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

Armands ŠLIHTE
Student of the Doctoral Study Program "Computer Systems"

THE INTEGRATED DOMAIN MODELING: AN APPROACH & TOOLSET FOR ACQUIRING A TOPOLOGICAL FUNCTIONING MODEL

Summary of Doctoral Thesis

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

RTU Press
Riga 2015

Šlihte A. The Integrated Domain Modeling: An Approach and Toolset for Acquiring a Topological Functioning Model. Summary of Doctoral Thesis.-R.: RTU Press, 2015.-40 p.

Printed in accordance with the Resolution of the Council of the Institute of Applied Computer Systems, Faculty of Computer Science and Information Technology, Riga Technical University as of 30 June 2014, Minutes No. 12300-4.1/2

The research has been supported by the European Social Fund within the project "Support for the Implementation of Doctoral Studies at Riga Technical University".

The research is supported in part by the Latvian National Research Program SOPHIS under grant agreement No.10-4/VPP-4/11

ISBN 978-9934-8260-5-4

DOCTORAL THESIS HAS BEEN PROPOSED TO RIGA TECHNICAL UNIVERSITY FOR THE PROMOTION TO THE SCIENTIFIC DEGREE OF DOCTOR OF ENGINEERING SCIENCES

To be granted the scientific degree of Doctor of Engineering Sciences, the present Doctoral Thesis will be publicly defended on 8 June 2015 at Riga Technical University, the Faculty of Computer Science and Information Technology, Institute of Applied Computer Systems, Meža Street 1/3, Room 202.

REVIEWERS

Professor, Dr.sc.ing. Mārīte Kirikova
Riga Technical University, Latvia

Professor, Dr.sc.ing. Artis Teilāns
Rezekne Higher Education Institution, Latvia

Associate professor Vincente García Díaz
University of Oviedo, Spain

DECLARATION OF ACADEMIC INTEGRITY

I hereby declare that the Doctoral Thesis submitted for the review to Riga Technical University for the promotion to the scientific degree of Doctor of Engineering Sciences, is my own and does not contain any unacknowledged material from any source. I confirm that this Thesis has not been submitted to any other university for the promotion to any other scientific degree.

Armands Šlihte ……. (Signature)

Date: ……

The Doctoral Thesis has been written in English. It consists of an introduction, 4 chapters, conclusions, bibliography with 100 reference sources, and 10 appendices. To illustrate the contents of the research, 52 figures have been used. The volume of the present Doctoral Thesis is 216 pages.

TABLE OF CONTENTS

INTRODUCTION

In the context of software engineering, a domain is most often understood as an application area, a field for which software systems are developed [74]. A domain model can be used as input for implementation of the solution within a software development process. The model elements that compose the domain model can serve as a basis for code construction, which can be done manually or by using automated code generation as suggested by Model Driven Architecture (MDA) [41]. Domain model is an integrated system of models that reflect the enterprise, where software is to be applied [89]. In other words, a domain model is the AS-IS model of the business organization and processes. Moreover, domain modeling is a human activity that leads to the creation of different types of domain representations. These representations may be tacit (in human minds) and explicit/externalized (on paper or in a software tool) [36]. Furthermore, domain analysis is a process where the necessary information for developing software systems for a specific domain is identified, captured, structured, and organized for further reuse [74]. According to [58], there are many domain modeling approaches, but these are mostly based on unification of standards rather than mathematically formal models, with exception of Petri Nets and Topological Functioning Model (TFM). This is a serious issue for software engineering, because there is no formal connection between the domain model and the solution.

Motivation of the Research

In his research, Capers Jones [30] analyzes positive and negative innovations in software engineering, and concludes that the way software is built remains surprisingly primitive. His research [30], [96] has found that majority of software development projects fail because of overrun budget and schedule, or have hazardously bad quality level of the produced software. Some of the issues (which have not changed during the last 30 years) mentioned in the research are as follows: “Initial requirements are seldom more than 50% complete”, “There are more defects in requirements and design than in source code”, “Finding and fixing bugs is the most expensive software activity”. The main problem is that software development is not a well-structured discipline, requires high content of manual labor and the processes are not automated. The author of this Doctoral Thesis believes that a formal domain model is the key to software development automation, and that MDA is a positive innovation for software development. However, inability to produce a formal domain model within software engineering leads to the following problems. The scope and content of the acquired domain model cannot be

validated in an automated way (only manual, subjective validation is possible). It is not possible to transform the domain model to the solution model automatically (again only manual, subjective transformation is possible). Thus, there is no formal transformation, and it can also be problematic to formally trace the elements of the domain model to the solution model (after several iterations of manual transformation elements can be lost or incorrectly transformed). This leads to low quality of the software and an inefficient way of producing it. The ultimate goal of this research is to lower the cost and raise the quality of software development by introducing a methodology and a toolset, which would allow a comprehensive analysis of the domain in the beginning of software development based on a formal domain model, thus minimizing the number of bugs and change requests due to an inconsistent understanding of the domain.

Research Area

Model Driven Architecture (MDA) proposes software development to abstract from the code as the uppermost of the functionality of the information system to the model of the information system [23]. MDA is a software development framework, which defines 3 layers of abstraction for system analysis: Computation Independent Model (CIM), Platform Independent Model (PIM), and Platform Specific Model (PSM). CIM describes system requirements and the way system works within its environment while details of the application structure and implementation are hidden or are yet undetermined. CIM is also known as the domain model or the business model. As the business model CIM should be a precise description of the business in its environment, by the business, in the language of business people, dedicated to business purposes [5]. However, the only formal means to define a CIM is Petri nets, which are too complex for business people since they are based on "heavy" mathematics [5]. Another formal model that can be used as a CIM is the Topological Functioning Model (TFM).

The present research is a part of the Topological Functioning Model for Software Engineering (TFM4SE) research. TFM is a domain model, which offers a formal way to define a system by describing both the system's functional and topological features [4]. TFM represents the system in its business environment and shows how the system functions, without details about how the system is constructed. This research suggests using a TFM as the Computation Independent Model (CIM) likewise the Topological Functioning Model for the Model Driven Architecture (TFM4MDA) approach [53], [64], [4], [55] and [61]; acquiring a mathematically formal and, thus, transformable CIM. In the related research [12] the TopUML approach is described for software development with an emphasis on topology, where the Platform Independent

Model/Platform Specific Model (PIM/PSM) is supplemented with topology. TopUML is a UML profile and an approach for introducing cause and effect relationships into the UML based on the topology of TFM. TopUML approach suggests sequential phases of TFM4MDA approach to be combined for fulfilling the Model Drive Architecture (MDA) life cycle taking the TFM as source for PIM/PSM. Although the TFM, TFM4MDA and TopUML provide a solid basis for CIM construction within MDA and further transformations to PIM/PSM, until now the construction of the TFM relies on a heavy manual process with no tool support and a poor integration with the common IT practices. The AS-IS processes of TFM4MDA are described in [56], [58], [60] and for TopUML in [13].

Purpose of the Research

The goal of the Doctoral Thesis is to improve the process of domain analysis by providing an approach and a supporting toolset for acquiring a formal domain model that is transformable and can be used as a CIM within the MDA, thus lowering cost of software development projects, raising the quality of produced software and enabling their success by improving the understanding of the domain.

To achieve the goal set, **the following objectives** have been defined: 1) To analyze the existing domain modeling approaches and to identify their strengths, weaknesses and conformance to a CIM within MDA; 2) To evaluate the domain modeling approaches based on their formality, conformance to MDA and practical usability; 3) To analyze the TFM approach, to identify its strengths, weaknesses and points for improvement; 4) To develop the Integrated Domain Modeling (IDM) approach for acquiring a formal domain model in the form of TFM based on formal knowledge about the domain using existing IT practices; 5) To develop a toolset in order to support the IDM approach based on MDA standards, which consists of Use Case Editor, TFM Editor, and Use Cases to TFM Transformation tool; 6) To conduct a case study using the IDM approach and toolset for a real software development project.

The research subject is domain modeling with focus on the domain model within software engineering.

The research objects are MDA and TFM, focusing on how to acquire a CIM in a formal way with accordance to MDA standards.

The research methods used – analysis by comparison, metamodeling method and model transformation, design science, and case study.

Thesis statements to be defended are the following: 1) If for a particular business domain corresponding domain knowledge is formally defined, then it should be possible

to generate a formal domain model automatically; 2) The usability of TFM can be significantly improved by addressing the issues of informal description and lack of tool support with a new approach based on common practices; 3) Declarative and procedural knowledge complement each other and enable a thorough domain analysis.

Scientific Novelty and Practical Value

The **scientific novelty** of this research is a novel approach called Integrated Domain Modeling (IDM) for domain modeling, which provides a means to acquire a mathematically formal domain model in the form of TFM and at the same time uses common standards as key input for the domain modeling process and introduces model transformation. The IDM is based on the declarative an procedural aspects of domain knowledge using Ontology and Use Cases as input, and executing model transformation to a TFM using also Natural Language Processing (NLP).

The **practical value** of this research is the supporting toolset for the IDM approach that consists of the Use Cases Editor, TFM Editor and Use Cases to TFM transformation tools as well as a case study of applying the IDM approach for e-commerce software development project. Before there were no tools to support the TFM approaches. But now with the IDM approach it is possible to acquire a TFM with automatic model transformation.

Approbation of the Research Results

The main results of the research have been presented in the following 8 international scientific conferences (4 were held in Latvia and 4 in foreign countries): 1) The 11th International Baltic Conference on DB and IS (DBIS 2014), Estonia, Tallinn, June 8–11, 2014; 2) The 54th Scientific Conference of Riga Technical University, Riga, Latvia, October, 2013; 3) The 7th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2012), Poland, Wrocław, June 29–30, 2012; 4) The 3rd International Workshop on Model-Driven Architecture and Modeling-Driven Software Development (MDA & MDSD 2011), China, Beijing, June 8–11, 2011; 5) The 9th International Baltic Conference on DB and IS (DBIS 2010), Latvia, Riga, July 5–7, 2010; 6) The 2nd International Workshop on Model-Driven Architecture and Modeling Theory-Driven Development (MDA & MTDD 2010), Greece, Athens, 2010; 7) The 51st Scientific Conference of Riga Technical University, Riga, Latvia, October, 2010; 8) The 13th East-European Conference (ADBIS 2009), Latvia, Riga, September 7–10, 2009.

The main results have been published in the following 11 scientific papers:

1. Šlihte A. Introduction to Integrated Domain Modeling Toolset// Scientific Journal of RTU. Computer Science. - 2014. (to be published)
2. Šlihte A. The Integrated Domain Modeling: A Case Study// In Proceedings of the 11th International Baltic Conference, Baltic DB&IS 2014. TUT Press, 2014. - pp. 465–470. [ISBN 978-9949-23-633-6]
3. Osis J., Šlihte A., Jansone A. Using Use Cases for Domain Modeling // Proceedings of the 7th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2012). Poland, Wrocław, June 29–30, 2012. Lisbon: SciTePress, 2012. – pp. 224–231. [ISBN 9789898565136, Indexed by Thomson Reuters, Inspec, EI, DBLP]
4. Doniņš U., Osis J., Šlihte A., Asņina Ē., 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 (MDA & MDSD 2011). China, Beijing, June 8–11, 2011. Lisbon: SciTePress, 2011. – pp. 79–88. [ISBN 9789898425591, Indexed by Thomson Reuters, Inspec, EI, DBLP, ISTP]
5. Šlihte A., Osis J., Doniņš U. Knowledge Integration for Domain Modeling// Proceedings of the 3rd International Workshop on Model-Driven Architecture and Modeling-Driven Software Development (MDA & MDSD 2011). China, Beijing, June 8–11, 2011. Lisbon: SciTePress, 2011. – pp. 46-56. [ISBN 9789898425591, Indexed by Thomson Reuters, Inspec, EI, DBLP, ISTP]
6. Šlihte A., Osis J., Doniņš U., Asņina Ē., 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 (MDA & MDSD 2011). China, Beijing, June 8–11, 2011. Lisbon: SciTePress, 2011. – pp. 91–100. [ISBN 9789898425591, Indexed by Thomson Reuters, Inspec, EI, DBLP, ISTP]
7. Asņina Ē., Gulbis B., Osis J., Alksnis G., Doniņš U., Šlihte 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 (MDA & MDSD 2011). China, Beijing, June 8–11, 2011. Lisbon: SciTePress, 2011. – pp. 36–45. [ISBN 9789898425591, Indexed by Thomson Reuters, Inspec, EI, DBLP, ISTP]
8. Šlihte A. The Concept of a Topological Functioning Model Construction Tool// Advances in Databases and Information Systems: 13th East-European Conference,

ADBIS 2009. Associated Workshops and Doctoral Consortium, Local Proceedings, Latvia, Riga, September 7–10, 2009. - pp. 476–484.
9. Šlihte A. The Specific Text Analysis Tasks at the Beginning of MDA Life Cycle// Databases and Information Systems Doctoral Consortium, Latvia, Riga, July 5–7, 2010. – pp. 11–22.
10. Osis J., Šlihte A. Transforming Textual Use Cases to a Computation Independent Model// Model-Driven Architecture and Modeling Theory-Driven Development: Proceedings of the 2nd International Workshop on Model-Driven Architecture and Modeling Theory-Driven Development (MDA & MTDD 2010). Greece, Athens, July 22–24., 2010. Lisbon: SciTePress, 2010. - pp. 33–42. [ISBN 9789898425164, Indexed by SCOPUS, Thomson Reuters, Inspec, DBLP]
11. Šlihte A. Implementing a Topological Functioning Model Tool // Scientific Journal of RTU. 5 series, Computer Science. - 43. vol. - 2010. - pp. 68–75.

