

AN APPROACH FOR GENERATING
NATURAL LANGUAGE SPECIFICATIONS
BY UTILIZING BUSINESS PROCESS MODELS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

AHMET COŞKUNÇAY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

AUGUST 2010

Approval of the Graduate School of Informatics

Prof. Dr. Nazife BAYKAL
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Tuğba TAŞKAYA TEMİZEL
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Onur DEMİRÖRS
Supervisor

Examining Committee Members

Prof. Dr. Semih BİLGEN (METU, EEE)_____

Assoc. Prof. Dr. Onur DEMİRÖRS (METU, II)_____

Assist. Prof. Dr. Aysu BETİN CAN (METU, II)_____

Assist. Prof. Dr. Altan KOÇYİĞİT (METU, II)_____

Dr. Ayça TARHAN (HACETTEPE Ü., BİL)_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name and Surname : Ahmet COŞKUNÇAY

Signature :

ABSTRACT

AN APPROACH FOR GENERATING NATURAL LANGUAGE SPECIFICATIONS BY UTILIZING BUSINESS PROCESS MODELS

COŞKUNÇAY, Ahmet

M.Sc., Department of Information Systems

Supervisor: Assoc. Prof. Dr. Onur DEMİRÖRS

August 2010, 82 pages

Business process modeling is utilized by organizations for defining and reengineering their business processes. On the other hand, software requirements analysis activities are performed for determining the system boundaries, specifying software requirements using system requirements and resolving conflicts between requirements. From this point of view, these two activities are considered in different disciplines. An organization requiring its business processes to be defined and supported with information systems would benefit from performing business process modeling and requirements analysis concurrently.

In this study, an approach enabling concurrent execution of business process modeling and requirements analysis is developed. The approach includes two business process modeling notations adapted to the research needs, a process defining the steps for implementing the approach and the requirements generation tool that generates natural language specification documents by using business process models. Within this study, two case studies are introduced; one describing

the development of the approach and the other exploring if the total efficiency of performing business process modeling and requirements analysis activities would be increased by using the approach.

Keywords: Business Process Model, Natural Language Specification, Function Allocation Diagram, EPC, Automated Requirements Generation.

ÖZ

İŞ SÜRECİ MODELLERİNİ KULLANARAK DOĞAL DİLDE BELİRTİM ÜRETME İÇİN BİR YAKLAŞIM

COŞKUNÇAY, Ahmet

Yüksek Lisans, Bilişim Sistemleri Bölümü

Tez Yöneticisi: Doç. Dr. Onur DEMİRÖRS

Ağustos 2010, 82 sayfa

İş süreci modelleme organizasyonlar tarafından iş süreçlerinin tanımlanması ve yeniden yapılandırılması için kullanılmaktadır. Diğer taraftan, yazılım gereksinim analizi aktiviteleri sistem sınırlarının belirlenmesi, sistem gereksinimlerini kullanarak yazılım gereksinimlerinin belirlenmesi ve gereksinimler arasındaki ihtilafların çözülmesi için gerçekleştirilir. Bu bakış açısıyla, bu iki aktivitenin farklı disiplinlerde yer aldığı sayılmaktadır. İş süreçlerinin tanımlanmasına ve bilgi sistemleri ile desteklenmesine ihtiyaç duyan bir organizasyon iş süreci modelleme ve gereksinim analizinin eşzamanlı gerçekleştirilmesinden fayda sağlayabilir.

Bu çalışmada, iş süreci modelleme ve gereksinim analizinin eşzamanlı yürütülmesine olanak sağlayan bir yaklaşım geliştirilmiştir. Yaklaşım araştırma ihtiyaçlarına uyarlanan iki iş süreci modelleme gösterimi, yaklaşımın uygulaması için basamakları tanımlayan süreci ve iş süreci modellerinden doğal dilde belirtim dokümanları üreten gereksinim üretme aracını içermektedir. Bu çalışma içinde, biri yaklaşımın geliştirilmesini betimleyen ve diğeri yaklaşımı kullanarak iş süreci modelleme ve

gereksinim analizi aktivitelerinin uygulanmasındaki toplam verimliliğin arttırılıp arttırılamayacağını inceleyen iki vaka çalışması uygulanmıştır.

Anahtar Kelimeler: İş Süreci Modeli, Doğal Dilde Belirtim, Fonksiyon Dağıtım Diyagramı, EPC, Otomatik Gereksinim Üretme.

To my family...

ACKNOWLEDGMENTS

I would like to offer my sincere thanks and appreciation to my advisor Assoc. Prof. Dr. Onur DEMİRÖRS for his guidance, support and patience during my thesis study.

I am grateful to my mother Hafize COŞKUNÇAY, my father Hasan COŞKUNÇAY, my grandmother Mürvet COŞKUNÇAY, my aunts Aysel COŞKUNÇAY and Ayfer KORKUTLU and my sister Merve COŞKUNÇAY for their emotional support and encouragement during my school life, especially during my thesis study.

I am forever grateful to Duygu FINDIK, since she always made me calm and peaceful when I was stressful. Thanks for her patience while I was preparing this study.

I would like to offer my sincere thanks to Banu AYSOLMAZ and Barış ÖZKAN for their invaluable contribution and support throughout my thesis study.

I am also grateful to my friends Ali YILDIZ, Burak OĞUZ, Mahir KAYA and Bilgin AVENOĞLU for their positive touch on my study.

I would like to thank the members of Turkish State Planning Organization for their contribution throughout the research.

TABLE OF CONTENTS

ABSTRACT	iv
ÖZ	vi
ACKNOWLEDGMENTS	ix
TABLE OF CONTENTS	x
LIST OF TABLES	xiii
LIST OF FIGURES	xiv
LIST OF ABBREVIATIONS AND ACRONYMS.....	xvi
CHAPTER	
1 INTRODUCTION	1
1.1 Problem Statement and Motivation	1
1.2 Research Approach.....	3
1.3 Organization of the Study.....	4
2 BACKGROUND and RELATED RESEARCH.....	5
2.1 Software Requirements	5
2.1.1 Natural Language Specifications	6
2.2 Business Process Modeling	8
2.2.1 Extended Event-driven Process Chain	9
2.2.2 Function Allocation Diagram.....	11
2.3 Deriving Software Requirements from Business Process Models	12
2.3.1 Approach of Cox et al. (2005).....	13

2.3.2	Approach of Stolfa & Vondrak (2004)	15
2.3.3	Approach of Specht et al. (2005)	17
2.3.4	Approach of Su (2004).....	18
3	PROCEDO: REQUIREMENTS GENERATION APPROACH.....	21
3.1	Utilization of Business Process Models	21
3.2	A Unified Process: Bridging the Gap between Business Process Modeling and Requirements Analysis.....	27
3.3	Automated Generation of Natural Language Specifications.....	33
3.3.1	Natural Language Specification Sentence Structure.....	33
3.3.2	Software Specification Document Structure	37
3.3.3	Requirements Generation Tool	38
4	CASE STUDIES	44
4.1	Case Study Design.....	45
4.1.1	Case Selection Criteria	45
4.1.2	Background	46
4.2	Case Study 1	47
4.2.1	Case Study Plan	47
4.2.2	Case Study Implementation	48
4.2.3	Results	52
4.2.4	Threats to Validity.....	53
4.3	Case Study 2.....	53
4.3.1	Case Study Plan	54
4.3.2	Case Study Implementation	54
4.3.3	Results	56

4.3.4	Threats to Validity.....	58
5	CONCLUSIONS.....	60
5.1	Summary	60
5.2	Contributions	63
5.3	Future Study	64
	REFERENCES.....	66
	APPENDICES	72
	A: A sample business process model developed in the case study	72
	B: Manually written requirements for a selected business process.....	74
	C: Function Allocation Diagrams for a selected process	76
	D: Natural language specifications generated by tool support for a selected business process.....	79
	E: User manual for the requirements generation tool	81

LIST OF TABLES

Table 1: Comparison of the mainstream approaches in the literature.....	20
Table 2: eEPC model element representations in Proceso	22
Table 3: Object representations in FAD notation	25
Table 4: Connection representations between function and entity type objects in FADs	27
Table 5: Transition from connection types to sentence structure	35

LIST OF FIGURES

Figure 1: An example function allocation diagram (Davis & Brabander 2007).....	12
Figure 2: An example RAD (Cox et al. 2005)	14
Figure 3: An example Jackson context diagram derived from RAD (Cox et al. 2005)	15
Figure 4: Sequential pattern (Stolfa & Vondrak 2004).....	15
Figure 5: Optional pattern (Stolfa & Vondrak 2004).....	16
Figure 6: Branching pattern (Stolfa & Vondrak 2004)	16
Figure 7: An example eFAD (Specht et al. 2005).....	17
Figure 8: An example eEPC model used for requirements generation by KAOS tool (Su 2004).....	19
Figure 9: Process for bridging the gap between business process modeling and requirements analysis	28
Figure 10: Flowchart for requirements generation tool	38
Figure 11: eEPC model of “Nihai Ödeme” process.....	72
Figure 12: Function Allocation Diagram for “proje toplam uygun maliyetlerinin belirlenmesi” function in “Nihai Ödeme” process.....	76
Figure 13: Function Allocation Diagram for “yararlanıcıya yapılacak toplam ödeme miktarının eş finansman gerçekleşme oranına göre indirilmesi” function in “Nihai Ödeme” process	76
Figure 14: Function Allocation Diagram for “nihai ödemenin tespiti” function in “Nihai Ödeme” process.....	77
Figure 15: Function Allocation Diagram for “nihai ödeme miktarının onaylanması” function in “Nihai Ödeme” process	77

Figure 16: Function Allocation Diagram for “kalan eksi değerin Ajans hesabına iade edilmesi için yazı gönderilmesi” function in “Nihai Ödeme” process	78
Figure 17: Function Allocation Diagram for “nihai ödeme miktarının sözleşmede belirtilen hesap numarasına transfer talimatı ile iletilmesi” function in “Nihai Ödeme” process	78
Figure 18: Function Allocation Diagram for “yararlanıcıdan geri ödeme ve ilgili bilgilerin tahsil edilmesi” function in “Nihai Ödeme” process.....	78
Figure 19: User manual for the requirements generation tool – Steps 1 and 2.....	81
Figure 20: User manual for the requirements generation tool – Steps 3, 4 and 5.....	81
Figure 21: User manual for the requirements generation tool – Steps 6, 7, 8 and 9..	82

LIST OF ABBREVIATIONS AND ACRONYMS

ARIS	: Architecture of Integrated Information Systems
BPEL	: Business Process Execution Language
BPMN	: Business Process Modeling Notation
CRUDL	: Create, Read, Update, Delete and List
eEPC	: Extended Event-driven Process Chain
eFAD	: Extended Function Allocation Diagram
EPC	: Event-driven Process Chain
FAD	: Function Allocation Diagram
MKII	: Mark 2
RAD	: Role Activity Diagram

CHAPTER 1

INTRODUCTION

1.1 Problem Statement and Motivation

Business process modeling 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. Business process modeling 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 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 business process modeling and requirements analysis are critical activities for the success of organizations. Requirements analysis is positioned in early stages of information systems development projects. Avoiding poor software specifications is a crucial motivation for lowering costs, as total costs would increase exponentially in relation to the number of errors detected in later stages of development life cycle (Westland 2002). Business process modeling, especially for complex organizations,

increases efficiency in determining deficiencies in current business processes (Tarhan et al. 2007).

For many cases, the need for business process modeling and requirements analysis emerges concurrently or consecutively. This need has become more significant as the focus of information technology has shifted from data-driven approaches to process-driven approaches (van der Aalst et al. 2003). Especially, newly-founded organizations using business process modeling to define their 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. In conventional approaches business process modeling is not considered as core for requirements analysis, but as supporting the phase where it is important for making it certain that people from different backgrounds in both customer and supplier sides reach an agreement on business processes (Dehnert & Rittgen 2001). Business process modeling notations and tools lack supporting an integrated approach for software requirements activities that would complement these considerations.

In a generic waterfall development model, requirements engineering takes about 16% of total development effort (Yang et al. 2008). Enterprises spend high amounts of effort in describing their procedures and interactions in terms of business process models with the aim of describing and standardizing their processes (Roser & Bauer 2005). In a study where software acquisition is planned for systems supporting business processes, approximately 13 person-months of business process modeling effort is spent on system of 10.000 MKII Function Points, and 20 person-months for 25.000 MKII Function Points (Tarhan et al. 2007). In software development projects, much of the effort spent on business process modeling is duplicated for requirements analysis activities, while additional effort is needed for keeping models and requirements synchronized.

In this study, it is hypothesized that unifying business process modeling and requirements analysis activities and automatically generating requirements specifications from business process models can create an opportunity to decrease the total effort of business process modeling and requirements analysis, and also to

create a one-way synchronization from business processes to requirements artifacts and synchronize 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 process models and requirements, exposing IS integration points within business process models and in these ways, improving completeness and traceability of requirements (Nicolas & Toval 2009).

1.2 Research Approach

With the aim of developing a unified approach to perform business process modeling and software requirements analysis activities concurrently, we conducted a case study. The case study is conducted in a governmental organization. It provided a means to develop the approach. Within the case study, business process modeling notation was characterized based on extended Event-driven Process Chain and Function Allocation Diagram notations in the way that business process modeling and software requirements analysis activities could be conducted concurrently. A process that utilizes the developed modeling notation was structured to generate software requirements based on business processes. Finally, a requirements generation tool was developed to generate natural language software specifications from the business process models. Implementation of the case study is documented as an approach that contains modeling notation, process and tool support for unifying business process modeling and requirements analysis activities. The documented approach would guide the early phases of a process-driven information systems development project. After the approach was developed via the case study, another case study was performed where the approach was utilized to explore whether the total effort required for requirements analysis and business process modeling activities could be decreased or not.

1.3 Organization of the Study

Chapter 2 presents the background for business process modeling and software requirements and related research for deriving software requirements from business process models.

Chapter 3 describes utilization of business process models, the unified process definition and automated generation of natural language specifications in Proceso approach.

Chapter 4 introduces the two case studies conducted in a governmental organization. Case study 1 is performed for developing Proceso and case study 2 is performed for exploring the benefits of utilization of Proceso.

In Chapter 5, the finding and contributions of the study are discussed and directions for future studies are suggested.

CHAPTER 2

BACKGROUND and RELATED RESEARCH

This chapter is composed of three sections. In the first section, background information and definitions are given for software requirements and natural language specifications. Second section includes background information on business process modeling and two business process modeling languages that are used in this study are described. Third section contains a literature review of transformation approaches in software engineering and focuses on four mainstream studies in the literature that demonstrate approaches for deriving software requirements from business process models.

2.1 Software Requirements

In this study, the focus is on requirements analysis activities. IEEE std. 610.12-1990 (1990) defines requirements analysis as a process for defining system, hardware or software requirements by studying the user needs.

Berenbach et al. (2009) differentiates requirements analysis inside requirements engineering by stating that;

“Whereas requirements analysis deals with the elicitation and examination of requirements, requirements engineering deals with all phases of a project or product life cycle from innovation to obsolescence.”

Some of the most established requirements specification styles are natural language specifications, use case models, use case specifications, formal software

specifications and data flow diagrams. Natural language specifications are the major concern in this study among these and will be described in detail in this chapter.

Use case models are one of the most widely used techniques in requirements engineering (Jacobson 2004). In use case models, there are use cases that represent the functionalities supplied to external actors by the system and the external actors that represent the users and external systems that use the system (Jacobson & Ng 2004).

Use case specifications describe the scenarios that consist of a path of actions. The path of actions includes external actors, sequence of actions, constraints for actions and definition of actions that include inputs to the system (Achour et al. 1999).

Formal software specifications are formal language expressions of properties the system should satisfy (van Lamsweerde 2000). The Z notation, which is a formal specification notation, contains static aspects that include states and invariant relationships and dynamic aspects that include operations, relationships between operations' inputs and outputs and changes of states (Spivey 1990).

Data flow diagrams, on the other hand, defines logical data flow in a pictorial representation and includes external entities, processes, data stores and the data flow between them (Schach 1995).

2.1.1 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).

Natural language specifications are probably the most practiced type of requirements specification styles in industry. Kamsties (2005) agrees with this by stating that the natural language is the most frequently used representation in stating requirements and diagrams, semi-formal and formal representations are used for supporting the natural language specifications.

Natural language specifications have both advantages and disadvantages when compared with other styles. They are the most appropriate means of communication between the customers and suppliers, however they might also be ambiguous and software engineers might find them inadequate for describing the system (Athanasakis 2006).

och Dag & Gervasi (2005) provides explanations for why natural language specifications are utilized for requirements specifications;

- All stakeholders in development process share natural language as the primary communication language.
- By means of natural language, arbitrary domains and arbitrary levels of abstraction can be stated.
- There is not much motivation for formalizing the requirements, since not all are expected to be implemented.
- Management and analysis of erroneous, incomplete or partially specified requirements, which take a large part in the requirements phase, are adapted naturally by natural language specifications.
- Although formal language is advantageous in verifying the requirements by checking the internal consistency and completeness of requirements, they lack in capturing the external properties of requirements such as relating the requirements with actual user intentions.

As mentioned before, natural language specifications have some disadvantages. Wiegers (2005) describes some of the shortcomings of natural language specifications as;

- Natural language specifications bring ambiguity that creates risks to the quality of the requirements.
- Natural language would result in bulky and verbose specifications.

- A low level of abstraction would be led by detailed natural language statements.

2.2 Business Process Modeling

A process is a set of actions that are performed within a time interval with the aim of achieving or progressing to some objective (Havey 2005).

Workflow Management Coalition (1999) defines business process as a set of procedures and activities that are connected and realize a business objective or policy goal, where functional roles and relationships defined by an organizational structure describes the context.

Dehnert & Rittgen (2001) states that business processes are at the core of reorganization of a company and design or redesign of the corresponding application systems.

Business process modeling has a central role in business process management domain. According to van der Aalst et al. (2003), business process management aims to design, enact, control and analyze operational processes, which involve people, organizations, applications, documents and other information, by supporting business processes with methods, techniques and software.

Minoli (2008) defines the purpose of business process modeling as to seek standardization in business process management where the related business processes might include several applications, data repositories, corporate departments or even companies.

Stolfa & Vondrak (2004) states the main purpose of business process modeling as managing and stimulating processes.

Conceptual business process modeling languages might contain different perspectives. According to List & Korherr (2006), there are five perspectives that are contained in the conceptual business process modeling languages;

- *“Functional perspective represents the activities that are performed.”*

- *“Organizational perspective represents the agents that perform the activities.”*
- *“Behavioral perspective represents sequencing, loops, iterations, decision making conditions, entry and exit criteria within business processes.”*
- *“Informational perspective represents the informational elements that are input to or output from business processes.”*
- *“Business process context perspective represents an overview of the process containing goals and their measures, deliverables, process owners, process types and customers.”*

Some of the business process modeling languages that are most referred in research are Event-driven Process Chain (EPC), Business Process Modeling Notation (BPMN), Role Activity Diagram (RAD) and Petri Nets. All of these notations represent functional and behavioral perspectives, all except Petri Nets represent organizational perspective, BPMN and EPC represent informational perspective while none of them represents business process context perspective (List & Korherr 2006).

