USE OF BUSINESS MODELS WITHIN MODEL DRIVEN ARCHITECTURE

BIZNESA MODEĻU LIETOŠANA MODEĻVADĀMAS ARHITEKTŪRAS IETVAROS

E. Asnina

Business model, topology, computation independent model, Model Driven Architecture

1. Introduction

Model Driven Architecture® (MDA®) is a framework for software development driven by models. It was introduces in 2001 by the *Object Management Group* (OMG) [1]. The main purpose of MDA is to separate viewpoints in specifications and strengthen the role of analysis and design in the project development. MDA suggests three viewpoints on the system (a computational independent, a platform independent, and a platform specific one) that are reflected in a *Computation Independent Model* (CIM), a *Platform Independent Model* (PIM) and a *Platform Specific Model* (PSM) correspondingly. The CIM specifies problem domain knowledge and requirements to the system. The PIM specifies system's structure and behavior, but does not show platform-specific details. The PSM specifies system's structure and behavior enhancing the PIM with platform-specific details. From the viewpoint of fully automated transformations, the one of bottlenecks of MDA is informal transformation from CIM to PIM, because this is exactly that point, where conversion from informal information to more formal one is being performed.

This paper continues research related to formalization of foundations and applications of MDA. The research has begun from the inquisition of provision a formal feedback in MDA life cycle [2]. The research of formalization of a CIM resulted in the creation of description of system's functioning features in form of unique tuples that are being obtained from the informal description of the system [3]. The common framework of application of a formal mathematical model (Topological Functioning Model discussed here) to the problem domain modeling was described in brief in [4]. This framework enables making a solution in compliance to a problem in a formal way. This approach application to the enterprise modeling is discussed in [5]. The main objective of this paper is to discuss what capabilities business models used within MDA have in order to provide precise as possible transformations within the CIM from existing domain to planned one.

The paper is organized as follows. Section 2 discusses distinct parts of the CIM. Some of business models that can be or were used as the CIM are discussed in Section 3. Section 4 analyzes capabilities that these business models have in order to make a planned domain in conformity with the existing one. Section 5 concludes main results and states further direction of research.

2. What is MDA Computation Independent Model?

Transformations are the main way of refinement of MDA models (see **Fig.1**). MDA foresees automated refining transformations starting from the PIM, namely, from PIM to PSM to code [1]. The reason of ignoring the CIM is that it is used to reflect information about the universe of discourse, i.e. about existing ("as is") as well as planned ("to be") domains (an organization and an application, correspondingly), while the PIM is used mostly to reflect information about a domain "to be" (and usually this is an application). Besides that, the CIM describes mostly business people's viewpoint on those domains, while the PIM describes software developers' viewpoint on the application. Thus, the CIM is described informally in most cases.

Generally, two kinds of transformations related to CIMs and PIMs can be defined [6]:

- *Transformation from CIM to CIM*. It is manual, intuitive, non-automatable transformation that is dedicated to refinement of a description and requirements of the business system "as is".
- *Transformation from CIM to PIM*. It is manual, intuitive, possibly automatable transformation that is dedicated to transition from requirements of the domain "as is" to models of the domain (software system) "to be".

Informal nature of CIMs hardens (or even makes it impossible) automation of transformations. However, a computation independent level has not to be ignored as it is done in modern applications of MDA, because the existing domain (a system "as is") is an environment where the solution (a system "to be") will operate. This means that system-to-be requirements are constraints on the world "as is". In other words, systems-to-be requirements must be in compliance with the domain where these systems will operate.

