RIGA TECHNICAL UNIVERSITY
Faculty of Computer Science and Information Technology
Institute of Applied Computer Systems

Erika ASNINA

FORMALIZATION OF PROBLEM DOMAIN MODELING WITHIN MODEL DRIVEN ARCHITECTURE

Doctoral thesis summary

Scientific adviser
Dr.habil.sc.ing, professor
J. OSIS

RTU Publishing House
Riga 2006

GENERAL CHARACTERISTICS OF THE RESEARCH

One of the primary activities that is very important in software development is problem domain analysis. Software developers' community uses different techniques for identification and specification of problem domain characteristics and requirements for **a** planned system. It must be mentioned that they are aimed primarily at application analysis, while the problem domain is regarded almost as a black box describing a number of aspects of the system. Until now, there is **a** lack of formal ways to map knowledge about the problem domain into the software development process in those approaches.

The above described situation raises a few questions. The first one, which is the most important in my opinion, is a question about a gap between the application and the real world, i.e. what domain is to be modeled first: the domain of today reality ("as is") or the domain of customer expected reality ("to be"). The second question is the separation of concerns in specifications, and the related one is about unambiguous representation of the system in specifications. The main idea is that system requirements are constraints set by real world phenomena not vice versa. This means that if we develop software for some purposes in the real world, we must know how it will affect the world. It has critical importance for mechatronic and embedded systems, where failure costs could be human lives as well as for complex business systems, wherein failures could lead to huge financial losses.

Formal mathematical methods could reduce inaccuracies and ambiguities of specifications. However, contrary to informal methods, they usually require additional efforts in study and use. Therefore, most developers of modeling languages try to avoid pure usage of those methods and suggest practice of so called semi-formal modeling languages that are formally described, but do not have any formal mathematical foundations like, for example, Unified Modeling Language *(UML).*

The actuality of the research is related to software development quality and adequacy of the developed product to the environment, where this product will operate. A finished software product that is inadequate to the work environment requires high financial costs for correcting that inadequacy. However, the new architecture for separation of concerns, i.e. the MDA, will benefit only if it provides application model conformity to the problem domain, otherwise it would be just another architecture of the existing ones.

The purpose of the given work is introducing more formalism into the problem domain modeling within OMG Model Driven Architecture *(MDA).* The main idea is to determine the relation between the problem domain (in real world) and its representation in the application domain (in specifications) to be reasonable. This means introducing a more formal definition of consistency between real world phenomena and an application that will work within this whithout introducing complex, hard understandable mathematics.

In order to achieve the goal of the given research the following tasks are performed:

- Consider solutions of above-described questions, adopted in object-oriented software development such as the MDA, the UML, and use case driven approaches, as well as consider problems that are unsolved by them until this time;
- Describe main characteristics of the formal methods chosen to analyze problem domain, and compare their potential for formalization of the system functionality and structure description;
- Develop a formal approach that allows achieving the following: a) a formal specification of problem domain characteristics that can be constrained by requirements imposed for the corresponding application, b) a formal specification that describes application business processes at the high level of abstraction, conformed to business processes of the real world system;

- Compare main features of the developed approach with the Unified Process, Business Object Oriented Modeling and Alistair Cockburn's approach from the aspect of problem domain modeling;
- Demonstrate that the developed approach reaches the aim established by the case study.

The research object is the formalizing of modeling of conformity between the real world phenomena and the application that will work within it.

The research subject is problem domain modeling techniques that are used for software system modeling with use cases in the object oriented software development within the Model Driven Architecture.

For research tasks performance word analysis, graph theory, capabilities of topological modeling of system functioning, and categorical logic are used. **These techniques** are used as follows:

1) Word analysis is used for informal description analysis;
2) Graph theory rules are used in transformations between different viewpoint graphs and in their refinement thereof;
3) Topological functioning modeling is used for analysis of the domain functionality and static relations;
4) Categorical logic is used in specifications of complex relations.

A scientific innovation of the research is *functioning and structure definition, which is noncontradictory to the problem domain "as is", in the application specification.* It is done using formal mathematical constructs and rules supported by the topological functioning modeling and universal arrow logic. This developed formal background enables less intuitive problem domain "as is" model construction, moreover, it also formalizes some aspects of the conformed information specification by use cases. Characteristics of the developed approach called *Topological Functioning Model for MDA (TFMjMDA)* are compared with the broadly used modeling technique features. Besides that, this research includes comparison of formalization capabilities of the topological functioning modeling with those capabilities of Sowa's conceptual graphs and universal categorical logic, which makes it easy to understand the formal base strengths and weaknesses.

The practical value of the research is *development of the TFMjMDA modeling approach* that enables formal problem domain analysis and conformity setting between the planned software system and the existing one in the real world. It allows to be sure that the planned software system is conformed to real world system. The TFMfMDA is recommended to application in case of multifunctional system modeling, where projects use use case driven approaches. The TFMfMDA application is described in detail and demonstrated in a developed case study (library functioning). The doctoral thesis defines all TFMfMDA concepts and their relations in a metamodel and an UML profile. Both metamodel and UML profile coordinate the TFMfMDA with MDA ideas and facilitate TFMfMDA comformity with other MDA modeling languages as well.

The performed **research tasks** and the developed **approaches** during this research work are the following:

- The TFMfMDA approach dedicated for application domain model conformity to the topological functioning mode! of the real world system is developed and realized in a case study; a MOF-based standalone metamodel and an UML Profile for the TFMfMDA are developed and described in detail;
- The TFMfMDA is compared with the UP, the B.O.O.M., and Alistair Cockburn's approach of use case structuring with goals in the problem domain modeling aspect; use case identification formalism within these approaches is compared;

additionally, a use case specification obtained by the developed approach is evaluated using a metric-based inspection technique.

- A method for problem domain model construction based on the topological functioning modeling is developed;
- MDA main concepts, benefits and main critics are discussed; UML formalization history and its current state are discussed; peculiarities of problem domain modeling with use cases and corresponding use case driven techniques such as the Unified Process *(UP)* with UML, the Business Object Oriented Modeling *(B.O.O.M.)* with UML, and an approach of use case structuring with goals introduced by Alistair Cockburn are discussed;
- A choice of the formalization base for problem domain modeling is established by discussion and comparison of formalization capabilities of Sowa's conceptual graphs, topological functioning modeling, and universal categorical logic. The capabilities are illustrated in examples;

The main research results are represented and published (or accepted for publishing) in the following international issues:

1) Asnina E. Formalization Aspects of Problem Domain Modeling within Model Driven Architecture// Databases and Information Systems. Seventh International Baltic Conference on Databases and Information Systems. Communications, Materials of Doctoral Consortium, July 3-6, 2006, Vilnius, Lithuania. - Vilnius: Technika, 2006 (ISBN 9955-28-013-1). - 93-104 p.
2) Asnina E., Osis J. The Computation Independent Viewpoint: a Formal Method of Topological Functioning Model Constructing// Scientific Proceedings of Riga Technical University, Series — Computer Science (5), Applied Computer Systems. -Riga: RTU, 2006 (in press).
3) Asnina E. The Formal Approach to Problem Domain Modelling Within Model Driven Architecture// Proceedings of the 9nd International Conference "Information Systems Implementation and Modelling" (ISIM'06), April 25-26, 2006, Pferov, Czech Republic. -Ostrava: Jan Stefan MARQ., 2006. - 97-104 p.
4) Alksnis G., Asnina E., Osis J., Silins J. Formalization of Software Development: Problems and Solutions// Scientific Proceedings of Riga Technical University, Series — Computer Science (5), Applied Computer Systems, Volume 22. - Riga: RTU, 2005. -204-216p.
5) Asnina E. Formalization Aspects of Problem Domain Analysis// Proceedings of the 8"' International Conference "Information System Implementation and Modelling" (ISIM'05), April 19-21, Hradec nad Moravia', Czech Republic- Ostrava: Jan Stefan MARQ., 2005.- 195-202 p.
6) Asnina E. Topological Modeling and Arrow Diagram Logic Formalism Application for Software Development// Scientific Papers of University of Latvia, Volume 673, Databases and Information Systems, Doctoral Consortium, Sixth International Baltic Conference BalticDB&IS 2004, Riga, Latvia, June 6-9, 2004. -Riga: University of Latvia, 2004.-220-231 p.
7) Asnina E. Formal Integration Perspective in the Software Development// Scientific Proceedings of Riga Technical University, Series — Computer Science (5), Applied Computer Systems, Volume 17. - Riga: RTU, 2003. - 145 - 154 p.
8) Asnina E., Osis J. Formalization Problems and Perspectives of the Software Development// Scientific Proceedings of Riga Technical University, Computer Science, Applied Computer Systems, 3rd thematic issue.- Riga: RTU, 2002. - 145-156 p.

The thesis consists of introduction, four parts, conclusions, five appendices, and bibliography. The main part of thesis is represented in 162 pages, illustrated by 49 figures and 31 tables. Bibliography includes 104 information sources.

Introduction motivates topicality of the performed research, formulates the main purpose of the research work, and necessary research tasks; additionally, it represents a brief description of the research directions.

Part I considers in detail main principles and statements (such as viewpoint separation and transformations), utilization, and critics of the MDA. It also describes in detail hardships in semantic specifications (semantic heterogeneity, syntactical overloading, semantic specification hiding in the realization, and specification logic) that leads to specification ambiguity and impreciseness. Additionally, Part 1 gives an account of UML formalism development, considering it in UML version 1.4 and the most recent one, i.e. UML version 2.0. Furthermore, this part presents peculiar properties of problem domain modeling with use case driven approaches, expounds use case formal underpinnings, and shows what problems still exist in this area.

Part 2 illustrates nature of formal methods that can be suggested in order to satisfy the goal mentioned above and compares their formalization capabilities. They are categorical logic, Sowa's conceptual graphs and topological functioning modeling.

Part 3 suggests the formal approach for problem domain modeling and derivation of the application domain model from the problem domain model and functional requirements to the application by means of topological functioning modeling called TFMfMDA.. The suggested approach is conformed to MDA principles. The TFMfMDA application is illustrated by an example.

Part 4 provides a case study of the use of the TFMfMDA to specifying library work, and compares the developed approach with the approaches considered in Part 1. Besides that, this part represents metric based inspection of obtained use cases.

Conclusions represent a summary of research results, conclusions and possible subjects of further research.

Appendix 1 represents detailed titles of the acronyms used in this thesis.

Appendix 2 represents informal description of the library functioning used in Part 4 as an information for the case study.

Appendix 3 represents functional requirements to the library application used for the case study in Part 4.

Appendix 4 represents details of the topological functioning model of the library construction, i.e. applying a closure operation.

Appendix 5 represents the detailed specifications of use cases identified by the suggested approach.

SUMMARY OF THESIS CONTENTS

Introduction motivates topicality of the performed research, formulates the main purpose of the research work, and necessary research tasks; additionally, it represents a brief description of the research thesis contents.

The first part (Modern Solutions in Object-Oriented Software Development) considers problems of the problem domain modeling and solutions of them in the object-oriented software development.

The first section describes problems that are related to semantic specifications. A visual model that is described in some modeling language has to represent the problem domain in a clear and appropriate (non-contradictory) way. However, a lack of the formal mathematical base of modeling languages as well as a problem domain variety raises the following hardships [33]:

- Syntactical overloading of modeling languages;
- Semantic specification in the realization constructs as well as aspect mixing in specifications;
- Informal or semi-formal nature of specification logic.

The second section describes main principles, a foundation model, use, and critics of Model Driven Architecture *(MDA)*. The main purpose of the Model Driven Architecture that exists since 2001 is separation of the viewpoints in specifications and strengthening the analysis and design role in the project development.

