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

Uldis DONIŅŠ
Student of the Doctoral Study Program "Computer Systems"

TOPOLOGICAL UNIFIED MODELING LANGUAGE: DEVELOPMENT AND APPLICATION

Summary of Doctoral Thesis

Scientific Supervisor
Dr.habil.sc.ing., Professor
J. OSIS

Riga 2012

UDK 004.414.23 (043.2)
Do 515 t

Donins U. Topological Unified Modeling Language: Development and Application. Summary of Doctoral Thesis. - R.:RTU, 2012. - 36 p.

Printed in accordance with the decision No 77 of June 26, 2012 of the board of Applied Computer Systems, Faculty of Computer Science and Information Technology, Riga Technical University.

This work has been supported by the European Social Fund within the project «Support for the implementation of doctoral studies at Riga Technical University».

ISBN 978-9934-10-351-3

DOCTORAL THESIS
SUBMITTED FOR THE DOCTORAL DEGREE OF
ENGINEERING SCIENCE AT RIGA TECHNICAL UNIVERSITY

The defence of the thesis submitted for doctoral degree of engineering science will take place at an open session on October 15, 2012 in Meza street 1/3, auditorium 202, Riga Technical University, Faculty of Computer Science and Information Technology.

OFFICIAL OPPONENTS:

Dr.habil.sc.ing., Prof. Janis Grundspenkis
Riga Technical University, Riga, Latvia

Dr.habil.sc.comp., Prof. Janis Barzdins
University of Latvia, Riga, Latvia

Ph.D., Prof. Leszek Maciaszek
Macquarie University, Sydney, Australia

APPROVAL

I confirm that I have developed this thesis submitted for the doctoral degree at Riga Technical University. This thesis has not been submitted for the doctoral degree in any other university.

Uldis Donins ……. (signature)

Date: ……

The doctoral thesis is written in English and includes introduction, 5 chapters, conclusions, and 14 appendices. It contains 224 pages, 71 figure, and 32 tables. Bibliography includes 134 information sources.

TABLE OF CONTENTS

INTRODUCTION

The analysis of problem domain and design of desired solution within software development process has a major impact of the achieved result – developed software. While the software developer community uses a set of tools and different techniques to create detailed specification of the solution, the proper analysis of problem domain functioning is ignored or covered insufficiently. One of such techniques is object-oriented software analysis and development which states that there are two fundamental aspects of systems modeling: analysis and design. The analysis defines what the solution needs to do within the problem domain to fit the customer's requirements, and the design states how the solution will be implemented. The design of object-oriented software for the last decade has been leaded by the Unified Modeling Language (UML) [14]. UML is an approved industry standard modeling notation for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system [57]. While the UML has elements for designing and specifying artifacts of a software system, it lacks the ability to document the functioning of a problem domain by using computation independent constructs. Since the UML is a notation and not a technique or method, its application within software analysis and design is promoted by a set of different software development methods and approaches.

Motivation of the Research

Despite that exists a bunch of software modeling languages (including the UML approved and promoted by Object Management Group (OMG)) and methods that consumes such modeling languages, the way the software is built remains surprisingly primitive (by meaning that major software applications are cancelled, overrun their budgets and schedules, and often have hazardously bad quality levels when released) as outlined by Jones in [37]. This phenomenon can be explained by the fact that the problem domain exists separately from the solution domain (i.e. by not paying appropriate attention to the analysis of the problem domain functioning) [64]. Furthermore, in particular cases the software is built as the developers see the solution and not as the problem domain functions. By reducing or even avoiding proper analysis of the problem domain, the traces between artifacts of problem and solution domains cannot be established. Without these traces the acceptance process of developed software gets meaningless since the customer cannot fully verify the delivered solution (in terms of relating functioning of the problem domain to the delivered software) [6]. To motivate software developers to pay more attention on the analysis and understanding of problem domain and its functioning, an appropriate models and their application method should be provided. The research results outlined by Jones ([37]) has proved that the UML and the existing UML modeling driven methods do not provide such appropriate model and modeling guidelines.

Research Area

The research area in the focus of this research is the topological modeling of system functioning. The topological modeling of system functioning was started in the middle of 1960's in Riga Technical University by Janis Osis. The first theoretical foundations of Topological functioning model (TFM) and its application in the topological modeling of system functioning are represented in [63]. The initial problem domain in which TFM is applied is the diagnostic of mechanic devices (e.g. motor vehicles) based on cybernetics and computer science. Large number of high-quality algorithms and methods related to the TFM application in diagnostic tasks are summarized in [70]. The application of TFM within different problem domains and areas is developed today as well. In fact, the [62] and [63] propose a new foundation of the system theory.

Topological modeling of system functioning has been successfully applied in the field of medical problems solving and diagnostics by Zigurds Markovics since beginning of 1970's [38][48]. The research work continued by Janis Grundspenkis initially was related to investigations of cycle hierarchies for the purpose of rational diagnostic algorithm development [29], [33]. Later the topological modeling research direction by Grundspenkis is evolved as structural modeling [32] which is developed to support systematic causal domain model based knowledge acquisition. The essence of structural modeling is the systematic procedure for construction of three structural (i.e. topological) models representing the morphology, functions and behavior of complex technical system [30]. Structural modeling approach has been implemented in automated structural modeling system ASMOS [31].

The research direction continued by Osis is related to TFM application in the field of object-oriented analysis and object-oriented software development. Recent research results are published as follows: topological modeling application for business process modeling and simulation [73], TFM application in the software development for mechatronic and embedded systems [60], introducing more formalism in problem domain analysis ([4], [5], [16], and [18]), formal analysis of Computation Independent Model (CIM) within Model Driven Architecture (MDA [51]; [61], [64], [68], [85] and [86]), formally specifying Platform Independent Model (PIM) and performing transformation CIM-to-PIM within MDA ([19] and [21]), and analysis and design of embedded systems by applying topological function–architecture co-design method [71]. The theoretical foundations of TFM are summarized and published in monograph [65] where the definitions of TFM are given and the powerfulness of TFM is demonstrated in the context of formal problem domain analysis. This research is the continuation of topological modeling of system functioning evolution within the field of object-oriented analysis and software development.

Purpose of the Research

The goal of the thesis is to supplement UML with theoretical foundations in order to create grounds for converting notation into a formal modeling language and to define

modeling method which allows to clearly trace cause-and-effect relationships in both problem and solution domains.

The tasks of thesis in order to achieve the goal are defined as follows:

1. Explore the evolution of UML and its specification in order to outline the positive and negative aspects of the current language's version application within software development thus identifying aspects of UML that should be improved,
2. Identify UML extension mechanisms and options in order to determine the best suitable extension mechanism to implement the new version of UML,
3. Analyze the main characteristics of a set of UML modeling driven software development approaches and compare their potentialities formalizing the problem domain and creating solution domain design in accordance with the functioning characteristics of problem domain,
4. Develop template for describing UML profile by performing analysis on a set of currently available UML profiles,
5. Develop a new language – Topological UML (TopUML) – in accordance with the identified aspects of UML that should be improved and in accordance with the developed template to specify UML profiles,
6. Specify software development method that supports formal application of created TopUML profile that allows to clearly trace cause-and-effect relationships in both problem and solution domains,
7. Approbate the developed language and its application method in experimental software development project involving into software design process several groups of software development experts, and
8. Approbate the developed language and its application method in a real software development project providing step-by-step case study exploration of developed artifacts.

The research objects are UML and its application methods that support application of UML within software development.

The research subject of the thesis is UML and its application methods, focusing on the formal development of software design models and the establishment of traces between problem domain and solution domain artifacts.

Research Methods

The following research methods are used: mathematical model and modeling language to specify problem domain and solution domain, metamodeling method, model transformation, as well as the following parts of mathematic – general topology, combinatory topology, graph theory, and mathematical logic.

Scientific Innovation and Practical Value

The **scientific innovation** of the research is formal modeling of solution domain in strong accordance with the functioning of problem domain by using TopUML and formal

software designing and development method which is specially developed to drive the application of TopUML diagrams within software development process. Characteristics of the developed TopUML modeling method are compared with the set of currently existing UML modeling driven software development methods and techniques.

The **practical value** of the research is specification of TopUML profile which combines formalism of TFM mathematical topology and specification standard of OMG, and specified software analysis and design method. The developed modeling method enables application of TopUML diagrams in a formal software development process and consists of formally defined designing activities thus enabling solution development in accordance with functioning characteristics of problem domain and clearly tracing cause-and-effect relationships in artifacts of problem and solution domains. Since the TopUML is developed as UML profile, it can be implemented in any existing UML modeling tool that supports definitions of custom profiles.

Approbation of the Work Results

The main results of the research are presented in the following international scientific conferences (two were held in Latvia and five in foreign countries):

1. 7th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2012), Wroclaw, Poland, June 29-30, 2012,
2. 13th International Conference on Enterprise Information Systems (ICEIS 2011), Beijing, China, June 8-11, 2011,
3. 3rd International Workshop on Model-Driven Architecture and Modeling Driven Software Development (MDA & MDSD 2011) in conjunction with 6th International Working Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2011), Beijing, China, June 8-11, 2011,
4. 2nd International Workshop on Model-Driven Architecture and Modeling Theory-Driven Development (MDA & MTDD 2010) in conjunction with 5th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2010), Athens, Greece, July 22-24, 2010,
5. 13th East-European Conference on Advances in Databases and Information Systems (ADBIS 2009), Riga, Latvia, September 7-10, 2009,
6. 4th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2009), Milano, Italy, May 9-10, 2009, and
7. The 49th Scientific Conference of Riga Technical University, Riga, Latvia, October 13-15, 2008.

