ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2011, Vol.40, No.3

ENTERPRISE KNOWLEDGE BASED SOFTWARE REQUIREMENTS ELICITATION

Aurelijus Morkevičius[1], Saulius Gudas[1, 2]

[1]*Kaunas University of Technology, Faculty of Informatics, Information Systems Department,*
Studentu St. 50-313a, LT-51368 Kaunas, Lithuania,
e-mail: aurelijus.morkevicius@stud.ktu.lt,
[2]*Vilnius University, Kaunas Faculty of Humanities,*
Muitines St. 8, LT-44280 Kaunas, Lithuania,
e-mail: gudas@vukhf.lt

crossref http://dx.doi.org/10.5755/j01.itc.40.3.626

Abstract. One of the ways to capture enterprise knowledge is Enterprise Architecture (EA). EA allows indentifying the majority of software "to-be" requirements for information systems (IS) engineering. However, the transition between enterprise architecture model and IT resource design still lacks a clear approach and tools for implementing it in practice. The paper presents an approach for the enterprise knowledge based software requirements elicitation. The suggested approach is based on the Unified profile for Ministry of Defence Architecture Framework (MODAF) and Department of Defense Architecture Framework (DoDAF), System Modeling Language (SysML) requirements model, and a Semantics of Business Vocabulary and Business Rules (SBVR) standard as a formal background for elicited software requirements. A real world example is presented to validate the suitability of the approach.

Keywords: UPDM, SysML, Enterprise Architecture, SBVR, Requirements Elicitation, Enterprise Knowledge, Software Requirements Specification.

1. Introduction

Enterprise Architecture (EA) has been a hot topic since 1980-ies [1]. However, it was not very widely applied in practices due to lack of modeling languages and tools suitable for EA [1]. The EA movement was reinforced with the successful adoption of the Unified Modeling Language (UML) [2]and the Model-Driven Architecture (MDA) [3]. There have been multiple attempts to apply Unified Modeling Language (UML) for Enterprise Architecture (EA) modeling [4], but many modelers found it too complicated and non-natural for solving their domain-specific problems [5]. In 2005, the Unified profile for MODAF and DoDAF (UPDM) initiative has been started in Object Management Group (OMG), but the first version of UPDM was released only in 2009, four years later [6]. As soon as the UPDM has been officially released, US Department of Defense mandated UPDM as Information Technology Standard and Profile Registry (DISR) standard. As UPDM is a profile of UML, it has been easily adopted by the majority of UML tool vendors. The versatility of UML and its compatibility with its profiles allows integrating UPDM with the other OMG standards based on UML, such as System Modeling Language (SysML), Service Oriented Architecture Modeling Language (SoaML), etc [7]. This enables creating large and versatile EA models to preserve enterprise knowledge [8] in order to solve a range of problems: business transformation into knowledge-based business, business and IT alignment, and the computerization of business management tasks [9].

Enterprise Knowledge allows indentifying the majority of software "to-be" requirements. However, first of all, enterprise knowledge models should be verified in accordance to the meta-knowledge of the enterprise. In case correctness and completeness of the enterprise knowledge models are sufficient enough, software requirements can be elicited or existing software requirements can be verified in accordance to enterprise knowledge. We can state that the enterprise knowledge can be either:

- A source for requirements elicitation [10],
- A constraint for requirements verification.

The purpose of this paper is to propose a new technique for software requirements elicitation from preserved enterprise knowledge.

The subject of the research is the software requirements model based on preserved enterprise knowledge. The rest of this paper is structured as follows: in section 2, the related works are analyzed; in section 3,

the proposed approach is presented; in section 4, experimental evaluation of the proposed approach on a small real world EA model is described; in section 5, the achieved results, conclusions, and future work directions are indicated.

2. Related Work

Several authors have emphasized the importance of enterprise modeling before requirements elicitation [11, 12 13]. Enterprise models capture the structural and the behavioral aspects of an organization and are very useful in helping software engineers properly understand the organizational environment and the requirements that the information system must fulfill.

Requirements elicitation techniques are described, analyzed, classified and evaluated by Zheying Zhang in [14]. The author separates conversational, observational, analytical and synthetic methods. We will focus only on analytical model based techniques providing ways to explore the existing enterprise knowledge. According to the observation of related works, we have classified such techniques into solution oriented such as based on UML, BPMN and problem oriented based on KAOS, i*, EDK, Control View (Figure 1).

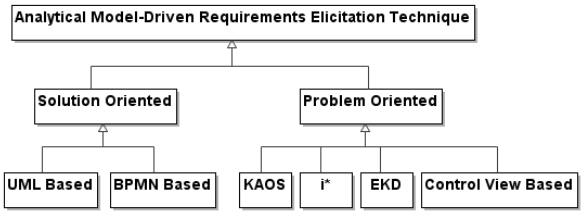

Figure 1. The taxonomy of analytical software requirements elicitation techniques

Authors using UML are De la Vara [15], Eriksson [16], Silingas [5] and Marshall [17].

Software requirements elicitation method from business process models based on the BPMN notation and UML use case model has been proposed in [15]. The approach transforms business process model to the goal tree. Goal tree leaves are then transformed to use cases and presented as the requirements for the software system. However the main objective of this proposal is the extension of OO-Method 0, which is a methodology for automatic software generation based on conceptual modeling (solution oriented approach). We are mostly focusing to the problem orientted approach; to the requirements elicitation where human factor matters such as i* 0, KAOS [19], EKD [20], and Control View based [21].

