PIM level (Figure 4) is presented as two models – sub-process model and transitional auxiliary model. Sub-process model is constructed based on Model of system functioning, for each business process to be automated. It is the formal base of further system design. For easing of transition from sub-process model to interaction model transitional auxiliary model is used. Transitional auxiliary model is generated from sub-process model using theory of graph transformation and synthesis [15]. The nodes of sub-process model become arcs of transitional auxiliary model, and arcs of sub-process mode become nodes of transitional auxiliary model.

Collaboration and state diagrams are introduced at the level of PSM. Collaboration diagram is added as more logical transition from sub-process diagram to present object interaction. Information flows from collaboration diagram to class diagram is the same as from interaction diagram to class diagram, which is shown on the Figure 4. New information flows are added – information from collaboration diagram to interaction diagram, from interaction to state and from state to class.

Class diagram is amplified with elements necessary for code generation such as interfaces, OCL constraints, stereotypes, operations, attributes, its signatures and so on. As in the original 2HMD approach information into class diagram transfers from conceptual model. And the additional part is the refinement of class diagram with the information coming from state transition. Class diagram is a final model of PSM. The following stage of MDA is generating of the component model which is similar with original 2HMD approach.

The modified 2HMD approach is illustrated by the small practical example – hotel room reservation, where is shown the most important transitions between models of system design.

4. Application Case: Hotel Room Reservation

Hotel room reservation is chosen as example to illustrate how the formalized 2HMD approach could be applied. There are shown main steps, which should be done during software development for hotel. Brief description of the room reservation process is the following:

A system gives the opportunity to reserve a room in the hotel. Client fills a blank for reservation of the room in the hotel by using hotel's Web-site. Client has to input his name, type of the room to reserve (single or double), and the period for staying in the hotel. The system updates the information about requested room and if a room is available for the defined period the system makes a reservation and sends a confirmation. If there is no available room in the hotel the system displays a message that reservation is impossible.

When client arrive to the hotel, at the reception he has to request the room reserved and the administrator has to check all the information. Administrator input all the information about client's staying in the hotel. Every day 1 p.m. the system checks reservation records to define either reservation is valid (i.e. client is taking the room requested) or invalid (i.e. at the requested date client is not coming to the hotel). In the case reservation is invalid the room defined in the reservation is marked as free for further reservations.

Computing independent model level: The first model, proposed by 2HMD approach is business process model. The simplified version of the business process for room reservation is shown in Figure 5. The 2HMD approach declares that business process model serves as a base of the following construction of system model. To identify the real-world classes relevant to the software system and their relationships was preformed conceptual modeling. The conceptual model shows the objects which exist in the hotel problem domain and their relations to other objects. It is expressed in terms of classes. The notational conventions of the business process diagram give a possibility to identify concepts also by analysing all data stores in this diagram. On the Figure 5 is shown only business process diagram as a base for the conceptual model. In fact for conceptual model is using sub-process diagrams too. One of these diagrams is shown on Figure 5. Concepts, which looks like received from process real are received from data stores of sub-process diagrams.

The hierarchical structure of data stores in the business process model gives a possibility to detect potential relationships between system concepts. Data stores are characterized by a set of attributes, which are useful for definition of class structure.