In this study, the focus is on extended EPC (eEPC) models and the Function Allocation Diagrams that are represented hierarchically under eEPC models in ARIS methodology (Davis & Brabänder 2007). Descriptions of these two business process modeling notations in the literature are provided in the rest of this section.

2.2.1 Extended Event-driven Process Chain

Event-driven process chain (EPC) is a business process modeling 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 (eEPC) notation is based on activity flow combining static resources of business, such as organizations, systems, rules, input and outputs (Davis & Brabander 2007). eEPC is regarded as a business process

modeling notation that does not require much modeling expertise by describing the business processes with business logic instead of formal process specification logic (van der Aalst 1999). The model elements in lean EPC models are functions, events and logical operators.

Functions are the activities that add value to the process and the events are the states that result from the changes in the world the process is operated in (Davis & Brabander 2007). According to Davis & Brabander (2007), each function should be initiated and resulted by at least one event. The events and functions are the main building blocks of the modeling notation that are used for designating the activity flow in lean EPC notation.

“Events and functions have exactly one incoming and one outgoing arc except start and end events” (Dehnert & Rittgen 2001). Therefore, logical operators are used to define logical separations and connections in the business process flows. By using logical connectors connecting functions and events, flow of control is defined (van der Aalst 1999). There are three types of logical connectors in EPC modeling notation. Davis & Brabander (2007) defines these logical connectors as;

AND (\wedge) rule;

- *“Following a function, process flow splits into two or more parallel paths.”*
- *“Preceding a function, all events must occur in order to trigger the following function.”*

OR (\vee) rule;

- *“Following a function, one or many possible paths will be followed as a result of the decision.”*
- *“Preceding a function, any one event, or combination of events, will trigger the function.”*

XOR (\times) rule;

- *“Following a function, one, but only one, of the possible paths will be followed.”*
- *“Preceding a function, one, but only one, of the possible events will be the trigger.”*

Functions, events and logical connectors are common in all representations of eEPC models. In eEPC models, the set of model elements used might differ based on modeling purpose and business domain.

2.2.2 Function Allocation Diagram

Specht et al. (2005) states that;

“In order to avoid overloading EPCs with details about involved roles and application software systems, details of a function and its context can be shifted to Function Allocation Diagrams (FADs). However, FADs do not introduce any additional modeling artifacts in comparison to EPCs.”

Davis & Brabander (2007) agrees with this statement and adds that by drilling down into EPCs, additional information and relationships about a function would be visible in FADs. The advantage of using FADs for this purpose is that EPCs would keep their focus on the process flow without being overloaded with the information related to functions.

There is not any study in the literature that establishes a standard notation for FADs. The FAD notation is structured with personal preferences most of the time. Davis & Brabander (2007) comments in this issue by providing some guidelines for FAD notation and states that the FADs have the same objects that are available for EPCs, except the logical connectors and events, since there is no process flow representation in FADs. Davis & Brabander (2007) presents an example FAD that is provided in Figure 1.

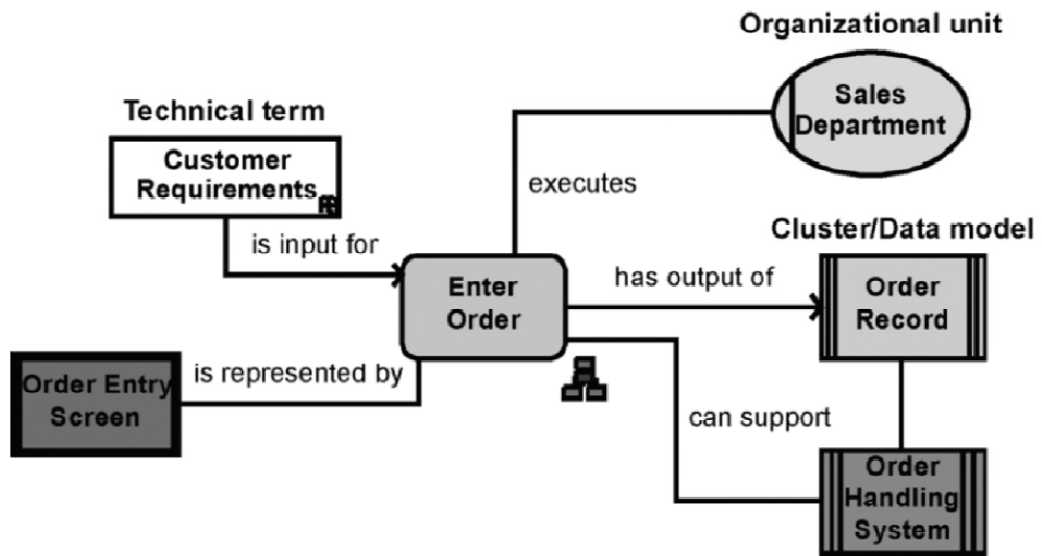


Figure 1: An example function allocation diagram (Davis & Brabander 2007)

2.3 Deriving Software Requirements from Business Process Models

Transformation between software engineering artifacts is a popular research area in the last decade. There are some studies in the literature that introduce transformation to or from requirements engineering artifacts.

Cabral & Sampaio (2008) presents tool support for generating formal specifications in CSP process algebra from user and component view use case written in Controlled Natural Language (CNL). Santander & Castro (2002) describes heuristics to derive use case models from strategic rationale models. Meziane et al. (2008) establishes backward verification from design to requirements by generating natural language specifications from three fundamental types of relationships; associations, aggregations and generalizations in UML class diagrams. Estrada et al. (2003) presents transition from Goal Refinement Trees to Strategic Dependency models and from Strategic Dependency models to Strategic Rationale models. In Strategic Rationale models analysts select the tasks to be automated that will be included in requirements specifications. In Lee & Bryant (2002), an application is developed to enable transition from natural language specification Two-Level Grammar (TLG) to formal specification in Vienna Development Method meta-language (VDM++). Maiden et al. (1998) presents the CREWS-SAVRE tool that utilizes an algorithm to

generate user scenarios by utilizing action-link rules between actions in use cases. Danlos et al. (2000) introduces the tool prototype named Flaubert that takes event graphs as input and produces natural language specifications in French as output. In Jungmayr & Stumpe (1998), extended usage models are used in generating user documentation and tests cases with tool support.

In terms of software development life cycle, some of these studies aim to go one or more steps forward, some generate materials to ensure backward traceability and verification and some aim to derive supporting documentation.

The focus of our study is on deriving software requirements from business process models. Four mainstream studies in this research area are summarized below.

2.3.1 Approach of Cox et al. (2005)

In Cox et al. (2005), an approach is introduced to derive software requirements from business process models. Role Activity Diagram (RAD) notation is used to define business processes and a set of steps for mapping from RADs to Jackson's problem frames is introduced. These steps as quoted from Cox et al. (2005) are;

- *“Explore the problem context.”*
- *“Produce (or revisit) process model (as role activity diagrams).”*
- *“Identify outcomes of interactions.”*
- *“Identify domains from outcomes.”*
- *“Identify potential rules that govern interactions.”*
- *“Identify problem frames.”*

Examples of RAD and Jackson context diagram in the study are provided in Figures 2 and 3 respectively.

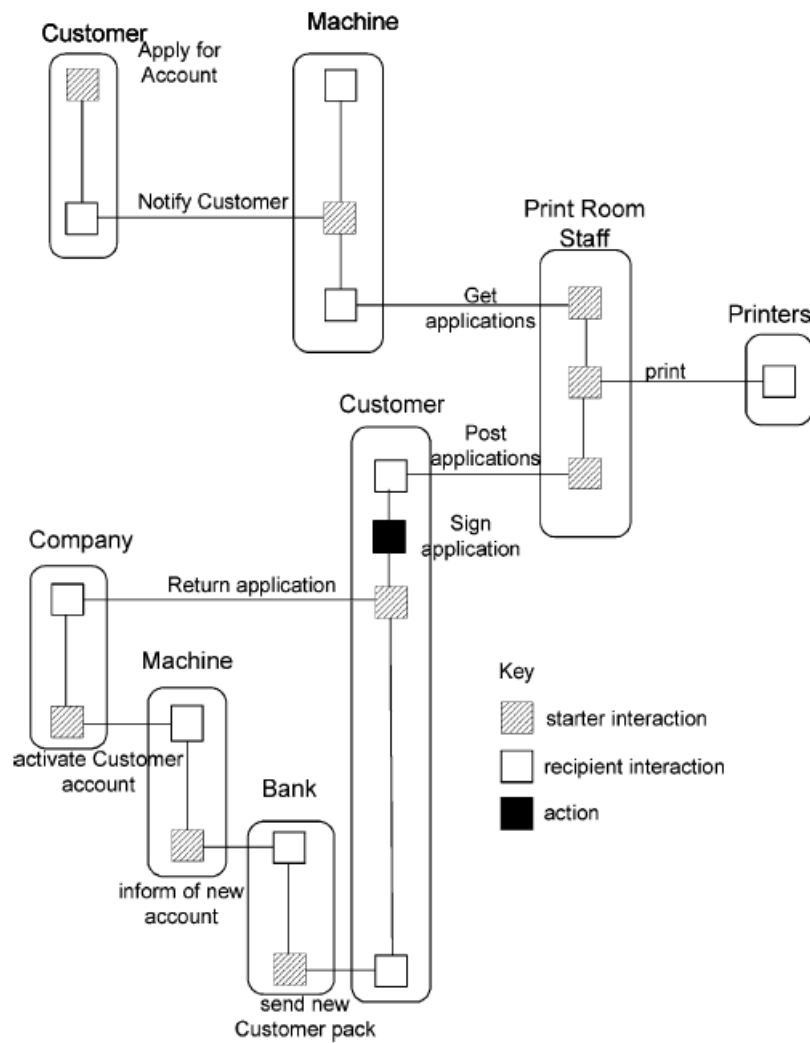


Figure 2: An example RAD (Cox et al. 2005)

Approach of Cox et al. (2005) is an illustration of a systematic methodology for deriving requirements engineering artifacts by using business process models. Automated generation is not supported in this study.

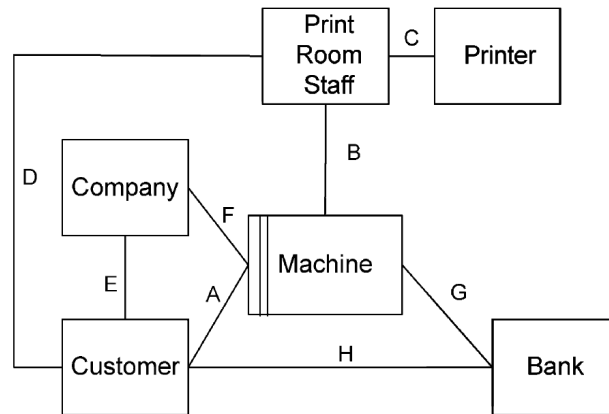


Figure 3: An example Jackson context diagram derived from RAD (Cox et al. 2005)

2.3.2 Approach of Stolfa & Vondrak (2004)

Stolfa & Vondrak (2004) describe business process models as a tool for deriving software requirements. The study presents mapping from activity diagrams to use case models. There types of mapping patterns are defined that are sequential, optional and branching patterns as provided in Figures 4, 5 and 6 respectively.

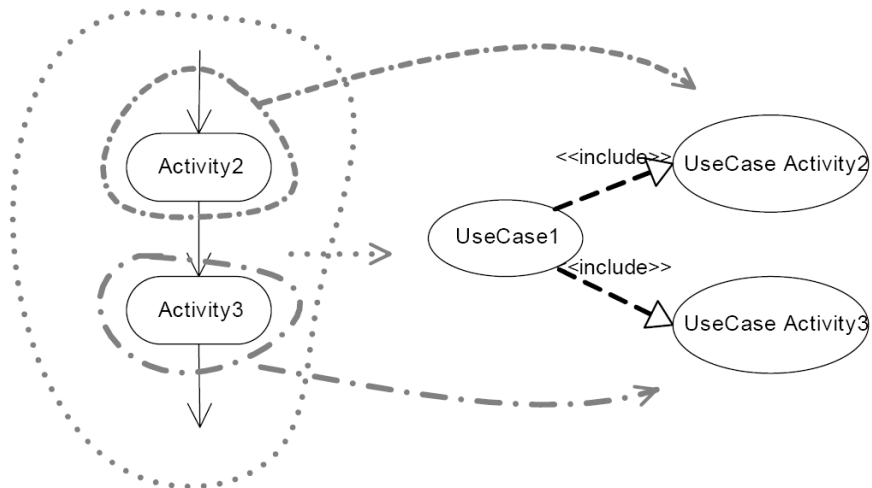


Figure 4: Sequential pattern (Stolfa & Vondrak 2004)

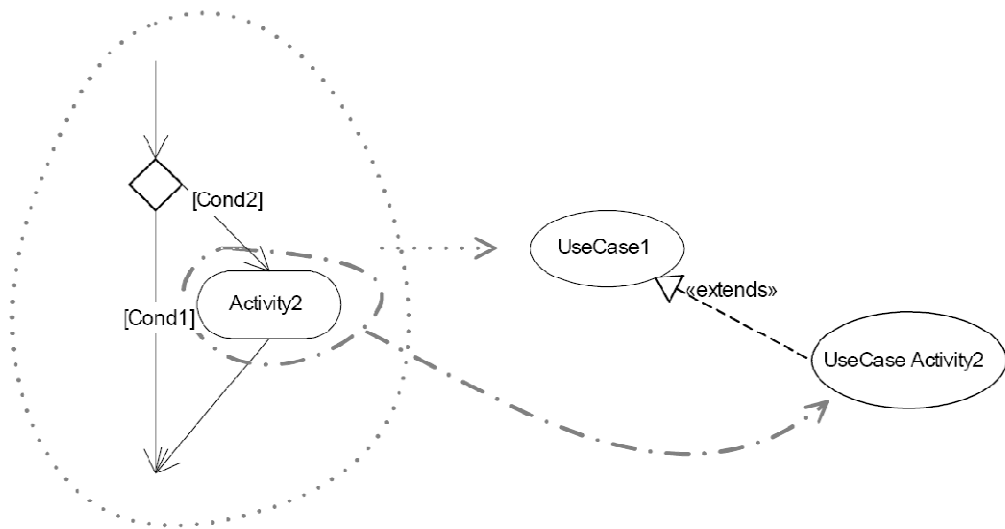


Figure 5: Optional pattern (Stolfa & Vondrak 2004)

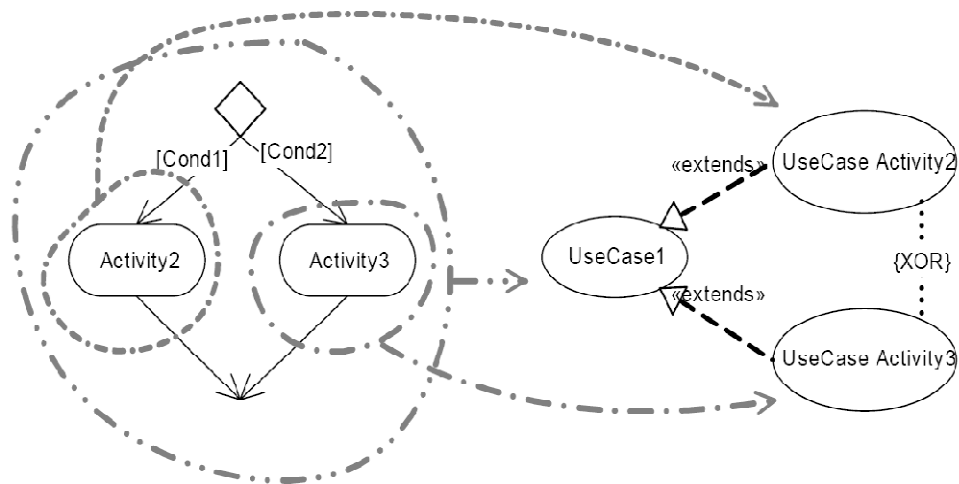


Figure 6: Branching pattern (Stolfa & Vondrak 2004)

The mapping activities consist of three phases. The first phase of the mapping is to decide which activities in the activity diagrams will be supported by information systems and which will be performed manually. Next phase is to determine the activities that will be included in each use case. The use cases might be composed of one or many activities that are represented in activity diagrams. Finally, in order to derive the relationships in use case models from activity diagrams, the three types of mapping patterns are utilized.

Automation tool for the transformation in this study is reported to be currently being developed.

2.3.3 Approach of Specht et al. (2005)

Specht et al. (2005) presents a methodology to model business processes and transform them to Business Process Execution Language (BPEL). For modeling business processes EPC models and function allocation diagrams are used. Since EPC models do not include details about flow of activities and operations behind each function that are necessary for transformation to BPEL, function allocation diagrams are used and characterized with flow of user interaction activities, data entities and application systems and their operations. The function allocation diagrams that are referred as extended function allocation diagram (eFAD) contain all the operations with inputs and outputs on the application systems. An example eFAD is provided in Figure 7.

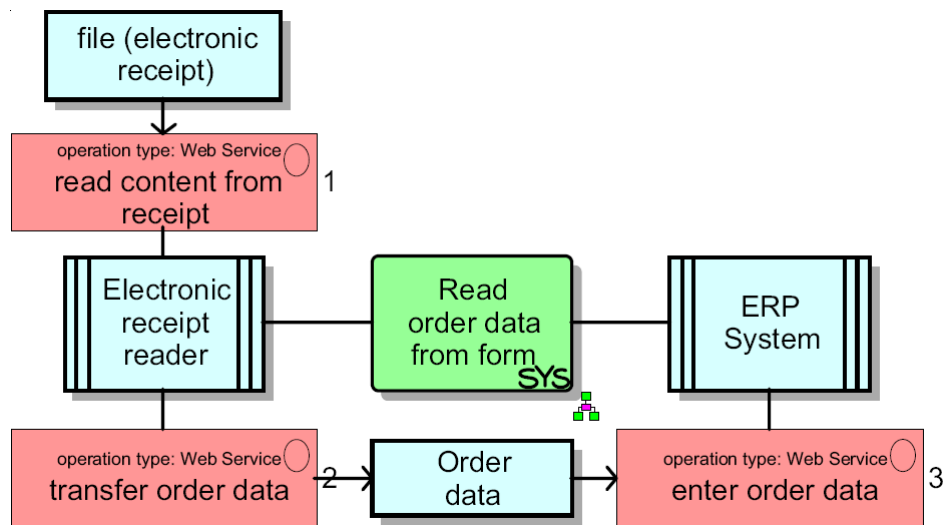


Figure 7: An example eFAD (Specht et al. 2005)

The methodology does not introduce a full transformation to BPEL that EPC models and eFADs lack some technical details that are required for BPEL. So, Specht et al. (2005) provides a methodology for transformation from business process models to a BPEL skeleton. An example BPEL skeleton derived in the study is provided below.

```

<scope name="readOrderDataFromForm">
  <variables>
    <variable name="archiveReferenceMessage"
      messageType="tns:ReadOrderDataRequestMessage"/>
    <variable name="orderData"
      element="datatypes:orderData"/>
    <variable name="orderDataMessage"
      messageType="tns:collectOrderDataRequestMessage"/>
  </variables>
  <sequence>
    <assign name="assignArchiveReference">...</assign>
    <invoke name="readOrderDataFromReceipt"
      partnerLink="ReceiptReader"
      portType="tns:ReceiptReader"
      operation="readOrderData"
      inputVariable="archiveReferenceMessage"
      outputVariable="orderDataMessage"/>
    <invoke name="collectOrderData" partnerLink="ERP-System"
      operation="collectOrderData"
      portType="tns:ERPSystem"
      inputVariable="orderDataMessage"/>
  </sequence>
</scope>

```

Although Specht et al. (2005) claims that the transformation from EPC models and eFADs to BPEL skeleton is suitable for automatic generation, the transformation in the study is reported to be done manually. The approach is a rare example for utilizing function allocation diagrams in separating software requirements related information from business process models.