Another attempt to extract functional software requirements from the enterprise model has been proposed in [22]. The approach aims at the use case model extraction from business function described using Enterprise Meta-Model (EMM). However, the approach focuses only on functional user-level requirements; the same as Control View Based Elicitation of Functional Requirements proposed in [21]. The approach in [21] is the knowledge-based (KB) and concerns usage of formally defined management and control view based structural component of Enterprise model. KB construct of Enterprise model is formally defined in [23] as Elementary Management Cycle (EMC), and comprises management information transformations of any Enterprise management function. The semantics of Enterprise management function correlates with the definition of Primary Activity of the M. Porter's Value Chain Model [9]. According to [21], the KB requirements elicitation methods should be related with the KB enterprise modeling techniques.

The KAOS methodology is aimed at supporting the whole process of requirements elaboration, starting from the high-level goals to be achieved and finishing with the requirements, objects, and operations to be assigned to the various agents in the composite system. Each construct in the KAOS language has a two-level generic structure: an outer semantic net layer for declaring a concept, its attributes, and its links to other concepts; an inner formal assertion layer for formally defining the concept [24]. The declaration level is used for conceptual modeling (through concrete graphical notation), requirements traceability (through semantic net navigation), and specification reuse (through queries) [19]. The assertion level is usually written in a real-time temporal logic described in [25].

Tropos has also an associated formal specification language called Formal Tropos (FT) for adding constraints, invariants, pre- and post- conditions capturing more of the subject domain's semantics to the graphical models in the i* notation [26]. FT has been designed to supplement i* models with a precise description of their dynamic aspects. In FT, the focus is not only on the intentional elements themselves, but also on the circumstances in which they arise, and on the conditions that lead to their fulfillment [27].

We think both the KAOS temporal logic and the FT languages are barely readable by humans other than IT experts. This limits the awareness of the stakeholders of the aspects that are not captured in graphical KAOS and i* notation. For this reason we have discovered a human and machines readable format that allows capturing software requirements in a formal way which is called SBVR and is defined in [28].

Semantics of Business Vocabulary and Business Rules (SBVR) standard is targeted to capture business concepts and business rules in a language close enough to ordinary language (such as Structured English) to permit business people to read them, and at the same time formal enough (based on predicate and common logic) to be suitable for interchanging among software tools. Formalized methods such as SBVR could ensure avoiding of many mistakes when applying it in practice for IS engineering [29].

There are multiple attempts to capture enterprise knowledge expressed in SBVR in UML models

supplemented with OCL constraints [30, 31, 32] and in BPMN models [31]. However only in [30, 33] authors deal with the behavioral aspect by transforming SBVR rules to the BPMN process, UML activity, and UML sequence models. Though our work is focusing on reverse transformation, the ideas presented in [30, 31, 32, 33] are captured and reused where applicable in this paper.

Further the requirements elicited can be a source for the automated business rules transformation to software design models realizing the MDA transformation of the context independent model to the platform independent model. However this is out of the scope for this paper.

3. Requirements Elicitation from Enterprise Models

Standards, patterns, and techniques for the suggested approach of requirements elicitation from preserved enterprise knowledge are discussed in this section.

3.1. Enterprise Knowledge

Requirements elicitation is recognized as one of the most critical, knowledge-intensive activities of software development [34]; poor execution of elicitation will almost guarantee that the final project is a complete failure. The early requirements analysis or in other words requirements elicitation process as a possible source of Software Requirements states domain experts, documents, existing system, etc. According to Wiegers, documents describe corporate or industry standards that must be followed or regulations and laws with which the product must comply. Descriptions of both present and future business processes also are helpful [10]. The other wider concept used to describe what Wiegers described as sources for requirements specification is domain knowledge. There are multiple ways to record domain knowledge within enterprise. One of the ways to capture domain knowledge is Enterprise Architecture. Schekkerman defines Enterprise Architecture as “an emerging approach for capturing complex knowledge about organizations and technology” [35]. It is obvious that an EA is the source of knowledge for requirements elicitation process.

To be more concrete (Figure 2), EA usually captures enterprise goals, business processes within an enterprise, business rules, business and IT objects. Goals can be directly translated to requirements. If we are looking deeper into requirement tree, goals can even be either functional, or non-functional software requirements [36, 37]. Processes, as we will take a more detailed look later in this paper, can also be directly translated to software functional requirements, for example, in a form of use cases or enterprise goals [15]. Business rules are both the non-functional and functional requirements for the software [10].

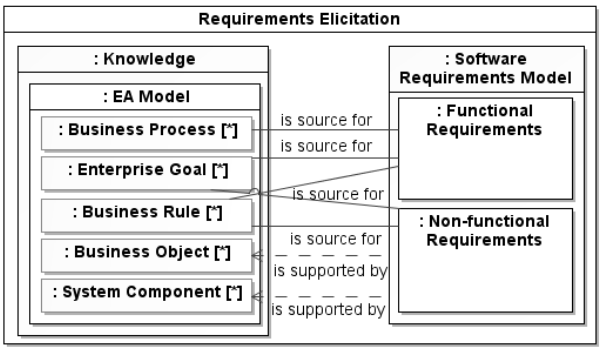

Figure 2. Software requirements elicitation form EA model

Business objects and IT objects act as supportive entities for requirements elicitation from enterprise knowledge models. For example, business objects could be the actors for the derived use case model and IT components can be used for requirements grouping.