MDA authors foresee three specification viewpoints and their corresponding models [60J:

- A Computation Independent Model (*CIM*) that represents system requirements and the way in which the system works within the environment. Details of the system structure and application implementation are undetermined, or hidden;
- A Platform Independent Model *(PIM)* expresses a system in such abstraction level that renders this model to be suitable for use with different platforms of similar type;
- A Platform Specific Model *(PSM)* provides a set of technical concepts, representing different kinds of parts that make up a platform and its services to be used by an application, and, hence, does change transferring system functioning from one platform to another.

The conception of the MDA supports abstraction and refinement in models.

The core standards that are supported by the MDA are the Unified Modeling Language *(UML)*, the Meta Object Facility *(MOF)*, and the Common Warehouse Metamodel *(CWM)*.

The MDA Foundation Model formulated in 2005 [65J describes concepts used in the MDA and their interrelations. Besides that, it defines other (not only the MDA) modeling language definition and usage within the Model Driven Architecture.

The MDA foresees transformation between models from different viewpoints, but until now, it declares only four possible types of transformation [61]:

- From PIM to PIM that is PIM model refinement without adding any platform specific information;
- From PIM to PSM that is PIM model refinement with adding some platform specific information;
- From PSM to PSM that is PSM model refinement in order to enhance the model with information about component realization and deployment;
- From PSM to PIM that is PSM model abstraction from platform specific information; ideally, a result of this transformation should fit the PIM in the corresponding PIM to PSM transformation.

Additionally, each transformation must be recorded in a *record of transformation.* Within the MDA, six transformation approaches are defined: marking, metamodel transformation, model transformation, pattern application, model merging, and supplying addition information [60].

Despite all declared MDA advantages, some scientists and practitioners note possible barriers in its application [58J, [88], [100]:

- The MDA is not a panacea; it should be carefully examined and evaluated before accepting or rejecting using it for industrial needs;
- Tool vendor inertia in implementing changes without real observable benefits of all changes to be made;

- Implementation of all model transformation types, e.g. complete code generation from models can become a "nightmare" for programmers;
- It could introduce changes that have not been researched before into the development process organization.

However, despite all the abovementioned, one thing is underestimated in the software developers' community. It is a lack of traceability between computation independent and platform independent viewpoints, which does not promote elimination of the gap between business and application domains [15], [73], [79], [98]. Thus, the current research is a step towards MDA completeness and, therefore, towards MDA maturity.

The third section describes the development of the MDA formalism. The UML version 1.1 was accepted as a standard in 1997. In 2005, the OMG adopted the version 2.0 as the current UML version. This section considers formalism in versions 1.4 and 2.0.

The UML metamodel is described in precise constructs, although it cannot be considered as a formal language due to the usage of natural language. In both UML versions, constructs are specified in the metamodel from three aspects [66]:

- Abstract syntax describes UML metaclasses in an UML class diagram, determines well-formedness rules, and gives brief descriptions of metaclasses and metaattributes in natural language;
- Well-formedness rules are described in natural language, and a formal Object Constraint Language *(OCL).* These rules specify attribute and association constraints defined in the metamodel;
- Semantics (construct meaning) is defined in natural language.

Base construct preciseness and separation are the main changes in the formal base of the UML version 2.0. Additionally, this version has accepted the formal basis, i.e. Petri net formalism, for the Activity Diagram foundations. Nevertheless, language authors emphasize that formalism could be introduced only in case of its evident benefits [68J.

Relative weaknesses of the UML is a large size of the notation, incoherence between diagrams, possibility of different interpretations of constructs, and allowing developers to define their own UML subsets [1], [2], [3], [46], [68], [70].

The fourth section discusses problem domain modeling with use case driven modeling approaches.

There are two fundamental aspects for system modeling that need to be distinguished: analysis, which defines *what* an application has to do with a problem domain to fit customer's requirements, and design, which defines *how* the application will be built. The line between the analysis and design is fuzzy in the object-oriented software development [79]. Besides that, the analysis implies that a software developer should analyze client's current situation as precise as possible, and gather requirements to the system planned to build because there is no separate object oriented requirements gathering in the object oriented software development. Usually, requirement analysis is erroneously considered as determination of what software the client *wants.* Actually, requirements must determine what software the client *needs.* Moreover, it must determine not just the software that the client needs, but also within what environment this software will work [15], [86], [98].

For requirements model representation, object oriented system analysis uses the so-called *use cases* that were introduced by Ivar Jacobson in 1987. The main goal of introducing use cases was to improve traceability between functional requirements specification and analysis model. Therefore, I. Jacobson *et al.* separated three parts of the requirements model that apply use cases [62]:

1) A use case model that defines what exists outwards the system and what should be done by the system,

2} Interface description, and

3) A problem domain object model.

Use cases may be described in different levels of details [87]. A use case is "a special sequence of behaviorally related transactions performed by an actor and the system in a dialogue... This transaction consists of different actions to be performed... The set of all use case descriptions specifies the complete functionality of the system..." [62J. Use cases can describe common functionality (included use cases), and functionality subordinated to some condition (extending use cases). The use case and actor notation is illustrated in Figure 1.

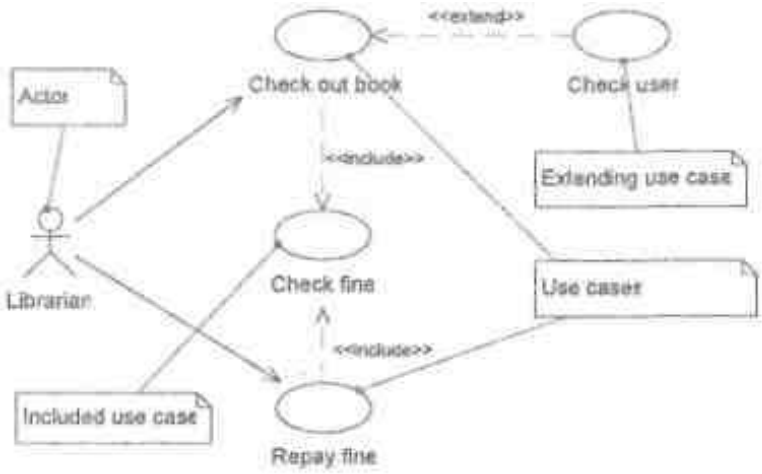


Figure 1. Use case model

Use cases have the following *advantages* [47], [87]:

- Use cases are user-centric;
- Use cases are good means for communication between developers and customers/users;
- Complex functionality of large-scale systems can be decomposed into major functions;
- Use cases have accented the use of lower-level scenarios;
- Use cases can serve as a good basis for validation of functional requirements and verification of higher-level models; use cases are traceable;
- Use cases can serve for project management as measurement in development process planning.

Use cases have the following *limitations* [28], [47]:

- Information capturing. They allow capturing interface information, and can be described at different levels of details;
- Limitation of Thinking. Their simple concept causes developers to think that requirements gathering can be limited with creation of a list of use cases; they allow excluding other requirement gathering techniques and, thus, proper problem domain analysis;
- Completeness Checking. Use cases is a notation not an approach, besides that their usage is not systematic in comparison with systematic approaches that enable identifying of all system requirements. They do not give any answer on questions about: a) identifying all of the use cases for the system; b) conflicts among use cases; c) gaps that can be left in system requirements; d) how changes can affect behavior that other use cases describe.

In order to avoid these limitations, approaches that apply use cases until now are being improved. For example, it is interesting to consider how use cases are applied in *the Unified*

Process (UP) requirements workflow [7]. Within the UP, use cases are applied as the base of construction of different viewpoints on the system, because they are dedicated to the main requirements capturing. Critical points of the UP are the following:

- The UP and, therefore, use cases are requirements driven. The UP does not decline some business model using, but does not specify a formal way from the business to the application. The business model can serve as an accessory in identifying use cases and actors.
- Actors and use cases are identified using a *tentative* idea about what the system boundary is.
- The requirement traceability to use cases is defined *ad hoc* using only developer's intuition and experience in this field.
- Inclusion and extension use case identification is monotonous work that requires careful investigation of all specified use cases. Although, it needs to be noticed, these relationships are advanced and can be not used.

Business Object Oriented Modeling with use cases (B.O.O.M.) developed by Howard Podeswa [84j is dedicated to relate business analyst documentation to the object oriented software development. Within this approach, *business use cases* specify business requirements to the project, i.e. interaction with a business system (or a problem domain), and system use cases specify system requirements, i.e. interaction with a planned application. Summarizing activities in the initiation phase of the B.O.O.M., the following weaknesses can be defined:

- This approach excludes other requirements gathering techniques except use cases;
- Business use case identification is IT project driven not business driven. Although business use cases are checked for conformance to the existing business processes;
- Use case package creation in accordance with logical relation among system use cases is an intuitive and *ad hoc* approach;
- If packages of system use cases are created in separation from business use cases, then changes in business processes cannot be traceable to the system use cases in a natural way.

Alistair Cockburn suggested an approach for *structuring use cases with goals,* within which four use case dimensions are defined: purpose (stories, requirements), contents (contradicting, consistent prose, and formal content), plurality (one or multiple) and structure (unstructured, semi-formal, and formal structure) [42J. As this approach suggests use case structuring with goals, within it use cases can be located at the *system scope,* or at the *strategic scope.* A. Cockburn suggests locating use cases in hierarchies, where strategic use cases and summary goals are at the highest level. Then, summary goals are decomposed to strategic-level user goals and system-level summary goals that both reference system-level user goals, which in turn reference system-level subfunctions.

The benefits of such structuring are the possibility to attach non-functional requirements to goals, to manage a project by goals, to recognize subtle requirements earlier apart from goal failures, and to facilitate work with requirements specifications.

However, a few problems also exist in this approach. There is some confusion about the levels, goals, and data variations.

The fifth section summarizes hardships in the problem domain modeling [28], [70J, [73], [79], and formulates the motivation of the research:

- The MDA suggests specifying system requirements, a system and an environment the system will work within in the CIM. This model can serve as an input of

transformation to a PIM. However, the PIM relation to the CIM is assumed to be performed by *ad hoc* approaches.

- Use cases that are widely used for problem domain and requirements specification describe the problem domain as a "black box"; moreover, the priority of problem domain modeling is very low. Thus, system functioning and its structure are based on intuitive understanding of the environment the system will work within.
- Use cases that are not supported by the proper problem domain analysis can cause the poor quality of the system; therefore, until now use cases relate to the narrow area, where the real world interacts directly with the system *{the application)*, and, hence, focuses requirement analyst's attention on events that happen within *the application* boundaries, but the properties of the surrounding real world can remain underestimated. Consequently, problem domain modeling and understanding should be the primary stage in the software development.

This means that use cases must be applied as a part of a technique, whose first activity is well-defined *problem domain model* construction.

The sixth section summarizes and concludes the contents of the first part.

The second part (Formal Approaches for Problem Domain Modeling) discusses formal problem domain modeling approaches and researches formalization capabilities of them in order to satisfy goals of the doctoral research.

Bran Selic enumerated five key characteristics that a useful and effective engineering model needs to satisfy [88]; abstraction, understandability, accuracy, predictability, and inexpensiveness.

This part discusses topological functioning modeling, Sowa's conceptual graphs, and universal categorical logic. These approaches satisfy those characteristics. Petri nets are not considered, because they are less understandable for structure representation, and this research is not related to simulation.

The first section gives a brief look into the formal logic usage for knowledge representation. The question about a formal description of semantics still exists especially in the context of visual schemes. Besides that, it is necessary to distinguish string-based and graph-based logic [25].

