Advancements of the Topological Functioning Model for Model Driven Architecture Approach[1]

Armands Slihte, Janis Osis, Uldis Donins, Erika Asnina and Bernards Gulbis

Faculty of Computer Science and Information Technology
Institute of Applied Computer Systems, Riga Technical University, Riga, Latvia

Abstract. This paper describes the advancements of the Topological Functioning Model (TFM) for Model Driven Architecture (MDA) approach. This approach recognizes the computation independent nature of a TFM and suggests it to be used as the Computation Independent Model (CIM) within MDA, thus partially automating system analysis. Since the proposal of this approach, there have been a number of significant improvements, revealing new possibilities for system analysis, domain modeling and system design modeling. These advancements include integrating knowledge engineering with system analysis for domain modeling and acquiring a Topological Class Diagram, thus providing unique features within Platform Specific Model (PSM) for further transformation and code generation.

1 Introduction

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 [1]. 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). The CIM describes system requirements and the way system works within its environment while details of the application structure and realization are hidden or are yet undetermined. The 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 [2]. The CIM may include 3 main parts: 1) Knowledge Model; 2) Business Model; 3) Business Requirements for the System.

TFM offers a formal way to define a system by describing both the system's functional and topological features [3]. TFM represents the system in its business environment and shows how the system is functioning, without details about how the system is constructed. Related research [4] is suggesting using TFM as CIM by con-

[1] This work has been supported by the European Social Fund within the project „Support for the implementation of doctoral studies at Riga Technical University".

Slihte A., Donins U., Osis J., Asnina E. and Gulbis B..
Advancements of the Topological Functioning Model for Model Driven Architecture Approach .
DOI: 10.5220/0003586700910099
In *Proceedings of the 3rd International Workshop on Model-Driven Architecture and Modeling-Driven Software Development* (MDA & MDSD-2011), pages 91-99
ISBN: 978-989-8425-59-1

structing it with TFM for MDA approach; acquiring a mathematically formal and thus transformable CIM.

TFM for MDA approach introduces a way to acquire a formal CIM and provides the necessary methods to construct the CIM from domain knowledge (which can also be considered as part of CIM) and further transform CIM to PIM/PSM. As described in [5] and [6] an informal description of the system in textual form can be produced as a result of system analysis. Construction of the CIM is part of related research [7] and [8]. Research [7] describes a way to use Natural Language Processing (NLP) for defining domain knowledge that can be further formally analyzed. Research [8] shows how it is possible to automatically acquire a CIM from domain knowledge. An algorithm is introduced to automatically derive the TFM from business use cases. This algorithm utilizes the statistical parser to analyze the syntax of use case sentences and identify functional features for the TFM. The problem of potential ambiguity and inconsistency of the business use case steps can be resolved by using ontology. This issue is discussed further in this paper. Additionally TFM for MDA approach improves PIM/PSM by introducing topology. This topology is acquired from the CIM. In related research [9] author is showing an approach for software development with emphasis on topology, where PIM/PSM is supplemented with topology acquired from the CIM. In this paper we are suggesting sequential phases of TFM for MDA approach to be combined for fulfilling the MDA life-cycle.

In [10] author is proposing a toolset for TFM for MDA approach and has developed a tool for manually constructing a TFM by analyzing domain knowledge. In this paper we are proposing to expand the toolset: 1) making it possible to automatically acquire CIM if the domain knowledge is properly defined; 2) including the topological UML as target model from TFM as source.

TFM for MDA approach has been suggested to deal with lack of a formal CIM, but since then this approach has significantly evolved, providing also a formal way for defining knowledge about a domain and discovering unique features within PSM for further transformation and code generation. This paper describes the advancements of the TFM for MDA approach, proposes a framework for further research and suggests expanding the toolset.

This paper is organized as follows. Section 2 is looking into domain knowledge and proposes a way for defining CIM-Knowledge Model. Section 3 is looking into domain modeling and proposes a way to acquire CIM-Business Model in correspondence to the knowledge model defined before. Section 4 is describing the topological UML and the benefits of using it as the PIM/PSM. Section 5 is showing the new schema for TFM for MDA approach integrating ontology, business use cases, TFM and TopUML. Section 6 is describing the necessary supporting toolset and how these tools will work together to support MDA life-cycle.