The quality of the process of software requirements model derivation directly depends on the quality of knowledge. Quality of knowledge depends on the quality of meta-knowledge. To ensure quality of meta-knowledge it should be validated according to the domain knowledge. It might be done by evaluation of domain experts. Quality of meta-knowledge can be also ensured by picking the best practice configuration within the domain.

3.2. UPDM based Enterprise Model

The Unified Profile for MODAF and DoDAF (UPDM) defines a set of UML and optional SysML stereotypes and model elements and associations. UPDM is not an architectural framework. It is a language dedicated to build MODAF and DoDAF requirements meeting enterprise architecture models [6].

We will use UPDM language based models to demonstrate our proposed technique. The main reason for the choice of UPDM is UML, especially when we are using SysML; other UML based modeling language. The integrity of both would make the starting point easier; integrated OMG metamodels allow designing various enterprise systems in one CASE tool [38]. The other obvious reason is the support of DoDAF and MODAF: emerging enterprise architecture frameworks. However the proposed approach is planned to be extended to meet TOGAF and Archimate requirements in the future.

UPDM consists of seven viewpoints: all views, strategic, acquisition, operational, service oriented, systems and technical [6]. We will focus on operational and systems viewpoints. A suite of operational viewpoint products are used to describe a requirement for a to-be architecture in logical terms. Systems viewpoint specifies a requirement for a System without delving deep into systems design [39].

Software requirements engineering usually separates two or more types of functional requirements

according to their origin. Authors in [10, 40], and [41] separate requirements into user requirements (user needs in [41]) and system requirements. System requirements are also called as functional [10] or simply software requirements [7]. Based on this classification and the definition of the DoDAF viewpoints, we can state that the Operational viewpoint model is the source to elicit user requirements and the systems viewpoint model is the source to elicit system requirements.

UPDM closely enough to other enterprise modeling languages defines the concepts of operational activity and function. Operational activity is performable by logical entity called Node and the function by the physical entity called Resource. It is also important to note that both the operational activities and the functions consume and produce the information we need to capture during the requirements elicitation from the enterprise model.

3.3. SBVR Transformation Patterns

Object Management Group (OMG) has published the Semantics of Business Vocabulary and Business Rules (SBVR) specification [28] that defines the metamodel for documenting the semantics of business vocabulary, business facts and business rules. SBVR is targeted to capture business concepts and business rules in a language close enough to ordinary language (such as Structured English) to permit business experts to read them, and at the same time formal enough (based on predicate and common logic) to be suitable for interchanging among organizations or software tools. Thus, SBVR is specially suited for acting as an intermediate representation between the stakeholders and the designers.

Enterprise model transformation to human readable SBVR format consists of two phases:

- Transformation of user (operational in defense frameworks) requirements, and
- Transformation of system requirements.

The proposed approach does not support full transformation of the model yet. Currently it deals with activities, information flows, activity performing entities, and data ignoring state and interaction models of the performing entities.

Suggested patterns are formed according to [31, 33, 30, 31]. Patterns are common for both the operational model and the systems model of the enterprise. A single unit of transformation is activity (an operational activity or a function dependent on the enterprise viewpoint). In UPDM, activity is performed by performer [6]. It also produces and consumes exchange items and participates in the sequential activity flow model usually comprised of more than one activity. These aspects semantically differentiate UPDM activity models from the UML ones.

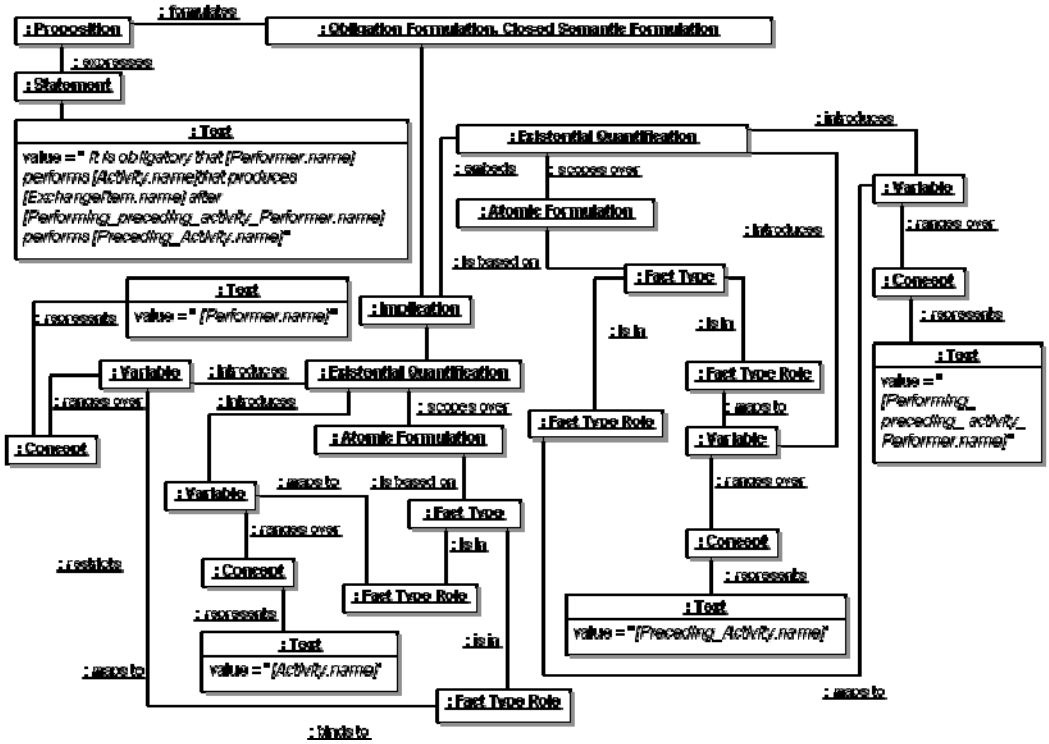

Figure 3. The example of the transformation pattern as the instance of SBVR MOF metamodel

