

9-2010

BRIDGING THE GAP BETWEEN BUSINESS PROCESS MODELING AND SOFTWARE REQUIREMENTS ANALYSIS: A CASE STUDY

Ahmet Coskuncay

Middle East Technical University, Ankara, & Bilgi Group Ltd. Ankara, Turkey, ahmet@ii.metu.edu.tr

Banu Aysolmaz

Middle East Technical University, Ankara, & Bilgi Group Ltd. Ankara, Turkey, banu@ii.metu.edu.tr

Onur Demirors

Middle East Technical University, Ankara, & Bilgi Group Ltd. Ankara, Turkey, demirors@ii.metu.edu.tr

Omer Bilen

Turkish Republic State Planning Organization, Ankara, Turkey, obilen@dpt.gov.tr

Idris Dogani

Turkish Republic State Planning Organization, Ankara, Turkey, dogani@dpt.gov.tr

Follow this and additional works at: <http://aisel.aisnet.org/mcis2010>

Recommended Citation

Coskuncay, Ahmet; Aysolmaz, Banu; Demirors, Onur; Bilen, Omer; and Dogani, Idris, "BRIDGING THE GAP BETWEEN BUSINESS PROCESS MODELING AND SOFTWARE REQUIREMENTS ANALYSIS: A CASE STUDY" (2010). *MCIS 2010 Proceedings*. 20.

<http://aisel.aisnet.org/mcis2010/20>

This material is brought to you by the Mediterranean Conference on Information Systems (MCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in MCIS 2010 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

BRIDGING THE GAP BETWEEN BUSINESS PROCESS MODELING AND SOFTWARE REQUIREMENTS ANALYSIS: A CASE STUDY

Ahmet Coskuncay, ahmet@ii.metu.edu.tr

Banu Aysolmaz, banu@ii.metu.edu.tr

Onur Demirors, demirors@ii.metu.edu.tr

*Informatics Institute, Middle East Technical University, Ankara, Turkey
and Bilgi Group Ltd. Ankara, Turkey*

Omer Bilen, obilen@dpt.gov.tr

Idris Dogan, dogani@dpt.gov.tr

*General Directorate of Regional Development and Structural Adjustment
Turkish Republic State Planning Organization, Ankara, Turkey*

Abstract

Business process definitions might serve variety of purposes. Two common uses of process models are business process management and software requirements analysis. Although commonalities exist, different models are established for these goals. Establishment of different models not only requires larger amount of effort but also results in inter-model inconsistencies and creates overheads for keeping the models integrated. This paper describes a case study to establish a unified modeling approach for overcoming these difficulties. The case study was performed in a large scale governmental organization. As part of the case study, process models were developed and requirements definitions were generated from process models. The outcomes of the case study were validated by the organization who utilized them in generating quality manuals as well as acquiring software artifacts. The results show that, business process models can be used for software requirements analysis, while the total effort for business process modeling and software requirements analysis is significantly decreased.

Keywords: Business process modeling; requirements engineering; function allocation diagram; natural language software specification

1 INTRODUCTION

Business process modeling (BPMoD) is utilized to analyze, define and improve business processes of organizations. It has become a common tool for business process reengineering during the last few decades. BPMoD is most critical in situations where the environment is complex, multi-dimensional and many people are directly involved in using the system (Recker et al. 2009, Yourdon 2000). This is also the situation in enterprises that need information systems (IS) to automate their business processes. Organizations need to perform requirements analysis to develop software systems that correspond to their needs. SWEBOK (Abran et al. 2004) defines requirements analysis as a step within requirements engineering activities that focuses on determining the bounds of the software and its interactions with the environment, specifying software requirements using system requirements and detecting and resolving conflicts between requirements.