2 Domain Knowledge

At the beginning of system analysis, it is necessary to acquire knowledge about the domain – business system and its environment. System analysis is interested in knowledge that can be articulated or captured in form of text, tables, diagrams, product

specifications and so on. 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 [11]. In this section we discuss the domain knowledge, which can also be considered as part of the CIM, i.e., Knowledge Model.

Previously the TFM for MDA approach assumed that knowledge about a business system can be can be represented as an informal description in from of text in natural language. Because such an informal description can be far too complex, ambiguous, redundant and inconsistent for a formal analysis, using formally defined knowledge with correspondence to well known standards is considered as one of the advancements of this approach.

Developing business use cases is a popular approach for defining the procedural knowledge of a business system. There are many different business use case templates and the structure of a use case can be adjusted depending on the situation and the development team [12]. For defining procedural knowledge TFM for MDA approach is using textual business use cases with the following structure: 1) use case title, 2) actors, 3) pre-conditions, 4) main scenario, 5) extensions, and 6) subvariations. In order for the business use cases to be understandable by a computer, the step sentences are defined using a controlled natural language. Attempto Controlled English (ACE) appears perfectly natural, but — being a controlled subset of English — is in fact a formal language [13]. ACE texts are enabled for computer processing and can be unambiguously translated into discourse representation structures, a syntactic variant of first-order logic.

This solves some of problems when using natural language, but still not all. The open problems are: 1) ambiguity, e.g., possibility to express the same meaning using different words; 2) inconsistency of business use cases, i.e., there might be steps defined that do not make sense in the given business system. We cannot put this much responsibility on a system analyst who will be developing these business use cases. This kind of ambiguity and inconsistency should be automatically discovered and eliminated. TFM for MDA suggests ontology to be used as a predefined lexicon for the specific domain to deal with these problems. Ontologies provide logical statements that describe what terms are, how they are related to each other, and how they can or cannot be related to each other. This feature is exploited to validate the procedural knowledge. If there is ontology for a given domain and corresponding use cases, we consider that the domain knowledge has been formally defined.

3 Domain Modeling

Related research is mostly focusing on the Business Model and Business requirements for the System of CIM, which include more or less structured language models accepted by requirements community, e.g. the Language Extended Lexicon (LEL) and scenarios, Semantics for Business Rules and Business Vocabulary, and use cases. Another means for CIM definitions are different languages and notations for business process modeling that may include semi-formal languages, such as ARIS-Event Process Chains, Business Process Modeling Notation (BPMN) used in Business

Process Modeling (BPM), Vide CIM Level Language (VCLL), UML profiles, UML Activity Diagrams. The only formal means for definition of business models is Petri nets and their extensions [2].

TFM for MDA approach is suggesting not avoiding the CIM, but instead starting the transformations from this model, thus providing evident consistency of PIMs, at least with the Business Model and also with the Knowledge Model. At the high level of abstraction the TFM reflects cause-effect relations among business functions. At the lower level of abstraction TFM reflects cause-and-effect relations among business activities. Moreover, at this level TFM can be decomposed to business processes. The TFM contains knowledge about its business functions, organizational units (positions rules, subsystem, units), and relations between business functions and between business functions and organizational units. From [2], transformation from the Knowledge Model to the Business Model cannot be performed automatically, because it requires human participation since information in the Knowledge Model is specified informally and used to have implicit knowledge.

With the advancements of the TFM for MDA approach we have acquired a formal Knowledge Model and it is now possible to transform this model to the Business Model automatically. As shown in [8] it is possible to automatically acquire TFM from business use cases using a special algorithm. Nevertheless, creating these business use cases still require human participation – defining and analyzing the knowledge about the domain. In context of MDA, from perspective of PIM/PSM the input model for transformation is the Business Model. Knowledge Model and Business Requirements are the source models for the Business Model.

4 Topological UML

Another branch of this research is suggesting a new UML profile – TopUML, which incorporates the topological nature of TFM with UML. UML approaches usually are strong with defining the class hierarchy, but TopUML provides unique benefits for MDA, defining the Topological Class Diagram. By using TopUML it is possible to acquire cause-effect relationships between methods for PIM/PSM from CIM. Class diagrams reflect the statistic structure of the system, and by using class diagrams it is possible to model objects and their methods that are involved in the system. Using the standard UML class diagram specification it is not possible to reflect the cause-and-effect relations within a system to indicate which certain activity of an object triggers another objects certain activity.