Outline of the Doctoral Thesis

The Thesis includes an introduction, 4 chapters, conclusions, 10 appendices, and bibliography with 100 reference sources. The volume of the Doctoral Thesis is 216 pages. It has been illustrated by 52 figures and 4 tables.

Introduction explains the motivation of the Thesis, defines the research goal and the objectives to achieve the goal, novelty and practical value of the research, its approbation and the main results. *Chapter 1* gives a definition of a domain model, domain modeling and Model Driven Architecture (MDA), analyzes some of domain modeling approaches to give an overview of the different approaches available. *Chapter 2* first defines a solid basis for the Integrated Domain Modeling (IDM) approach developed by the author of the Doctoral Thesis. Then the basic principles of the IDM approach are defined and demonstrated. *Chapter 3* introduces the supporting toolset for the IDM approach developed by the author, discussing the scope, architecture, used technologies, implementation, model transformation, and application. *Chapter 4* provides a case study of applying the IDM approach and the supporting toolset to a real software development project for a Subscription Commerce Business (SCB) in the area of e-commerce. *Conclusions* summarize the results of this Doctoral Thesis and give possible future research directions. The Thesis contains ten *appendices*: 1) List of Figures; 2) List of Tables; 3) IDM Use Case Metamodel According to Ecore; 4) IDM TFM Metamodel According to Ecore; 5) Use Case Editor Artifacts of IDM Toolset; 6) TFM Editor Artifacts of IDM Toolset; 7) TFM Diagram Tool Artifacts of IDM Toolset;

8) IDM Toolset Use Cases to TFM Transformation Tool Artifacts; 9) IDM Toolset User Guide; 10) Survey on the IDM Toolset.

1. DOMAIN MODELING AND MODEL DRIVEN ARCHITECTURE

This section gives an overview of domain modeling, analyzes some of the existing approaches and discusses the Model Driven Architecture (MDA) as an umbrella standard for all modeling. The author first gives the definition and purpose of domain modeling, then reviews literature on the different approaches available and analyzes their strengths and weaknesses.

1.1. Domain Modeling

Domain model and domain modeling are a very important part of software engineering. In this research, the author is going to use the following definition of a domain model. *Domain model is a representation of a particular problem domain or business domain from a "computation independent" perspective that exists or can exist in real life, which includes terminology, concepts, relationships, rules, and business processes.* In MDA guide [43] the Object Management Group (OMG) points out that if we built buildings the way we build software, we would be unable to connect them, change them or even redecorate them easily to fit new uses; and worse, they would constantly be falling down. A proper designing phase and a domain model are necessary to save development effort, since there are less "bugs" or inconsistencies with what was expected of the software. Another reason is to integrate and maintain the produced software after the implementation phase, which requires proper documentation. OMG mentions another benefit for having a formal domain model, i.e., it is possible to automate at least some of the construction of the software solution [43]. Software development projects do not have a positive historical track record according to [30]. Capers Jones' analysis shows that majority of software development projects fail because of overrun budgets and schedules, or hazardously bad quality levels. Software development is not a well-structured discipline and the processes are not automated in any way, as oppose to traditional engineering, like the construction example given by the OMG. The authors in [56] state that the domain model should be the cornerstone of software development – it is a design, documentation and a way of decreasing inconsistencies and over-budget costs. Another purpose of domain modeling that is rarely used separate from software engineering is a domain analysis. Usually when stakeholders talk about a domain analysis this is in context of a software solution that

could improve the business process by saving money for the company or earning more money with a better service to their customers. A domain model can provide useful insights for the company of its business processes and structure. Businesses may choose to use domain modeling to analyze their business organization, to improve their business processes without additional software, or to improve their business governance and internal documentation.

1.2. Model Driven Architecture

MDA is an approach to using models in software development proposed by the OMG [86]. OMG considers the MDA as a small step on the long road to turning software development from a craft into an engineering discipline [43]. MDA is neutral from language, vendor and middleware [35]. OMG describes different levels of abstraction and their relations; however, it does not specify how to create the models and which notation to use for representation. There are some recommendations from various researchers in the field that can be similar in certain points and different in others [33]. MDA defines 3 viewpoints and the corresponding models – Computation Independent Model (CIM), Platform Independent Model (PIM), Platform Specific Model (PSM). At a conceptual level, the MDA is a holistic approach for improving the entire IT life cycle – specification, architecture, design, development, deployment, maintenance, and integration – based on formal modeling [58]. J. Osis believes that the potential power of MDA is in model transformation, because model transformation requires the use of formal languages for description of models and this lead to increasing the role of mathematics for software development [54].

Some other standards proposed by the OMG that complement the MDA are the following. UML, the Unified Modeling Language, allows a model to be constructed, viewed, developed, and manipulated in a standard way at the analysis and design time. The Meta-Object Facility (MOF) standardizes a facility for managing models in a repository [42]. XML Metadata Interchange (XMI) is an interchange format for models, based on the MOF and the popular language XML. Query/View/Transformation (QVT) is an OMG standard for model transformations, which consists of several model transformation languages – declarative, imperative, operational mappings (QVTo) and black box [42].

1.3. Domain Modeling Approaches and Evaluation

This Doctoral Thesis gives an overview of the currently used domain modeling approaches and positions the TFM among them. The approaches that are going to be

described here are Business Process Modeling and Notation (BPMN), Event-Driven Process Chains (EPC), Topological Functioning Model (TFM), Ontology and Use Cases. Petri Net is a mathematically formal model, but it is usually hidden from users of business models. The cause is both hard mathematics and representation of domain information that is not intuitive for users [5]. For these reasons, also Petri Nets will not be discussed. Some innovative and interesting domain modeling approaches are explored to show some potential trends in domain modeling approaches, which exploit artificial intelligence – Linguistic Assistant for Domain Analysis (LIDA) [76], Natural Language Requirements Analysis (NIBA in German) [72].

There are, of course, a lot of domain modeling approaches and the author has only considered a few. However, these approaches give an indication of how the domain model can be modeled today. Domain modeling approaches will be evaluated based on 3 sets of criteria in Table 1.1: 1) criteria for a formal domain model; 2) criteria for conformance to MDA; and 3) criteria for practical usability. Formalism is a theory or a view of philosophy of mathematics as per Nicolas D. Goodman [46], which states that mathematics is a rule-governed manipulation of symbols and nothing else. Thus, this manipulation is formal. How to evaluate whether a domain model is formal? A metamodel does provide a level of formalism since it defines the rules for defining the elements of the domain model. If the metamodel complies with a meta-metamodel, this brings even more formalism. To analyze the formalism of the domain modeling approaches, the author has defined 3 criteria: a mathematically formal model, formal scope definition, and formal model validation. To measure conformance of domain modeling approaches to MDA, the author proposes the following 3 criteria: computation independence, MOF compatibility, and model transformation to PIM/PSM. Since the discussion is about domain modeling, the corresponding MDA level of abstraction is a CIM. In addition to the formal domain model and conformity to MDA, there are also practical considerations for a domain modeling approach. To analyze the practical usability of the domain modeling approaches, the author has defined 4 criteria in Table 1.1: supports declarative knowledge, supports procedural knowledge, tool support, and popular/familiar. Popularity will be measured based on the count of publications mentioning the approach in the Google Scholar database [54].

Although BPMN, EPC and NIBA do have a metamodel in general, the model itself is not a mathematically formal model because the rules to build this model are not defined by mathematics. On the other hand, a TFM is based on mathematics and, thus, is considered to be a mathematically formal model.

Table 1.1

Evaluation of Domain Modeling Approaches

Criteria	**Value Range**	**BPMN**	**EPC**	**TFM**	**Ontology**	**Use Cases**	**LIDA**	**NIBA**
Mathematically formal model	Yes/No	No	No	Yes	Yes	No	No	No
Formal scope definition	Yes/No	No	No	Yes	No	No	No	No
Formal model validation	Yes/No	No	No	Yes	Yes	No	No	No
Computation Independence	Yes/No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Compatibility to MOF	Yes/No	Yes	No	Yes	Yes	No	No	Yes
Model transformation to PIM/PSM	Yes/No	Yes	Yes	Yes	No	No	Yes	Yes
Supports declarative knowledge	Yes/No	No	Yes	No	Yes	No	No	No
Supports procedural knowledge	Yes/No	Yes	Yes	Yes	No	Yes	Yes	Yes
Tool support	Yes/No	Yes	Yes	No	Yes	Yes	Yes	Yes
Popular/Familiar	Yes/No	Yes	Yes	No	Yes	Yes	No	No
Total Score	Score from 0 to 10	**6**	**6**	**7**	**5**	**4**	**4**	**5**

Ontology is based on mathematical logic or predicate logic; thus, the author considers this also a mathematically formal model. Another example of a mathematically formal model would be Petri Nets [27]. The scope of a TFM is formally defined by the closure procedure based on the input and output analysis. Validation of the model for a TFM is done by a cycle analysis and the main rule is that there should be at least one cycle for the system to be functioning. Since Ontology is based on predicate logic, it is possible to validate it with querying the Ontology with a semantic reasoner. BPMN is fully conformant with the MDA with all 3 criteria true. Authors of [52] describe transformation from the BPM to the UML Activity Diagram. Business Process Execution Language for Web Services (WS-BPEL) is a standard for implementing business processes on top of web services [26]. There is model transformation from BPMN to BPEL, thus acquiring working software based on web

services via modeling [67]. From the analyzed approaches, Ontology and EPC support declarative knowledge. Tool support exists for all analyzed approaches except a TFM. This is a major drawback of the TFM approach. To evaluate popularity, a threshold of 1000 publications was introduced by the author to determine whether an approach is popular or not. The publication count is as follows ordered in descending order: 1) Ontology – 1,250,000; 2) Use Cases – 98,500; 3) BPMN – 7,690; 4) EPC – 2,970; 5) NIBA – 101; 6) LIDA – 60; 7) TFM – 57.

The strongest approaches from a formalism perspective are TFM and Ontology. For conformance to MDA three approaches got highest score – BPMN, TFM and NIBA. From the perspective of practical usability, EPC scored the highest with BPMN, followed by Ontology and Use Cases. However, none of the analyzed approaches reached the maximum possible score of 10. The closest to score of 10 was the TFM. TFM approach lacks in practical usability mainly because it does not have tool support, does not support declarative knowledge and also is unfamiliar. Comparing to other approaches, strong formalism is the main advantage of the TFM approach. Practical usability aspects can and should be improved in context of TFM research and this is one of the objectives of this Thesis.