2.3.4 Approach of Su (2004)

Su (2004) presents the KAOS tool that automatically generates requirements in natural language from business process models. KAOS tool is a plug-in for the ARIS toolset. KAOS tool utilizes business process models which are in the form of eEPC models. The eEPC models used are the TO-BE representations of the business processes. The TO-BE representations of the business process models in the study are the definitions of the business processes resulted from the business process reengineering activities. These representations define the business processes which include functions that are supported by information systems without any exceptions.

eEPC notation is modified by defining color codes to information carriers and introducing a naming convention for functions. An example business process model is provided in Figure 8.

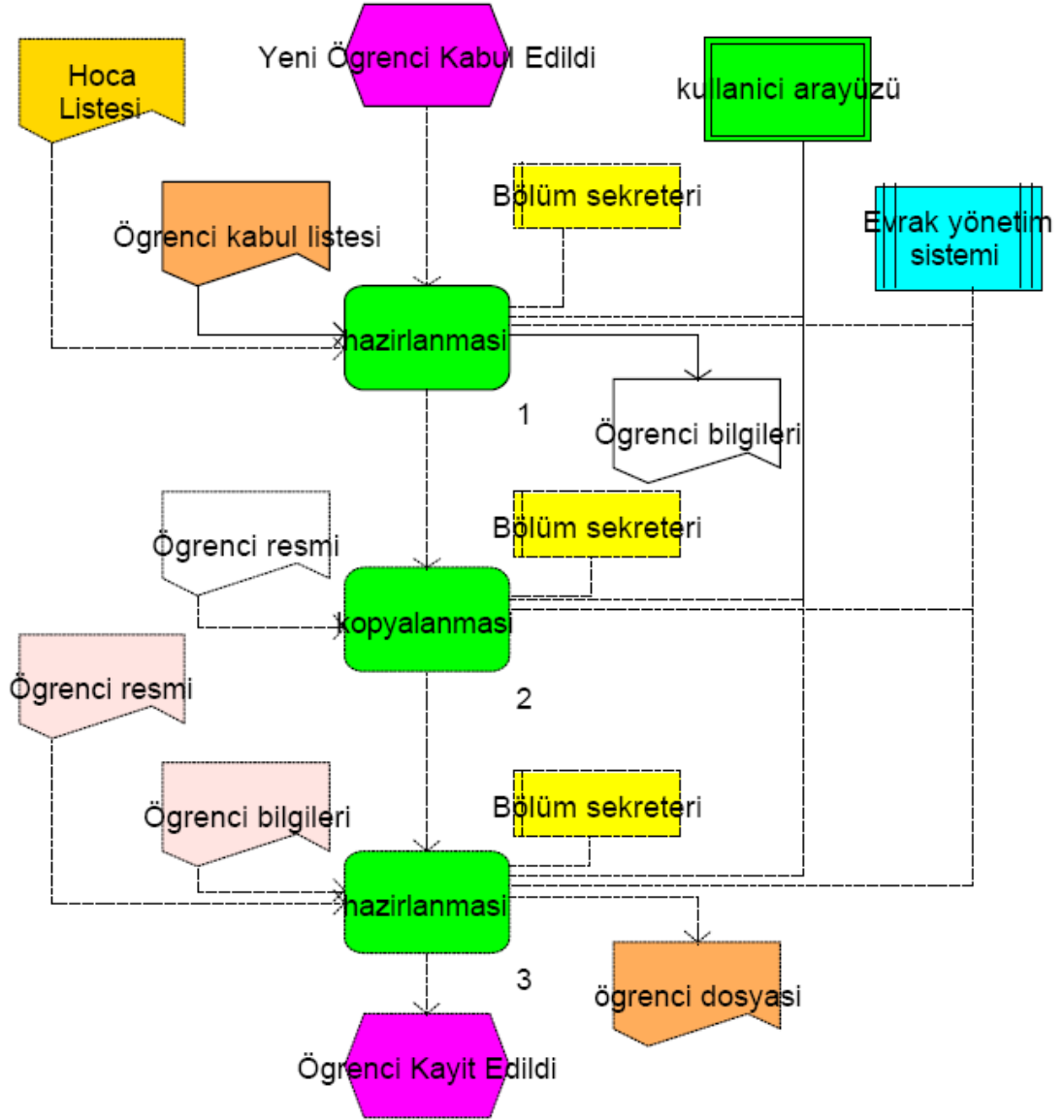


Figure 8: An example eEPC model used for requirements generation by KAOS tool (Su 2004)

Natural language requirements sentences generated by KAOS tool using the eEPC model in Figure 8 are provided below.

“Evrak yönetim sisteminde, Öğrenci kabul listesi ve Hoca Listesi kullanılarak Bölüm sekreteri tarafından, kullanıcı arayüzü ile Öğrenci bilgileri'nin hazırlanmasına olanak sağlamalıdır.”

“Evrak yönetim sisteminde, Öğrenci resmi'nin, Bölüm sekreteri tarafından, kullanıcı arayüzü ile Kopyalanmasına olanak sağlamalıdır.”

“Evrak yönetim sisteminde, Öğrenci bilgileri ve Öğrenci resmi kullanılarak Bölüm sekreteri tarafından, kullanıcı arayüzü ile öğrenci dosyası'nin hazırlanmasına olanak sağlamalıdır.”

Study of Su (2004) is the only study in the literature that utilizes eEPC models in natural language software specification generation.

These four approaches in deriving software requirements from business process models are compared in Table 1.

Table 1: Comparison of the mainstream approaches in the literature

	Initial business process model	Generated requirements artifact	Generation tool	Validation
Cox et al. (2005)	RAD	Jackson context diagram	Not reported	E-business system
Stolfa & Vondrak (2004)	Activity diagram	Use case model	Reported to be in process of development	Car sale example
Specht et al. (2005)	eEPC and eFAD	BPEL	Not reported	Document processing scenario
Su (2004)	eEPC	Natural language specifications	KAOS tool (plug-in for ARIS)	Military project

CHAPTER 3

PROCEDO: REQUIREMENTS GENERATION APPROACH

This chapter includes three sections. First section describes models utilized, specific sets of model elements and links between them. In second section, the unified process for performing business process modeling and software requirements analysis is introduced. Third section presents sentence and document structures for natural language specifications to be generated and the tool support for generating natural language specifications automatically from business process models.

3.1 Utilization of Business Process Models

Extended EPC (eEPC) models and Function Allocation Diagrams are at the core of Proceso.

A restricted set of eEPC model elements are used in Proceso. Among the restricted set of model elements; the ones that are the main building blocks of lean EPC models, namely the functions, events and logical operators are defined in Section 2.2.1.

Business rule objects that are required to be stated in business processes are represented in eEPC models as connected to functions. The business rules are utilized to denote rules enforced by legislations and state process specific constraints that cannot be depicted by the activity flow.

Information carriers represented with different symbols that define their types define the physically stored data in the form of inputs and outputs of the functions (Davis & Brabander 2007).

Process interface objects are used to define connections between business process models by providing links between two consecutive EPC models (Mendling 2008). A process interface in a business process model indicates that the model continues with another business process linked by the process interface.

The restricted set of eEPC model element representations in Proceso is provided in Table 2.

Table 2: eEPC model element representations in Proceso

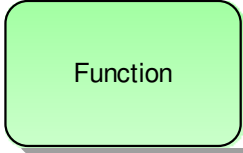
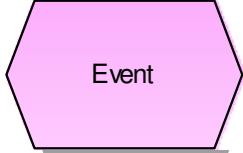



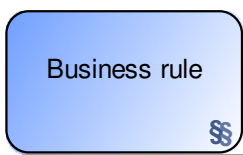


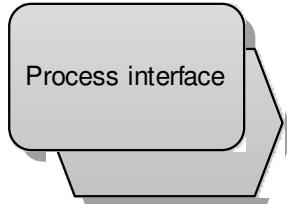
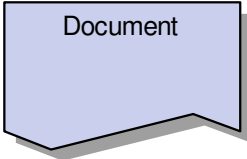
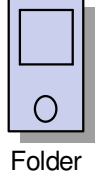
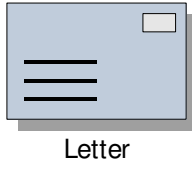
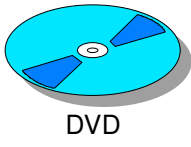
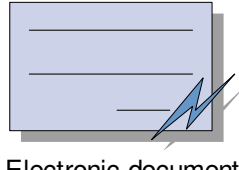
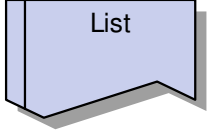
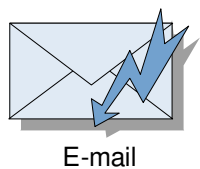
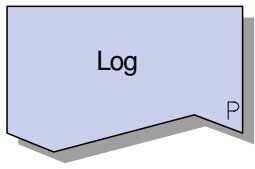
Object Name	Object Symbol
Function	
Event	
AND	
OR	
XOR (eXclusive OR)	
Business Rule	

Table 2: (continued)

<p>Position and Organizational Unit</p>	 
<p>Process Interface</p>	
<p>Information Carrier</p>	       

eEPC models would be organized in different settings. In this study, it is recommended to use EPC column display to improve the readability of the models. However, this is not a restriction that constraints the approach. In eEPC column display, the models are separated into columns and at the top of each column there exists the actors represented by position and organizational unit objects that are

responsible to perform the activities. The actors at the top of the columns are linked to functions implicitly, ensuring the actors responsible to perform the functions are defined (Scheer et al. 2006).

There exist hierarchical relationships between eEPC models. These relationships are maintained by assignment relations created from function objects of superior models to the subordinate models.

Component processes that are required by more than one eEPC model can be referenced anywhere as a process interface. Process interfaces and sub-processes are the key mechanisms to form the hierarchical and modular structure of processes. Hierarchy of processes and process interfaces can be utilized to form a process map in high level and reveal interfaces between process modules.



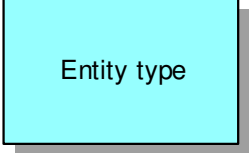
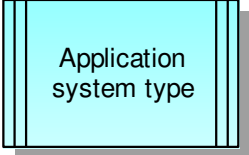

The other business process model type used in Proceso is the FADs. FADs are maintained in business process models via assignments created from the function objects in eEPC models. By this way; each function in an eEPC model might have one FAD assignment and similarly, each FAD should be assigned to exactly one function in eEPC models.

The purpose of utilizing FADs in Proceso is to define the roles, entities, actions, application systems and business rules that takes part while conducting the related activity defined by a function. The restricted set of model element representations of FADs utilized in Proceso are provided in Table 3.

The FAD of the Proceso takes a function object in the center of the notation and each FAD should include exactly one function object. Each function object in a FAD is an occurrence copy of another function object in an eEPC.

In FADs, the position objects represent the roles that perform the activity on the application systems. There might be multiple roles that are authorized to perform the defined activity on the application system. Also the position object in FADs is not necessary to be a copy the positions or organizational units in the hierarchically superior eEPC model, since the activities would be performed by other roles on the application system.

Table 3: Object representations in FAD notation

Object Name	Object Symbol
Function	
Position	
Entity Type	
Application System Type	
Business Rule	

Entity Type objects are the representation of system entities maintained in application systems. This object can represent any entity that can exist, can be used, changed or deleted in information system. The inputs and outputs in eEPC models and additional entities that are required are specified as entity type in FADs.

Application System Type object represents the application system that is intended to be developed in accordance with the IS integrated representation of defined business process. There might be one or many application systems used when performing an activity, so there might be multiple application systems represented in each FAD.

Business Rule object is used for specifying the business rules in business processes, which can be translated into system specifications. Since we are constraint by the

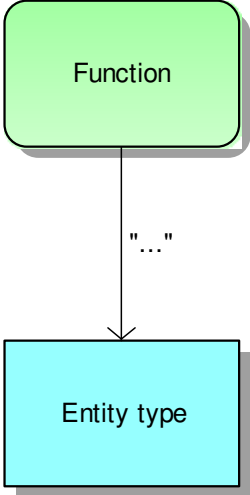
tool capabilities, an existing object representation that is not intended to be used in this study is used in designating the business rules. Business Rule objects are connected to the Application System Type objects for which the rules are provided.

The connection types between function and entity objects are an important part of FADs. The connection type designates the operation on an entity while the function is performed on the related application system. These connections do not define sequence. They are rather representations of behavioral unit responses. All operations in a FAD are completed when the function is performed.

There are seven connection types between function and entity type objects. These connection types are inspired by the CRUDL. The connection type, **creates**, indicates that the entity on the target is created on the application system. **Changes** connection type is used to show that the entity already exists on the application system and it is updated during the related activity. **Reads** connection type indicates that the related entity is read from the database, while **views** connection type means that the entity is read from the database and then viewed by the user. **Reads** connection types are used if the existence of the entity is prerequisite for the activity to be performed or the entity is input to another entity to be created or updated. **View** connection type on the other hand, shows that the entity is needed to be displayed to the user so that the user performs the activity. **Lists** connection type is used if there is an entity of type list that the user is required to list the entities and select one. **Uses** connection type is used if the related entity's use is not clear or may include any or many of operations of type create, update, read, view and delete. Finally, **deletes** connection type exhibits that the entity is required to be deleted. The connection representation between function and entity objects is given in Table 4.

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. The numbered label property is provided by the use of connection rule attributes of the connection types in ARIS Business Architect tool. These attributes are null in their default states. Connection role attribute of a connection being not null means that one or many of the connections that have connection role attributes of the same value are performed.

Table 4: Connection representations between function and entity type objects in FADs

Connection Type Name	Connection Representation
Creates	
Changes	
Reads	
Views	
Lists	
Uses	
Deletes	

The eEPC models and FADs are named after the function objects that they are assigned to. Since an occurrence copy of the function object, which the FAD is assigned to, is included by the FAD; we might also say that each FAD is named after the function object that it includes.

The process hierarchy is defined by using group structures that include process representations in a folder view. Each business process model is included in a group having the same name with the eEPC model. The master objects and FADs created in related eEPC models are included in the same group structure. Within each group belonging to a process model which has sub-processes, there exist the groups that belong to these sub-processes. So, by this way the business process models are organized in a hierarchical manner using a grouping approach.

3.2 A Unified Process: Bridging the Gap between Business Process Modeling and Requirements Analysis

The process described in this section bounds the modeling notation presented in Section 3.1 and natural language specification generation presented in Section 3.3. It constitutes guidelines for real life application. Also the process would guide and

constitute the initial phases of many software development life cycle models in practice.

The unified process for performing business process modeling and software requirements analysis activities concurrently is composed of seven consecutive high level activities or, in other words, steps. Details of these steps are described below and a high level overview of the unified process is provided in Figure 9.

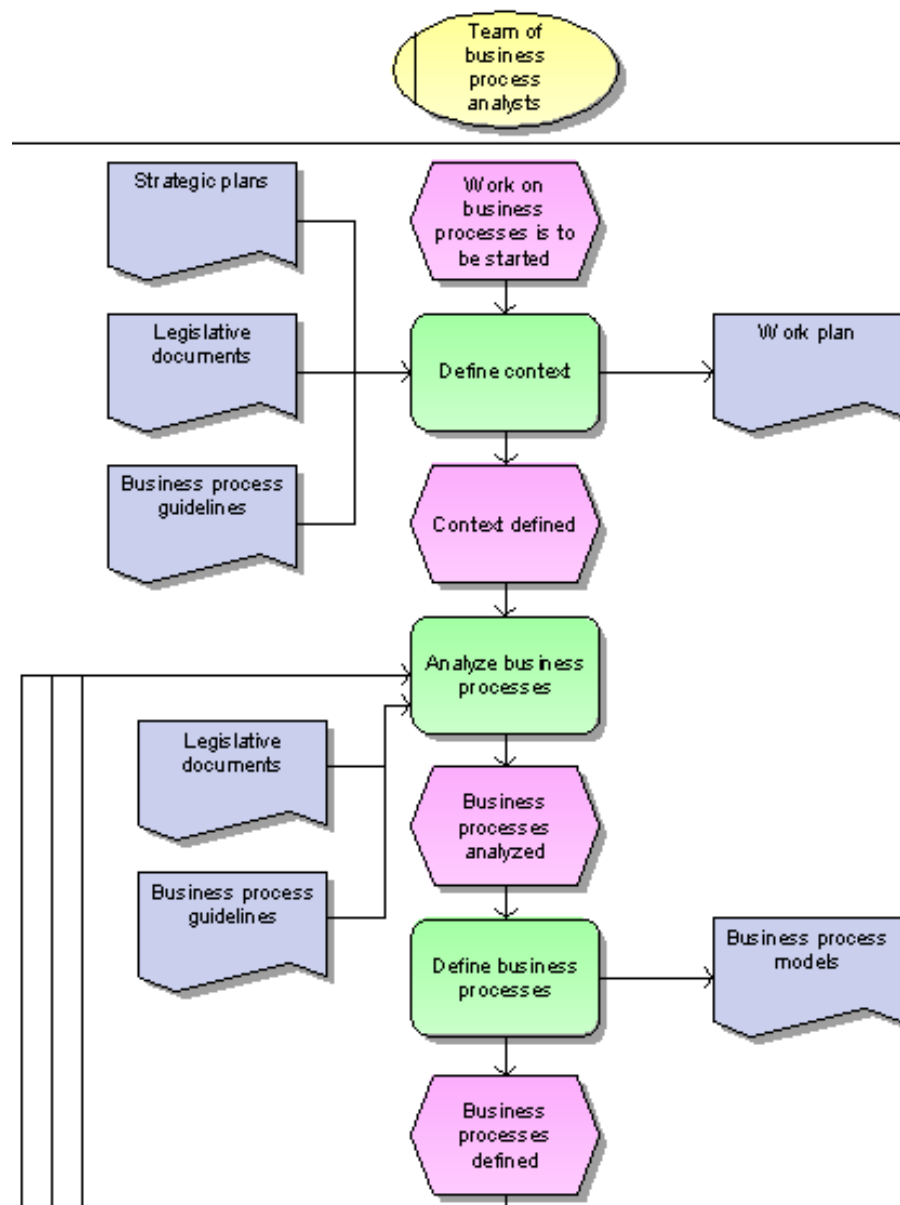


Figure 9: Process for bridging the gap between business process modeling and requirements analysis