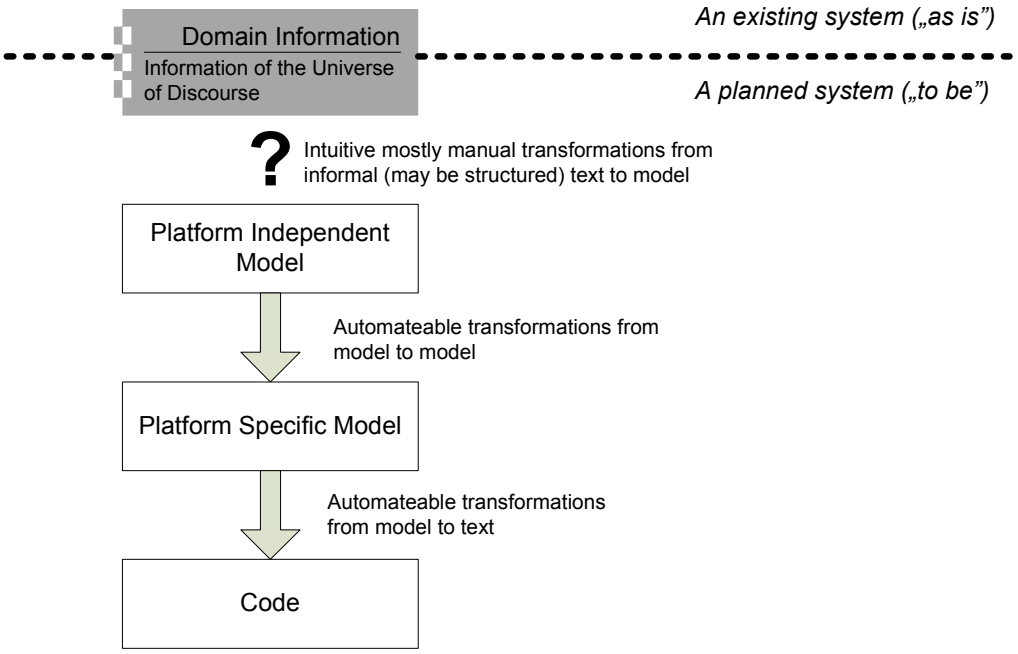

Fig.1. Transformations between models within MDA

As stated in [1] "…The requirements for the system is modeled in a computation independent model, CIM describing the situation in which the system will be used. Such a model is sometimes called a *domain model or a business model*." The term "business model" and related terms such as "a business fact", "a business rule" and "a business process" originated from business modeling. But what do they actually mean?

According to [7], a *business model* is a precise description of the business in its environment, by the business, in the language of business people, dedicated to business purposes (not necessarily IT). A model can be textual, tabular, graphic or a combination of them. If underpinnings of the business model are formal, then it is possible to handle this model by machine. Moreover, different objectives can require different models, thus it is necessary to determine purposes of construction of business models.

A more complete description of a *business model* is given by Stan Hendryx in [8]: a business models is a "collection of related architectures or blueprints of, by, and for business people, aimed toward capturing ... the essential workings of the business (not IT capabilities per se) from a purely business perspective. A business model provides comprehensive answers to the six basic interrogatives: What? How? Where? Who? When? Why? In doing so, the business people intend to provide a sufficient understanding of the business that may be used in a variety of ways to solve business problems as perceived by business people, one of which is providing business requirements for information systems."

Thus, a business model reflects some business knowledge. Usually business knowledge is expressed using words and phrases that business people know and understand, i.e. in other words by using terms. *Business facts* are constructed using these terms as foundations, and express things that business peoples know about their own businesses. *Business rules* make a use of facts in order to help in control of business operations and to ensure that business is executed in the way required by business people [9].

Summarizing all the above mentioned, a business model is able to reflect those parts and rules of the business that are not related to information systems as well as those ones that are related to information systems. Besides that, a *business process* can be considered as a category of a business model that focuses on transforming input resources to the output in order to add value for people inside and/or outside the business [8].

Hence, the CIM is the business people's modeling sandbox [10]. The CIM is mainly oriented on the business people. However, there are still discussions about what is suitable to be modeled as the CIM and how the CIM can be organized. And what does it mean "computation" and "computation independent"?

The conception of "computation" comes from mathematics, where this means an algorithmic process that generates certain results by following an effective procedure. In turn, the poly-semantic conception "information processing" comes from the control engineering, where this means things that are transmissible (mentally or physically) using messages to the target point. As stated in [11], these two conceptions were separated in cybernetics at the beginning, but as the years go by, they merged. However, when we speak about computation in the context of software development, it is necessary to understand that "computation" means rather digital computation than information processing as such. Thus, the CIM means a model that does not show exactly information processing by digital computation means. In other words, the CIM is allowed to reflect computation and logic without specification of participating actors or mechanisms. The CIM is allowed to specify that the computation or logical inference must be executed by a computer, but specification of a type of the computer or how the computer organizes or performs this computation or logical inference is forbidden. Stan Hendryx stated in [7] that the CIM should not be defined in any way to limit what business people can say in the CIM, regardless of how much of a computational flavor it may or may not have: "Business process charts, business arithmetic, business formulae, and business decision tables ... all are always fair game in the CIM, to describe business terms, business facts, business processes, business events, business organization, business locations, and business policies." If IT people think that there is too much digital computation in the CIM, then the CIM must be modified in accordance with the decision made together by business and IT people after their discussions.