The main results of the research are published in the following scientific papers:

1. Donins U. Semantics of Logical Relations in Topological Functioning Model// Proceedings of the 7th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2012) – 2012. (To be published)
2. Donins U., Osis J., Asnina E., Jansone A. Formal Analysis of Objects State Changes and Transitions// Proceedings of the 7th International Conference on

Evaluation of Novel Approaches to Software Engineering (ENASE 2012) – 2012. (To be published)

3. Donins U., Osis J. Topological Modeling for Enterprise Data Synchronization System: A Case Study of Topological Model-Driven Software Development// Proceedings of the 13th International Conference on Enterprise Information Systems, Volume 3. - Beijing, China: SciTePress, 2011. - pp. 87-96 [Indexed by Thomson Reuters, Inspec, EI, DBLP, ISTP]
4. Donins U., Osis J., Slihte A., Asnina E., Gulbis B. Towards the Refinement of Topological Class Diagram as a Platform Independent Model// Proceedings of the 3rd International Workshop on Model-Driven Architecture and Modeling-Driven Software Development. - Beijing, China: SciTePress, 2011. - pp. 79-88 [Indexed by Thomson Reuters, Inspec, EI, DBLP]
5. Slihte A., Osis J., Donins U., Asnina, E., Gulbis, B. Advancements of the Topological Functioning Model for Model Driven Architecture Approach// Proceedings of the 3rd International Workshop on Model-Driven Architecture and Modeling-Driven Software Development. - Beijing, China: SciTePress, 2011. - pp. 91-100 [Indexed by Thomson Reuters, Inspec, EI, DBLP]
6. Asnina E., Gulbis B., Osis J., Alksnis G., Donins U., Slihte A. Backward Requirements Traceability within the Topology-based Model Driven Software Development// Proceedings of the 3rd International Workshop on Model-Driven Architecture and Modeling-Driven Software Development. - Beijing, China: SciTePress, 2011. - pp. 36-45 [Indexed by Thomson Reuters, Inspec, EI, DBLP]
7. Slihte A., Osis J., Donins U. Knowledge Integration for Domain Modeling// Proceedings of the 3rd International Workshop on Model-Driven Architecture and Modeling-Driven Software Development. - Beijing, China: SciTePress, 2011. - pp. 46-56 [Indexed by Thomson Reuters, Inspec, EI, DBLP]
8. Donins U. Software Development with the Emphasis on Topology// Advances in Databases and Information Systems (Lecture Notes in Computer Science, Vol.5968). - Berlin, Germany: Springer-Verlag, 2010. - pp. 220-228 [Indexed by SCOPUS, Springer, DBLP]
9. Osis J., Donins U. Platform Independent model Development by Means of Topological Class Diagrams// Proceedings of the 2nd International Workshop on Model-Driven Architecture and Modeling Theory-Driven Development - Portugal: SciTePress, 2010. - pp. 13-22 [Indexed by SCOPUS, Thomson Reuters, Inspec, DBLP]
10. Osis J., Donins U. Formalization of the UML Class Diagrams// Evaluation of Novel Approaches to Software Engineering (Communications in Computer and Information Science (CCIS), Volume 69). - Berlin, Germany: Springer-Verlag, 2010. - pp. 180-192 [Indexed by SCOPUS, Springer]

11. Osis J, Donins U. Modeling Formalization of MDA Software Development at the Very Beginning of Life Cycle// Advances in Databases and Information Systems. 13th East-European Conference, ADBIS 2009: Associated Workshops and Doctoral Consortium, Local Proceedings. - Riga, Latvia: JUMI Publishing House Ltd., 2009. - pp. 48-61 [ISBN 978-9984-30-163-1]
12. Osis J., Donins U. An Innovative Model Driven Formalization of the Class Diagrams// Proceedings of the 4th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2009). – Portugal: INSTICC Press, 2009. - pp. 134-145 [Indexed by SCOPUS, Thomson Reuters, Inspec, DBLP]
13. Donins U., Osis J. Reconciling Software Requirements and Architectures within MDA// Scientific Proceedings of Riga Technical University, Computer Science (Series 5), Applied Computer Systems (Vol. 38). - Riga, Latvia: RTU Publishing house, 2009. - pp. 84-95 [Indexed by DBLP]

In addition to the scientific papers, a monograph has been published:

1. Doniņš U. Topological Business Systems Modeling and Software Systems Design. - Riga, Latvia: RTU Publishing house, 2011. - 65 p. (in Latvian) [ISBN: 978-9934-10-136-6]

Thesis Outline

The thesis consists of introduction, five chapters, conclusions, twelve appendices, and bibliography. The doctoral thesis contains 224 pages, 71 figure, 32 tables, and 14 appendices. Bibliography includes 134 information sources.

Introduction gives motivation of the thesis, research goal, tasks defined to reach the goal, novelty and practical value of the research together with the approbation and the main results achieved as well.

Chapter 1 represents the research on UML, including the review of its evolution. The research on UML shows the benefits and limitations of applying it in software development. As a result UML improvement options are outlined.

Chapter 2 analyzes methods and approaches that support and promote the use of UML within software development process. Result of review shows positive and negative aspects of the analyzed UML modeling driven methods.

The UML extension mechanism – profiles – is covered in *Chapter 3* as well as the development of the TopUML profile. TopUML is a combination of UML and formalism of TFM and is based on the principles of metamodeling; it extends the UML version 2.4.1 by adding TFM to UML and topological functioning characteristics into UML diagrams.

Chapter 4 presents the TopUML modeling – a method intended for systematical application of TopUML profile within software development analysis and design phase. TopUML modeling is defined as a set of activities. Each activity defines the input and the output artifacts. The application of these activities can vary from project to project. Additionally Chapter 4 compares TopUML modeling with the UML modeling driven approaches covered in Chapter 2.

In *Chapter 5* application and approbation of TopUML language and modeling method in the context of experimental software development and case study is explored and described. Case study is a step-by-step exploration of TopUML application in real software development project.

The *Conclusions* summarizes the results of this research, gives conclusions and future research directions.

Thesis contains fourteen *appendices*: 1) Used Abbreviations, 2) TopUML Specification, 3) Mappings Between TopUML Diagrams, 4) Informal Description of Laundry Functioning, 5) Functional Requirements and System Goals of the Laundry Software System, 6) Functional Features of Laundry Functioning, 7) Closuring of Laundry Functioning Topological Space, 8) Sequence Diagrams Representing Behavior of Laundry, 9) Laundry System Topological Class Diagram, 10) Lattelecom Technology Ltd. Acknowledgement of Software Development by using TopUML Modeling Method, 11) Specification of Enterprise Data Synchronization System, 12) Functional Features of Enterprise Data Synchronization System, 13) Self-Evaluation Questionnaire, and 14) OMG-Certified UML Professional Certificate.

1. UNIFIED MODELING LANGUAGE – A STANDARD FOR SOFTWARE DESIGN SPECIFICATION

UML is a graphical language officially defined by OMG for visualizing, specifying, constructing, and documenting the artifacts[1] of a software-intensive system [57]. It offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components [28]. UML became widely accepted as the standard for object-oriented analysis and design soon after it was first introduced in year 1997 [42] and still remains so today [14]. Since the release of first UML version a large number of practitioner and research articles and dozens of textbooks have been devoted to articulating various aspects of the UML, including guidelines for using it. Some of the research areas on UML are as follows:

- Formalization of UML semantics (e.g., [23], [34] (both after UML 1.1 was released), and [89] (after UML 2.0 was released)),
- Extending the UML (e.g., [49], [69], and review of a number of UML profiles developed by different researchers and groups [75]),
- Formalizing the way the UML diagrams are developed (e.g., [64] and [67]),
- Ontological analysis of UML modeling constructs (e.g., [92]),
- Empirical assessments (e.g., [14] and [24]),
- Analysis of the UML's complexity (e.g., [22], [81], and [82]),

[1] An artifact in software development is an item created or collected during the development process. Example of artifacts includes use cases, requirements, design, code, executable files.

- Difficulties of learning UML (e.g., [83]) and how to avoid them (e.g., [7]),
- Transformations between UML diagrams (e.g., [46], [43], and [50]),
- Software code generation and related issues with generated code quality (e.g. [45] and [80]), and
- Experiments that evaluate aspects of UML models effectiveness (e.g., [12]).

The large number of researches regarding UML evolving and strengthening is caused by the basis on which UML was developed. According to Dobing and Parsons [14] the "*UML was not developed based on any theoretical principles regarding the constructs required for an effective and usable modeling language for analysis and design; instead, it arose from (sometimes conflicting) "best practices" (e.g. Booch* [8], *OMT* [77], *OOSE* [36]*) in parts of the software engineering community*".

The review of elements that build up UML within this chapter is based on UML version 2.4.1 specification which is divided into two volumes: Infrastructure [56] (core metamodel); and Superstructure [57] (notation and semantics for diagrams and their model elements). Actually, the Superstructure specification is based on Infrastructure specification. The set of modeling concepts of UML is partitioned into horizontal layers of increasing capability called compliance levels (the compliance level is needed to take into consideration when developing or choosing modeling tools [28]).

1.1. Formalism of UML and UML Formalization Attempts

The UML specification is defined by using a metamodeling approach which adapts formal specification techniques. A metamodel is used to specify the model that comprises UML. In spite of using metamodeling approach, the UML specification method lacks some properties of formal specification methods. The specification of UML cannot be considered as formal specification because of natural language (English) use in it. UML specification [56] underlines that the specification as a metamodel does not eliminate the option of specifying it later by using formal/mathematical language (e.g., OCL [91], Z [87], PVS [74], or RAISE [52]). The formalization of UML specification has following benefits [23]: clarity, equivalence and consistency, extendibility, refinement, proof, and tools that make use of semantics require that semantics to be precise. The current UML semantics are not sufficiently formal to realize all of the above listed benefits.

Part of formalization researches is restricted to the semantics of models, while others are concerned with issues of reasoning about models and model transformations. Currently there exist a number of approaches for specifying and formalizing semantics of UML by using: formal languages (e.g., using Z [23] or Object-Z [40]), category theory (captures relationships between specification objects; e.g., [1] and [13]), stream theory (e.g., [11]), π-calculus or process algebra (e.g., [93]), and algebraic approaches (e.g., [89]). The researches on UML semantics formalization relate to the internal consistency of the UML, not to its relationship to problem domains [25]. To address the relation of UML elements to problem

domains, researches are ongoing on formalizing the way the software is developed by using UML diagrams ([16], [65]) and describing UML constructs by using ontology ([25], [92]).