Both BPMoD and requirements analysis are critical for the success of organizations. Avoiding poor software specifications is a crucial motivation for lowering cost of quality, as cost of quality increases exponentially with the errors detected in later stages of development life cycle (Westland 2002). BPMoD, especially for complex organizations, increases efficiency in determining deficiencies in current business processes (Tarhan et al. 2007). For many cases, the need for BPMoD and requirements analysis emerges concurrently or consecutively. For example, newly-founded organizations using BPMoD to define its business processes or existing organizations conducting business process reengineering consider developing software to improve their process efficiency, which results in the need for requirements analysis for those systems. Current BPMoD languages and tools do not directly support an integrated approach for software requirements activities (Specht et al. 2005, Cox et al. 2005), while few requirements engineering approaches as in Berenbach et al. (2009) associate themselves with BPMoD. Our previous studies in military domain show positive results in forming requirements directly from business process models (Demirors et al. 2003).

In a generic waterfall development model, requirements engineering takes about 16% of total development effort (Yang et al. 2008). BPMoD also requires considerable amount of time and effort. In a study where software acquisition is planned for systems supporting business processes, approximately 13 person-months of BPMoD effort is spent on a system of 10.000 MK2 Function Points, and 20 person-months on a system of 25.000 MK2 Function Points (Tarhan et al. 2007). In software development projects, much of the effort spent on BPMoD is duplicated for requirements analysis activities, while additional effort is needed for keeping models and requirements synchronized.

We hypothesize that, unifying BPMoD and requirements analysis activities and generating requirements specifications from business process models through automation can create an opportunity to decrease the total effort of BPMoD and requirements engineering, and also to synchronize BPMoD artifacts, requirements artifacts and the activities to develop them. Such a unified approach would also bring other benefits like providing a better communication environment between customers and developers, ensuring that process owners and software engineers are on the same terms, allowing process knowledge to be used within the requirements phase (Cox et al. 2005), revealing relations between models and requirements, exposing IS integration points within business process models and in these ways, improving completeness and traceability of requirements (Nicolas & Toval 2009).

In this paper, we present the results of a case study we have performed in a governmental organization to identify the opportunities to bridge the gap between requirements analysis and BPMoD, and to evaluate if the above benefits can be achieved. During the case study, business processes were defined and software requirements were depicted in business process models by utilizing a unified modeling approach. Requirements specification documents were generated from these models via an automation tool

developed in the study. The main deliverables of the study are business process models and requirements specifications. These outputs were validated by customers through systematic walkthroughs.

The paper is organized as follows. Part 2 describes the research questions and the case study plan. Implementation of the case study is detailed in Part 3, including the approach for BPMod and requirements analysis. The results and possible threads to the study are discussed in Part 4. Discussions of the results, benefits obtained from the study and possible future work are discussed in Part 5.

2 CASE STUDY DESIGN

2.1 Research Questions

The research questions are;

- How can business process models be utilized for requirements analysis?

We aim to constitute an approach to conduct requirements analysis activities concurrently with BPMod and by using business process models, so that business process model knowledge can be transferred to the software requirements.

- Does using process models for requirements analysis increase the efficiency of BPMod and requirements analysis activities in total?

We aim to track the effort required for unified requirements generation and compare with the estimated traditional requirements analysis effort. Effort estimation would be performed using the functional size.

2.2 Case Study Plan

To evaluate the above questions, activities to conduct the case study started with developing a case study plan. Following tasks are envisaged within the plan:

- Develop business process models for a small set of processes in the case.
- Tailor the BPMod implementation approach to derive the software requirements from these models.
- Write down and review requirements statements about the selected process set manually, independent of the process models.
- Examine the BPMod implementation approach specified before to fulfill the structural, language and data needs of the requirements statements.
- Define extensions and modifications required in BPMod notation to attain individual requirements statements.
- Define the approach for generating the whole requirements specifications document from process models that are developed using the approach.
- Manually generate requirements specifications for the process set and review them.
- Automate the generation of individual requirements statements and the requirements specifications document from process models.
- Extend the application of implementation approach to process modules in the case.

The BPMod and review steps specified above were planned to be conducted together with subject matter experts via workshops. Before starting modeling activities, a training session was reserved to train subject matter experts in BPMod concepts. The roles and responsibilities in the case study were identified and available business process guidelines were gathered and analyzed before starting modeling activities. Efforts, decisions, issues, and improvement opportunities were decided to be recorded for evaluating results of the case study.