1.4. Summary

To summarize, the author discusses drawbacks of the existing domain modeling approaches analyzed above. Some of the problems include the following. TFM is an uncommon approach and has a complex process of construction. There are no tools to support TFM4MDA and it also relies on the informal description of the business domain (text), which might not be there. Of course, it is always possible for the system analyst to produce the informal description, but would it make more sense to put the gathered and analyzed knowledge in a more structured way if the system analyst had a choice? The author doubts the effectiveness of producing texts about a domain from scratch just to satisfy an entrance state of a domain modeling approach. Also there is currently no tool support for constructing a TFM and helping with TFM4MDA.

2. THE INTEGRATED DOMAIN MODELING APPROACH

This chapter introduces the Integrated Domain Modeling (IDM) approach developed by the author. Use Cases and Ontology are used for the domain knowledge and Topological Functioning Model (TFM) is used as the domain model. The IDM approach provides a formal way to facilitate Ontology for software engineering by

providing means to acquire a Computation Independent Model (CIM) within a Model Driven Architecture (MDA).

2.1. Solid Basis for the Approach

The solid basis of the IDM approach is the Topological Functioning Model (TFM), TFM4MDA, and TopUML approaches. According to [5], the only formal means for definition of a domain model is Petri Nets [46] and their extensions, and also TFM. The mathematical basis of the TFM is one of its main strengths. The functional features are based on connectedness, closure, neighborhood and continuous mapping. The second strength of the TFM is inputs and outputs, which correspond to system theory [65]. Inputs and outputs play an important role for domain modeling, specifically for identifying the scope of the domain model. Moreover, input and output analysis gives the opportunity to validate the domain model from the perspective of dependencies and scope. Cycle structure within a TFM and the main functioning cycle are the third strength of TFM. From the perspective of functioning, the common thing for all systems (technical, business and biological) should be an oriented cycle (a directed closed path) [60]. Every TFM needs to have at least one directed closed loop, but usually it is even an expanded hierarchy of cycles. This TFM property enables analysis of similarities and differences among functioning systems [53]. The forth strength of the TFM is the model-to-model transformations defined as part of TFM4MDA and improved in the research on TopUML by Uldis Doniņš[13].

Researchers in a similar field have noticed the TFM approach and consider TFM to be a drive towards CIM specification for the MDA. However, they conclude that some important pre-CIM and design considerations may have not been taken into account [21]. The first weakness of the TFM is that it does not provide all aspects of a domain model. From the perspective of domain, the TFM only describes the procedural aspect and lacks the declarative aspect. The second weakness of the TFM and TFM4MDA in particular is their beginning – the informal description in a natural language. Informal description can be too unstructured to serve as a good basis for modeling and after a model is produced there is no way to trace element back to the source. The third weakness is that there is no tool support for TFM, TFM4MDA or TopUML. The forth weakness is that although there are guides and examples available, this process is heavy and distant from standard software engineering approaches.

Based on the strengths of the TFM approaches, it is clear that this is a powerful approach that brings a lot of benefits to domain modeling. However, the drawbacks mentioned in the previous section are preventing this approach from being effectively

used by system analysts other than the TFM research group. First of all, tool support is vital for any domain modeling approach today. In order to develop a tool support for TFM4MDA as described until now, one would need to deal with very complex natural language processing and the informal description. The author has considered some approaches that use NLP for domain modeling (e.g., LIDA and NIBA), but these approaches provide limited results or are still based on some structure of the text. Thus, one of the necessary improvements is an introduction of a more formal (structured) way to describe the business system functioning. Furthermore, in research [5] the authors state that transformation from the CIM-Knowledge Model to the CIM-Business Model cannot be performed automatically, because it requires human participation since information in the CIM-Knowledge Model is specified informally and used to have implicit knowledge. Thus, another necessary improvement is to acquire a formal CIM-Knowledge Model and enable the possibility to transform this model to the CIM-Business Model with a formal model transformation (automatically).

2.2. Integrating Declarative and Procedural Knowledge

Knowledge means understanding of a subject area, including concepts and facts about that subject area, as well as relations among them and mechanisms for how to combine them to solve problems in that area [25]. The domain knowledge can be considered part of the CIM, i.e., Knowledge Model [5]. The term knowledge can be used to refer to a state of knowing facts, methods, principles, techniques and so on. This common usage corresponds to what is often referred to as “know about”. Second, usage of the term “knowledge” is when it refers to understanding facts, methods, principles and techniques sufficient to apply them in the course of making things happen. This corresponds to “know how”. Cognitive psychologists and business analysts sort knowledge into two categories: declarative and procedural [73], [34]. From the perspective of a student who is learning knowledge: 1) Declarative knowledge is that the student knows or understands, for example, “Riga is the capital of Latvia” or “Book catalogue has entries of books”; 2) Procedural knowledge is that the student is able to do something, for example, he knows how to buy an airplane ticket to Riga or get a book at the library.

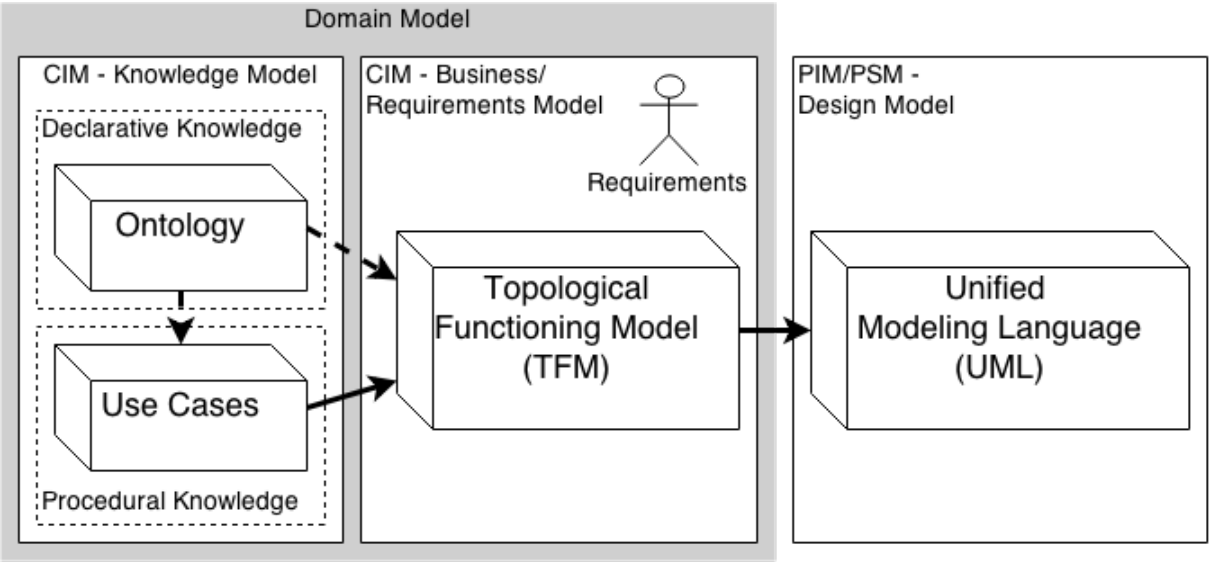

Fig. 2.1. Outline of the IDM approach.

The domain knowledge about the business system and its environment usually exists in different documents inthe form of natural language or only implicitly in human minds. It is necessary to store this knowledge in a way, so that a computer could understand it (there has to be more structure than a natural language can provide). To properly describe a business system in its environment, it is necessary to have both – declarative and procedural knowledge. In Fig. 2.1, the author shows how declarative and procedural knowledge can be integrated to acquire a CIM for the MDA. It starts with the domain knowledge defined as a formal CIM-Knowledge Model, which includes business use cases and ontology. It then provides formal transformation to the CIM-Business/Requirements Model, which is defined by the TFM. Then according to the MDA, it is possible to transform the CIM to the PIM/PSM. Detailed transformation to a UML from the TFM is described in U. Doniņš's Doctoral Thesis [13].

The Ontology standard OWL [70] is reused for the IDM approach. Ontology is a perfect candidate for representing declarative knowledge about a business system and its environment. In Fig. 2.2 on the left-hand side, the author gives an example ontology for a library domain – the class hierarchy shown. The author used Protégé [75] to develop the Ontology. In addition to a class hierarchy, also properties for the classes are defined in the Ontology. For example, there is an object property for the "Librarian" class "checkOut", which states that a "Librarian" would check out a "Book", not any other class.

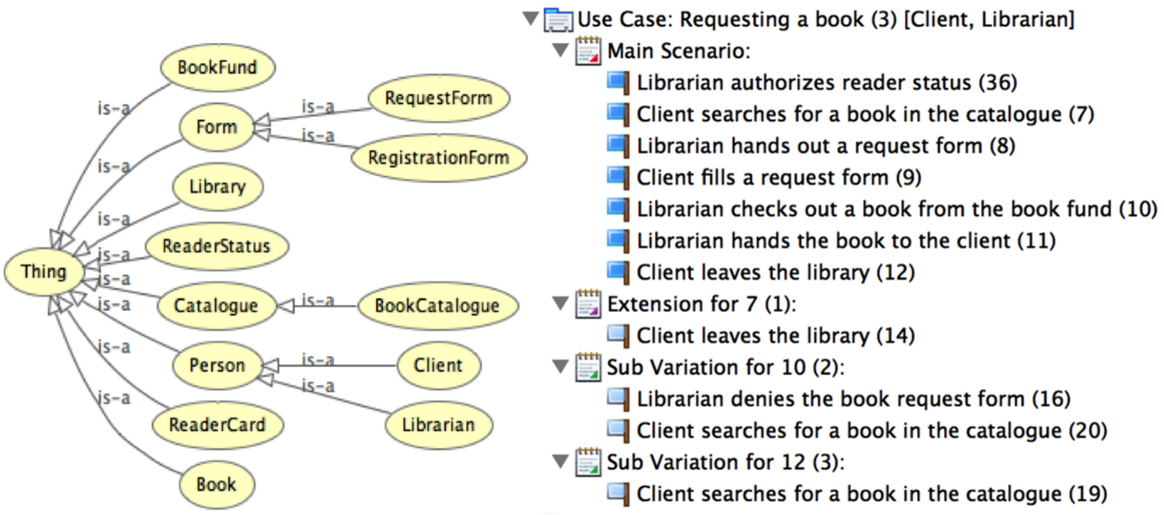

Fig. 2.2. Ontology class hierarchy and use cases for a library domain.

Use cases should not be confused with the UML Use Case diagram, and there are many different use case templates and the structure of a use case can be adjusted depending on the situation and the development team [39]. A special form of Use Cases (meta-model) is required and developed for the IDM approach. On the right hand-side in Fig. 2.2, a use case for requesting a book in a library is shown. The following structure of a use case is going to be used: 1) use case description; 2) actors; 3) main scenario; 4) extensions; 5) sub-variations.

Natural Language Processing (NLP) is applied within the IDM to enable the knowledge to be understood by a machine and, thus, to allow for a validation and transformation possibility. The IDM validates a Use Cases model against the Ontology of the business domain to ensure the unambiguity of Use Case steps as well as the correspondence to the business domain. A statistical parser [88] can be used for analyzing the sentences of use cases, and thus retrieving data for functional features to construct the TFM of the business system. This is done to help the system analyst to prevent incompleteness, ambiguity or inconsistency of use case sentences according to the Ontology.

2.3. Acquiring a Topological Functioning Model

This subchapter introduces the principles of Use Cases model transformation to the TFM model and also the use of NLP for identifying the corresponding components for the model-to-model transformation. For model transformation the NLP is used to identify the actor or entity responsible and the description of the functional feature. An example is provided of doing the transformation from Use Cases for a library domain to a corresponding TFM to demonstrate the IDM approach in action.

At this point of the IDM life cycle, there is an ontology and use cases defined for the business domain. The next goal is to acquire the TFM. To achieve this, 2 main steps are necessary: 1) to retrieve functional features; 2) to retrieve topology. First of all, the main scenario of every use case is an ordered sequence of functional features. Additionally, by analyzing the extensions and sub-variations it is possible to detect branching in the TFM. Extension adds an effect to the functional feature represented by the step referenced by the extension.