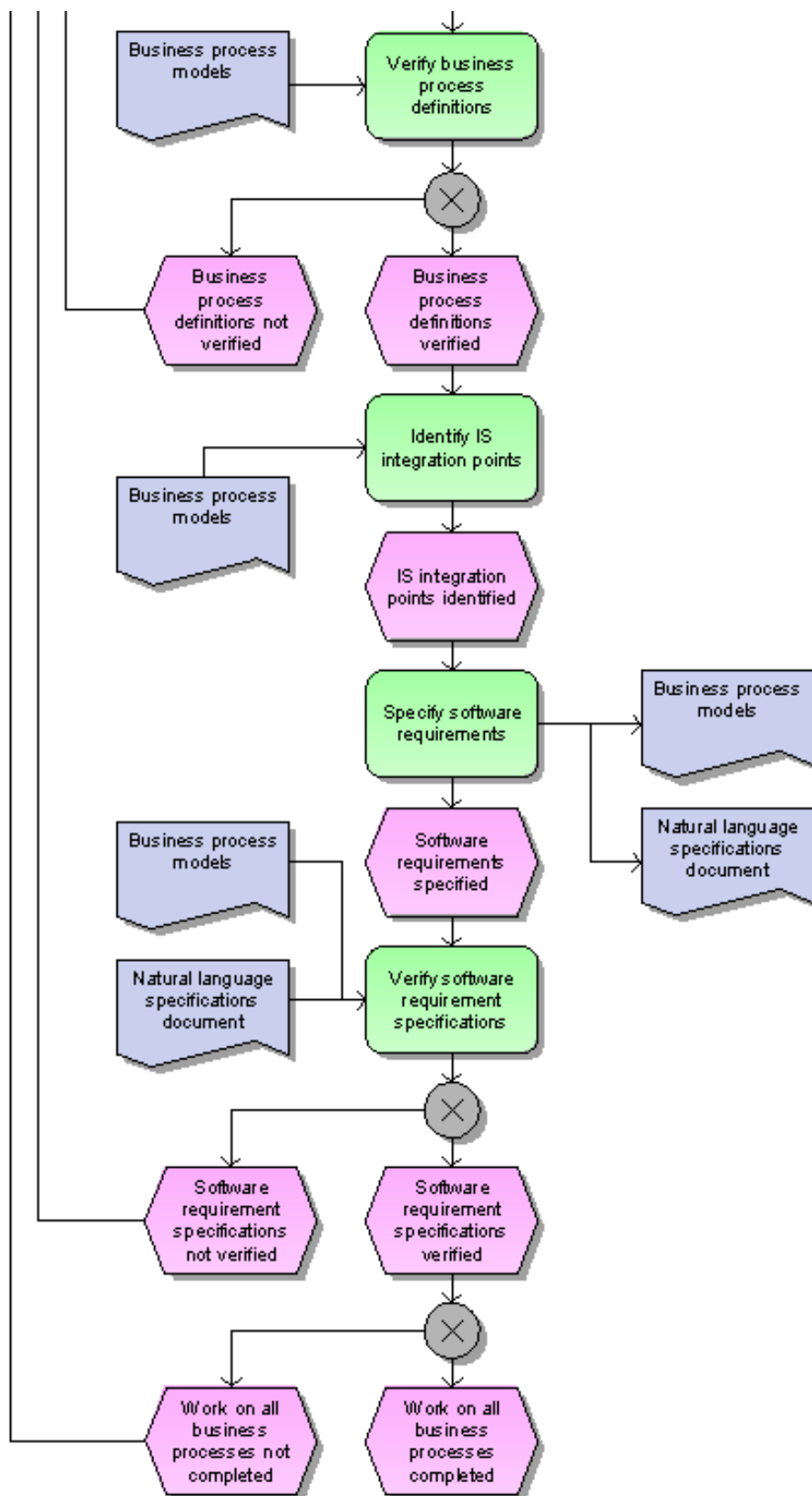


Figure 9: (continued)

Define context: This is the initial step of the process. It is the preparation step for the process where the boundaries and plans of the study are established, by which the success of the study is highly stimulated. Activities given below for this step are crucial to be approved by all stakeholders of the study.

- Identify purpose and scope of the study:

The purpose of the study is determined in this step. Business processes to be defined and the information systems to be developed are identified in high level. By this way, the scope of the study would be established. Strategic plans, if there are any, are the inputs of this step.

- Identify business process modules:

The work is divided into business process modules. The boundaries of the modules would be inspired from the boundaries of information systems, if there would be multiple of them to be developed or from the boundaries of the legislative documents and process guidelines that are already established within the organization. Coordination teams for each module are set.

- Plan execution:

Work plan is developed including work breakdown structure, schedule for tasks and milestones, deliverables, resource allocation plan and risk and configuration management plans. Different process modules might require different process expertise, so different subject matter experts for each module should be considered in resource allocation.

Analyze business processes: Process analysis is conducted in a top-down approach. That is; high level processes are analyzed first and then lower levels are detailed as sub-processes. In each iteration, the business processes to be analyzed are a selected set of business processes in a process module that are designated in the work plan.

- Identify process guides and rules:

In process analysis, the starting point would be identification of the guidelines and legislative documents that are related to the highlighted processes. Guides and rules

that govern the processes should be extracted from these documents and brought together.

- Identify inconsistencies and resolve conflicts:

The guides and rules that are brought together should be analyzed with the subject matter experts. Conflicts would rise from inconsistencies between several documents that guide the processes or from the inconsistencies between the subject matter expertise and these documents. Conflicts should be negotiated with the subject matter experts and as a result, agreement on terms should be achieved for the processes prior to the process definition phase.

Define business processes: This step basically is focused on development of eEPC models.

- Define process flows:

First, the process flows in eEPC models are constructed. The process flow in eEPC models includes functions, events, logical operators and process interface object types. By having process flows defined, skeletons of eEPC models would be formed and so, debates on remaining aspects of the business processes, namely; the process roles, rules, inputs and outputs, would be done based on the process flow.

- Define roles, inputs, outputs and business rules in processes:

After the process flow is constructed, business process models are extended with other aspects of processes. Roles that perform the activities, inputs and outputs of the processes and business rules are defined within eEPC models.

Verify business process definitions: This step aims that the business process models prior to requirements analysis are complete and correct.

- Perform walkthrough:

eEPC models are reviewed one by one and correction explanations are determined. The walkthroughs are performed with the subject matter experts. In order to include

various viewpoints in the validation; subject matter experts, who have not participated in previous activities, would take part as an external review team.

- Revise process definitions:

eEPC models are revised with respect to the walkthrough results.

Identify IS integration points: Requirements analysis with business process models starts with choosing the functions in eEPC models that are intended to be automated with information systems. When determining the points in the business processes that are desired to be automated, eEPC models are visited one by one and expectations of the subject matter experts are elicited in high level of detail. The functions that are chosen to be supported by information systems will be assigned to a FAD.

Specify software requirements: This step consists of revising the business process models by constructing FADs and generating natural language specifications.

- Define software requirements information in business process models:

Software requirement information is added to business process models via FADs. Construction of FADs is a stepwise process described as follows. 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 operations between the function and the application systems. Each entity is connected to an application system that the related entity is to be contained in. Finally, the business rules on FADs are inspired by the business rules already placed in EPC models and business process guidelines.

- Generate natural language specifications:

After the business process models are constructed with EPC models and FADs; natural language specifications are generated with the tool support.

Verify software requirement specifications: Requirement specifications are reviewed and revised in this step.

- Review requirement specifications:

FADs and natural language specifications are reviewed and detected issues are recorded to be revised.

- Revise requirements specifications

Revision of the requirement specifications starts with revision of FADs according to the issues detected and recorded in review. Then, natural language specifications are generated with tool support again from the business process models.

3.3 Automated Generation of Natural Language Specifications

3.3.1 Natural Language Specification Sentence Structure

The sentence structure for a natural language specification sentence is provided below.

“activity sırasında, role tarafından operations on system”

In the sentence structure given above, the words written in bold represent the dynamic parts of the sentence structure and the remaining words and characters represent the static parts. The structures of the dynamic parts, which contain information referred from FADS, of the sentence structure are explained below.

Activity structure is composed of the name of the function object in FAD.

Example; **“Nihai ödemenin tespiti** sırasında, İDB Uzmanı tarafından PFDS sistemi üzerinde Ödeme Planı değiştirilebilmeli, Ön Ödeme Miktarı, Ara Dönem Ödeme Miktarı, Proje Nihai Destek Miktarı okunabilmeli, Proje Nihai Ödeme Miktarı, Onay İsteği yaratılabilmelidir.”

Role structure represents the names of the position objects in FAD. Since there would be one or multiple position objects in FADs, **role** structure is characterized in the sentence structure as follows;

- If there is only one position object named “P” connected to the function, then the **role** structure would be; “P”.

Example; “Nihai ödemenin tespiti sırasında, **İDB Uzmanı** tarafından PFDS sistemi üzerinde Ödeme Planı değiştirilebilmeli, Ön Ödeme Miktarı, Ara Dönem Ödeme Miktarı, Proje Nihai Destek Miktarı okunabilmeli, Proje Nihai Ödeme Miktarı, Onay İsteği yaratılabilmelidir.”

- If there are more than one position objects named “P1”, “P2”...and “Pn” connected to the function, then the **role** structure would be; “P1, P2, ... ve Pn”.

Example; “Teklif Çağrısı Konularının belirlenmesi sırasında, **PPB Uzmanı ve PYB Uzmanı** tarafından PFDS sistemi üzerinde Teklif Çağrısı Konu indexi (NACE US97) okunabilmeli, Teklif Çağrısı Konusu yaratılabilmeli, ÇPBMS sistemi üzerinde Çalışma Programı görüntülenebilmelidir.”

Example; “Başvuru rehberi taslağının gözden geçirilmesi sırasında, **PPB Uzmanı, İDB Uzmanı ve GS** tarafından PFDS sistemi üzerinde Onay Listesi değiştirilebilmeli, Başvuru Rehberi görüntülenebilmeli, Onay isteği okunabilmeli, Onay durumu yaratılabilmelidir.”

Operations on system structure relies on entity type and application system type objects in FADs and the connection types between them. The connection types that are translated into Turkish words in the form to be included in the sentence structure are provided in Table 5.

Operations on system structure has both dynamic and static aspects that characterizes the linguistic properties of the sentences. These dynamic aspects originate from the type and number of objects in FADs and are described with definitions and examples below.

- If there is one application system type named “A”, one entity type named “E” and connection type of “C”, then **operations on system** structure would be; “A sistemi üzerinde E Cdir.”.

Example; “Ajans mali yönetim yeterliğinin değerlendirilmesi sırasında, DPT Uzmanı tarafından **DenetYS sistemi üzerinde Denetim raporu görüntülenebilmelidir.**”

Table 5: Transition from connection types to sentence structure

Connection Type Name	Transition to Sentence
Creates	yaratılabilmeli
Changes	değiştirilebilmeli
Reads	okunabilmeli
Views	görüntülenebilmeli
Lists	listelenebilmeli
Uses	kullanılabilmeli
Deletes	silinebilmeli

- If there is one application system type named “A”, more than one entity types named “E1”, “E2”,...and “En” and for all entity types the connection type is of “C”, then **operations on system** structure would be; “A sistemi üzerinde E1, E2, ..., En Cdir.”.

Example; “Ceza ve tazminatların kaldırılması sırasında, Muhasebe Uzmanı tarafından **ÇPBMS sistemi üzerinde Gelirler hesabı kaydı, Kişilerden alacaklar hesabı kaydı, Muhasebe işlem fişi yaratılabilmelidir.**”

- If there is one application system type named “A”, more than one entity type objects named “E1”, “E2”,...and “En” and for the entity types the connection types are of “C1”, “C2”,...and “Cn” respectively, then **operations on system** structure would be; “A sistemi üzerinde E1 C1, E2 C2, ..., En Cndir.”.

Example; “Yararlanıcıya yapılacak toplam ödeme miktarının eş finansman gerçekleşme oranına göre indirilmesi sırasında, İDB Uzmanı tarafından **PFDS sistemi üzerinde Proje Nihai Destek Miktarı değiştirilebilmeli, Başvuru Rehberi okunabilmelidir.**”

- If the above statement is changed as two or more entities have the same connection type on the same application system, **operations on system** structure would be; “A sistemi üzerinde E11, E21, ..., En1 C1, E12, E22, ..., En2 C2, ..., E1n, E2n, ..., Enn Cndir.”.

Example; “Nihai ödemenin tespiti sırasında, İDB Uzmanı tarafından **PFDS sistemi üzerinde Ödeme Planı değiştirilebilmeli, Ön Ödeme Miktarı, Ara Dönem Ödeme Miktarı, Proje Nihai Destek Miktarı okunabilmeli, Proje Nihai Ödeme Miktarı, Onay İsteği yaratılabilmelidir.**”

- If there are more than one application system type objects named “A” and “B”, each of which have entity type objects connected to it, then **operations on system** structure would be; “A sistemi üzerinde E11, E21, ..., En1 C1, E12, E22, ..., En2 C2, ..., E1n, E2n, ..., Enn Cn, B sistemi üzerinde ... dir.”.

Example; “DFD çalışmalarının planlanması sırasında, PYB Uzmanı tarafından **PFDS sistemi üzerinde DFD Listesi değiştirilebilmeli, DFD, DFD Planı yaratılabilmeli, PPS sistemi üzerinde Bölgesel Operasyonel Program görüntülenebilmeli, ÇPBMS sistemi üzerinde Çalışma Programı görüntülenebilmeli, Yıllık Bütçe okunabilmeli, DFD Bütçesi yaratılabilmelidir.**”

- If there are selective execution of operations in FADs, then **operations on system** structure includes the word “veya” as given in the examples below;

Example; “Eğitim/danışmanlık hizmetleri için satınalmanın başlatılması sırasında, PYB Uzmanı tarafından **PFDS sistemi üzerinde PTÇ Destek Faaliyetleri Planı, veya Bilgilendirme toplantı planı, veya Teklif Çağrısı Eğitim Planı(TÇEP), veya BD eğitim ve çalışma takvimi görüntülenebilmelidir.**”

Example; “Raporun şekli uygunluk kontrolünün yapılması sırasında, İzleme Uzmanı tarafından **PFDS sistemi üzerinde Ara rapor, veya Nihai Rapor görüntülenebilmeli, Nihai Rapor Kontrol Listesi, veya Ara Rapor Kontrol Listesi yaratılabilmeli, Ek I-19: Nihai Rapor Kontrol Listesi, veya Ek I-18: Ara Rapor Kontrol Listesi okunabilmeli, Proje Sözleşme Listesi listelenebilmeli, Ek I-20: Ara ve Nihai Donem Raporu Veri Giriş Formu okunabilmelidir.**”

Another specification sentence structure is developed for specifying software requirements related to business rules. Sentence structure utilizes the application system type objects in FADs and the business rules connected to them. The sentence structure is provided below.

“Application System sistemi üzerinde; business rule.”

Example; **“PFDS sistemi üzerinde; nihai ödeme miktarı, nihai destek miktarından Ajans tarafından yapılan ödemelerin toplamı düşülerek hesaplanabilmelidir.”**

3.3.2 Software Specification Document Structure

A document structure is constructed to manage the natural language specification sentences. To enable forward traceability between business processes models and natural language specifications, process names and paths are added to the document structure. Also, with the same reason, each specification sentence is tagged with a requirement number which includes module information that indicates the specification sentence belongs to the specified module. Additionally, specification numbers are given to each specification sentence that increment cumulatively, which can be seen in the document structure with indicators as s1, s2 and so on. The resulted document structure is as follows;

- n1. Süreç adresi: **“process path 1”**
 - n1.m1. Süreç adı: **“process name 1”**
 - n1.m1.k1. **“Module name”** s1: **“Specification sentence 1”**
 - “business rule sentence 1”
 - n1.m1.k2. **“Module name”** s2: **“Specification sentence 2”**
 - “business rule sentence 2”
 -
 - n1.m1.kX. **“Module name”** sX: **“Specification sentence X”**
 - “business rule sentence X”
- n2. Süreç adresi: **“process path 2”**
 - n2.m1. Süreç adı: **“process name 2”**
 - n2.m1.k1. **“Module name”** sX+1: **“Specification sentence X+1”**
 - “business rule sentence X+1”
 -

An example specification that is constructed by using the document structure described above is provided in Appendix D.

3.3.3 Requirements Generation Tool

A requirements generation tool is developed to generate natural language specifications based on the sentence and document structures described in previous parts. The flowchart that denotes the design of the tool is provided in Figure 10.