The work was planned to be divided into modules that are composed of business processes; and modeling activities were planned to be performed in a module by module basis.

Deliverables of the case study are business process models, software requirements specifications document, data dictionary, workshop records and progress reports. Internal reviews by subject matter experts, external reviews by peer BPMoD experts, external subject matter experts and subcontract authorities were planned for validation of products.

BPMoD and requirements analysis activities were to be repeated for each module, and outputs of each module were delivered to customer and acceptance procedures were applied. The case study was planned to be finalized when all deliverables are developed, delivered and accepted by the customer.

3 CASE STUDY IMPLEMENTATION

The case study was implemented in a governmental organization that launched a project to define its business processes and develop information systems to support those processes. The organization under study is a recently established governmental organization specialized in supporting regional development. One of the main practices of the organization is to develop and conduct regional grant programs. It is important for the success of the implementation approach to be developed that, this organization has complex process sets with properties like conditionals, loops, constraints, inputs/outputs, relational, hierarchical and cross-referencing processes. Resulting business process models are planned to be used by almost 900 personnel. The total project is scheduled for a year.

The unified implementation approach for BPMoD and requirements analysis was developed following the tasks described in the case study plan.

Eight process modules were identified. The modules were determined as process areas that constitute core and supporting sets of processes of the organization. Modeling activities for these modules are conducted by workshops, modeling sessions, trainings and reviews. Six subject matter experts on average from the organization and an external team of three BPMoD experts took part in the workshops. Legislative documents together with subject matter expertise are primary inputs for business process models.

The selected set of process models, manually developed requirements specifications and generated requirements were reviewed by subject matter experts and external BPMoD experts during the development of implementation approach.

Peer reviews were applied for process models by external BPMoD experts. Review sessions were conducted by subject matter experts. Review results were documented, and models were updated and approved before finalizing the products. Software requirements specifications were reviewed by subject matter experts and subcontract authority. All of the findings were documented, updates were planned and applied. Progress reports were prepared and acceptance procedures were applied for each process module.

The implementation approach developed within the case study is described below.

Available BPMoD notations and tools were analyzed considering research questions to specify BPMoD technique to be used in the case study. Extended event-driven process chain (eEPC) notation was chosen as the BPMoD technique. eEPC is one of the notations suitable for BPMoD with subject matter experts who are not familiar with process modeling, describing the business processes with business logic instead of formal process specification logic (van der Aalst 1999).


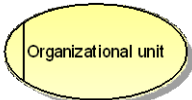
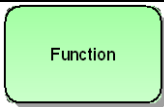


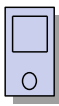
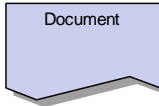


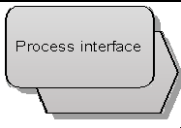
EPC is a BPMoD notation that became popular in 1990s and used to define logical and temporal dependencies between activities that are performed in business processes (Scheer & Schneider 2006, Mendling 2008). Extended EPC notation is based on activity flow combining static resources of business, such as organizations, systems, rules, input and outputs (Davis & Brabänder 2007). Restricted set of object representations we have utilized in eEPC models are provided in Table 1.

Process analysis was conducted in a top-down approach. High level process groups and relations between these groups were identified. Then, each of these process groups were broken down into lower level processes. A sub-process of a process can be reached by a link from the function.

Guidelines from literature and experiences were utilized to determine the granularity and organization of the models (Mendling et al. 2010). There are at most fifteen functions and fifty objects in a process. Average number of function objects in a process is six. On the other hand, the depth of the hierarchy is kept no more than five.

Reusable processes that are required by more than one higher level process were developed. A reusable process can be triggered anywhere as a process interface. Process interfaces and sub-processes are the key mechanisms to form the hierarchical and modular structure of processes in the case study. They enhance the understandability and maintainability of the processes. Hierarchy of processes and process interfaces can be utilized to form a high level process map and reveal interfaces between process modules.

