UNIVERSITY OF TARTU

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Institute of Computer Science

Software Engineering

Sander Valvas

Requirements Elicitation from BPMN Models

Master's Thesis (30 EAP)

Supervisor: Fredrik P. Milani

TARTU 2015

Requirements Elicitation from BPMN Models

Abstract

When building a software system, it is crucial to understand the actual needs and the interfering constraints that apply in the surrounding environment. Elicitation of requirements is all about learning the environment and discovering the needs of users and other stakeholders. One of the primary sources for requirement elicitation is the system (processes, organization, environment and legacy systems) currently being used. The system is often captured in the form of graphical models, which are an important source of information for requirements elicitation. BPMN models are gaining popularity and are frequently used to model systems. Despite the fact that they are a valuable source of knowledge, they are rarely used as a source for eliciting requirements. One reason for this is the lack of concrete and comprehensive guidelines that would assist a systematic requirements elicitation from such models. This thesis presents a method for eliciting functional requirements from BPMN models. The method covers all components of a requirement and gives guidelines where in the BPMN model the information about the components can be found. It also provides a set of questions to be asked from domain experts to make sure that the requirement specification is complete, consistent, bounded and on the required level of granularity. The method was applied on a case study and it was proved that the method is applicable and provides a structured approach to eliciting requirements. The method elicited more requirements than the method previously used by the case organization, and the elicited requirements were also of better quality. The method took considerably less time to apply, it gave better control over the elicitation process, it was easier to evaluate the needed effort, and it enabled to better plan the process. The structured approach makes it easier to delegate work, and there are less situations where something might be overlooked.

Keywords: Requirements Elicitation, Requirements Discovery, Requirements Derivation, Business Process Modeling, BPMN.

Nõuete tuvastamine BPMN mudelitest

Lühikokkuvõte

Tarkvarasüsteemi loomiseks on väga oluline mõista, millised on tegelikud vajadused ja nende rahuldamist takistavad piirangud. Nõuete tuvastamise käigus õpitakse tundma ümbritsevat keskkonda ja tehakse kindlaks kasutajate ning teiste osapoolte vajadused. Üheks peamiseks kohaks, kust nõudeid leida, on hetkel kasutatavad süsteemid (protsessid, organisatsioon, keskkond ja kasutatavad infosüsteemid). Kasutusel olevaid protsesse kujutatakse tihti graafiliselt mudelitena ja need mudelid kujutavad endast väga olulist informatsiooniallikat nõuete tuvastamisel. BPMN mudelid on saanud väga populaarseks ja neid kasutatakse tihti süsteemide kirjeldamiseks, kuid vaatamata sellele, et nad on väärtuslikud teadmiste allikad, kasutatakse neid nõuete tuvastamisel siiski harva. Üheks selliseks põhjuseks on asjaolu, et puuduvad konkreetsed ja põhjalikud juhised, mis aitavad süstemaatiliselt mudelist nõudeid tuvastada. Selles töös esitletakse meetodit funktsionaalsete nõuete tuvastamiseks BPMN mudelitest. Meetod läbib süsteemselt kõiki nõude komponente ja annab juhised, kuidas BPMN mudelist komponendi kohta informatsiooni leida ning annab lisaks kogumi küsimusi, mida valdkonna spetsialistidele esitada, et nõue oleks põhjalik, järjepidev, piiritletud ja nõutava detailsusega. Loodud meetodit rakendati ka juhtumiuuringu käigus ja tõestati, et uus meetod on rakendatav ning on struktureeritud lähenemine nõuete tuvastamiseks. Meetod tuvastas rohkem nõudeid kui meetod, mis oli algselt kasutusel juhtumi organisatsiooni poolt ja tuvastatud nõuded olid ka parema kvaliteediga. Meetodi rakendamine võttis märkimisväärselt vähem aega, tuvastamise protsess oli hästi kontrollitav, see võimaldas täpsemalt hinnata tuvastamisele kuluvat aega ja seeläbi on meetodit kasutades lihtsam protsessi planeerida ja ülesandeid delegeerida.

Märksõnad: Nõuete tuvastamine, nõuete avastamine, äriprotsesside modelleerimine, BPMN.

Table of Contents

1. Introduction

"The single hardest part of building a software system is deciding precisely what to build" [1]. A software system aims at resolving a problem or satisfying a need. Its effectiveness is highly dependent on how well it can resolve or address the need it was designed to satisfy. In order to provide the best solution, it is crucial to understand the actual needs and the interfering constraints that apply in the surrounding environment. These issues are addressed within the field of Requirement Engineering (RE) [2].

Uncovering and extracting the information needed for building a software solution, is one of the initial tasks of Requirements Elicitation [3],[4]. Elicitation of requirements is concerned with learning the environment and discovering the needs of users and other stakeholders such as customers. One of the primary sources for elicitation of requirements is the system (processes, organization, environment and legacy systems) currently being used [1]. Although a large extent of this information lies with the stakeholders, it is often captured in written form, such as manuals, policies, standards, and graphical models. [5] These documents are therefore important sources of information for requirements elicitation. Graphical representations like models and diagrams in particular, are gaining more and more popularity when it comes to describing current systems.

Such models and diagrams facilitate communication between stakeholders, help to better understand the domain, provide input for solution designs and documentation of systems [6]. There are many modeling notations created for specific purposes and they are used to represent either static phenomena (e.g things and their properties) or dynamic phenomena (e.g events and processes) or both [6]. In an organization, managers need to coordinate the efforts of workers, and therefore behavioral aspects, such as processes or workflows, are often modeled. For this purpose, business process models (BPMs) are used. These models are also valuable sources of information for requirements elicitation. In fact, these models are not only used to understand the environment [7] but are increasingly becoming an important part of the requirements specification process [8].

There are many methods to model business processes [9]. Most of them were created before the bloom of information technology and are therefore more business oriented, aiming at improving decision-making. With the advent of information technology, software developers sought to understand the environment for better solution designs. In this quest, business process models proved to be helpful. However, the notations and levels of abstraction used

were not of a satisfactory level of detail for system design. To remedy this, notations more suited for software engineering were developed (e.g the Unified Modeling Language). These were later extended to cover the needs of business process modeling as well. Unfortunately, business oriented modelers did not start using the same notation. This might be due to the notational languages being overly complex and not aligned with the main focus of business processes [10]. The challenge to make the process notations more intuitive, understandable and usable by a broader range of stakeholders, has always been and still remains there for the RE community. As such, it is becoming increasingly more important to satisfy a delicate balance between formal (analyzable) and informal (often high-level and intuitive) artifacts that invites the many stakeholders to participate in the process of eliciting requirements [11],[4]. It is predicted that the future of software engineering and RE in particular, is likely drifting towards the minimization of the gap between the business and the technical side [12],[11]. It seems that the business analysts must start providing models that are more useful for technical use and vice versa.

Today Business Process Model and Notation (BPMN)[1], is gaining popularity among business analysts and technical developers [13]. This is because BPMN is aimed at creating a notation that is easy to understand by both business users and software developers, but is powerful enough to support the development of systems from business process design to process implementation [14]. BPMN as a notation language covers over 100 symbols and can be very complex when needed. However, it is scalable and only a handful of intuitive symbols is enough to start modeling business processes [13]. As such, BPMN models are increasingly becoming an important source of information for software requirements elicitation.

Although BPMN models are widely used and gaining popularity by the business side, they are rarely on the level required for requirements elicitation. So despite the fact that process models are a valuable source of knowledge for software projects, they are rarely used as a source or common artifact for discussing requirements. One reason for this is the lack of concrete and comprehensive guidelines, methods or other tools created to systematically analyze and improve the BPMN models so that they would be normalized, complete, consistent, bounded and on the necessary level of granularity for requirements elicitation.

[1] Created by an international, open membership, not-for-profit computer industry standards consortium Object Management Group [23]

In the light of this context, the goal of the thesis is to create a systematic method for eliciting high quality requirements from BPMN models. More specifically a method that elicits requirement specifications that are:

- complete (include all the data needed for a requirement);
- consistent (with no internal contradictions);
- bounded (include relevant data for the software engineering project);
- and on the required level of granularity for a specific project.

The rest of the thesis is structured as follows: Chapter 2 introduces the conceptual foundation of the method. Chapter 3 describes the proposed method. Chapter 4 presents a case study and its results, and Chapter 5 discusses related work. The thesis is concluded in Chapter 6 with conclusions and a description of future work.

2. Conceptual Foundation

To elicit requirements from BPMN models, the conceptual foundations of the proposed method of requirement elicitation from BPMN models must be set. Firstly, it is important to define what a requirement is and what the components of a requirement are. Secondly, it is necessary to discuss BPMN and the elements it is made of. Thirdly, it is necessary to map BPMN elements with the components of a requirement. These three issues are presented and discussed in the following sections of this chapter.

2.1 Requirement

Many definitions and attempts to decompose the essence of a requirement have been made [15]. A simplified approach is to state that a requirement is a description of what a product must do and how it should do it [5], but this statement is too generic for evaluating whether the BPMN model has got the knowledge required to elicit requirements.

In order to understand what a requirement consists of, it should be decomposed into more detailed components. There are many domain analysis methods and ontology based methods focusing on RE which suggest ways to decompose the requirement into a set of components. Domain Theory for Requirements Engineering [16] decomposes requirements into components. This decomposition is considered as complete [17] and is widely accepted in the field of RE [18]. Furthermore, Domain Theory is not domain dependent and is specifically useful for requirements elicitation and specification [17]. As such, we define the components of a requirement, based on this theory.

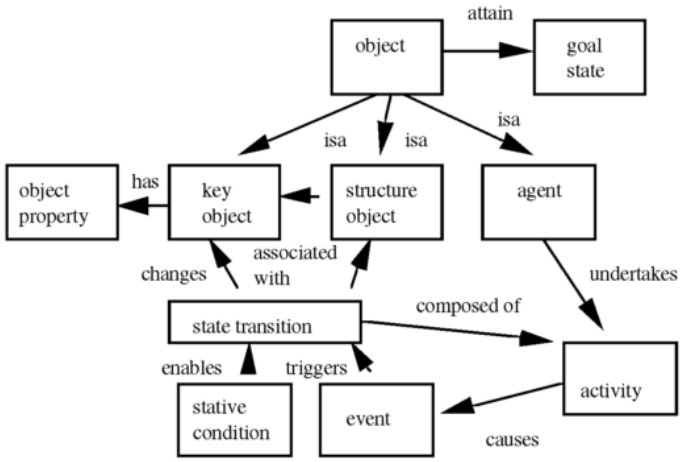

Figure 1 Meta-schema of knowledge types for domain modeling [16]

Domain Theory for Requirements Engineering [16] is an attempt to give structure to the knowledge needed for requirements engineering. It is created based on cognitive science and

the human use of analogical reasoning. The theory provides a structure of knowledge types (see ***Figure* 1**) present in RE and suggests domain knowledge to be represented in two types of generic models: Object System Models (OSM) and Information System Models (ISM). OSM describes the essential transaction of the application in terms of a set of cooperating objects and their behavior. ISM contains processes that report on and provide information about an OSM.

Knowledge types that form the primitive components of a requirement are Object, State Transition, Goal State, Activity, Event and Stative Condition. An Object can be of type Key, Structure or Agent. A Key Object is the subject matter of the essential system transaction and therefore undergoes state change. Structure Objects are passive objects and environmental conditions, which would not normally appear in data models (e.g a warehouse, a library, air corridors in air traffic control, etc). Structure Objects model approximations to the real world entities, which must be persistent, have spatial properties and express containment or possession of Key Objects (e.g a library contains books). They can be internal (e.g a warehouse, a shelf) or external (e.g at the supplier's, at the customer's). An agent carries out Activities, which may then create Events initiating State Transitions. Agents can be categorized as human or automated agents (e.g a computer system). Objects have properties that define their characteristics, which constrain their behavior. Properties can be of type physical, financial or conceptual.