By analyzing [7], [10], [12], the CIM model can be organized in three main parts:

- *CIM – Knowledge Model,*
- *CIM – Business Model,*
- *CIM – Business Requirements for the System:*
    - *Functional Requirements,*
    - *Interaction Requirements*
    - *Environment Contract.*

The *CIM – Knowledge Model* [12] relates to the highest level of the CIM model levels. It reflects an enterprise from the holistic point of view, thus providing the general vision of the enterprise with focus on enterprise knowledge. This knowledge should be locally refined throughout sequential lower levels. The CIM – Knowledge Model may include three types of diagrams, namely, block, ontological and knowledge diagrams.

The *CIM – Business Model* in [7] and [10] is a "pure" business model that is focused on the business scope and goals as well as terminology, resources, facts, roles, policies, rules, organizations, locations and events of concern to the business. However, it does not reflect system considerations. The scope of the Business Model in the CIM must, at a minimum, include the business functions served by the system-to-be. Authors in [12] defined three possible types of models within the CIM – Business Model: Organizational Model, and two system models – Structural Model and Behavior Model. The Organizational Model can be described in terms of goals, organizational structure, analysis diagrams

and business rule diagrams. In turn, the Structural Model includes product and resource diagrams, and the Behavior Model includes process and service diagrams.
The *CIM – Business Requirements for the System* [10] contains the contract between the business and IT about what the business people expect the system-to-be (an application) to do. These expectations are expressed in precise terms, clearly indicating what parts of the business the system-to-be will automate. This model is built on and refers to the CIM – Business Model. This model contains functional requirements, system user interaction requirements, and environmental contracts:

- The *functional requirements* specify the information stored and processed by the system-to-be, the business rules and business processes that will be implemented in the system-to-be. However, there may be some information, business processes and business rules the business does not want to implement in the system-to-be.
- The *system user interaction requirements* describe the details of use cases, leaving out user interface technical details. It reflects data presentation and navigation requirements, and look-and-feel standards. Even though screen shots are discouraged as excessive in this model.
- The *environment contracts* specify the non-functional requirements for the system-to-be, including performance, throughput, availability, and reliability.

Requirements need to be stated completely and unambiguously. Requirements may contain detailed technical specifications, or reference to technical models, if it is necessary from the business viewpoint.

3. Overview of business models used within MDA

In this paper, the main discussion relates to the two last defined CIM models, namely, the CIM – Business Model (structural and behavioral models) and CIM – Business Requirements for the systems. These models can be expressed using models in different notations. However, there are several models, which are specially defined for business modeling by the OMG, namely, *Semantics of Business Vocabulary and Business Rules* (SBVR) and *Business Process Modeling Notation* (BPMN). These models as well as *use cases* and *Topological Functioning Model* (TFM) are described in brief in this section.

3.1. Semantics of Business Vocabulary and Business Rules

SBVR is the first specification developed by the OMG for the so called Model-Driven Business. The current is SBVR version 1.0 [13]. SBVR specification defines:

- *Business Vocabulary,* which is a special purpose language for specification of business terminology – concepts (including terms, names, and definitions), definitional rules, and advices of possibility; it serves as a natural language ontology,
- *Business Rules,* which allow to specify business actions of an organization such as business policies, operative business rules, and advices of permissions.

The framework of SBVR suggested by John Hall in [14] is illustrated in **Fig.2**. It states that the main auditory of SBVR is business people. By using SBVR they can specify *meaning* of business concepts, facts and rules using expression predefined for a concrete natural language. Specifications are founded on *semantic formulations* and *formal logic*.
The SBVR formalism [13] states that semantic formulations are not representations or expressions of meaning. Rather, they are structures of meaning – the logical composition of meaning. Using SBVR, the meaning of a definition or statement is communicated as facts about the semantic formulation of the meaning, not as a restatement of the meaning in a formal language. There are two kinds of semantic formulations:

- Logical formulation, i.e. structures propositions, both simple and complex. Specializations of that kind are given for various logical operations, quantifications, atomic formulations based

on fact types and other formulations for special purposes such as objectification and nominalization;

- Projection, which structures intensions assets of things that satisfy constraints; projections formulate definitions, aggregations, and questions.