1.2. Benefits and Disadvantages of Applying UML

While the application of UML within software development has a number of benefits, it also has some disadvantages. The main benefits are ([2], [14], [27], [57], and [59]): UML is independent of software development methods, techniques and platforms; it has an extension mechanism thus allowing to solve specific modeling tasks; and the models can be transferred between different tools from different tool vendors since UML is defined in accordance with metadata interchange (XMI). The main disadvantages of UML application is its size, incoherence, different interpretations, frequent subsetting, and the lack of causality ([14], [16], [23], [39], [69], and [84]). From these disadvantages rises a set of problems like ambiguous semantics, cognitive misdirection during the development process, inadequate capture of properties of system under consideration, lack of appropriate supporting tools and developer inexperience, and inability to trace cause-and-effect relationships between the existing artifacts in problem domain and created artifacts in solution domain.

1.3. Summary

By taking a closer look at benefits and disadvantages of applying UML within software development, it is visible that some benefits turn into disadvantages (e.g., independency of software development methods leads to cognitive misdirection during the development process). To address the listed disadvantages, a bunch of researches on UML strengthening and formalization are performed.

UML can be strengthened by using mathematical topology thus addressing the disadvantage of lacking causality [16]. In this case UML needs to be improved by supplementing it with the topological and functioning characteristics of TFM. To allow using topology in UML diagrams, it should be extended thus creating a new kind of UML– *Topological Unified Modeling Language* (*TopUML*). The core framework proposal for TopUML profile is presented in [69]. The first research results shows that the transfer of topological and functioning characteristics from TFM to UML is sufficient for clearly tracing cause-and-effect relationships in both – problem and solution – domains.

Next chapter is dedicated to explore currently existing UML modeling driven software development approaches, thus addressing the disadvantages of UML's size, incoherence, different interpretations, and frequent subsetting.

2. SOFTWARE DESIGNING WITH UML MODELING DRIVEN APPROACHES

UML is a notation and as such its specification does not contain any guidelines of software development process. Despite that UML is independent of particular methods, most of the UML modeling driven methods uses use case driven approach [14]. This might be

caused by the originators (Booch, Rumbaugh, and Jacobson) of the UML since they recommend a use case driven process in "The Unified Modeling Language User Guide" ([10]). A majority of UML modeling driven approaches since then has endorsed this view, and most contain at least some further prescriptions for applying the UML in modeling (e.g., [44], [76], and [88]). There is also difference in the use of use case narratives across various methods due to the lack of guidance on narrative format in the UML specification. The UML specification [57] only states that "use cases are typically specified in various idiosyncratic formats such as natural language, tables, trees, etc. Therefore, it is not easy to capture its structure accurately or generally by a formal model."

A successful software development project can be measured against deliverables, delivery schedule, and that created result is resilient to change and adaptation. For software development project to be successful by means of given measurements, it should satisfy the following two characteristics [9]:

1. Solution should have a strong architectural vision, and
2. A well-managed development lifecycle should be used.

Software architecture is "fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution" [35]. Good software architectures tend to have several attributes in common [9]:

1. They are constructed in well-defined layers of abstraction,
2. They have a clear separation of concerns between the interface and implementation of each layer, and
3. The architecture itself is simple – common behavior is achieved through common abstractions and common mechanisms.

Currently exist a number of UML modeling driven software development approaches, e.g., software development lifecycles [79], use case driven methods [76], model driven architecture [41], pattern based development [44], component based development [88], and conceptual modeling [53]. The review of software development methods discusses a number of existing UML modeling driven software development approaches paying the most emphasis and attention on the use and application of UML diagrams (i.e., which diagram types for what purpose are used and in which sequence they should be created). The analysis of UML diagram usage additionally shows if there are included transformation rules or guidelines between different diagram types. Overview of the current state of the art of UML based software development approaches includes approaches that are well known in software development industry [14], formalizes the development process and problem domain [64], and are used in the conjunction of software development tools [47].

2.1. Benefits and Limitations of UML Modeling Driven Approaches

By combining UML together with some modeling method, it can be used as a powerful tool to analyze and understand both problem domain and software system and to design planned software system. Despite the fact that UML modeling driven approaches

provides a systematical use of UML diagrams, these approaches do cover different parts of a software development lifecycles and accordingly uses only a subset of UML diagrams. Due to this, the software developers are forced to combine together several modeling methods, thus the application of UML gets more complicated and incomprehensible. Whole software development lifecycle is covered only by the Unified process [3] and Microsoft Solutions Framework (MSF) [90], other methods focuses more on analysis (e.g. Business Object Oriented Modeling (B.O.O.M.) [76], TFM for MDA (TFMfMDA) [4], and Conceptual modeling [59]) while others – more on design and less on analysis (e.g., Pattern based design [44], Component based development[88]). This impacts the number of UML diagram types that are used by each of the method. Greatest amount of applied diagram types among the reviewed methods is within the Unified process.

While the benefit of applying Unified process is the coverage of whole software development lifecycle, it has some limitations – the Unified process promotes use case driven analysis of problem domain. As such the Unified process does not provide a formal way of analyzing and formalizing the problem domain. The only formal method for problem domain formalization among the reviewed methods is TFMfMDA. It uses TFM as a technique for both problem and solution domain analysis and formalization. While TFMfMDA has formalized the very beginning of software development lifecycle, its largest limitation is the Conceptual class diagram and its development. TFM describes the functionality of the problem domain and solution domain (including the responsibilities through the whole system). When TFM is transformed into Conceptual class diagram this important information of responsibilities from TFM is not transferred to Class diagram. "*Deciding what operations belong where, and how the objects should interact, is terribly important and anything but trivial. This is a critical step - this is at the heart of what it means to develop an object-oriented system, not drawing domain model diagrams, package diagrams, and so forth*" [44].

The analysis of UML application in software development industry [14] shows that the five most applied diagram type among UML diagrams are: Class, Use case, Sequence, Activity, and State diagram. This fact is tightly related with the UML modeling driven methods – the review of methods shows that the five most applied UML diagram types within them are the same as listed in [14].

2.2. Summary

The application of modeling methods within software development process reduces and even solves several disadvantages of UML identified in previous chapter. They are as follows:

- *Size* – systematic and consistent software analysis and design activities solves issue related with the large amount of UML diagrams and their elements,
- *Incoherence* – through the predefined actions the modeling method tries to develop diagram by diagram thus showing the seams and transitions between them,

- *Different interpretations* – UML semantics together with methodical application of UML diagrams creates shared understanding among stakeholders, and
- *Frequent subsetting* – providing UML extension (e.g. profile) together with a proper modeling method it is clearly visible how it is related to UML elements and diagrams and how software development process can benefit from the developed extension.

Unfortunately, the partial coverage of the software development lifecycle and the fragmentary application of UML diagrams within reviewed modeling methods do not eliminate above listed disadvantages at a sufficient level that is required for an effective and usable software analysis and design method. An effective and usable method has the following characteristics: it allows achieving the desired result (formalization of problem domain and designing of solution in accordance to identified functioning characteristics of problem domain), and it offers adequate means for clearly identifying cause-and-effect relationships within problem domain artifacts, as well as in solution domain artifacts.

3. IMPROVING UNIFIED MODELING LANGUAGE

Extension of UML can be done in two ways – by using "*lightweight*" extension and by using "*heavyweight*" extension [56]. The lightweight extension is done by using profiles thus defining a new dialect of UML. The heavyweight extension is done by using metamodeling based on Meta Object Facility [54] (MOF). The MOF based extension of UML is intended to redefine existing metamodels and define new ones in accordance with the metamodeling principles. It is needed to remark, that by using MOF based extension all the benefits of creating profile are lost and it can be a difficult task to put the new language into practice. If there is need to extend the UML, at first it is needed to draw the scope of UML extension:

- If the new language will use most of the UML, then profiles are suitable choose for that solution, and
- If the new language uses only small part of UML or there is need to use more complex features of UML such as redefinition, then creating a complete new language by using MOF metamodeling should be considered.

In fact, the most common and suitable way for improving UML is to use its extensibility mechanisms – the profiles. By improving UML with the profile mechanism, it is possible to adapt and use ordinary UML compliant modeling tools [78]. Thus, by creating a profile of UML the costs of adaption in industry for such new language is lowered and it can be adapted faster.

3.1. Developing a Profile for UML

Developing a profile for UML should be done in consistent way by using some unified profiling approach or template. Since UML specification contains only the definition of elements that are building up a profile and does not provide guidelines or process on how to apply these elements, before creating a profile for UML it is needed to define guidelines of

profile development. Guidelines for profile definition are based on the review of four different profiles (*Executable UML* [49] (xUML), *TFMfMDA* [4], *Object Modeling Group System Modeling Language* [58] (*OMG SysML*), and *Service Oriented Architecture Modeling Language* [55] (*SoaML*)). The review of UML profiles shows that there is no unified profile definition template or approach – each author defines profile on its own ([75]). Only two of four reviewed profiles – OMG SysML and SoaML – have huge similarities in the profile specification (the specification structure is about 85% the same). The specification of these two profiles follows the overall specification structure of UML; thus if the reader is familiar with the UML specification understanding of these profiles is relieved. Summarizing up issues related to UML profile specification techniques and templates, guidelines for profile development are provided. These guidelines are applied for profile specified in next subsection.

3.2. Topological Unified Modeling Language – an UML Improvement

The main aim of improving UML is by refining its elements with formalism and mathematics of TFM, thus eliminating the lack of cause-and-effect relationships within the current UML specification. Topological Unified Modeling Language (TopUML) is a combination of UML and formalism of TFM and is based on the MOF metamodeling principles. Idea of Topological UML is adapted from [60] where it is shown that "there is a lack of mathematical formalism by drawing UML diagrams".