Uniqueness and main value of the Topological Class Diagram is that it is able to show the relations between different methods from the same or different classes [9]. This property can be used for PSM and code generation. It allows defining part of the logic within the methods, not only the structure. TFM for MDA approach allows doing this automatically by means of transformation from CIM. No other approach provides this possibility from perspective of CIM; this kind of logic has to be added manually by the PSM designer.

It is possible to develop a topological class diagram describing the business system, if the corresponding TFM has been developed. The transformation between

TFM and topological class diagram, i.e., transformation between CIM and PIM/PSC is described in [9]. To execute the transformation several steps have to take place: 1) creation of the TFM; 2) creation of the problem domain objects graph; 3) transformation into class diagram. To obtain a problem domain object graph, it is necessary to detail each functional feature of the TFM to a level where it uses only one type of objects. After construction of problem domain object graph all the vertices with same type of objects and operations must be merged, while keeping all relations with other graph vertices. As a result, object graph with direct links is defined. This then can be transformed into a corresponding TFM.

It is possible to develop a topological class diagram where the established TFM topology between classes is retained. In traditional model-driven development relations (mostly associations and generalizations) between classes in the UML class diagram are defined by the modelers' discretion. Moreover, by using TFM for MDA approach it is also possible to partially automatically acquire a more complete UML sequence diagram from the CIM by exploiting the topology.

5 TFM for MDA Approach

To step towards the completeness of MDA and enable the automation of system analysis and software development the TFM for MDA approach is considered. This paper introduces sufficient advancements to this approach. The main idea behind this approach is MDA transformation from CIM to PIM to PSM.

This approach starts the system analysis process from formally defined declarative and procedural knowledge. TFM for MDA approach integrates declarative and procedural knowledge providing a common approach for system analysis with perspective of integration with MDA. As shown in Fig. 1 we are exploiting ontology and business use cases for defining the Knowledge Model. The ontology is constructed by a knowledge engineer and business use cases are constructed by a system analyst (also known as business analyst – the one who is competent to analyze organization and design of businesses and business processes). While constructing business use cases, they have to be validated in order to correspond to the ontology. This is an iterative process, because the ontology or business use cases have to be modified until they correspond.

The next step is acquiring the Business Model. When Knowledge Model is constructed and verified, it is possible to generate the Business Model automatically using the TFM generation algorithm described in [8]. This algorithm utilizes the statistical parser to analyze the syntax of use case sentences and identify functional features for the TFM. Nevertheless, TFM will have to be validated as well. If any changes are necessary, they will have to be done in the Knowledge Model and then the TFM can be regenerated. Additionally, within the Business and Requirements Model it is possible to derive the Business Processes and UML Use Case diagram from TFM.

The next step of TFM for MDA lifecycle is transforming CIM to PIM/PSM. The source for this transformation is the Business Model (CIM) and the target is the Design Model (PIM/PSM), which includes Topological Class Diagram, Sequence Dia-

gram and other UML diagrams. Until now we have provided transformation for Topological Class Diagram and Sequence Diagram, but research continues to provide transformations for other diagrams.