Table 1. Object Types in eEPC Models

Object Name	Object Symbol
Organizational Objects	 
Function	
Event	
Logical Operators (And-Or-Exclusive Or)	
Information Carrier	  
Business Rule	
Process Interface	

An eEPC model developed for a sub-process in the case study is provided in Figure 1. An organization unit, DK, is responsible for carrying out the functions. The process is initiated if assessment of project proposals is complete and reassessment request comes. Project list is the input and DK assessment schedule is the output of making preparations for reassessment function, which is constrained by the rule that each project proposal to be reassessed is to be reassessed by two DK members. During the next activity, report reassessment rationale, reassessment rationale report is produced. DK assessment process interface is the next object which is used to state that DK assessment sub-process will be performed. After DK assessment process is complete, either the process ends if the reassessment are complete or the

process returns to the first activity if there still exists project proposals to be reassessed. In this example, “DK” is a term in our case study and is used here for illustrative purposes.

Besides eEPC, function trees are also used where there is no activity flow relation between the functions of a process. The function trees are composed of functions that are connected according to a functional hierarchy. The use of function trees contributes to the hierarchical structure. An organization chart was maintained in BProMod studies covering all organizational objects in processes and their relations.

After developing eEPC models, next task was discovering the activities within the business process models that will be automated by means of information systems. These will be referred to as “IS integration points”. After this task, next step was embedding rest of the information required for software requirements to process models.

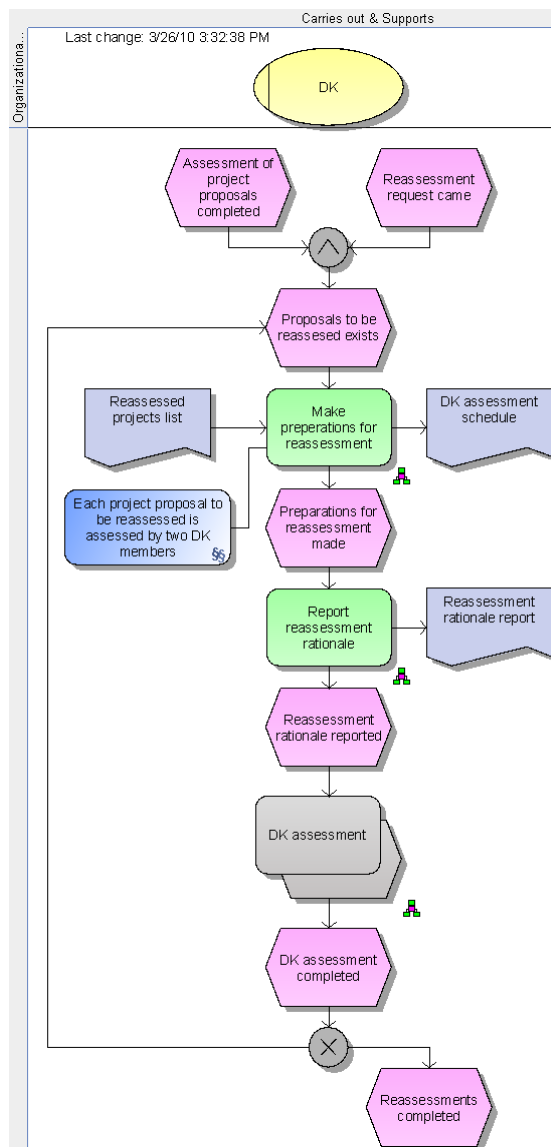


Figure 1. Example extended Event-driven Process Chain model.

Considering the eEPC models, the following points were taken into consideration to determine how the rest of the implementation approach would be specified to be able to define a unified approach for BPro and requirements analysis.

- Processes in eEPC models can be considered as features of the system and as use cases in some instances.
- Functions in business process models can be regarded as candidates for use cases. However, they should be detailed with actions to reach an adequate level of detail in transformation to software specifications.
- Inputs and outputs can give ideas of the system inputs and outputs, but they do not constitute the system inputs and outputs themselves. Not all would be desired to be maintained by the system for process oriented reasons such as legislative constraints. Besides, information systems might require additional inputs and outputs.
- Operations executed on inputs and outputs are not depicted in eEPC models. These operations need to be defined in a standard way to attain a complete and nearly formal approach.
- Business rules define the legislative rules and process constraints in eEPC models. Regarding IS integration points, some of the existing business rules are not related to software characteristics and some supplementary rules need to be defined for software specifications.
- In eEPC models, organizational objects are used to identify responsibilities. However in an information system, other organizational objects may be authorized to perform the related function.
- eEPC notation does not support modeling of application systems.