TopUML is developed as a profile of UML and its specification takes advantage of the package merge feature of UML to merge extensions into UML. TopUML development is based on following steps:

1. Extend UML by using its extension mechanism, thus developing a TopUML profile, and
2. Define guidelines for using TopUML in practice (thus formalizing the way the TopUML is used).

According to the TopUML base idea to combine formalism of TFM with UML and to create TopUML in accordance with UML extension mechanisms, the new language includes all diagram types from UML and a new diagram type – Topological functioning model (thus making a family of fifteen diagrams). The analysis of topology in UML diagrams shows that there are two diagrams which should be extended in order to include topological relationship: Class diagram and Use case diagram. Thus, the extended version of UML allows achieving goal of the thesis: clearly tracing cause-and-effect relationships in both problem and solution domains. The extended versions of these two diagrams are called *"Topological class diagram"* and *"Topological use Case diagram"*. The profile diagram specifying TopUML language consists of four packages, eight stereotypes, two enumerations and three metamodels (one for TFM and each extended diagram). The top-level profile diagram of TopUML is given in Figure 3.1 which shows the related metamodel and relationships between packages in the profile. The TopUML profile diagram is developed according to UML specification and by using elements of UML.

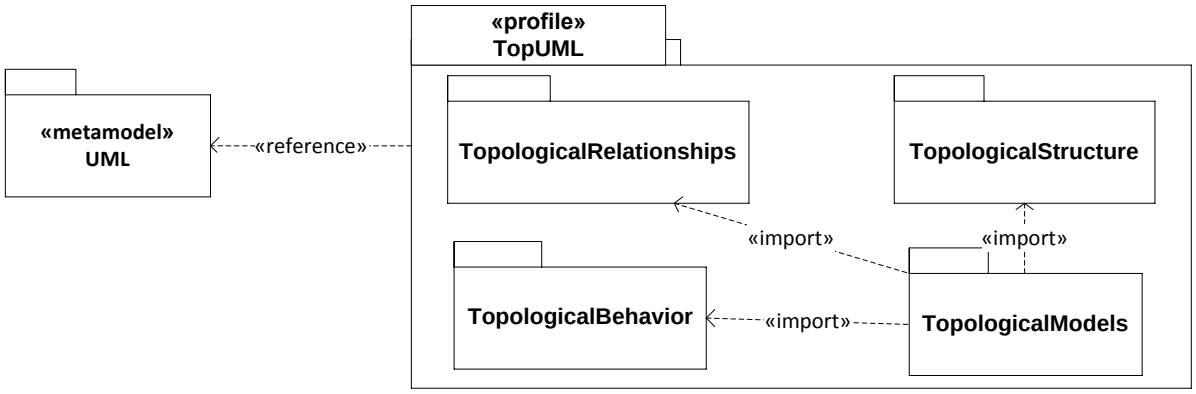

Figure 3.1. TopUML profile top level package

The packages are used to group together elements basing on their intent and semantics and to ease the evolution of TopUML (i.e. creation of new TopUML versions). The packages that build up TopUML profile are as follows:

- *TopologicalRelationships* – contains constructs related to relationships:
  - *TopologicalRelationship* – topological relationship is a binary relation that shows a cause-and-effect relation between two elements – source and target element,
  - *LogicalRelationsip* – represents logical relation between two or more topological relationships; shows conjunction, disjunction, and exclusive disjunction,
- *TopologicalBehavior* – contains constructs related to behavior modeling:
  - *FunctionalFeature* – functional feature is a description of an atomic business action; each functional feature is a unique tuple (stereotype *FunctionalFeature* is an abstraction of this tuple),
  - *Condition* – shows pre- and post- conditions within system; to enter the execution of behavior (e.g., functional feature) all preconditions of it should be true and to exit the execution of this behavior all postconditions should be evaluated to true,
  - *ActionResult* – specifies a result of object's action together with affected objects,
- *TopologicalStructure* – contains constructs related to structure representation:
  - *TopologicalCycle* –represents directed functional cycle of system,
  - *TopologicalOperation* – a behavioral feature of classifier that specifies the name, type, parameters, and constraints for invoking an associated behavior, and related functional features and topological relationships for specifying cause-and-effect relations within system,
- *TopologicalModels* – contains diagram types added to UML by TopUML profile:
  - *TopologicalFunctioningModel* – represents TFM by using UML metamodeling constructs. TFM is a mathematical model that shows functioning of a system in the form of directed graph consisting of functional features and topology between them. Functional features embed information of systems functioning and its structural description while topology defines cause-and-effect relations between them.

TopUML profile packages are designed to provide the necessary constructs to create Topological functioning model, Topological class diagram, and Topological use case diagram. Stereotypes included into each package are used across multiple diagram types thus making TopUML profile more compact and without needless constructs.

Elimination of the UML disadvantage of *lacking causality* by supplementing it with mathematical topology and thus creating TopUML profile emerges another UML disadvantages – *size* and *frequent subsetting*. To address these issues, a TopUML modeling method needs to be provided together with the new profile. The TopUML modeling method should include following aspects:

1. Proper analysis of problem and solution domains (all software artifacts needs to be an abstraction of a well analyzed and understood problem domain unit),
2. Cover most of the UML diagrams and software development lifecycle to eliminate the need to combine together several modeling methods,
3. The developed artifacts are with high cohesion, and
4. Components of developed system need to have low coupling with the rest of the system and a well-defined interface.

3.3. Summary

Development of UML profiles is a challenging activity as well as construction of other UML diagrams while the UML specification defines only the modeling language together with all its elements. Since the UML specification is a specification of a notation, it does not include any guidelines for profile definition and specification. Thus, before a new UML profile development it is necessary to determine the method for specifying the profile and structure of this specification. Both the analysis of UML profiles within thesis and systematic review of UML profiles in [75] outline the lack of profile definition guidelines and structure which leads to the current situation when UML profiles are developed in inconsistent ways. This makes it hard to read and understand profiles proposed and created by different authors. A template for specifying an UML profile is proposed according to the UML profile analysis results. The template suggests that the most convenient way for specifying an UML profile is by using the same specification structure as used to specify UML itself (if the reader is familiar with UML specification it is easier to read and understand the specification of profile). TopUML profile is specified in accordance with the proposed specification template.

When a decision is made on developing a new modeling language based on UML, at first it is needed to draw the scope of desired language and to determine the extent of reusable parts of existing language. If the new modeling language will use most of the UML, then profile development is a suitable choice, while the MOF based solution is more suitable in situations when the new language uses only small part of UML or there is need to use more complex features of UML. An additional benefit of using UML profiles is ability to implement it in any existing UML modeling tool that supports definitions of custom profiles.

The proposed TopUML profile supplements UML with topological and functioning characteristics of TFM, thus eliminating the lack of causality relationships within the UML

specification. The developed TopUML profile ensures elements that allow clearly tracing causal relations in both problem and solution domains. In addition TopUML profile is supplemented with mappings between its diagrams and diagram elements thus showing transformation patterns between different diagram types. To use TopUML profile elements for sufficient definition of causal relationships it is needed to use appropriate modeling method. The next chapter defines such modeling method for applying TopUML profile in practice.

4. TOPUML MODELING – A METHOD FOR DESIGNING SOFTWARE

TopUML modeling for problem domain modeling and software systems designing is a model-driven approach. It combines TFM and its formalism with elements and diagrams of TopUML profile. This modeling method has been developed to eliminate the disadvantages identified in UML and its application within software development process, e.g., ignorance of cause-and-effect relationships, incoherence between diagrams, and different interpretations of the same language element. The TFM considers problem domain information separate from the solution domain information and holistically represents a complete functionality of the system from the computation independent viewpoint while TopUML profile has elements of representing system design at the platform independent viewpoint and platform specific viewpoint in the context of MDA.

4.1. Transitions between TopUML Diagrams

The proposed transitions between TopUML diagrams are given in Figure 4.1 where the diagrams are shown as object nodes and the edges between them as object flow within Activity diagram.

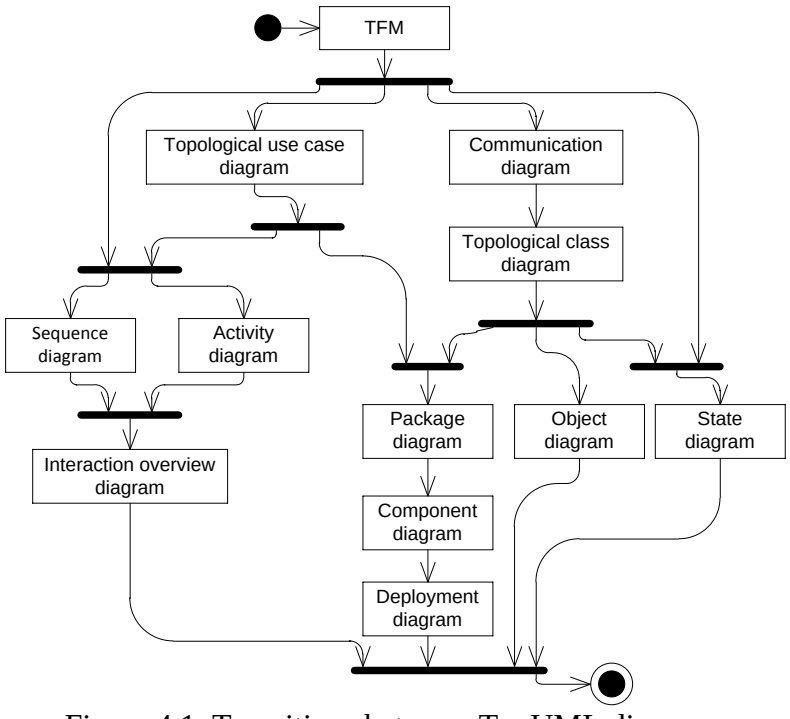

Figure 4.1. Transitions between TopUML diagrams

Most of the transitions can be automated while the validation and checking of the acquired diagrams are needed by the domain experts. The development of the root model – TFM – can be partly automated as shown in [72] where business use cases are transformed into functional features and topological relationships between them; while the other diagrams are obtained by transforming and applying developed TFM (the later development activities uses also other types of diagrams as a transformation source model).