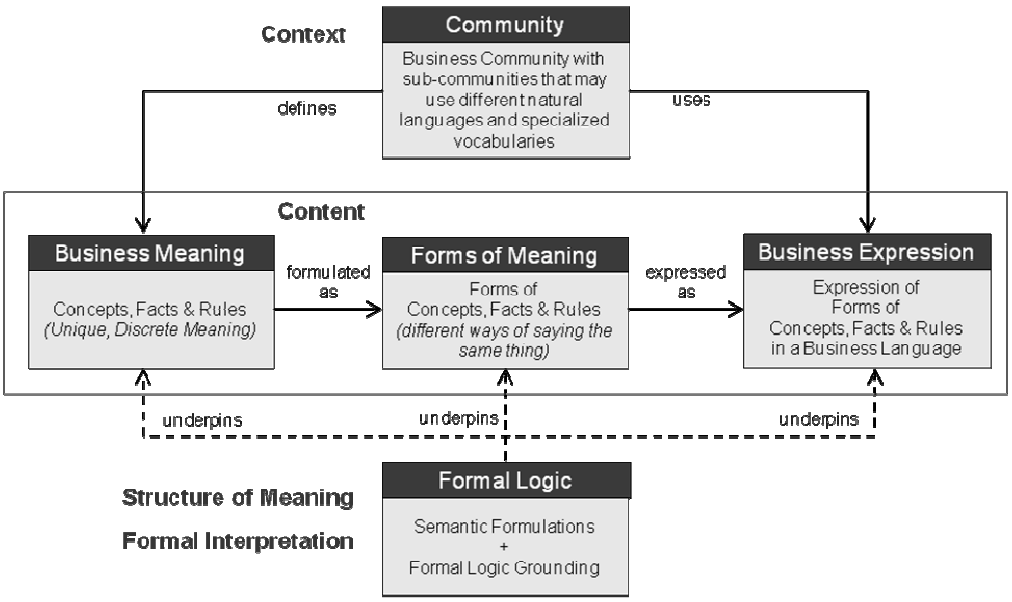

Fig.2. The framework of SBVR [14]

There are several things in SBVR taken from formal logics [13]. Static constraints may typically be expressed as logical formulations that are equivalent to formulae in 2-valued, first-order predicate calculus with identity. The rule formulations may use any of the basic alethic or deontic modal operators. These modal operators are treated as proposition-forming operators on propositions (rather than actions). Other equivalent readings may be used in whatever concrete syntax is used to originally declare the logical rule (e.g., "necessary" might be replaced by "required," and "obligatory" might be replaced by "ought to be the case"). Every constraint has an associated modality, determined by the logical modal operator that functions explicitly or implicitly as its main operator.

3.2. Business Process Modeling Notation

There has been much activity in the past few years in developing execution languages for Business Process Management (BPM) systems. The key element of such languages is that they are optimized for the operation and interoperation of BPM Systems. The optimization of these languages for software operations renders them less suited for direct use by humans to design, manage, and monitor business processes. At the same time, business people are very comfortable with visualizing business processes in an informal flow-chart format. This creates a technical gap between the format of the initial design of business processes and the format of the languages that will execute these business processes [15].

Business Process Modeling Notation (BPMN) provides a Business Process Diagram (BPD), which is designed for use by the people who design and manage business processes. BPMN also provides a mapping to an execution language of BPM Systems. BPMN will also advance the capabilities of traditional business process notations by inherently handling business process concepts, such as public and private processes and choreographies, as well as advanced modeling concepts, such as exception handling, transactions, and compensation [15].

BPMN allows specifying organization's private (internal) and abstract (public) business processes as well as collaboration (global) processes. However, not all of them can be mapped to the executable

languages. Detailed private business processes can describe "as-is" or old business processes as well as "to-be" or new business processes. Besides that, BPMN enables describing of relationships from the detailed private business processes to both abstract and collaboration processes [15].

3.3. Use cases for business purposes

Being introduced by Ivar Jacobson, the term *use case* is widely known since 1987. Use cases have become a very popular technique of the OOA from the end of 1990s.
I. Jacobson et al. [16] offered the use case model as a means for requirements model representation. This model should specify all the functionality of the system and support traceability. But the use case model is only a part of the requirements model. The main aim of use case introducing was to improve traceability between the functional requirement specification and an analysis model since the requirements specification is usually represented in an only textual description, and "it was usual that one "forgets" requirements in the requirement specification". Therefore, in order to overcome this problem and improve traceability Jacobson et al. separated three parts of the requirements model that applies use cases:

- A use case model that defines what exists outwards the system and what should be done by the system,
- Interface description, and
- A problem domain object model.