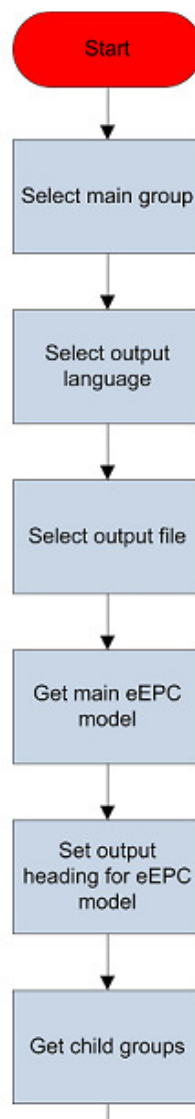


Figure 10: Flowchart for requirements generation tool

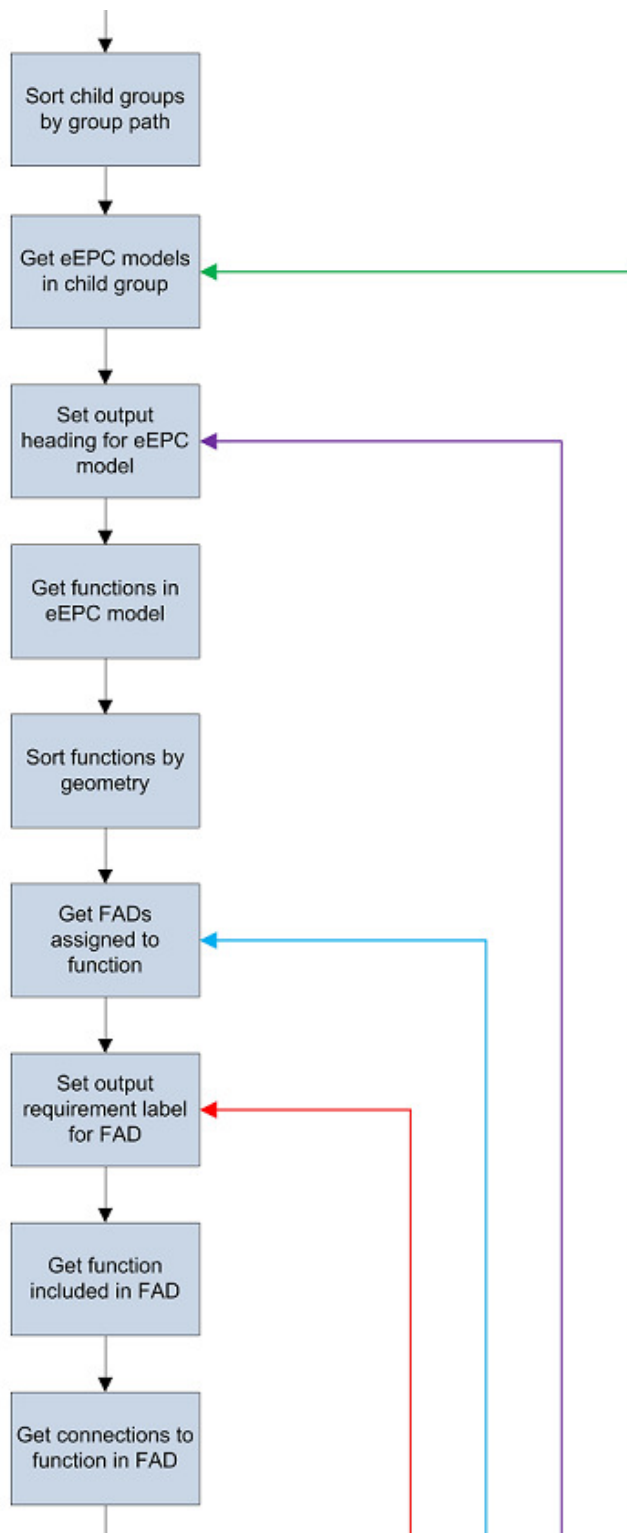


Figure 10: (continued)

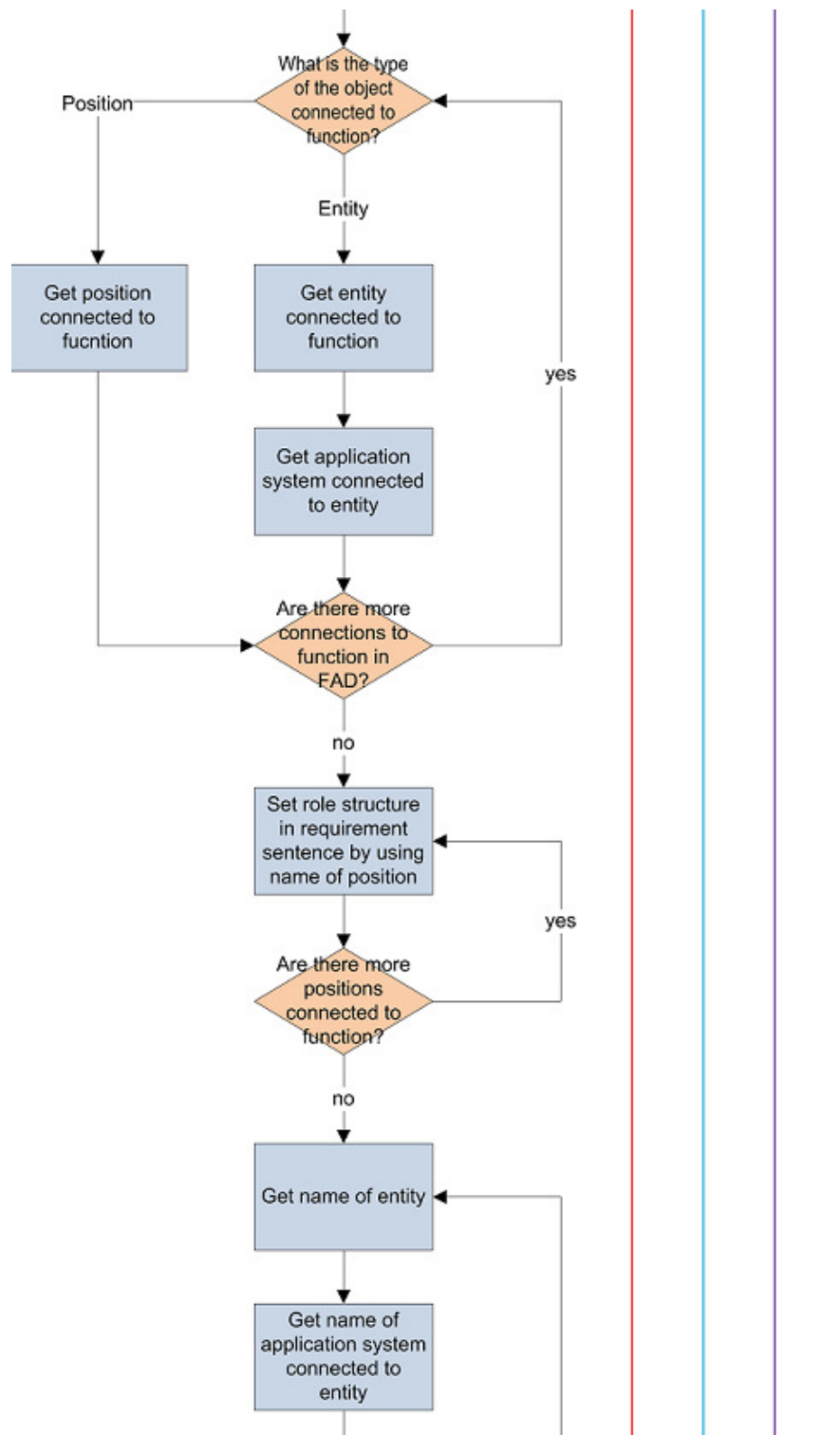


Figure 10: (continued)

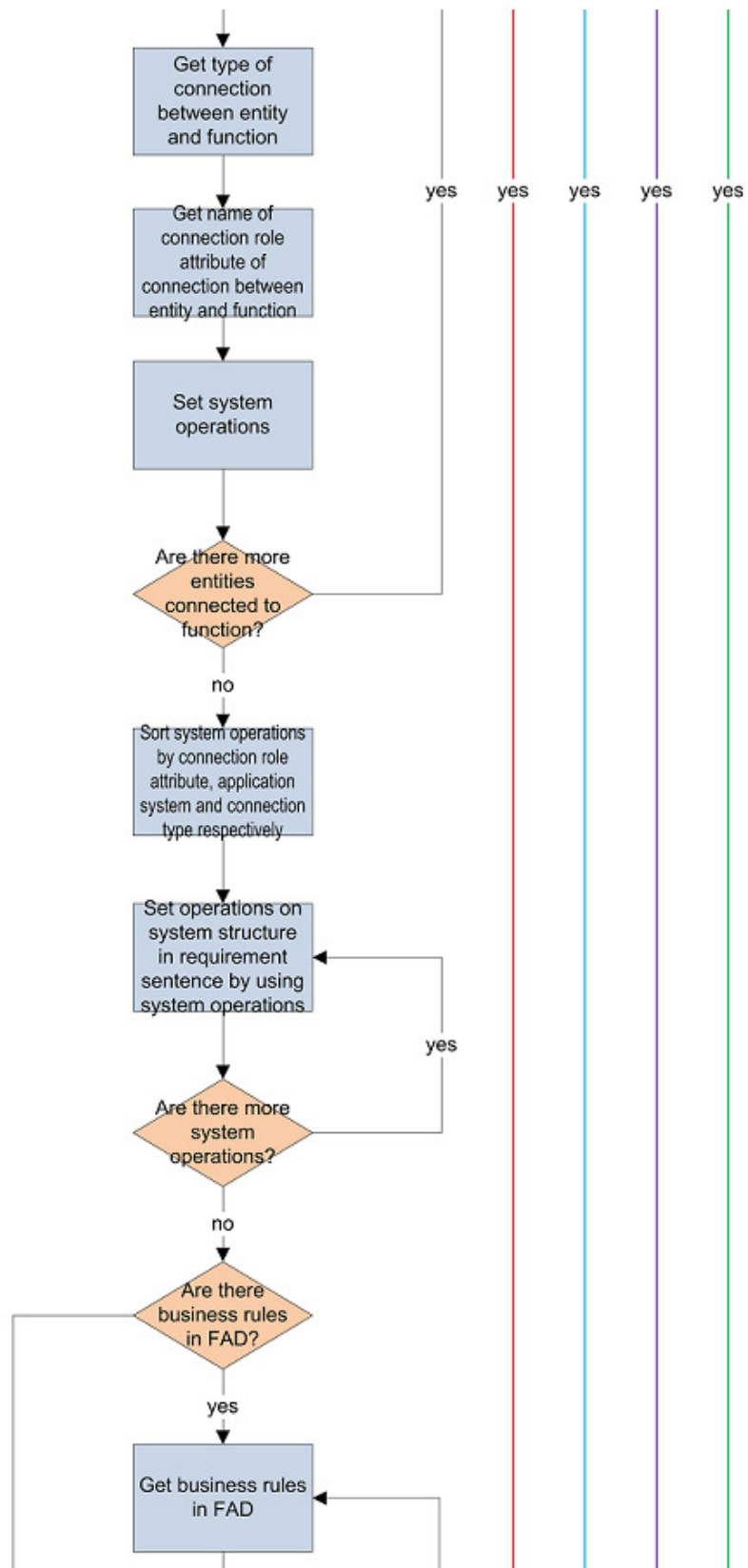


Figure 10: (continued)

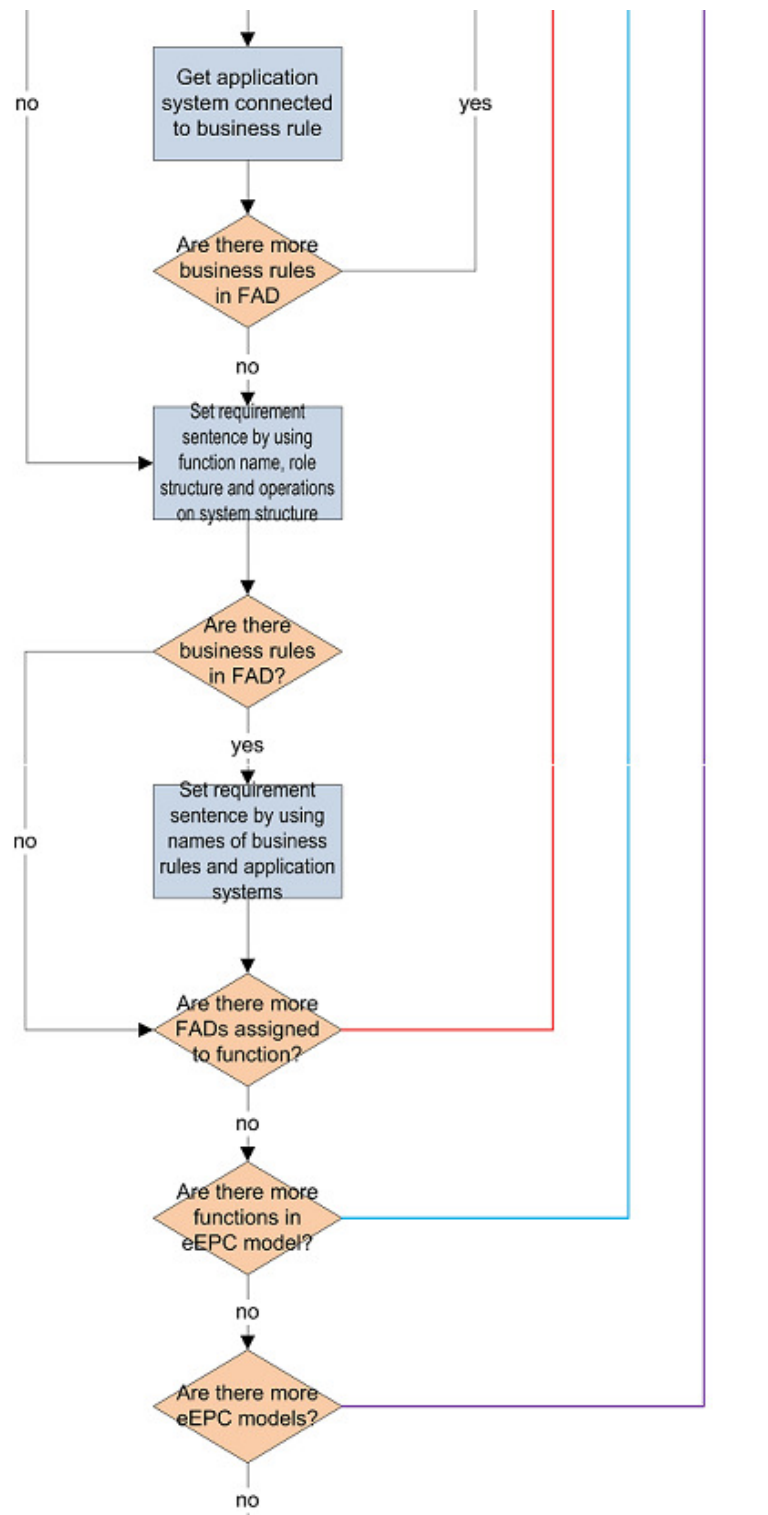


Figure 10: (continued)

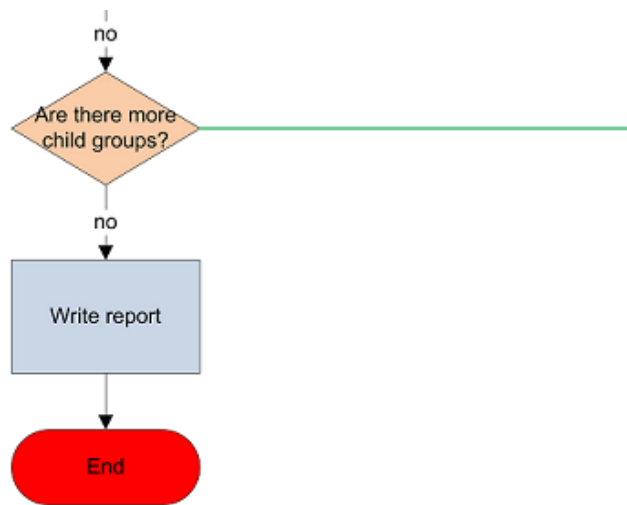


Figure 10: (continued)

The tool is a script developed on ARIS Script Editor in ARIS Business Architect v7.1. The script language is very similar to Java Script language. The classes and methods defined in ARIS Script Editor are used in the coding phase. The usage of the script in the ARIS tool suit is described in Appendix E.

CHAPTER 4

CASE STUDIES

Our purpose in this study is to explore utilization of business process models in requirements analysis. With this purpose, the following research questions are identified.

Research Question 1: How can business process models be utilized for requirements analysis?

The aim is to develop an approach to conduct requirements analysis activities concurrently with business process modeling, so that business process knowledge can be transferred to the software requirements.

Research Question 2: Does using business process models for requirements analysis increase the efficiency of business process modeling and requirements analysis activities in terms of total effort?

The aim is 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.

In order to explore answers for the research questions, two case studies were performed. First case study focuses to seek answers for the first research question, whereas second case study explores answers for the second research question.

In this chapter, case study design is described. Then, the two case studies that were carried out to develop the approach and explore the benefits that would be gained by

using the approach are introduced. The results of the case studies and the threats to the validity of the case studies are discussed.

4.1 Case Study Design

Two case studies were envisaged to be performed to explore answers for the research questions.

First research question required an approach to be developed by utilizing multiple review mechanisms. At least a small set of business processes was required for developing the approach. Since the approach was devised to bound two different disciplines, in different phases of the development, reviews by subject matter experts and software engineers were needed.

Second research question required a case study performed on a wider set of business processes. The developed approach needed to be performed on this wider set and effort records needed to be kept. The resulting effort would be compared with the estimated effort for requirements analysis based on industry data.

These required us to perform two discrete case studies, each aims to answer one of the two research questions.

Case selection criteria were formed to select the case studies where the needs for the research environment required to look for answers to our research questions are specified. Then, the background information of the selected organization, where the case studies were conducted, are introduced.

4.1.1 Case Selection Criteria

There were three selection criteria identified for case study design, in order to select the cases that are effective in exploring answers to the research questions. These criteria are considered in case selection of both case studies 1 and 2.

First selection criterion is that the organization under study should require business processes models that define its processes. This requirement is needed to be supported by the organization management that management support is crucial for

conducting such a study. Only then, the allocation of necessary resources for business process modeling activities would be ensured. These resources need to be allocated throughout the case study since, without time and effort spent by the organization itself; business process modeling would lose its focus on serving the business needs. Management support is also invaluable for the motivation of the allocated personnel.

Second criterion for the case to be selected is having complex business processes. Business processes having loops, conditionals, inputs and outputs, and hierarchical and cross-referencing relations between business processes make defining the business processes a necessity for the selected organization. Presences of business process guidelines that define constraints and legislative rules solely increase process complexity. Also necessity of identifying inconsistencies between several guidelines constitutes a motivation for business process modeling studies from the organization's point of view. Having to define complex business processes makes the organization allocate resources for the studies instead of perceive the studies as a burden. Allocating adequate resources is critical since, business process modeling activities are considerably dependent on subject matter experts' contribution besides business process modeling experts'. The success of the study resides in cooperation of subject matter experts and business process modeling experts since; only then the business process knowledge and technical expertise come together and result in a complete and correct set of business process models.

Third criterion is to have a case that requires software requirements to be elicited. According to this consideration; the case study should be conducted on an organization that is in need of integrating its business processes with information systems. Requirements analysis in parallel to business process modeling activities would then be a necessity for the organization to conduct successful acquisition and integration of information systems.

4.1.2 Background

The case studies were implemented as a part of a project in a recently established governmental organization specialized in regional development. Two other

governmental organizations are stakeholders of the project with the cooperation and coordination responsibilities. The project was launched with the aim of defining the business processes of the organization under study and specifying the software requirements to support these business processes. The outputs of the project are intended to be used by 26 organizations and by 962 personnel. The project was scheduled for a one year period.

One of the main practices of the organization is to develop and conduct regional grant programs. Besides the processes of grant programs management, a wide variety of process sets such as managing human resources, developing strategic plans, conducting performance management, investment supporting and conducting budget and accounting, archive and document management are included in the scope of the project. The common properties of these process sets are;

- Processes are complex with loops, conditionals, constraints, hierarchies and cross-references between processes.
- Few of them are applied yet by the organization that makes the processes to be well defined to enhance repeatability and detect defects in process flows.
- They are constrained by several guidelines that make tracing the conflicts between these guidelines difficult to determine and solve.

