Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

MARINA TRKMAN

Metoda za uporabo modelov poslovnih procesov pri zajemu zahtev uporabniških zgodb

DOKTORSKA DISERTACIJA

Mentor:
prof. dr. Marjan Krisper

Somentor:
prof. dr. Jan Mendling

Ljubljana, 2016

University of Ljubljana

Faculty of computer and information science

MARINA TRKMAN

A method for using business process models in the elicitation of user stories

DOCTORAL THESIS

Advisor:
prof. dr. Marjan Krisper

Co-advisor:
prof. dr. Jan Mendling

Ljubljana, 2016

Povzetek

Agilni razvojni projekti za modeliranje uporabniških zahtev uporabljajo uporabniške zgodbe. Za njihovo pridobivanje obstaja veliko načinov, vendar jih le malo izhaja iz obstoječe dokumentacije. Drugi problem je razumevanje konteksta posamezne uporabniške zgodbe, kar zahteva razumevanje izvedbenih in integracijskih odvisnosti med uporabniškimi zgodbami. Trenutno ne obstajajo pristopi, ki bi s pomočjo obstoječe dokumentacije naročnika prispevali k tovrstnemu razumevanju konteksta. V tej disertaciji predlagamo BuPUS metodo, ki 1) omogoča pridobivanje uporabniških zgodb, in 2) podpira boljše razumevanje izvedbenih in integracijskih odvisnosti med uporabniškimi zgodbami iz obstoječih modelov poslovnih procesov. Metoda poveže uporabniške zgodbe z ustreznimi aktivnostmi, ki so del BPMN modela, oziroma z ustreznimi dogodki, ki so del modela primera uporabe. Definirali smo tri nivoje povezljivosti med uporabniškimi zgodbami in aktivnostmi/dogodki: uporabniška zgodba je bolj abstraktna, približno enako velika ali bolj podrobna od aktivnosti/dogodka, s katerim je povezana. V eksperimentih smo ovrednotili vse tri nivoje. Pripravili smo dva eksperimenta. Za ugotavljanje razumevanja integracijskih in izvedbenih odvisnosti med uporabniškimi zgodbami smo pripravili vprašalnik z več testi: test razumevanja, test reševanja problemov in test spomina. Učinkovitosti pridobivanja uporabniških zgodb iz dveh različnih modelov poslovnih procesov pa smo ocenjevali s štetjem pravilno ugotovljenih uporabniških zgodb. Statistična analiza dveh raziskav je potrdila osem od enajstih hipotez. Prva raziskava je potrdila, da je razumevanje izvedbenih in integracijskih odvisnosti med uporabniškimi zgodbami večje, v kolikor k seznamu uporabniških zgodb dodamo modele poslovnih procesov. V drugi raziskavi smo primerjali različne modele poslovnih procesov in ugotovili sledeče. Uporaba BPMN modelov poslovnih procesov je bolj učinkovita pri razumevanju izvedbenih odvisnosti kot uporaba modelov primerov uporabe. Pri razumevanju integracijskih odvisnosti ni opaziti statističnih razlik med modeloma. Prav tako jih ni opaziti pri pridobivanju uporabniških zgodb.

Ključne besede: uporabniške zgodbe, integracijska odvisnost, izvedbena odvisnost, pridobivanje uporabniških zgodb, eksperiment

Abstract

Agile software development projects often manage user requirements with models that are called user stories. Approaches for eliciting user stories from customer's existing documentation are missing. Furthermore, proper understanding of user story's context requires an understanding of execution-order and integration dependencies among user stories, which are also missing. In this thesis we propose so-called BuPUS method which 1) facilitates elicitation of user stories from existing business process models, and 2) supports better understanding of execution-order and integration dependencies among user stories from customer's existing documentation. The method associates user stories with corresponding BPMN's activity elements, or with corresponding text-written use case model's events. We defined three levels of association granularity: a user story can be more abstract, approximately equal to, or more detailed than its associated business process model's event/activity element. In our experiments we evaluated these three levels. We run two experiments. We applied comprehension, problem-solving and recall tasks to evaluate the hypotheses which refer to understanding of the dependencies. On the other hand, we measured user story elicitation's effectiveness with counting correctly defined user stories. The statistical results provide support for eight out of eleven of the hypotheses. The results of our first experiment show, that understanding of the execution-order and integration dependencies among user stories, when associated business process models are available, is significantly greater. In our second experiment, we compared text-written use case model and BPMN model. There appears to be greater understanding of the execution-order dependencies when using BPMN models, while there were no significant differences in understanding integration dependencies. Similarly, for the elicitation of user stories there are no significant differences when using either of the mentioned models.

Keywords: user story, integration dependency, execution-order dependency, elicitation of user stories, experiment

IZJAVA O AVTORSTVU

Spodaj podpisana Marina Trkman z vpisno številko 63080457 sem avtorica doktorske disertacije z naslovom "Metoda za uporabo modelov poslovnih procesov pri zajemu zahtev uporabniških zgodb".

S svojim podpisom zagotavljam, da:

- sem doktorsko disertacijo izdelala samostojno pod vodstvom mentorja profesorja Marjana Krisperja in somentorja profesorja Jana Mendlinga;
- so elektronska oblika doktorske disertacije, naslov (slov., angl.) povzetek (slov., angl.) ter ključne beside (slov., angl.) identični s tiskano obliko doktorske disertacije;

Soglašam z javno objavo elektronske oblike doktorske disertacije v zbirki "Dela FRI".

V Ljubljani, dne 30.5.2016 Podpis avtorice:_____

ACKNOWLEDGEMENTS

On this page, I would like to thank those who supported me in writing my doctoral thesis. It took almost eight years to prepare it. Four of those eight years were spent on four maternity leaves. Two years were taken to complete the exams and move deeper into the selected doctoral topic. About one year was spent changing the topic, twice. And, finally, there was one big fat year of sticking with the last topic and taking it through to fruition.

My maternity leaves created many interruptions, blurring the path to finishing my PhD thesis. I appreciate my mentor Prof. Marjan Krisper for sticking by me when leaving would have been the easiest course. I can imagine how stressful the interruptions were for him. Also, I would like to thank him for accepting Prof. Jan Mendling as my co-mentor. Jan introduced a lot of stability into our relationship and made us complete. Further, in the last year my mentor retired. Prof. Rok Rupnik became involved as my third mentor. I wish to thank Rok for approving my "wild" actions and supporting them financially. Working with this team enabled me to produce deliverables such as workbooks, data from experiments, a paper, and finally a PhD thesis.

There were many lecturers who accepted me in their classes so I could complete the experiments. I thank Selim Erol, Reinhard Fisher, Johann-Rudolf Göpfrich, Monika Malinova, Nikolai Neumayer, Natascha Pfleger, Andreas Rogge-Solti, and Peter Skamrada who all teach the Business Information Systems II course at Vienna University of Economics and Business in Austria.

I would also like to my children for their patience when their mummy had to go to the basement office to work and not play with them. I recall how my Marcel (then 4 years) missed me but despite his age was able to communicate that to me in a grown-up way. I also remember how my daughter Julija (then 6 years) was genuinely happy for me when my paper was accepted. I would like to thank my mother-in-law Slavica for enabling me to work on my thesis by babysitting my children in her typical joyful manner. And, last, I would like to thank by husband Peter for tolerating my emotional rollercoasters and, again, my absences. At the time of carrying out my second experiment I was in my final stage of pregnancy. Peter got involved by becoming the best assistant in the world. His patience is endless, and so is my love for him.

TABLE OF CONTENT

TABLE OF FIGURES

TABLE OF TABLES

1 Introduction

This chapter gives an introduction to the doctoral thesis. The term approach is used to generally address models, methods, methodologies, and techniques [1]. In Section 1.1 we start with the general motivation and clarify the importance of business process models in agile development with user stories. Afterwards, in Section 1.2 we present the research contributions. In Section 1.3 we continue by introducing the methodological background. Finally, Section 1.4 gives an outlook on the thesis structure.

1.1 Motivation

The software development industry employs many different approaches for developing software [2]. A major challenge most of the approaches face with is how to effectively identify and manage requirements [3]. There is a big difference between the old waterfall and newer agile development approaches. Among others, they differ in how they deal with requirements analysis. In waterfall's requirements analysis, we assume there is a final set of detailed requirements that must be determined before the development starts. The agile development's requirements analysis acknowledges the impossibility of writing down all of the requirements at once, although it still attempts to write those that can be written upfront but at a higher-level of detail [3]. Agile requirements analysis is perceived as a highly evolutionary and collaborative process in which developers and project stakeholders actively work together on a just-in-time basis to understand the domain, identify the requirements, and estimate and prioritize them [4].

User stories are the most commonly used requirements model in agile development projects [5-7]. The goal of user stories is not to document every detail about a desired feature, but to write down a few short sentences that will remind developers and customers to hold future conversations [3]. User stories prioritize communication through the whole development process [3] and are there to bridge the developer-customer communication gap [2, 8]. It is important that the development team understands the context in which user stories support the business domain so they can provide better estimates of developmental resources (e.g. time). Therefore, in early days of a development project, where the preparations for agile development take place, agile requirements analysis focuses on eliciting user stories which are just good enough for providing good estimates [4] and for understanding elicited user stories in the context of the customer's business domain [2, 9].