All these three parts of the requirements model are application-oriented.
Use cases may be described in different levels of detail – it depends on the point of view use cases describe. There are three concepts associated with use cases – *actors*, *use cases* and the *subject*. The latter one defines the system-to-be. Actors always represent entities that are outward the subject, i.e. outward the system under consideration. One or more use cases might be used for specification of the required behavior of the system; use cases define offered behavior of the subject without reference to its internal structure, i.e. implement the so-called "black box" approach. Use cases, actors and systems are described with use case diagrams. The same use case can be associated to several subjects.
In time, software developers have distinguished system and business use cases. A business use case is a stereotype of a usual use case introduced for purposes of business modeling. Business use cases specify the functionality of the business in order to be used to drive application development. Actors are users and systems that interact with the business. A business object model related to the business use case model describes the entities – departments, paychecks, systems – and their interactions in order to provide the functionality necessary to the implementation of the business use cases [17].
The main attraction of use cases is their simplicity, informality and business-orientation. However, their main disadvantages are also founded in their informality -their usage is not systematic in comparison with systematic approaches, which allow identifying all of the system requirements. They do not give any answer on questions about completeness and unambiguousness of them, conflicts among them and change affects on other use cases.

3.4. Topological Functioning Model

A topological functioning model (TFM) was developed in 1960s at Riga Technical University by Janis Osis. **A topological functioning model** is represented in the form of a topological space (X, θ), where X is a finite set of functional features of the system under consideration, and θ is the topology that satisfies axioms of topological structures and is represented in the form of a directed graph. The necessary condition for creating the topological space is a meaningful exhaustive verbal, graphical, or mathematical description of the system. *Properties of the TFM* are topological (connectedness, closure, neighborhood, and continuous mapping) and functional (cause-effect relations, cycle structure, inputs and outputs). These properties set model capabilities such as formal separation of

subsystems, formal abstraction and refinement of models, and analysis of similarities and differences of functioning systems [18].
The common characteristic of functionality of all systems (technical, business, or biological) is ***feedback cycles.*** The TFM reflects such feedbacks as cycles of an oriented graph. The proper analysis of cycles is mandatory in creating the TFM, because it supports careful analysis of system's operation and interaction with its environment [2].
Construction of the TFM consists of the following steps: a) Definition of business functional characteristics – objects and their properties, external and partially-dependent systems, and functional features b) Introduction of the topology between functional features, and c) Separation of the TFM. This process is described in detail in [4], [5].
Each **TFM functional feature** is a unique tuple <*A, R, O, PrCond, PostCond, E, S*>, where:

- *A* is an object's action,
- *R* is a result of this action (optional),
- is an object or objects that receive the result or that is used in this action (for example, a role, a time period, a catalog, etc.),
- *PrCond* is a set $PrCond = f(c_1, \ldots, c_i)$, where c_i is a precondition or an atomic business rule (it is an optional element),
- *PostCond* is a set $PostCond = f(c_1, \ldots, c_i)$, where c_i is a post-condition (it is an optional element),
- *E* is an entity responsible for action performing; this can be an external or partially-dependent system;
- *S* is functional feature's belonging (subordination) to system's functionality (inner or external).

Each precondition, post-condition and atomic business rule must be either defined as a functional feature or assigned to the already defined functional feature. A form of the textual description is *<action>-ing [the <result>] [to,into,in,by,of,from] a(n) <object>, [PrCond, PostCond] E, S*. For example, "Checking out the availability of a copy, PrCond = {a valid reader account}, E is a librarian, S is inner".
Cause-and-effect relations are visualized as arcs of a digraph that are oriented from a cause vertex to an effect vertex. Each chain that cause-and-effect relations form must be checked whether it creates a cycle. The cycles of system functioning must be investigated before starting further analysis. There must be at least one cycle that describes main functionality of the system under consideration.
By mapping functional requirements onto the TFM of the system-as-is, it is possible to obtain formal business model of the system-to-be.
Application of the TFM has evident advantages:

- The TFM heightens a degree of formality within the CIM. Its functional and topological properties offer mathematics that is not very complex.
- Careful analysis of TFM cycles enables making a decision about acceptability of changes in problem domain's functioning before implementation of those changes.
- The TFM is a basis for a new formalized approach to creation of use cases. Using the TFM, some limitations of functional description by use cases can be solved, namely, information capturing, thinking limitation, and completeness checking. The latter means that the TFM provides completeness of use cases, avoids conflicts among use cases, and shows their affect on each other.

4. Model capabilities

All the discussed models are dedicated to be used for modeling of a business domain. However, it is interesting to analyze what parts of the CIM they reflect and what formal mechanisms they use to relate the domain "to be" to the domain "as is".
The defined features are explained in Table 1, and their values (based on studied specifications or research) are given for each model in Table 2. Table 1 defines the following common features related

to both domain “as is” and domain “to be”: determination of structural contents, dynamic contents, and boundaries. Additional features referred the domain “to be” concern different kinds of requirements, namely, functional, interaction requirements and environmental contracts. The remaining features – a part of the CIM, formal underpinnings and domain conformity illustrates inter-domain properties of the models.