Generally, if some diagram *D* exists, then it has some sense *M(D),* This sense must be described in precise (ideally- mathematical) statements. This description makes some precise specification S_D that has the precise semantic $M(S_D)$. Therefore, $M(S_D)$ is abstraction of formal intuitive sense *M(D)* and S_D may be considered as some (internal) logic specification, that is hidden in the diagram D. Then this means that when one starts to think about *M* formal description, this huge amount is being narrowed down to several domains of mathematical constructs such as e.g. set theory, type theory, and high-order predicate logic or category theory, and so on. Therefore, any formal semantic (some specification S_D) can be specified in any formal language thereof [25].

In mathematics and computer science, one of the more common approaches to formalization is using the so-called predicate languages - first-order predicate logic and higher-order predicate logic [101]. For example, the formal basis for Z language is first-order predicate logic extended with type set theory [18], [50], [56]. However, there is a lack of representative power in such logic. The problem with using mathematics alone is that large specifications very quickly become unmanageable and unreadable.

Semantic nets and frames have been developed in order to solve this problem [89]. These representations have their advantages and limitations. The main advantages are direct problem domain knowledge representation, a support of default reasoning (i.e.inference using

"is-a" and "a-kind-of" links), efficient, and a support of procedural knowledge. The main disadvantages are lack of semantics (e.g. the usage of the "is-a" links is ambiguous, because it allows different interpretations), and limitations in semantic expressions [21], [55J, [85], [89J. Graph theory is widely used in mathematics and computer science. The main research object is studies of properties of graphs. Many applications of graph theory exist in the field of network analysis as well as in the field of problems related to the topology studying [104].

The second section discusses Sowa's Conceptual Graphs (Sowa's CGs) that were derived from Pierce's *existential graphs* by John Sowa in 1984. A working draft for ANSI Standard for Conceptual Graphs is being developed now [93], [94], [95], [96].

A conceptual graph (CG) is visualization (the way of representation) of logic by means of the abstract syntax. The display form of a conceptual graph that describes a sentence "Balvis thinks that Laima wants to register in the library" is illustrated in Figure 2.

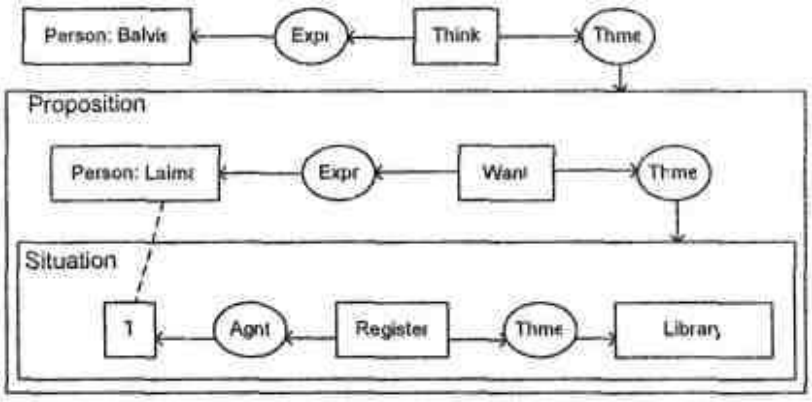


Figure 2. CG illustrates that "Balvis thinks that Laima wants to register in the library"

A linear form of this conceptual graph is represented as follows:

[Person: Balvis]←(Expr)←[Think]→(KThme)-

[Proposition: [Person: Laima *x]←(Expr)← [Want]→(Thme)-

[Situation: [?x]←(Agnt)←[Register]→(Thme)→[Library]]].

The abstract syntax of the Sowa's CGs is described in the thesis. Their main constructs are the following:

- Concepts that are represented in boxes have a type and a referent (which allows defining a particular exemplar);
- Conceptual relations that are represented in ovals have a type and a valence;
- A context is a concept with a nested conceptual graph that describes the referent; for example, Figure 2 shows two contexts, i.e. *Proposition* and *Situation.*

Concepts and conceptual relations are organized in hierarchies that begin from two primitive labels - *Entity* (the universal type) and *Absurdity* (the absurd type).

Conceptual graphs support canonical formation rules: specialization, generalization, and equivalence rules, and inference rules.

The main advantage of Sowa's conceptual graphs is that they can serve as an intermediate level for human language specification in the formal way. They are suitable for knowledge keeping, reasoning, and computation in both conceptual graph and predicate calculus forms.

The third section discusses Topological Functioning Modeling (TFM) that was developed at Riga Technical University by Janis Osis in 1969 [72], [81]. Janis Osis, Zigurds Markovitch, Janis Grundspenkis and other scientists applied this approach in medicine

problem solving, knowledge acquisition, embedded system development, model driven architecture, and other problem solving [5], [6], [8], [9], 110], [11], [12], [13], [14], [24], [36], [37], [38], [39], [40], [51], [52J, [53], [54], [59], [73], [74], [75], [77], [78], [79], [80], [82], [83].

The TFM has the rigor mathematical base. The main idea of the TFM is that functionality of a complex or large system can be represented as topological space *(X,Θ)*, where *X* is a finite set of functional features or properties of the system under discuss, and *Θ* is topology represented in the form of a directed graph [81].

An abstract topological functioning model of the system may be represented in the form of a directed graph *G(X, U)*, where *X* is is a finite closed set of elements with some certain topology *Θ* among them, and *U* is a set of arcs that illustrates the topology. The topology in the set *X* is any system *Θ* of open sets *A* of the set *X*. The system *Θ* has to satisfy two Kolmogorov's axioms (1) and (2).

$$X \in \Theta; \quad \emptyset \in \Theta, \tag{1}$$

$$\forall \eta \left(\bigcup_{\eta} A_{\eta} \in \Theta \right); \forall \varphi \bigcup_{\varphi=1}^{\kappa} A_{\varphi} \in \Theta \tag{2}$$

The main condition on the topological model construction is a complete exhaustive verbal informal description of the system functioning. In order to separate a topological functioning model from this, the following must be done:

- First, physical, business or biological properties or features from the system description that constitute the system Z by the expression (3) must be defined. These properties must be significant for regular functioning of the system.

$$Z = N \cup M \tag{3}$$

  Where Z - system functional properties;

  N - a set of inner properties or functional features of the system;

  M - a set of properties or functional features of other systems, constituting the environment affecting the system or that is affected by the system;

- Set topology *Θ* in the form of a graph or a matrix that indicates cause-and-effect relations between physical or biological properties of the system. It is assumed that a cause-and-effect relation exists between two properties if display of one of them causes display of another without participation of any intermediate property. Therefore, adequacy of the abstract topological model of functioning of a concrete system is reached by this that the substantial sense of investigated system is appropriated to abstract mathematical objects.
- The topological functioning model is separated with the closuring operation that is illustrated by the expression (4).

$$X = [N] = \bigcup_{\eta=1}^{n} X_{\eta} \tag{4}$$

  Where X_{η} - an adherence point of the set *N;*

  N - a set of inner properties or functional features of the system;

  n - the number of adherence points of *N*, i.e. capacity of *X*.

A topological functioning model has topological (connectedness, closure, neighborhood, and continuous mapping) and functional properties (cause-effect relations, cycle structure, inputs and outputs) [79]. This section describes statements and corresponding

formalizing corollaries. Additionally, an example of the topological functioning model is discussed in this section.

The fourth section discusses properties of the universal categorical logic that is based on category theory and principles of arrow thinking [6], [16], [22], [43]. This discussion is based mainly on Zinovy Diskin's and Boris Kadish's works in this field [25], [29], [30], [31], [32], [33].

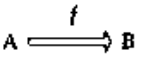
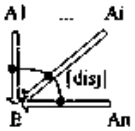
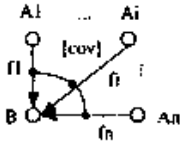
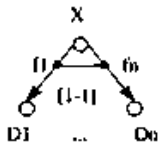
A specification in category logic is a graph, the so-called sketch. The key idea is that any problem domain can be specified as a collection of objects (vertices) and their morphisms (arrows) that are closed in composition. The internal structure of the objects as well as relations among the objects is represented by arrow diagram predicates.

In semantic modeling, the universal categorical logic is especially flexible. According to the context, objects and arrows can be interpreted as follows:

- Sets and functions (in data modeling),
- Object classes and associations (in OOA&D),
- Data types and procedures (in functional programming),
- Theorems and proves (in logic/logical programming),
- Interfaces and processes (in process modeling),
- States and transitions (in transaction modeling),
- Specifications and its mappings (in metamodeling).

Table 1

Arrow-diagram predicates for a system of sets and functions

Predicate name	Arity shape with visualization	Denotation semantics
Set Inclusion is the source set is a subset of the target set and mapping *f* is their inclusion.		$A \subset B$ and $f(a) = a$ for all $a \in A$
Disjointness is an element of the target set may be an element *of the only one* subset.		$\bigcup_{i=1}^{n} Ai \subset B$ and $\bigcap_{i=1}^{n} Ai = \varnothing$
Covering is each element of the target set is a value *of at least one* of the mapping f_i.		$(\forall b \in B)(\exists i \le n)b \in fi(Ai)$
Separating family of functions precisely expresses the internal tuple-structure of X-objects by declaring the corresponding property for some arrow diagram adjoining to the node X.		For any $x' \in X$, $x \neq x'$ implies $f_i(x) \neq f_i(x')$ for some i Note, however, that in such a case the tuple-function $f = \langle f_1 \ldots f_n \rangle$ into the Cartesian product of D_i $f = (f_1 \ldots f_n): X \to D_1 \times \ldots \times D_n$, $fx = (f_1 x, \ldots, f_n x)$ is injective (one-one) so that elements of X can be considered as unique names for tuples from a certain subset of $D_1 \times \ldots \times D_n$, i.e. the image of *f*.

By the definition, arrow logic is graph-based logic. In other words, a predicate argument consists of vertex placeholders and arrow placeholders that are organized in an

oriented graph construct. Some of the predicates mentioned in the doctoral thesis for a system of sets and functions are illustrated in Table 1 [25J. In the thesis, the discussed predicates are explained by examples.

The main idea is that any property represented in a diagram that has some meaning can be assigned to a predefined configuration of vertices and arrows, i.e. to an arrow predicate. The couple *(<predicate marker>, <predicate structure marked with names>)*, i.e. *(P, d)* is the arrow predicate notation.

A composition of arrow predicates constitutes a sketch. A sketch example is illustrated in Figure 3. This illustrates three object classes *Reader*, *BirthDate*, and *Order* with attributes in the arrow form, and types of the attributes *[String]*, *[Integer]* and *[{F.M}]*.

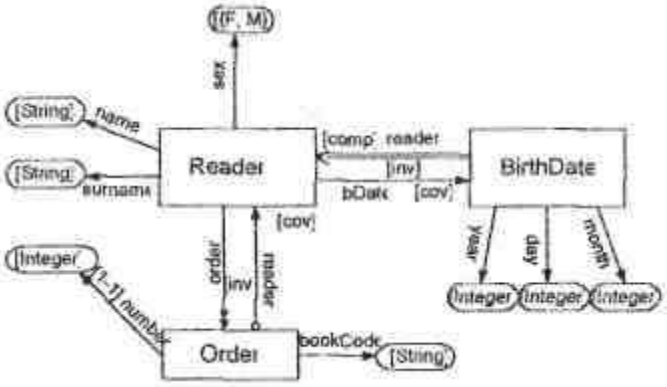


Figure 3. Sketch example

The universal categorical logic supports basic modularization principles, and sketch abstraction and refinements by functor natural transformation.

The fifth section researches formalization capabilities of the approaches discussed in the previous sections.