Eliciting user stories requires intense communication that needs to overcome any cultural gap or semantic differences that may exist between users and the development team [10]. User stories are often written by the customer side (namely, customer, users, managers) which possesses business domain knowledge. The customer side is responsible for delivering as many user stories as early as possible and ensuring all user roles are appropriately represented [2, 3]. The choice of a user story elicitation approach depends on the situation and available resources [11, 12]. The most commonly used elicitation approach is an interview [3, 13]. Cohn [3] proposes a story-writing workshop approach which combines elements of brainstorming and low-fidelity prototyping. Ambler [7] suggests starting elicitation with existing documentation and loosely describes how to elicit user stories from a text-written use case model. There are also many other elicitation approaches such as questionnaires, observations [3] and focus groups [2].

Understanding of the elicited user stories needs to be accomplished by all stakeholders (from the customer and development team side) in the early days of a development project. But, since elicitation involves different project stakeholders it is communication-intensive [10] and as such it hinders the understanding. The communication barrier between representatives of the customer side and development team (namely, software engineers, system testers, system maintainers) can be caused by a gap in business domain knowledge of the two sides [14]. The development team members needs to understand customer side's business in order to properly support the business with a new application. The understanding of individual user story content is very important, nevertheless a single user story does not convey the entire business which is to be supported by a new application but only part of it. From a business perspective that makes the single user story potentially depended on other user stories [15]. In this context, insights into dependencies among user stories are crucial for project success [15-21]. Being unaware of the dependencies can lead to missing information regarding the context of a project, namely, its domain [2, 22]. Martakis and Daneva [17] emphasized the importance of execution order and integration dependencies. *Integration dependencies* present information about how certain user stories require other user stories to be previously developed. On the other hand, e*xecution order dependencies* present information about how the completion of a user story directly impacts another. Strode [23] refers to these two types of requirements as activity dependencies and technical dependencies.

Recent literature suggests several solutions for building up an explicit knowledge base for a project's integration dependencies. For example, Lin et al. [16] propose a method

for decomposing complex processes into phased goals and grouping low-level user stories with high-level goals. Clarke and Kautz [24] have developed a method for factoring epics into user stories. Leffingwell [2] proposes a model which decomposes a project's requirements in a tree view. Liskin [25] concludes that managing the granularity levels of user stories needs to be further investigated. Recent literature also discusses how to create an explicit knowledge base capturing the execution order dependencies among user stories. Milicic et al. [26] propose a user-centric method which organizes user stories along scenarios and users. Leffingwell [2] suggests that the sequence of execution should be represented by use case diagrams and use case specification documents. Patton [27] proposes a story-mapping technique for breaking big stories down while still maintaining the big picture of a project. The overall purpose of the mentioned solutions is to create a conceptual model that complements the list of user stories with information about the integration and execution order dependencies.

Conceptual models are key artifacts for understanding an application domain and its requirements [28]. Since most agile development projects aim to support business needs [29], it is essential to understand the processes of the business domain the user stories are intended to support [30, 31]. A process is a sequence (or a flow) of activities in an organization with the objective of carrying out work [32]. Processes can be represented as business process models. System development often makes use of process models [33, 34]. Therefore, the business process models of an organization have the potential to be reused as a means for eliciting user stories and as a complementary model to explicate the integration and execution order dependencies among user stories; however, no prior research has investigated this potential.

1.2 Research contributions

Our initial research question is how available business process models can be exploited for agile software development with user stories. The goal of this thesis is to propose and experimentally validate a novel approach called BuPUS (Business Process User Story) method (also referred to only as BuPUS) which enables the elicitation of user stories and understanding of the execution order and integration dependencies among user stories. The proposed method associates user stories with a business process model in such a way that user stories can be elicited, and execution order and integration dependencies understood.

With regard to measuring the understanding of user stories we were inspired by the work of [34-37]. Similarly to them, we build on the theoretical framework [38] for empirically evaluating conceptual modeling techniques. We measure the understanding of a user story with problem-solving questions [35, 39] which address the execution order and integration dependencies.

Our research involves two sequential phases. In the first phase, we argue that associated business process models are a beneficial complementary model for understanding the dependencies. We build upon the following propositions:

Proposition 1: Understanding of the execution order dependencies among user stories is greater when reading the BPMN material in comparison to reading the list of user stories.

Proposition 2: Understanding of the integration dependencies among user stories is greater when reading the BPMN material in comparison to reading the list of user stories.

The results show a greater understanding of the execution order and integration dependencies of a user story when the associated business process model is available in comparison to not having it.

In the second phase, we argue that visual business process models are a more beneficial for understanding the dependencies compared to textual business process models. We additionally argue that visual business process models are more beneficial for user story elicitation than textual business process models. We measured the effectiveness of two user story elicitation techniques where one is based on a BPMN model and another on informational equivalent text-written use case model. We developed the following two propositions:

Proposition 3: Understanding of the integration dependencies of user stories is greater when reading the (visual) BPMN material in comparison to reading the text-written use case material.

Proposition 4: Understanding of the execution order dependencies of user stories is greater when reading the (visual) BPMN material in comparison to reading the text-written use case material.

Proposition 5: Elicitation of user stories is more effective when using the (visual) BPMN material in comparison to the text-written use case material.

The results show that BPMN material supports greater understanding of the execution order, while text-written use case model supports greater understanding of the integration

dependencies. Furthermore, the results also show that the elicitation is more effective when having a BPMN material.

1.3 Methodological background of our research

Design science (methodology) promotes design and investigation of artifacts, such as constructs, models, methods, etc. [40], in context of existing knowledge disciplines [41]. In this thesis we refer to the context of information systems and software engineering sciences. These sciences aim at existential generalizations and make realistic assumptions about the artifact [41]. The main artifact in this thesis is a method for using business process models in the elicitation of user stories, which we named Business Process User story (BuPUS) method. The method consists of multiple (sub-) arifacts techniques, models and processes. We clarify and illustrate the contributions of these artifacts by discussing them in the context of the so-called design science research guidelines proposed by Hevner et al. [42]:

- *Problem relevance.* The relevance of BuPUS is determined by its value for the agile community. The need for the research conducted in this thesis can be derived from the increasing adoption of process models by many organizations. The BuPUS supports an agile project by proposing novel user story elicitation techniques and techniques for understanding the execution order and integration dependencies among user stories.
- *Design an artifact*. The goal of design science research in information systems is to create artifacts that address important organizational problems [42]. In this thesis, we define the constructs, models (which present relations between the constructs), techniques (which use the constructs), and processes (which explain how to use techniques and the main BuPUS method).
- *Research contribution.* Design science research must provide a contribution to the body of knowledge [42]. The research contributions of this thesis include a novel conceptualization of user story constructs that we used for the integration of user story concepts with (more general) business process model concepts. This integration was then used for two more specific integrations: the integration of user story concepts with (visual) BPMN constructs, and the integration of user story concepts with text-written use case constructs. These two specific integrations were used in the creation of: 1) two novel techniques for eliciting user stories from visual and textual business process models, 2) for two novel techniques for tracing execution order dependencies by using BPMN model and

text-written use case model, and 3) for two novel techniques for tracing integration dependencies by (again) using BPMN model and text-written use case model.

- *Design evaluation*. As suggested by [42], we demonstrate the utility (also usefulness), quality, and efficacy (the ability to produce a desired or intended result) of the proposed BuPUS by conducting two experiments. In the first experiment, we focused on investigating the utility of the BuPUS by comparing two configurations, namely, a list of user stories with and without associated business process models. We evaluated if there was a positive impact of having an additional corresponding associated business process model next to the list of user stories when trying to understand the execution order and integration dependencies among user stories. We measured understanding with problem-solving questions as proposed by Mayer [39]. After investigating a positive impact, we conducted the second experiment in which we focused on the BuPUS quality by comparing another two configurations, namely, a list of user stories with BPMN models and a list of user stories with text-written use case models. In addition, we considered to evaluate the efficacy of the BuPUS when measuring subjects' performance in user story elicitation from a BPMN model and from a text-written use case model.
- *Research rigor***.** The rigor of constructing information technology artifacts is one of the things that distinguishes information systems as design science from the practice of building information technology artifacts [43]. Rigor refers to the general way in which research is conducted. In this thesis, we paid a lot of attention to the description of the BuPUS with its grammar. We adopted situational method engineering approach, namely Process Configuration Approach proposed by Bajec et al. [44], to clearly state the point of departure, and the granularity of proposed method constructs used to create project-specific methods. As suggested by Saghafi and Wand [28], we used ontological concepts to clarify the meaning of the constructs. More specifically, we used Bunge's ontology which is the most widely used ontology in the analysis and design of information systems [35] to clarify method's constructs. We derived more detailed constructs from them. For defining most of the detailed constructs we adopted existing definitions of BPMN specifications proposed by Object Management Group [45] and use case specifications proposed by Rational Unified Process.