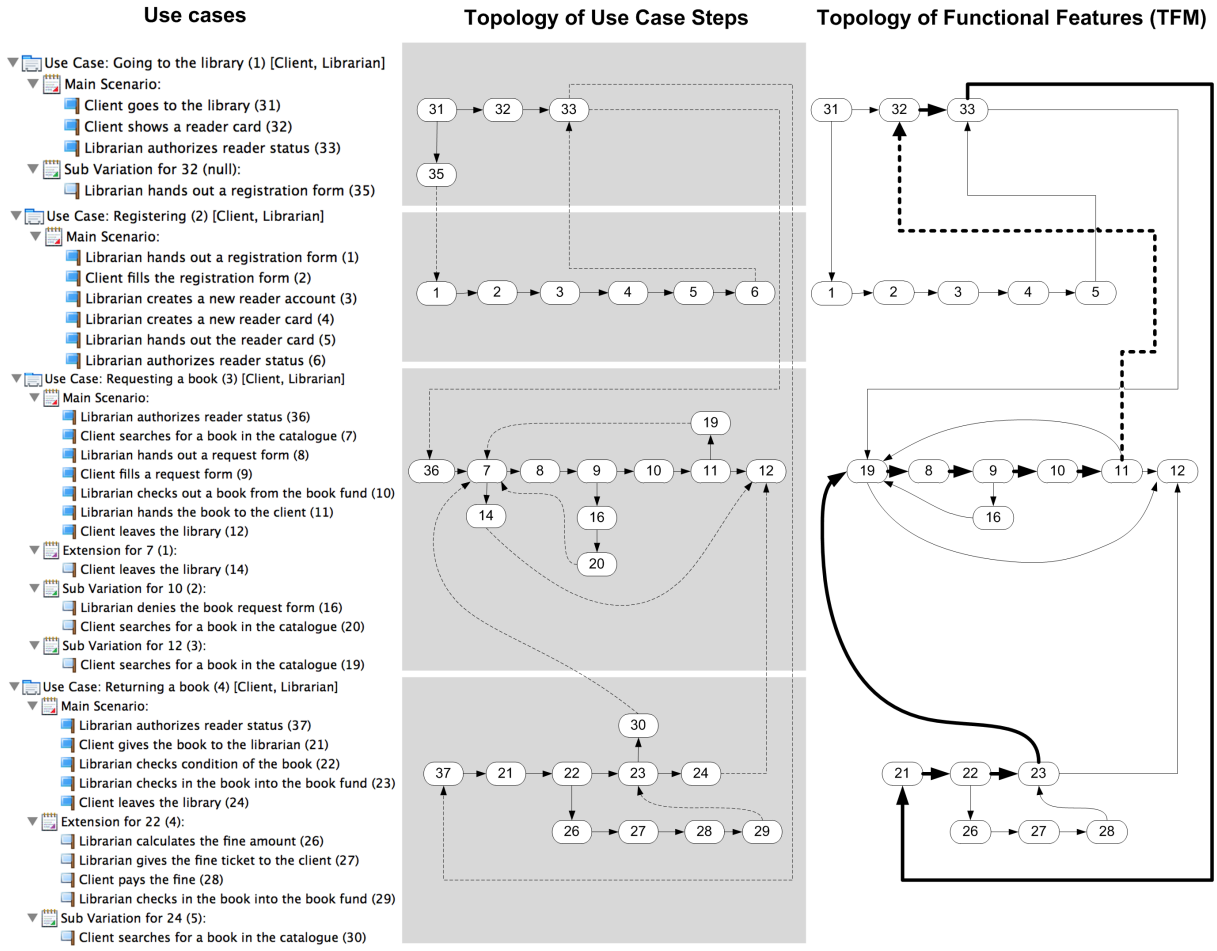

Fig. 2.3. Acquiring topological relationships from use cases.

However, sub-variation adds an effect to the functional feature represented by the previous step referenced by the sub-variation. Therefore, the setting of cause-effect relations between functional features represented within the same use case is very straightforward. As shown in Fig. 2.3, the 4 main sequences of functional features come from main scenarios of use cases. As a demonstration of use case extension and sub-variation analysis, let us consider functional features number 1 and 11. Functional feature number 1 has an additional effect because of the sub-variation 2a, but functional feature 11 has an additional effect because of the extension 4a.

In Fig. 2.3 on the right-hand side, the topology of use case steps can be seen in compliance with cause-effect relations between functional features (the ids of use case steps are re-used). IDM suggests several iterations back and forth between use cases and TFM until the system analyst can verify it is correct. The mapping between use case sentences, functional features and TFM should be intuitively illustrated and easily editable, so that any incompleteness, redundancy or inconsistency could be corrected. The bold arrows represent the main functioning cycle of the TFM. The functional features are set to be part of the main functioning cycle by the system analyst manually. By analyzing the use cases, it is possible to derive the TFM.

For the example shown in Fig. 2.3, the main functioning cycle consists of these functional features in the following sequence (represented by the identifier and corresponding use case step description): 32 – Client shows a reader card; 33 – Librarian authorizes a reader status; 21 – Client gives a book to the Librarian; 22 – Librarian checks the condition of the book; 19 – Client searches for a book in the catalogue; 8 – Librarian hands out a request form; 9 – Client fills in a request form; 10 – Librarian checks out a book from the book fund; 11 – Librarian hands the book to the client. The main functioning cycle includes all main processes of the example library system starting from a client coming to the library, client returning a book and client requesting a book. There are also other cycles in this TFM, including the registration of a client or requesting a book without returning one first. The main functioning cycle must be defined and set by the analyst based on input from domain experts. In the library example, the system analyst for the completeness of main functioning cycle sets a cause-effect relation between functional features with identifier 11 and 32 as shown in Fig. 2.3 with a dotted arrow. It cannot be determined automatically.

Another task is setting the cause-effect relations between functional features fetched from different use cases. Precondition section of use cases are used to define this relation, because it contains the use case step, which is the cause of the particular functional feature. For example, the precondition of use case “Requesting a book” is “Librarian authorizes a reader status”, which is the 3rd step of main scenario of use case “Arriving”. Moreover, as different use case sentences represent the same functional feature if their tuples conform, relation between different use cases can be fetched from extensions and sub-variations, as well.

2.4. Summary

To step towards the completeness of MDA and enable the automation of system analysis and software development, the IDM approach is considered. The IDM approach

provides a formal way to facilitate ontology for software engineering. It does not suggest a novel methodology for ontology development, but instead is based on the existing methodologies. This approach suggests ontology to be directly used as an input for domain modeling by exploiting business use cases and natural language processing (NLP), thus combining the declarative and procedural knowledge for the domain modeling. The IDM approach shows how declarative and procedural knowledge complement each other and can be compared for validation purposes. The IDM approach integrates the common standards used in business environment – use cases and ontology, and then provides a means to generate the initial domain model automatically. This level of automation and reuse of enterprise standards is the main innovation of the IDM approach. The IDM provides the opportunity to do this by using the Topological Functioning Model (TFM) as the domain model.

However, today there is no use of an approach without a supporting toolset. This model-to-model transformation to acquire a domain model was specifically developed by the author with the possibility for automation in mind. Since it has been shown here how the transformation can be developed, all pre-requisites for developing a supporting toolset for the IDM approach are in place.

3. SUPPORTING TOOLSET AND APPLICATION

The Integrated Domain Modeling approach provides an elegant solution to the complexity of domain model construction. However, for this to work in a real business environment for a business system modeling case there have to be tools to support the IDM approach. Moreover, these tools need to correspond to MDA standards and be based on available MDA frameworks, so that they are extendable and fit into MDA.

3.1. Scope and Architecture of the IDM Toolset

The IDM toolset needs to support the user to perform the following main activities: 1) To construct or reuse an existing domain ontology; 2) To develop the use case model describing the business processes of the domain; 3) To validate the use case model by using natural language processing and the domain ontology; 4) To automatically generate a CIM for this domain in the form of TFM; 5) To allow the user to further model the TFM of the domain by adding the main functional cycle and logical relationships; 6) To validate each of the models against the corresponding meta-model.

The users of this toolset would be the knowledge engineer and/or the system analyst who would work together with the business to gather and validate the input knowledge and resulting domain model.

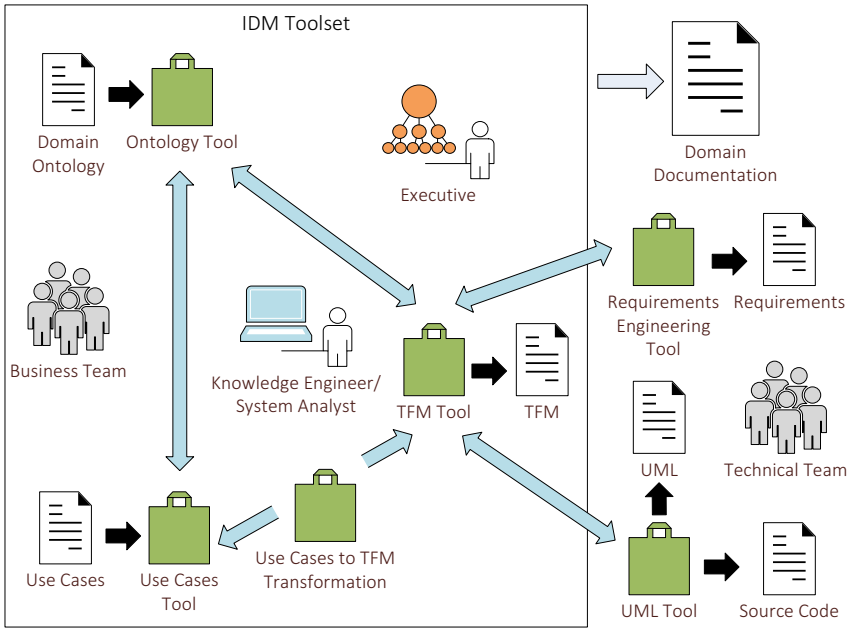

Fig. 3.1. Scope of the IDM toolset.

In Fig. 3.1, the author shows tools, artifacts and roles involved within the scope of the IDM Toolset and beyond. Document icon represents the artifacts – domain ontology, use cases, TFM, UML, source code, requirements and domain documentation. Briefcase icon represents the tools – ontology tool, use cases tool, use cases to TFM transformation, TFM tool, requirements engineering tool, and UML tool. Black arrows show the artifacts of the tools. Blue arrows represent the interaction between tools. The roles involved are represented with person icons – knowledge engineer, system analyst, business team, executive and technical team. Grey arrow shows the relationship between the IDM Toolset and documentation. The IDM toolset consists of: 1) ontology tool; 2) use cases tool; 3) TFM tool; 4) use cases to TFM transformation.

IDM specific MOF compatible meta-models have been developed by the author for the IDM use case template as well as for the TFM. In Fig. 3.2, the author shows the architecture of the IDM toolset. Document icon represents the artifacts – use case model and TFM model. Briefcase icon represents the tools – Use Case Editor, Use Cases to TFM transformation, TFM Editor, and TFM Diagram Tool. The arrows show conformity to the meta-models, Eclipse frameworks or use of a 3rd party Java library. The intermittent arrows show the transformation direction and also the initialization of the diagram.