Table 1

The list of features for analysis of model capabilities

Nr.	Feature Name	Feature Description
1	A part of the CIM	A part of the CIM that the model under consideration reflects
2	Formal underpinnings	Underlying formalism of the model
3	Structural contents of the domain “as is”	Specification constructs for the static content of the business domain
4	Dynamic contents of the domain “as is”	Specification constructs for the dynamic content of the business domain
5	Structural contents of the domain “to be”	Specification constructs for the static content of the application domain
6	Dynamic contents of the domain “to be”	Specification constructs for the dynamic content of the application domain
7	“As is” boundaries	The technique used for identification of the domain “as is” boundary
8	“To be” boundaries	The technique used for identification of the domain “to be” boundary
9	Functional requirements	The way of specification of the functional requirements
10	Interaction requirements	The way of specification of the interaction requirements
11	Environmental contracts	The way of specification of the non-functional requirements
12	Domains conformity	Conformity of requirements to the domain “as is”

Table 2 illustrates that only SBVR model may be used in all three parts of the CIM, but as a business requirements model it is limited with functional requirements. SBVR, BPMN, use case and TFM models are not able to express environmental contracts. BPMN, use case and TFM models cannot be used as CIM - Knowledge Model. Only BPMN and use cases can reflect interaction requirements.
All the models, excepting use cases, have or map to formal underpinnings. Only SBVR formally defines structural contents of domains. Only SBVR and TFM formally define domains’ boundaries and relationships between domains.

Table 2

Model capabilities

Nr.	Feature Name	SBVR	BPMN	Use cases	TFM
1	A part of the CIM	CIM Knowledge Model, CIM Business Model CIM Business Requirements for the System (Functional Requirements)	CIM Business Model, CIM Business Requirements for the System (Functional Requirements, Interaction Requirements)	CIM Business Requirements for the System (Functional Requirements, Interaction Requirements)	CIM Business Model, CIM Business Requirements for the System (Functional Requirements)
2	Formal underpinnings	semantic formulations, first-order logic, limited	mapping to pi-calculus	none	topology

		higher-order logic, modal logic			
3	Structural contents of the domain “as is”	business vocabulary (concepts and facts)	none	domain object vocabulary	domain object vocabulary
4	Dynamic contents of the domain “as is”	business rules	higher-level and detailed private BPD, abstract and collaboration BPD	a business process model or a business use case model	topological functioning model
5	Structural contents of the domain “to be”	business vocabulary (concepts and facts)	none	an analysis class diagram with inheritance, generalization, and multiplicities	a conceptual class diagram with undirected associations, inheritance and generalization derived from the TFM
6	Dynamic contents of the domain “to be”	business rules	higher-level and detailed private BPD, abstract and collaboration BPD	a use case model, activity or interaction diagrams, textual descriptions	topological functioning model (with mapping to a use case model, and activity diagrams), textual descriptions
7	“As is” boundaries	determined by the closed semantic formulations	initial estimation based on stakeholders’ opinions	determined by the chosen business model	topological closure operation
8	“To be” boundaries	estimation based on the gathered requirements and projection to the domain “as is”	intuitive estimation based on gathered requirements	initial intuitive estimation (further refined by actors and user case finding)	based on mapping direct users’ goals to the TFM of the domain “as is”
9	Functional requirements	constraints put by the business rules	BPD	a use case model	constraints put by requirements specification
10	Interaction requirements	none	BPD (if used with use-case driven techniques)	a use case model	none
11	Environmental contracts	none	none	none	none
12	Domains conformity	projections based on expert knowledge	intuitive or based on expert knowledge	intuitive or based on expert knowledge	continuous mapping to the TFM of the domain “as is”

5. Conclusions

This paper discussed a use of business models as a computation independent model within MDA. It was determined that the CIM contains three parts – the knowledge model, business model and business requirements for the system. Thus, a business model (in general sense) is only one part of the CIM.

Separation between a business model and business requirements illustrates that the CIM can be used to define the domain "as is" as well as the domain "to be", but making these two domains in conformity is still a challenge.