Furthermore, the essence of information systems as design science lies in the scientific evaluation of artifacts [43]. Our research design is structured as suggested by Gemino and Wand's framework for the empirical evaluation of conceptual modeling techniques [38]. We run two experiments:

- In first experiment, we are comparing two different grammars. One is used to support user stories and the other one to support BPMN material, so we have an inter-grammar comparison. Rules within the grammar were not varied. The medium of presentation was paper using text and graphics. The task was to interpret the given material. The focus was placed on the evaluation of the cognitive model created by viewing the given material. Therefore, the product of script interpretation that we measure is understanding of a user story. Effectiveness of the product was measured with comprehension, problem-solving and recall tests.
- In the second experiment, we continue evaluating the BuPUS by comparing another two different grammars, where one is used to support BPMN material and the other one to support text-written use case material. Again, we have an inter-grammar comparison again. Same as in first experiment, the rules within the grammar were not varied, and the presentation was paper using text and graphics. There were two tasks. The first task was to create user story clauses where the focus was placed on the evaluation of correct user stories. Therefore, the product of this task is creation of the list of user stories. We measured its effectiveness by counting correct, wrong and forgotten user stories. The second task was to interpret the given material so the focus was (as in first experiment) placed on the evaluation of the cognitive model created by viewing the given material. The product of this task is understanding of a user story. Its effectiveness was measured with comprehension, problem-solving and recall tests.

With both experiments we paid a lot of attention to the rigor by conducting multiple pilots prior to operationalizing the experiments. Also, we gave a lot of attention to the discussion of validity of the experiments.

- *Design as a search process*. The main issue of many design problems in information system research is the size and complexity of the solution space [42]. This means that it is often not possible to describe all the means, goals, and laws. As proposed by [46], in this thesis we did not consider all possible kinds of

business process models on the input of the proposed BuPUS (e.g. business process model's notation, granularity of activities). We build BuPUS gradually by adding the first project-specific method focused on BPMN model to the base method, and by adding the second project-specific method focused on text-written use case model to the first project-specific method. We specify exact characteristics of BPMN and text-written use case models that we work with when describing BuPUS constructs and when describing material of the two experiments.

- *Communication of research***.** The results of design science research must be presented. The goal is to provide practitioners with the possibility to take advantage of the solution and to enable researchers to build a cumulative knowledge base [42]. Some results of our research conducted in the context of this thesis have been published in the journal Information and Software Technology [47], in the Slovenian national journal Uporabna informatika [48], and at the international Information Society multiconference [49].

The discussion of the design science research guidelines regarding the work presented in this thesis shows that the contributions meet internationally established research standards.

1.4 Structure

This thesis is subdivided into six chapters:

1 Introduction: In this chapter, we motivated the research topic of complementing user stories with business process models, highlight the research contributions, and discussed the methodological background.

2 Background: This chapter starts by presenting the change in requirements analysis which has occurred over the years. The changes were brought by new approaches to software development. Nowadays, agile software development approaches have attracted the attention of many software development teams. Two of the most popular software development approaches use so-called user stories as a primary requirements model. The literature review focuses on presenting previous approaches to eliciting user stories, and earlier approaches to enabling understanding of the execution order and integration dependencies among user stories. The chapter finishes by suggesting the use of as business process model

as a means for eliciting user stories, and as a conceptual model for enabling understanding of the dependencies among user stories.

3 Proposed method: Business Process User Story (BuPUS) method. This chapter presents the approach of creating BuPUS method. By referring to ontological concepts we presented method's constructs. The constructs were used to integrate user story constructs with both BPMN and text-written use case constructs. When the integration is done several techniques are presented, namely for elicitation of user stories, and tracing execution order and integration dependencies among user stories. At the end we show when to use the techniques within BuPUS process.

4 Hypotheses. This chapter provides a theoretical background to the creation of the hypotheses. Firstly, we derive five propositions from our research question. Secondly, we ground our expectations by discussing special contiguity principle, extraneous processing, expressiveness of grammars, and information equivalence of material. Finally, we derive eleven hypothesis from the five propositions.

5 Research methodology. This chapter explains how we prepared, executed, analyzed and validated two experiments. The preparations, which included multiple pilot experiments, were performed at University of Ljubljana. Afterwards, the experiments were executed at Vienna University of Economics and Business in Austria. For the analysis of collected data from both experiments we used Mann-Whitney test. Finally, we discuss internal, external, construct and conclusion validity of results.

6 Conclusion: This chapter summarizes the results of the analysis. Afterwards, it discusses contributions to both practice and research. Finally, it proposes several ideas for future work.

2 Background

This chapter provides background on existing body of knowledge, which we need to present the problems that this thesis deals with, and propose the solution. Section 2.1 makes an overview of software development approaches, and sets the focus on problems of requirement analysis. Further, Section 2.2 focuses on one particular requirements model – a user story and presents problems in user story elicitation and understanding. Section 2.3 presents visual and textual business process models as a potential tool to support the elicitation and understanding. Finally, Section 2.4 summarizes Chapter 2.

2.1 Requirement analysis

2.1.1 Overview of software development approaches

The inception of the software industry was in the 1950s and 1960s [2]. The pre-methodology era in the 1960s and 1970s included few formal approaches and lacked controls, standards, and training [50]. Soon the need to better predict and control software projects emerged. In the 1970s, an important approach to developing information systems called the Systems Development Life Cycle (SDLC) was proposed [2]. The lifecycle presents a series of chronological phases to plan and manage the systems development process [51]. According to Shelly et al.'s book [51] there are five phases: systems planning, systems analyses, systems design

Systems planning is the first phase of the SDLC. The phase is performed by an analyst who aims to understand the business domain's reasons/justifications for an application proposal. The deliverable that is produced is a preliminary investigation report which includes a feasibility study. While in Shelly et al.'s book the feasibility study is part of systems planning, in Avison et al.'s book it is the first, independent phase of the SDLC [1]. The feasibility study looks at the organization's previous information system and its requirements: the requirements it was intended to meet, problems in satisfying those requirements, functionalities that have been missed, and briefly an investigation of an alternative solution. A requirement model is a high-level blueprint for a software product. These models must be read by different stakeholders [31]. The requirements are elicited (gathered, created) from the following stakeholders [52]: the customer (who will be purchasing the system), the users (who will actually operate with the software), managers (who will not be directly using the software but are interested in eliciting requirements which address the achievement of business objectives), software engineers (designers who have to translate requirements into software designs and code), system testers (who are responsible for ensuring that the designed system is reliable and meets the performance criteria), and system maintainers (who will make sure the software is running and will modify it if needed). Further, the requirements can also be elicited from external constraints such as laws, regulations, a certificate's conditions, etc. In the continuation, we address stakeholders as falling within one of two groups: 1) the customer organization side (the customer, the users, and the managers); or 2) the development team side (software engineers and the system testers, system maintainers).

When proposing a new application, a description of the costs and benefits of developing and operating is needed [53]. The proposed system must be feasible in multiple ways: a)

legal feasibility means that the new application does not infringe any national or company law [1]; b) organizational feasibility means that it is acceptable for the organization [1]; c) social feasibility means it is acceptable for the users [1, 51]; d) technical feasibility means that the technology is available and there is enough expertise to use it [1, 51]; e) economic feasibility means the new application is financially affordable and justifiable [1, 51]; f) schedule feasibility means that the new application can be implemented within an acceptable timeframe [51]; and g) operational feasibility which means that the application will be used effectively after it has been developed [51].

Systems investigation is the second independent phase of the SDLC in Avison et al.'s book [1] while in Shelly et al.'s book it is again part of the systems planning phase. In this phase, the focus is placed on requirements of the organization's existing information system (if there is one), and on requirements of a new application which will become a new part of the organization's information system. The information is collected through questionnaires, observations, interviewing, sampling (doing statistics), and viewing other available documentation. The systems investigation ends with a presentation of the results and recommendations to management.

The systems analysis phase is the second phase of SDLC. It starts when the management agrees with the recommendations proposed in the preliminary investigation report. This phase continues the work of the first phase and focuses on requirements modeling, enterprise modeling, and analysis of development strategies [51]. There are three main tasks that need to be done. Firstly, during requirements modeling, the analyst must identify and describe all the requirements. For each requirement, information about who, what, where, when, how, and why must be noted. For example, the Zachman Framework for Enterprise Architecture is a model that asks these traditional questions in a system development context so it can help with the systematic gathering of information about each requirement. Many existing enterprise architecture frameworks have been inspired by it, such as the Extended Enterprise Architecture Framework, Enterprise Architecture Planning, the Federal Enterprise Architecture Framework, and the Integrated Architecture Framework [54]. Secondly, enterprise modeling reveals how the system transforms data into useful information. The deliverable is a logical model which shows what the system must do. This model can be presented in various ways, for example, as an entity relationship diagram (ERD) or a data flow diagram (DFD). ERD is a model that shows the logical relationships among system entities and is also one of the most popular database design models [55], while DFD shows how the system transforms input data

into useful information. Thirdly, analysis of development strategies includes discussion of software trends, acquisition and development strategies, traditional versus web-based development, outsourcing versus in-house development, the system requirements document, prototyping, codes, and preparing for the transition to a system design phase. Therefore, the deliverables of the analysis phase are: a list of requirements, logical models, and software development strategies.