A State Transition changes the state of an Object by transferring its membership between Structure Objects to achieve a desired Goal State (e.g a book is borrowed and moves from the library to the borrower). States can be primary or secondary. Primary states record the containment or possession of Objects in structures. Secondary states belong to Objects independent of structures, and describe states such as being reserved or scheduled.

Goal States describe a future, required state, which the system should satisfy, maintain or sometimes avoid. Goals can be specified by either describing the state, which the object system must achieve, or by describing algorithms and processes, which must be carried out. Also, sometimes a goal can be the production of some information, which is satisfied by activities in the ISM.

Activities are processes, which normally run to completion resulting in a state change. They are carried out by actors, trigger the state changes and cause Events. An Event is a single point in time when something happens and can be of type domain or time. Events initiate State Transitions. Stative Conditions are preconditions and post conditions to State

Transitions. Relationships add further structure information to OSMs and show the relationships between the components described above. They can be of type cardinality, temporal or scale.

Domain Theory for Requirements Engineering and especially the meta-schema of knowledge types for domain modeling gives a complete enough list of components needed for RE. The theory provides a systematic method of examining where the BPMN model contains the information needed and what information can be found. The Domain theory also provides a procedure for applying the theory on requirements elicitation. The procedure suggests the following steps: identifying any sub-systems in the application, establishing the purpose of the sub-systems, describing the activities that the agents carry out, and integrating them into a generic system model for the application.

2.2 Business Process Model and Notation

BPMN is an initiative that provides a modeling notation for people who design and manage business processes [14]. A business process is a collection of related, structured activities or tasks that produce a specific service or product for a particular customer [19].

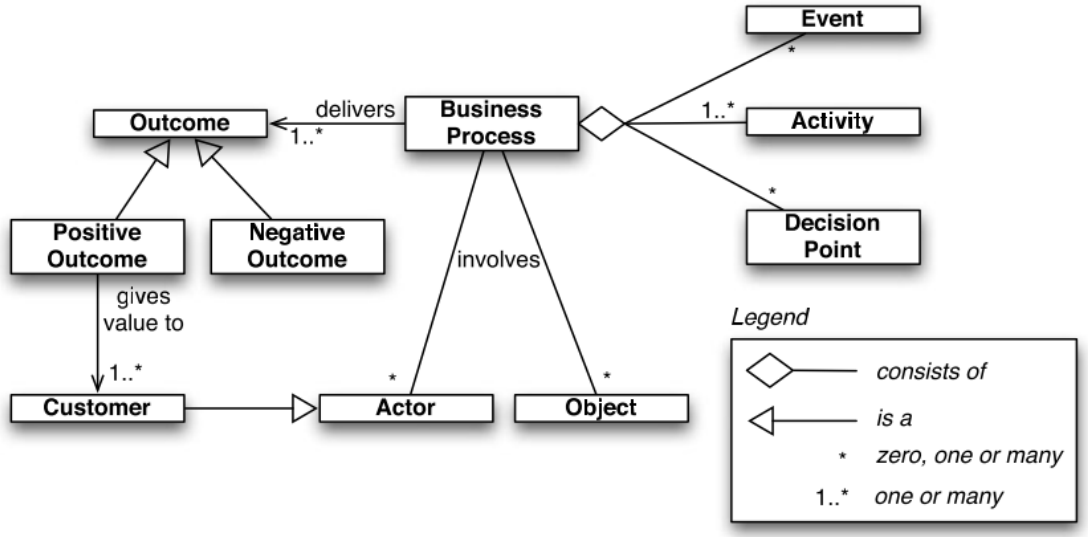

Figure 2 Ingredients of a business process [13]

Dumas, Rosa, Mendling and Reijers [13] split the business process into elementary components as seen in ***Figure* 2**. Events correspond to things that happen atomically, meaning that they have no duration. An event may trigger the execution of a series of activities. An activity is a major task that must take place in order to fulfill an operation contract [20]. Decision Points are points in time when a decision is made that affects the way the process is executed. An actor is a human actor, organization or software system acting on behalf of

human actors or organizations that take part in the process. An actor can be the one that carries out the activities or the one that benefits from the output of the process (such as a customer). Objects are things that are needed to carry out the activities (e.g tools, information) or things that are created, changed or disposed during the activity (e.g a cake, a book, a report). Objects can be physical (e.g materials, paper documents) or immaterial (e.g electronic records). A process results in an outcome. Ideally, an outcome should deliver a value to the actors involved in the process (such as a customer), but this is not always the case and a process can also lead to negative outcomes.

Business process models can visualize business processes. As discussed in the introduction, BPMN is becoming a mainstream process modeling notation. BPMN is a notation language that provides possibilities to define business processes that can be applied in an execution language (BPM Systems WS-BPEL 2.0). As such, with BPMN, processes can have fairly complex process semantics [14] while being intuitive to business users. It is a notation that aims to bridge the gap between business users and technical experts [13].

BPMN supports the concepts that are applicable to business processes but not high-level modeling like organizational modeling, data and information modeling, strategy modeling and business rules modeling. Although it is possible to show the flow of data and the association of data artifacts to activities, it is not a data flow language.

In BPMN there are five basic categories of elements. These are Flow Objects, Data, Connecting Objects, Swim Lanes and Artifacts. Flow objects consist of Events, Activities and Gateways (a gateway is the equivalent of a decision point). Data is represented by Data Objects and Data Stores. Data Objects show what data is required or produced (data inputs and outputs) during an Activity. Data Stores represent data that is preserved beyond the scope of the process. Connections between Activities and Data Stores represent data retrieval or data update. Data elements represent the information part of the object component described in the business process ontology. Connecting Objects make up a Sequence Flow that shows the order the Activities are performed in. A Message Flow shows the flow of messages between 2 separate participants. Associations associate data and text artifacts with flow elements. Swim Lanes represent participants (actors) in the process. There are two levels of Swim Lanes: Pools can consist of Lanes that are sub-partitions of Pools (e.g the sales department as the Pool and a sales person as a Lane). Artifacts consist of Text Annotations or Groups. Text Annotations allow to add notes that describe the process, or they can be used to give

instructions to the tasks or processes. Groups organize the tasks or processes that have some kind of significance in the overall model.

All of the described elements of BPMN have a standardized design and must be similar in all BPMN models. ***Figure* 3** shows one way how the core elements of the model could look.

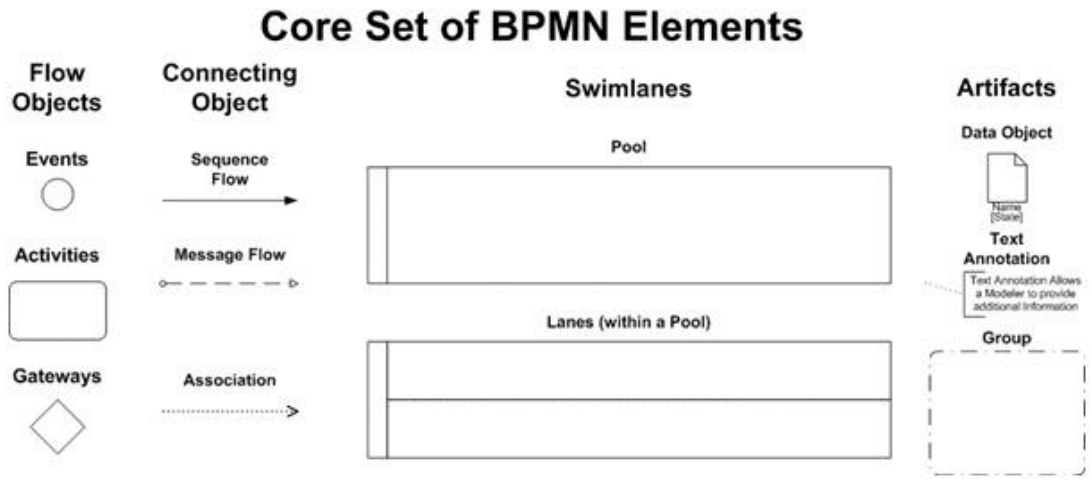

Figure 3 Core set of BPMN elements [21]

The core elements can be supplemented with different additional markers that specify a specific attribute or behavior of the element. For instance, a marker representing a letter inside an Event circle or an Activity box means that the element is involved in either sending or receiving a message. The full set of elements can be found and studied in detail in the documentation of the notation [14]. Additionally it is possible to add additional attributes to the core elements and this can further enrich their meaning. Furthermore it is possible to add custom elements to satisfy a specific need, but such extensions are not included in standard BPMN.

2.3 Mapping a Requirement to BPMN

Once a requirement and BPMN are decomposed into components, it is possible to compare and analyze whether the component in one domain has got corresponding counterparts in the other, at what level of detail and whether the level of detail is sufficient for requirements elicitation.

Table 1 shows the mapping of the components of a requirement to their counterparts in BPMN. The first column of the table lists the components of the requirement. The second column gives a definition of the requirement's component in order to better grasp its essence. The third column provides the corresponding element(s) of BPMN. In the fourth column, a brief comment is made on the given matching.

Table 1 Mapping of Requirement and BPMN Components

Requirement	**Definition**	**BPMN**	**Analysis**
Key Object	A Key Object is an object that goes through a state change	Data (Data Objects, Data Store, Data output, Data input) or Artifact (Text Annotation)	Information about a Key Object in BPMN can be found in Data elements or in Text Annotations added to the model. Data elements give information about the Key Object. A knowledge base can be built up by examining the Data elements more closely. Also additional information about the Key Object can be found in Text Annotations.
Structure Object	A Structure Object represents passive objects and environmental facts	Data (Data Objects, Data Store, Data output, Data input) or Artifact (Text Annotation)	A Structure Object is basically a certain type of property of a Key Object and the information in BPMN can be found in the same form as in case of Key Objects, thus from Data elements or Text Annotations.
Agent Object	An Agent carries out Activities	Pool, Lane	In order to determine who is performing an Activity, it must be examined, which Pool or Lane the Activity belongs to.
Object Property	Objects have properties that define the characteristics that constrain their behavior	Data (Data Objects, Data Store, Data output, Data input) or Artifact (Text Annotation)	Key, Structure and Agent Objects have properties that play a crucial role in requirements definitions. They define Stative Conditions under which the process can proceed. Information about object properties can be found in Data elements or Text Annotations.
Goal State	Goal States describe a future, required state, which the system should satisfy, maintain or sometimes avoid.	Data (Data Objects, Data Store, Data output, Data input) or Artifact (Text Annotation)	A Goal State is a set of Key Object properties and its relationship to Structure Objects when the process has reached a positive outcome. Since the information about the Key and Structure Objects is found in Data elements and Text Annotations, the Goal State is also described in the model the same way.
Activity	Activities are processes which normally run to completion resulting in a state change	Task, Activity, Transaction	Presented clearly as Tasks, Activities or Transactions in the model.
Event	An Event is a moment in time that may trigger the execution of a series of	Event	Presented clearly as Events.

	Activities.		
Stative Condition	Stative Conditions are preconditions and post conditions to State Transitions	Data (Data Objects, Data Store, Data output, Data input) or Artifact (Text Annotation), in description of the outgoing node of a Gateway.	Stative Conditions consist of Object Properties and therefore can be found in the Data elements and Text Annotations of a preceding Event or an Activity element. Also, if the preceding element is a Gateway, some of the conditions are described as descriptions of the outgoing node of a Gateway.
Relationship	Relationships show the relationship between components	Connecting objects	Presented clearly as Sequence Flows, Message Flows, Associations or Data Associations.
Information System Model	Contains processes, which report on and provide information about an Object System Model.	Data Object, Data Store, Message Flows, message Events, send task, receive task.	ISM is presented by a number of components and is represented as a Data element or as Message Flows.

The table shows that every component of a requirement has a corresponding counterpart in BPMN. Oftentimes, the one component of a requirement will have its matching counterpart in several BPMN elements. As such, a complete set of data for requirements must be gathered from multiple elements. This is, in particular, applicable when the components involved have to do with static phenomena such as Objects, Object Properties and Goals.