This paper has analyzed capabilities of four business models expressed in terms of SBVR, BPMN, use cases and TFM in order to determine what parts of the CIM they cover and how they relate the domain "to be" to the domain "as is". After defining twelve basic features and analyzing capabilities of these models, the results are as follows:

- None of four models completely covers all parts of the CIM. However, SBVR is less suitable for analysis of the domain "to be" that other models.
- Only use cases do not have even minimal formal foundations.
- Only SBVR uses its own constructs for definition of domain structure and behavior. Other models use additional, borrowed constructs.
- Only SBVR and TFM use formalism for identification of domain boundaries and for making conformity between them.

Therefore, the further research will be related to using together SBVR and TFM in order to get a formal and more complete CIM.

References

1. MDA Guide Version 1.0.1 / Object Management Group, 2003. - http://www.omg.org/cgi-bin/doc?omg/03-06-01. - 12th June 2003.
2. Osis J. Formal Computation Independent Model within the MDA Life Cycle // International Transactions on Systems Science and Applications. – Glasgow: Xiaglow Institute Ltd, 2006.- V. 1, Nr. 2, P.159 – 166.
3. Asnina E. The Formal Approach to Problem Domain Modelling Within Model Driven Architecture // Proceedings of the 9th International Conference "Information Systems Implementation and Modelling" (ISIM'06).- Přerov: Jan Štefan MARQ., 2006.- P.97-104.
4. Osis J., Asnina E., Grave A. Computation Independent Modeling within the MDA // Proceedings of IEEE International Conference on Software, Science, Technology & Engineering (SwSTE07), Herzlia: IEEE Computer Society, 2007. - P.22-34.
5. Osis J., Asnina E. Enterprise Modeling for Information System Development within MDA // Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008). – Washington, DC: IEEE Computer Society, 2008. - P.490.
6. Tekinerdogan B., Aksit M., Henninger F. Impact of Evolution of Concerns in the Model-Driven Architecture Design Approach // Proceedings of the Second International Workshop on Aspect-Based and Model-Based Separation of Concerns in Software Systems (ABMB 2006), Electronic Notes in Theoretical Computer Science. – Elsevier, 2007. - Volume 163, Issue 2. – P.45-64.
7. A Home for Business Models in the OMG (by Stan Hendryx) [Business Rules Journal] / BRCommunity, 2003. - http://www.BRCommunity.com/b127.php. - January 2003.
8. Architecture of Business Modeling (by Stan Hendryx) / Hendryx & Associates, 2003. – www.omg.org/docs/br/03-11-01.pdf. - November 14, 2003.
9. Term-Fact Modeling, the Key to Successful Rule-Based Systems (by Oscar Chappel) [Business Rules Journal] / BRCommunity, 2005. - http://www.BRCommunity.com/a2005/b250.html. - October 2005.

10. Integrating Computation Independent Business Modeling Languages into the MDA with UML 2 (by Hendryx S.) / Object Management Group, 2003. - http://www.omg.org/docs/ad/03-01-32.doc. - 2003.
11. Computation vs. Information Processing: How They Are Different and Why It Matters (by Gualtiero Piccinini, Andrea Scarantino) [Conference on Computation and Cognitive Science 2008] / King's College Cambridge, 2008. - http://people.pwf.cam.ac.uk/mds26/cogsci/program.html. - 2008.
12. Grangel R., Chalmeta R., Campos C. Using UML Profiles for Enterprise Knowledge Modelling // Proceedings of the 26th International Conference on Conceptual Modeling (ER 2007), the 3rd International Workshop on Foundations and Practices of UML (FP-UML 2007), LNCS, Computer Science, Theory & Methods. – Berlin: Springer Verlag, 2007. – P.125-132.
13. Semantics of Business Vocabulary and Rules (SBVR) V. 1.0. / Object Management Group, 2008. - http://www.omg.org/spec/SBVR/. – January 2008.
14. SBVR Overview (by John Hall) / MetaData Open Forum, 2007. - http://www.metadataopenforum.org/index.php?id=21,95,0,0,1,0. – June 2007.
15. Business Process Modeling Notation, V1.1 / The Object Management Group, 2007. - http://www.omg.org/spec/BPMN/1.1/PDF. - January 2008.
16. Object-Oriented Software Engineering: A Use Case Driven Approach / Jacobson I., Christerson M., Jonsson P., Overgaard G. – New York: Addison-Wesley, 1992. – P.528.
17. Leffingwell D., Widrig D. Managing Software Requirements: a use case approach. 2nd ed. – New York: Addison-Wesley, 2003. - P.502.
18. Osis J. Software Development with Topological Model in the Framework of MDA // Proceedings of the 9th CaiSE/IFIP8.1/EUNO International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'2004) in connection with the CaiSE'2004 – Riga: Riga Technical University, 2004. – Vol. 1, P.211-220.