The topological functioning modeling and the universal categorical logic has much in common:

- Both of the approaches have the graph-based structure, wherein vertices can represent any object according to the context from the abstract aspect;
- Both of the approaches satisfy the connectedness requirement;
- Both of the approaches support refinement and abstraction in accordance with the continuous mapping rules;
- Due to the continuous mapping, lack of data that sometimes occurs can be filled up by data that are obtained, when models of the same type systems are being continuously mapped into the model of the system under study;
- An adequacy of a topological mode! and a sketch is achieved by assigning a proper meaning of the modeled system to the defined mathematical constructs;
- Both of the approaches enable the research of similarities and differences of the same type systems;

These approaches have also differences:

- For problem domain property modeling, the topological functioning model uses a notation that consists of a vertex and an arc; in turn, the universal categorical logic uses a predicate as a minimal logic unit;
- There is no clear formalism for the system model obtaining from environment (problem domain) space and for formal obtaining of subsystems in the categorical logic, but the topological functioning modeling provides such mechanisms based on mathematics;

- The universal categorical logic represents system functioning by means of (n>l)-arrow graph constructs, which are difficult to comprehend.

The main conclusion is that the topological functioning modeling is more suitable at the very beginning of analysis, because it does not only provide formal definitions of the system and the subsystem, but it also enables representing and exploring of the problem domain in the mathematically precise and understandable way.

The topological functioning modeling has been compared with Sowa's conceptual graphs, too:

- In contrast of the TFM, Sowa's CGs do not support the graph logic on the whole -these graph-like constructs are rather graphic representation of predicate logic;
- CGs do not support the continuous mapping rule, they support the "nested graph" conception that depends on some context and that can be either true or false;
- The notation size of Sowa's CGs is larger, thus, representation constructs are larger too;
- Conceptual graphs represent not only relations and concepts, but also the level of statement trustworthiness;
- The cyclic structure, as distinct from the topological functioning model, is being represented in the "unnatural" way, because the conceptual graphs need a separate concept for representation of the place that was previously visited, i.e. the new instance must be created;
- Logical relations are represented in the nested contexts - this enlarges specifications and makes it hard to understand.

The main conclusion is that in order to satisfy the aims of the doctoral research Sowa's CGs and the universal categorical logic provide unnecessary detailing and complexity, but do not provide a formal mathematical definition of such an important construct as "system" and "subsystem".

The sixth section gives a brief summary of the second part results and a reason of Topological Functioning Modeling for Model Driven Architecture (TFMfMDA) development.

The third part (Attacking Complications in Problem Domain Modeling) suggests the developed approach TFMfMDA, which satisfies the main aim of the doctoral research, and its application example.

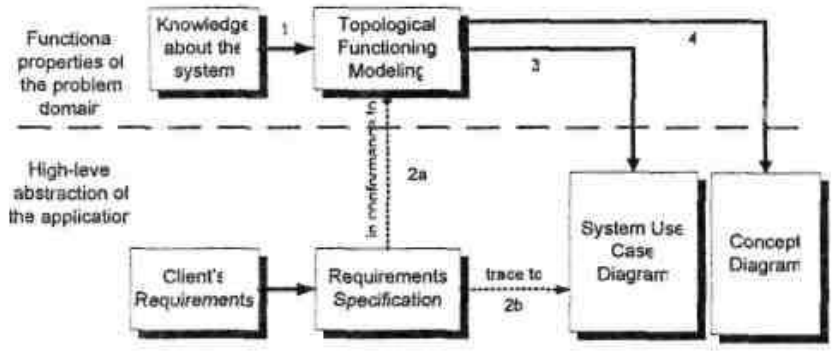


Figure 4. A place of the topological functioning model in problem domain modeling

The key idea of the approach is as follows (Figure 4). Having knowledge about the system that runs in the real world, the topological functioning model of this system can be composed using the refined method of the TFM composition (the arrow 1). Then client's

requirements can be conformed to it, i.e. can be mapped onto the topological model. As a result, functional requirements are defined more exact and validated in conformance to the existing problem domain functionality. Additionally, the model can be enhanced with new functionality defined in the requirements (the arrow 2a). Subsequently, system use cases can be defined in conformance to the functional requirements using a goal-based method (the arrow 3), and a concept diagram can be defined using model transformation (the arrow 4). Functional requirements can be traced to the system use cases through the formal basis, i.e. functional properties of the topological functioning model (the arrow 2b) [11].

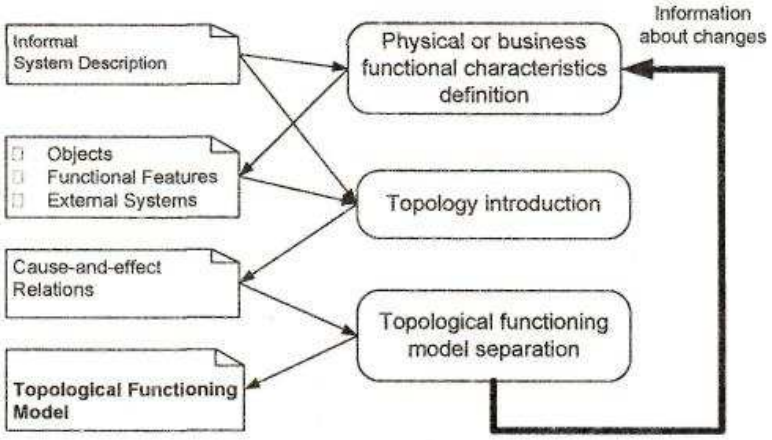


Figure 5. The formal method of construction of the topological functioning model

The first section discusses the developed method of the topological functioning model construction for problem domain modeling in business system context (Figure 4, the arrow 1). Its steps are illustrated in Figure 5 and are the following [10], [11], [14]:

- *Definition of physical or business functional characteristics* consists of:
  - o Definition of objects and their properties from the problem domain description, that is performed by noun analysis, i.e. by establishing as meaningful nouns and their direct objects as handling synonyms and homonyms;
  - o Identification of external systems (objects that are not subordinated to the system rules) and partially-dependent systems (objects that are partially subordinated to the system rules, e.g. system workers' roles);
  - o Definition of functional features is preformed by verb analysis in the problem domain description, i.e. by founding meaningful verbs. Each functional feature represents an object action, a result of the object, and an object that receives the result or that is used in this action (for example, a role, a time period, a catalog, etc.) as well as preconditions or atomic business rules, and an entity responsible for execution. Each precondition and atomic business rule must be either defined as a functional feature or assigned to the already defined functional feature. In the thesis, two expression forms are defined:
    - The more detailed form
      <action>-ing the < result > [to, into, in, by, of from] a(n) <object>
    - The more abstract form
      <action> –ing a(n) <object>
- *Introduction of topology Θ* is cause and effect relation between functional features identification. Cause-and-effect relations are represented as arcs of a digraph that

are oriented from a cause vertex to an effect vertex. The main properties of cause-and-effect relations are the following: a) a cause chronologically precedes an effect; b) a cause can be sufficient or necessary (complete or partial, correspondingly); it is assumed that there are necessary causes in the topological functioning model, because risks of the system functioning can be unknown during the analysis; c) a cause not only precedes an effect and always is followed by it, it causes and is condition on an effect; d) the causality is universal, i.e. it exists in any problem domain even if it is not evident for a human. A structure of cause-and-effect relations can form a causal chain, wherein each relation is important. Some advice is considered for cause-and-effect relation identification.

- *Separation of the topological functioning model* is the same as in the TFM approach, i.e. it is performed by applying the closure operation over a set of system inner functional features [81].

The topological functioning model of a library constructed in the **application example** is illustrated in Figure 6, and corresponding functional features are defined in Table 2. The main functional cycle **„11-12-13-14-15-17-18-19-23-24-11"** is represented by bold arrows.

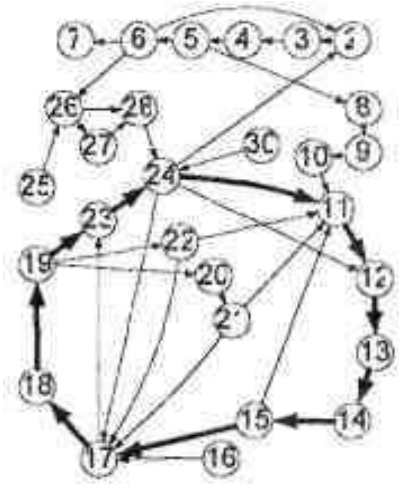


Figure 6. The topological functioning model of the library

Table 2

Functional features and their preconditions

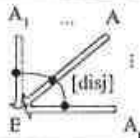
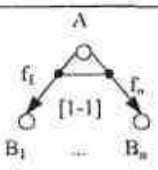
Nr.	Functional Feature	Preconditions	Respons.	External
2	Finding out the name, the surname, the age of a person		Registrar	no
3	Creating a reader card		Registrar	no
4	Filling in a reader card		Registrar	no
5	Assigning the code to a reader card		Registrar	no
6	Filling in a library ticket		Registrar	no
7	Giving out a library ticket		Registrar	no
8	Consulting a catalogue		Reader	no
9	Completing a request form		Reader	no
10	Giving the request form to a librarian		Reader	no
11	Taking the request form from a reader		Librarian	no
12	Checking the availability of a copy		Librarian	no
13	Calculating the book amount of a reader	[if the copy is available]	Librarian	no
14	Checking the book limit of a reader		Librarian	no
15	Checking out the copy to a reader	[if the book amount is smaller than the book limit]	Librarian	no

Nr.	Functional Feature	Preconditions	Res pons.	External
16	Returning the copy by a reader		Reader	no
17	Checking the condition of a copy		Librarian	no
18	Calculating the fine to a reader		Librarian	no
19	Imposing the fine to a reader		Librarian	no
20	Completing the restoration form to a copy	[if the copy needs restoration]	Librarian	no
21	Sending the copy to a Restorer		Librarian	no
22	Removing a copy	[if the copy cannot be restored]	Librarian	no
23	Checking in a copy	[If the copy has good condition]	Librarian	no
24	Ensuring the availability of a copy		Librarian	no
25	Purchasing a copy		Registrar	no
26	Checking the entry in a catalogue		Registrar	no
27	Adding the entry to a catalogue	[If such an entry does not exist)	Registrar	no
28	Assigning the code to a copy		Registrar	no
30	Returning a restored copy		Restorer	no

The second section considers functional requirements mapping onto the topological functioning model with the TFMfMDA (Figure 4 the arrow 2a). Such mapping is possible, because functional features specify functionality that *exists in the problem domain*, and functional requirements specify functionality that *must exist in the application.* Within the TFMfMDA, five types of mappings and corresponding arrow predicates (for visual representation) are defined (Table 3) [10], [11].

Table 3

Arrow-diagram predicates for mapping representation

Mapping	Arrow-diagram predicate	Description
One to One		***Inclusion predicate*** is used if the functional requirement *A* completely specifies what will be implemented in accordance with the functional feature *B*
Many to One		***Covering predicate*** is used if functional requirements $A_1, A_2, \ldots, A_n$ overlap the specification of what will be implemented in accordance with the functional feature *B*
		***Disjoint (component) predicate*** is used if functional requirements $A_1, A_2, \ldots, A_n$ together completely specify the functional feature *B* and do not overlap each other
One to Many		***Projection*** is used if some part of the functional requirement *A* incompletely specifies some functional feature B_i
		***Separating family of functions*** is used if one functional requirement *A* completely specifies several functional features $B_1, \ldots, B_n$

Five types of mappings are the following:

- *One to One.* One functional requirement completely specifies one functional feature;
- *Many to One.* Several functional requirements specify one functional feature, because: a) these are covering requirements, or b) it is a simplified (abstracted) functional feature;
- *One to Many.* One functional requirement specifies several functional features, because: a) the functional requirement joins several requirements and can be splitted up; b) functional features are more detailed than the requirement;
- *One to Zero.* One functional requirement specifies some new or undefined functionality. In this particular case it is necessary to define possible changes in the problem domain functioning.
- *Zero to One.* Requirements specification does not contain any functional requirement corresponding to the defined functional feature. This means that this problem domain functionality will not be implemented in the application.