The systems design phase is the third phase. It delivers a physical model, user interfaces, and system architecture, which all rely on the logical model from the analysis phase. It is very important that the analyst obtains user feedback and involvement in all design decisions. In order to deliver a physical model (which describes how the system will be conducted), the system analysis leads to the design of a data structure as either a file-oriented system or as database management system (DBMS). Unlike a file-oriented system, DBMS avoids data redundancy and supports a real-time environment [56]. DBMS is a collection of models, tools, features, and interfaces that enables users to add, update, manage, access, and analyze the contents of a database. An example model is an ERD diagram which depicts relationships among system entities.

A user interface is a key design element which describes how users interact with a computer system. Therefore, analysts must deliver effective user interfaces with suitable inputs and outputs. A user interface consists of all the hardware, software, screens, menus, functions, and features that affect two-way communications between a user and a computer. There are many ways to document interface designs. For example, it can be done by applying basic user-centered design principles which are applied when planning, designing, and delivering a successful user interface [57].

System architecture defines hardware, software, data, procedures, and people. The architecture is affected by seven issues [51]: 1) will the project engage an enterprise resource planning (ERP) system; 2) what will have a major impact on the initial and total cost of ownership; 3) what kind of scalability is desired; 4) is there going to be any web integration; 5) is there any legacy system involved; 6) what are security issues and how to handle them; and 7) how will the data be processed?

Systems implementation is the fifth phase. Application development is a process of coding modules that are the building blocks of an application. Further, coding is a process of turning program logic into specific instructions that a computer can execute. Next, the code needs to be tested. Unit, and, consequently, integration testing have to be performed.

When all the tests are acceptable, there is a need to write the documentation which is essential for successful system operation and maintenance. That documentation explains the application and helps people interact with it. The documentation includes program documentation, system documentation, operations documentation, and user documentation. When all the documentation is ready, the users, managers, and IT staff members need to be trained. Finally, data conversion must to be performed, which means that existing data are loaded into the new application/system.

Systems operation and support is the sixth phase. The developed application is in operation at this point. It has to be able to meet user expectations and provide support for business objectives in order to be in operation for a longer period of time. It must also be maintained and improved continuously to meet changing business demands. Since users constantly require assistance, further user trainings can be organized.

By observing SDLC phases we can see that the SDLC is highly prescriptive. The phases are performed in chronological order and they should never repeat, thereby resembling a waterfall. Consequently, we refer to the SDLC as the waterfall approach. The waterfall approach assumes that a full set of detailed requirements can be determined up front [2]. But development tasks and durations are too unpredictable to accurately plan more than a few weeks in advance [9].Consequently, the waterfall's assumption has been found to be a root cause of project failure because any assumption that there will be little change to requirements once they have been documented in the analysis phase is fundamentally flawed [2]. Further, the literature is full of criticisms of why and how waterfall development approach constantly fails to provide solutions to the problems of developing robust and flexible information systems [58]. Bad requirement analysis was a frequent cause of projects failing [52] so there was a need for more innovative, lightweight, discovery-based approaches that treat requirements differently.

In the 1980s and 1990s iterative approaches emerged. These changed the nature of the requirement analysis [4]. It has become part of an iterative and incremental process [4]. Iterative approaches apply lighter-weight documents. The main iterative approaches are [2]: spiral model, RAD (Rapid Application Development), and RUP (Rapid Unified Process). Spiral model stands for the rapid development of understanding via experimental discovery. It is very important for the spiral model to understand requirements and perform some validation of them before any more serious development begins. Thereafter, the model assumes another, bigger "spiral" intended to develop the solution in largely sequential steps of design, coding integration, and testing. As such,

this was the first published model to be based on a more discovery-based requirements process. Most give credit goes to this way of thinking as the starting point of the iterative and incremental software development methods movement.

RAD is related to the rapid building of models, prototypes, and initial systems using more advanced tools. RAD is considered to be a more generic term for a type of development process introduced by James Martin in the early 1990s. Although originally a fairly well-defined software process model focusing on the iterative development and construction of an increasingly capable series of prototypes, today it generally stands for any number of lighter-weight approaches using frameworks which accelerate the availability of working software. RAD approaches emphasize the speed of deployment and feedback over performance and scalability. Further, from a requirements perspective, the idea was to test the code before the requirements changed. If the requirements turned out to be wrong, the tools enabled a quick change of code which was still faster than the use of traditional methods. RAD methods are mostly object-oriented [59].

And, RUP stands for the iterative and incremental development of ever larger and more complex systems. It was launched in the late 1990s by the Rational Software Division of IBM. It is based more on the spiral model than RAD is. Moreover, it is intended for large-scale applications where robustness, scalability, and extensibility are mandatory. RUP has proven to be an effective framework for the practical guidance and management of large-scale application development. It has seen widespread industry use and been successfully applied to thousands of projects of all types, including projects on a very large scale. RUP was the first widely adopted software process that recognized the necessary overlap of the various activities that occurred during the life cycle phases of inception, elaboration, construction, and transition. With RUP, requirements elicitation is no longer relegated to a single phase. Although requirements elicitation activities are particularly intensive during the early inception and elaboration phases, a change in requirements is considered to be a continuous process that occurs throughout the life cycle.

There were many more iterative approaches. In 2001, many creators of the iterative approaches came together and created an Agile Manifesto in which they summarized their beliefs for the better development of applications [6]. The beliefs are presented in the manifesto which claims that the following should be preferred [60]: individuals and interactions over processes and tools; customer collaboration over contract negotiation; working software over comprehensive documentation; and responding to change over

following a plan. Behind the manifesto there is a set of core principles that serves as a common framework for all agile methods [60]:

- The highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- To welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The two main agile development approaches are: extreme programming (XP), and Scrum [6]. A survey in 2009 (published by Courtesy of VersioOne, Inc.) reports the most commonly used agile development approaches in practice [2]. Scrum was used in 50% of all agile projects, XP in 6%, Scrum/XP hybrid in 24%, and other agile approaches in 20%.

Scrum is an agile project management method [3, 9, 61]. In Scrum, the product backlog is a list of requirements that represents everything that might be needed in the application. Even though Scrum does not mandate any specific requirements model, most Scrum practitioners recommend the use of user stories. Therefore, Scrum is user-story-driven development (usage-driven development). A collection of user stories is estimated and prioritized for development iterations. The user stories may be changed or reprioritized

at any time. However, radical changes to the product backlog need to be carefully managed.

Development team makes a release plan. The team develops an initial, high-level plan. Release planning includes abstracting the project plan to a higher-level that describes higher-level milestones such as the delivery of small increments of demonstrable functionality. These higher-level plan items are then allocated to a set of development iterations over the life of the agile development project. A goal of each development iteration is to deliver potentially shippable software at the end of each iteration (also referred to as sprints). At the beginning of the development iteration, the team plans in detail what it will deliver and how it will do the work. This involves taking higher-priority requirements off the list of user stories and then decomposing these requirements into detailed tasks. The team again estimates the work so the development iteration lasts a four weeks maximum. At the end of each iteration, the team demonstrates what it has completed to the project's stakeholders, and takes opportunity to identify ideas for improvements in the development process. The team also assesses progress made against the release plan. The project is over when the customer determines that they do not want to fund another iteration and are satisfied with sufficient functionality.

XP is an agile software development method that primarily focuses on construction-oriented (coding) practices [9, 62, 63]. These practices appear straightforward on the surface, but many of them require not only technical skill but also significant discipline. For example, pair programming is practice where two team members work together on the same code. One team member types the code while the other looks at the bigger picture and provides a real-time code review. They both seek the simplest way to write their code, since simple design is believed to increase productivity. Development team members validate their new functionality's code every time they integrate the code with the existing code. The validation is done via automated regression tests and potentially even through dynamic and static code tests. A test for the functionality is coded before the functionality. The test validates at a confirmatory level that the code is working as expected, and specifies the team member's work in detail on a just-in-time basis. If some code does not work according to expectations, the refactoring is performed. Refactoring is a small change to a code, database, user interface, etc. in order to improve an application's design. It enables an iterative and incremental approach to development. Furthermore, every team member is allowed to view and edit another team member's code or any other project artifact. So any team member can refactor any code. This

encourages transparency and accountability for coding quality. All development team members should follow recommended patterns and mechanisms to ensure consistency, reduce defects, and simplify readability and maintenance. Furthermore, all of them together should have all the skills required to deliver the application.

Similar to Scrum, XP promotes small releases. Frequent deployment of working software in production is encouraged. This functionality should be delivered in increments that provide the greatest value to the customer side. Frequent deployments build confidence in the development team and trust from the customer side. High-level release planning called planning game encourages thinking and monitoring big issues. On the other hand, iteration planning is detailed and performed regularly on a just-in-time basis. Detailed requirements are also captured on a just-in-time basis but in the form of acceptance tests. Overall, the development team should be able to sustain an energized approach to work at a constant pace at the beginning and gradually improve its velocity. If they have any business related questions, the customer side (or their representatives such as a product owner) should be available to answer questions and make decisions in a timely manner.

Interestingly, even though all of these mentioned development approaches became popular in different periods of time, they are all still in use today [2] – see Figure 1. Even the waterfall approach is still in use, for a number of reasons [2]:

- It appears to be extremely logical and prescriptive: understanding requirements, designing a system, coding and testing it. The flow is sensible and logical.
- It works to a point.
- It reflects a continuing market reality, namely that customers still impose fixed-date/fixed-requirements agreements on suppliers.