4.2 Case Study 1

This case study is performed to explore answers for the first research question. The Proceso approach described in Chapter 3 was developed in case study 1.

4.2.1 Case Study Plan

The case study plan detailing the activities to be conducted in case study 1 was developed. The activities in the case study plan are as follows;

- Select a small set of processes.
- Develop business process models for the selected set.

- Tailor the business process models to derive software requirements from these models.
- Write the functional software requirements manually for the selected process set and review them.
- Evaluate the business process model elements and links between them for fitness to including structural, language and data needs of software requirements.
- Update the business process model elements and links between them for fulfilling the need of attaining software requirements.
- Define the approach for generating requirements specification documents from business process models.
- Write functional software requirements manually by using the business process models and review.
- Develop tool support to generate software specifications and specification documents by utilizing business process models.

4.2.2 Case Study Implementation

The details regarding the development of Proceso approach in case study 1 are as follows.

Available business process modeling notations and tools were investigated to determine the business process models to be used in the case study. The need of the case study was a modeling approach that is suitable for performing business process modeling activities with the participation of subject matter experts who lack expertise in process modeling studies. As a result, extended event-driven process chain (eEPC) was chosen as the main business process modeling notation, since eEPC is one of the notations suitable for business process modeling 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).

A small set of business processes, involving 11 business processes, were determined and eEPC models for these processes were formed. The granularity and organization of the eEPC models were determined by considering the guidelines from the literature (Mendling et al. 2010) and the experiences of the team of business process modeling experts. There are at most fifteen functions and fifty objects in a process; whereas the average number of function objects in a process are six. On the other hand, the depth of the process models hierarchy is kept no more than five. Besides the main model elements that are functions, events and logical operators; position and organizational unit, process interface, information carriers and business rules were determined to be utilized in eEPC models. A sample eEPC model developed in the case study is provided in Appendix A.

After developing the business process models mainly by means of eEPC models, next task was to distinguish the activities in the business process models that are intended to be automated by information systems support. After the activities to be automated were discovered, the information needed for software requirements were to be embedded to business process models.

For discovering the aspects necessary to take part in the approach, some of the most commonly used software specification techniques in literature were investigated. Reviewed approaches were natural language specifications (Saeki et al. 1989), use case models (Jacobson 2004, Jacobson & Ng 2004), use case specifications (Ahour et al. 1999), formal software specifications (van Lamsweerde 2000, Spivey 1990) and data flow diagrams (Schach 1995).

The need was a unified implementation approach that supports business process modeling and requirements analysis. Considering the eEPC models and the software specification techniques above, the following points were taken into consideration to come up with such an approach.

- Processes in eEPC models can be considered as features of the system and in some instances as use cases.

- 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 input 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 carried 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. 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.

These considerations above are taken as requirements and the approach is characterized in respect to these requirements.

An approach based on deriving requirements directly from business process models was concluded not to be able to cover all the points above and to lack in keeping business process models lean without overloading them with software requirements related details. In order to satisfy the need of the approach to be easily understood and used by process owners, it was decided that the approach would be a model based approach. 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 the

case study there is a project specific advantage that business process modeling efforts are spent with the contribution of the business people. So, a model based approach would fit these efforts by connecting strongly to the business process models and making requirements analysis an integrated part of business process modeling studies. FADs, which are represented hierarchically under eEPC models in ARIS methodology (Davis & Brabander 2007), were determined to be the most appropriate candidate for fulfilling these purposes by extending the business process modeling approach to cover software requirements information.

Natural language was chosen to specify the software requirements, since natural language is a requirement specification style that the subject matter experts were familiar with. Also requirement specifications were to be delivered to a contractor developer. So, natural language was also appropriate, since it is the most frequently used specification style in the industry.

The next task was to determine the way to transform the software requirements related information in the business process models to natural language requirements. For preventing from being biased against the validity of the natural language specifications generated from business process models, natural language software specification sentences were written manually to explore how they would be like if the business process models and their objects were not used. The resulting specifications are provided in Appendix B. These sentences were reviewed by two peer software engineers. Following this review, three improvement points were detected and added to the business process modeling notation. These points are explained below.

- Business rule object was added to 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 were determined for fitness to natural language sentences in Turkish.

Utilization of Function Allocation Diagrams was adapted to the needs of the case study. FADs were characterized with function, position, entity type, application system type and business rule model elements and connection types between them. FADs were formed for the selected process set for which the eEPC models were formed before. The FADs for the eEPC model in Appendix A are provided in Appendix C.

The next step was to develop the sentence and document structures to generate natural language specification from business process models. Sentence and document structures were developed as introduced in Section 3.3. The requirements specification document and the sentence structures were used in writing natural language specifications manually. These manually written software specifications were reviewed by two peer software engineers and the organizations involved in the case study. Positive feedbacks were received from both parties and the sentence and document structures were validated.

Following the validation, a requirements generation tool was developed to automate the generation of natural language specifications. The tool generates the natural language software specifications in predetermined sentence and document structures. Samples of the specifications generated by the tool for the Function Allocation Diagrams in Appendix C are provided in Appendix D. The tool takes database objects from the ARIS tool and creates natural language specifications in an MS Word document. Validation of the outputs of the tool was done by comparing them with manually written natural language specifications, where the expected output generated by the tool was required to be exactly the same with the manually written specifications.

4.2.3 Results

A small set of business processes containing 11 business processes was utilized in the case study. As the approach is developed, the outputs were produced by using this process set. Multiple review mechanisms were applied throughout the development of the approach.

eEPC models and FADs were used in business process models. Software requirements related information was encapsulated in FADs, leaving the eEPC models simplified. Restricted model element sets were determined for both eEPC models and FADs. Model elements in FADs were referred from some of the most commonly used requirements engineering techniques.

Sentence and document structures were formed to enable derivation of natural language specifications from business process models. These structures were utilized by the requirements generation tool that was developed to automatically generate natural language specifications, which the subject matter experts are familiar with and the contractor software developers might use.

Forming eEPC models and FADs were referred as business process modeling, while forming the FADs and natural language specifications was referred as software requirements analysis. The approach, which was named later on as Procedo approach, enabled performing business process modeling and requirements analysis concurrently.

4.2.4 Threats to Validity

There is one threat to validity of the case study 1. The threat is that the validity regarding the development of the Procedo approach in case study 1 would have been biased. The approach was developed with collaboration of all the stakeholders of the case study. This threat is eliminated by adapting the approach from some of the most commonly used requirements engineering techniques and reviews by peer software engineers.

4.3 Case Study 2

Procedo is performed for a large set of business processes where the benefits are observed and analyzed for seeking answers to the second research question.

4.3.1 Case Study Plan

Work plan for case study 2 was developed as specified below.

- Divide the work into process modules.
- Determine roles and responsibilities in the case study.
- Perform workshops for forming eEPC models with the participation of subject matter experts and business process modeling experts.
- Review eEPC models and rework them based on review results.
- Perform requirements analysis by forming FADs and natural language specifications.
- Review FADs and natural language specifications and rework them based on review findings.
- Deliver the outputs of the process module under study and apply acceptance procedures.

The tasks for the succeeding process modules would be repeated until all deliverables are developed, delivered and accepted.

4.3.2 Case Study Implementation

Core and supporting sets of business processes of the organization were clustered around eight process modules. The modules were composed of business processes that cluster around the scope of the module. Although there were relationships between the process modules, they were identified to include business processes that have strong relationships between each other.

Six subject matter experts from the sponsor organizations and an external team of three business process modeling experts were determined to participate in the case study. Subject matter experts that take part in the case study were selected among personnel of the three participating organizations. The subject matter experts are

profound in business processes and authorized in process improvement. A project coordinator, who is responsible to coordinate activities and develop communication channels between business process modeling experts and subject matter experts, was determined among subject matter experts. Business process modeling experts, on the other hand, are contractor firms' software engineers who are specialized in business process modeling studies. Team of business process modeling experts includes three experts, one of whom is the project coordinator of the team of business process modeling experts' side. Contact points were established to maintain communication between these roles. An Internet forum was also reserved to enable accessing the outputs and information shared between the stakeholders.

Legislative documents together with subject matter expertise are primary inputs for business process models. Prior to the start of work on process modules; samples of inputs and outputs of the processes and documents that define business rules were delivered to business process modeling experts. In this way, it was ensured that the legislative documents were utilized in business process models. Prior to the start of the workshops, a training session was reserved to train subject matter experts in business process modeling approaches and concepts.

In workshops, the business process models were constructed with the cooperation of subject matter experts and business process modeling experts. Therefore, the process knowledge and technical competence came together to define a set of business process models that are complete and correct. For evaluating the results of the case study, records were kept for efforts spent, decisions made, issues and improvement opportunities detected throughout the modeling activities.

Besides eEPC models, function trees were also used in the case study indicating that there is no activity flow relation between the functions of a process. An organization chart was maintained in business process modeling studies showing all organizational objects in processes and the relations between them. A data dictionary was composed that defines the roles, inputs, outputs and application systems that are referred by the business process models.

Peer review was applied for process models by external business process modeling experts. Review sessions were conducted by subject matter experts with a systematic walkthrough method on each business process model. Review results were documented and models were updated and approved respectively before finalizing the products.

After the eEPC models were formed and reviewed, FADs were developed to perform requirements analysis. FADs were formed by the team of business process modeling experts. Then, the natural language specifications were generated from business process models with tool support. Both FADs and natural language specifications were reviewed by subject matter experts in workshops. Also peer reviews were conducted by software engineers. The findings of the reviews were documented and updates to the products were planned and applied.

As FADs and eEPC models were formed and approved, business process models for the process module under study were finalized.

Deliverables of each process module in the case study were business process models, software requirements specifications documents, data dictionary, workshop records and progress reports.

The tasks have been continued to be iterated for each module. Until now, deliverables of six process modules have been delivered and accepted. Progress reports were prepared, the outputs of the process modules were delivered to customer and predetermined acceptance procedures were carried out.

4.3.3 Results

Business process modeling activities for the process modules have been conducted according to the Procedeo approach defined in case study 1. In case study 2, for the first two modules, 946 business process models, 791 of which are FADs, were delivered.

Natural language software specifications were generated from business process models automatically via the tool support. In the requirements specification

documents delivered to sponsor organization, there are 1002 natural language specifications. The natural language specifications documents were generated with requirements generation tool for each module in approximately 15 seconds.

The deliverables were reviewed and accepted by the sponsor organization. Walkthroughs for business process models and reviews for software specifications were conducted. 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 that can be referred to as requirements analysis 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 for the IS that is intended to provide support for the first two modules was calculated as 11000 Cosmic Function Points according to another study (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, as the estimated effort for requirements specification was 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 the realized effort of 3000 person-hours in the case study.

Considering these values, it is concluded that with spending the sole effort of performing requirements analysis, performing both requirements analysis and business process modeling activities were managed to be completed. On the whole,

the total effort of business process modeling and requirements analysis activities was managed to be decreased.

Besides increasing the efficiency of requirements analysis in terms of total effort, many other benefits were observed by utilizing Proceso approach. Business process models formed an intuitive environment to discuss, gather and analyze 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 business process modeling forces developers to define the system in a structured way. The hierarchical structure of process models enables definition of modular processes and reveals relations between those processes in different levels. By all these means, the business processes can be explained in a more unambiguous and consistent way compared to a natural language explanation. As a result, unambiguous and consistent requirements documents are formed as much as the process models are. 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.

4.3.4 Threats to Validity

There are two threats to the validity of the case study as identified.

First threat is that, the software specifications generated from business process models in case study 2 are not utilized in software development yet. However, they are validated by reviews of peer software engineers and subject matter experts and delivered to the contractor software developer which has not returned any negative feedback. For these reasons, this threat is not expected to bring any significant negative effect on validity.

The second threat is that the benefits obtained from the Proceso 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.

CHAPTER 5

CONCLUSIONS

This chapter presents main findings and contributions of the study and suggests future research directions.

5.1 Summary

In this study we have two main goals. The first goal is to explore the potential of business process models for software requirements definition and propose a systematic approach. The second goal was to determine if the proposed approach would significantly decrease the amount of total effort required for process modeling and requirements analysis. For these purposes, two case studies were performed, one for forming the approach and the other for investigating if the foreseen benefits, in terms of a reduction in total effort, would be achieved.

Procedo approach is formed based on conventional business process modeling notations, and included a process description and tool support.

eEPC models and FADs are used in business process modeling notation. A restricted set of model elements is determined to be used in eEPC models and process hierarchy is described by the hierarchy between eEPC models. By utilizing FADs, eEPC models are extended. Using FADs prevented eEPC models to be overloaded with software related information. FADs are enriched with aspects such as CRUDL

based operations, system entities maintained in application systems, business rules and selective execution of operations.

The unified process for performing business process modeling and requirements analysis is defined with seven consecutive steps. These steps are detailed with activities, inputs and outputs within them. The process is described from context definition to business process modeling and to requirements analysis. Several review sessions are also defined within the unified process.

Generation of natural language specification documents from business process models are successfully achieved in Proceso. Automated derivation of software requirements statements and formation of a complete software requirements specification document are achieved by means of the requirements generation tool developed. The sentence and document structures formed for natural language specifications are utilized by the tool. The natural language specifications generated by the requirements generation tool denote software requirements with system functions, system entities, application systems, operations on system entities, roles that are authorized to perform operations and rules that constrain the systems. The generated specifications in natural language are appropriate to be used by both subject matter experts and software developers.

Proceso approach bridges the gap between business process modeling and requirements analysis. The approach was developed in case study 1 by utilizing a small process set for forming business process models, engineering them, writing requirements manually and developing the requirements generation tool. Multiple review sessions were utilized in development phase of the approach to validate it. The approach enabled us to transfer the business process knowledge to software requirements and generate the software requirements documents automatically. By using eEPC models and FADs in business process modeling and generating natural language specifications from them, the approach made it possible to perform business process modeling and software requirements analysis concurrently and to provide a one-way synchronization between business process models and software requirements. We accomplished to define the approach where the same business process models would be utilized for a variety of purposes such as business process

definition, reengineering and requirements definition. Proceso approach replaces, in part, the analysis phase of software development life cycle in a generic waterfall development model.

Case study 2 was applied in an organization whose business processes were modeled and software requirements specifications were delivered. 946 business process models including 791 FADs, requirements specification document with 1002 natural language specifications were formed spending 3000 man hours in total and for a system of 11000 function points. These outputs were validated through walkthroughs on business process models by the organization and reviews on requirements specifications by the organization and peer software engineers. The requirement specifications were also delivered to a contractor software developer.

A reduction in total effort realized for business process modeling and requirements analysis activities was observed by calculating the efforts spent in the studies and comparing them with the industry data considering the size of our system. It is seen that, even if it is assumed we have covered 40% of requirements engineering activities as a lowest estimate, the realized total effort of business process modeling and requirements analysis correspond to the same percentage of requirements engineering estimated of industry data. It was concluded that by utilizing the Proceso approach, total effort for business process modeling and requirements analysis would be significantly decreased.

Besides this, improvements in completeness, correctness, consistency, unambiguousness, traceability and maintainability properties of software requirements were observed. Omissions and duplications in requirements were prevented. Structured requirement sentences generated with tool support were unambiguous and consistent. Tool support also enabled ease of maintenance as updates in business process models were reflected to requirements specifications. Traceability between business processes and software requirements were made implicit.

5.2 Contributions

One major contribution of the study is to show that the total effort of business process modeling and requirements analysis was significantly decreased by performing them concurrently. Such an efficiency improvement was achieved by applying the Proceso approach that we have developed in this study. This indicates that if business process modeling and requirements analysis are performed by using Proceso as described in this study, it is possible to save from the total amount of required effort.

Another contribution is that Proceso defines a systematic process for business process based requirements specification supported by a requirements generation tool. None of the studies in the related research, except Su (2004) and Specht et al. (2005), use eEPC models as the main business process modeling notation. Also none, except Su (2004), provides tool support for requirements generation or describe deriving natural language specifications from business process models. In Su's (2004) study, however, the business process models are redesigned for requirements generation purpose only and are formed with color codes given to information carriers, model element naming conventions and activities where all of them are suggested to be automated. The business process models in this study are not only formed to be used for deriving requirements, but also to be used as business process definitions. Also FADs are utilized in Proceso so that software related information is separated from eEPC models by keeping them lean, whereas Su (2004) overloads eEPC models with such information and so decreases the readability of the models. Compared with the natural language specifications generated in Su (2004), the generated documents in Proceso include structured and complete natural language specifications with system entities, business rules and standard and selective execution of system operations. Specht et al. (2005) does not provide derivation of natural language specifications and tool support for automatically generating software requirements but, it is the only study detected in the literature to use FADs for deriving software requirements. However, FADs in Proceso are enriched with business rules, CRUDL based standard operations, selective execution of operations

and different model element connection rules when compared with Specht et al. (2005).

5.3 Future Study

One direction of the future studies is envisaged as applying the proposed approach in multiple case studies and using 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 business process modeling field. By performing these case studies it is also envisaged to validate the improvements in quality attributes of software requirement statements achieved via Procedo.

A limitation of Procedo is that the natural language specifications generated within the approach do not include system attributes, action sequences, system states and logical separations and connections. FADs would be extended with such software requirements related aspects in order to cover a wider range of requirements engineering activities.

By using FADs in Procedo, 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 software engineering activities. Automated size estimation for the development of information systems is already explored by utilizing the FADs (Kaya 2010). It is possible to investigate the opportunities within requirements engineering research area for derivation of use case specifications and formal specifications. Deriving class diagrams, natural language test case specifications and code skeletons from business process models is also a future research opportunity. These infer us that the scope of the research is possible to be expanded to software development life cycle by attaining forward traceability from software requirements artifacts to software design, testing and coding artifacts.

In Proceso, requirements change management is kept out of scope. Forming a change management process for a situation where the software requirements change would be a future research direction.

Process improvement is another area that has not been mentioned explicitly in this study. However, by resolving conflicts between process guidelines and suggesting information system support to the activities, Proceso is related to process improvement. Still, generating quality manuals from business process models to be used in process improvement might be a future 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.

Achour, C.B., Rolland, C., Maiden, N.A. and Souveyet, C. (1999). Guiding Use Case Authoring: Results from an Empirical Study. In: Fourth IEEE International Symposium on Requirements Engineering (RE'99), pp. 36-43.

Athanasakis, N. (2006). Generating Natural Language from UML Class Diagrams. MS Thesis, Informatics Research Institute, University of Salford, Salford, UK.

Berenbach, B., Paulish, D., Kazmeier, J., and Rudorfer, A. (2009). Software & Systems Requirements Engineering: in Practice. 1st ed. McGraw-Hill, Inc.

Cabral, G. and Sampaio, A. (2008). Formal Specification Generation from Requirement Documents. Electron. Notes Theor. Comput. Sci. vol. 195, pp. 171-188.

Cox, K., Phalp, K.T., Bleistein, S.J., and Verner, J.M. (2005). Deriving requirements from process models via the problem frames approach. Inf. Softw. Technol. 47, 5, pp. 319-337.