Additionally, it must be pointed out that mappings can be represented also in a matrix form, but in this case mapping particularities will not be specified.

Within the application example, the conformity between the topological functioning model and the functional requirements (Table 4) is defined as illustrated in Figure 7(b). The obtained topological functioning model is illustrated in Figure 7(a).

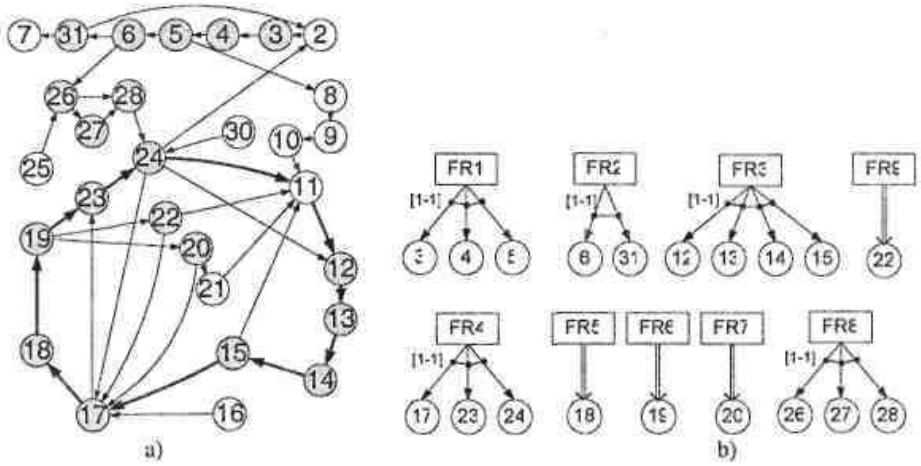


Figure 7. The obtained topological functioning model (a) and functional requirements mappings (b)

Table 4

Functional requirements for the application example

Label	**Functional requirement**
FR1	The system shall perform new reader registration, which includes reader card creation.
FR2	The system shall perform library ticket printing on the letterhead.
FR3	The system shall execute book copy giving out to a reader.
FR4	The system shall execute book copy giving back by the reader, including book condition checking.
FR5	The system shall calculate a fine.
FR6	The system shall impose a fine to the reader.
FR7	The system shall perform recording of sending a book to restoration.
FR8	The system shall perform an entry adding to a catalogue of copies.
FR9	The system shall perform a copy removal from the library

The third section describes transition from the initial computation independent model to the output model, i.e. a developed method of use case model obtaining within the TFMfMDA (Figure 4 the arrow 3). This method includes the following steps [10], [11]:

- *Identification of business system users and their goals.* Business system users can be actors and workers:
  - Business system actors are *external* entities that interact directly with the business system and establish *business goals* within the business system. In the topological functioning model, actors are represented as external system functionality or functional properties of the system under consideration that interact with external systems (in this case, their identification is necessary). Actors are external companies, clients, etc.
  - Business workers are system *inner* entities that interact directly with the business system either establishing *system goals* or implementing those goals. Workers are humans, roles, etc.
  - Identification of business system users' direct goals is related to the identification of the corresponding set of functional features that are necessary for the goal satisfaction. For each goal, an input functional feature (input transaction), an output functional feature (output transaction), and a functional feature chain between them can be defined.

  As business workers, the business actors can be users of the application. System (application) goal identification helps for additional requirements validation, i.e. for "missing" requirements discovering. A goal as a means for use case identification has been chosen because a goal can be achieved performing some process that can be long running. The type gap cannot do it.
- *Identification and refinement of system use cases.* Functional features that are specified by functional requirements and that are needed to achieve a business goal describe a system use case. A business system user that established this goal is an (UML) actor that communicates with this use case. This principle enables formal identification of a use case model from the topological functioning model. However, this principle provides also additional possibilities for system use case refinement:
  - *An inclusion use case* [87] is some common sequence for several use cases. In the topological functioning model, it is an intersection of functional feature sets of several system goal areas. The common functional features must be analyzed, because can be a situation, when a common functional feature is located in an alternative workflow of the use case, i.e. in a sub-cycle or a branch in the topological functioning model;
  - *An extension use case* [87] shows an alternative way of the scenarios execution. In the topological functioning model, it is a sub-cycle or a branch, existing within the system goal. The point of branch beginning is the extending point;
  - Identified use cases can be represented in a UML activity diagram by transforming functional properties into diagram activities, and cause-and-effect relations into control flows.
- *Use case prioritizing* can be done in accordance with client's requirements or can be defined using some requirements attribute systems, e.g. MoSCoW or GRASP [7]. Within the developed approach, use case implementation priorities are defined as follows (in accordance with Rational Unified Process):
  - *Critical (must be implemented otherwise the application will not be acceptable)* - if a use case will implement any functional feature that belongs to the main functional cycle;

- *Important (it would significantly affect the usability of the application)* - if a use case will implement any functional feature that is a cause or an effect of a functional feature that belongs to the main cycle;

- *Useful (it has no significant impact on the acceptability of the application)* - if **a** use case will not implement any main cycle functional feature or a functional feature that affects or is affected by some functional feature that belongs to the main cycle.

Within the application example, a process of obtaining use case model, specification of use case flows in a UML activity diagram, and functional requirements traceability to use cases (Figure 4 the arrow 2b) are discussed. The obtained use case model is illustrated in Figure 8.

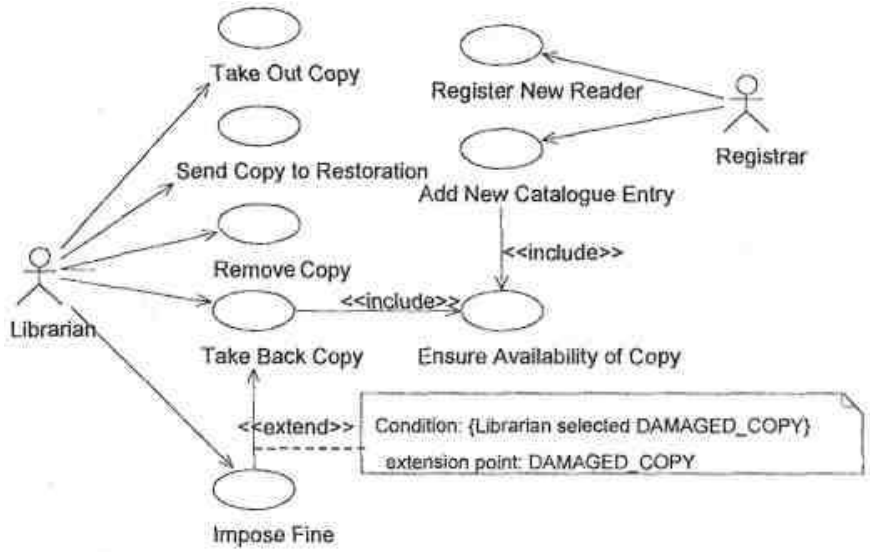


Figure 8. The obtained use case mode!

The fourth section considers identification of a conceptual diagram and control classes within the TFMfMDA (Figure 4 the arrow 4). A concept or conceptual class is an idea, thing, or object [48]. After functional requirements mapping, the topological functioning model represents also functionality that must be implemented in the application, besides that, it includes all concepts that are necessary for proper functioning.

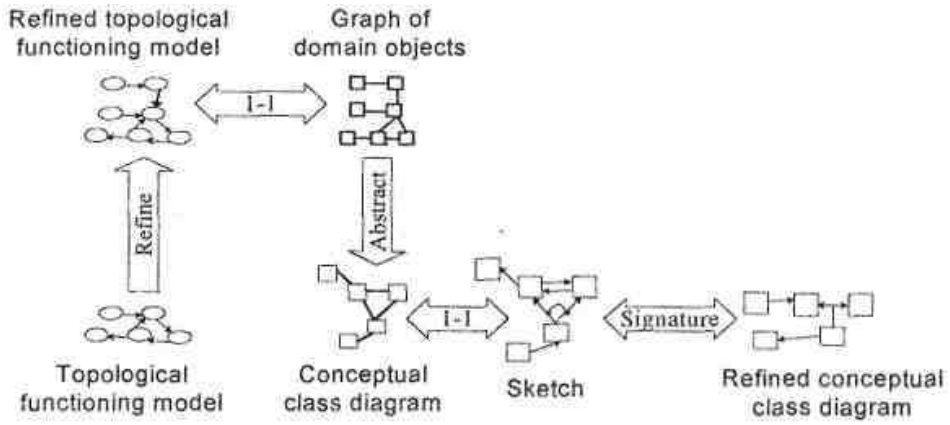


Figure 9. Conceptual class diagram obtaining

In order to obtain a conceptual class diagram, it is necessary to detail each functional feature of the topological functioning model to the level when it uses only one type objects. After that, this more precise model must be transformed one-to-one into a problem domain object graph, and then vertices with the same type of objects must be merged keeping all relations with other graph vertices. Hence, a conceptual class graph with indirect associations is defined [9], [12], [73]. In order to make these relations more precise, the graph can be transformed into a sketch, refined, and represented as a refined conceptual class diagram (Figure 9).

Additionally, two corollaries of this detailing are defined in this section. These corollaries indicate possible inheritance relations among types, and common operations, which can further be transformed into use case interfaces.

Besides that, control objects can be identified using TFMfMDA properties. The control objects represent a layer between a user interface layer and class objects [7J, [48]. If a user initiates some functional property, then actions of it initiated by a user can be assigned to a use case controller.

Within the application example, the conceptual class diagram obtainment by transformation of the topological functioning model is considered (Figure 10), and use case controllers are identified.

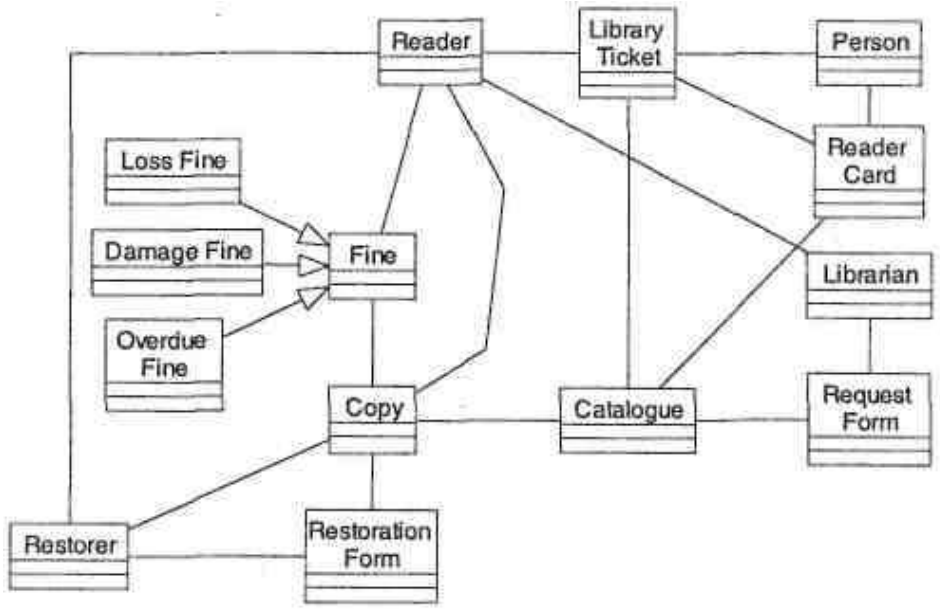


Figure 10. The conceptual class diagram

The fifth section discusses an example of universal arrow logic application for scheme integration [8], [25], [26], [31], [32].