In conclusion, BPMN models have all the required elements needed to represent all components of a requirement. Nevertheless, it should be born in mind that BPMN models will describe business processes at varying levels of granularity. Furthermore, BPMN models usually have no information about non-functional requirements [22]. There might be some reference to performance related requirements [23] or other information about non-functional requirements in associated annotated text artifacts [24], but this is not a systematic way to specify the non-functional requirements in BPMN.

3. Requirements Elicitation Method (REM)

In this chapter, a method for eliciting requirements from business process models is described. The method approaches a process model systematically to enable requirements elicitation that is complete, consistent, non-contradictory, relevant and on the required level of granularity. The chapter is structured as follows: Section 3.1 introduces the method and discusses the prerequisites and structure. Section 3.2 describes the proposed method. Section 3.3 presents a compact template of the method.

3.1 Introduction to REM

The input of the method is any set of business process models that are captured using a notational language that largely uses the same or similar elements as those used by BPMN. If the business process is modeled with elements that do not have a corresponding match in BPMN, the elicitation of requirements will suffer. Furthermore, as many models require domain specific knowledge in order to be well understood, access to domain experts is necessary. If domain experts are not involved, it will be next to impossible to elicit requirements, as this method is based on questions that will bring clarity about the actual needs of stakeholders. If either of these two prerequisites is not met, problems might arise.

The Activities of the models are the focal point of the method. Every Activity in the model is thoroughly investigated by applying a series of actions that help to elicit the requirements. The chosen model is examined and discussed by following the logical sequence of Activity elements in the model. In the case of splits (parallel or exclusive) in the process models, it is recommended to follow one branch to the end of the process and then follow the other branch until it joins the path already covered or until the end of its own path.

On every Activity element in the model, steps as illustrated in **Figure *4***, are applied.

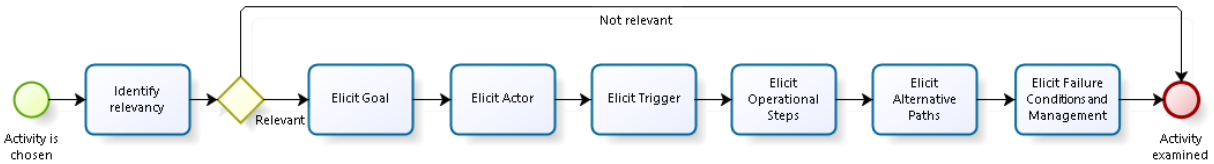

Figure 4 Illustration of the method steps

1. Relevancy to the system under construction (SUC) is identified. The SUC is the system-to-be, which the requirements are elicited for.
2. The goal of the Activity is determined.
3. The actor performing the Activity is elicited.

4. The trigger of the Activity is elicited.
5. The operational steps contained in the Activity are elicited.
6. The alternative paths, by which the purpose of the Activity can be achieved, are explored.
7. The failure conditions and management of failures is elicited.

The method provides a set of questions to be asked the domain experts. These questions ensure that the information needed to complete a requirement is elicited. In addition, the method clarifies which BPMN elements contain the implicit information needed to specify a requirement. In practice, the information is derived primarily through workshops but also via interviews, introspection and observation. In fact, prior to the workshops, an examination of the model by the analyst is recommended. During the examination, specification templates are filled in with preliminary data. The data is gathered by following the method without applying the questions, but examining the BPMN elements suggested by the method. This prior examination helps to get acquainted with the domain, prepares the specifications and saves time during the workshops.

For each relevant Activity of the model, a requirement specification is created. A requirement specification template (see ***Table 2*** Requirement Specification Template) is filled in with information gathered during the application of the method. In some cases, it might be possible to represent several Activities in the same specification template. This is usually the case when the model is highly detailed. In order to determine if several Activities would benefit from being managed together as one requirement, the following questions (inspired by Cockburn [25]) can act as a guide:

- Are the consecutive Activities carried out by one person, in one place, and at one time?
- Is a break between the Activities not possible/reasonable?

If both questions get positive answers, it makes sense to manage the Activities together as one requirement. If Activities are performed by the same actor, in the same location and the Activities follow each other immediately without a break, they form one logical Activity and should be taken as one.

The requirement elicitation and documentation of the functional requirements is approached in accordance with the level of granularity that has been agreed upon prior to starting the work. The level of granularity in regards to the requirement specifications is obviously

dependent on the project or the phase it is in. However, these questions are assumed to have been clarified together with the domain experts before the elicitation process commences.

3.2 Description of the Method

The output of the method is a set of requirement specifications. The requirement specification template (see ***Table* 2** Requirement Specification Template) contains all the information necessary for a requirement to be complete. Its design was inspired by Cockburn [25] and Luis, Vara, Sánchez and Pastor [26]. A completed specification template covers all relevant components for a complete requirement.

Table 2 Requirement Specification Template

Component	**Description**
ID:	
Business Process (optional):	
Activity:	
Goal:	
Actor:	
Trigger:	
Steps of Activity (positive scenario)	Operational steps: Step 1: ……. Step 2: ……. Step 3: ……. …
	Alternative paths: In case 1: …….. In case 2: …….. In case 3: …….. …
Failure conditions and management (optional):	

Fields of the requirement specification explained:

- "ID" - a unique ID for the requirement specification. The ID can be used as a reference e.g in other specifications or correspondence.
- "Business Process" - the name of the process (sub process) model in focus. This field will be necessary when there is more than one process model.

- "Activity" - the name of the Activity that is being subjected to requirement elicitation.
- "Goal" - the expected outcome of the Activity.
- "Actor" - the name of the actor that performs the Activity.
- "Trigger" - when the actor of the Activity should start the Activity.
- "Steps of Activity" is divided into:
    - "Operational steps" - the most preferable path to successfully finishing the Activity.
    - "Alternative paths" - situations where the preferable path cannot be used, but where alternative paths exist.
- "Failure conditions and handling" - situations where the Activity cannot start or must be interrupted and what actions must or can be executed additionally. The Failure conditions and handling part of the template is optional to fill in, as it takes a lot of effort to elicit and it might not bring considerable value to the project (especially in case of smaller projects).

As stated before, all relevant components of a complete requirement are covered by the Specification Template. In **Table *3*** Mapping of the Requirement Components to the Specification Template, each component of a requirement (by Domain Theory of RE) is mapped to a field in the Specification Template and the connection is discussed.

Table 3 Mapping of the Requirement Components to the Specification Template

Requirement	**Specification field**	**Comments**
Key Object and its Properties	Goal, Steps of the Activity, Failure conditions and handling	Information about a Key Object in the template can be found in the Goal field, as an Activity always does something to the Key Object of the process. Since the Steps of the Activity form the Goal, Step fields of the template describe the formulation of the Key Object during the Activity in more detail. The Steps can also describe intermediate states of the Key Object. Failure conditions and handling describe states that are not allowed or that the Key Object should have in order to handle the failure.
Structure Object and its Properties	Goal, Steps of the Activity, Failure conditions and handling	The same applies as to the Key Object, as the Structure Object can be part of the Goal and it forms during the Activities. A certain state can also be a cause of failure and can affect failure handling.

Agent Object	Actor	Described in the Actor field, but can also be described as a secondary actor in the Steps of the Activity.
Goal State	Goal	Goal State is described in the Goal field, but also the Steps of the Activity describe the formulation of the Goal.
Activity	Activity, Steps, Failure conditions and handling	An Activity is straightforward, but also, depending on the granularity, Activities can be described in the Steps section and in Failure handling
Event	Trigger, Steps, Failure conditions and handling	Events are described in the Trigger field, but also under description of actions (especially Alternative Paths). Also, Failure conditions happen due to some Events.
Stative Condition	Trigger, Steps, Failure conditions and handling	The trigger of the Activity will appear under certain conditions. Steps (especially Alternative paths) will follow a path under certain conditions. Failure takes place under certain conditions and can be handled under certain conditions.
Information System Model	Goal, Steps, Failure conditions and handling	The Goal of an Activity can be to produce some information and to perceive it. Information needed to carry out the Activity can be found in Steps of Activity, as they might be required. The missing of information can result in a failure. The information created during the Activity can be part of the Goal or result of some Step or Failure handling.

Every field of the Specification Template corresponds to the elements of the Domain Theory of Requirements Engineering (see details in Chapter 2.1). As such, the template covers all elements of the Domain Theory. Therefore, a template that has all its fields populated with data, is a complete requirement specification. In the following sections, the elicitation of information needed to fill in the template is described.

Step 1: Identify Relevancy

The first step is to determine whether the Activity is relevant, i.e. will the Activity require some form of system support and as such, need to have its functional requirements specified. An Activity that is not related to the SUC, is not further dealt with.

The following questions are to be asked in order to determine whether an Activity is relevant or not:

- Is a computer based system used during the Activity?
    - Is the SUC used or involved (in the background) by providing, executing or receiving any data during the Activity?
    - Are there external systems (e.g customers, a bank, other departments, etc) involved and should the SUC communicate with them?

If the answer to one or both of the questions above is yes, the Activity is relevant, as the Activity has or requires some form of support from an IS.

In a BPMN model, the relevancy of an Activity can be determined by:

- A Manual Task Marker: If a Manual Task Marker is attached to the Activity, the Activity is performed manually and has no relation to/support of an IS. Therefore it has no relevancy for the SUC and can be disregarded (provided it has no implicit associations with databases).

For each relevant Activity a requirements specification is created and assigned a unique ID. In addition, the name of the Activity and the process model it belongs to will be filled in.

Step 2: Elicit Goal

An Activity is always performed in order to meet some interest of the stakeholders (a person, an organization or a system). In this step, the expected outcome that meets the interests of the relevant stakeholders is elicited and described.

The following questions must be asked to elicit the goal:

- What changes after the Activity has been performed?
    - What needs to be accomplished?
    - What form and/or format do the results come in?

In a BPMN model the following elements indicate the result of the Activity:

- Outgoing Message Flow: If an outgoing Message Flow is attached to the Activity, it indicates that during the Activity a message is created and sent to an external stakeholder. Therefore, it forms at least a part of the Goal of an Activity.
- Data Object connected with an outgoing Arc: If a Data Object is attached to the Activity with an outgoing Arc, it indicates that during the Activity a Data Object is created or updated (e.g a document is printed or a report is created). Therefore, it forms at least a part of the Goal of an Activity.
- Data Store connected with an outgoing Arc: If a Data Store is attached to the Activity with an outgoing Arc, it indicates that data is changed (created, updated or deleted) in some Data Store (e.g an invoice is saved to the database). Therefore it forms at least a part of the Goal of an Activity.

All gathered information must be specified in the Goal section of the Specification Template.

Step 3: Elicit Actor

In this step, the actor performing the Activity is elicited. The actor can be human (a role, a department or an organizational unit) or a resource (non-human, such as a machine or an information system). If the actor is an organization, it is assumed that some person working in that unit is performing the Activity. The actor elicited here, might not be the one doing all the operational steps needed to finish the Activity. The actor might use a resource (e.g a computer program) to achieve the Goal of the Activity. These are called secondary actors and will be elicited in Step 5: Elicit Operational Steps of This Method.

The following question is asked in order to elicit the actor:

- Who are the actors that execute the Activity in order to achieve its Goal?

In a BPMN model the following elements indicate the actor of an Activity:

- Pool and Lane: If the Activity is inside a Pool box or in both the Pool and a Lane box, the Pool and Lane name indicate who the performing actor of the Activity is. The performing actor is a participant in the business process and can be a specific entity (e.g a department) or a role (e.g an assistant manager, a doctor, a student, a vendor).

All gathered information must be specified in the Actor section of the specification.

Step 4: Elicit Trigger

It is important to understand how the actor performing the Activity knows that it is time to start the Activity i.e the trigger of the Activity. There are three ways to trigger an Activity: 1. The actor receives a message. 2. The Activity starts at a certain time. 3. The Activity starts right after a preceding Activity is finished.