During the transformation activities, performers and exchange items are transformed to concepts, performs relationship is transformed to the fact type "<performer> performs <activity>", consumption of exchange item is transformed to the fact type "<activity> consumes <data>", and production of the

exchange item is transformed to the fact type "<activity> produces <data>".

Using basic set of logical operations provided in SBVR standard there is no way to express first order logic such as before/after relationship. In order to involve temporal meanings the introduction of new keyword *after* is required into SBVR structured English [33]. The pattern supporting introduced keyword *after* is provided bellow.

`after` <*antecedent*> `then` <*consequent*>.

Note that both <antecedent> and <consequent> are binary fact types and both *after* and *then* are keywords [33].

Activity preceding the transformed activity is added as the antecedent in the SBRV statement. In case only the start node is before the activity, after/then pattern won't be used. Complete SBVR structured English pattern for the activity producing an exchange item is:

```
[Obligation formulation]
[Performer.name] performs
[Activity.name] that produces
[ExchangeItem.name] after
[Performing_preceding_activity_Perfo
rmer.name] performs
[Preceding_Activity.name]
```

SBVR sentences are written in a different font and color to make it easier to read. The "`Performer.name`" font is devoted for object types; the "*performs*" font renders fact types and "after" font is for keywords. The notation has been taken from [28].

As an instance of the SBVR MOF-based metamodel, the rule pattern representing the activity that does not produce and does not consume an exchange item is displayed in Figure 3. Note that there is only a core fragment of the pattern displayed: fact type forms are not shown.

Activities consuming exchange items are transformed in the same manner. In case the activity is producing and consuming an exchange item, both clauses will be added and separated by the logical conjunction operation.

Sample SBVR based statement definition is:

```
It is obligatory that the
administration performs send_invoice
that produces the invoice after the
administration performs
prepare_invoice
```

Activity that is not consuming and producing an exchange item will be transformed using the following pattern:

```
[Obligation
formulation][Performer.name]
performs [Activity.name] after the
[Performing_preceding_activity_Perfo
rmer.name] performs [Preceding_Ope-
rationalActivity.name]
```

Sample SBVR based requirement definition is:

It is obligatory that the administration *performs* cancel_order after the administration *performs* receive_order.

In both cases above, activity control nodes such as decision, merge, fork and join are also supported. In case of decision before the activity, guard condition is added as the fact type underlying logical conjunction in combination with the preceding activity fact type.

In case of merge node, number of preceding activities is added in the logical disjunction operation.

In case of join node, number of preceding activities is added in the logical conjunction operation.

In case of the fork node, operational activities following the node are transformed to two or more sentences separated by the logical conjunction operations.

3.4. SysML Requirements Model

SysML is a systems modeling language based on UML [42]. Both SysML and UML languages are based on the same metametamodel, the OMG Meta Object Facility (MOF) [43]. SysML is considered both a subset and an extension of UML [42].

Besides other features we will not discuss in this paper, SysML provides multiple ways for capturing requirements and their relationships, in both graphical (SysML Requirements diagram) and tabular notations. A requirement that is captured in text is represented in SysML using the Requirement model element [44]. It can be related to other model elements through a set of relationships: containment, derive, satisfy, verify, refine and trace [42]. Relationships used in our approach are briefly explained below.

A containment relationship in UML expresses ownership and is very rarely represented in UML diagrams. In SysML it is frequently used to express requirements hierarchy. Meanwhile, a deriveReqt (derivation) relationship is a dependency between two requirements in which a client requirement can be derived from the supplier requirement. For example, a system requirement may be derived from a user requirement, or a business requirement etc [42]. We will also use refine relationship to establish traces between requirements and the behavioral architecture elements that the requirement is directly elicited from.

SysML Requirements model in the scope of the proposed requirements elicitation technique is the repository for the generated SBVR sentences. For each SBVR sentence requirement model element is created. Requirement text property represents the SBVR structured English sentence. Activities that are of higher abstraction level (non-atomic) within the enterprise model will be converted directly to grouping requirements. The name of the grouping requirement is the name of the process or function and the text is

"Requirements for <Operational Activity / Function. name> Business Process / Application Function". The grouping requirement contains other requirements transformed from lower level activities. Only the requirements representing atomic activities are representing SBVR sentences. Such requirements will always be the leaf requirements in the requirements tree.

Derivation relationship will be created between user requirements and system requirements in case the application function represented by system requirement implements the process represented by user requirement in the UPDM based enterprise model.

3.5. The process of Requirements Elicitation from Enterprise model

As all the techniques required for requirements elicitation from enterprise model have been discussed separately, it requires a clear workflow definition of how to associate these techniques together to achieve the desired results.