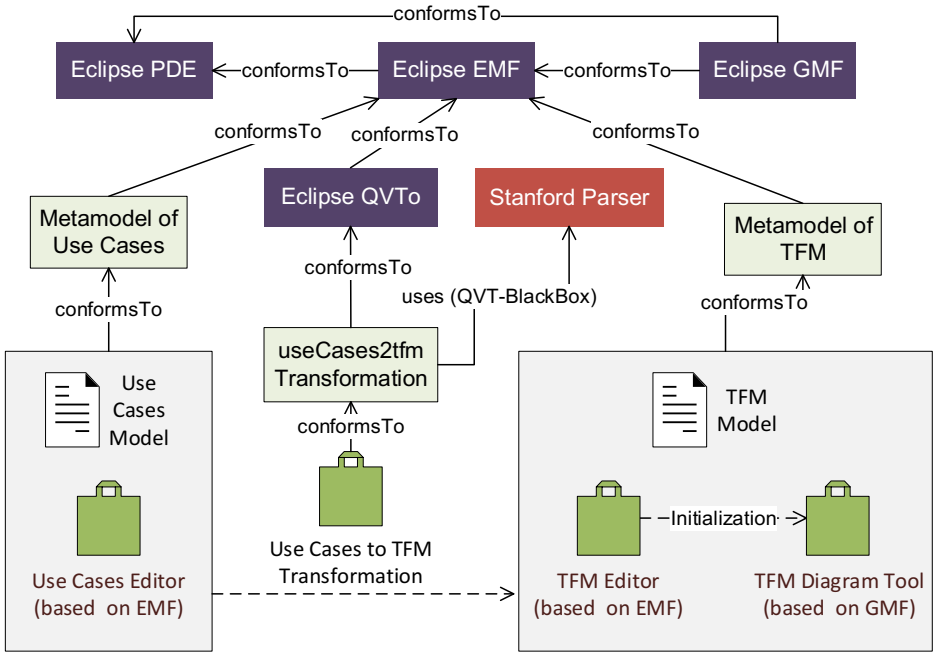

Fig. 3.2. Architecture of the IDM toolset.

This excludes the 3rd party OWL tool, for which the Protégé [75] tool is used. The following Eclipse frameworks are used – PDE, EMF, GMF and QVTo. Since EMF and GFM allow generating Eclipse plug-in, it conforms to the PDE, which can be later used to extend the functionality of the tools. In addition, a 3rd party statistical parser (Java library) is used for natural language processing – Stanford Parser [88]. QVT/BlackBox mechanism is used for integrating existing non-QVT libraries or transformations like statistical parser. The author developed the following tools – Use Case Editor, TFM Editor, TFM Diagram Tool, and Use Cases to TFM Transformation.

3.2. Use Case Editor

The Use Case Editor was implemented as an Eclipse plug-in by the author, which is based on the Eclipse Modeling Framework (EMF). The first and the most important step is to develop a Use Case Editor is to define the use case meta-model. Figure 3.3 shows a MOF-compatible meta-model for a set of use cases developed by the author. A set of use cases consists of 1 or more use cases, which have a main scenario, extensions and sub-variations, which consist of use case steps.

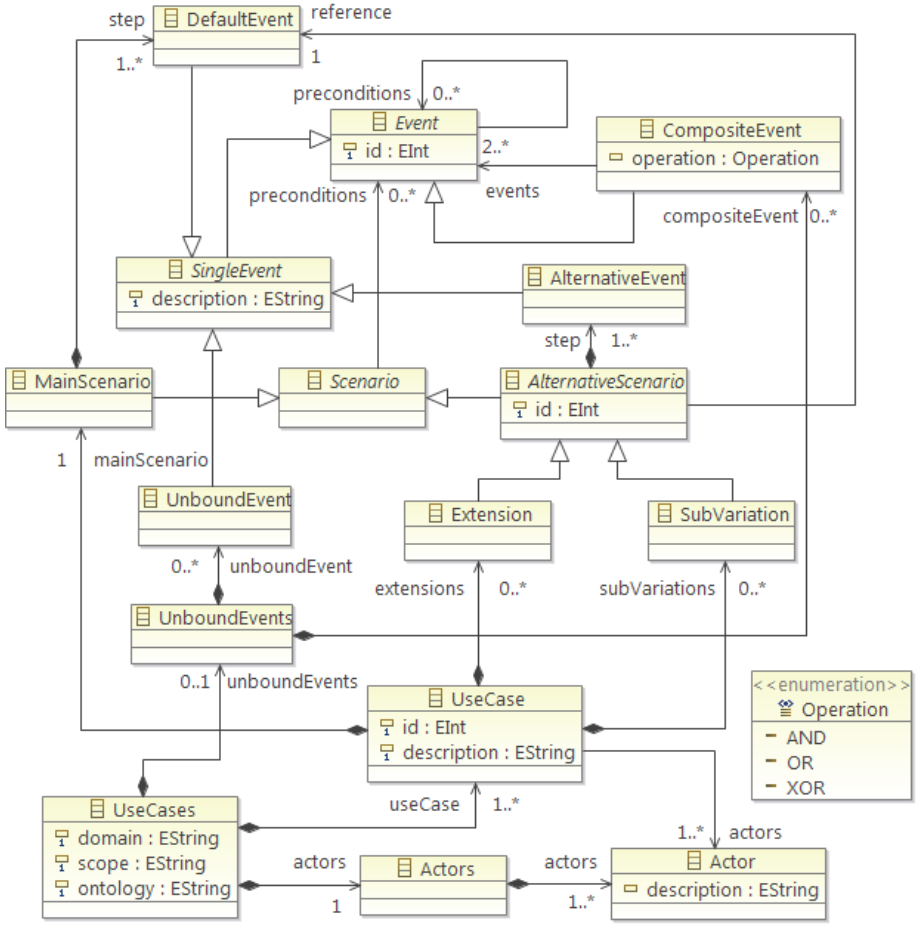

Fig. 3.3. Use case meta-model.

Each use case has a title, list of actors (at least 1 actor), and can have preconditions. Each use case step has its number description and can have a precondition. For extensions and sub-variations the reference attribute is used. This shows which of the steps in the main scenario it references, so for the main scenario steps reference will be empty.

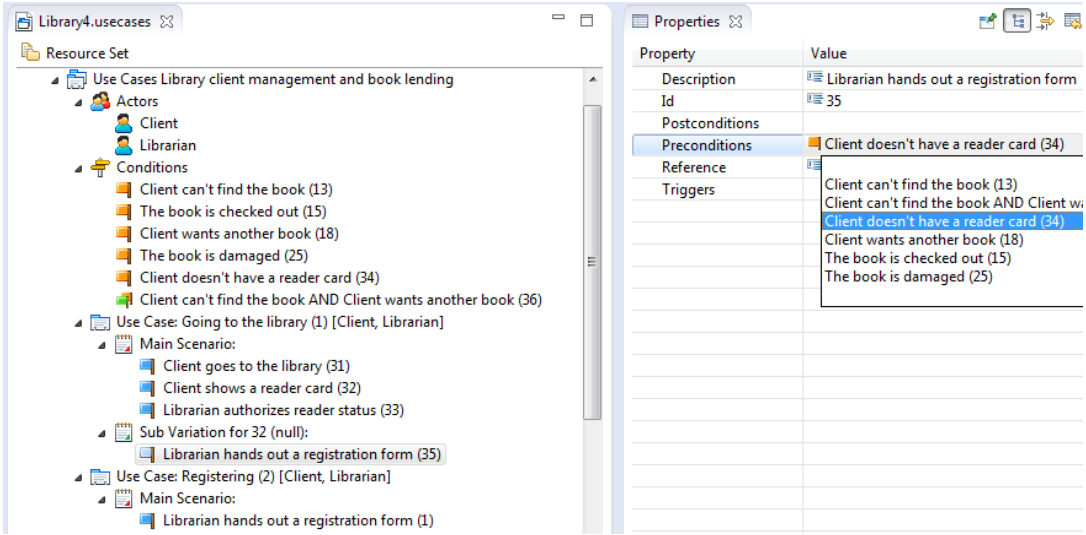

Fig. 3.4. Use Case Editor tool.

Figure 3.4 shows the Use Case Editor tool. On the left side, there is a use case model constructed by the user and on the right – the property view of a use case step, where it is possible to set preconditions. There are specific icons for each of the elements of the meta-model. This tool allows the users to define Actors, Conditions, Composite Condition, Use Cases, Main Scenarios, Alternative Scenarios and their steps. The file extension of the Use Case artifact is ".usecases". This file can also be opened with a text editor and then it is possible to see that underneath it has an XMI format.

3.3. TFM Editor and Diagram Tool

The TFM Editor looks very similar to the Use Cases Editor since it is also based on Eclipse EMF. However, TFM Diagram Tool implementation is based on the Eclipse GMF. Figure 3.5 shows the resulting TFM Diagram Tool, which was generated according to the GMF workflow and based on the TFM meta-model. This tool allows the users to define the TFM in a diagram form – actors, functional features, topological relationships, cycles and logical relationships. The file extension of the TFM is ".tfm" and TFM diagram artifact is ".tfm_diagram". These files can also be opened with a text editor and then it is possible to see that underneath it has an XMI format.

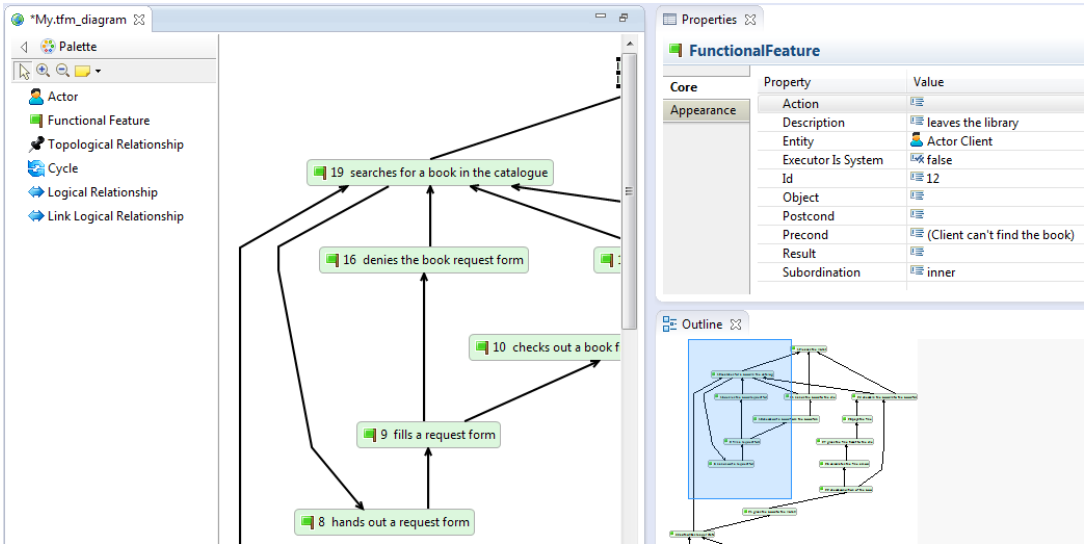

Fig. 3.5. TFM Diagram tool.

3.4. Use Cases to TFM Transformation

The Use Cases to TFM model-to-model transformation implementation as an Eclipse plug-in are based on Eclipse QVTo and Stanford Parser [75]. To execute the transformation a user has to right click on a file with extension ".usecases" and select the option "Transform to TFM". After that the user can check the console to see the log

of the model transformation. This shows references with identifiers, Actors created, Functional Features created and Topological Relationships created. In the same project a new file will appear after the model transformation with the extension ".tfm". This is the file of TFM model.

3.5. Summary

This Chapter introduced the supporting toolset of the Integrated Domain Modeling (IDM) approach, discussing the architecture, used technologies, implementation, model transformation, and application. The IDM toolset allows defining the business processes with Use Cases, validating them against the corresponding Ontology and then generating the domain model automatically in the form of Topological Functioning Model (TFM). This toolset enables the system analyst to acquire a formal domain model. This way it is possible to validate the business processes at the beginning of the software development, by ensuring that they correspond to the Ontology, by defining the scope based on inputs and outputs, and by also checking the functioning cycles of the processes.

4. APPROBATION OF THE INTEGRATED DOMAIN MODELING APPROACH

This Chapter provides a case study of applying the IDM approach and toolset to a Subscription Commerce Business case. The case study is based on a project done by Pearl Consulting AS, Norway (Ltd "Pearl Baltic Labs" in Riga, Latvia) [71] for a customer who is in subscription commerce business, which operates in Norway, Sweden, Denmark, Finland and the United Kingdom. The name of the particular Pearl Consulting AS customer will not be mentioned, but referred to as an SCB – Subscription Commerce Business. The integrated solution for SCB consists of the following main components: ERP, CRM, BI and e-commerce. It is based on the following SAP [77] products: SAP ECC, SAP CRM, SAP BW, SAP PI and Hybris. For the IDM approach first the corresponding ontology was build, then the use cases were identified and created, then use cases were validated against the ontology, and finally an initial TFM was generated, which then was analyzed and improved.