In the first case the message notifying the actor to start the Activity can be e.g a verbal message, an email, a letter, a document received, a horn sound, etc. In the case of a scheduled trigger, the Activity can start e.g every 5 seconds, at 10 o'clock, etc. An Activity starting right after a preceding Activity is only an option if the actor of both Activities is the same. In this case, the actor is aware of when the preceding Activity is finished and thus knows when it is time to start the next Activity.

The following questions must be asked to elicit the trigger:

- How does the actor (human or resource) know when to start the Activity?

    - Is the actor informed by a message? What form or format does the message come in?
    - Does it start depending on time? How is the actor aware of time?
    - Is the actor also responsible for the preceding Activity in the process?

In a BPMN model, the following elements indicate the trigger of the Activity:

- A preceding Event element: If the element preceding the Activity is an Event, it indicates the trigger of the Activity. The Event is a moment in time that happens, and once the Event happens, the Activity is triggered. The type of the Event (the marker of the element) and the description of the Event give further information about the trigger. A marker can clearly say what type of a trigger it is (e.g message or scheduled) and a description can add further detail (e.g email received or at 10 o'clock).
- A preceding Activity element: In the case the Activity is not preceded by an Event element but by another Activity element instead, it is necessary to check if the Activities both belong to the same Pool or Lane. If they do, the Activity is triggered when the previous Activity ends. If they do not belong to the same Pool, the questions presented must be applied, as it is not clear how the actor knows when to start the Activity.

All gathered information must be specified in the Trigger section of the specification.

Step 5: Elicit Operational Steps

An Activity might consist of one or many operational steps that must be completed in order to reach the Goal of the Activity. Although there might be different ways to reach the Goal, in this step the standard set of operational steps performed to reach the Goal is described.

There are three types of operational steps: 1. Actor interaction - The performer of the step interacts with some other actor (e.g another person, the SUC, an external system, a barcode scanner). 2. Action verification – the SUC verifies that some conditions are met (e.g a customer credit limit must not be exceeded). 3. Internal action – the SUC changes some data internally (e.g enters to transaction log, creates a financial transaction, updates the warehouse).

The following questions must be asked to elicit the operational steps:

- What actions are performed during the Activity?
    - Who performs the operational steps?
    - What actions does the performer do during the execution of the Activity?
    - What tool does the performer use (e.g the SUC, another person, an external system)?

    - How is the tool used?

- Is verification of certain conditions needed at any point? Should the SUC verify the conditions?

- Is the SUC additionally changing something internally? Should the SUC do something automatically in the background (e.g create logs, create some transactions, send notifications)?

In a BPMN model the following elements indicate the steps of the Activity:

- A Sub-Process Marker: If the Activity is marked with a Sub-Process Marker, the actions of the Activity are described in a separate model. In such case it is the analyst to decide whether the method is applied separately to the Sub-Process or whether the actions of the Sub-Process are described in this specification.

- A Data Store connected with an outgoing Arc: If a Data Store is attached to the Activity with an outgoing Arc, it indicates that data is changed (created, updated or deleted) in some Data Store (e.g an invoice is saved to the database). Therefore, it can be concluded that at least one of the operational steps is changing data in the Data Store.

- A Data Store connected with an incoming Arc: If a Data Store is attached to the Activity with an incoming Arc, it indicates that data is retrieved from a Data Store (e.g customer data is fetched). Therefore, it can be concluded that at least one of the operational steps is fetching data from the Data Store.

- A Data Object: If a Data Object is attached to the Activity, it indicates that one of the operational steps is either the creating or reading of that Data Object. E.g a document is printed or a document received is read.

- Message Flow: Associated Message Flows indicate a message exchange with external stakeholders. Therefore, one of the operational steps of the Activity is either creating and sending or reading a message.

All gathered information must be specified in the Operational Steps subsection of the Steps of the Activity.

Step 6: Elicit Alternative Paths

In addition to the standard set of operational steps that achieve the Goal of an Activity (described in Step 5: Elicit Operational Steps), there could be situations requiring other operational steps (alternative paths) to be taken. For instance, entering an order when the customer is not registered in the system, requires a deviation from the standard set of operational steps. An alternative path needs to be taken to add the customer. This aspect is elicited and described in this step of the method.

The following questions must be asked to elicit the alternative paths:

- Compared to the operational steps, are there situations where additional or alternative steps must be taken to reach the Goal?
    - What are the conditions?
    - What steps must be taken additionally and what steps must be replaced?

In a BPMN model the following elements indicate alternative paths:

- A Non-Interrupting Boundary Event: If a Non-Interrupting Boundary Event is attached to the Activity, it indicates that in case the Event happens, an alternative set of operational steps will be executed. Therefore, the Event describes certain conditions under which additional operational steps are required. The Activities following the Event indicate the actions that must be taken in case of such Event.
- A Sub-Process Marker: In case the Activity is marked with a Sub-Process Marker, the alternative paths of the Activity may be described in a separate model. In such case it is the analyst to decide whether the method is applied separately to the Sub-Process or the actions of the Sub-Process are described in this specification.
- An Event Sub-Process: If Event Sub-Processes are used, they indicate the conditions under which an alternative path is executed. Event Sub-Processes are surrounded by dotted-line frames and their Start Events represent the conditions when they are triggered. Activities in the Sub-Process are the operational steps.

All gathered information must be specified in the Alternative Paths subsection of the Steps of the Activity.

Step 7: Elicit Failure Conditions and Failure Management

Sometimes it is not possible to execute all the steps needed to finish an Activity successfully. In such cases, the Activity is interrupted, the goal is not reached and interests of the stakeholders are not met or are met partially. In this step, conditions that hinder an Activity from being initiated or where an Activity is interrupted, are elicited. These are called failures in the method.

Additionally, in case of a failure situation, some additional actions must be taken in order to get the best out of the situation. It might be required to protect the stakeholders' interests and limit their losses. For example, a customer must be informed if it is not possible to deliver the goods. Actions that must be taken in case of a failure, are also elicited in this step.

The following questions must be asked to elicit the failure conditions and how the conditions should be managed:

- In what case the Activity should not be started? What are the preconditions that must be fulfilled to carry out the Activity?
- In what case the Activity should not be continued? What might interrupt the Activity?

- Are preliminary actions needed to limit the losses of the failure (e.g auto save functionality, condition detectors, etc)?
- What actions are necessary in case of a failure (e.g undo of actions, error log, notification of stakeholders, etc)?

In a BPMN model the following elements indicate a failure condition and failure management:

- Start failure (preconditions):
    - A preceding Event: An Event element preceding the Activity indicates when the Activity is triggered, but it also describes the preconditions that must be fulfilled in order to start the Activity. For example, an email must be received, otherwise it is not possible to proceed. Furthermore, if it is known that an email must be received, it is possible to discuss the form and format the email must come in, in order to start the Activity.
    - An entering Arc: An Arc can enter an Activity from a preceding element, a Data Object or Data Store, or be an incoming Message Flow. All of these entrances can represent a potential failure situation if the attached element is not available or comes in a wrong form or format. Therefore, they can be possible causes of failure and must be examined.
- Interruption:
    - Boundary Events: If a Boundary Event is attached to the Activity, it indicates the condition when the Activity is interrupted. The type of the Event (a marker attached) gives more detailed information about the condition. Activities following the Boundary Event indicate the steps of failure handling.

All gathered information must be specified in the Failure Conditions and Management section.

3.3 Method Summary

In **Table *4*** Method Summary, a compact template of the method is presented that can be used during the meetings with domain experts. The first column of the table represents the field in the Specification Template that is to be populated with information. The second column shows the questions to be asked the domain experts. The third column lists all relevant BPMN elements related to the field.

Table 4 Method Summary

Specification field	**Questions**	**BPMN elements**
Activity	• Is a computer based system used during the Activity? o Is the SUC used or involved (in the background) by	• Manual Task Marker: If a Manual Task Marker is attached to the Activity, the Activity is performed manually and has no relation to/support of an IS. Therefore, it has

	providing, executing or receiving any data during the Activity? o Are there any external systems (e.g customers, banks, other departments, etc) involved, and should the SUC communicate with them?	no relevancy for the SUC and can be disregarded (provided it has no implicit associations with databases).
Goal	• What changes after the Activity has been performed? o What needs to be accomplished? o What form and/or format do the results come in?	• Outgoing Message Flow: If an outgoing Message Flow is attached to the Activity, it indicates that during the Activity a message is created and sent to an external stakeholder. Therefore, it forms at least a part of the Goal of an Activity. • Data Object connected with an outgoing Arc: If a Data Object is attached to the Activity with an outgoing Arc, it indicates that during the Activity a Data Object is created or updated (e.g a document is printed or a report is created). Therefore, it forms at least a part of the Goal of an Activity. • Data Store connected with an outgoing Arc: If a Data Store is attached to the Activity with an outgoing Arc, it indicates that data is changed (created, updated or deleted) in some Data Store (e.g an invoice is saved to the database). Therefore, it forms at least a part of the Goal of an Activity.
Actor	• Who are the actors that execute the Activity in order to achieve its Goal?	• Pool and Lane: If the Activity is inside a Pool box or in both the Pool and a Lane box, the Pool and Lane name indicate who the performing actor of the Activity is. The performing actor is a participant in the business process and can be a specific entity (e.g a department) or a role (e.g an assistant manager, a doctor, a student, a vendor).
Trigger	• How does the actor (human or resource) know when to start the Activity? o Is the actor informed by a message? What form or format does the message come in? o Does it start depending on time? How is the actor aware of time? o Is the actor also responsible for the preceding Activity in the process?	• A preceding Event element: If the element preceding the Activity is an Event, it indicates the trigger of the Activity. The Event is a moment in time that happens, and once the Event happens, the Activity is triggered. The type of the Event (the marker of the element) and the description of the Event give further information about the trigger. A marker can clearly say what type of a trigger it is (e.g message or scheduled) and a description can add further detail (e.g email received or at 10 o'clock). • A preceding Activity element: In the case the Activity is not preceded by an Event element but by another Activity element instead, it is necessary to check if the Activities both belong to the same Pool or

		Lane. If they do, the Activity is triggered when the previous Activity ends. If they do not belong to the same Pool, the questions presented must be applied, as it is not clear how the actor knows when to start the Activity.

Steps of Activity - Operational Steps	• What actions are performed during the Activity?
	  ○ Who performs the operational steps?
	  ○ What actions does the performer do during the execution of the Activity?
	  ○ What tool does the performer use (e.g the SUC, another person, an external system)?
	  ○ How is the tool used?
	• Is verification of certain conditions needed at any point? Should the SUC verify the conditions?
	• Is the SUC additionally changing something internally? Should the SUC do something automatically in the background (e.g create logs, create some transactions, send notifications)?
	  ○
		• A Sub-Process Marker: If the Activity is marked with a Sub-Process Marker, the actions of the Activity are described in a separate model. In such case it is the analyst to decide whether the method is applied separately to the Sub-Process or whether the actions of the Sub-Process are described in this specification.
		• A Data Store connected with an outgoing Arc: If a Data Store is attached to the Activity with an outgoing Arc, it indicates that data is changed (created, updated or deleted) in some Data Store (e.g an invoice is saved to the database). Therefore, it can be concluded that at least one of the operational steps is changing data in the Data Store.
		• A Data Store connected with an incoming Arc: If a Data Store is attached to the Activity with an incoming Arc, it indicates that data is retrieved from a Data Store (e.g customer data is fetched). Therefore, it can be concluded that at least one of the operational steps is fetching data from the Data Store.
		• A Data Object: If a Data Object is attached to the Activity, it indicates that one of the operational steps is either the creating or reading of that Data Object. E.g a document is printed or a document received is read.
		• Message Flow: Associated Message Flows indicate a message exchange with external stakeholders. Therefore, one of the operational steps of the Activity is either creating and sending or reading a message.