The process of applying the proposed approach consists of several steps:

1. Verify enterprise model to make sure the model is correct and complete. The technique for Enterprise Model verification will not be discussed in this paper. The model we use in the experimental section is treated as verified.
2. Identify processes for transformation. The technique will not be discussed in this paper.
3. Transform Business Processes to SBVR model. The transformation will be made according to the proposed transformation patterns.
4. Transform SBVR model to structured English.
5. Build requirements tree using SysML requirements model and notation.
6. Identify Application Functions for transformation.
7. Transform Application Functions to SBVR model. The transformation will be made according to the proposed transformation patterns.
8. Transform SBVR model to structured English.
9. Build software requirements tree using SysML requirements model and notation.
10. Add traces between different layers of requirements and requirements and architecture elements.
11. Generate software requirements specification (SRS) document according to the predefined report template.

Summarizing the process of the proposed approach step by step, overview diagram is provided in Figure 4 that covers the process of eliciting user and system requirements from the enterprise architecture model.

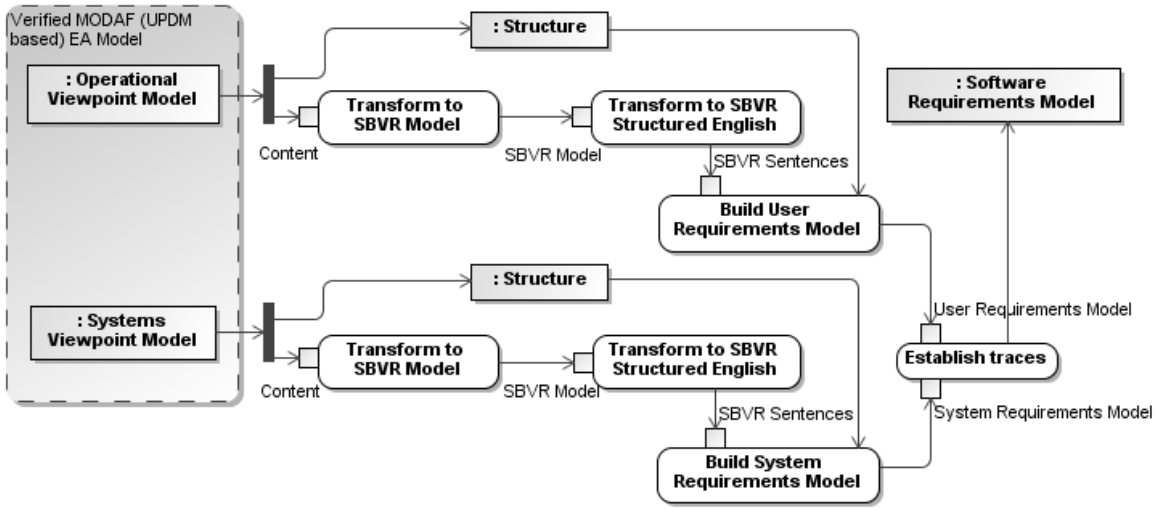

Figure 4. The process of requirements elicitation from the enterprise architecture model

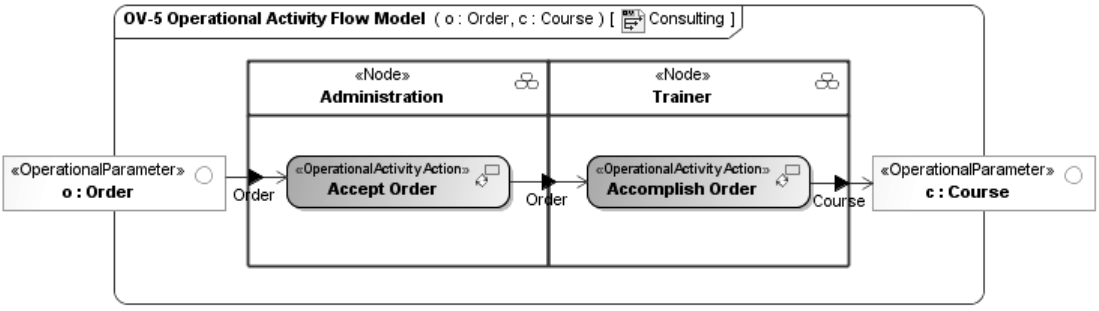

Figure 5. The process of consulting

4. Experimental Evaluation

Let us define a simple operational and systems viewpoint fragments of UPDM based Enterprise Architecture in order to demonstrate the proposed enterprise knowledge based software requirements elicitation approach.

The given EA fragment represents IT consulting enterprise. The consulting process shown in Figure 5 is defined as operational activity composed of two atomic operational activities *accept order* and *accomplish order*. Consulting operational activity has one input and one output. The input is *order* and the output is *course*; both representing information. Atomic operational activity accept order is performed by Administration business entity and the operational activity accomplish order is performed by the Trainer.

Accept order operational activity is implemented by function called *Accept order* in the systems viewpoint. The function is decomposed into multiple atomic Functions in Figure 6.

EA fragment, according to the defined SBVR transformation patterns and following the defined process steps, is transformed to the SysML requirements model:

- First, the *accept order* and *accomplish order* operational activities are transformed to user Requirements. The SysML user requirements model containing SBVR sentences is shown in Figure 7.
- Second, the *accept order* function and only its sub functions performed by the *Course Management IS (CMIS)* are transformed to system Requirements. The SysML system requirements model containing SBVR sentences is shown in Figure 7.
- Third, the derivation relationships are created between related user and system requirements.
- Finally the refine relationships are created between the requirements and the operational activity/function the requirement has been elicited from.