Considering the points above, we concluded that embedding rest of the information regarding the software requirements on eEPC models would not cover all the needs of the implementation approach and prevent keeping business process models lean without overloading them with IS related details. We also needed to ensure that the approach is easily understood and used by subject matter experts. Model based representations of requirements engineering products are advantageous against textual representations, since they are culture and language neutral and are reviewed faster and thoroughly (Berenbach et al. 2009). Also, in our case study previous eEPC modeling efforts create the advantage that subject matter experts are already used to doing analysis on visual models. Function allocation Diagrams (FADs), which are represented hierarchically under eEPC models in ARIS methodology (Davis & Brabänder 2007), were determined to be the most appropriate candidate for extending our approach to model software requirements information not covered by existing eEPC models. FADs are most often used to keep business process models lean, by enabling business process models not to be overloaded with details such as inputs, outputs, application systems and possibly roles. By this definition, FADs are not designed to keep more information compared to business process models (Specht et al. 2005). In this study, the FADs are used to separate business process models from information specific to software. The most significant deviation from the common use of FADs is that, software support to the business processes are much more strongly defined with the system entities of application systems, business rules, and operations on system entities.

In the next step, this model-based structure of the approach enabled us to form natural language specification documents.

Object representations utilized in FADs are given in Table 2. The connection representation between function and entity objects is given in Table 3.

Table 2. Object Types in Function Allocation Diagrams.



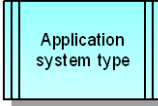
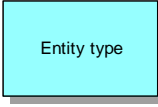
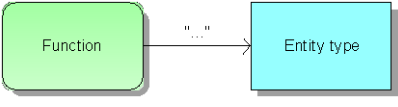
Object Name	Object Symbol
Function	
Position	
Application System Type	
Entity Type	

Table 3. Connection Type Representations between Function and Entity Objects.

Connection Name	Connection Representation
Creates	
Changes	
Reads	
Views	
Lists	

The next task in defining the approach was transforming process models into natural language specifications. Requirements embedded inside process models as objects and connections were gathered and translated into natural language specifications. Natural language specifications are used to define software requirements in non-formal sentence structures. Although there are different practices of natural language sentences; in general, they include verbs that represent actions and nouns that represent actors, target objects and input-output parameters (Saeki et al. 1989).

Based on these considerations, natural language software specification sentences were written manually to explore how they would be like if the process models and their objects were not used. The sentences were reviewed by peer software engineers. Following the review, three points were identified to be added to modeling notation. These additions are explained below.

- Business rule object is added to the FAD notation for specifying the business rules in eEPC models which can be translated to system specifications.
- If there is a need for selective execution of operations for a set of connections, those connections are identified with the same numbered label on them.
- Naming conventions of objects in FAD notation are determined for fitness to natural language sentences in Turkish.

An example FAD is provided in Figure 2. In Figure 2, the function, make preparations for reassessment, is performed by DK member. DK assessment schedule is created and reassessed projects list is listed on PFDS system. The business rule constraints the PFDS system. On ABYS system, archive exit record is created and project proposal or multiphase grant program project proposal is read. In this example, “ABYS”, “PFDS”, and “DK Member” are the terms in our case study and are used here for illustrative purposes.