The sixth section describes the TFMfMDA conformity to the MDA foundation model, i.e. approach concepts are represented in a metamodel in terms of the MOF and UML profile constructs. The TFMfMDA metamodel is illustrated in Figure 11.

The metamodel is described at the MOF metalevel M2, and represents the topological functioning model as an instance of the metaclass *TFMTopologicalFunctioningModel* that includes at least two functional features of the metaclass *TFMFunctionalFeature.* They can be united in functional feature sets (the metaclass *TFMFunctionalFeatureSet).* This means that a functional feature represented in a TFM can visualize a functional feature set. One functional feature can contain only one set and one functional feature can belong only to the one set. A

functional feature can be subordinated to a business system itself or to an external system (the type *Subordination).* Functional features can form functioning cycles (the metaclass *TFMCycle)* of different order. Only one cycle can be the main one. Cause-and-effect relations connect functional features. A cause functional feature must have at least one effect. An effect functional feature must have at least one cause. Functional features are related to functional requirements (the metaclass *TFMFunctionaiRequirement)* via some correspondence (the metaclass *TFMCorrespondance).* The correspondence generally is many-to-many. The correspondence can be complete or incomplete, overlapping or disjoint. The functional features can be associated with several goals (the metaclass *TFMUserGoal)* that are established by direct users (the metaclass *TFMBusinessUser)* of the business system. The users can be external entities that interact with the business system (the metaclass *TFMBusinessActor)* or workers that interact within the business system (the metaclass *TFMBusinessWorker).* A user goal can be specialized to a business goal (the metaclass *TFMUserBusinessGoal)* and to a system one (the metaclass *TFMUserSystemGoal).* The latter includes functional features to be implemented. A user goal and, thus, corresponded functional requirements are associated with the functioning cycle, whose order establishes their benefit value (the type *Benefit).*

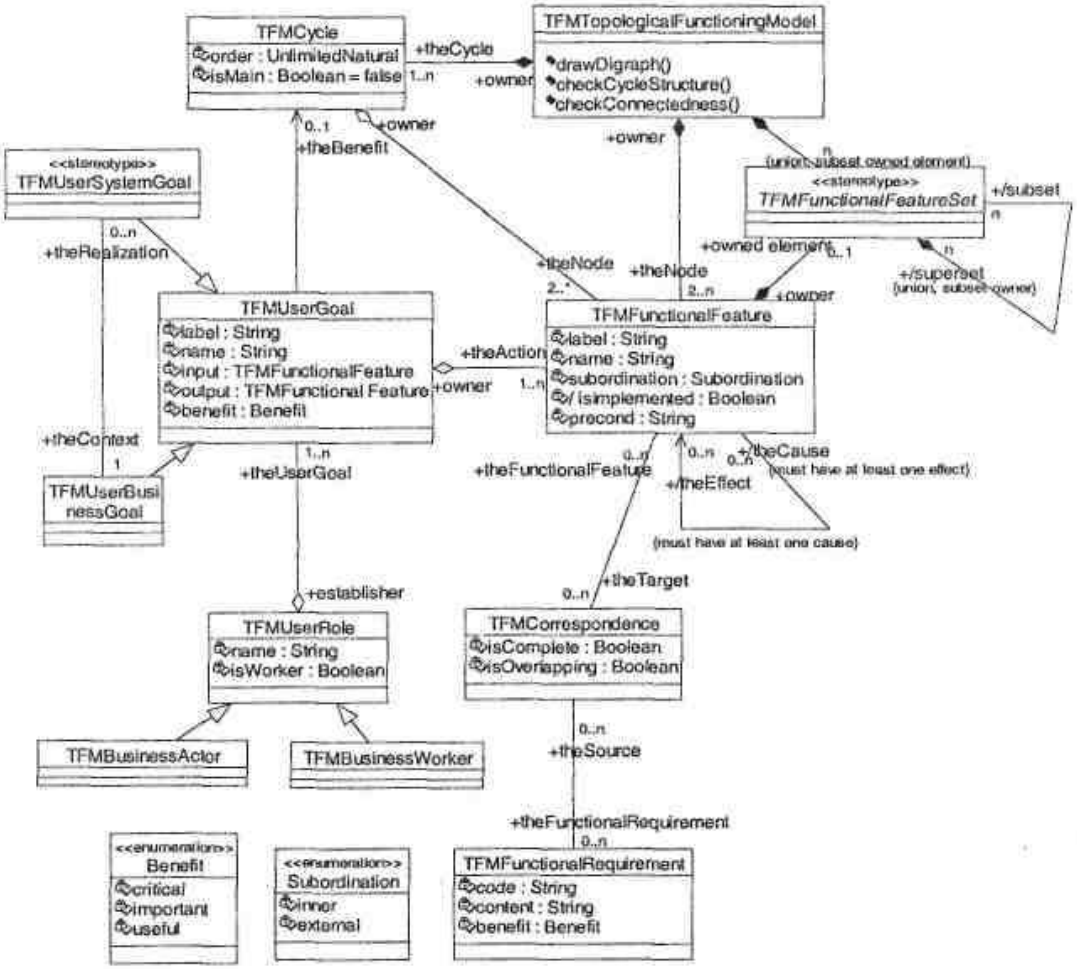


Figure 11. The TFMfMDA metamodel defined in terms of the MOF

The seventh section gives a brief summary of the third part, and conclusions about application of the developed TFMfMDA.

The fourth part (TFMfMDA Evaluation) discusses properties of the Topological Functioning Modeling for MDA for the problem domain modeling and from the use case formalization aspect.

The first section uses a non-empirical research technique, i.e. feature comparison of properties of the TFMfMDA, UP requirements workflow, B.O.O.M. initiation phase and Cockburn's approach. It is established that only the TFMfMDA and B.O.O.M require obligatory problem domain analysis, and only the TFMfMDA provides the formal foundations for such analysis. In turn, Cockburn's approach does not define modeling of the initial static structure of the problem domain and the application.

Within each approach, excluding the developed one during the doctoral research, the initial boundaries of a problem domain are defined basing on an intuitive estimation, or on expert decisions. Within the developed approach, the problem domain boundaries are defined using a mathematical mechanism, i.e. the closuring operation.

The developed approach suggests mathematical identification of the common functionality, but other approaches suggests such identification during the review process of use case specifications, or by definition of common subfunctions as Cockburn's approach does.

Table 5

Existence of problems with use cases in the considered approaches

Problems	TFMfMDA	The UP with UML	B.O.O.M.	Cockburn's approach
Information Capturing				
Information about user interface	no	yes	no	no
Different levels of details	no	yes	no	yes
Limitation of Thinking				
Requirements gathering limited with creation of a list of use cases	no	no	yes	no
Exclude other requirements gathering techniques	no	no	yes	no
Exclude proper domain "as is" analysis	no	no	no	yes
Completeness Checking				
All of the use cases for the system not identified	no	yes	yes	yes
Conflicts among use cases	no	yes	no	no
Gaps that can be left in the system	no	yes	no	yes
Change affect on behavior that other use cases describe not clear	no	yes	yes	yes

Only the TFMfMDA provides functional requirements represented in the textual form traceability to use cases via the formal base - the topological functioning model.

Use case disadvantages are solved or still exist in the approaches as it is illustrated in Table 5.

The second section discusses empirical approach application, i.e. a case study, and application of a metric-based approach.

Since a use case count cannot be a measurement [84], a case study of library functioning is developed. This problem domain includes more roles and functionality than the previously considered example. Eighteen use cases and four actors were defined from the problem domain (from 82 functional features) using the TFMfMDA. Then, according to the

functional features of the topological functional model of the library, use case specifications in consistent prose were developed. Use cases were evaluated using a metric-based approach- a Metric-Based Reading technique [17]. This metric evaluation helped to conclude the following:

- The TFMfMDA reduces a hidden incomplete use case number, because all incomplete use cases are defined directly either as inclusion or as extension use cases;
- Manual creation of textual specifications enables ambiguity introducing in use case descriptions.

The metric-based inspection gave the attention to questions that were not considered during the research. These questions are related to the understanding the use case specifications in case of multi-step specifications, and to the balance between actor and system actions in specifications. It should be noticed that these questions depend a lot on the defined problem domain functioning, and a universal answer is hard to be found.

The third section gives a brief summary of the performed tasks and their results in the fourth part.

THE MAIN RESULTS OF THE RESEARCH

The main purpose of this work was to research the possibility of formalization of the problem domain modeling within the Model Driven Architecture and the development of an approach that realizes research findings.

The main result of the doctoral research *is the developed and demonstrated in two case studies new problem domain modeling approach TFMfMDA (Topological Functioning Modeling for Model Driven Architecture).*

The more significant results of the doctoral research are the following:

1. A method that formally describes and analyzes a problem domain from its informal description is developed.

   The method foundations are based on formal logic, i.e. on principles of the topological functioning modeling. Functioning of a problem domain is formally holistically understandably described in the topological functioning model. Functional and topological properties of that model make the formal basis of the composed model conformity to the problem domain.

2. A method that helps to define system functional requirement conformity to the defined functionality in the problem domain is developed.

   The method defines possible mappings of system functional requirements (in the text form) to the functionality specified in a topological model. A formal description of the mappings is based on the universal categorical logic. System functional requirements are validated and refined within this method.

3. A method for use case determination from the problem domain representation in the topological functioning model is developed.

   Problem domain functionality is precisely transformed into a use case model by user goals in accordance with client's requirements, and existing constraints on the functionality.

4. A method for determination of conceptual classes, their relations, and control classes from the problem domain representation in the topological functioning model is developed.

The method applies graph transformation rules. System structure is based on the system functionality not vice versa.

5. The TFMfMDA is coordinated with the MDA ideas.
   Each concept of the TFMfMDA is described in the developed *MOF-based standalone metamodel* as well as in the first version of the developed *UML profile.*

6. The developed TFMfMDA approach was approbated in the case study of library functioning, and a positive result has been obtained - the composed use case model corresponds to the user functional requirements, and is in conformance to the problem domain.
   Advantages of the TFMfMDA application (ordered by importance) are the following:
   - It changes a problem domain description by use cases from the "black box" to the "white box";
   - The careful cycle analysis helps in identifying all cause-and-affect relations in the complex business systems;
   - It enables clarifying what the client really needs at the very beginning of the analysis. It enables defining which business goals achievement will be automated at first, and what inputs and outputs are necessary for achieving those business goals. This means that the analyst considering entire system business logic can clarify for the client how client requirements will affect the logic of the organization, therefore, the client can decide about acceptability of changes before the product is implemented;
   - The TFMfMDA enables validating functional requirements for completeness, functional requirements tracing to use cases through the formal base- functional features of the topological functioning model of the problem domain, and checking functional requirements changes according to the existing business functionality;
   - The TFMfMDA solves some of the use case limitations: information capturing, limitation of other requirements gathering techniques, and completeness checking;
   - The developed approach has a simple, understandable notation. This can stimulates user input in solving the problem.

   A disadvantage of the TFMfMDA application is a lack of tool support. But it should be noticed that methods developed within the TFMfMDA can be partially automated.

The TFMfMDA approach is recommended to apply for complex business systems with expressed functionality and multiple user roles.