The TopUML diagrams that are used within TopUML modeling are listed in Table 4.1, where a development order (column "*D.o.*") of the diagram is given as well as the diagrams to which it can be transformed or has information for development. The development order is given for the top-down development.

Table 4.1

TopUML diagrams used within TopUML modeling

No	**TopUML diagram**	**D.o.**	**Development information for**	**Description**
1.	Topological Functioning Model	1	Topological use case, Sequence, Activity, Communication, and State diagrams	Initial TFM is developed by analyzing functional characteristics of the problem domain. The refinement of TFM includes adjusting TFM to the functional requirements of the desired software system since the requirements can introduce new functionality to the problem domain. By refining TFM the functional requirements are validated, i.e. the TFM shows missing requirements.
2.	Topological Use Case diagram	2	Sequence, Activity, and Package diagrams	The scope of Use Cases is set either by functional requirements or by system goals. The functionality represented by each Use Case is obtained from the TFM according to the mappings between functional features and functional requirements.
3.	Sequence diagram	3	Interaction overview diagram	Sequence diagram shows the messaging between actors and objects. Usually a set of Sequence diagrams is created – one for each Use Case. Use Case is used to set the scope of Sequence diagram while TFM is used to set the messages and their order.
4.	Activity diagram	3	Interaction overview diagram	Activity diagram shows the workflow of a Use Case. Usually a set of Activity diagrams is created – one for each Use Case. Use Case is used to set the scope of Activity diagram while

No	TopUML diagram	D.o.	Development information for	Description
				TFM is used to set the action nodes and edges.
5.	Interaction overview diagram	4	-	Defines interactions through a variant of Activity diagram in a way that promotes overview of the control flow. Interaction overview diagram focus on the overview of the flow of control.
6.	Communica -tion diagram	2	Topological class diagram	Communication diagram is used as an intermediate model between TFM and Topological class diagram. It is developed by transforming TFM – the functional features representing the same object type are merged and the cause-and-effect relations become links between lifelines.
7.	Topological class diagram	3	Package, State, and Object diagrams	Topological class diagram is used to represent a domain model and a system design model. The key idea behind domain model is a visual dictionary of abstractions. The topological relations between classes show the causal relations between entities in the problem domain.
8.	Object diagram	4	-	Object diagram can be developed during the refinement process of Topological class diagram when the associations are analyzed. It is useful in situation when object of one type plays more than one role at a time. Object diagram also can be used to provide examples of system at a specific time
9.	State diagram	5	-	State diagrams are used to show the state transitions of objects; one State diagram is created for each object type.
10.	Package diagram	6	Component diagram	Package diagram is used to organize and group classes into logical structure – packages. Each package represents a subsystem and groups a set of cohesive responsibilities of classes.
11.	Component diagram	7	Deployment diagram	Component diagram represents modular, deployable, and replaceable parts of a system; one component is created for each package.
12.	Deployment	8	-	Component diagram shows how instances of

No	TopUML diagram	D.o.	Development information for	Description
	diagram			components are deployed on instances of nodes. The content of Deployment diagram is denoted by components and nonfunctional requirements.

4.2. TopUML Modeling Activities

The process of TopUML modeling method is given in Figure 4.2 as Activity diagram, where each action shows one modeling activity and links between them – sequence of activities for the top-down development.

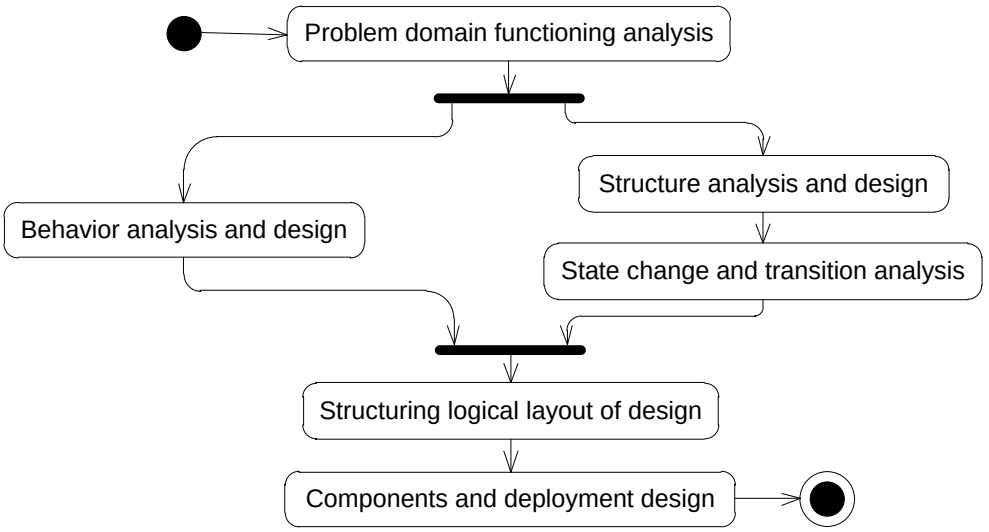

Figure 4.2. TopUML modeling activities

Problem domain analysis and software system design with TopUML modeling method consists of following six activities (for each activity an input (i.e., required artifacts) and an output (i.e., produced artifacts) is defined together with involved modeling actions):

1. *Problem domain functioning analysis* – during this activity a TFM representing functioning of problem domain [16], TFM representing functionality of desired software system [15], and mappings between functional features and functional requirements are developed [18].
2. *Behavior analysis and design* – using TFM as an information source Topological use case, Sequence, Activity, and Interaction overview diagrams are constructed. Topological use case diagram reflects information about subsystems according to results of performing TFM closuring operation. [15][19]
3. *Structure analysis and design* – by transforming TFM that specifies solution domain a Communication diagram is obtained, after that it is transformed to Topological class diagram which contains classes and topological relationships between them. In fact, the transformation of TFM ensures that responsibilities are assigned to classes

precisely and formally in accordance to functioning characteristics of the solution domain. By refining initial Topological class diagram it gets supplemented with relationships of other type (e.g., associations, generalizations) and Object diagrams are developed. [17][67]

4. *State change and transition analysis* – object state change and transition analysis is based on the State diagram and consists of TFM transformation into it. State diagram is obtained and analyzed for each object participating in the main functioning cycle of the system (these objects are most important within the system). [20]
5. *Structuring logical layout of design* – the logical layout is depicted by using Package diagram where each package initially represents one subsystem. The contents of packages are added from the Topological class diagram accordingly to the Use Cases in each system and the mappings between functional features and use cases. [19]
6. *Components and deployment design* – the input of this activity is packages from Package diagram and nonfunctional requirements, and as the output a Component and Deployment diagrams is created. [19]

4.3. Summary

The problem domain analysis and software design within TopUML modeling method consists of six activities that cover analysis and design of behavior, structure, layout, and deployment. By following the TopUML modeling activities one by one, the system gets designed in top-down way starting with formalization of problem domain and ending with deployment planning of designed components, thus it is ensured that all developed artifacts are defined in accordance with characteristics of problem domain functioning and that causal trace links exist between artifacts of both problem and solution domains. The benefit of such formalized modeling method application within software development process is that the solution is prepared according to the properties of problem domain and its functioning and that changes in the functioning of problem domain can be formally evaluated in solution domain (and vice versa).

The comparison of TopUML modeling with other UML modeling driven methods shows evaluation of a set of criterions which are divided into following four groups: analysis and design models, problem domain analysis and design, requirements management, and usage. Since TopUML modeling and TFMfMDA both are based on TFM, several characteristics of them are equal or similar. The TopUML modeling solves one of the weakest points of the TFMfMDA approach – assignment of responsibilities to appropriate classes. TFMfMDA uses a powerful tool to analyze functioning of the problem domain – the TFM, but the TFMfMDA lacks the ability to transfer responsibilities of objects from TFM to the classes. TopUML modeling solves this issue by transferring system's functioning information from TFM to other diagrams.

In summary, proposed TopUML profile and modeling method together solves identified disadvantages of UML and its application within software development.

5. IMPLEMENTATION AND APPROBATION OF TOPUML

The implementation and approbation of TopUML language and modeling is shown and discussed in the context of two software designing projects:

1. *Business support application development* – shows a practical experiment in which the software is designed for a laundry problem domain, and
2. *Enterprise data synchronization system development* – covers a case study of software development project in which software is developed to perform enterprise data synchronization taking data from multiple data sources and placing into centralized data storage.

A case study is considered as an observational study in which data is collected for a specific purpose throughout the study, and an experiment is a formal and controlled investigation [26]. Each of the discussed projects consumes slightly different parts of TopUML modeling, e.g., the experiment uses system goals to set scopes of Sequence diagrams while the case study uses Use cases.

Additionally this chapter includes empirical evaluation of TopUML profile and modeling method provided by two expert groups participating in the experiment of business support application development [17]. The modeling and development knowledge of experts before the experiment is determined by using self-evaluation questionnaire thus showing preliminary knowledge of each participant.

5.1. Business Support Application Development

Business support application development is demonstrated on the basis of practical experiment in which a software design for laundry functioning at the platform independent viewpoint is developed. The laundry business system is an experimental system created to demonstrate the capabilities of TopUML profile and modeling method. TopUML modeling Experimental software designing includes creation of artifacts according to the TopUML modeling method:

1. *Initial* and *refined TFM* – developed in accordance with informal system description and defined functional requirements,
2. *Sequence diagrams* and *Interaction overview diagram* – prepared by transforming TFM, scope of each Sequence diagram is denoted by system goals,
3. *Communication diagram* – TFM transformation provides actors, objects, and links between them, as well as messages sent from object to object and their order, and
4. *Topological class diagram* – obtained by performing transformations on TFM and Communication diagram, thus defining classes, their responsibilities, and topological relationships between them.

All the artifacts are created in accordance with the functioning properties of problem domain. The application of TFM to formalize functioning of problem domain and the transfer of TFM characteristics to other TopUML diagrams ensures that it is possible to clearly trace

developed artifacts in both problem and solution domains. The used theory within experiment and obtained results are published in [21] and [69], as well as in guidance manual [17].