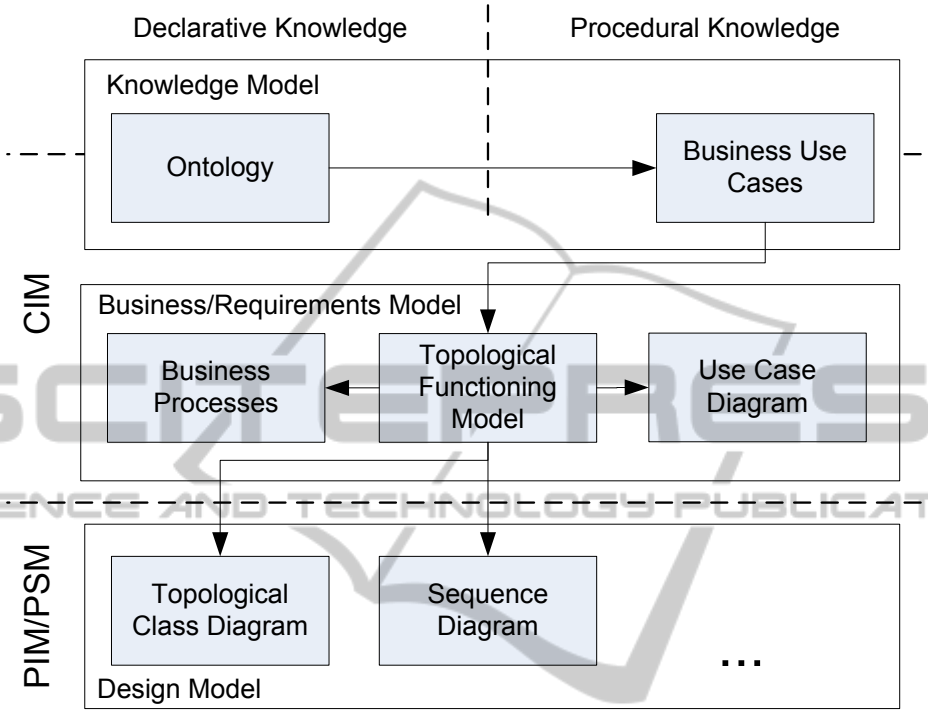

Fig. 1. This schema shows the integrated TFM for MDA approach. It starts with the domain knowledge defined as formal CIM-Knowledge Model, which includes business use cases and ontology. It then provides a formal transformation to CIM-Business Model, which is defined by TFM. The CIM can then be further expanded to include Business Processes and UML Use Case Diagram (corresponding transformation are also provided). After that it is possible to exercise a transformation from CIM to PIM/PSM, acquiring a Topological Class Diagram, Sequence Diagram, etc.

6 Supporting Toolset

In earlier work [10] and [8] some suggestions have been made what tool support would be necessary for TFM for MDA approach. In this paper expand the toolset to support the new workflow suggested in previous section. Advantage of using MDA standards is that MOF compatible meta-models can be created for business use cases using XMI, as well as for a TFM, which authors have already is defined in [7] and [4]. A statistical parser can be used for analyzing the sentences of use cases, and thus retrieving functional features for a TFM of the system. To prevent incompleteness, redundancy or inconsistency of the business use cases ontology and controlled natural language is used. At last, for retrieving the cause-effect relations between these func-

tional features the structure of the business use cases is exploited.

TFM for MDA approach lacks the necessary tool support. The purpose of these tools would be to enable users: 1) to construct or reuse a domain ontology; 2) develop business use cases for this domain; 3) verify these business use cases via controlled natural language and the ontology defined previously; 4) automatically generate the CIM for this domain in form of a TFM; 5) verify the functional requirements; 6) transform the CIM to PIM/PSM in a form of a UML profile - TopUML. The users of this toolset would be the knowledge engineer and the system analyst.

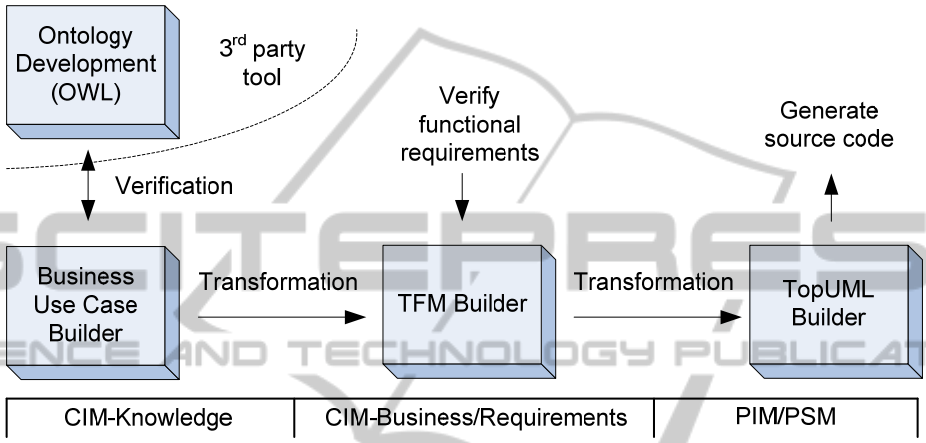

Fig. 2. This schema shows the toolset necessary for TFM for MDA approach. This toolset consists of an Ontology Development tool, Business Use Case Builder, TFM Builder and TopUML Builder. Ontology development tool has to support OWL standard, but other than that it can be a 3rd party tool, i.e., Protégé. You can also see the distinction between CIM and PIM/PSM that correspond to these tools from perspective of MDA.