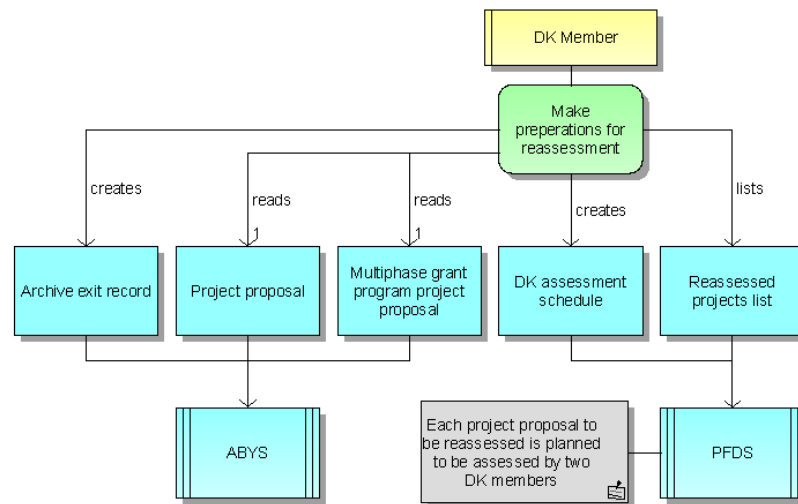


Figure 2. Example Function Allocation Diagram.

In this study, we dealt with functional software requirements analysis, since business process models contain scarce information on non-functional aspects that would be transformed to non-functional requirements. However, by the use of business rule objects for determining non-functional expectations from IS to be developed, gathering information on non-functional software requirements is also possible.

Activity of developing a FAD starts with choosing the function in business processes that is intended to be automated with information systems. Then the roles authorized to execute the function on IS are determined. The entities are defined by considering the system inputs and outputs of the function. Connection types are determined considering the function and the application system. Each entity is connected to an application system that the related entity is to be contained in. Lastly, the business rules on FAD are inspired by the business rules already placed in business process models and business process guidelines.

After specifying the modeling approach, a natural language software specification sentence structure was formed by using the review results of manually written sentences. The sentence structure adapted to English language in the case study is provided below. Underlined words represent the objects in FAD notation.

“In Function, Entity shall be Connection Type on Application System system by Position.”

According to the sentence structure above, an example natural language specification for the FAD in Figure 2 is as follows.

“In making preparations for reassessment, archive exit record shall be created and project proposal or multiphase grant program project proposal shall be read on ABYS system, DK assessment schedule shall be created and reassessed projects list shall be listed on PFDS system by DK member.”

Business rule in Figure 2 is used in a natural language specification as follows.

“Each project proposal to be reassessed shall be planned to be assessed by two DK members on PFDS system.”

Requirements were numbered and grouped under the name of the process they are related to. The process hierarchy in eEPC diagrams was utilized to form heading levels of the requirements specification document and to name the sections. As a result, requirements specification document structure was identified, containing numbered specifications organized into headings.

The requirements specification document and the sentence structure were used in forming natural language specifications manually. These manually written specifications were reviewed by peer software engineers and the organization under study. Positive feedbacks were received from both parties and the sentence and document structures were validated.

Following the validation, a tool was developed to automate the generation of natural language specifications (Coskuncay 2010). The tool generates the specifications in the predetermined sentence and document structure. The tool takes database objects from the ARIS Business Designer tool and creates natural language specifications as an MS Word document. Validation of the outputs of the tool was done by comparing them with manually written natural language specifications.

4 RESULTS AND THREATS

BPMod activities for the process modules are being conducted according to the approach defined in the case study. Two modules have been completed and approved so far in the study. In these modules, 946 process models, 791 of which were FADs, were delivered.

Natural language software specifications were generated from process models automatically via the tool developed within the case study. The requirements specification documents delivered to sponsor organization contained 1002 natural language specifications.

The deliverables were reviewed and accepted by the sponsor organization. We conducted three walkthroughs for process models, one review for software specifications and one review for our approach in the case study. The outputs were revised based on the review results. Both business process models and software specification documents were also delivered to a contractor software development organization.

The outputs of requirements engineering activities are functional and non-functional requirements. In this study, we determined functional requirements in entity and use case levels. Functional requirements in attribute and user scenario levels need to be specified to complete the functional requirements. We made an experience based assumption that this part takes at most 50% of requirements engineering activities and specifying non-functional requirements takes 10%. This assumption concludes that we completed at least 40% of the requirements engineering tasks in this study for the information systems to be developed.

3000 person-hours of effort were spent on the whole for the first two modules. The size of the information system to be developed for the first two modules is 11000 Cosmic Function Points (Kaya 2010).