Erika Asnina, Dr.sc.ing., lecturer, Riga Technical University, Meza 1/3, Riga, LV 1048, Latvia,
erika.asnina@rtu.lv

Asņina Ē. Biznesa modeļu lietošana modeļvadāmas arhitektūras ietvaros

Modeļvadāmas arhitektūras (MDA) ietvars ir paredzēts lielu un sarežģītu sistēmu izstrādei. Tas nosaka un realizē uzskatu arhitektoniskās atdalīšanas principu. Tas nozīmē, ka sistēma var būt modelēta no trīs dažādiem, bet saistītiem, skatījumiem. Skatījums, kas ir izskatīts šajā rakstā, tiek saukts par no skaitļošanas neatkarīgu. MDA specifikācija noteic, ka modelis, kas parāda sistēmu no šā skatījuma, tiek saukts par biznesa modeli. Ņemot vērā MDA paredzētās transformācijas starp skatījumiem, tam vajadzētu būt noderīgam programmatūras izstrādes procesu automatizācijai. Šajā rakstā ir diskutēts par no skaitļošanas neatkarīga modeļa (CIM) būtību, un biznesa modeļu vietu no skaitļošanas neatkarīgā modelēšanā. Šajā rakstā ir izskatīti četri biznesa modeļu tipi, proti, SBVR, BPMN, lietošanas gadījumi un Topoloģiskais Funkcionēšanas Modelis (TFM). Biznesa personas izmanto SBVR, lai definētu biznesa vārdnīcas un biznesa likumus eksistējošam un plānotam domēnam; BPMN izmanto, lai definētu biznesa procesus gan eksistējošam, gan plānotam domēnam; un lietošanas gadījumus izmanto, lai definētu biznesa prasības plānotam domēnam. Savukārt, TFM izmanto, lai definētu eksistējoša un plānota domēna funkcionalitāti. Šajā rakstā ir izskatītas šo modeļu spējas pilnīgi aprakstīt CIM modeli, kurā plānotā domēna atbilstība eksistējošajam domēnam būtu formāli definēta.

Asnina E. Use of Business Models within Model Driven Architecture

Model Driven Architecture is a framework dedicated for development of large and complex computer systems. It states and implements the principle of architectural separation of concerns. This means that a system can be modeled from three different but related to each other viewpoints. The viewpoint discussed in this paper is a Computation Independent one. MDA specification states that a model that shows a system from this viewpoint is a business model. Taking into account transformations foreseen by MDA, it should be useful for automation of software development processes. This paper discusses an essence of the Computation Independent Model (CIM) and the place of business models in the computation independent modeling. This paper considers four types of business models, namely, SBVR, BPMN, use cases and Topological Functioning Model (TFM). Business persons use SBVR to define business vocabularies and business rules of the existing and planned domains, BPMN to

define business processes of both existing and planned domains, and use cases to define business requirements to the planned domain. The TFM is used to define functionality of both existing and planned domains. This paper discusses their capabilities to be used as complete CIMs with formally defined conformity between planned and existing domains.

Аснина Э. Использование бизнес-моделей в управляемой моделями архитектуре
Управляемая моделями архитектура (MDA) - это подход, предназначенный для разработки больших и сложных систем. Он определяет и реализует принцип архитектурного разделения интересов. Это означает, что можно создать модель системы, рассматривая её с трёх разных, но взаимосвязанных, точек зрения. Рассматриваемая в данной статье точка зрения называется независимой от вычислений. В спецификации MDA определено, что модель, рассматривающая систему с данной точки зрения, называется бизнес-моделью. Принимая во внимание предусмотренные MDA трансформации, она должна быть пригодна для автоматизации процессов разработки программного обеспечения. В данной статье обсуждается сущность независимой от вычислений модели (CIM) и место бизнес-моделей в независимом от вычислений моделировании. В статье рассматриваются четыре бизнес-модели, а именно, SBVR, BPMN, прецеденты использования и Топологическая Модель Функционирования (ТФМ). Представители бизнеса используют SBVR для определения бизнес-терминологии и бизнес-правил в существующем и планируемом доменах; BPMN - для определения бизнес-процессов в существующем и планируемом доменах; а прецеденты использования - для определения требований бизнеса к планируемому домену. ТФМ используется для определения функциональности существующего и планируемого доменов. В данной статье обсуждаются способности рассмотренных моделей быть использованными в качестве полной модели CIM, в которой формально определено соответствие планируемого домена уже существующему.