5.2. Enterprise Data Synchronization System Development

The main idea of this section is to explore a case study of applying TopUML modeling in a real software project in which a service application is developed for synchronizing enterprise data. Synchronization is done by taking data from multiple data sources and placing in one data storage. The case study covers full software development life cycle (the software now is at maintenance phase, the case study covers design and implementation phase). The software is developed at Lattelecom Technology Ltd. Software Development Department. TopUML modeling case study includes development of the following TopUML diagrams:

1. *Initial* and *refined TFM* – developed in accordance with informal system description and defined functional requirements,
2. *Topological use case diagram* – defined in accordance with developed TFM and mappings between functional features of TFM and determined functional requirements,
3. *Sequence diagrams* and *Activity diagrams* – prepared by transforming TFM, scope of each diagram is denoted by use cases (an example of TFM transformation to Activity diagram is given in Figure 5.1),
4. *Communication diagram* – obtained by performing transformations on TFM which provides actors, objects, links between them, and messages together with their order,
5. *Initial* and *refined Topological class diagram* – the initial diagram is defined by transforming TFM and Communication diagram, while the refined diagrams is obtained by performing additional problem domain analysis in accordance with refinement steps given in TopUML modeling method (the refinement result is addition of associations, generalizations, and dependencies between classes, definition of required and provided interfaces according to inputs and outputs of TFM), and
6. *State diagram* – within the case study a State diagram is prepared for the main object of data synchronization system by applying transformations on TFM. The main object is determined by its membership to the main functioning cycle.

Due to the TopUML modeling method application within enterprise data synchronization system development all created artifacts are traceable in both problem and solution domains. Thus, changes in the functioning of problem domain can be formally and precisely evaluated in solution domain (and vice versa). The main results of case study exploration are published in [19], as well as the new theoretical insights in [15] and [20].

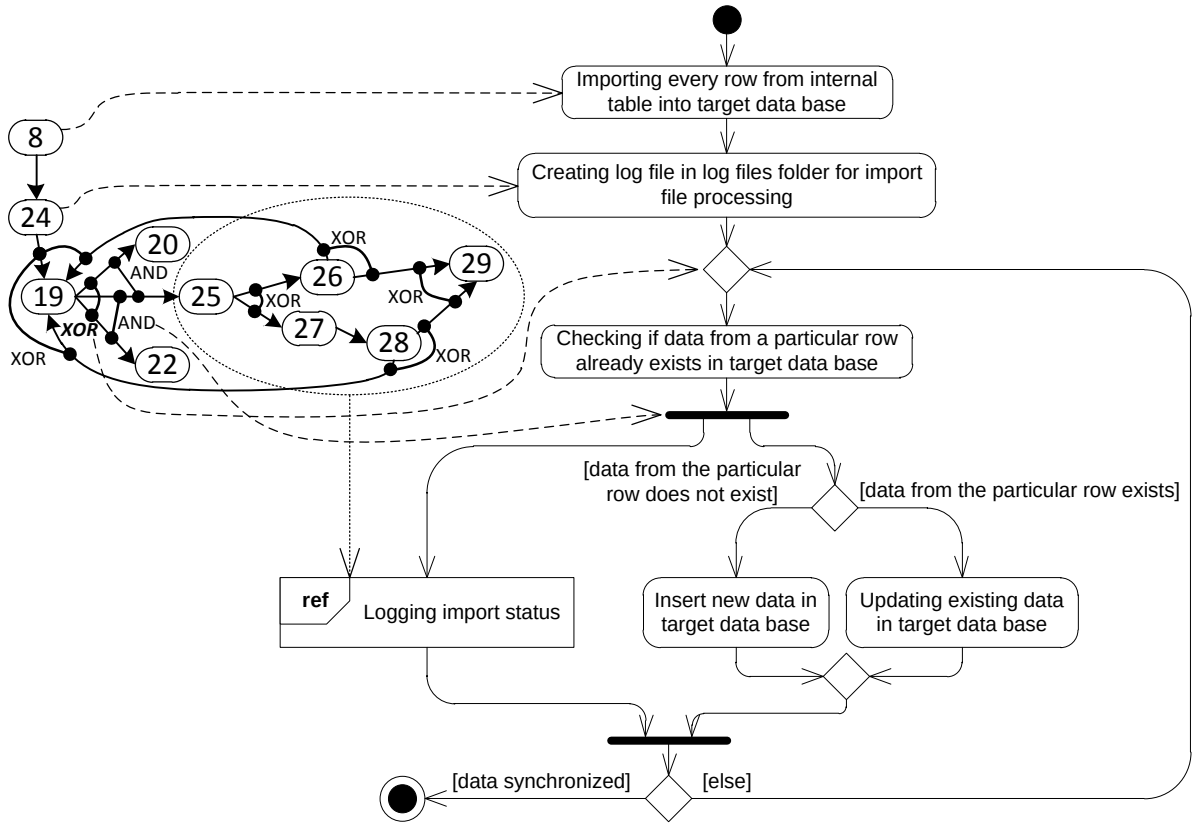

Figure 5.1. Part of TFM representing functioning of enterprise data synchronization system and activity diagram representing workflow of Use Case "Importing data in target data base"

5.3. Empirical Evaluation of TopUML Profile and Modeling Method

The empirical evaluation of TopUML profile and modeling method is based on practical experiment with two expert groups in which a business support application for the laundry problem domain is designed. The work with expert groups includes following aspects:

1. *Goal and process of empirical experiment* – the goal is development of software design by using TopUML modeling method in order to verify possibilities of learning it and its usability. The design process is divided into eight workshops. Expected results are defined for each workshop.
2. *Participants* – total count of participants in both groups is 32 (15 participants are holding bachelor degree in computer science and 17 – master degree). Each participant has background of previous experience in software development by performing different roles (analyst, programmer, tester, and project manager).
3. *Result of experiment* – a number of system designs are prepared thus allowing to evaluate the benefits and advantages of applying formal modeling method and models – the differences between constructed models are minimal. In addition, an empirical evaluation of proposed profile and modeling method is performed.
4. *Positive and negative aspects of TopUML modeling* – one of the experiment results is evaluation by its participants of advantages (theoretical foundations, formal modeling activities, transformations between models, reduced risk of

rewriting software code) and disadvantages (lack of supporting tool) of the proposed TopUML modeling method application.

The main result of practical experiments is preparation and publication of guidance manual "Topological business systems modeling and software systems design" [17]. The guidance manual includes both – the TopUML modeling theory and a practical example of applying it within software designing for laundry problem domain.

5.4. Summary

The approbation of proposed TopUML profile and modeling method includes elaboration of two different projects: experimental software design creation for laundry problem domain and a case study exploration for a real software development project in which software for enterprise data synchronization has been developed. During approbation it is demonstrated how analysis and design artifacts are developed in a strong accordance with the functioning properties of problem domain by applying defined modeling activities and using TopUML profile. In the same time the causal relationships are retained in both problem and solution domain artifacts, as well as between them. The result of applying proposed profile and modeling method is software that complies with the functioning of the problem domain and its characteristics, and causal relationships that facilitate further maintenance and development of developed software. Due to the identified traceability links it is possible to formally and precisely evaluate problem domain functioning changes in solution domain (and vice versa).

The proposed modeling method deals with the Computation independent viewpoint and the Platform independent viewpoint within MDA. The TopUML modeling is concerned with the "*what*" and not "*how*". The "*what*" means what is what and what should do what (i.e. identification of classes and their responsibilities) while the "*how*" means how the responsibility will be implemented in a specific platform or technology (e.g. how the data will be saved in data base).

RESULTS AND CONCLUSIONS

The goal of the doctoral thesis was to supplement UML with theoretical foundations in order to create grounds for converting notation into a formal modeling language and to define modeling method which allows to clearly trace cause-and-effect relationships in both problem and solution domains. The main result of the work is TFM supplementation with logical relations, specification of TopUML profile and definition of modeling method which is used to put into practice the created profile. All of the specified tasks for achieving the goal of thesis are completed and the following results and conclusions are obtained:

1. Results of analyzing UML, its specification and application in software development are as follows:
    a. Despite of the benefits gained by using UML within software development process, analysis of its specification and application shows a number of disadvantages and limitations,

   b. Although that language extension mechanisms are provided starting with the UML version 2.0, the development of profiles is a difficult task while the UML specification is a specification of a notation and thus it defines only elements of language, their notation and semantics,
   c. By analyzing a number of existing UML profiles an opinion is established that the most suitable way for specifying an UML profile is by using the same specification structure as used in UML specification,
   d. Modeling methods determine the application of UML within software development process and not the UML itself (review of UML application in industry and UML modeling methods review shows that the top five most applied UML diagrams are the same), thus a part of UML disadvantages and limitation can be solved by using an appropriate modeling method, and
   e. The fragmental application of UML diagrams and partial software development lifecycle coverage within analyzed methods do not eliminate disadvantages of UML at a sufficient level.
2. Above mentioned results of performed analysis lead to the conclusions that the following should be completed to achieve the goal of thesis:
   a. Supplement UML with the theoretical foundation, thus developing specification of TopUML profile, and
   b. Define effective and usable modeling method for TopUML profile application within software development.
3. TFM supplementation with logical relations gives adequate base for transformation of TFM into other diagrams with complex structure.
4. By adding formalism of TFM mathematical topology to the UML, a modeling language specification is obtained which creates grounds for converting notation into a formal modeling language and which contains sufficient language elements to clearly identify, specify and trace cause-and-effect relationships.
5. The proposed modeling method includes the following aspects: proper analysis of problem and solution domains, application of most of the TopUML diagrams, it covers most of software development lifecycle; thus the identified UML disadvantages (size, complexity, incoherence and different interpretations) are reduced and even eliminated.
6. Application of TFM as a root model to synthesize other diagrams ensures that in a formal way are achieved following results:
   a. All artifacts created for solution domain are in conformance with the functioning characteristics of problem domain,
   b. It is possible to clearly trace cause-and-effect relationships within and between developed artifacts at the same and different abstraction levels,
   c. The developed artifacts are with high cohesion, and
   d. Components of developed system have low coupling with the rest of the system and a well-defined interface.
7. Developed TopUML profile and its modeling method have been successfully applied in an experimental software analysis and design project, as well as in a real software development project.