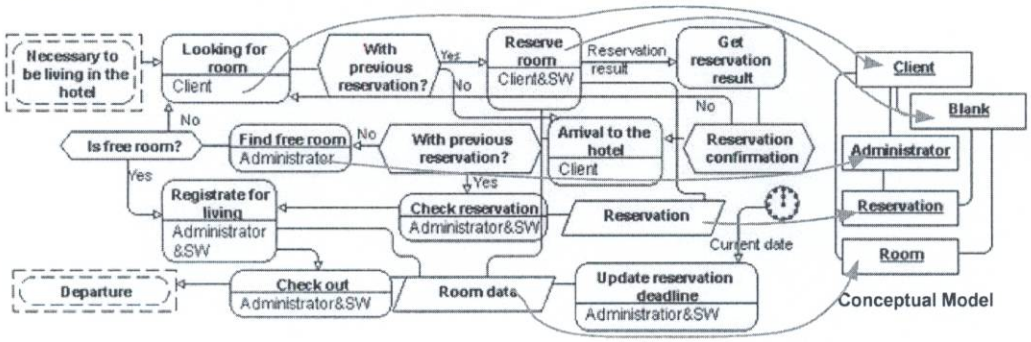

Figure 5. Construction of a conceptual model for the hotel

Platform Independent Model: The sub-process diagrams were constructed only for automated processes. Analysis of the business process identifies the boundary of the software system and helps to decide, which processes refer to the software system. On the Figure 6 is shown sub-process diagram for room reservation. Based on this diagram was constructed collaboration diagram for room reservation process. On the PIM level there is one more model - transitional auxiliary model. It serves to make easier transition from sub-process models to collaboration models. For the transition from sub-processes to transitional auxiliary model as discussed above is used theory of graphs. During system analysis the same transitions was performed for each automated sub-process.

Platform specific Model: The first model of Platform Specific level is collaboration model. This model is received from sub-processes and transitional auxiliary models, as it is shown on Figure 6. Received collaboration diagrams were used as a base for class diagram construction. So as for class diagram construction were user interaction and state diagrams too, but development of these models is not shown in this paper.

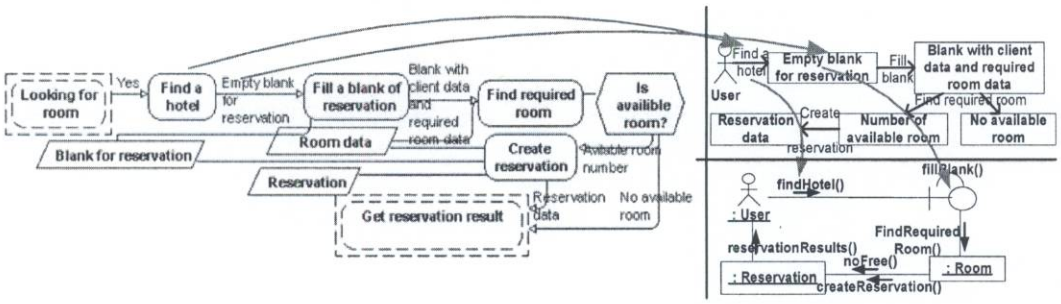

Figure 6. Generation of object collaboration from business processes

To construct class diagram from collaboration model were used objects interactions among it, operations and attributes. State and interaction diagrams serve for specifying of operations and dynamic classes.

Received class diagram for hotel room reservation is shown on Figure 7. There are introduced stereotypes, interfaces, attributes and operations. Data types are defined, but don't show to don't overload the Figure 7.

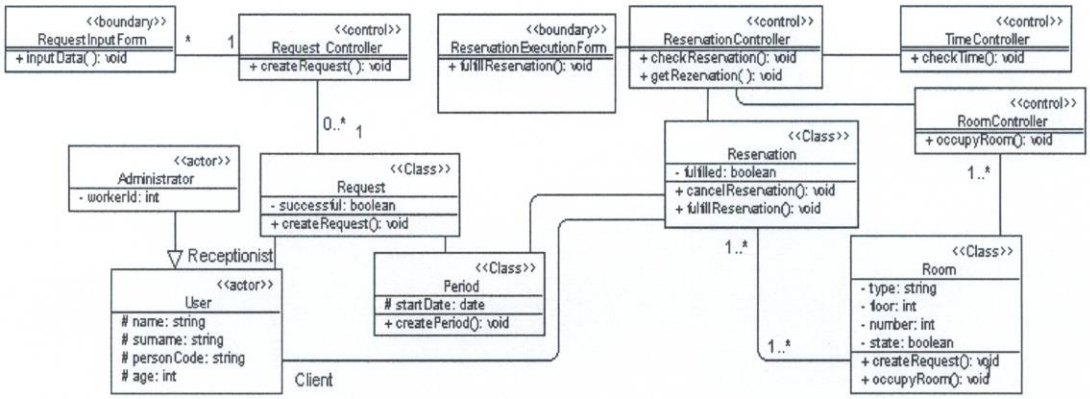

Figure 7. Final class diagram for hotel room reservation

Further implementation of the design model by components is based on traditional object-oriented approach.