Steps of Activity - Alternative Paths	• Compared to the operational steps, are there situations where additional or alternative steps must be taken to reach the Goal?
	  ○ What are the conditions?
	  ○ What steps must be taken additionally and what steps must be replaced?
		• A Non-Interrupting Boundary Event: If a Non-Interrupting Boundary Event is attached to the Activity, it indicates that in case the Event happens, an alternative set of operational steps will be executed. Therefore, the Event describes certain conditions under which additional operational steps are required. The Activities following the Event indicate the actions that must be taken in case of such Event.
		• A Sub-Process Marker: In case the Activity

		is marked with a Sub-Process Marker, the alternative paths of the Activity may be described in a separate model. In such case it is the analyst to decide whether the method is applied separately to the Sub-Process or the actions of the Sub-Process are described in this specification. • An Event Sub-Process: If Event Sub-Processes are used, they indicate the conditions under which an alternative path is executed. Event Sub-Processes are surrounded by dotted-line frames and their Start Events represent the conditions when they are triggered. Activities in the Sub-Process are the operational steps.
Failure conditions and failure management	• In what case the Activity should not be started? What are the preconditions that must be fulfilled to carry out the Activity? • In what case the Activity should not be continued? What might interrupt the Activity? • Are preliminary actions needed to limit the losses of the failure (e.g auto save functionality, condition detectors, etc)? • What actions are necessary in case of a failure (e.g undo of actions, error log, notification of stakeholders, etc)?	• Start failure (preconditions): o A preceding Event: An Event element preceding the Activity indicates when the Activity is triggered, but it also describes the preconditions that must be fulfilled in order to start the Activity. For example, an email must be received, otherwise it is not possible to proceed. Furthermore, if it is known that an email must be received, it is possible to discuss the form and format the email must come in, in order to start the Activity. o An entering Arc: An Arc can enter an Activity from a preceding element, a Data Object or Data Store, or be an incoming Message Flow. All of these entrances can represent a potential failure situation if the attached element is not available or comes in a wrong form or format. Therefore, they can be possible causes of failure and must be examined. • Interruption: o Boundary Events: If a Boundary Event is attached to the Activity, it indicates the condition when the Activity is interrupted. The type of the Event (a marker attached) gives more detailed information about the condition. Activities following the Boundary Event indicate the steps of failure handling.

4. Case Study

Requirements elicitation is a complex real life process that is heavily influenced by social aspects such as human involvement and interactions with technology. It is a process that is unique and variable, depending on the domain and stakeholders. To test, whether the method presented in this thesis improves the quality or quantity of the requirement elicited, analytical and controlled empirical studies are often not sufficient to make conclusions, because there exists a trade-off between the level of control and the degree of realism [27]. Case studies, on the other hand, provide a deeper understanding of the phenomena under study, exploring the situation in a more realistic environment, and are therefore suited for this thesis.

4.1 Case Study Design

In this chapter, the objective, the research questions, the hypothesis, the selection strategy, and the setting of the case study are summarized. A more detailed description of the case study design can be found in Appendix 1 Case Study Design.

The objective of the case study is "to test whether the application of the method created in this thesis improves the quality and/or quantity of the requirements". The more precisely formulated research questions are: RQ1: Did the application of REM elicit more requirements than the previously used method? RQ2: Did the application of REM result in better quality requirements than the previously used method?

In order to answer the research questions, three sets of requirements are compared during the case study: 1. Requirements gathered using REM. 2. Requirements gathered using the original method of the case company (ICM). The set elicited consists of requirements gathered during the requirements analysis phase of the original project done by the case company. 3. The total number of requirements at the end of the project (the Final Set). The Final Set consists of requirements elicited initially and also during the development and support phase of the project. The Final Set also contains requirements that are not implemented but are deferred or rejected (e.g because of the budget constraints) as they are correct and relevant, just not implemented. The Final Set represents a complete list of requirements at the moment of the case study.

To answer RQ1, the total number of requirements elicited by both REM and ICM must be counted and compared. Additionally, as the magnitude of improvement or decline in quantity

is relative to how many requirements there exist in the Final Set, the coverage percentage of the elicited requirements compared to the Final Set will be calculated.

To answer RQ2, the IEEE Computer Society [28] defines that a good quality requirements specification should be correct, unambiguous, complete, consistent, ranked for importance and stability, verifiable, modifiable, traceable. The four latter attributes are not important in the light of this thesis, as they come to importance in the later stages of requirements engineering process. Although, in order to measure the quality of the elicitation process, it is very important that the requirements elicited are correct, unambiguous, complete and consistent. How correct and unambiguous the requirements were, can be assessed by measuring how many of the requirements elicited were not superfluous but were clear, understandable, unambiguous, and relevant. Additionally, the percentage of non-superfluous (correct) requirements in the total number of requirements elicited will be calculated.

To measure whether the method resulted in a more complete and consistent set of requirements, it must be found, how many of the requirements that were elicited during the project in total (including the development and support phase) were missed by the method. Additionally, it is interesting to see whether REM was able to elicit requirements earlier than ICM, and for this purpose, how many of the requirements, that in the real project were found only during the development and support phase, REM was able to find.

The hypothesis for the study is that *"application of the method improves the quantity and quality of the requirements elicited"*.

In order to give answers to the research questions above and to test the hypothesis, the subject case was to meet the following selection criteria: (1) the requirements elicitation for IS was completed and possibly the IS system was already implemented, (2) the IS was process-oriented, the process was nontrivial and a BPMN model of the process existed, (3) the elicitation method used originally was well defined and used in various projects, and (4) the requirements were documented so that it was possible to separate list of requirements gathered by ICM and also the Final Set.

A company manufacturing branded electric motors was chosen as the subject case and their quality assurance process was chosen for the case study. The process had been modeled in BPMN beforehand by the analyst of the solution developer. The project under the inspection of the case study had been completed 2 months before and the solution was up, running and used daily by the customer. The analyst of the solution developer had documented all the

requirements gathered during the analytical phase of the project and also all additional requirements that had evolved during the project development and post-project (support phase).

The case study is divided into four parts. The first part introduces the method to the analyst, and involves preparations for the next part. The second part is held in form of workshops (interviews) with the analyst. During the workshops the method is applied and specifications are updated with the gathered information. The third part is about converting and verifying the specifications created. Also, summarization of the results and calculation of the measures is done in this part of the case study. The forth part presents an interpretation of the results, comments and discussions.

4.2 Case Study Execution

This section describes the execution of the case study. The section describes in detail the stages of the case study, provides an example of how the requirements specification was filled in and discusses the situations that appeared during the application of REM.

4.2.1 Introduction and Preparation

First, an introduction of the method was conducted, as it is important for the customer to understand how the method is built up, prior to its application. In this way the customer can contribute to the elicitation process more effectively. Second, a preliminary elicitation of requirements from the process model of the case was conducted by the author of the thesis. During the preliminary elicitation, the Activities were examined following the logical sequence of the process model elements. For every Activity, a requirements specification was created and filled with preliminary data gathered by following the method without applying the questions, but examining the BPMN elements suggested by the method. In total, 32 requirement specifications were created. All Activities of the process models were included as no manual tasks were identified. A number of questions and problems rose during the preliminary elicitation. There were situations that did not comply with the rules of BPMN - the elements (especially markers) were not used as intended, often it was unclear what triggered the Activity, the Goal of an Activity was often uncertain, etc. All these questions and problems were written down to be addressed during the next stage.

4.2.2 Application of the Questions

Workshops with the analyst (domain expert) were held next. During the workshops, the logical sequence of the elements of the model was followed and discussed with the analyst.

The set of questions was applied, problems and questions recorded in the previous part were discussed. The gathered information about requirements was documented by updating and altering the specifications created. In the following **Table *5*** Example of a Filled Requirement Specification an example of a requirement specification is presented.

Table 5 Example of a Filled Requirement Specification

Component	**Description**
ID:	003
Business Process (optional):	Supply chain security (purchase)
Activity:	Check the order confirmation and update the order
Goal:	Updated order in SUC (suggested delivery date and order status updated)
Primary Actor:	Purchase department
Trigger:	Order confirmation received by e-mail
Steps of Activity (positive scenario)	**Operational steps:**
	1. Open PDF format order confirmation received by email
	2. Find the relevant purchase order in SUC
	3. Check that ordered materials are the same as on the order
	4. Enter suggested delivery date and change the status to "Confirmed"
	5. Reply the email confirming the order confirmation
	6. Save the order
	Alternative paths:
	1. If order confirmation differs from the order (e.g quantity smaller than ordered), contact the person who created the order and ask for advice; If changes OK follow the normal flow.
	2. If suggested delivery date is later than the needed delivery date, take same actions as in alternative path 1.
Failure conditions and handling:	1. If order confirmation differs from the order and is not acceptable, the order will be deleted and the process will be interrupted.

The template was filled in following the steps of the method. The following paragraphs describe how the example specification was filled in.

Step 1 Identify Relevant Activities – In this step an Activity (presented in **Table *5***) was chosen and its relevancy to the SUC was evaluated. The Activity did not have a Manual Task marker attached to it, and it updated order information in the SUC. The Activity was considered to be relevant to the SUC. Since the specification was created before the meeting,

the ID, the Business Process Name and the Activity were filled in beforehand as they were clearly identifiable from the model and no update of the specification was needed.

Step 2 Elicit Goal – The next step was to identify the Goal of this Activity. In preliminary examination of the Activity it was discovered that the Activity had a Text Annotation suggesting that the delivery date should be updated, which indicated that an updated order with an appropriate delivery date was part of the Goal of the Activity. In addition, the Activity had an outgoing Message Flow to the supplier. The domain expert was asked the suggested questions. By asking the domain expert, "In what form and/or format does the message (the result in the method) come in", it turned out that the received e-mail was replied manually using an e-mail client, and no automation was required. Therefore sending a message to the supplier was not part of the Goal for the SUC. With follow-up questions to clarify the context, the Goal of the Activity was determined to be "An updated order in the SUC (suggested delivery date and order status updated)".

Step 3 Elicit Actor – The next step was to elicit the actors carrying out the Activity. It was clear that the actor was "the purchase department". Any further specification of "the purchase department" was not considered necessary, and as such, the specification was not updated as it had been filled in during the preliminary examination.

Step 4 Elicit Trigger – Once the actor had been identified, the trigger was elicited. A message Event preceded the Activity indicating an incoming message from the supplier. This was already registered in the specification and was considered to be the trigger. Additionally the question "In what form or format does the message come in", was asked. It turned out to be an email with a PDF file attachment. The file was read manually and there was no need for automation, as the suppliers did not send the files in any other format. This was additionally marked down in the specification.

Step 5 Elicit Operational Steps – Next the operational steps required to reach the Goal of the Activity were elicited. Questions provided by the method were applied and 6 steps were elicited (read email, find purchase order, check materials, enter data, reply email, save data). The steps were performed by the same actor elicited in step 3. The following tools were discovered: PDF reader, an e-mail client, purchase order search (in the SUC), save order (in the SUC). No need for verification was elicited, and no internal automations were required. Some of the steps had already been discussed under previous steps and they were recorded now in more detail. Some steps like "Finding an order" led into broader discussions as to

what parameters were used to find the order and where this information was taken from. Still, the search criteria were not registered in the template, as they were not that important and were considered self-evident.

Step 6 Elicit Alternative Paths – Now that the standard operational steps were elicited, the non-standard situations were discussed. As no alternative paths had been elicited beforehand, now questions of the method were applied. It was discovered that alternative paths and failure conditions were somewhat connected, as in the case the order confirmation received was different from the original order (especially the delivery date and the quantities available), a decision had to be made whether to interrupt the process or to accept the changes. In the case of accepting, an alternative path was elicited that required contacting of the creator of the order.

Step 7 Elicit Failure Conditions and Management – In this step, situations that prevented the Activity from starting or interrupted the Activity were discussed. Additionally, the steps needed to be taken in case of an interruption were discovered. As described in Step 6, one failure condition was already discovered and discussed. In this step it was described. Also, method questions were applied and other possible conditions in addition to the already discovered failure situation were discussed, but none was discovered.

4.3 Results

This section describes how the gathered data was prepared and presents the results of the case study.