The mode value of productivity range for requirements phase in software development life cycle is 0.75 person-hours per function point (Jones 1998). So, estimated effort for requirements specification is 8250 person-hours and 40% of this estimated effort corresponds to 3300 person-hours. This estimated effort of 3300 person-hours is almost equal to our realized effort of 3000 person-hours. Considering these values, we conclude that with spending the sole effort of performing requirements analysis, we managed to perform both requirements analysis and BPMod activities. On the whole, the total effort of BPMod and requirements analysis activities is managed to be decreased in our study.

Four threats to the validity of our case study are identified.

First threat is that, natural language specifications are not utilized in software development yet. However, they are based on most commonly used requirements engineering techniques, approved by sponsor organization and delivered to a contractor software developer. For these reasons, we believe this is not a threat that would have a detrimental effect on validity.

The second threat requires us to question ourselves about being biased in perceiving the results, in case study. We tried to eliminate this threat by adapting the approach from some of the most commonly used requirements engineering techniques and reviews by peer software engineers in pilot study.

The third threat is that the developed approach has been validated through one case study yet. Resolution of this threat is left to further studies where the approach will be practiced in future case studies.

Final threat to validity is that, natural language specifications are delivered in Turkish to the sponsor organization, whereas they are presented in English in this paper. Validity of case study in this paper is not investigated for natural language specifications in English. As model elements can also be directly used to structure an English sentence, this is also believed not to constitute a major threat to validity of our case study.

5 CONCLUSIONS

In this paper; our purpose was to explore utilization of business process models for requirements analysis. For this purpose, a case study was conducted, which was applied in an organization whose business processes were modelled and software requirements specifications were delivered. 946 process models including 791 FAD diagrams, requirements specification document with 1002 natural language specifications were formed spending 3000 man hours in total and for a system of 11000 function points. Automated derivation of software requirements statements and formation of a complete software requirements specification document were achieved by means of the tool developed.

We have answered the first research question by developing a unified modelling approach that bridges the gap between BPMoD and requirements analysis. This implementation approach was validated by developing the outputs of the case study by means of multiple review mechanisms. The approach enabled us to transfer the business process knowledge to software requirements by generating the software requirements document automatically. Our implementation approach replaces, in part, the analysis phase of software development life cycle.

We have answered the second research question by calculating the efforts spent in the studies and comparing them with the industry data considering the size of our system. We see that, even if we assume we have covered 40% of requirements engineering activities as a lowest estimate, our realized total effort of BPMoD and requirements analysis correspond to the same percent of requirements engineering estimated of industry data.

Besides increasing the efficiency of requirements analysis in terms of total effort, we observed many other benefits by utilizing this approach. Business process models formed an intuitive environment for us to discuss, gather and analyse requirements in a structured way. In this way, the possibility of skipping and duplicating any part of the information systems requirements decreased. These aspects increased the completeness and correctness properties of requirements. The activity of BPMoD forces developers to define the system in a structured way. In the above section, we discussed that hierarchical structure of process models enables definition of modular processes and reveals relations between those processes in different levels. By all these means, the processes can be explained in a more unambiguous and consistent way compared to a natural language explanation. As a result, we formed a requirements document that is unambiguous and consistent, as much as the process models are. We are also sure that the set of requirements were specified one time and only one time, as it was not possible to create duplicate objects in the process models. Maintainability of requirements was also increased; as traceability between business processes and requirements are clearer. By means of the automated tool, updated requirements specification document was also easily rebuilt when the requirements changed.

One direction of the future studies is envisaged as applying the proposed approach in multiple case studies and use the results to define the approach as a formal methodology. The case studies shall be applied in different industries and for organizations in different sizes. In this way, it can be ensured that the methodology covers divergent requirements in the BPMo field.

By using FADs, information of all data transformations in the system is maintained and in this way, skeletons of data types are derived. Other research direction may include investigating utilization of this information in other requirements engineering and software design techniques. We have already conducted some studies on automated size estimation for the development of information systems (Kaya 2010). It is also possible to investigate the opportunities for derivation of use cases and formal specifications. For such a study, FAD notation would be extended with action sequences, system states and logical separation and connections.