4.1. Business Domain

SCB offers their customers to subscribe to different kinds of products to be mailed to the address they want on a periodic basis. A central part of the SCB business process is product management, which consists of managing articles, bundles, products and adding subscription rules to the bundles. To clarify the product structure, the author gives an example. There is a product a customer can subscribe to named "Wilkinson Hydro 5". It consists of several bundles – sample bundle, subscription bundle and periodic bundle. The sample bundle consists of 2 articles – Wilkinson Hydro 5 razor and Wilkinson Foam. For example, the sample bundle will be shipped to the customer at subscription or re-subscription (this is the subscription rule of the sample bundle). The subscription bundle with Wilkinson Razors will be shipped every 3 months. The subscription happens on SCB e-commerce website.

4.2. Developing the Use Cases

The following 9 use cases were identified for SCB's TO-BE process: 1) Article management; 2) Bundle management; 3) Product management; 4) Customer subscription; 5) Bundle delivery; 6) Batch purchase; 7) Web content management; 8) Campaign management; and 9) Customer segmentation. In Fig. 4.1, a screenshot of the IDM Use Case tool is shown with the Article Management use case "UC-1" expanded. The actors involved in this use case are Product Manager and Hybris. There is the main scenario and 2 sub-variations (alternative scenarios of type sub-variation) – "A-1.1-

Edit" and "A-1.2-ArticleVariant". The main scenario walks through the process of Product Manager logging into the Hybris Product Cockpit and creating a new article with the necessary attributes. The first sub-variation is for the scenario of editing or modifying an existing article. The second sub-variation is for the scenario of creating article variants with a base article. Alternative scenarios specify the alternative flow of events. For example, sub-variation "A-1.1-Edit" references step "S-1.0.7", which states "Product Manager chooses to create a new article". Since the alternative scenario is of type sub-variation (not the other possible type, i.e., extension), the new flow of events will come in parallel to the referenced step (this will be clearly demonstrated in Fig. 4.2). The alternative scenario ends with step "S-1.1.3", which states "Hybris shows the Product Cockpit".

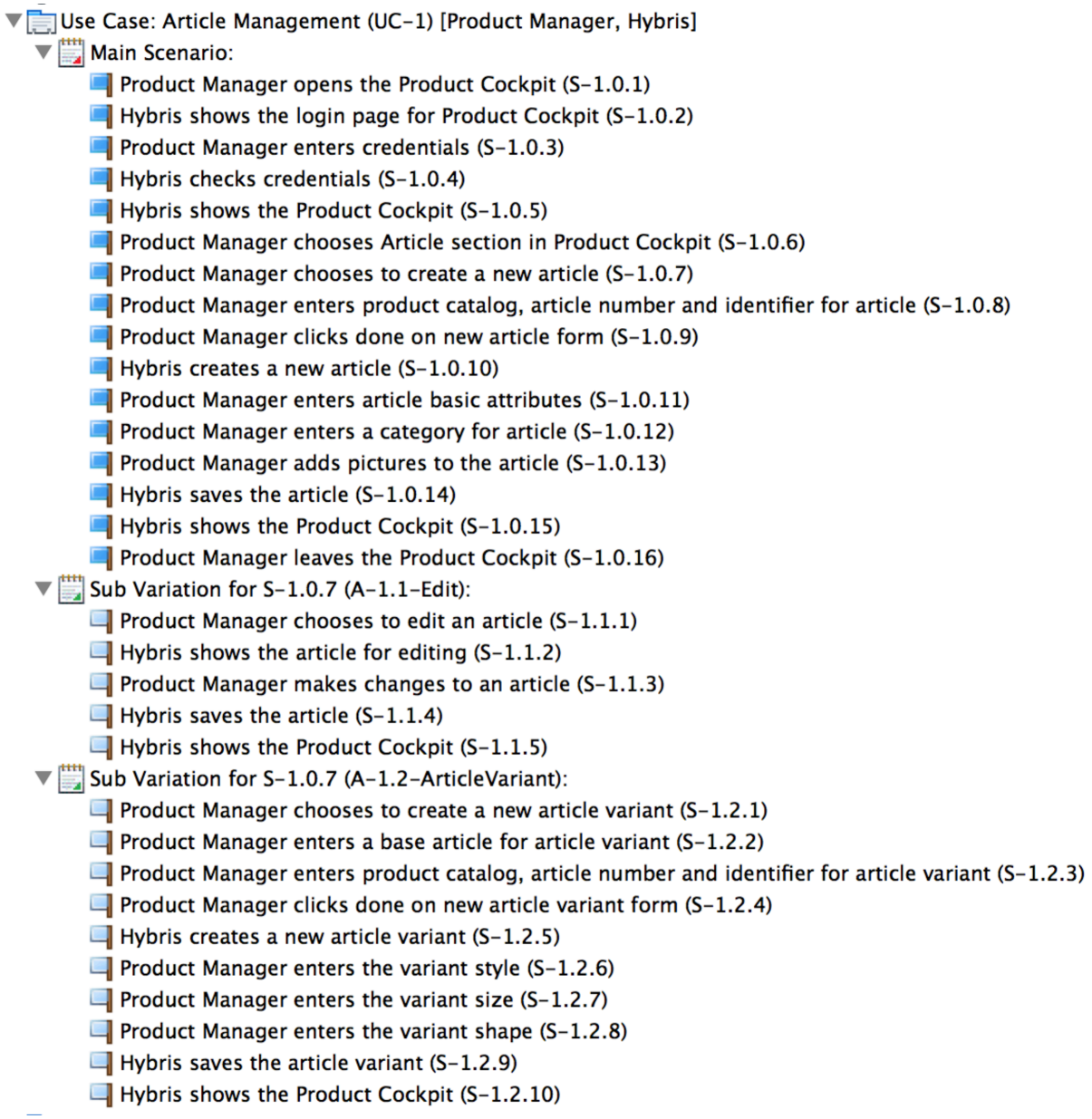

Fig. 4.1. Article Management use case defined with the IDM toolset.

4.3. Acquiring the TFM

Once the use cases are developed and have been validated against the Ontology, it is possible to do a model-to-model transformation using the IDM toolset's Use Cases to TFM Transformation tool. This automatically generates a TFM that corresponds to the defined Use Cases. Figure 4.2 shows the TFM for the SCB Article Management process, which corresponds to the Article Management use case. After acquiring the TFM, the system analyst in addition to the generated topological relationships needs to link the processes together (which are based on the independent use cases) creating functioning cycles.

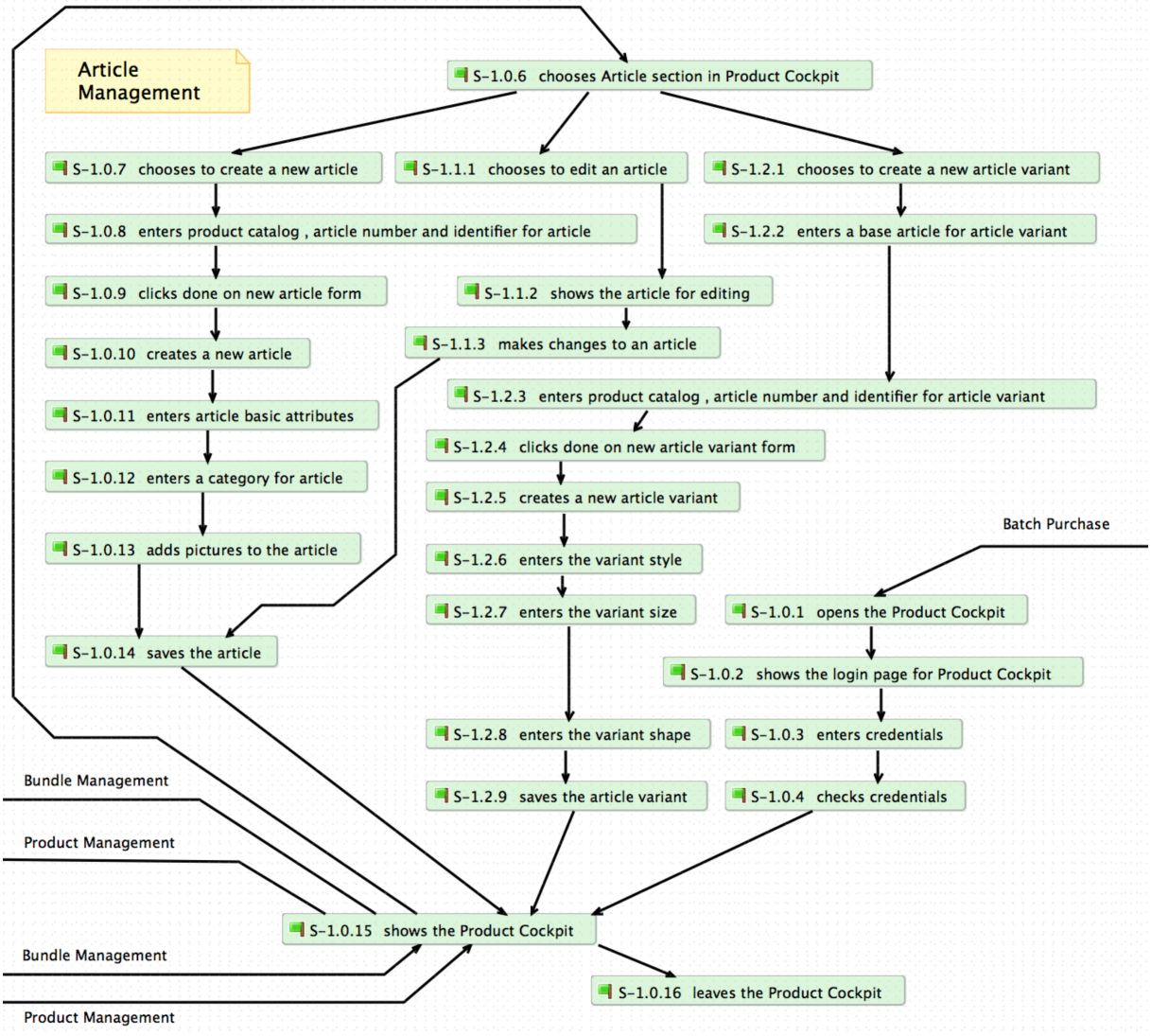

Fig. 4.2. Article Management in the form of TFM.

System analyst also needs to identify the main functioning cycle. In the case of SCB, the main functioning cycle goes through several processes – Batch Purchase, Customer Subscription and Bundle Delivery. These 3 processes define SCB daily

operation and enable the business. To link the 3 processes, the system analyst needs to create 3 topological relationships connecting the corresponding functional features. In addition to the main functioning cycle, there is also the cycle for creating and publishing the products in the first place, which covers the following processes – Article Management (shown in Fig. 4.2), Bundle Management and Product Management. To link this cycle with the main functioning cycle, the system analyst needs to create 2 additional topological relationships connecting Batch Purchase with Article Management and Product Management with Customer Subscription (the corresponding functional features). Thus, just by adding 5 additional topological relationships the TFM is complete.

4.4. Summary

The initial TFM for the business domain was acquired automatically using the IDM toolset based on the use cases and then manually corrected also using the IDM toolset. The main business processes that were analyzed were Article Management, Bundle Management, Product Management, Customer Subscription, Bundle Delivery and Batch Purchase. The ontology, use cases and the TFM were part of the project blueprint documentation delivered to SCB. Overall, the IDM approach and toolset provided the necessary means for business domain analysis and modeling.

The author has compared IDM to TFM4MDA, which is the closest approach for domain modeling, since it also produces a TFM. The initial starting point of both approaches can be considered similar, but the first artifact to be produced is completely different. For IDM it is the Ontology and Use Cases, but for TFM4MDA it is the informal description. From the effort point of view, TFM4MDA requires less effort; however, the IDM provides a better quality because of a better structure. The real strength of IDM shows at the next stage – acquiring the TFM. This introduces a significant level of automation for developing the domain model and saves effort, at the same time improving the quality.