Functions performed by the human resource *Manager* are not transformed to the requirements model directly; however, the functions are used when transforming following functions performed by *CMIS*.

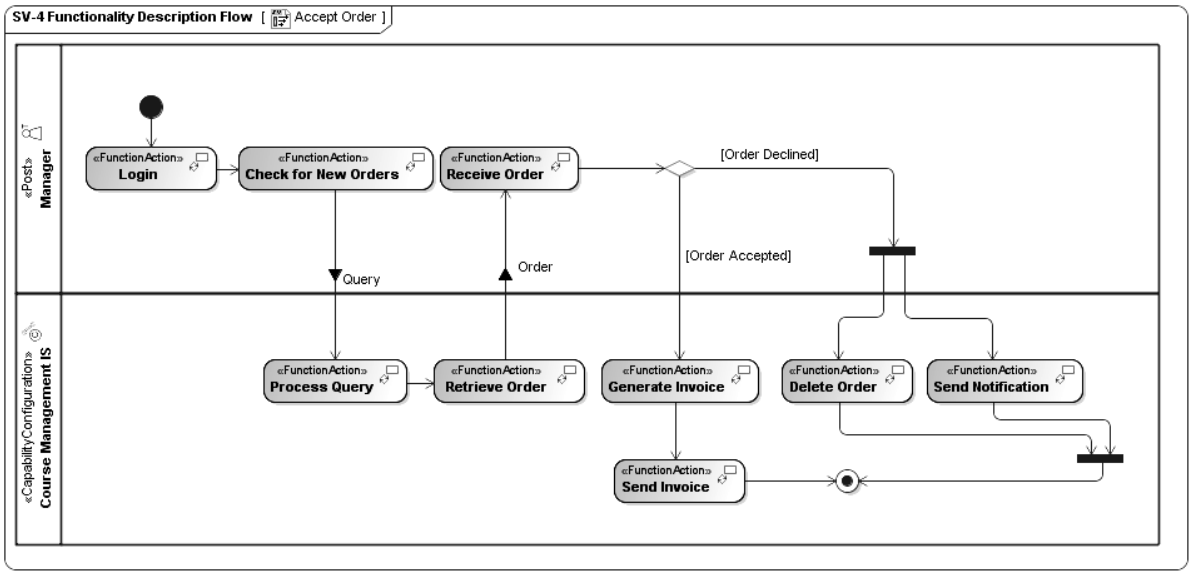

Figure 6. *Accept order* functionality description diagram

5. Conclusions and Future works

We have presented an approach for requirements elicitation from EA models based on a new UPDM standard. Elicited requirements are based on the emerging SBVR standard. SysML Requirements model is generated and SysML Requirements Diagram is used to visualize it. The proposed approach has been implemented in MagicDraw and has been evaluated on a small illustrative fragment of real world EA model.

Based on the experience on implementing and evaluating the proposed approach, we can make the following conclusions:

- An approach currently enables user and system requirements elicitation from verified UPDM based EA models; business requirements elicitation from the strategic aspect of the architecture will be considered supporting in the future.