Acknowledgment

We would like to thank to Nahit Bingol and other members of the Turkish Republic State Planning Organization's General Directorate of Regional Development and Structural Adjustment, Cukurova Development Agency and Izmir Development Agency for their efforts and support in this study.

References

- Abran, A., Bourque, P., Dupuis, R., Moore, J. W. and Tripp, L. L. (2004). Guide to the Software Engineering Body of Knowledge – SWEBOK. 2004th ed., A. Abran, P. Bourque, R. Dupuis, J. W. Moore, and L. L. Tripp, Eds. Piscataway, NJ, USA: IEEE Press.
- Berenbach, B., Paulish, D., Kazmeier, J. and Rudorfer, A. (2009). Software & Systems Requirements Engineering: in Practice. 1st ed. McGraw-Hill, Inc.
- Coskuncay, A. (2010). An Approach for Generating Natural Language Software Specifications by Utilizing Business Process Models. M.S. Thesis, METU Informatics Institute, Ankara.
- Cox, K., Phalp, K. T., Bleistein, S. J. and Verner, J.M. (2005). Deriving Requirements from Process Models via the Problem Frames Approach. *Information and Software Technology*, 47, pp. 319-337.
- Davis, R. and Brabänder, E. (2007). ARIS Design Platform: Getting Started with BPM, 1st ed. Springer.
- Demirors, O., Gencel, C. and Tarhan, A. (2003). Utilizing Business Process Models for Requirements Elicitation. *Proceedings of the 29th Conference on EUROMICRO*, p. 409.
- Jones, T. C. (1998). *Estimating Software Costs*. McGraw-Hill, Inc.
- Kaya, M. (2010). E-Cosmic: A Business Process Model Based Functional Size Estimation Approach. M.S. Thesis, METU Informatics Institute, Ankara.
- Mendling, J. (2008). *Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness*. Springer Publishing Company, Incorporated.
- Mendling, J., Reijers, H. A. and van der Aalst, W. M. P. (2010). Seven Process Modeling Guidelines (7pmg). *Information and Software Technology*, 52(2), pp. 127–136.
- Nicolas, J. and Toval, A. (2009). On the Generation of Requirements Specifications from Software Engineering Models: A Systematic Literature Review. *Information and Software Technology*, 51, pp. 1291-1307.
- Recker, J., Rosemann, M., Indulska, M. and Green, P. (2009). Business Process Modeling - A Comparative Analysis. *Journal of the Association for Information Systems*, 10(4), pp. 333–363.
- Saeki, M., Horai, H. and Enomoto, H. (1989). Software Development Process from Natural Language Specification. 11th International Conference on Software Engineering.

- Scheer, A.W. and Schneider, K. (2006). Aris - Architecture of Integrated Information Systems. In P. Bernus, K. Mertins and G. Schmidt (Ed.), Handbook on Architectures of Information Systems, pp. 605-623.
- Specht, T., Drawehn, J., Thranert, M. and Kuhne, S. (2005). Modeling Cooperative Business Processes and Transformation to a Service Oriented Architecture. In 7th International IEEE Conference on E-Commerce Technology, pp. 249-256.
- Tarhan, A., Gencel, C. and Demirors, O. (2007). Pre-Contract Challenges: Two Large System Acquisition Experiences. Book chapter in Enterprise Architecture and Integration: Methods, Implementation and Technologies, IGI Global.
- van der Aalst, W. (1999). Formalization and Verification of Event-driven Process Chains. Information and Software Technology, 41(10), pp. 639-650.
- Westland, J.C. (2002). The Cost of Errors in Software Development: Evidence from Industry. The Journal of Systems and Software, 62, pp. 1-9.
- Yang, Y., He, M., Li, M., Wang, Q. and Boehm, B. (2008). Phase Distribution of Software Development Effort. In Proceedings of the Second ACM-IEEE international Symposium on Empirical Software Engineering and Measurement, pp. 61-69.
- Yourdon, E. (2000). Managing Software Requirements. Addison Wesley Publishing Company.