Danlos, L., Lapalme, G., and Lux, V. (2000). Generating a Controlled Language. In Proceedings of the First international Conference on Natural Language Generation, vol. 14, pp. 141-147.

Davis, R., and Brabander, E. (2007). ARIS Design Platform: Getting Started with BPM, 1st ed. Springer.

Dehnert, J. and Rittgen, P. (2001). Relaxed Soundness of Business Processes. In Proceedings of the 13th international Conference on Advanced information Systems Engineering. K. R. Dittrich, A. Geppert, and M. C. Norrie, Eds. Lecture Notes In Computer Science, vol. 2068, Springer-Verlag, London, pp. 157-170.

Estrada, H., Martínez, A. and Pastor, A. (2003). Goal-based business modeling oriented towards late requirements generation. Proceedings of the 22nd International Conference on Conceptual Modelling, pp. 277-290.

Havey, M. (2005). Essential Business Process Modeling. O'Reilly Media, Inc.

IEEE std. 610.12-1990 (1990). Standard Glossary of Software Engineering Terminology, IEEE CS Press.

Jacobson, I. (2004). Use cases – yesterday, today, and tomorrow. Software and Systems Modeling, vol. 3, no. 3, pp. 210-220.

Jacobson, I. and Ng, P. (2004). Aspect-Oriented Software Development with Use Cases, Addison-Wesley Professional.

Jones, T.C. (1998). Estimating Software Costs. McGraw-Hill, Inc.

Jungmayr, S. and Stumpe, J. (1998). Another Motivation for Usage Models: Generation of User Documentation. Proceedings of CONQUEST '98.

Kamsties, E. (2005). Understanding ambiguity in requirements engineering. In A. Aurums and C. Wohlin, editors, Engineering and Managing Software Requirements, Springer-Verlag, pp. 245–266.

Kaya, M. (2010). E-Cosmic: A Business Process Model Based Functional Size Estimation Approach. MS Thesis, Informatics Institute, METU, Ankara, Turkey.

Lee, B. and Bryant, B.R. (2002). Automated conversion from requirements documentation to an object-oriented formal specification language. In Proceedings of the 2002 ACM Symposium on Applied Computing, SAC '02. ACM, New York, NY, pp. 932-936.

List, B. and Korherr, B. 2006. An evaluation of conceptual business process modelling languages. In Proceedings of the 2006 ACM Symposium on Applied Computing, ACM, New York, NY, pp. 1532-1539.

Maiden, N.A., Minocha, S., Manning, K., and Ryan, M. (1998). CREWS-SAVRE: Systematic Scenario Generation and Use. In Proceedings of the 3rd international Conference on Requirements Engineering: Putting Requirements Engineering To Practice, pp. 148-155.

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. (2010). Seven process modeling guidelines (7pmg). Information and Software Technology, vol. 52, no. 2, pp. 127–136.

Meziane, F., Athanasakis, N. and Ananiadou, S. (2008). Generating Natural Language specifications from UML class diagrams. *Requir. Eng.* Vol. 13, no. 1, pp. 1-18.

Minoli, D. (2008). Enterprise Architecture A to Z: frameworks, business process modeling, SOA, infrastructure technology. Boca Raton:, CRC Press.

Nicolas, J. and Toval, A. (2009). On the Generation of Requirements Specifications from Software Engineering Models: A Systematic Literature Review. *Information and Software Technology*, vol. 51, pp. 1291-1307.

och Dag, J.N. and Gervasi, V. (2005). Managing Large Repositories of Natural Language Requirements. In A. Aurums and C. Wohlin, editors, *Engineering and Managing Software Requirements*, Springer-Verlag, pp. 219-244.

Recker, J., Rosemann, M., Indulska, M. and Green, P. (2009). Business Process Modeling - A Comparative Analysis. *Journal of the Association for Information Systems*, vol. 10, no. 4, pp. 333–363.

Roser, S. and Bauer, B. (2005). A Categorization of Collaborative Business Process Modeling Techniques. In *Proceedings of the Seventh IEEE international Conference on E-Commerce Technology Workshops*, pp. 43-54.

Saeki, M., Horai, H., and Enomoto, H. (1989). Software Development Process from Natural Language Specification. *11th International Conference on Software Engineering*.

Santander, V.F. and Castro, J. (2002). Deriving Use Cases from Organizational Modeling. In *Proceedings of the 10th Anniversary IEEE Joint international Conference on Requirements Engineering*, pp. 32-42.

Schach, S. R. (1995). *Classical and Object-Oriented Software Engineering*. 3rd ed., McGraw-Hill Professional.

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.

Scheer A.W., Kruppke, H., Jost, W. and Kindermann, H. (2006). *Agility by ARIS Business Process Management: Yearbook Business Process Excellence 2006/2007*. Springer-Verlag New York, Inc.

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.

Spivey, J.M. (1990). An introduction to Z and Formal Specifications. *Software Engineering Journal*, vol. 4, no. 1, pp. 40–50.

Stolfa, S. and Vondrak, I. (2004). A Description of Business Process Modeling as a Tool for Definition of Requirements Specification. *Systems Integration 12th Annual International Conference*, pp. 463-469.

Su, O. (2004). Business Process Modeling Based Computer-Aided Software Functional Requirements Generation. MS Thesis, Informatics Institute, METU, Ankara, Turkey.

Tarhan, A., Gencil, 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*, vol. 41, no. 10, pp. 639-650.

van der Aalst, W. M., Hofstede, A. H. and Weske, M. (2003). Business process management: a survey. In *Proceedings of the 2003 international Conference on Business Process Management*. W. M. Van Der Aalst, A. T. Hofstede, and M. Weske, Eds. *Lecture Notes In Computer Science*. Springer-Verlag, Berlin, Heidelberg, pp. 1-12.

van Lamsweerde, A. (2000). Formal specification: a roadmap. In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*, pp. 147-159.

Westland, J.C. (2002). The Cost of Errors in Software Development: Evidence from Industry. *The Journal of Systems and Software*, 62, pp. 1-9.

Wiegers, K.E. (2005). *More about Software Requirements: Thorny Issues and Practical Advice*. Microsoft Press.

Workflow Management Coalition (1999). Workflow Management Coalition Terminology & Glossary (Document No. WFMC-TC-1011). Workflow Management Coalition Specification.

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.

APPENDICES

APPENDIX A: A sample business process model developed in the case study

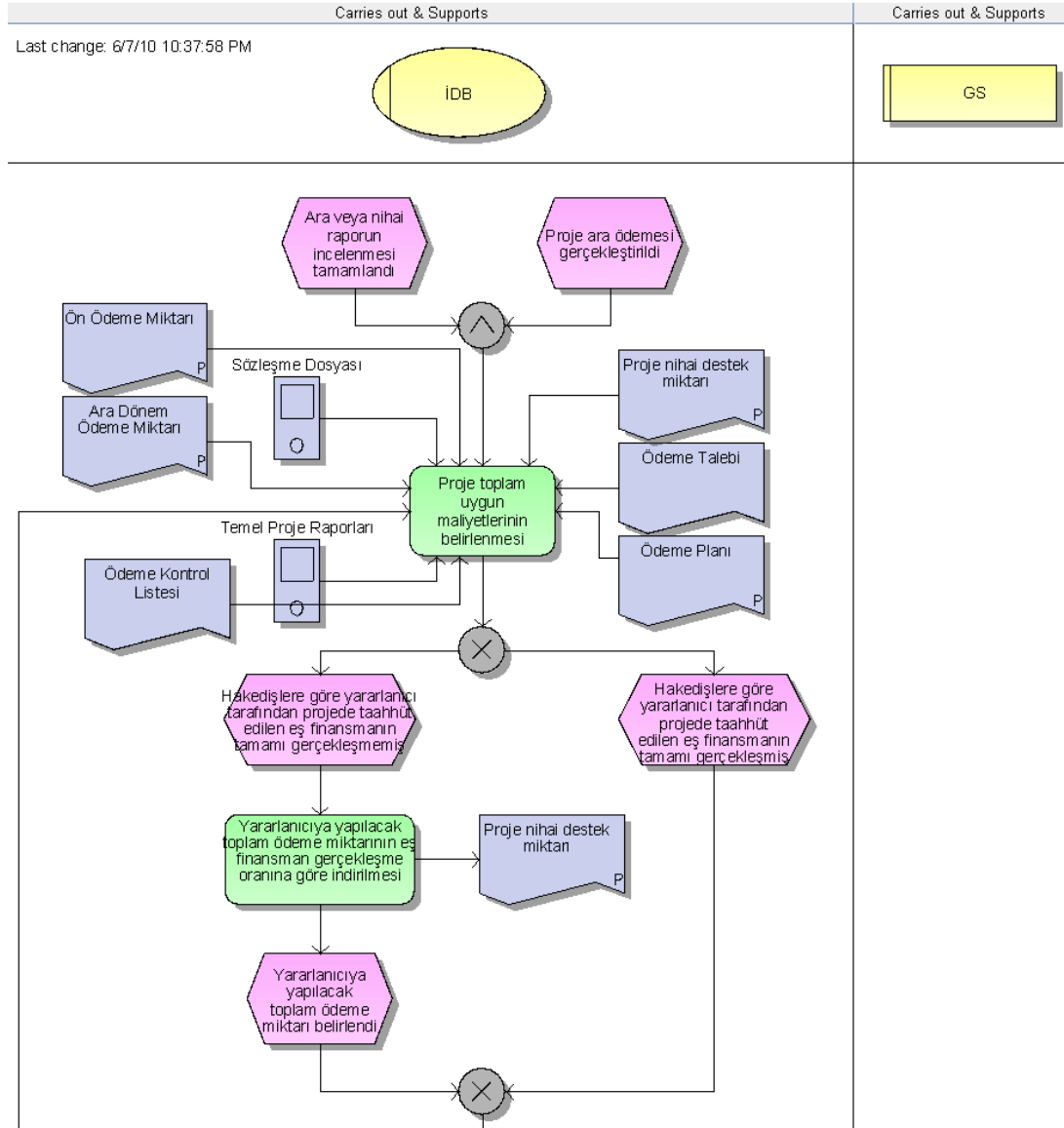


Figure 11: eEPC model of “Nihai Ödeme” process

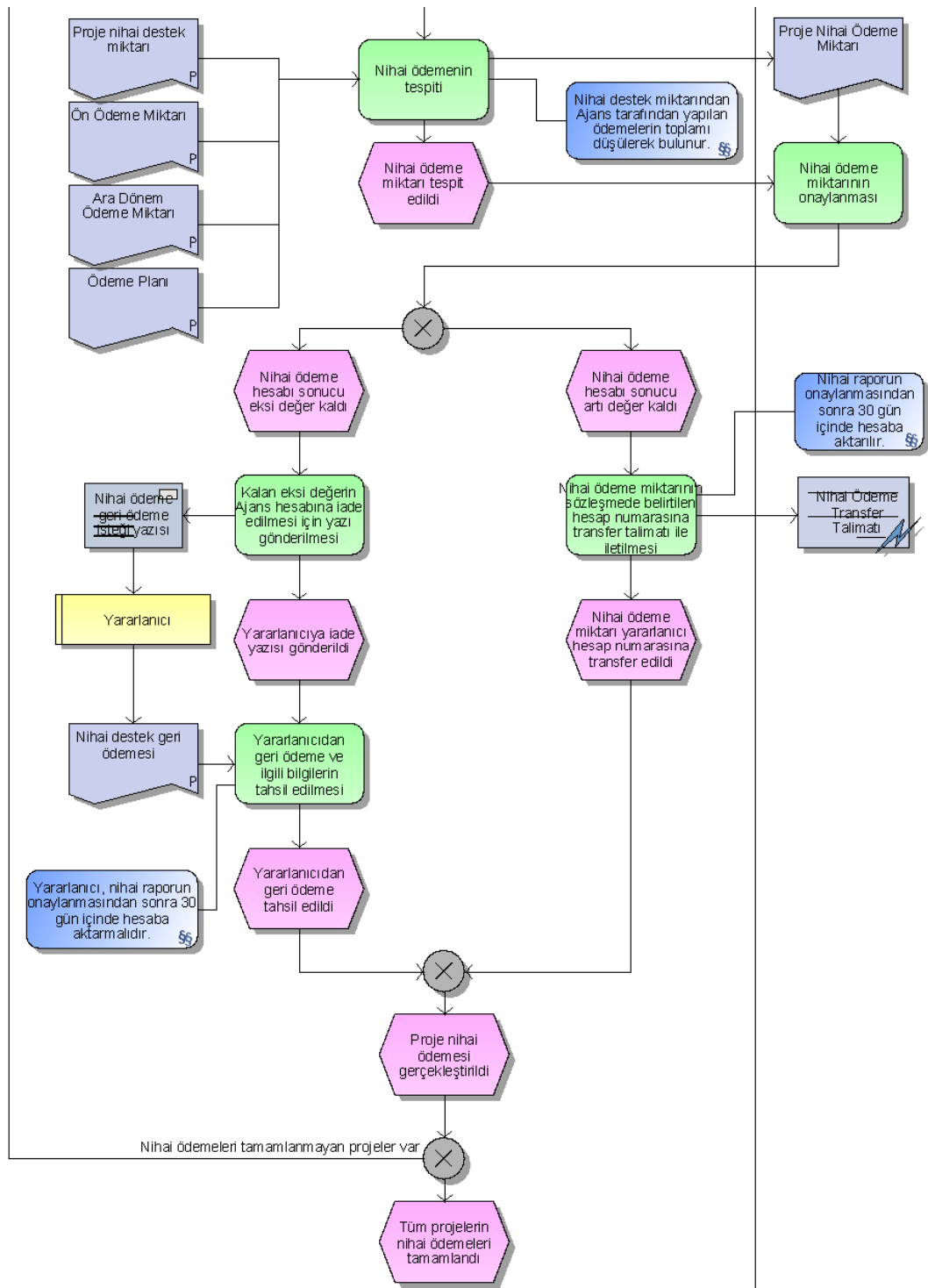


Figure 11: (continued)

APPENDIX B: Manually written requirements for a selected business process

- Nihai Ödeme
- PFDS sistemi, proje toplam uygun maliyetlerinin belirlenmesi sırasında İDB Uzmanı tarafından Ön Ödeme Miktarı, Ara Dönem Ödeme Miktarı, Proje Uygun Maliyetleri, Temel Proje Raporları, Sözleşme Dosyası ve Ödeme Kontrol Listesi'nin okunmasına ve Proje Nihai Destek Miktarı'nın yaratılmasına olanak sağlamalıdır.
- PFDS sistemi, yararlanıcıya yapılacak toplam ödeme miktarının eş finansman gerçekleşme oranına göre indirilmesi sırasında İDB Uzmanı tarafından Başvuru Rehberi'nin okunmasına ve Proje Nihai Destek Miktarı'nın değiştirilmesine olanak sağlamalıdır.
- PFDS sistemi, nihai ödemenin tespiti sırasında İDB Uzmanı tarafından Proje Nihai Destek Miktarı, Ön Ödeme Miktarı ve Ara Dönem Ödeme Miktarı'nın okunmasına, Ödeme Planı'nın değiştirilmesine ve Proje Nihai Ödeme Miktarı ve Onay İsteği'nin yaratılmasına olanak sağlamalıdır.
- PFDS sistemi, nihai ödeme miktarının onaylanması sırasında GS tarafından Ödeme Planı, Proje Nihai Ödeme Miktarı ve Onay İsteği'nin okunmasına ve Onay Durumu'nun yaratılmasına olanak sağlamalıdır.
- PFDS sistemi, nihai ödeme miktarının sözleşmede belirtilen hesap numarasına transfer talimatı ile iletilmesi sırasında İDB Uzmanı tarafından Proje Nihai Ödeme Miktarı'nın okunmasına olanak sağlamalıdır.
- PFDS sistemi, kalan eksi değerın Ajans hesabına iade edilmesi için yazı gönderilmesi sırasında İDB Uzmanı tarafından Proje Nihai Ödeme Miktarı'nın okunmasına olanak sağlamalıdır.
- PFDS sistemi, Yararlanıcıdan geri ödeme ve ilgili bilgilerin tahsil edilmesi sırasında İDB Uzmanı tarafından Proje Nihai Ödeme Miktarı'nın okunmasına olanak sağlamalıdır.

- İş kuralları:
- Nihai ödemenin tespiti sırasında: Nihai destek miktarından Ajans tarafından yapılan ödemelerin toplamı düşülerek bulunur.
- Nihai ödeme miktarının sözleşmede belirtilen hesap numarasına transfer talimatı ile iletilmesi sırasında: Nihai raporun onaylanmasından sonra 30 gün içinde hesaba aktarılır.
- Yararlanıcıdan geri ödeme ve ilgili bilgilerin tahsil edilmesi sırasında: Yararlanıcı, nihai raporun onaylanmasından sonra 30 gün içinde hesaba aktarmalıdır.