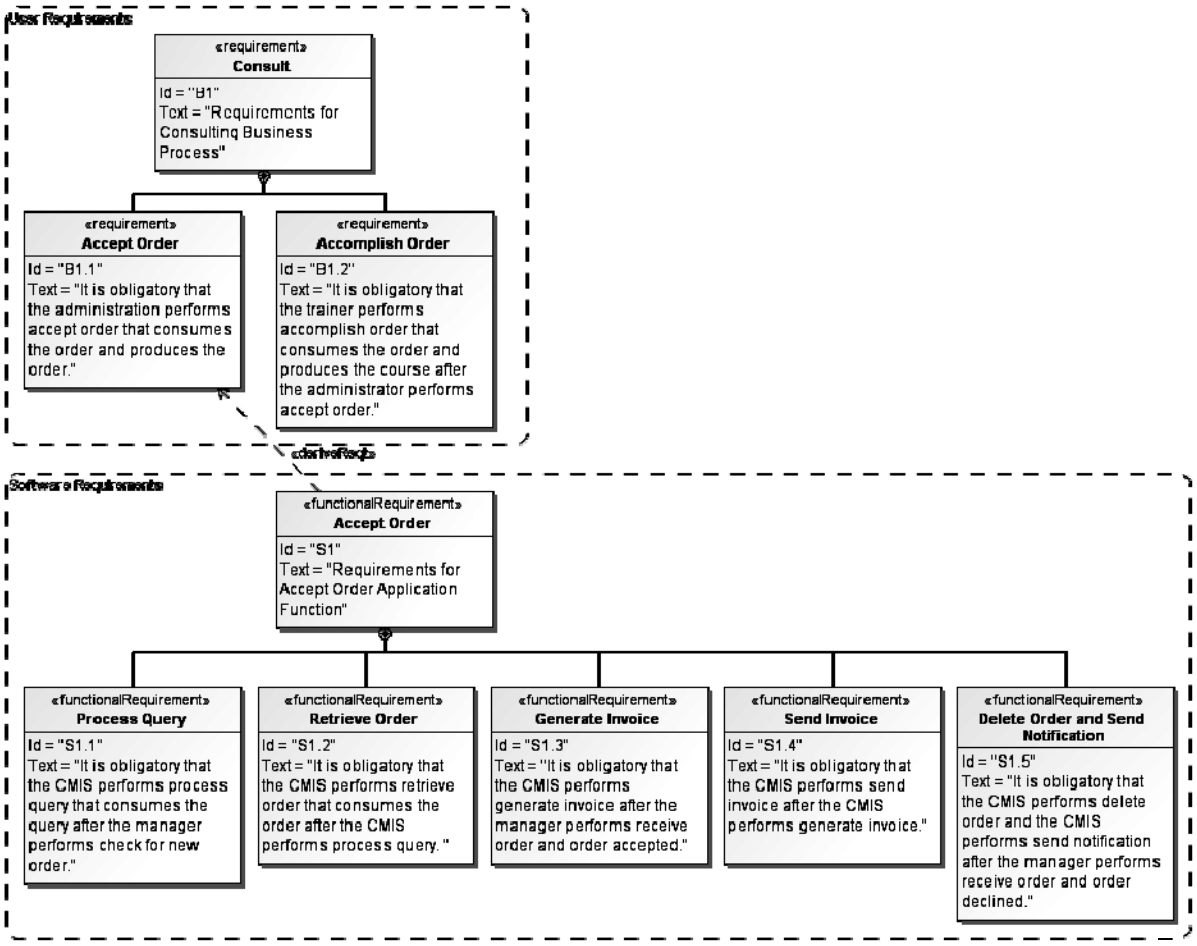

Figure 7. Transformation outcome: SysML user and system requirement models

- Elicited Requirements are stored into the same UML based repository the UPDM models are. Such an integrated repository ensures integrity and traceability between the architecture models and several different abstraction layers of software requirement models; integrated UML based repository is supported by the most of CASE tools.
- Compared to others, the proposed approach is both human and machine readable; readability by machines allows performing automated software design generation directly from the requirements model (MDA).
- The proposed approach currently elicits requirements from behavioral activity based UPDM models only; support of data models, business rules and state machines will be considered in the future.
- Verification of enterprise model and techniques of process categorization are subjects to the future works improving the proposed approach.

The proposed approach is a starting point for the more detailed future works on performing the requirements elicitation from preserved enterprise knowledge.

Acknowledgement

The authors would like to thank No Magic, Inc, especially the MagicDraw product team and professor Lina Nemuraite from Kaunas University of Technology for the comprehensive support**.**

References

[1] **J.A. Zachman.** A Framework for Information Systems Architecture. *IBM Systems Journal*, 26(3), 1987, 276-292.

[2] **OMG.** Unified Modeling Language (OMG UML) Infrastructure. v2.1.2. *Needham, MA, USA: Object Management Group*, 2007.

[3] **OMG.** MDA Guide Version 1.0.1. *Object Management Group, Needham, MA, USA: Object Management Group*, 2003.

[4] **M. Dalgarno, M. Fowler.** UML vs. Domain-Specific Languages. *Methods and Tools* , 16(2), 2008, 2-8.

[5] **D. Silingas, R. Butleris.** Towards customizing UML tools for enterprise architecture modeling. *Information Systems* 2009: *proceedings of the IADIS international conference, Barcelona, Spain*, 2009, 25-27.

[6] **OMG.** Unified Profile for the Department of Defense Architecture Framework (DoDAF) and the Ministry of Defence Architecture Framework (MODAF).

Needham, MA, USA: Object Management Group, 2009.

[7] D. Šilingas, R. Butleris. Towards Implementing a Framework for Modeling Software Requirements in MagicDraw UML. *Information Technology and Control*, 38(2), 2009, 153 – 164.

[8] A. Morkevicius, S. Gudas, D. Silingas. Model-Driven Quantitative Performance Analysis of UPDM-Based Enterprise Architecture. *Proceedings of the* 16*th International Conference on Information and Software Technologies.* Kaunas, 2010, 218-223.

[9] S. Gudas. Enterprise knowledge modelling domains and aspects. *Technological and economical development of Economy*, 15(2), 2009, 281-293.

[10] K.E. Wiegers. Software Requirements, Second Edition. *Redmond, Washington: Microsoft Press*, 2003.

[11] E. Kavakli, P. Loucopoulos. Goal Modeling in Requirements Engineering: Analysis and Critique of Current Methods. *Information Modeling Methods and Methodologies*, 2004, 102-124.

[12] G. Regev, A. Wegmann. Defining Early IT System Requirements with Regulation Principles: The Lightswitch Approach. *Proc.* 12*th IEEE International Requirements Engineering Conference* (RE'04*)*, 2004.

[13] E. Yu. Towards Modelling Strategic Actor Relationships for Information Systems Development – With Examples from Business Process Reengineering. *Proceedings of the* 4*th Workshop on Information Technologies and Systems, Vancouver, B.C., Canada*, 1994, 21-28.

[14] Z. Zheying. Effective Requirements Development - A Comparison of Requirements Elicitation techniques. *INSPIRE, Tampere, Finland*, 2007.

[15] J. L. De la Vara González, J.S. Díaz. Business process-driven requirements engineering: a goal-based approach. *Proceedings of the* 8*th Workshop on Business Process Modeling*, 2007.

[16] H. Eriksson, M. Penker. Business Modeling with UML: Business Patterns at Work. *John Wiley and Sons*, 2000.