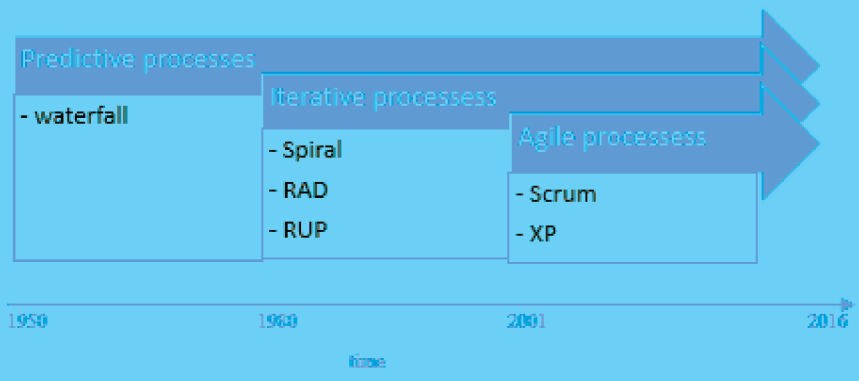


Figure 1: Software process movements over the past few decades (adapted from [2])

Nevertheless, agile approaches and their hybrids are believed to be answering the question of how to deal with requirements analyses in a better way so more projects are completed and better estimated so the product is delivered on time and on budget [2].

2.1.2 Agile requirements modeling

The agile approaches have changed the requirements analyses dramatically. Agile approaches suggest that the requirements analysis is constantly repeated thru the project, and, therefore, focused on the needs of a particular development iteration. So-called agile analysis is a highly iterative and collaborative process in which all stakeholders actively work together on a just-in-time basis to understand the domain, identify what needs to be built as requirements, and estimate and prioritize the requirements [4]. With agile analysis the date and resources are fixed while requirements are variable [2]. As such, the requirements are determined on a high level of abstraction with less effort. Even though agile development approaches acknowledge the impossibility of writing all of the requirements at once, they still make an initial upfront attempt to write those that can be written. Consequently, this kind of requirements are modeled at a higher-level of abstraction [3] while the detailed requirements modeling is done on a just-in-time basis during development iterations [4].

Agile modeling is a practice-driven methodology for effective requirement modeling and documentation of software-based systems [9]. It is a collection of values, principles, and practices for modeling. According to Ambler [7] modeling values are: communication, simplicity, feedback, courage and humility. When communicating, participants gain and give information. When modeling, the development team applies simplicity by following the KISS rule (Keep It Simple Stupid). Furthermore, developers need to ensure that the created models (abstractions) are correct. The feedback can be gained in many ways: when developing the models as a team, when reviewing the models with a target audience, with implementation of the model and with the acceptance testing. Next, courage is important for many situations such when making important decisions about the architectural approach. The team members need to trust that they can overcome tomorrow's problems tomorrow. And finally, humility. Team members need to understand that other project's stakeholders have their own areas of expertise. Humility comes in to play when interacting with people, since different stakeholders have different priorities, experiences, and viewpoints. Arrogance leads to communication problems.

Agile modeling values have a problem. They are not specific enough to provide guidance to software development efforts [7]. There was a need for more concrete concepts –

principles. Software is the primary goal is one of agile modeling core principles. The focus should be on coding and testing, not on producing extraneous models. Second principle is suggesting that enabling the next effort is a secondary goal. The next effort applies to supporting the following project. In order to enable effective continuation, the developed software must be high in quality and the documentation needs to be just right. Third principle proposes to travel light. The modelers should create just enough models and documentation to get by. The development team members need to have good communication among themselves to be able to effectively travel lightly. By following the fourth principle we need to assume simplicity. The simplest solution is the best solution because it is easiest to be implemented. The fifth principle teaches to embrace change. Since requirements change over time, the stakeholders' understanding of their requirements can change, new people may be added and existing ones may leave. The stakeholders can also change their viewpoints, their goals, and success criteria. That is ok. The sixth principle is embracing incremental change. The application should be changed with a series of small incremental changes. The seventh principle teaches to model with a purpose. Modeling is done for the purpose of understanding and communicating. Furthermore, the eight principle suggests using multiple models. Many modeling techniques are available. We need to use those ones which are appropriate to the situation/project. The ninth principle is quality work, which improves communication on a project. Agile developers understand that effort is needed to make artifacts such as source code, user documentation, and technical system documentation permanent. And finally, the tenth core principle is maximization of stakeholder investment. Stakeholders deserve to invest their resources in the best way possible and to have the final say in how those resources are invested.

Values and principles are used to define concrete practices. Selected agile modeling practices are [9]:

- Architecture envisioning. Initial, high-level architectural modeling is needed to identify a viable technical strategy for developing the application. This is done iteratively parallel to requirements envisioning.
- Requirements envisioning. An initial prioritized list of requirements is needed to identify the scope of the project.
- Iteration modeling. Iteration planning activities help identify what exactly needs to be built and how it will be built.

- Just barely good enough artifacts (sufficient artifacts). A model or document needs to be sufficient for the solution at hand and no more.
- Multiple models. Each type of model has its strengths and weaknesses. Effective developers use a range of models, enabling them to apply the right model in the most appropriate manner for the situation at hand.
- Prioritized requirements. In order to provide the greatest return on investment, the agile team implements requirements in the order of priority set by the stakeholders.

These are just a few selected practices. Ambler presents them as a part of Agile Model Driven Development (AMMD). AMMD is an agile modeling approach which includes a particular set of practices and promotes just enough high-level modeling at the beginning of a project to understand the scope and potential architecture of the system [9]. In general, model-driven development could become even more greatly used in software development over the next several years and there is reason to hope for significant improvements in software quality and time to value, but it is far from a foregone conclusion that model-driven development will succeed where previous software-engineering approaches have failed [64]. Figure 2 visually presents the lifecycle of the AMDD approach. During iteration 0 (also called inception) the project becomes organized. Part of that effort is the initial envisioning of the requirements and the architecture so that critical questions about the project's scope, cost, schedule, and technical strategy can be identified [4].

In the initial requirements envisioning days, the agile requirements models are not meant to be detailed, just good enough to provide credible estimates [4]. Our research in this thesis focuses on supporting the initial requirement envisioning days and making those days as effective as possible in terms of eliciting user stories, understanding them, and providing a sense of the project's scope. Below we refer to the person who is in charge of these days as an analyst. We do not specify whether the analyst is part of the development team side or part of the customer organization side because it depends on the development project.

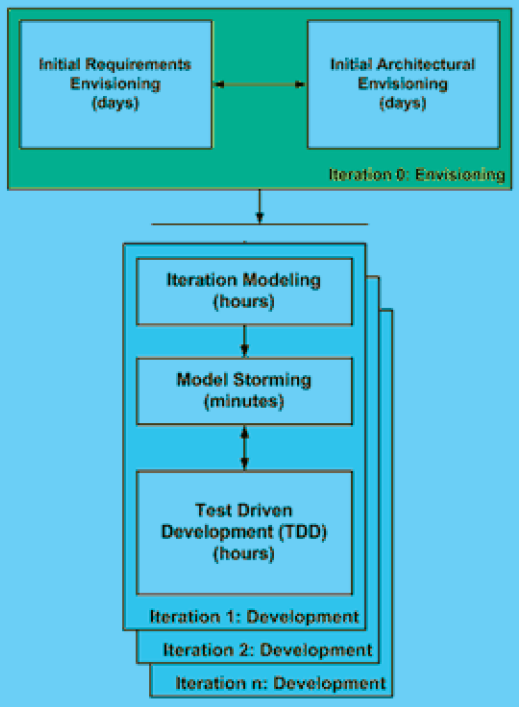


Figure 2: Lifecycle of the AMDD approach [4]

2.1.3 Understanding requirements

Requirements need to be understood by all stakeholders [3, 7, 8]. Nevertheless, they are exposed to misinterpretation. It is appealing to think that if everything is explicitly written down and agreed to then there can be no disagreements, developers know exactly what to build, testers know exactly how to test it and, most importantly, customers obtain exactly what they wanted [65]. But that is not the case. Customers frequently obtain the developers' interpretation of what was modeled, which may not be exactly what they wanted [65]. What the stakeholders need is an effective way of communicating, which can be achieved by combining different requirement models [7, 8] and by supporting the development team with business domain knowledge [14, 66].

Gottesdiener [8] combines different requirements models in her model called the evolution of requirements. The requirement elicitation starts with a definition of the *business requirements*. These requirements enable the organization to do things such as increase revenue, reduce costs, improve service, and meet regulatory requirements [8]. On the other side, there are *software requirements*. They are the most detailed requirements which are implemented by the development team. They consist of both functional and non-functional requirements [8, 67]. Functional requirements are the *doing* parts of software. They are about what a software should do, how it should behave, what it should contain, or which components it should have [52]. They can also be implemented

as software algorithms and data structures. The non-functional requirements are the *being* part of software, such as: the users' desired quality attributes, technical requirements related to auditing and security, quality attributes such as capacity and reliability, and constraints such as the platform and database [52]. These non-functional requirements are often critical. It is important to discover them simultaneously while eliciting functional requirements. They are not directly implementable in software [52].