8. Approbation of proposed profile and modeling method shows that TopUML modeling deals with the Computation independent viewpoint and the Platform independent viewpoint within MDA.
9. Performed approbation of TopUML profile and modeling method together with two independent expert groups in a practical software design experiment gives an empirical evaluation for the proposed modeling language and method. The following advantaged are outlined: theoretical foundations, formal modeling activities, transformations between models. Previously mentioned advantages together reduce the risk of rewriting software code.

The future research directions are as follows:

- Development of a tool supporting the TopUML profile. Since the TopUML is developed as a profile of UML version 2.4.1, it can be introduced to any tool that supports its extensions with UML profiles thus eliminating the need of a completely new tool creation.
- Research on the application of TopUML for platform independent viewpoint transformation into platform specific viewpoint and generating software code.

BIBLIOGRAPHY

[1] Alksnis G. Formal Specification Languages and Category Theory Within the Framework of MDA// Computer Science, Applied Computer Systems, Vol.26, Nr.5, Scientific Proceedings of Riga Technical University, Riga, Latvia: RTU Publishing, 2006. - pp. 33-41

[2] Ambler S. Elements of UML 2.0 Style. - New York, USA: Cambridge University Press, 2005. - 200 p.

[3] Arlow J., Neustadt I. UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design. - Upper Saddle River, NJ, USA: Addison-Wesley, 2nd ed., 2005. - 624 p.

[4] Asnina E. Formalization of Problem Domain Modeling within Model Driven Architecture. Doctoral thesis. - Riga, Latvia: RTU Publishing house, 2006. - 195 p.

[5] 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), Přerov, Czech Republic: Jan Štefan MARQ, 2006. - pp. 97-104

[6] Asnina E., Gulbis B., Osis J., Alksnis G., Donins U., Slihte A. Backward Requirements Traceability within the Topology-based Model Driven Software Development// Proceedings of the 3rd International Workshop on Model-Driven Architecture and Modeling-Driven Software Development. - Beijing, China: SciTePress, 2011. - pp. 36-45

[7] Batra D., Satzinger J. Contemporary Approaches and Techniques for the Systems Analyst// Journal of Information Systems Education. - 2006. - 17(3) - pp. 257–265

[8] Booch G. Object Oriented Analysis and Design with Applications. - Upper Saddle River, NJ, USA: Addison-Wesley, 2nd ed., 1993. - 608 p.

[9] Booch G., Maksimchuk R., Engel M., Young B., Conallen J., Houston K. Object-oriented analysis and design with applications. - Upper Saddle River, NJ, USA: Addison-Wesley, 3rd ed., 2007. - 720 p.

[10] Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language User Guide. - Upper Saddle River, NJ, USA: Addison-Wesley, 2nd ed., 2005. - 475 p.

[11] Breu R., Hinkel U., Hofmann C., Klein C., Paech B., Rumpe B., Thurner V. Towards a Formalization of the Unified Modeling Language // ECOOP'97 – Object-Oriented Programming, 11th European Conference (Lecture Notes in Computer Science, Vol. 1241). – Berlin, Germany: Springer, 1997. - pp. 344-366

[12] Burton-Jones A., Meso P. Conceptualizing Systems for Understanding: An Empirical Test of Decomposition Principles in Object-Oriented Analysis// Information Systems Research. - 2006. - 17(1) - pp. 38-60

[13] DeLoach S., Hartrum T. A Theory-Based Representation for Object-Oriented Domain Models// IEEE Transactions on Software Engineering. - 2000. - Volume 26, Issue 6. - pp. 500-517

[14] Dobing B., Parsons J. Dimensions of UML Diagram Use: Practitioner Survey and Research Agenda// Principle Advancements in Database Management Technologies: New Applications and Frameworks. – Hershey, New York, USA: Information Science Reference, 2010. - pp. 271-290

[15] Donins U. Semantics of Logical Relations in Topological Functioning Model// Proceedings of the 7th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2012) – 2012. (To be published)

[16] Donins U. Software Development with the Emphasis on Topology// Advances in Databases and Information Systems (Lecture Notes in Computer Science, Vol.5968). - Berlin, Germany: Springer-Verlag, 2010. - pp. 220-228

[17] Doniņš U. Topological business systems modeling and software systems design. - Riga, Latvia: RTU Publishing house, 2011. - 65 p. (in Latvian)

[18] Donins U., Osis J. Reconciling Software Requirements and Architectures within MDA// Scientific Proceedings of Riga Technical University, Computer Science (Series 5), Applied Computer Systems (Vol. 38). - Riga, Latvia: RTU Publishing house, 2009. - pp. 84-95

[19] Donins U., Osis J. Topological Modeling for Enterprise Data Synchronization System: A Case Study of Topological Model-Driven Software Development// Proceedings of the 13th International Conference on Enterprise Information Systems, Volume 3. - Beijing, China: SciTePress, 2011. - pp. 87-96

[20] Donins U., Osis J., Asnina E., Jansone A. Formal Analysis of Objects State Changes and Transitions// Proceedings of the 7th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2012) – 2012. (To be published)

[21] Donins U., Osis J., Slihte A., Asnina E., Gulbis B. Towards the Refinement of Topological Class Diagram as a Platform Independent Model// Proceedings of the 3rd International Workshop on Model-Driven Architecture and Modeling-Driven Software Development. - Beijing, China: SciTePress, 2011. - pp. 79-88

[22] Erickson J., Siau K. Theoretical and Practical Complexity of Modeling Methods// Communications of the ACM. - 2007. - 50(8) - pp. 46-51

[23] Evans A., Kent S. Core Meta-Modelling Semantics of UML: The pUML Approach// «UML»'99: The Unified Modeling Language. Beyond the Standard (Lecture Notes in Computer Science, Vol. 1723). - Berlin, Germany: Springer, 1999. - pp. 140-155

[24] Evermann J., Wand Y. Ontological Modeling Rules For UML: An Empirical Assessment// Journal of Computer Information Systems. - 2006. - 46(5) - pp. 14-29

[25] Evermann J., Wand Y. Towards Ontologically-Based Semantics for UML Constructs// Conceptual Modeling – ER 2001, 20th International Conference on Conceptual Modeling (Lecture Notes in Computer Science, Vol. 2224). - Berlin, Germany: Springer, 2001. - pp. 341-354

[26] Fenton N., Pfleeger S. Software Metrics: A Rigorous and Practical Approach. - Scottsdale, Arizona, USA: Coriolis Group, 2nd ed., 1996. – 649 p.

[27] Fowler M. Why use the UML?// Software Development. - 1998. - Volume 6, Issue 3

[28] Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. - Upper Saddle River, NJ, USA: Addison-Wesley, 3rd ed., 2003. - 208 p.

[29] Grundspenkis J. Fault Localisation Based on Topological Feature Analysis of Complex System Model// Diagnostics and Identification. - Riga: Zinatne, 1974. - pp. 38-48 (in Russian)

[30] Grundspenkis J. Structural Modelling of Complex Technical Systems in Conditions of Incomplete Information: A Review// Modern Aspects of Management Science. - Riga, Latvia: RTU Publishing house, 1997. - pp. 111-136

[31] Grundspenkis J. Structural Modelling with ASMOS in the Early Stages of Design// Software for Manufacturing. – Amsterdam, Holland: North-Holland Publishing Company, 1989. - pp. 229-239

[32] Grundspenkis J. The Synthesis and Analysis of Structure in Computer Aided Design// Computer Applications in Production and Engineering: Proceedings of the First International Conference. – Amsterdam, Holland: North-Holland Publishing Company, 1983. - pp. 301-316

[33] Grundspenkis J., Blumbergs A. Investigation of Complex System Topological Model Structure for Analysis of Failures// Issues of Technical Diagnosis. - Rostov-on-Don, Russia: Rostov Institute of Building Engineering, 1981. - pp. 41-48 (in Russian)

[34] He X. Formalizing UML Semantics// 25th Annual International Computer Software and Applications Conference (COMPSAC'01). - Chicago, Illinois, USA: IEEE Computer Society, 2001. - pp. 277

[35] International Organization for Standardization (ISO): ISO/IEC/IEEE 42010:2011 "Systems and software engineering -- Architecture description", 2011. - 37 p.

[36] Jacobson I., Christerson M., Jonsson P., Overgaard G. Object-Oriented Software Engineering: A Use Case Driven Approach. - Upper Saddle River, NJ, USA: Addison-Wesley, 1992. - 552 p.

[37] Jones C. Positive and Negative Innovations in Software Engineering// International Journal of Software Science and Computational Intelligence. - 2009. - Volume 1, Issue 2. - pp. 20-30

[38] Karpics I., Markovics Z. Development and Evaluation of Normal Performance Recovery Method of a Functional System // Proceedings of 9th IEEE International Symposium on Applied Machine Intelligence and Informatics (SAMI), 2011, Slovakia, Smolenice, 2011. - pp. 171-175

[39] Kent S. The Unified Modeling Language// Formal Methods for Distributed Processing: A Survey of Object-Oriented Approaches. - Cambridge, England: Cambridge University Press, 2001. - pp. 126-151

[40] Kim S., Carrington D. Formalizing the UML Class Diagram Using Object-Z// «UML»'99: The Unified Modeling Language. Beyond the Standard (Lecture Notes in Computer Science, Vol. 1723). - Berlin, Germany: Springer, 1999. - pp. 83-98

[41] Kleppe A., Warmer J., Bust W. MDA Explained. The Model Driven Architecture: Practice and Promise. - Upper Saddle River, NJ, USA: Addison-Wesley, 2003. - 192 p.