CONCLUSIONS

The goal of this Doctoral Thesis has been to improve the process of domain analysis by providing an approach and a supporting toolset for acquiring a formal domain model that is transformable and can be used as a CIM within the MDA. The main result of the present research is the development of the Integrated Domain Modeling (IDM) approach and toolset, which allows defining the business processes

using Use Cases, validating them against corresponding Ontology and then generating a formal domain model automatically in the form of a Topological Functioning Model (TFM) with model transformation. All of the objectives defined for achieving the goal have been successfully accomplished and the following results and conclusions have been obtained:

1. Results of analyzing the existing domain modeling approaches are as follows:
    a. Since domain model is intended to be used by the business people, it should be simple to understand without special training; however, this is not the case with existing approaches, which tend to be quite complex;
    b. Some of the approaches are strong with declarative knowledge and others are strong with procedural knowledge; however, there is no approach to exploit both aspects of the domain to a full extent;
    c. Use Case is a simple approach for describing procedural knowledge and it is easy to understand by business people, however, being expressed in a natural language it can be inconsistent, ambiguous, hard to manage and hard to transform;
    d. Some novel approaches make use of Natural Language Processing (NLP) to deal with natural language texts for the domain analysis; however, they rely on an informal description in the beginning, which can be too unstructured and hard to manage to give enough input for the domain model;
    e. Tool support is very important for a domain modeling approach for the resulting domain model to be further used for software development;
    f. The domain modeling approaches have a low level of formality providing only a meta-model, in contrast the TFM and Ontology provide a mathematically formal model and formal model validation, thus having a high score as formal approaches;
2. Results of analyzing strengths, weaknesses and points for improvement of the TFM approach are as follows:
    a. The main strength of the TFM is its mathematical basis since it provides a formal way to capture the procedural aspects for the domain model;
    b. Another strength of the TFM is its inputs and outputs, which play an important role for identifying the scope of the domain model and enable the distinction between a system and its environment;
    c. The third strength of the TFM is the cycle structure and the main functioning cycle, which provides the opportunity to validate the domain model;

d. The forth strength of the TFM is the model-to-model transformations defined as part of TFM4MDA and improved with TopUML;
e. One of the weaknesses of the TFM is that it describes the procedural knowledge for a domain and lacks the declarative knowledge;
f. The other weakness of the TFM and TFM4MDA in particular is that it depends on an informal description at its beginning, which is hard to manage and analyze;
g. The third weakness is that there is no tool support for TFM, TFM4MDA or TopUML, and as of now the only way to acquire the TFM is by a heavy manual process;
h. Some suggested improvements for the TFM approaches include substitution of informal description with more formal domain knowledge, integration with declarative knowledge, model transformation within a CIM and proper tool support.

3. The Integrated Domain Modeling (IDM) approach has been developed by the author for acquiring a formal domain model in the form of TFM based on formal knowledge about the domain using Use Cases and Ontology as input. The IDM addresses the issues described above in the analysis;
4. A toolset to support the IDM approach has been developed by the author based on MDA standards and Eclipse EMF, GMF, M2M, which consists of Use Case Editor, TFM Editor, and Use Cases to TFM Transformation tools. By exploiting the domain model acquired by the IDM toolset, the system analyst together with the business can validate the business processes before the actual software developments start. Thus, it is possible to make sure that the business processes correspond to the domain and also to the ontology. This way debugging can be done in the domain model even before any line of code is written;
5. The case study for an e-commerce software development project demonstrated how the IDM approach and toolset can be successfully used to gather and record the domain knowledge and acquire a domain model in the form of TFM via a model-to-model transformation, thus also acquiring a graphical representation of the business process automatically. Conclusions from conducting the case study are as follows:
    a. The IDM toolset requires minimal training (up to 1 day) to start using it, because the only mandatory pre-knowledge is Use Cases;
    b. IDM use cases are a very convenient way to document the business process during the blueprinting phase, because both the technical people and the business people understand it without special training;

c. High-level Ontology does help the domain analysis process, but going into details (e.g., object property definition) takes a lot of effort and is not mandatory (class hierarchy may well be enough as in case of the Subscription Commerce Business);
d. Acquiring a graphical representation of the business process is quick and easy if the use cases are defined (TFM is acquired automatically via model transformation, which corresponds to MDA standards);
e. Splitting the processes by use case and introducing Actor pools can improve graphical representation of the TFM.

Future research directions are as follows:

- In the current state, the Use Case Editor of IDM toolset is able to support the use case development process, but it lacks the functionality to upload an ontology and do automatic use case validation against the ontology. This can be done by using Stanford Statistical Parser for the use case analysis, OWL API for importing Ontology, and Eclipse EMF Validation Framework for validating the Use Case model.
- Integration of the IDM toolset with UML tools by implementing the model transformations from the TFM to UML described in TopUML research.

BIBLIOGRAPHY

[1] Alphabetical list of part-of-speech tags used in the Penn Treebank Project / Internet. - https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html [Accessed: June 14, 2014]
[2] ARIS / Internet. - http://www.ariscommunity.com/ [Accessed: June 14, 2014]
[3] Arlow J., Neustadt I. Introduction to BPMN 2. Mountain Way Limited, 2012. – 182 p.
[4] Asnina E. The Formal Approach to Problem Domain Modeling within Model Driven Architecture// In 9th International Conference on Information Systems Implementation and Modelling. - Prerov, Czech Republic, Ostrava, 2006. - pp. 97-104.
[5] Asnina E., Osis J. Topological Functioning Model as a CIM-Business Model// Model-Driven Domain Analysis and Software Development: Architectures and Functions. - IGI Global, 2011. - pp. 40-64.
[6] Baclawski K., Kokar M. K., Kogut P., Hart L., Smith J. E., Letkowski J., Emery P. Extending the Unified Modeling Language for Ontology Development// Int. Journal Software and Systems Modeling (SoSyM). - 2002. - Vol. 1, No. 2. - pp. 142-156.

[7] Broy M. Domain Modeling and Domain Engineering: Key Tasks in Requirements Engineering// Perspectives on the Future of Software Engineering. - Springer Berlin Heidelberg, 2013. - pp. 15-30.
[8] Business Process Model and Notation (BPMN) version 2.0.2 / Internet. - omg.org/spec/BPMN/2.0.2/PDF [Accessed: June 14, 2014]
[9] Caliusco M. L., Galli M. R., Ruidías H. J. Towards the integration of ontologies in the context of MDA at CIM level// XVIII Congreso Argentino de Ciencias de la Computación. - 2012.
[10] Coleman D. A. Use Case Template: draft for discussion// Fusion Newsletter. - April 1998. Available. - http://www.engr.sjsu.edu/~fayad/current.courses/cmpe202-Fall2009/docs/lecture3/CmpE202-22-UC-Template-Ex-L3-3g.pdf
[11] Cranefield S. Networked knowledge representation and exchange using UML and RDF// Journal of Digital Information. - 2001. - Vol. 1, No. 8. Available. - https://journals.tdl.org/jodi/article/viewArticle/30/31 [Accessed: June 14, 2014]
[12] Donins U. Software Development with the Emphasis on Topology// In Proceeding of 13th East-European Conference on Advances in Databases and Information Systems (ADBIS 2009). - Volume 5968 of LNCS. Springer, 2010. pp. - 220-228.
[13] Doniņš U. Topological Unified Modeling Language: Development and Application. Doctoral thesis, Riga Technical University, 2012. - 224 p.
[14] Eclipse GMF: Graphical Modeling Framework / Internet. - http://wiki.eclipse.org/Graphical_Modeling_Framework [Accessed: June 14, 2014]
[15] Eclipse Papyrus / Internet. - http://www.eclipse.org/papyrus/ [Accessed: June 14, 2014]
[16] Eclipse Requirements Management Framework / Internet. - http://www.eclipse.org/rmf/ [Accessed: June 14, 2014]
[17] EMF Developer Guide: The Eclipse Modeling Framework (EMF) Overview. - 2005. / Internet. - http://help.eclipse.org/ganymede/index.jsp?topic=/org.eclipse.emf.doc/references/overview/EMF.html [Accessed: June 14, 2014]
[18] Evans E. Domain-driven Design: Tackling Complexity in the Heart of Software. - Addison-Wesley Professional, 2004. - 359 p.
[19] Falkovych K., Sabou M., Stuckenschmidt H. UML for the Semantic Web: Transfor-mation-Based Approaches// In Knowledge Transfor-mation for the Semantic Web. - IOS Press, 2003. - pp. 92–106. Available. - http://www.cwi.nl/~media/publications/UML_for_SW.pdf [Accessed: June 14, 2014]
[20] Fliedl G., Kop C., Mayr H. C., Salbrechter A., Vohringer J., Weber G., Winkler C. Deriving static and dynamic concepts from software requirements using sophisticated tagging// Data & Knowledge Engineering. - 2007. - Vol. 61, Iss. 3. - pp. 433-448.
[21] Fouad A. Embedding requirements within model-driven architecture// Software Quality Journal 19.2. - 2011. - pp. 411-430.
[22] Francu J., Hnetynka P. Automated Generation of Implementation from Textual System Requirements// In Proceedings of the 3rd IFIP TC 2 CEE-SET. - Brno, Czech Republic, Wroclawskiej. - 2008. pp. 15-28.

[23] Frankel D. S. Model Driven Architecture: Applying MDA to Enterprise Computing. Wiley, Indianapolis, 2003. - 352 p.
[24] Fuchs N. E., Kaljurand K., Kuhn T. Attempto Controlled English for Knowledge Repre-sentation// In Reasoning Web, Fourth International Summer School 2008, Lecture Notes in Computer Science 5224. - Springer, 2008. - pp. 104–124.
[25] Gasevic D., Djuric D., Devedzic V. Model Driven Architecture and Ontology Development. Springer, Heidelberg, 2006. - 311 p.
[26] Goodman N. D. Mathematics as an objective science// American mathematical monthly. - 1979. pp. 540-551.
[27] Google Scholar / Internet. - http://scholar.google.com [Accessed: June 14, 2014]
[28] Guarino N. Formal Ontology and Information Systems// In Proceedings of Formal Ontology and Information Systems. - Trento, Italy, IOS Press, Amsterdam, 1998. pp. 3–15.
[29] Jacobson I., Spence I. Use case 2.0: Scaling up, scaling out, scaling in for agile projects. Ivar Jacobson International, 2011. - 54 p.
[30] Jones C. Positive and negative innovations in software engineering. International Journal of Software Science and Computational Intelligence (IJSSCI) 1.2. - 2009. - pp. 20-30.
[31] Jurafsky D., Martin J. H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. - Pearson Education, 2000. - 934 p.
[32] Kaindl H. Structural Requirements Language Definition, Defining the ReDSeeDS Languages / Internet. - http://publik.tuwien.ac.at/files/pub-et_13406.pdf
[33] Kardoš M., Drozdová M. Analytical method of CIM to PIM transformation in Model Driven Architecture (MDA)// Journal of Information and Organizational Sciences 34.1. - 2010. - pp. 89-99.
[34] Kaur A., Mansaf A. Role of Knowledge Engineering in the Development of a Hybrid Knowledge Based Medical Information System for Atrial Fibrillation// American Journal of Industrial and Business Management. Vol. 3., No 1. - 2013. - pp. 36-41.
[35] Kirikova M, Finke A., Grundspenkis J. What is CIM: an information system perspective// Advances in Databases and Information Systems. Vol. 5968. - Springer Berlin Heidelberg, 2010. - pp. 169-176.
[36] Kirikova M. Domain Modeling Approaches in IS Engineering. Model-Driven Domain Analysis and Software Development: Architectures and Functions. - IGI Global, 2011, pp. 388-406.
[37] Kolmogorov A. N., Fomin S. V. Introductory Real Analysis (Silverman, R. A., Ed.). - Mineola, NY: Courier Dover Publications, 1975. - 416 p.
[38] Kontio M. Architectural manifesto: Choosing MDA tools / Internet. - http://www.ibm.com/developerworks/library/wi-arch18.html [Accessed: June 14, 2014]
[39] Malan R., Bredemeyer D. Functional Requirements and Use Cases / Internet. - http://www.bredemeyer.com/pdf_files/functreq.pdf [Accessed: June 14, 2014]
[40] Mayr H. C., Kop C. A User Centered Approach to Requirements Modeling// In Modellierung. - 2002. - pp. 75-86.
[41] MDA: Model Driven Architecture / Internet. - http://www.omg.org/mda/ [Accessed: June 14, 2014]