4.3.1 Data Conversion, Verification and Summarization

The template used by REM to specify the requirements was not of the same form and format as the one originally used by the company. In order to compare the specifications, it was necessary to convert them to the same form and format. It was decided to convert all specifications to the form and format used by the analyst. The converted specifications were recorded on the Microsoft Team Foundation Server as this was the system used by the developer. One specification created by REM in most cases resulted in multiple requirements specifications in the form and format used by the developer. E.g the example provided in **Table *5*** Example of a Filled Requirement Specification resulted in three specifications after the conversion, as the ability to search for an order, update the order and send it by email in PDF format are registered as separate specifications by the developers' method.

After conversion, a verification of the specifications was done, and the requirements specifications were assessed whether they were superfluous or not. The verification was carried out together with the analyst of the developer during a workshop. In a lot of cases, the specifications were considered superfluous by the analyst of the developer, as they were not registered by the developer and were considered self-evident. This is a peculiarity of ERP projects, as the platform has built-in functionality. This does not mean that these requirements were not captured, but they were just not registered by the developer. Such requirements are especially important if the development platform is unknown. Because of that it was decided that obvious requirements that the analyst had not recorded but which still were requirements of the customer, would be classified as not superfluous, but would be marked as obvious and counted in the end and added to ICM and the Final Set. The most frequent example of such requirements was the need to enter or search for/filter a customer or item while entering records like invoices, orders, etc. On an ERP platform this is self-evident and not recorded.

Summarization and calculation of the results was conducted by the author of the thesis. First the number of records in the requirements lists (gathered by REM, ICM and the Final Set) was counted and the result was entered into a spreadsheet. Then the measures described previously were calculated and preliminary conclusions were drawn. Additionally, all comments and suggestions of the analyst were summarized into a short overview. Time spent on the case study was summarized and also time spent on the initial requirements elicitation by the developer was discussed with the analyst over the phone.

4.3.2 Quantity (RQ1)

During the application of REM, 128 requirement specifications were created. After the assessment of the requirements, 7 of them were classified as superfluous (not relevant or incorrect) and 121 as correct (relevant and correct) requirements. ICM elicited 115 requirements, 6 requirements on the list were classified as superfluous and the total number of correct ones was 109. It can be concluded that REM elicited more requirements (121 against 109).

To find out how significant the improvement was, a ratio between the requirements elicited and the total number of requirements in the Final Set (see 4.1 for the definition of the Final Set) was found. The Final Set consisted of 128 requirement specifications. The ratio for REM was 95% (121/128=0.95) against 85% (109/128=0.85) for ICM. The following **Table *6*** Quantity summarizes the gathered information about which method resulted in more requirements.

Table 6 Quantity Measures

Measures	REM	ICM	Final Set
No of requirements after conversion	128	115	128
No of correct requirements	121	109	128
Ratio %	95%	85%	

4.3.3 Quality (RQ2)

In order to find out how correct and unambiguous the elicited requirements were, the number of requirements that were correct (not classified as superfluous) and the number of all elicited requirements of the method was found. REM was able to elicit 121 correct requirements and 128 in total, which makes the proportion of correct requirements in the total pack for REM 95% (121/128=0.95). For ICM 109 correct requirements were elicited out of 115 requirements in total and the proportion is also 95% (109/115=0.95), meaning that the correctness and unambiguousness of both methods was the same. The following ***Table 7*** Correctness and Unambiguousness Measuring summarizes the measures and calculations.

Table 7 Correctness and Unambiguousness Measuring

Measures	REM	ICM
No of requirements after conversion	128	115
No of superfluous requirements	7	6
No of correct and unambiguous requirements	121	109
Proportion in total pack %	95%	95%

REM was not able to elicit 12 requirements that were registered in the Final Set, which makes 9.4% (12/128=0.094) out of the total number of requirements in the Final Set. In the case of ICM, 19 requirements were not elicited and this makes 14.8% (19/128= 0.148) out of all requirements in the Final Set. In this aspect, REM was more complete and consistent than ICM. The following ***Table 8*** Completeness and Consistency Measuring summarizes the measures and calculations.

Table 8 Completeness and Consistency Measuring

Measures	REM	ICM	Final Set
No of requirements not elicited (in the Final Set)	12	19	128
Proportion of not elicited requirements %	9.4%	14.8%	

Additionally, REM elicited 2 new requirements that were not registered in the Final Set, but were still considered to be correct requirements. These two requirements might be

implemented in the future and therefore the completeness and consistency of REM is even higher. Another aspect that can be taken as a compliment to the quality measure is whether REM was able to discover requirements that were originally discovered only in the development and support phase of the project, and indeed, it was able to discover 4 requirements out of 6 that in real life were discovered only in the development and support phase.

4.3.4 Effort

The last aspect that must be considered is the effort spent on the methods. Introduction of the method in total took 4 man-hours (one 4 hour session). The preliminary elicitation of the requirements from the model was conducted by the author of the thesis and it took in total 10 hours. Workshops with the analyst (domain expert) took in total 16 man-hours (in a series of 4 workshops, 4 hours each). After each session, the author refined the specifications as they were written in a hurry during the sessions. The work done after the sessions took an additional 16 hours. In case of REM, the time spent on the application of the method was 46 man-hours.

The time spent by the developer on elicitation of requirements was not straightforward, because the elicitation of requirements in the initial project was done in parallel with the understanding of the domain and modeling of the process. In order to find out what was the time spent purely on elicitation, the share of time spent on other activities was assessed and deducted from the total amount of time spent on the requirements engineering phase of the project. For ICM, the effort was assessed to have been 60 hours.

4.3.5 Discussion

The results show that REM was able to elicit more requirements than ICM. The improvement in quantity was noticeable, as coverage rose from 85% to 95% when REM was used. From this, the answer to RQ1 “Did the application of REM elicit more requirements than the previously used method?” is “Yes, it did elicit more requirements.” Also, the results show that the quality of the elicited requirements was better despite the fact that the percentage of non-superfluous requirements in the total amount of requirements was the same. The REM missed less requirements and was able to elicit requirements earlier than the previously used method. Therefore, answer to RQ2 “Did the application of REM result in better quality requirements than those elicited by the previously used method?” is “Yes, REM resulted in better quality requirements.” Additionally, the application of REM required 14 hours less and can therefore be considered less time consuming and less costly to apply. The formulated hypothesis for the

study *"application of the method improves the quantity and quality of the requirements elicited"* is correct.

The general impressions form the analyst of the developer used as the domain expert in the case study were that the approach is more structured than the method used today. A more structured approach gives better control over the elicitation process, it is easier to evaluate how much effort is needed, it is better to plan and delegate the work, and there are less chances for something to get overlooked. The method was good at evaluating the consistency and completeness of the model. It was amazing how many mistakes were found in the original model, although REM did not result in better correctness.

4.4 Threats to Validity

The case study method has validity issues that ought to be considered. These threats can be in regard to construct validity and external validity [27].

Construct validity is concerned with to what extent the operational measures that are studied really represent what the researcher has in mind. Styles of writing specifications can vary drastically in different projects and the counting of the number of requirement specifications can be considered as a threat to construct validity, as the measure is subjective. The problem was addressed in this thesis by converting the specifications that were verified by the domain expert to assure that they were created on the same level of detail and using the style used by the company. In short, the domain expert verified the new set of requirement specifications.

External validity is concerned with to what extent it is possible to generalize the findings. The method was applied on one case study, and therefore it has the inherent limitations of the case study method in regards to how much the results can be generalized. The results are naturally dependent on aspects such as the domain expert, the type and size of the project, and the elicitation method used by the company. On the other hand, it was a real life application of the method on a non-trivial project. As such, although the results cannot be generalized, they are still valuable.

5. Related Work

The concept of using process descriptions or models during the requirements elicitation process has been deployed before in the literature. This chapter gives an overview of state of the art approaches to the subject of eliciting requirements from the process models and other models.

5.1 Eliciting Requirements from Business Process Models

Luis et al. [26] work is probably the closest to the thesis. It describes a method to elicit requirements in three stages, where first organizational modeling is done in BPMN, then the model is validated by purpose analysis (which is the main contribution of the paper) and finally functional requirements specifications are created from the refined BPMN models. It creates use-case like specifications and suggests the elements of the model to be used in order to fill in the specification. The purpose analysis stage of this method can be a strong addition to REM as it can derive the goals and problems in a systematic way and completes the business-process-to-be in a systematic way. Elicitation of requirements from the to-be model and the filling in of the specification is still superficial and no systematic approach is provided. Erfurth and Kirchner [29] propose an elicitation technique based on CUTA cards and then generating BPMN and/or UML AD models from them. The approach is not eliciting requirements from models but is generating the models. They map the attributes of the cards to the elements of the notations. Their approach is interesting from the point of view of mapping the components of requirements to the elements of models. Despite the name of the paper written by Cox, Phalp, Bleistein and Verner [30] the derivation of the requirements plays a secondary role and the main focus is on connecting Problem Frames[2] to the derived requirements. This is rather useful in terms of selecting the appropriate development method for problem solving but not so much in terms of requirements elicitation. Although the paper provides guidelines to assist with the mapping of business process diagram elements to requirements, and to some extent can be used as an approach to elicit requirements from the BPMN models, when it comes to the step where a more detailed elicitation takes place, standard elicitation techniques like interviews, observation etc are suggested and no guidelines are provided.

[2] Problem Frames approach, developed by British software consultant Michael A. Jackson is an approach to software requirements analysis.[51]

For all the above cited works the business process model is the central artifact in the requirements elicitation, verification and specification process. They all deal with requirements elicitation on some level. However, how to elicit requirements from these models is patchy, not complete and focusing on only some aspects. The method provided in this thesis addresses this aspect by using the Domain Theory of Requirements Engineering to define the elements needed for a complete requirement and provides a systematic method for deriving the needed information from a BPMN model.

5.2 Eliciting Requirements from Use Cases and Scenarios

Use-cases and scenarios can be considered to be close enough to business process models as they describe how a business works. This is why the literature focusing on elicitation of requirements from them is studied. Maidens, Minochas, Mannings and Ryans' [31] research is aimed at improving the completeness of requirements by analyzing scenarios. This process uses the existing use case model as a starting point and derives new scenarios, taking into account situations, which have not yet been considered (alternative courses). It proposes a technique to validate the completeness of models and concentrates more on the alternative paths and failure conditions. Maiden and Robertson [32] apply RESCUE requirements process to discover requirements for an air traffic management system. Various elicitation techniques are used to discover the requirements of stakeholders (including the one described in [31]). The paper suggests a process and analyzes the effectiveness of different techniques. Berenbach's [33] approach concentrates on generating a hierarchy of requirements rather than on the requirements text itself and in the follow-up paper [34] more suggestions how to aid the organization of text based requirements with graphical modeling approaches is given. Firesmith [35] analyzes the pros and cons of user stories, scenarios and use cases and proposes an improvement how to create a more complete set of requirements using textual requirements. The approach concentrates more on quality attributes (e.g performance, security). They can be added to triggers, preconditions, required actions and post conditions. The method can be used as an addition to the method provided in this thesis and be used in future work for eliciting non-functional requirements. Cabral and Sampaio [36] have an idea how to automatically translate use cases written in a subset of English (CNL, Controlled Natural Language) into a specification in CSP process algebra. It is an approach that gives guidelines on requirements specification rather than on requirements elicitation. Daniels and Bahill [37] state that the best way to specify the requirements is to complement use cases and use case models with traditional shall-statement requirements. The paper is about

requirements specifications and very little about requirements elicitation. Probasco and Leffingwell's [38] work is also mainly about requirements specification and persistence.

The above cited works all use either use cases or scenarios as the central artifact in the requirements engineering process and they either concentrate on the improvement of a specific aspect of requirements specification (non-functional, alternative paths), classification of requirements already gathered, or give guidelines for very specific formal methods. However, no systematic method for eliciting requirements form a system level use-case or scenario, which was the aim of the method created in this thesis, can be found.