At the end of the research, the belief that problem domain modeling can be formalized in some degree within the MDA without adding significant complexity is approved.

The further research direction can be related to the following subjects:

- The research of non-functional requirements handling aspect;
- Tool support development for the TFMfMDA with the integration possibility to the UML supporting development tools;
- Model traceability using the topological functioning model properties.

BIBLIOGRAPHY

[1] Alhir, S. S. UML Extension Mechanisms// Distributed Computing. - 1998. - December. - 29 - 32 p. (Internet. - http://home.comcast.net/~salhir/UMLExtensionMechanisms.PDF) *[2]* Alhir, S. S. Understanding Use Case Modeling// Methods & Tools: Newsletters. - 2000. - Spring (Volume 8 - number 1). - 10-16 p. (Internet.- http://home.comcast.net/~salhir/UnderstandingUseCaseModeling.PDF. http://www.methodsandtools.com/PDF/DMT0100.pdf)

[3] Alhir, S. S. Understanding Structural Modeling// Methods & Tools: Newsletters. - 2001. - Winter (Volume 9 - number 4). - 2-13 p. (Internet.- http://home.comcast.net/~salhir/UnderstandingStructuralModeling.PDF. http://www.methodsandtools.com/PDF/DMT040l .pdf)

[4] Alhir, S. S. Understanding the Model Driven Architecture (MDA)// Methods & Tools: Newsletters. - 2003. - Winter (Volume 11 - number 3). - 17-24 p. (Internet. - http://home.comcast.net/~salhir/UnderstandingTheMDA.PDF, http://www.methodsandtools.com/PDF/dmt0303.pdf)

[5] Alksnis G., Asnina E., Osis J., Silins J. Formalization of Software Development: Problems and Solutions// Scientific Proceedings of Riga Technical University, Series — Computer Science (5), Applied Computer Systems, Volume 22. - Riga: RTU, 2005. - 204 - 216 p.

[6] Alksnis G., Osis J. Formalization of Software Engineering by Means of the Theory of Categories// Scientific Proceedings of Riga Technical University, Computer Science, Applied Computer Systems, 3rd thematic issue.- Riga: RTU, 2002. - 157-163 p.

[7] Arlow J., Neustadt I. UML2 and the Unified Process: Practical Object-Oriented Analysis and Design. - Addison-Wesley, Pearson Education, 2005. - 592 p.

[8] Asnina E. Formal Integration Perspective in the Software Development// Scientific Proceedings of Riga Technical University, Series — Computer Science (5), Applied Computer Systems, Volume 17. - Riga: RTU, 2003,-145-154 p.

[9] Asnina E. Formalization Aspects of Problem Domain Analysis// Proceedings of the 8th International Conference "Information System Implementation and Modelling" (ISIM'05), April 19-21, Hradec nad Moravici, Czech Republic- Ostrava: Jan Stefan MARQ., 2005.-195-202 p.

[10] Asnina E. Formalization Aspects of Problem Domain Modeling within Model Driven Architecture// Databases and Information Systems. Seventh International Baltic Conference on Databases and Information Systems. Communications, Materials of Doctoral Consortium, July 3-6, 2006, Vilnius, Lithuania. - Vilnius: Technika, 2006 (ISBN 9955-28-013-1). - 93-104 p

[11] Asnina E. The Formal Approach to Problem Domain Modelling Within Model Driven Architecture// Proceedings of the 9th International Conference "Information Systems Implementation and Modelling" (ISIM'06), Aprii 25-26, 2006, Pferov, Czech Republic, 1st edn. (ISSN 1613-0073).- Ostrava: Jan Stefan MARQ., 2006.-97-104 p.

[12] Asnina E. Topological Modeling and Arrow Diagram Logic Formalism Application for Software Development// Scientific Papers of University of Latvia, Volume 673, Databases and Information Systems, Doctoral Consortium, Sixth International Baltic Conference BalticDB&IS 2004, Riga, Latvia, June 6-9, 2004. -Riga: University of Latvia, 2004.- 220 - 231 p.

[13] Asnina E., Osis J. Formalization Problems and Perspectives of the Software Development// Scientific Proceedings of Riga Technical University, Computer Science, Applied Computer Systems, 3rd thematic issue.- Riga: RTU, 2002. - 145-156 p.

[14] Asnina E., Osis J. The Computation Independent Viewpoint: a Formal Method of Topological Functioning Model Constructing// Scientific Proceedings of Riga Technical University, Series — Computer Science (5), Applied Computer Systems.- Riga: RTU, 2006. - in press.

[15] AT&T Research: The Real World (by Jackson M., 18/07/2003) / Internet. - http://www.ferg.org/papers/jackson--the real world.pdf

[16] Barr M., Wells C. Category theory for computer science, 2nd ed. - London: Prentice Hall International, 1995.-344 p.

[17] Bernardez B., Genero M., Duran A., Toro M. A Controlled Experiment for Evaluating a Metric-Based Reading Technique for Requirements Inspection// Proceedings of the 10th IEEE International Symposium on Software Metrics (METRICS'04).- IEEE, September 2004.- 257-268 p.

[18] Bowen J.: Formal Specification and Documentation using Z: A Case Study Approach (Revised 2003) / Internet.- http://www.jpbowen.com/pub/zbook.pdf
[19] Cambridge University Computer Laboratory: Categorical Logic (by Pitts A.M., 1995) / Internet. - http://citeseer.ist.psu.edu/pitts01categorical.html
[20] Coad P., Lefebvre E. De Luca J. Java Modeling in Color with UML. - Prentice-Hall, 1999. - 182-203 p.
[21] Columbia University: Knowledge Representation in Health Sciences / Internet. http://www.dbmi.columbia.edu/homepages/wandong/KR/krframes.html
[22] Computer Science, Newcastle University: The Categorical Product Data Model as Formalism for Object-Relational Databases (by Rossiter B.N., Nelson D.A., 1994) / Internet - http://citeseer.ist.psu.edu/rossiter94categorical.html
[23] CPN Group, Department of Computer Science, University of Aarhus, Denmark: The practitioners' guide to coloured Petri nets (by Kristensen L.M., Christensen S., Jensen K., 1998) / Internet. - http://www.daimi.au.dk/~kiensen/papers books/pract.pdf
[24] Diagnosis based on graph models. (By the examples of aircraft and automobile mechanisms). (Diagnostirovanie na graf-modeljah)./ J. Osis, J. Gefandbein, Z. Markovitch, N. Novozhilova - Moscow: Transport, 1991. - 244 p. (in Russian)
[25] Diskin Z., Kadish B., Piessens F., Johnson M. Universal Arrow Foundations for Visual Modeling. // Proc. Diagramms'2000: lst Int. Conference on the theory and application of diagrams. - Springer LNAI, 2000. - No. 1889. - 345-360 p.
[26] Eder J., Frank H. Schema Integration For Object Oriented Database Systems // Proceedings of the ETCE, Software Systems in Engineering.- New Orleans: ASME, 1994.
[27] Farlex Inc. The Free Dictionary Com: Causality/ Internet. - http://encyclopedia.thefreedictionary.com/causality
[28] Ferg S.: What's Wrong with Use Cases? / Internet. - http://www.fere.org/papers/ferg--Whats_wrong_with_use_cases.html
[29] FIS/LDBD: Algebraic Graph-Oriented = Category Theory Based. Manifesto of categorizing database theory (by Diskin Z., Kadish B. 1994) / Internet. - http://citeseer.ist.psu.edu/diskin94algebraic.html.
[30] FIS/LDBD: Database As Diagram Algebras: Specifying Queries and Views via the Graph-Based Logic of Sketches (by Z.Diskin, 1996) / Internet. - http://citeseer.ist.psu.edu/116057.html
[31] FIS/LDBD: Formalization of graphical schemas: General sketch-based logic vs. heuristic pictures (by Z.Diskin, 1995) /Internet.- http://citeseer.ist.psu.edu/diskin95formalization.html
[32] FIS/LDBD: Generalized Sketches as an Algebraic Graph-Based Framework for Semantic Modeling and Database Design (by Z. Diskin, 1997) / Internet. - http://citeseer.ist.psu.edu/diskin97generalised.html
[33] FIS/LDBD: The Arrow Manifesto: Towards software engineering based on comprehensible yet rigorous graphical specifications (by Diskin Z., Kadish B. 1998)/ Internet.- http://citeseer.ist.psu.edu/167037.html
[34] Frankel D.S. MDA and the Object Technology Barrier // The MDA Journal. Model Driven Architecture Straight from the Masters. - Meghan-Kiffer Press, USA, 2004. - 29-46 p.
[35] Frankel D.S. Model Driven Architecture™: Applying MDA™ to Enterprise Computing. - Indiana: Wiley Publishing, Inc., 2003. - 328 p.
[36] Grundspenkis J. Automated transformation of the functional model into the diagnosis knowledge base// Proceedings of 5* Int. Conf. on Quality, Reliability and Maintenance, QRM2004, Oxford, April 1-2. - London, UK: Professional Engineering Publishing (Ed. McNulty), 2004. - 295-298 p.
[37] Grundspenkis J. Automation of Knowledge Base Development Using Model Supported Knowledge Acquisition// Databases and Information Systems: Proceedings of the 2nd International Baltic Workshop, Tallinn, June 12-14, 1996, vol.1.-Tallinn: Institute of Cybernetics 1996.-224-233 p.
[38] Grundspenkis J. Causal Model Driven Knowledge Acquisition for Expert Diagnostic System Development// Application of AI to Production Engineering. Lecture Notes of the Nordic-Baltic Summer School'97. K. Wang and H.Pranavicius (ed.).- Kaunas, Lithuania: Kaunas University Press, 1997.-251-268 p.
[39] Grundspenkis J. Fault Localisation Based on Topological Feature Analysis of Complex System Model// Diagnostics and Identification. - Riga: Zinatne, 1974. - 38- 48 p. (in Russian).
[40] Grundspenkis J., Blumbergs A. Investigation of Complex System Topological Model Structure for Analysis of Failures// Issues of Technical Diagnosis. - Rostov-on-Don: Rostov Institute of Building Engineering, 1981.- 41-48 p. (in Russian).