5. Conclusions

According to the framework of MDA the Platform Independent Model (PIM) has to be derived from Computation Independent Model (CIM). CIM role is to make a bridge between business environment experts and information system development experts to provide correct information flow from business to software domain to make further construction of Platform Specific Model (PSM) would complete and consistent and code generation from it possible.

To process PIM automatically it has to be represented and derived from CIM in the formal way. PIM is describing part of business environment, which is important for information system and some realization aspects that are independent of any particular platforms. Today we do not have methods how to represent PIM level information to maintain all necessary information and express it on the high level of abstraction and in platform independent way. That is why today the main problem is PIM formalization.

Authors are trying to produce a solution for derivation of PIM from information about problem domain presented in the form of Business process diagram and conceptual model, by introducing some aspects of theory of graph and transitions from one model into another by replacing graph vertices with edges and vice versa.

The paper can be of interest for scientists and practitioners involved in the stream of people intent in the MDA realization ideas.

This work has been partly supported by the European Social Fund within the National Program "Support for the carrying out doctoral study program's and post-doctoral researches" and by a grant No. ZP/2005-02 of Riga Technical University within the project "Application of Two-Hemisphere Approach for Development of Flexible Architecture for Software Engineering Body of Knowledge".

References

1. MDA Guide Version 1.0.1/ Internets.- http://www.omg.org/docs/omg/03-05-01.pdf
2. Anneke Kleppe, Jos Warmer, Wim Bast, MDA Explained : The Model Driven Architecture – Practise and Promise, Addison Wesley, 2003., 192.lpp.

3. Nikiforova O. General Framework for Object-Oriented Software Development, Scientific Proceedings of Riga Technical University, The 2nd Series – Computer Science, Applied Computer Systems, 2002.
4. Nikiforova O., Kirikova M., Enabling Problem Domain Knowledge Transformation during Object Oriented Software Development, Conference of Information System Development, Melbourne, Australia, 2003
5. Nikiforova O., Kirikova M., Two-Hemisphere Model Driven Approach: Engineering Based Software Development, Proceeding of the 16th International Conference Advanced Information Systems Engineering, Springer – Verlag Berlin Heidelberg, 2004., lpp. 219 – 233.
6. Nikiforova O., Kirikova M., Wojtkowski W., Role of Models in Knowledge Transfer during OO Software Development, The 15th European – Japanese Conference on Information Modeling and Knowledge Bases, 2005, Tallinn, pp. 305-320
7. Pavlova N., Ņikiforova O. An overview of advanced approaches for construction of platform-independent system model, Scientific Proceedings of Riga Technical University, The 5th Series – Computer Science. Applied Computer Systems, 2005, pp. 156-168
8. Nikiforova O., Kuzmina M., Pavlova N. Formal Development of PIM in the Framework of MDA: Myth or Reality, Scientific Proceedings of Riga Technical University, The 6th Series – Computer Science. Applied Computer Systems, 2006
9. Ambler S.W., Approaches to Agile Model Driven Development (AMDD) /Internets.- http://www.agilemodeling.com/essays/amddApproaches.htm#Manual
10. Ceponiene L., Nemuraite L. Reconsilation of UML models for development of information systems, Scientific Proceedings of Riga Technical University, The 5th Series – Computer Science. Applied Computer Systems, 2005
11. Stan H., Integrating Computation Independent Business Modeling Languages into the MDA with UML2, available at http://www.omg.org/docs/ad/03-01-32.pdf
12. European Software Institute : Enriched PIM with Project Management Information, available at http://modeldrivenarchitecture.esi.es/mda_publicDocuments.html
13. Object Constraint Language Specification, v 1.1, available at http://www.omg.org/cgi-bin/apps/doc?ptc03-10-14.pdf
14. Raistrick C., Francis P., Wright J., Carter C., Wilkie I. Model Driven Architecture with Executable UML, Cambridge university press, 2004., 394.lpp
15. L. Steen editor, For All Practical Purposes: Introduction to Contemporary Mathematics 3ed. W. H. Freeman and Company, New York 1994.
16. Jacobson I., Booch G., Rumbaugh J., "*The Unified Software Development Process*", Addison-Wesley, 2002