As Fig. 2 shows the TFM for MDA toolset consists of: 1) Ontology Development tool – a tool for defining ontology according to OWL standard; 2) Business Use Case Builder – this tool will allow the user to define the business use cases for this domain and check if the correspond to the ontology; 3) TFM Builder – a tool to fetch the functional features and generate the TFM for the domain (it will also allow to verify the functional requirements); 4) TopTFM Builder – a tool for acquiring TopUML from a TFM and later allow to generate the source code for the system.

7 Conclusions

This paper describes the advancements of the TFM for MDA approach. This approach was suggested to acquire a formal CIM, but since has significantly evolved, providing a way for defining domain knowledge and discovering unique features within PSM for further transformation and code generation. An informal description of the domain can be far too complex, ambiguous, redundant and inconsistent for a formal analysis, so using formally defined knowledge with correspondence to well

known standards is proposed. A new UML profile TopUML has been introduced to enable cause-effect relationships between class methods for PIM/PSM, which can be retrieved from CIM.

This paper also shows how the different step of the TFM for MDA approach relate to each other and how it fits into MDA lifecycle. This approach provides a new perspective for domain modeling, allowing the domain model to be generated automatically if the knowledge is gathered and defined before accordingly. This way we are integrating knowledge engineering and system analysis. Main value of the Topological Class Diagram is that it can provide unique properties for use in PSM and code generation.

Further research includes the development of the supporting toolset for this approach. This would complement the automation of system analysis, and introduce artificial intelligence to software development.

References

1. Frankel D. S.: Model Driven Architecture: Applying MDA to Enterprise Computing. Indianapolis. OMG Press, Wiley (2003).
2. 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) 40-64.
3. Osis, J.: Topological Model of System Functioning (in Russian). In: Automatics and Computer Science, J. of Acad. of Sc., pp. 44--50, Zinatne, Riga (1969).
4. Osis, J., Asnina, E., Grave, A.: Computation Independent Representation of the Problem Domain in MDA. J. Software Eng. Vol. 2, Iss. 1, (2008) 19—46 Available: http://www.e-informatyka.pl/e-Informatica/Wiki.jsp?page=Volume2Issue1 [Accessed: Feb 28, 2010].
5. Asnina, E.: The Formal Approach to Problem Domain Modeling within Model Driven Architecture. In: 9th International Conference on Information Systems Implementation and Modelling, pp. 97 – 104. Prerov, Czech Republic, Ostrava (2006).
6. Osis, J., Asnina, E.: Enterprise Modeling for Information System Development within MDA. In: 41th Annual Hawaii International Conference on System Sciences, pp. 490. HICSS, USA (2008).
7. Šlihte A.: The Specific Text Analysis Tasks at the Beginning of MDA Life Cycle. In: Databases and Information Systems Doctoral Consortium, Latvia, Riga, 5.-7. July (2010) 11–22.
8. Šlihte A.: Transforming Textual Use Cases to a Computation Independent Model. MDA & MTDD 2010, Greece, Athens, 22.-24. July (2010) 33–42.
9. 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) 220-228.
10. Šlihte A.: Implementing a Topological Functioning Model Tool. In: Scientific Journal of Riga Technical University, 5. series., Computer Science, Vol. 43, Riga (2010) 68–75.
11. Gasevic, D., Djuric, D., Devedzic, V.: Model Driven Architecture and Ontology Development. Springer, Heidelberg (2006).
12. Malan, R., Bredemeyer, D.: Functional Requirements and Use Cases, March 2001. Available: http://www.bredemeyer.com/pdf_files/functreq.pdf [Accessed: Mar 31, 2011].
13. Fuchs, N. E., Kaljurand, K., Kuhn, T.: Attempto Controlled English for Knowledge Representation. In Cristina Baroglio, Piero A. Bonatti, Jan Maluszynski, Massimo Marchiori,

Axel Polleres, and Sebastian Schaffert, editors, Reasoning Web, Fourth International Summer School 2008, Lecture Notes in Computer Science 5224. Springer (2008) 104–124.
14. Object Management Group, Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. Available: http://www.omg.org/cgi-bin/doc?ptc/07-07-07.pdf [Accessed: Mar 31, 2010].