[41] Holt, Rinehart, Winston: Elements of Language: Examining Causes and Effects. 4* Course (2005) 92-103 p. / Internet. - http://po.hrw.com/eiotM/0030526671/student/ch03/lgl403092 103.pdf
[42] Human and Technology: Structuring Use Cases with Goals (by Alistair Cockburn) / Internet.— http://alistair.cockbum.us/crvstal/articies/sucwg/structuringucswithgoals.htm
[43] Informatique Theorique el Applications: A Category Theory Approach to Conceptual Data Modeling (by Lippe E„ ter Hofstede A.H.M., 1996) / Internet. - http://citeseer.ni.nec.com/lippe96categorv.litml
[44] Institute for Automation and Control Processes, Far East Division, Russian Academy of Science: Experimental Version of the Petri Net Translator into the Executable Code: Petri Nets in Brief, 2000 / Internet. - http://www.iacp.dvo.ru/lab_1 l/otchet/ot2000/pn3.htmi (in Russian)
[45] Jensen K. An Introduction to the Theoretical Aspects of Coloured Petri Nets// A Decade of Concurrency, Lecture Notes in Computer Science vol. 803. - Springer-Verlag, 1994. - 230-272 p.
[46] Kent S. The Unified Modeling Language// Formal Methods for Distributed Processing: An OO Approach. - 2000. - December. - 1-31 p.
[47] Knowledge Systems Corporation: Use Cases: the Pros and Cons / Internet. http://www.ksc.com/article7.htm
[48] Larman Cr. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd ed.- Prentice Hall PTR, 2005. - 703 p.
[49] Leffingwell D., Widrig D. Managing Software Requirements: a use case approach. 2nd ed. - Addison-Wesley, 2003.- 502 p.
[50] Logica UK Ltd.: Object Orientation in Z; Why an Object Oriented Z? (by eds. Stepney S., Barden R. and Cooper D., 1992) / Internet. - http://wvvw-users.cs.york.ac.uk/--susan/bib/ss/ooz/cl.htm
[51] Markovitch Z., Markovitcha I. Modelling as a Tool for Therapy Selection// Proc. Of the 14rd European Simulation Multiconference "Simulation and Modelling". - Ghent, Belgium, May 23-26, 2000.- 621-623 p.
[52] Markovitch Z„ Rekners Ya. Synthesis of Systems Model on Basis of Topological Minimodels// Automatic Control and Computer Sciences, Vol. 32 (3). - New-York: Allerton Press Incorp., 1998. - 59-66 p.
[53] Markovitch Z., Stalidzans E. Expert Based Model Building Using Incidence Matrix and Topological Models// Proc. Of the 12th European Simulation Symposium "Simulation in Industry 2000". - Hamburg, Germany,, Sept. 28-30, 2000. - 328-332 p.
[54] Markovitcha I„ Markovitch Z. Mathematical Model of Pathogenesis of Hard Differentiable Diseases (Matematicheskaja model' patogeneza trudno differenciirujemyh boleznej)// Cybernetics and Diagnostics (Kibernetika 1 Diagnostika), vol. 4. - Riga: Zinatne, 1970. - 21-28 p. (in Russian)
[55] Marshall D. Artificial Intelligence II. Courseware/Internet. http://www.cs.cf.ac.uk/Dave/AI2/Al notes.html
[56] Mathworld: First-Order Logic / Internet. - http://mathworld.wolfram.com/First-OrderLogic.html
[57] Mellor S.J., Clark A.N., Futagami T. Model-Driven Development// IEEE Software.- 2003. - September/October. - 14-18 p.
(Internet, -http://www.es. umb.edu/~ixs/courses/2005/681/readings/mda-mellor.pdf)
[58] Meservy Th.. O., Fenstermacher K. D. Transforming Software Development: An MDA Road Map// Computer. - 2005. - September (Vol. 38, Nr. 9). - 52-58 p.
(Internet.- http://www.cs.umb.edu/~jxs/courses/2005/68 l/readings/mda-meservy.pdf)
[59] Method of Dynamic Parameter Selection for Efficiency Check-up of Continuous Objects that are Represented by Topological Models Taking into Account Non-Equivalence of Checks/ J. Osis, Z. Marckovich, J. Grundspenkis, J. Salenieks, V. Shaitsane. - All-Union Scientific Research Institute for Normalization in Engineering (VNI1NMASH), Gorky Branch, Gorky, 1980. - 39 p. (in Russian)
[60] Miller J., Mukerji J. (eds.). OMG: MDA Guide Version 1.0.1,2003 / Internet. - http://www.omg.org/docs/omg/03-06-01.pdf
[61] Miller J., Mukerji J. (eds.): Model Driven Architecture (MDA). Architecture Board ORMSC, ormsc/2001-07-01 / Internet.- http://www.omg.org/docs/ormsc/01-07-01.pdf
[62] Object-Oriented Software Engineering: A Use Case Driven Approach./ I.Jacobson, M.Christerson, P.Jonsson *et al.* - Addison-Wesley, 1992. - 528 p.
[63] OMG official web-page on MDA / Internet. - http://www.omg.org/mda
[64] OMG official web-page on UML / Internet. - http://www.omg.org/uml
[65] OMG: A Proposal for an MDA Foundation Model. ORMSC White Paper, V00-02, ormsc/05-04-01 / Internet. - www.omg.org/docs/ormsc/05-04-01.pdf
[66] OMG: OMG Unified Modeling Language Specification. Version 1.4. September 2001 / Internet. - http://www.omg.org/docs/formal/01-09-67.pdf.

[67] OMG: UML Extension for Business Modeling, version 1.1 / Internet. – http://umlcenter.visual-paradigm, com/umlresources/exte_1 l.pdf
[68] OMG: Unified Modeling Language (UML) Specification: Infrastructure. Version 2.0. ptc/2004-10-14. October 2004. / Internet.- http://www.omg.org/cgi-bin/doc?ptc/2004-10-14
[69] OMG: Unified Modeling Language: Superstructure. Version 2.0. August 2005 / Internet. - http://www.omg.org/docs/formal/05-04-07.pdf
[70] Osis J. Brief Survey of Object-Oriented Approach// Scientific Proceedings of Riga Technical University, Computer Science, Applied Computer Systems, Vol. 3. - Riga, 2001. - 23-32 p.
[71] Osis J. Development of Object-Oriented Methods for Hybrid System Analysis and Design // Proceedings of the 23rd ASU Conference. - Stara Lesna, Slovakia, 1997. -162-170 p.
[72] Osis J. Diagnostics of Complex Systems (Dissertation of Dr. Habil. Sc. Eng.). - Riga: Latvian Academy of Sciences, 1972.
[73] Osis J. Extension of Software Development Process for Mechatronic and Embedded Systems// Proceeding of the 32nd International Conference on Computer and Industrial Engineering. - Limerick, Ireland, University of Limerick, August 11-13, 2003. - 305-310 p.
[74] Osis J. Investigating troubles of complex system functioning and category theory (Issledovanie narushenij funkcionirovanija slozhnyh system i teorija kategorij).// Cybernetic and Diagnosis (Kibernetika i Diagnostika), vol. 4. - Riga: Zinatne, 1970. - p. 15-20. (in Russian)
[75] Osis J. Mathematical description of complex system functioning (Matematicheskoe opisanie funkcionirovanija slozhnyh sistem).// Cybernetic and Diagnosis (Kibernetika i Diagnostika), vol. 4. - Riga: Zinatne, 1970. - p. 7-14. (in Russian)
[76] Osis J. Programming Language ADA. 1st part, Lections. - Riga: Riga Technical University, 1993. - 59 p. (in Latvian)
[77] Osis J. Software Development with Topological Model in the Framework of MDA// Proceedings of the 9th CaiSE/IFIP8.1/EUNO International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'2004) in connection with the CaiSE'2004, Vol. 1. - Riga: RTU, 2004. -211-220 p.
[78] Osis J. Some questions of microprogramming optimization using topological model properties (Nekotorye voprosy optimizacii mikroprogramm s ispol'zovaniem svoistv topologicheskih modelei).// Cybernetic methods in the diagnosis (Kiberneticheskie metody v diagnostike).- Riga: Zinatne, 1973.- p. 30-34 (in Russian)
[79] Osis J. Topological functioning model support for software engineering// Scientific Proceedings of Riga Technical University, Computer Science, Applied Computer Systems, 4th thematic issue. - Riga, 2003. - 31-42 p.
[80] Osis J. Topological Functioning Model Within the MDA Life Cycle// Accepted to publishing in Scientific Proceedings of Riga Technical University, Computer Science, Applied Computer Systems. - 2006.
[81] Osis J. Topological Model of System Functioning// Automatics and Computer Science. - J. of Acad. Of Sc, Riga, Latvia #6, 1969. - 44-50 p. (in Russian)
[82] Osis J., Beghi L. Topological Modelling of Biological Systems// Proceedings of the third IFAC Symposium on Modelling and Control in Biomedical Systems (Including Biological Systems), D.A. Linkens, E.R. Carson (eds). - Oxford, UK: Elsevier Science Publishing, 1997 - 337-342 p.
[83] Osis J., Sukovskis U., Teilans A. Business Process Modeling and Simulation Based on Topological Approach// Proceedings of the 9th European Simulation Symposium and Exhibition. - Passau, Germany, 1997.-496-501 p.
[84] Podeswa H. UML for the IT Business Analyst: A Practical Guide to Object-Oriented Requirements Gathering/ - Boston: Thomson Course Technology PTR, 2005. - 378p.
[85] Robinson G.P. Model-Based Recognition of Anatomical Objects from Medical Images: Knowledge Representation / Internet. - http://noodle.med.vale.edu/alums/robinson/ivc/section3 2.html
[86] Schach St. R. Classical and Object-Oriented Software Engineering with UML and Java. International edition. 4th edn., - WCB/McGraw-Hill, 1999.
[87] Schneider G., Winters J.P. Applying Use Cases, 2nd ed. A Practical Guide. - The Addison-Wesley, 2001.-245 p.
[88] Selic B. The Pragmatics of Model-Driven Development// IEEE Software. - 2003. - September/October. -19-25 p.

[89] Sharpies M., Hogg D., Hutchison Chr., Torrance S,, Young D. Computers and Thought: A Practical Introduction to Artificial Intelligence. Chapter 6 / Internet. - http://www.informatics.susx.ac.uk/books/computers-and-thought/index.html
[90] Siau K., Rossi M. Evaluation of Information Modeling Methods - A Review// Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences-Volume 5.- IEEE, January 1998.-314-322 p.
[91] Sindre G., Opdahl A.L. Eliciting Security Requirements with Misuse Cases// Requirements Engineering. -2005-October-34-44 p.
[92] Softeam: Making a Success of Preliminary Analysis using UML (by Ph. Desfray) / Internet - www.objecteering.com/pdf/whitepapers/us/modelisation des phases amont.pdf
[93] Sowa J.F. Conceptual Graphs, working draft IS0/JTC1/SC 32/WG2 N 000 (2001)/ Internet - http://www.ifsowa.com/cg/cgstand.htm
[94] Sowa J.F. Conceptual Structures: Information Processing in Mind and Machine. - Addison-Wesley Publishing Company, 1984. - 481 p.
[95] Sowa J.F. Knowledge Representation: Logical, Philosophical, and Computational Foundations.- Brooks Cole Publishing Co., Pacific Grove, CA, 2000. - 594 p.
[96] Sowa J.F. Towards the Expressive Power of Natural Language// Principles of Semantic Networks: Explorations in the Representation of Knowledge. - Morgan Kaufmann Publishers, Inc., 1991. - 157-190 p.
[97] The Object Agency: Be Careful With "Use Cases" (by Edward V. Berard, 1998)/ Internet. - www.toa.com/pub/use-cases.htm
[98] The Open University: Problem Frames and Software Engineering (by Jackson M., 09/12/2004) / Internet -http://mcs.open.ac.uk/mi665/PFrame7.pdf
[99] The Unified Modeling Language Reference Manual/ J. Rumbaugh, I. Jacobson, and G. Booch. - Addison-Wesley, 1999. - 568 p.
[100] Thomas D. MDA: Revenge of the Modelers or UML Utopia? // IEEE Software. - 2004. - May-June. - 22-24 p. (Internet. - http://www.martinfowler.com/ieeeSoftware/mda-thomas.pdf)
[101] University of Latvia: Introduction to Mathematical Logic. Hyper-textbook for students by Detlovs V., Podnieks K. - 2000-2004. / Internet.- http://www.ltn.lv/~podnieks/mlog/ml.htm
[102] University of Tasmania: Object Oriented Modelling with Object Petri Nets (by Lakos Ch., 1997) / Internet. - http://citeseer.nj.nec.com/lakos97obiect.html
[103] Vaidyanathan S. The Role of Model-Driven Architecture in Business Integration // Business Integration Journal. - 2004. - September. - 14-16 p.
(Internet. - http://biionline.com/PDF/vaidvanathan%20sept.pdf)
[104] Wikipedia, the free encyclopedia: Graph Theory / Internet. - http://en.wikipedia.org/wiki/Graph theory