Due to differences in languages, models, knowledge, work styles, etc. of the project's stakeholders, the bridge between business requirements and software requirements is often difficult to pass [8]. In order to cross that bridge, Gottesdiener [8] proposes a new middle level of requirements which she calls *user requirements.* She suggests that every software requirement should be associated to a user requirement and every user requirement should be associated to a business requirement. The *doing* parts of software are derived directly from user requirements while the *being* parts are presented as constraints.

Business, user, and software requirements need to be modeled in order to facilitate communication [7, 8]. Modeling the requirements has a positive influence on the understanding of what to build [7]. Choosing the best model for requirements depends on a project's domain, which is to be supported by the new software [5]. The nature of a problem domain refers to how the business is organized (structured vs. ad-hoc). For example, if added value of the customer's business is provided ad-hoc (which is more common in small businesses) it is not very beneficial to model business processes for the purpose of software development simply because there is too little structure to be presented visually. In order to select the most appropriate requirements models, one must understand their basic purposes and what they can communicate [8, pp. 27]. A requirement model is attributed with a particular point of view, a perspective on a certain focus, and a specific level of detail [8]. First, requirements models can be categorized according to four views [8]: behavior (this models are oriented towards processes, tasks, or sequences), structural (this models describe parts and relationship in data models and class diagrams), dynamic (this models describe how things change over time), and control-oriented (this models describe decisions and policies that provide guidance for the other views). Second, the requirements models are attributed with five questions which present the specific focus: who, what, when, where, why, and how [8]. Lastly, the requirements models can be captured at varying levels of detail: charter, scope, high level,

and detailed [8]. In practice, requirements models do not fit neatly into a single level – they can cross levels [8].

The most important barrier in communication is diversity among groups of stakeholders [66]. It is partly caused by a gap in the stakeholders' domain knowledge [14, 66]. The most common systems development task is to understand the domain which is to be supported with a new application [68]. The domain knowledge the analyst already possesses is an important contributing factor to software development in general and requirements elicitation in particular [69]. The domain knowledge has positive effects on the effectiveness of problem-solving [68, 70-76], on better cognitive processes that take place during the interactions with the stakeholders [77], on understanding of the customer side's needs and on communication with the customer side [69]. Therefore, domain knowledge is one of the crucial factors for success in high-quality requirements elicitation [69, 78] since it enables better understanding of the requirements. By understanding the requirements, we can decrease the likelihood of a project delay, schedule overruns, and the need for task switching [23]. One way of understanding them is to know the dependencies among them.

A requirement dependency is a situation in which the progress of action on one requirement assumes the timely outcome of action on another requirement or the fulfilment of a specific condition [17]. Dependencies are ubiquitous when developing an application and occur between people, groups, tasks, and artefacts, including the software components under construction [79]. Understanding requirement dependencies needs to be accomplished before carrying out the design in order to avoid the unnecessary implementation of complex functionality [80]. Focusing on dependencies as early as possible in the project was deemed beneficial because of its potential to save rework and redesign [80]. On the other hand, ignoring the dependencies increases the risk attached to cost-effective project execution and, consequently, to project success [21, 81-83]. Indeed, missed dependencies can lead to a project's delay as people wait for resources, for the activities of others to be completed, or for necessary information [23]. In this thesis, we are interested in providing support to the analyst so that he/she can better understand requirement dependencies. We aim to support his/hers domain knowledge with business process models.

2.2 User story – a requirement model

2.2.1 What is a user story?

A user story is a requirement model which is the most commonly used non-visual artifact in agile development [5, 7]. User stories have become widely accepted in agile software development and are supported by a great number of software tools [84]. Cohn [3] defines a user story as a description of functionality that is valuable to the customer organization side. Leffingwell [2, pp. 100] adds that is a brief statement of intent that describes something the system needs to do for the user. Therefore, a user story is a tool for defining a system's behavior in a way that is understandable to both the developer team members and the customer organization members [2]. A user story supports a lightweight and effective approach to managing requirements for a system [2]. Moreover, user stories prioritize communication through the whole development process [3]. Accordingly, user stories are there to bridge the developer-customer communication gap [2]. User stories are traditionally handwritten on index cards as they are then easy to store, display, rearrange, and distribute to the co-located development team [85].

In agile development, it is the development team's job to speak the language of the user, not the user's job to speak the language of the developers [2]. Effective communication is the key, and there is a need for a common language. The user story provides the common language to build understanding between the user and the technical team. As such, they are a perfect agile requirement model representative. Still, writing a user story may appear to be a simple, atomic piece of requirements work but can turn into a complex, cognitive task necessitating a diverse range of interleaved cognitive processes as demonstrated by [86].

As mentioned, the goal of user stories is not to document every last detail about a desired feature, but to write down a few short sentences that will remind developers and customers to hold future conversations [3]. They grow into more detailed requirements during the project [7]. Accordingly, at the beginning of the initial requirement envisioning days there is just a list of user story clauses (descriptions) complemented with the development team's estimates and the customer's priorities.

The clauses are short, easy to read, and understandable to developers, stakeholders, and users [2]. A clause typically follows a certain template which defines a user story model. Agile literature offers many templates that look quite similar. Typically, a user story

clause is created by following a template such as this one [2, 3, 7]: *I as a <user role> can <function >*.

During development projects, the individual user story clause is put on a so-called user story card which contains three aspects [3]:

- a user story clause as a reminder of content;
- conversation notes about the story to flesh out the details; and
- tests which convey and document those details.

We conclude that the user stories elicited in the initial requirement envisioning days have much fewer attributes than the same user stories at the end of the development project. The notes about conversations and tests are made by the development team in the following development iterations [2]. We excluded those notes from the list of attributes since they are typically not available at the end of the initial requirement envisioning days. Attributes present at the end of the initial requirement envisioning days are listed in Table 1.

Attribute by Gottesdiener [8]	*User stories at the end of the requirement envisioning days*
Point of view	Behavior [2]
Perspective on a certain focus	Who → <user role>
	What → <function>
Specific level of detail	List of high-level requirements Scope (requirement estimates, priorities)

Table 1: Attributes of user story clauses at the end of the requirement envisioning days

The investment in good user stories is a worthy effort for the team. Bill Wake coined the acronym INVEST (independent, negotiable, valuable, estimable, small testable) to describe the qualities of a good user story, which is used by many agile teams [2]. Independence means that a story can be developed, tested, and potentially even delivered on its own, therefore, it can be independently valued. Unlike traditional requirements, a user story is not a contract for a specific functionality but a placeholder for requirements to be discussed, developed, tested, and accepted. This process of negotiation between the business and the team recognizes the legitimacy and primacy of the business input but allows for discovery through collaboration and feedback. Written requirements traditionally served as a record of past agreements. Agile, however, is founded on the concept that a team-based approach is more effective at solving problems in a dynamic collaborative environment. A user story is real-time and structured to leverage this

effective and direct communication and collaboration approach. Finally, the negotiability of user stories helps teams achieve predictability. Next, every user story must provide value to a stakeholder of the product. A typical challenge facing teams is learning how to write small, incremental user stories that can effectively deliver value. Next, a user story needs to be estimable. Although a story of any size can be in the backlog, in order for it to be developed and tested in an iteration the team should be able to provide an approximate estimation of its complexity and the amount of work required to complete it. The minimum outcome of the estimation is to determine whether it can be completed within a single iteration. Additional estimation accuracy will increase the team's work predictability. If the team is unable to estimate a user story, it generally indicates that the story is too large or uncertain. If it is too large, it should be split into smaller stories. One of the primary benefits of estimating user stories is not simply to derive a precise size but to draw out any hidden assumptions and missing acceptance criteria and to clarify the team's shared understanding of the story. Next, user stories should be small enough to be able to be completed in an iteration. Otherwise, they cannot provide any value or be considered as "done" at that point. However, even smaller user stories provide more agility and productivity. There are two primary reasons for this: increased throughput, and decreased complexity. Smaller stories not only go through faster because of their raw, proportional size, but they go through faster because of their decreased complexity. And, finally, user stories need to be testable. In proper agile, all code is tested code and so it follows that user stories must be testable. If a story does not appear to be testable, then the story is badly formed. To assure that stories do not get into an iteration from which they cannot get out, many agile teams today take a "write-the-test-first" approach. This started in the XP community using test-driven development, a practice of writing automated unit tests prior to writing the code to pass the test. The mentioned approach is being to the development of story acceptance criteria and the necessary functional tests prior to coding the story itself.

User stories have several advantages [3]:

- *User stories emphasize verbal communication*. Even though writing things down has advantages like overcoming the limitations of short-term memory, distractions, and interruptions, it is verbal communication that enables a feedback loop that leads to mutual learning and understanding.
- *User stories are comprehensible by all stakeholders*. Some would suggest the use of formal, mathematically-based specification languages which make the meaning

of requirements more explicit [52]. Unfortunately, formal specifications are not comprehensible to most users so formal languages present a communication barrier [52]. User stories can be more succinct than traditional requirements specifications and, as such, comprehensible by both business people and developers [3]. Further, since they are written and discussed as stories, the recall of stated and unstated actions is greater [87].