[42] Kobryn C. UML 2001: A Standardization Odyssey// Communications of the ACM. - 1999. - Volume 42, Issue 10. - pp. 29-37

[43] Lano K., Kolahdouz-Rahimi S. Model-Driven Development of Model Transformations// Theory and Practice of Model Transformations (Lecture Notes in Computer Science, Volume 6707). - Berlin, Germany: Springer-Verlag, 2011. - pp. 47-61

[44] Larman C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. - Upper Saddle River, NJ, USA: Prentice Hall, 3rd ed., 2005. - 736 p.

[45] Lazar I., Motogna S., Parv B., Lazar C. Realizing Use Cases for Full Code Generation in the Context of fUML// Proceedings of the 2nd International Workshop on Model-Driven Architecture and Modeling Theory-Driven Development - Portugal: SciTePress, 2010. - pp. 80-89

[46] Li D., Li X., Stolz V. QVT-Based Model Transformation using XSLT// ACM SIGSOFT Software Engineering Notes. - 2011. - Volume 36, Issue 1. - pp. 1-8

[47] Loton T. UML Software Design with Visual Studio 2010. – Breinigsville, PA, USA: LOTONtech Limited, 2010. - 136 p.

[48] Markovica I., Markovics Z. Mathematical Model of Pathogenesis of Hard Differentiable Diseases// Cybernetics and Diagnostics, Volume 4. - Riga: Zinatne, 1970. - pp. 21-28 (in Russian)

[49] Mellor S., Balcer M. Executable UML: A Foundation for Model-Driven Architecture. - Upper Saddle River, NJ, USA: Addison-Wesley, 2002. - 416 p.

[50] Mens T., Van Gorp P. A Taxonomy of Model Transformation// Electronic Notes in Theoretical Computer Science. - 2006. - Volume 152. - pp. 125-142

[51] Miller J., Mukerji J. (editors): MDA Guide Version 1.0.1 / Internet. - http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf

[52] Nielsen M., Havelund K., Wagner K., George C. The RAISE Language, Method and Tools// Formal Aspects of Computing. - 1989. - Volume 1 Issue 1. - pp 85-114

[53] Ņikiforova O. System Modeling in UML with Two-Hemisphere Model Driven Approach// Scientific Proceedings of Riga Technical University, Computer Science (Series 5), Applied Computer Systems (Volume 43). - Riga, Latvia: RTU Publishing house, 2010. - pp. 37-44

[54] OMG: Meta Object Facility (MOF) Core Specification Version 2.0 / Internet. - http://www.omg.org/spec/MOF/2.0/PDF/

[55] OMG: Service Oriented Architecture Modeling Language (SoaML) / Internet. - http://www.omg.org/spec/SoaML/1.0/Beta2/PDF

[56] OMG: Unified Modeling Language Infrastructure Version 2.4.1 / Internet. - http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF/

[57] OMG: Unified Modeling Language Superstructure Version 2.4.1 / Internet. - http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF/

[58] OMG: OMG Systems Modeling Language (OMG SysML) / Internet. - http://www.omg.org/spec/SysML/1.2/PDF

[59] Olive A. Conceptual Modeling of Information Systems. - Berlin, Germany: Springer, 2007. - 455 p.

[60] 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, 2003. - pp. 305-310

[61] Osis J. Formal Computation Independent Model within the MDA Life Cycle// International Transactions on Systems Science and Applications. - 2006. - Volume 1, Number 2. - pp. 159-166

[62] Osis J. Mathematical Description of Complex System Functioning// Cybernetic and Diagnosis, Volume 4. - Riga: Zinatne, 1970. - pp. 7-14 (in Russian)

[63] Osis J. The Topological Model of System Functioning// Automatics and Computer Science, Volume 6. - Riga, Latvia, 1969. - pp. 44-50 (in Russian)

[64] Osis J., Asnina E. Enterprise Modeling for Information System Development within MDA// Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008). - Chicago, Illinois, USA: IEEE Computer Society, 2008. - pp. 491

[65] Osis J., Asnina E. Model-Driven Domain Analysis and Software Development: Architectures and Functions. – Hershey, New York, USA: IGI Global, 2011. - 487 p.

[66] Osis J., Donins U. An Innovative Model Driven Formalization of the Class Diagrams// Proceedings of the 4th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2009). – Portugal: INSTICC Press, 2009. - pp. 134-145

[67] Osis J., Donins U. Formalization of the UML Class Diagrams// Evaluation of Novel Approaches to Software Engineering (Communications in Computer and Information Science (CCIS), Volume 69). - Berlin, Germany: Springer-Verlag, 2010. - pp. 180-192

[68] Osis J, Donins U. Modeling Formalization of MDA Software Development at the Very Beginning of Life Cycle// Advances in Databases and Information Systems. 13th East-European Conference, ADBIS 2009: Associated Workshops and Doctoral Consortium, Local Proceedings. - Riga, Latvia: JUMI Publishing House Ltd., 2009. - pp. 48-61

[69] Osis J., Donins U. Platform Independent model Development by Means of Topological Class Diagrams// Proceedings of the 2nd International Workshop on Model-Driven Architecture and Modeling Theory-Driven Development - Portugal: SciTePress, 2010. - pp. 13-22

[70] Osis J., Gefandbein J., Markovitch Z., Novozhilova N. Diagnosis based on graph models. (By the Examples of Aircraft and Automobile Mechanisms). - Moscow, Russia: Transport, 1991. - 244 p. (in Russian)

[71] Osis J., Silins J. Topological Function-Architecture Co-Design of Embedded Systems// Advances in Databases and Information Systems. 13th East-European Conference, ADBIS 2009: Associated Workshops and Doctoral Consortium, Local Proceedings. - Riga, Latvia: JUMI Publishing House Ltd., 2009. - pp. 424-431

[72] Osis J., Šlihte A. Transforming Textual Use Cases to a Computation Independent Model// Proceedings of the 2nd International Workshop on Model-Driven Architecture and Modeling Theory-Driven Development - Portugal: SciTePress, 2010. - pp. 33-42

[73] 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. - pp. 496-501

[74] Owre S., Rushby J., Shankar N. PVS: A Prototype Verification System// 11th International Conference on Automated Deduction (Lecture Notes in Artificial Intelligence, Volume 607). -Berlin, Germany: Springer, 1992. - pp. 748-752

[75] Pardillo J. A Systematic Review on the Definition of UML Profiles// Model Driven Engineering Languages and Systems (Lecture Notes in Computer Science, Volume 6394). - Berlin, Germany: Springer-Verlag, 2010, - pp. 407-422

[76] Podeswa H. UML for the IT Business Analyst. - Boston, MA, USA: Course Technology PTR, 2nd ed., 2009. - 372 p.

[77] Rumbaugh J., Blaha M., Premerlani W., Eddy F. Lorensen W. Object-Oriented Modeling and Design. - Englewood Cliffs, NJ: Prentice Hall, 1991. - 528 p.

[78] Rumbaugh J., Jacobson I., Booch G. The Unified Modeling Language Reference Manual. - Upper Saddle River, NJ, USA: Addison-Wesley, 2nd ed., 2004. - 721 p.

[79] Scott K. The Unified Process Explained. - Upper Saddle River, NJ, USA: Addison-Wesley, 2001. - 208 p.

[80] Sejans, J., Nikiforova, N. Practical Experiments with Code Generation from the UML Class Diagram// Proceedings of the 3rd International Workshop on Model-Driven

Architecture and Modeling-Driven Software Development. - Beijing, China: SciTePress, 2011. - pp. 57-67

[81] Siau K., Cao Q. Unified Modeling Language (UML) - a Complexity Analysis// Journal of Database Management. - 2001. - Volume 12, Issue 1. - pp. 26-34

[82] Siau K., Cao, Q. How Complex Is the Unified Modeling Language?// Advanced Topics in Database Research. - 2002. - Volume 1. - pp. 294-306

[83] Siau K., Loo P. Identifying Difficulties in Learning UML// Information Systems Management. - 2006. -Volume 23, Issue 3. - pp. 43-51

[84] Simons A., Graham I. 37 Things that Don't Work in Object-Oriented Modeling with UML// Proceedings of ECOOP 98 Workshop on Precise Behavioral Semantics. - Universitat Muchen, 1998. - pp. 209-232

[85] Slihte A., Osis J., Donins U. Knowledge Integration for Domain Modeling// Proceedings of the 3rd International Workshop on Model-Driven Architecture and Modeling-Driven Software Development. - Beijing, China: SciTePress, 2011. - pp. 46-56

[86] Slihte A., Osis J., Donins U., Asnina, E., Gulbis, B. Advancements of the Topological Functioning Model for Model Driven Architecture Approach// Proceedings of the 3rd International Workshop on Model-Driven Architecture and Modeling-Driven Software Development. - Beijing, China: SciTePress, 2011. - pp. 91-100

[87] Spivey J. The Z Notation: A Reference Manual. - Prentice Hall, 2nd ed., 1992. - 150 p.

[88] Stevens P., Pooley R. Using UML: Software Engineering with Objects and Components. - Harlow, England: Addison-Wesley, 2nd ed., 2005. - 250 p.

[89] Szlenk M. UML Static Models in Formal Approach// Balancing Agility and Formalism in Software Engineering (Lecture Notes in Computer Science, Volume 5082). - Berlin, Germany: Springer-Verlag, 2008. - pp. 129-142

[90] Turner M. Microsoft Solutions Framework Essentials: Building Successful Technology Solutions. - Redmond, Washington, USA: Microsoft Press, 2006. - 300 p.

[91] Warmer J., Kleppe A. The Object Constraint Language: Getting Your Models Ready for MDA. - Upper Saddle River, NJ, USA: Addison-Wesley, 2nd ed., 2003. - 240 p.

[92] Xueming L., Parsons J. Ontological Semantics for the Use of UML in Conceptual Modeling// ER (Tutorials, Posters, Panels & Industrial Contributions). - 2007. - pp. 179-184

[93] Zhao Y., Zong-Yuan Y., Xie J. Pi-Calculus Based Assembly Mechanism of UML State Diagram and Validation of Model Refinement// Proceedings of International Conference on Electronic Computer Technology 2009 (ICECT 2009). - Macau, China, 2009. - pp 604-609