APPENDIX C: Function Allocation Diagrams for a selected process

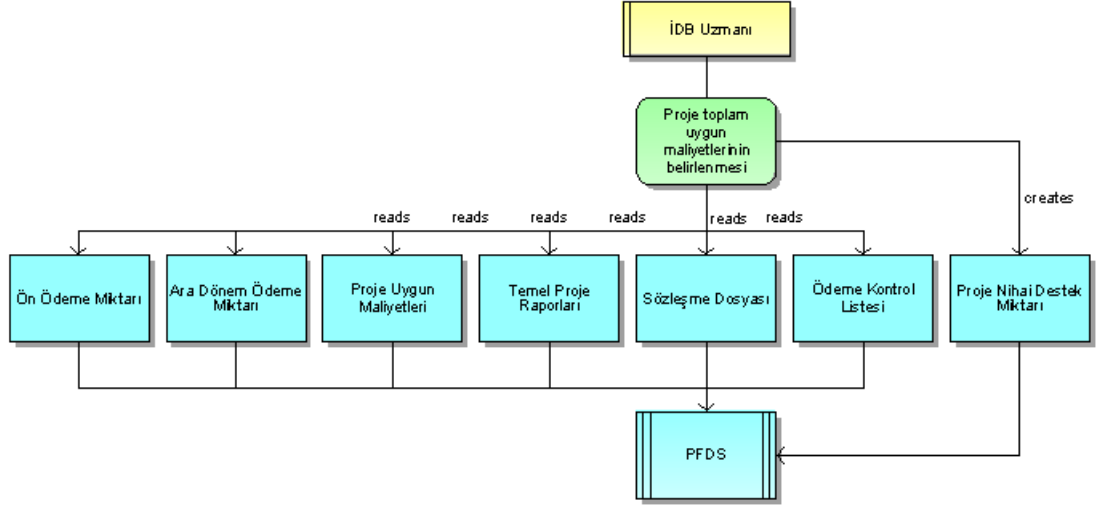


Figure 12: Function Allocation Diagram for “proje toplam uygun maliyetlerinin belirlenmesi” function in “Nihai Ödeme” process

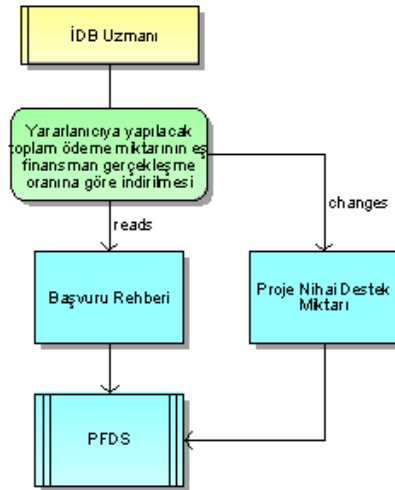


Figure 13: Function Allocation Diagram for “yararlanıcıya yapılacak toplam ödeme miktarının eş finansman gerçekleşme oranına göre indirilmesi” function in “Nihai Ödeme” process

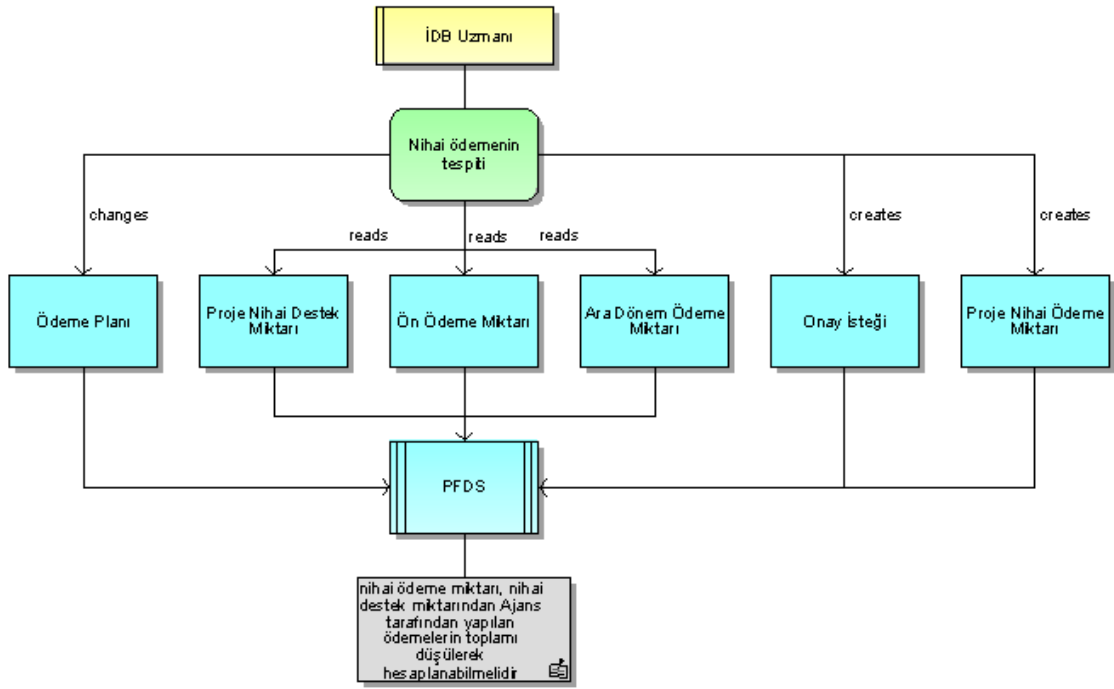


Figure 14: Function Allocation Diagram for “nihai ödemenin tespiti” function in “Nihai Ödeme” process

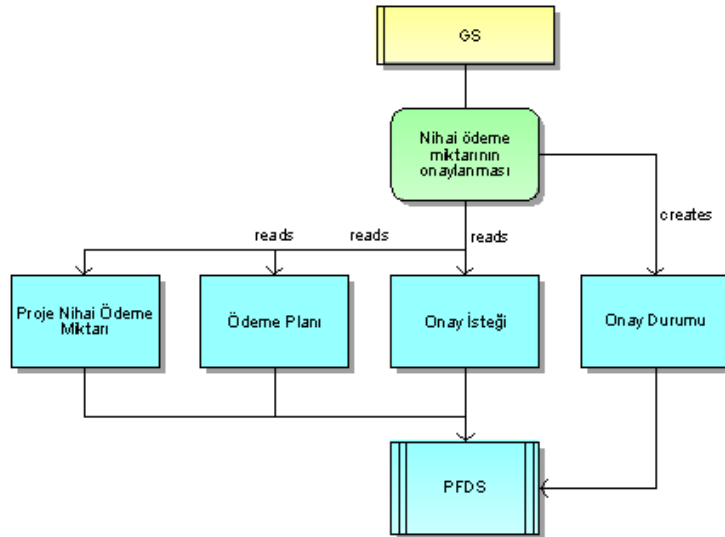


Figure 15: Function Allocation Diagram for “nihai ödeme miktarının onaylanması” function in “Nihai Ödeme” process

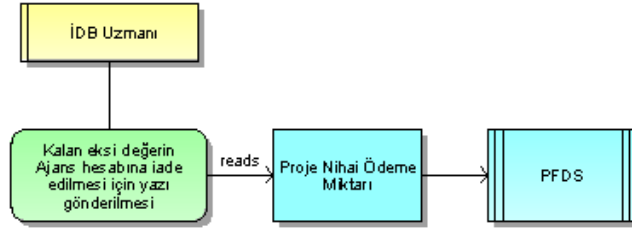


Figure 16: Function Allocation Diagram for “kalan eksi değerin Ajans hesabına iade edilmesi için yazı gönderilmesi” function in “Nihai Ödeme” process

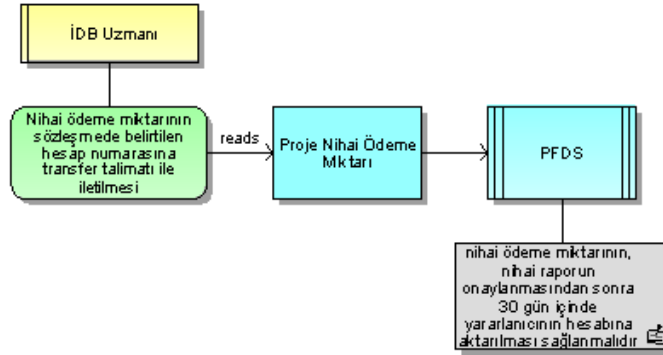


Figure 17: Function Allocation Diagram for “nihai ödeme miktarının sözleşmede belirtilen hesap numarasına transfer talimatı ile iletilmesi” function in “Nihai Ödeme” process

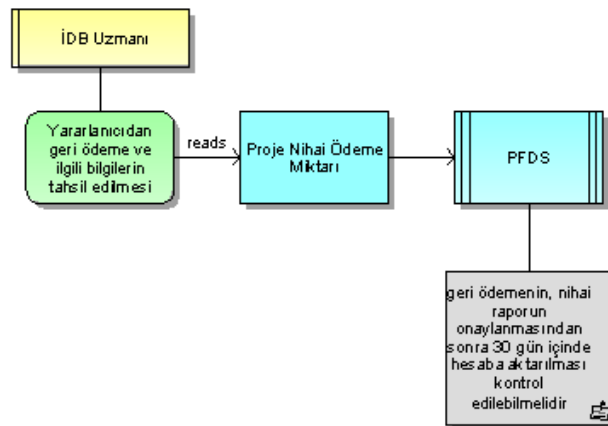


Figure 18: Function Allocation Diagram for “yararlanıcıdan geri ödeme ve ilgili bilgilerin tahsil edilmesi” function in “Nihai Ödeme” process

APPENDIX D: Natural language specifications generated by tool support for a selected business process

...

0. Süreç Adresi: KASA\01-PFDY

0.1. Süreç Adı: PFDY

1. Süreç Adresi: KASA\01-PFDY\01-PTÇ

1.1. Süreç Adı: Proje Teklif Çağrısı

...

69. Süreç Adresi: KASA\01-PFDY\01-PTÇ\05-PTÇ Uygulama\06-Ödemelerin Gerçekleştirilmesi\02-Ara Ödeme

69.1. Süreç Adı: Ara Ödeme

...

70. Süreç Adresi: KASA\01-PFDY\01-PTÇ\05-PTÇ Uygulama\06-Ödemelerin Gerçekleştirilmesi\03-Nihai Ödeme

70.1. Süreç Adı: Nihai Ödeme

70.1.1. PFDY.336:

Proje toplam uygun maliyetlerinin belirlenmesi sırasında, İDB Uzmanı tarafından PFDS sistemi üzerinde Ara Dönem Ödeme Miktarı, Proje Uygun Maliyetleri, Ön Ödeme Miktarı, Temel Proje Raporları, Sözleşme Dosyası, Ödeme Kontrol Listesi okunabilmeli, Proje Nihai Destek Miktarı yaratılabilmelidir.

70.1.2. PFDY.337:

Yararlanıcıya yapılacak toplam ödeme miktarının eş finansman gerçekleşme oranına göre indirilmesi sırasında, İDB Uzmanı tarafından PFDS sistemi üzerinde Proje Nihai Destek Miktarı değiştirilebilmeli, Başvuru Rehberi okunabilmelidir.

70.1.3. PFDY.338:

Nihai ödemenin tespiti sırasında, İDB Uzmanı tarafından PFDS sistemi üzerinde Ödeme Planı değiştirilebilmeli, Ön Ödeme Miktarı, Ara Dönem Ödeme Miktarı, Proje Nihai Destek Miktarı okunabilmeli, Proje Nihai Ödeme Miktarı, Onay İsteği yaratılabilmelidir.

• PFDS sistemi üzerinde; nihai ödeme miktarı, nihai destek miktarından Ajans tarafından yapılan ödemelerin toplamı düşülerek hesaplanabilmelidir.

70.1.4. PFDY.339:

Nihai ödeme miktarının onaylanması sırasında, GS tarafından PFDS sistemi üzerinde Ödeme Planı, Onay İsteği, Proje Nihai Ödeme Miktarı okunabilmeli, Onay Durumu yaratılabilmelidir.

70.1.5. PFDY.340:

Kalan eksi deęerin Ajans hesabına iade edilmesi için yazı gönderilmesi sırasında, İDB Uzmanı tarafından PFDS sistemi üzerinde Proje Nihai Ödeme Miktarı okunabilmelidir.

70.1.6. PFDY.341:

Nihai ödeme miktarının sözleşmede belirtilen hesap numarasına transfer talimatı ile iletilmesi sırasında, İDB Uzmanı tarafından PFDS sistemi üzerinde Proje Nihai Ödeme Miktarı okunabilmelidir.

- PFDS sistemi üzerinde; nihai ödeme miktarının, nihai raporun onaylanmasından sonra 30 gün içinde yararlanıcının hesabına aktarılması sağlanmalıdır.

70.1.7. PFDY.342:

Yararlanıcıdan geri ödeme ve ilgili bilgilerin tahsil edilmesi sırasında, İDB Uzmanı tarafından PFDS sistemi üzerinde Proje Nihai Ödeme Miktarı okunabilmelidir.

- PFDS sistemi üzerinde; geri ödemenin, nihai raporun onaylanmasından sonra 30 gün içinde hesaba aktarılması kontrol edilebilmelidir.

...

71. Süreç Adresi: KASA\01-PFDY\01-PTÇ\05-PTÇ Uygulama\07-Risk Deęerlendirmesi

71.1. Süreç Adı: Risk Deęerlendirmesi

...

APPENDIX E: User manual for the requirements generation tool

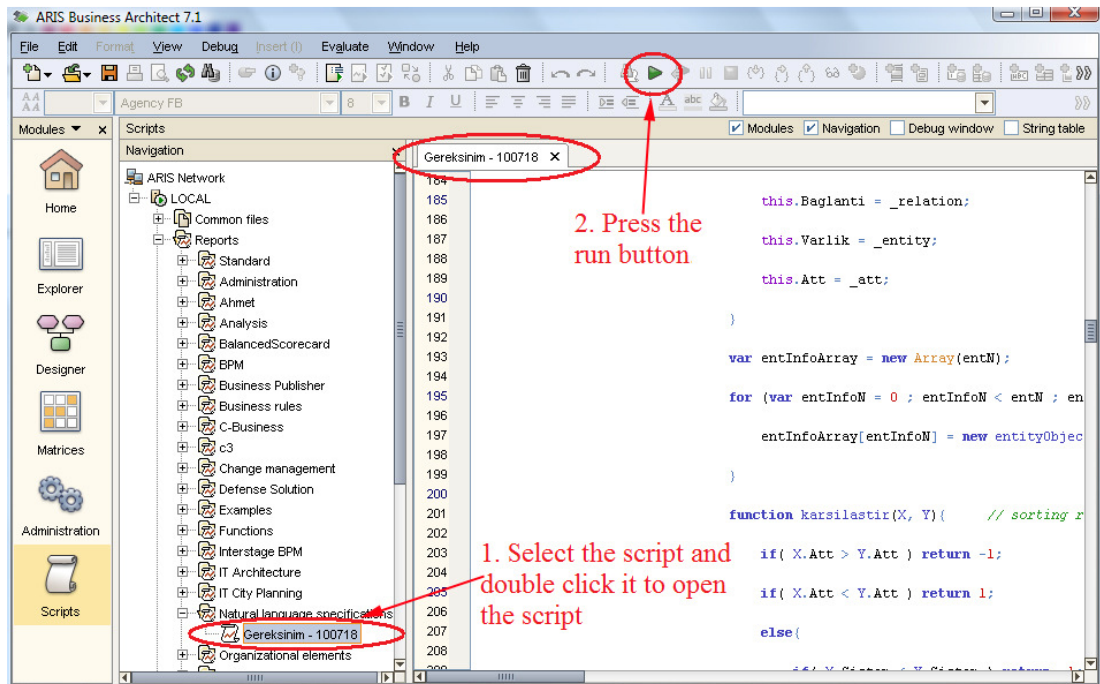


Figure 19: User manual for the requirements generation tool – Steps 1 and 2

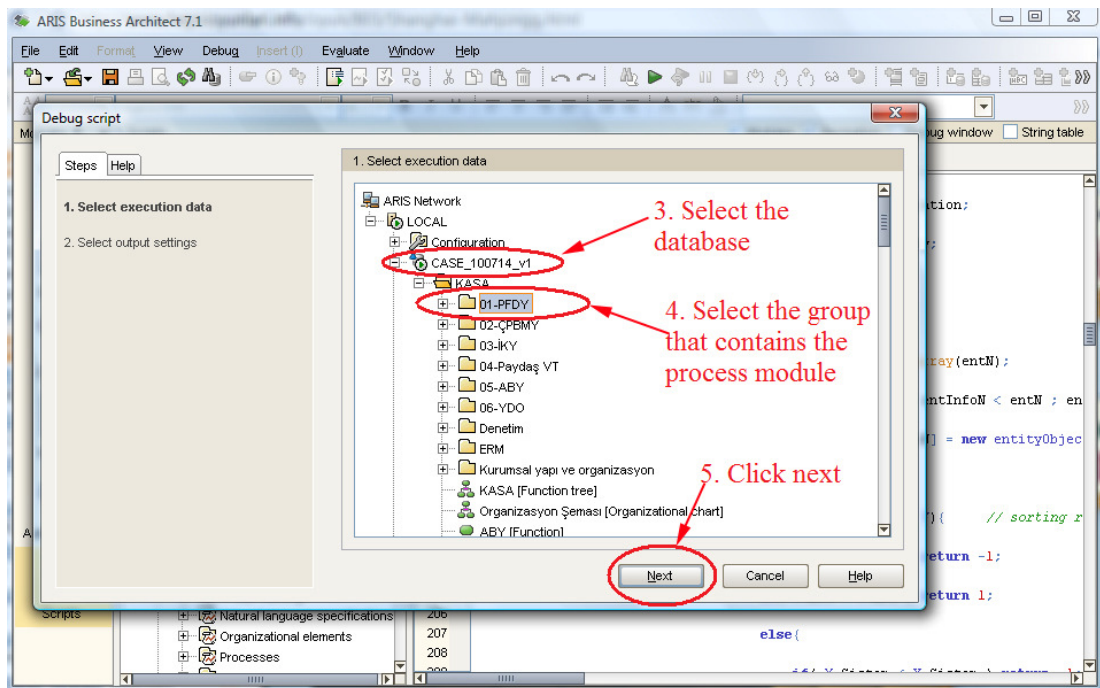


Figure 20: User manual for the requirements generation tool – Steps 3, 4 and 5

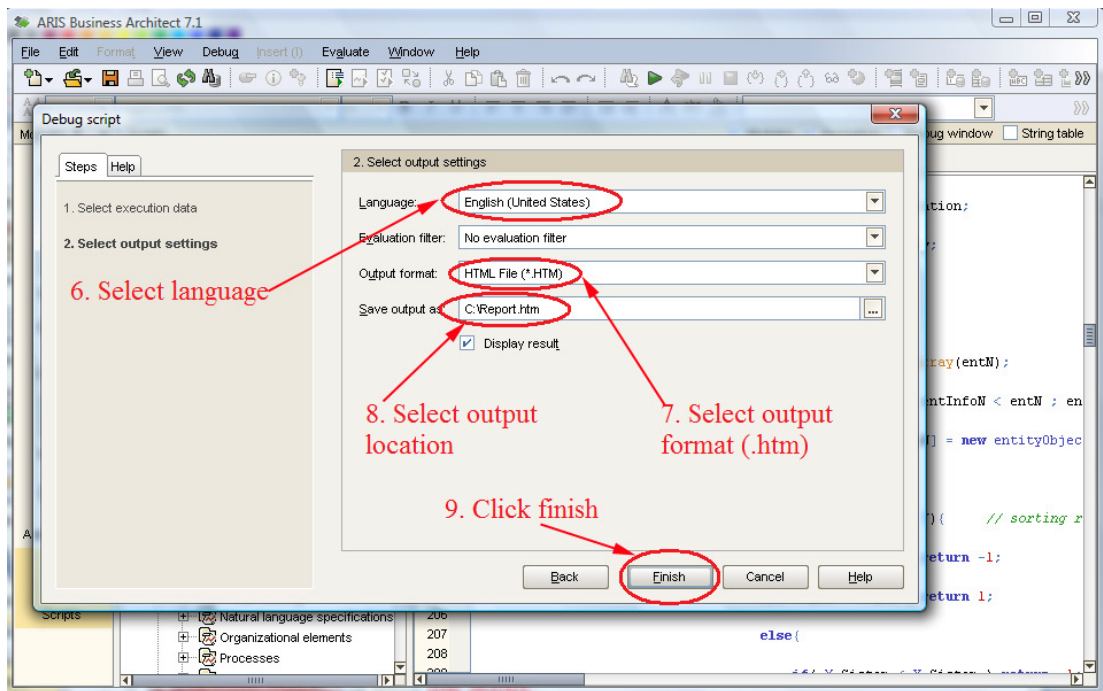


Figure 21: User manual for the requirements generation tool – Steps 6, 7, 8 and