- *User stories are of a manageable size for planning releases, for programming and testing, and for prioritizing*. Typically, in the initial requirements envisioning days the user stories are written with higher-level granularity (that is, they are more abstract) and as such are prioritized more easily. On the other hand, traditional requirements specification suffers from the problem of being too big.
- *User stories are compatible with iterative development since they encourage deferring detail*. As such, they can be written at whatever level of detail that is appropriate. More abstract (that is, more general) user stories are written for a larger portion of an application. The evolvement of more detailed user stories can be done later in development interactions. It is important to have an overall feel for the size and scope of an application long in advance before starting it. When having that, the analyses of costs and benefits can be performed. A development team can very quickly write a few dozen stories to give them an overall feel for the system. This is something that is not possible with traditional requirements models which have an overwhelming form/template. User stories are not specified by an overwhelming template.
- *User stories support opportunistic and participatory design*. This means that the development team does not follow a top-down approach but instead an opportunistic approach. In an opportunistic approach, the developers move freely between considering the requirements, discussing usage scenarios, and designing at various levels of abstraction. User stories allow users not to fully know and communicate their exact needs in advance, and also allow developers not to be able to fully comprehend a vast array of details. The developers can shift between high and low levels of thinking and talking about requirements.
- *User stories build up tacit knowledge*. This happens because of the emphasis placed on face-to-face communication. The user stories promote the accumulation of tacit knowledge across the stakeholders.

There are several disadvantages of using user stories, which also underpins the motivation for the work in this thesis:

- Sometimes more time is spent on writing about stories than talking about them. The customer in a project sometimes does not accept responsibility for writing user stories [3].
- In large projects, it can be difficult to keep thousands of user stories organized. Agile development teams have a problem of losing sight of the project's big picture [88]. The big picture represents the context of an individual user story which is on the list of multiple user stories [2]. By understanding the context of a user story in the customer's business domain, we can gain a sense of the project's scope.

We argue that the disadvantages of user stories originate from poorly performed initial requirement envisioning days during which the user stories need to be elicited and understood as part of the big picture.

2.2.2 Eliciting user stories

Eliciting user stories requires intense communication that needs to overcome any culture gap or semantic differences that may exist between users and the development team [10]. An analyst is there to help with bridging that gap. In agile development methods, the activities performed by an analyst are hidden in other roles. For example, in Scrum, the product owner writes user stories with input from the customers, the stakeholders, and the team. Nevertheless, in practice any team member with sufficient domain knowledge can write user stories but it is up to the product owner to accept and prioritize those potential stories into the product backlog.

On the other hand, in XP, user stories are often written by the customer, thus integrating the customer directly into the development process. An XP project's customer is responsible for writing as many user stories as early as possible [2]. Because a user story is intentionally an incomplete description of the requirements, an important XP practice is to converse with customers. Before beginning a story, the story's owner spends a few minutes with the customer to better understand what he or she wants [89]. Further, the customer is responsible for making sure that all user roles are appropriately represented [3]. If the customer wants help in writing the stories, the customer is responsible for scheduling and running several workshops. The developers can help by either doing the actual writing during an initial story writing workshop or suggesting new stories to the

customer while developing. The responsibility for writing stories resides with the customer and cannot be passed on to the developers. A project's customer is responsible for understanding multiple approaches for eliciting user stories [3].

User story elicitation approaches are used according to the situation and the available resources [11, 12]. Methods of eliciting requirements are now more cooperative. There are many techniques for obtaining requirements from customers. Selecting the right techniques according to the project's characteristics is very important [90]. The approaches used for eliciting user stories must be lightweight and non-obtrusive so they are applied more or less continuously. We divide user story elicitation approaches into two groups according to their point of departure.

In the first and biggest group of approaches, the elicitation starts with a blank piece of paper. Many of these approaches are also used in traditional development. Those that are explicitly employed for user story elicitation are:

- *The interview approach.* The most commonly used requirement elicitation approach is an interview [3, 69, 91]. Several research studies have focused on evaluating the effectiveness of the interview as a requirement elicitation technique [69]. Traditionally, an interview is known as the most effective elicitation approach [92] which is appropriate for drawing out explicit knowledge [93]. Interviews basically depend on the quality of interaction among the participants. Types of interviews include an unstructured interview and a structured interview [91].

  The interview approach can also consist of other techniques such as the recall technique, the retrieval technique, and the technique with schemas. The recall technique can be used for overcoming the cognitive limitations of the interviewee [69]. In addition, the interviewer can employ various retrieval techniques for activating the interviewee's recall [94]; for example, recalling by chronological order or from the perspective of a third person. The technique with schemas can be either context-dependent or context-independent [95]. Browne and Rogich state that context-dependent schemas are more powerful. Browne and Ramesh [96] stress that questions driven by context have greater power than generalized ones because they address a problem more specifically. However, in order to construct such context-dependent schemas the analyst must have significant expertise in the domain [69].

Cohn [3] stated that one of the keys to success with interviews is the selection of interviewees. He adds that interviews should be performed with real users who play different user roles. Nevertheless, most users are not very adept at understanding and/or expressing their true needs. Just because users have the problem does not necessarily mean that they are qualified to propose the solution to it. Therefore, which kind of questions the interviewer asks is important. Cohn proposes a technique consisting of open-ended and context-free questions. Open-ended questions let respondents express more in-depth opinions. Context-free questions are those that do not include an implied answer. By starting with this type of questions, there is a possibility for a wider range of answers from users which leads to eliciting user stories that may have remained undiscovered if only specific questions had been used. LaFrance [97] adds that going from context-free (generic) to specific questions is beneficial because the context-free questions are focused on defining the scope and characterize the domain while the specific questions go into greater detail.

- *Questionnaires.* Questionnaires are an effective technique for gathering information about existing user stories but are usually inappropriate as a primary technique for eliciting new user stories [3]. They foster one-way communication and an inherent time lag. Unlike in a conversation, it is impossible to follow a user down an interesting path. Cohn [3] suggests using questionnaires mainly for prioritizing user stories.
- *Focus group.* A focus group uses a semi-structured questionnaire at a meeting with a group of people [2]. It is used when it is necessary to collect many answers on qualitative information as quickly as possible. An efficient choice of participants and mediating tools is crucial for the success of elicitation via focus groups. The use of a web-based focus group method is also possible [98].
- *Observation.* The observation technique is typically used for observing how users interact with the release of an application [3]. Nevertheless, opportunities for user observation are rare unless the development is done for an in-house customer. Just like questionnaires and focus groups, this technique cannot be used for eliciting the initial list of user stories. In the past, it has been successfully used to verify specific needs/requirements defined by a user. For example, Silva et al. [99] combine techniques such as observations and interviews to generate artifacts (proper prototypes) that help define user stories.

- *Story-writing workshop.* The benefit of the workshop process is that it nurtures team communication, decision-making and mutual understanding [67]. A properly conducted story-writing workshop can be a very rapid way to write a large number of initial user stories [3]. A story-writing workshop is usually held near the beginning of the project [100]. It is a meeting that includes developers, users, and other parties who can contribute by writing user stories. Further, it combines the elements of brainstorming and low-fidelity prototyping. The idea is not to identify actual screens and fields, like in a traditional prototyping session, but to identify conceptual workflows.

  The low-fidelity prototype is done on paper, note cards, or on a white board. It is built up iteratively during the workshop as the participants brainstorm. Boxes represent a component of the application. The middle box represents the application's main screen. The analyst asks what a specific user role can do from there. The meeting participants brainstorm and, consequently, throw up ideas about which actions the user role can take. For each action, a new box and a line to it is drawn. Walking through the workflows typically helps the participants think of as many stories as possible. One low-fidelity prototype is drawn for a single user role. The result of the workshop is a list of user stories. Since the prototype is not a long-term artifact, it must be deleted soon after the user stories are elicited.

The second group of user story elicitation approaches starts with existing documentation. Sometimes when a development project begins a software development approach (e.g. XP, Scrum) may not have been chosen yet. The decision about it is made once the scope of the effort is understood and the people who will work on the project are identified [2, 22]. Even though the choice of the requirement model has not yet been made, the people who need to create initial requirements for the project can present them with, for example, a use case model [2].

2.2.3 Understanding user stories by understanding the dependencies among them

A specific user story is just one of many user stories that represent what is required of a cohesive software product. The dependencies among user stories are a subject of speculation when previous knowledge about the domain is insufficient. We argue that the list of user stories alone hardly depicts the execution order and integration dependencies

because of its three characteristics which we illustrate with user story examples in Table 2:

a) The naming of the user story functions can refer to different levels of abstraction (e.g. in Table 2 the function of user story US_299 "notify the customer about the credit committee's decision" says nothing about what kind of decision was made or why the decision had to be made).
b) Different synonyms can refer to the same artifact (e.g. in Table 2 the following terms and phrases are used for the same artifact: the limit, the account's attribute, the non-standard account limit, the non-standard limit).
c) User stories can take part in different execution orders (e.g. US_100 and US_300 in Table 2 do not execute in the same business context as others do).