[42] Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification / internet. - http://www.omg.org/cgi-bin/doc?ptc/07-07-07.pdf [Accessed: June 14, 2014]
[43] Miller J., Jishnu M. MDA Guide Version 1.0. 1. Object Management Group, 2003. - 51 p.
[44] Miller, Joaquin, and Jishnu Mukerji. "Model driven architecture (mda)." Object Management Group, Draft Specification ormsc/2001-07-01 (2001).
[45] Moore R., Lopes J. Paper templates// In TEMPLATE'06 1st International Conference on Template Production. - SciTePress, 1999.
[46] Murata T. Petri nets: Properties, analysis and applications// In Proceedings of the IEEE 77.4. - 1989. pp. 541-580.
[47] Nickols F. The Knowledge in Knowledge Management. The Knowledge Management Yearbook 2001–2002. - Butterworth-Heinemann, Boston, 2000. - pp. 12-21.
[48] Nikiforova, O., et al. "Development of the tool for transformation of the two-Hemisphere model to the UML class diagram: technical solutions and lessons learned." Proceedings of the 5th International Scientific Conference „Applied Information and Communication Technology. 2012.
[49] Noy N. F., McGuinness D. L. Ontology development 101: A guide to creating your first ontology. Technical Report SMI-2001-0880, Stanford Medical Informatics. - 2001.
[50] OMG Meta Object Facility (MOF) Core Specification Version 2.4.2 OMG April 2014 / Internet. - http://www.omg.org/spec/MOF/ [Accessed: June 14, 2014
[51] OMG: Object Management Group / Internet. - omg.org [Accessed: June 14, 2014]
[52] Ondrej M, Richta K. The BPM to UML activity diagram transformation using XSLT// Dateso. - Vol. 9. - 2009. - pp. 119-129.
[53] Osis J. Investigating Troubles of Complex System Functioning and Category Theory // Cybernetic and Diagnosis, Volume. 4. - Riga: Zinatne, 1970. - pp. 15-20 (in Russian)"
[54] Osis J. Software Development with Topological Model in the Framework of MDA// In Proceedings of the 9th CAiSE International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'2004) in connection with the CAiSE'2004. - Vol. 1. - RTU, Riga, 2004. pp. 211 – 220.
[55] Osis J., Asnina E. A Business Model to Make Software Development Less Intuitive// In Proceedings of the 2008 International Conference on Innovation in Software Engineering. - Vienna, Austria, IEEE Computer Society CPS, Los Alamitos, USA, 2008. - pp. 1240-1246.
[56] Osis J., Asnina E. Derivation of Use Cases from the Topological Computation Independent Business Model. Model-Driven Domain Analysis and Software Development: Architectures and Functions. - IGI Global, Hershey, New York, 2011. - pp. 65-89.
[57] Osis J., Asnina E. Enterprise Modeling for Information System Development within MDA// In 41th Annual Hawaii International Conference on System Sciences. - HICSS, USA, 2008. - pp. 490.
[58] Osis J., Asnina E. Is Modeling a Treatment for the Weakness of Software Engineering? Model-Driven Domain Analysis and Software Development: Architectures and Functions. - IGI Global, Hershey, New York, 2011. - pp. 1-14.

[59] Osis J., Asnina E. Model-Driven Domain Analysis and Software Development: Architectures and Functions. - IGI Global, Hershey, New York, 2011. - 487 p.
[60] Osis J., Asnina E. Topological Modeling for Model-Driven Domain Analysis and Software Development: Functions and Architectures. Model-Driven Domain Analysis and Software Development: Architectures and Functions. - IGI Global, Hershey, New York, 2011. - pp. 15-39.
[61] Osis J., Asnina E., Grave A. Formal Problem Domain Modeling within MDA// Communications in Computer and Information Science (CCIS). Software and Data Technologies. - Vol. 22. Springer-Verlag Berlin Heidelberg, 2008. - pp. 387-398.
[62] Osis J., Asnina E., Grave A. Computation Independent Modeling within the MDA// In Proceedings of the IEEE International Conference on Software Science, Technology and Engineering. - Herzlia, Israel, IEEE Computer Society, 2007. - pp. 22-34.
[63] Osis J., Asnina E., Grave A. Computation Independent Representation of the Problem Domain in MDA// J. Software Eng. - Vol. 2, iss. 1. - pp. 19--46. Available. - http://www.e-informatyka.pl/e-Informatica/Wiki.jsp?page=Volume2Issue1 [Accessed: June 14, 2014]
[64] Osis J., Asnina E., Grave A. Formal Computation Independent Model of the Problem Domain within the MDA. Information Systems and Formal Models// In Proceedings of the 10th International Conference ISIM'07. - Silesian University in Opava, Czech Republic, 2007. - pp. 47-54.
[65] Osis J., Asnina E., Grave A. MDA Oriented Computation Independent Modeling of the Problem Domain// In Proceedings of the 2nd International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2007). - Barcelona, Spain, 2007. - pp. 66 -71.
[66] Osis J., Donins U. Formalization of the UML Class Diagrams// Evaluation of Novel Approaches to Software Engineering. - Springer-Verlag, Berlin Heidelberg, New York, 2010. pp. 180-192.
[67] Ouvans C. From BPMN process models to BPEL web services// Web Services, 2006. ICWS'06. International Conference on. IEEE, 2006. - pp. 285-292.
[68] Overmyer S. Lavoie B. Rambow O. Conceptual Modeling through Linguistic Analysis Using LIDA// In Proceedings of 23rd International Conference on Software Engineering (ICSE 2001). - Toronto, Canada, 2001.
[69] Overmyer S., Lavoie B., Rambow O. Conceptual Modeling through Linguistic Analysis Using LIDA// In Proceedings of the 23rd International Conference on Software Engineering. - Toronto, Ontario, Canada, 2001. pp. 401-410.
[70] OWL: Web Ontology Language / Internet. - http://www.w3.org/TR/owl2-quick-reference/ [Accessed: June 14, 2014]
[71] Pearl Consulting / Internet. - http://www.pearlconsulting.no
[72] Plante F. Introducing the GMF Runtime, 2006. / Internet. - http://www.eclipse.org/articles/Article-Introducing-GMF/article.html [Accessed: June 14, 2014]
[73] Poesio M. Domain modelling and NLP: Formal ontologies? Lexica? Or a bit of both?// Applied Ontology, Vol. 1, No. 1. IOS Press, 2005. - pp. 27–33.
[74] Prieto-Díaz R. Domain Analysis: An Introduction// ACM SIGSOFT Software Engineering Notes 15.2. - 1990. - pp. 47-54.
[75] Protege / Internet. - http://protege.stanford.edu [Accessed: June 14, 2014]

[76] Santorini B. Part-Of-Speech Tagging Guidelines for the Penn Treebank Project// Technical report MS-CIS-90-47, Department of Computer and Information Science, University of Penllsylvania, 1990.
[77] SAP AG / Internet. - http://www.sap.com
[78] Scheer A. W., Oliver T., Otmar A. Process modeling using event-driven process chains// Process-Aware Information Systems. - 2005. - pp. 119-146.
[79] Šlihte A. Implementing a Topological Functioning Model Tool// In Scientific Journal of Riga Technical University, 5. series., Computer Science. - Vol. 43. - Riga, 2010. - pp. 68–75.
[80] Šlihte A. Introduction to Integrated Domain Modeling Toolset // Scientific Journal of RTU. Computer Science. - 2014. (to be published)
[81] Šlihte A. The Concept of a Topological Functioning Model Construction Tool// In 13th East-European Conference, ADBIS 2009, Associated Workshops and Doctoral Consortium, Local Proceedings. JUMI, Riga, Latvia, 2009. - pp. 476-484.
[82] Šlihte A. The Specific Text Analysis Tasks at the Beginning of MDA Life Cycle// In Data-bases and Information Systems Doctoral Consortium. Latvia, Riga, July 5-7, 2010. - pp. 11–22.
[83] Šlihte A. Transforming Textual Use Cases to a Computation Independent Model// MDA & MTDD 2010. Greece, Athens, July 22-24, 2010. - pp. 33–42.
[84] Šlihte A. Using Use Cases for Domain Modeling// In Proceedings of the 7th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2012) - 2012. - pp. 224-231.
[85] Šlihte A., Osis J., Doniņš U. Knowledge Integration for Domain Modeling// In Proceedings of the 3rd International Workshop on Model-Driven Architecture and Modeling-Driven Software Development. China, Bejing, June 8-11, 2011. - pp. 46-56.
[86] Soley R. Model driven architecture. OMG white paper, 2000. Available. - http://www.geocities.ws/pravin_suman/Resources/00-11-05.pdf [Accessed: June 14, 2014]
[87] Subramaniam K., Liu D., Far B., Eberlein A. UCDA: Use Case Driven Development Assistant Tool for Class Model Generation// In Proceedings of the 16th SEKE. Banff, Canada, 2004. Available. - http://enel.ucalgary.ca/People/eberlein/publications/SEKE-Kalaivani.pdf [Accessed: June 14, 2014]
[88] The Stanford Parser: A statistical parser. The Stanford Natural Language Processing Group, 2010 / Internet. - http://nlp.stanford.edu/software/lex-parser.shtml [Accessed: June 14, 2014]
[89] Van Lamsweerde A. Requirements engineering: from craft to discipline// In Proceedings of the 13th international Workshop on Early Aspects. - New York: Association for Computing Machinery, Inc, 2008. - pp. 238-249.
[90] W3C, OWL Web Ontology Language Overview, W3C Recommendation February 10 2004 / Internet. - http://www.w3.org/TR/owl-features/ [Accessed: June 14, 2014]
[91] White S. A., Introduction to BPMN. IBM Cooperation, 2004. Available. - http://yoann.nogues.free.fr/IMG/pdf/07-04_WP_Intro_to_BPMN_-_White-2.pdf [Accessed: June 14, 2014]

[92] XML Metadata Interchange (XMI) Specification, v2. 3.2, 2014 /Internet. - http://www.omg.org/spec/XMI/ [Accessed: June 14, 2014]
[93] Tsai A., Wang J., Tepfenhart W., Rosca, D: EPC workflow model to WIFA model conversion// In Proceedings of Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference, Vol. 4, pp. 2758-2763
[94] Suchanek F.M., Kasneci G., Weikum G. Yago: A large Ontology from wikipedia and wordnet// Web Semantics: Science, Services and Agents on the World Wide Web 6.3, 2008, pp. 203-217.
[95] Ferrario R., Oltramari A.: A first-order cutting process ontology for sheet metal parts// Formal Ontologies Meet Industry 198, 2009, pp. 22.
[96] Jones C.: Software project management practices: Failure versus success// CrossTalk: The Journal of Defense Software Engineering 17, 2004.
[97] Kruczynski K.: Business process modelling in the context of SOA–an empirical study of the acceptance between EPC and BPMN// World Review of Science, Technology and Sustainable Development 7.1, 2010, pp. 161-168.
[98] Nüttgens M., Feld T., Zimmermann V.: Business Process Modeling with EPC and UML: transformation or integration?// The Unified Modeling Language. Physica-Verlag HD, 1998, pp. 250-261.
[99] Strommer M, Murzek M., Wimmer M.: Applying model transformation by-example on business process modeling languages// Advances in Conceptual Modeling–Foundations and Applications, Springer Berlin Heidelberg, 2007, pp. 116-125.
[100] Bodrow W.: The dynamic of professional knowledge utilized in software applications for process controlling// Advances in Manufacturing, 2014, pp. 1-6.