5.3 Eliciting Requirements from UML Diagrams

Meziane, Athanasakis and Ananiadou [39] propose a system that generates natural language specifications from UML class diagrams. The main focus is on automatically converting models into natural language specifications using WordNet and linguistic ontology. Pavlovski and Zou [24] propose a method how to formally verify informal UML Activity Diagrams, and they also point out the concerns and problems associated with natural-language requirements specifications.

While both of the works found use UML diagrams, they both concentrate on formal methods that are considered to be difficult to use, as the sources of elicitation of requirements can be of various levels of quality. The method provided in this thesis is able to elicit requirements also from models that lack consistency and completeness, which the formal methods cannot provide.

5.4 Eliciting Requirements from Goal Models

Maiden, Manning, Jones and Greenwood [40] propose an approach that indeed provides a systematic way to create textual specifications of requirements from i* models, but the same approach cannot be used efficiently on BPMN models as BPMN models do not describe the goals of the actors in as much detail as required for the approach. Lamsweerde and Willemet [41] propose a formal method how to create declarative specifications of goals, requirements and assumptions from scenarios. Letier and Lamsweerde [42] describe a method how to build operational software specifications out of higher-level goal formulations. The paper concentrates on software design rather than on requirements elicitation. Alrajeh, Russo and Uchitel [43] provide a method to semi-automatically infer operational requirements from goal models. Landtsheer and Lamsweerde [44] propose an approach that derives event-based specifications written in the SCR tabular language from operational specifications. Yu, Bois,

Dubois and Mylopoulos [45] propose a method for refining the requirements gathered with the Albert Requirements Specification Language and i* goal-based modeling.

The above cited works all provide formal methods to elicit requirements from goal models and are well structured and systematic approaches, however, the BPMs that are used as the source of information in this thesis, often do not provide detailed enough descriptions of goals and are hardly ever complemented with goal models required for these methods. The method proposed in this thesis, on the other hand, also provides steps to elicit the goals of the process under examination. Additionally Luis et al. [26] believe that goal-oriented approaches are not the best approaches to requirements engineering, as they do not pay enough attention to business concerns and business process reengineering.

5.5 Models as a Useful Artifact in the RE Process

There is a lot of research done about using the models or descriptions as a supporting tool to other elicitation techniques. Models are used mainly as communication helpers or are used for documenting and preserving knowledge during the elicitation process. The literature referred to in this chapter is not providing any concrete techniques to elicit requirements from models but is just confirming the importance of models in the elicitation process.

There are many papers about how process descriptions or models can be helpful and proven tools in the process of requirements elicitation. For example Demorörs, Gencel and Tarhan [46] say in their paper that BPM is a way to define business requirements and is useful for creating visibility and consensus among different stakeholders. Abeti, Ciancarini and Moretti [47] suggest to use SI*, UML and BPMN models to model organizational knowledge and use the knowledge in the RE process. Decreus & Poels [48] suggest a goal-oriented way to model the goals of the project and to generate BPMN models and use the models during the elicitation process. Flynn & Jazi [49] suggest requirements models to be built by users themselves and give direct guidelines how to approach the major problem of the user-developer culture gap. Gorton & Reiff-Marganiec [22] propose a way to specify requirements as a model. Zapata, Losada and González-Calderón [15] propose a method for using procedure manuals as a source for requirements elicitation, but the focus of the paper is more on converting natural language descriptions into formal language descriptions. Hickey and Davis [50] conducted a survey among requirement engineering experts, asking whether modeling as an elicitation technique is important and helpful. Most of the experts mentioned the critical role played by models, but in summary they saw modeling as a means to facilitate communication and organize the information gathered using other elicitation techniques.

6. Conclusions and Future Work

BPMN models are widely used to model the dynamic phenomena of organizations and are good sources of knowledge for understanding the domain and behavior of an organization. However, models are often too abstract, incomplete or inconsistent for requirements elicitation purposes. As such, there is a need for a systematic approach to elicit requirements from process models.

In this thesis, a structured method is presented that maps the components of a requirement to the elements of a process model captured with BPMN language. Furthermore, the method provides a set of questions that will ensure the elicitation of complete and consistent requirements when using process models as the source of information. The main idea of the method is to study each relevant Activity of a process model. The information found in the model, together with certain questions as detailed by the method, ensures the elicitation of complete and consistent requirements. For each Activity, a requirement specification template is populated with the information discovered in dialogue with the domain experts.

The method was validated on a real-life case study. The case study findings showed that the proposed method elicits more requirements as compared to the baseline method (used by the company) and that the quality of the set of requirements is better in terms of fewer "faulty" requirements. Furthermore, the method proposed in this thesis is more time efficient in terms of man-hours it took to elicit the requirements as compared to the original baseline elicitation method.

The method elicits functional requirements, and as such, one direction for future work is to extend the method to also accommodate elicitation of non-functional requirements from process models. Furthermore, as the number of requirements will rapidly grow with the increase of project complexity, a semi-automated tool to support the documenting and structuring of the requirement specifications is needed. The development of such tool is another venue for future work.

References

[1] C. R. Coulin, “A Situational Approach and Intelligent Tool for Collaborative Requirements Elicitation,” no. Toulouse III, 2007.

[2] J. A. Gougen and C. Linde, “Techniques for Requirements Elicitation.” 1992.

[3] A. M. Hickey and A. M. Davis, “A Unified Model of Requirements Elicitation,” vol. 20, no. 4, pp. 65–84, 2004.

[4] D. Zowghi and C. Coulin, “2 Requirements Elicitation : A Survey of Techniques , Approaches , and Tools,” 2005.

[5] S. Adam, N. Riegel, A. Gross, O. Uenalan, S. Darting, F. Iese, and F. Platz, “A Conceptual Foundation of Requirements Engineering for Business Information Systems,” pp. 91–106, 2012.

[6] Y. Wand and R. Weber, “Research Commentary: Information Systems and Conceptual Modeling?A Research Agenda,” *Inf. Syst. Res.*, vol. 13, no. 4, pp. 363–376, Dec. 2002.

[7] B. Nuseibeh and S. Easterbrook, “Requirements Engineering : A Roadmap,” *ICSE ’00 Proc. Conf. Futur. Softw. Eng.*, vol. 1, no. ACM New York, NY, USA ©2000, pp. 35–46, 2000.

[8] J. Li, R. Jeffery, K. H. Fung, L. Zhu, Q. Wang, H. Zhang, and X. Xu, “A Business Process-Driven Approach for Requirements Dependency Analysis Juan,” pp. 200–215, 2012.

[9] T. Dufresne and J. Martin, “Process Modeling for E-Business,” pp. 1–28, 2003.

[10] D. Birkmeier and S. Overhage, “Is BPMN Really First Choice in Joint Architecture Development ? An Empirical Study on the Usability of BPMN and UML Activity Diagrams for Business Users,” pp. 119–134, 2010.

[11] B. H. C. Cheng and J. M. Atlee, “Research Directions in Requirements Engineering,” 2007.

[12] A. Wever and N. Maiden, “Requirements Analysis : The Next Generation,” pp. 0–1, 2011.

[13] M. Dumas, M. La Rosa, J. Mendling, and H. a. Reijers, *Fundamentals of Business Process Management*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[14] Object Management Group Inc., “Business Process Model and Notation (BPMN),” 2011.

[15] C. M. Zapata, M. B. Losada, and G. González-calderón, “An approach for using procedure manuals as a source for requirements elicitation,” 2012.

[16] A. Sutcliffe and N. Maiden, “The Domain Theoryt for Requirements Engineering,” vol. 24, no. 3, pp. 174–196, 1998.

[17] A. Sutcliffe, G. Papamargaritis, and L. Zhao, “Comparing requirements analysis methods for developing reusable component libraries,” *J. Syst. Softw.*, vol. 79, no. 2, pp. 273–289, Feb. 2006.

[18] J. Naish and L. Zhao, “Towards a generalised framework for classifying and retrieving requirements patterns,” *2011 First Int. Work. Requir. Patterns*, pp. 42–51, Aug. 2011.

[19] Wikimedia Foundation Inc., “Business process.” [Online]. Available: http://en.wikipedia.org/wiki/Business_process. [Accessed: 10-Oct-2014].

[20] Wikimedia Foundation Inc., “Activity (UML).” [Online]. Available: http://en.wikipedia.org/wiki/Activity_(UML). [Accessed: 12-Nov-2014].

[21] T. Dunstan, “BPMN Explained, a guide to the Business Process Modeling Notation,” 2014.

[22] S. Gorton and S. Reiff-marganiec, “Towards a Task-Oriented , Policy-Driven Business Requirements Specification for Web Services.”

[23] I. Object Management Group, “Open Managment Group,” 2014. [Online]. Available: http://omg.org/. [Accessed: 21-Aug-2014].

[24] C. J. Pavlovski and J. Zou, “Non-Functional Requirements in Business Process Modeling,” vol. 79, 2008.

[25] A. Cockburn, “Writing Effective Use Cases,” 2000.

[26] J. Luis, D. Vara, J. Sánchez, and Ó. Pastor, “Business Process Modelling and Purpose Analysis for Requirements Analysis of Information Systems 1,” pp. 213–227, 2008.

[27] P. Runeson and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” *Empir. Softw. Eng.*, vol. 14, no. 2, pp. 131–164, Dec. 2008.

[28] IEEE Computer Society, *IEEE Recommended Practice for Software Requirements SpeciÞcations IEEE Recommended Practice for Software Requirements Specifications*, vol. 1998, no. October. 1998.

[29] I. Erfurth and K. Kirchner, “Requirements Elicitation with adapted CUTA Cards: First Experiences with Business Process Analysis,” *2010 15th IEEE Int. Conf. Eng. Complex Comput. Syst.*, no. 4, pp. 117–126, Mar. 2010.

[30] K. Cox, K. T. Phalp, S. J. Bleistein, and J. M. Verner, “Deriving requirements from process models via the problem frames approach,” *Inf. Softw. Technol.*, vol. 47, no. 5, pp. 319–337, Mar. 2005.

[31] N. A. M. Maiden, S. Minocha, K. Manning, and M. Ryan, “CREWS-SAVRE : Systematic Scenario Generation and Use 1 2 : The CREWS-SAVRE Software Tool,” pp. 1–9.

[32] N. Maiden and S. Robertson, “Developing use cases and scenarios in the requirements process,” pp. 559–570, 2005.

[33] B. Berenbach, “The automated extraction of requirements from UML models,” *J. Light. Technol.*, p. 287, 2003.

[34] B. a. Berenbach, “Comparison of UML and text based requirements engineering,” *Companion to 19th Annu. ACM SIGPLAN Conf. Object-oriented Program. Syst. Lang. Appl. - OOPSLA ’04*, p. 247, 2004.

[35] D. Firesmith, “Generating Complete, Unambiguous, and Verifiable Requirements from Stories, Scenarios, and Use Cases,” vol. 3, no. 10, pp. 27–39, 2004.

[36] G. da F. L. Cabral and A. C. A. Sampaio, “Formal Specification Generation from Requirement Documents,” 2006.

[37] J. Daniels and T. Bahill, “The hybrid process that combines traditional requirements and use cases,” *Syst. Eng.*, vol. 7, no. 4, pp. 303–319, 2004.

[38] L. Probasco and D. Leffingwell, “Combining Software Requirements Specifications with Use-Case Modeling,” Oct. 1996.

[39] F. Meziane, N. Athanasakis, and S. Ananiadou, “Generating Natural Language specifications from UML class diagrams,” *Requir. Eng.*, vol. 13, no. 1, pp. 1–18, Sep. 2007.

[40] N. a. M. Maiden, S. Manning, S. Jones, and J. Greenwood, “Generating requirements from systems models using patterns: a case study,” *Requir. Eng.*, vol. 10, no. 4, pp. 276–288, Oct. 2005.

[41] a. van Lamsweerde and L. Willemet, “Inferring declarative requirements specifications from operational scenarios,” *IEEE Trans. Softw. Eng.*, vol. 24, no. 12, pp. 1089–1114, 1998.

[42] E. Letier and A. van Lamsweerde, “Deriving operational software specifications from system goals,” *ACM SIGSOFT Softw. Eng. Notes*, vol. 27, no. 6, p. 119, Nov. 2002.