[17] C. Marshall. Enterprise Modeling with UML. *Addison-Wesley*, 2001.

[18] E. Yu. Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering. 3*rd International Symposium on Requirements Engineering* (RE'97), Washington, USA. 1997.

[19] R. Darimont, E. Delor, P. Massonet, A. Van Lamsweerde. GRAIL/KAOS: An Environment for Goal-Driven Requirements Engineering. *Proc. ICSE*'98 – 20*th Int'l Conf. Software Eng., Kyoto, Japan*, 1998, 58-62.

[20] S. Zaharova, J. Stirna. Using Organizational Patterns as a Technique for Knowledge Management. 5*th CAiSE/IFIP*8.1 *International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD*'00*), Stockholm, Sweden*, 2000.

[21] A. Lopata, S. Gudas. Control View Based Elicitation of Functional Requirements. *Business Information Systems Workshops, BIS* 2009 *International Workshops, Poznan, Poland*, April 27-29, 2009, *Revised Papers Series: Lecture Notes in Business Information Processing*, 2009.

[22] S. Gudas, A. Lopata. Meta-Model Based Development of Use Case Model for Business Function. *Information Technology and Control*, 36(3), 2007, 302-309.

[23] S. Gudas. A framework for research of information processing hierarchy in enterprise. *Mathematics and Computers in Simulation*, 33, 1991, 281-285.

[24] A. Van Lamsweerde, R. Darimont, E. Letier. Managing Conflicts in Goal-Driven Requirements Engineering. *IEEE Transactions on Software Engineering*, 24(11), 1998, 908-926.

[25] R. Koymans. Specifying Message Passing and Time-Critical Systems with Temporal Logic. *Springer-Verlag*, 1992.

[26] A. Fuxman, M. Pistore, J. Mylopoulos, P. Traverso. Model Checking Early Requirements Specifications in Tropos. 5*th International Symposium on Requirements Engineering (RE'*01*).* Toronto, Canada. 2001.

[27] A. Fuxman, L. Liu, J. Mylopoulos, M. Pistore, M. Roveri, P. Traverso. Specifying and analyzing early requirements in Tropos. *Requirements Engineering*, 9(2), 2004, 132-150.

[28] OMG. Semantics of Business Vocabulary and Business Rules (SBVR), v1.0. *Needham, MA, USA: Object Management Group*, 2008.

[29] S. Gudas, A. Lopata. Workflow Models Based Acquisition of Enterprise Knowledge. *Information Technology and Control*, 36(1A), 2007, 103-109.

[30] L. Ceponiene, L. Nemuraite, G. Vedrickas. Semantic business rules in service oriented development of information systems. 15th *International Conference on Information and Software Technologies, IT* 2009, *Kaunas, Lithuania*, 2009, 404–416.

[31] L. Nemuraite, T. Skersys, A. Sukys, E. Sinkevicius, L. Ablonskis. VETIS Tool for Editing and Transforming SBVR Business Vocabularies and Business Rules into UML & OCL Models. 16th *International Conference on Information and Software Technologies, Kaunas: Kaunas University of Technology*, 377-384, 2010.

[32] J. Cabot, R. Pau, R. Raventós. From UML/OCL to SBVR specifications: A challenging transformation. *Information Systems*. 35(4), 2010, 417-440.

[33] R. Eder, T. Kurz, A. Filieri, A. Margarito. D2.3 - Extended vocabulary and rule set for an existing scenario. *OPAALS Project*, 2008.

[34] A. M. Hickey, A. M. Davis. Requirements Elicitation and Elicitation Technique Selection: A Model for Two Knowledge-Intensive Software Development Processes. *Proceedings of the* 36*th Annual Hawaii International Conference on System Sciences (HICSS*'03*). Washington, DC, USA: IEEE Computer Society*, 2003, 96.

[35] J. Schekkerman. Enterprise Architecture Tool Selection Guide v5.0. *Institute for Enterprise Architecture Developments*. 2009.

[36] A. Caplinskas. Requirements Elicitation in the Context of Enterprise Engineering: A Vision Driven Approach. *Informatica* , 20(3), 2009, 343–368.

[37] A. Van Lamsweerde, L. Willemet. Inferring Declarative Requirements Specifications from Operational Scenarios. *IEEE Trans. Software Eng.*, 24, 1998, 1089-1114.

[38] **S. Pavalkis, L.Nemuraite, P.Tarvydas, A. Noreika.** Specification of finite element model of electronic device using model-driven Wizard-based guidance. *Electronics and Electrical Engineering*, 2(98), 2010, 59-62.

[39] **Department of Defence.** DoD Architecture Framework Version 2.0. *Volume* 2: *Architectural Data and Models Architect's Guide. USA*, 2009.

[40] **N. Maiden.** User Requirements and System Requirements. *Software, IEEE*. 25(2), 2008, 90 – 91.

[41] **R.R. Young.** The requirements engineering handbook. *Nordwood, MA, USA: Artech House*, 2004.

[42] **OMG.** Systems Modeling Language, Version 1.1. *Needham, MA, USA: Object Management Group*, 2008.

[43] **OMG.** Meta Object Family Specification (MOF), *Needham, MA, USA: Object Management Group*, 2002.

[44] **S. Friedenthal, S., A. Moore, A., R. Steiner.** A Practical Guide to SysML. *Burlington, MA, USA: Elsevier*. 2008.

Received February 2011.