[43] D. Alrajeh, A. Russo, and S. Uchitel, “Inferring operational requirements from scenarios and goal models using inductive learning,” *Proc. 2006 Int. Work. Scenar. state Mach. Model. algorithms, tools - SCESM ’06*, p. 29, 2006.

[44] R. De Landtsheer, E. Letier, and A. Van Lamsweerde, “Deriving tabular event-based specifications from goal-oriented requirements models,” *Requir. Eng.*, vol. 9, no. 2, pp. 104–120, May 2004.

[45] E. Yu, P. Du Bois, E. Dubois, and J. Mylopoulos, “From Organization Models to System Requirements A ‘ Cooperating Agents ’ Approach 1 Introduction 2 Features of A LBERT and i *.”

[46] O. Demirörs, Ç. Gencel, and A. Tarhan, “Utilizing Business Process Models for Requirements Elicitation,” pp. 1–4, 2003.

[47] L. Abeti, P. Ciancarini, and R. Moretti, “Business Process Modeling for Organizational Knowledge Management,” pp. 301–311, 2008.

[48] K. Decreus and G. Poels, “A Goal-Oriented Requirements Engineering Method for Business Processes,” pp. 29–43, 2010.

[49] D. J. Flynn and M. D. Jazi, “Constructing user requirements: a social process for a social context,” *Inf. Syst. J.*, vol. 8, no. 1, pp. 53–83, Jan. 1998.

[50] A. M. Hickey and A. M. Davis, “Elicitation technique selection: how do experts do it?,” *J. Light. Technol.*, pp. 169–178, 2003.

Appendix 1 Case Study Design

Case study research is of flexible type, meaning that the design of the research might evolve or change during the execution of the study, but it is still important to set the objectives and methods of the case study beforehand to assure success of the research [27]. In this chapter the objective, research questions, hypothesis, selection strategy, the case, and the methods are introduced.

Objective, Research Questions, Hypothesis

First it is necessary to define the objective of the case study and to make a clear statement what is planned to achieve. The objective is a more generally formulated statement and is initially more like a focus point which evolves during the study [27]. In this thesis the objective is "*to test whether the application of the method created in this thesis improves the quality and/or quantity of the requirements"*.

In order to meet the objective stated above, more precisely formulated research questions should be created [27].

RQ1: Did the application of REM elicit more requirements than the previously used method?

RQ2: Did the application of REM result in better quality requirements than the previously used method?

To answer the research questions, the quantity and quality must first be clearly defined and the measures set. Measuring the quantity and the quality of the requirements is not straightforward, as the level of detail and the style of writing of specifications might differ remarkably depending on the method used by the subject case (a specification in one method might be 10 specifications in another). So the first step after the application of the method must be the conversion of the specifications created using the method to the same form and format as the specifications originally used in the subject case, or vice versa. The conversion must result in specifications of the same level of granularity. The elicited requirements must be relevant to the project (in scope).

Once the conversion of the specifications is done, it is possible to count the total number of requirements elicited by REM and by ICM. In a way it is possible to answer RQ1, but just measuring the total number of requirements elicited by both methods is not enough to make any serious assumptions as the improvement or decline in quantity or quality is relative to the total number of requirements existing in the project. E.g if 10 000 requirements exist in total,

an improvement by 10 is not significant. At the same time, if 100 requirements exist in total, an improvement by 10 is much more significant.

To find out what is the total number of requirements existing in the project, it is preferred that the project used for the case study is finished, the solution is implemented and has possibly been in use for some time, so that additional requirements that often arise only during the support phase are elicited. Also, not all requirements will necessarily be implemented as it might be irrational due to limited resources, but they are still valid requirements and might be captured during the elicitation process. So the Final Set of requirements in the project should, in addition to the implemented requirements, also contain the rejected and deferred requirements.

To answer RQ1, it is now possible to find out the total number of requirements elicited by both methods. If the total number of correct requirements gathered with the method described in the thesis is bigger than the total number of correct requirements gathered using the original method, it can be concluded that REM is able to elicit more requirements.

In addition, the coverage percentage (the percentage of all requirements in the Final Set covered by the requirements elicited by each method) must be found. The following tables ***Table 9*** Quantity Measure and ***Table 10*** Coverage of the Method summarize the quantity and coverage percentage measures needed to answer RQ1.

Table 9 Quantity Measure

Name	Quantity	Requirements of REM	Requirements of ICM
Abbreviation	Q	TOTTM	TOTOM
Description	Quantity coefficient	Number of requirements of REM	Number of requirements of ICM
Entity		List of requirements elicited by REM	List of requirements elicited by ICM
Attribute	TOTTM-TOTOM	No of requirements	No of requirements
Range	[-∞, ∞]	[0, ∞]	[0, ∞]

Table 10 Coverage of the Method

Name	Coverage percentage	Requirements of the method	Number of requirements in the

			Final Set
Abbreviation	COV	NOREQ	NOFIN
Description	Coverage	Number of requirements of REM or ICM	Number of requirements in the Final Set (including rejected and referred)
Entity		List of requirements elicited by the REM/ICM method	Final Set of requirements.
Attribute	NOREQ/NOFIN	No of requirements	No of requirements
Range	[0, 1]	[0, ∞]	[0, ∞]

In order to decide whether the method results in better quality requirements (RQ2), first it must be specified what is meant by quality. IEEE Computer Society [28] defines that a good quality requirements specification should be correct, unambiguous, complete, consistent, ranked for importance and stability, verifiable, modifiable, traceable. The four latter attributes are not important in the light of this thesis, as they come to importance in the later stages of the requirements engineering process. However, in order to measure the quality of the elicitation process, it is very important that the requirements elicited were correct, unambiguous, complete and consistent.

How correct and unambiguous the requirements are, can be assessed by measuring how many of the requirements elicited are not superfluous but are clear, understandable, unambiguous, and relevant. To do that, the requirements gathered must be verified and grouped in at least two groups (correct and superfluous) so that it was possible to count the number of superfluous requirements and to calculate the percentage of correct requirements in the total pack of elicited requirements. This must be done for both REM and ICM, which puts additional demands on the subject case as there should exist a list of all requirements verified as correct or superfluous (in case the list is missing, it is possible to compare the results with industry averages found in literature). The calculation of how correct and unambiguous the requirement are is summarized in the following **Table *11*** Correctness and Unambiguousness Measures.

Table 11 Correctness and Unambiguousness Measures

Name	Correctness and unambiguousness	No of correct requirements	Total no of requirements
Abbreviation	C	CR	TOT
Description	Correctness and unambiguousness coefficient	Number of requirements classified as correct (non-superfluous)	Number of total requirements elicited by the method.
Entity		List of requirements	List of requirements
Attribute	CR/TOT	No of correct	Total no
Range	[0, 1]	[0, ∞]	[0, ∞]

To measure whether the method resulted in a more complete and consistent set of requirement, it must be found out, how many of the requirements elicited during the project in total (including the development and support phase) did the method miss. More precisely, what is the coverage percentage of requirements that were not elicited by the method? The following **Table *12*** summarizes how the completeness and consistency is calculated.

Table 12 Completeness and Consistency Measures

Name	Completeness and consistency	No of missed requirements	Total no of requirements in the Final Set
Abbreviation	C	MR	F
Description	Completeness and consistency coefficient	Number of requirements found by the Final Set, but missed by the method	Number of requirements in the Final Set.
Entity		List of requirements	List of requirements
Attribute	MR/F	No of requirements	No of requirements

Range	[0, 1]	[0, ∞]	[0, ∞]

Additionally, the earlier the requirements are discovered in the project life-cycle, the more likely it is that the project is finished successfully and also better architectural decisions can be made. In order to measure this quality attribute, requirements elicited by the method should be compared to the requirements elicited in the development phase and after the go-live (in the support phase). If the new method was able to find requirements that in real life were discovered only in the development or support phase, it is also possible to conclude that it is of better quality. The last measure sets some demands to the subject case as it should be possible to filter out requirements discovered in the development and support stages.

The hypothesis for the study is that *"application of the method improves the quantity and quality of the requirements elicited"*.

In order to give answers to the research question above and to test the hypothesis, a subject case was sought where the requirements elicitation for IS was completed and possibly the IS system was already implemented. The IS to-be was process-oriented, the process was nontrivial and a BPMN model of the process existed (possibly created by the customer or by the analyst conducting the elicitation process). The elicitation method used originally was well defined and used in various projects. Requirements were documented, so that it was possible to compare the initial method to the Final Set of requirements.

Case and Subject Selection

The organizational setting of our case is a company that manufactures branded electric motors and motor components for European customers. The subcontracting includes, for example, machining of housings, shaft machining, coil manufacturing and final assembly. The company is changing the entire enterprise resource planning software (ERP) that involves all departments (sales, production, warehouse, quality assurance, payroll, finance, etc.). The project is ongoing, but many parts of the solution are already implemented (as of November 2014).

The quality assurance sub-process was chosen as the subject for the case study. Quality assurance is a very important process for the company as there are very strict rules on defects and it is by no means a trivial process. Quality assurance is a supporting process to the production process. The process is unique to the customer and therefore the supporting IS solution is custom made for the customer. The system is built using the ERP platform (other processes involved ready-made functionality of the ERP system).

The process had been modeled in BPMN beforehand by the analyst of the solution developer. The solution developer is a company involved in the ERP solutions for over 9 years and is well experienced in the field. They have their own methods for requirements elicitation that have been evolved and used a number of times for many years and in a number of projects. To model a system in BPMN, was relatively new to the developer and the analyst. The project under the inspection of the case study had been completed 2 months before the case study and the solution was up, running and used daily by the customer.

The developer had documented all the requirements gathered during the analytical phase of the project and also all additional requirements that had evolved during the project development and post-project (support phase). That gave a possibility to compare the results of REM to ICM and to the Final Set.

In order to apply the method, the developer's analyst was used as the domain expert on the case. The analyst was the one that conducted the original requirements elicitation of the project and was also involved in the elicitation, documentation and validation of additional needs. The analyst is an experienced requirement engineer (more than 12 years in the field) and was involved in the project from the beginning to the end. The analyst was chosen also to give expert opinion and critique on the created method and to help to compare the results.

Case Study Data Collection Procedures

The case study is divided into four parts. The first part introduces the method to the analyst and involves preparations for the next part. During the preparation, the model is examined by the author of the thesis and preliminary requirements analysis is conducted without the analyst and without the application of the questions of the method. The BPMN model is examined and the suggestions of the method are followed. During the examination, requirement specifications are created in the form of spreadsheets. Unclear or illogical things are written down to be addressed in the next session.

The second part is held in the form of workshops (interviews) with the analyst. During the workshops, the method is applied and specifications are updated with the gathered information.

The third part is about converting the specifications created to the form and format used by the developer. This is done using the help of the analyst, so that the new specifications would be as close in style as possible to the specifications created in real life. After conversion, verification of the requirements against the Final Set of requirements is done. For verification,

it is checked whether the requirement exists in the Final Set. If the requirement is missing from the Final Set, the validity of the requirement is discussed with the analyst. If the requirement is considered to be adequate, it will be classified as correct, if not, it will be classified as superfluous. The classification will be marked in a separate field in the specification. Also, a summary of the results and a calculation of the measures is done. The summary has to be done for 3 lists of requirements: REM, ICM and the Final Set. All the results will be saved in a separate spreadsheet.

The forth part is about interpretation of the results, comments and critique. To analyze the results, an additional session with the analyst must be held in order to discuss the validity, and threats to the validity, exchange opinions and discuss improvement suggestions. All opinions and critique will be documented. The time spent on interviews, documentation and other activities will be registered.

License

Non-exclusive license to reproduce thesis and make thesis public

I, **Sander Valvas** (date of birth: 07.04.1975),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:
   1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and
   1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright, of my thesis

Requirements elicitation from BPMN models, supervised by Fredrik P. Milani,

2. I am aware of the fact that the author retains these rights.
3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **26.02.2015**