List of user stories:	
US_189	I as a clerk in DEPT1 can modify the account's limit.
US_244	I as a clerk in DEPT1 can send a notification about a change in the account's attribute.
US_11	I as a clerk in the front office can create a request to modify the account.
US_299	I as a clerk in the front office can notify the customer about the credit committee's decision.
US_43	I as a clerk in the front office can give information about a rejection.
US_165	I as a clerk in the front office can read about the account's new limit.
US_99	I as a clerk in the front office can read about the decision.
US_999	I as a clerk in the front office can send a letter about the approval.
US_765	I as a credit committee (member) at a business unit can evaluate a non-standard account limit.
US_199	I as a credit committee (member) at a central unit can make an assessment report for a non-standard limit.
US_300	I as a clerk in the front office can notify the customer.
US_100	I as a credit committee (member) can evaluate the proposal.

Table 2: A list of user stories for Case 1

The development team can acquire information on dependencies among user stories from tacit and explicit sources of knowledge. In Scrum, a tacit source of knowledge is the Product Owner who is responsible for managing customer requirements [101, 102]. An analyst can choose complement the list of user stories with other models [3, 7] and, therefore, support the explicit base of knowledge. Several agile modeling practices promote the use of additional models next to the list of user stories, such as: "create several models in parallel", and "iterate to another artifact" [7]. Conceptual models are

the models that capture explicit knowledge about dependencies in order to promote understanding of the application domain [17].

The agile development literature offers several solutions for building an explicit knowledge base for managing the execution order dependencies among user stories:

- Leffingwell [2] suggests use cases (diagrams and text-written specifications). He suggests dividing the main scenario of a user case into smaller chunks and associating user stories with the corresponding chunk.
- Ambler [7] suggests an approach with a level-0 data-flow diagram.
- Kulshreshtha et al. [15] depict relationships between requirements as nodes in the requirement network.
- Patton [27] proposes an approach called story mapping which is based on understanding user scenarios. The big story (the backbone) is presented through a simple process diagram. The approach suggests placing physical user story cards on a timeline.
- Milicic et al. [26] propose a user-centric mapping method which organizes backlog items along scenarios and users by defining concepts of the domain and relations which exist between the concepts.

User stories are very easily written at different levels of abstraction (also of detail, granularity) [3]. In order to understand integration dependencies among user stories on different levels of abstraction, the agile development literature offers several solutions to build explicit knowledge about the dependencies:

- Patton's approach [27] suggests placing the cards of child user stories below the cards of large user stories in order to show a hierarchy of detail. Details help to imagine the product that will be built.
- Leffingwell [2] proposes a full enterprise requirement model (see Figure 3). The model represents a hierarchy of requirements to be developed. Epics are large-scale development initiatives that realize the value of investment themes. Epics are the highest-level requirements artifacts that are used to coordinate development. They are not implemented directly but are instead broken into features which, in turn, are broken into user stories that are the primitives used by the team for actual coding and testing [2]. Epics, features, and user stories are all requirements listed in the product backlog [2] and, as such, can all be written with the user story template no matter at which abstraction level they belong to.

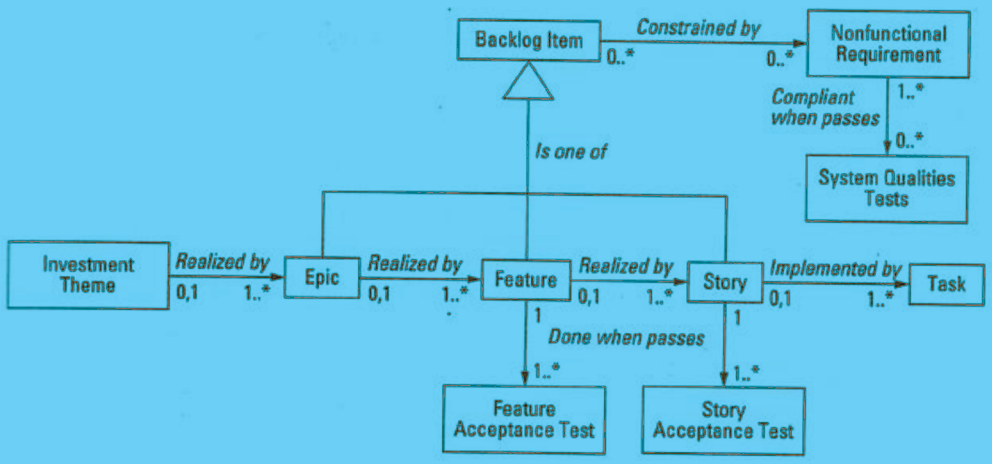


Figure 3: Full enterprise requirement model [2]

- Cohn [3] gives an overview of techniques for slicing one big user story up into smaller user stories. The first technique is to split the story along technical lines. A problem with this technique is that none of the user stories is then useful to users. Another technique called "slicing the cake" entails writing the replacement user stories such that each provides some level of end-to-end functionality. In this way, each user story has a little of each layer. Stories that represent a full slice of cake are to be preferred because they reduce the risk of finding last minute problems in one of the layers and because they present full functionality, albeit limited.

  If we keep track of how a particular user story was sliced, we gain a hierarchy of user stories on different levels of detail.

The list of user stories requires an additional model to clearly present how user stories depend on each other. As mentioned, conceptual models can capture explicit knowledge about integration and execution order dependencies among user stories in order to promote understanding of the application domain [17, 28]. Business process models are the most widely used type of conceptual model [103] and capture how the business works and how value is created for different business stakeholders [104].

2.3 Business process models

2.3.1 Business process modeling

Business processes are very important for the early design phase of a software development project since they provide a procedural view of what is going to be done, who is going to do it and why [105]. Accordingly, in order to have unambiguous and transparent business processes they need to be modeled [106].

Business processes modeling starts by understanding what employees do and how activities complement each other to make added value for the customer [107]. Process models are created by communicating with relevant stakeholders or derived from organizational documents such as business policies [108]. The practice of business process modeling has emerged as a key instrument to enable decision-making in the context of the analysis and design of process-aware enterprise systems, service-oriented architectures, workflow operation, and web services alike [30].

It is common for large organizations to maintain repositories of business process models in order to document and continuously improve their operations [109]. Kovačič and Bosilj Vukšič [110] claim there are three basic purposes for modeling business processes in the first place:

1. Documenting: Standard ISO9001 requires the modelling of business processes. In different countries, the law requires some types of businesses to have standardized processes.
2. Analyzing: a company wants to change the way it works to become more efficient and/or effective in the future. Simplifying is carried out to achieve the higher efficiency and better performance of processes. The results can be shorter waiting time, lower inventory costs, less administration, etc.
3. Information systems (IS) support: beside the analysis, a company usually wants to support its business processes in a new better way with an IS. Optimization results in the standardization of processes which means no more variability in business process execution. In order to make good decisions about the design of information systems, an essential skill is understanding the process models of the business domain the system is intended to support [30]. Therefore, analysts develop process models to capture relevant information about a business process they seek to re-design, analyze, or support with an appropriate information system [30]. The main goal of information technology is to give support to the business via automatization.

   The goal of many software development projects is to support business processes [29]. The documentation of business processes in the analysis and design of process-aware information systems has gained attention as a primary focus of modeling in information systems practice [103]. The so-called practice of process modeling has emerged as a key instrument to enable decision-making in the

context of the analysis and design of process-aware enterprise systems [111], service-oriented architecture [112-115], and workflow operation [116] alike.

Each business process has a so-called as-is version and, depending on the purpose of modeling, also a so-called to-be version. In an as-is version of a business process model, the process is presented as it is currently being implemented [8]. This version is typically modeled only for documentation purposes. For the purpose of optimizing, typically a to-be version of a business process model is proposed. Moreover, optimized business processes are usually supported by applications so the requirements need to be modeled. Thus, when modeling the requirements the as-is models can be used as a reference point [8]. In this case, business process models are used to identify the user roles and/or ensure that the requirements are not missing any important steps if selected business flows [8]. Further, business processes are essential for understanding a business domain which is intended to be supported by a new application [30, 106, 117].

One of the agile supplementary practices that supports the core practice of "maximize stakeholder investment" is called "reuse existing resources" [7]. Documentation such as higher-level business process models is typically available in both large [118] and small- and medium-sized companies [119]. Analysts are likely to uncover reasonably accurate business process models within the customer's organization [7].

Notations play an important role in business process modeling since they define rules of how models can be constructed [31]. There are many visual (also graphical, flowchart-like) notations, such as Event-driven Process Chains (EPC) [120], Yet Another workflow language (YAWL) [121], the UML (Unified Modeling Language) activity diagram [122], and Business Process Modelling Notation (BPMN) [45, 123]. In contrast, the alternative of utilizing textual descriptions for capturing business processes exists, such as a text-written use case model [31, 124]. In this thesis, we focus on one popular visual and one popular textual business process model, namely on BPMN and on a text-written use case model.

2.3.2 Textual business process model – a text-written use case model

A use case describes a sequence of actions between an actor and an application that produces a result of value for that actor [2]. There are two types of use cases that complement each other; namely, a use case diagram and a text-written use case model. A use case diagram visually presents how use case elements interact with other elements. Each use case element is described with a text-written use case model (also called a

specification). The Rational Unified Process (RUP) [125] defines both the use case diagram's elements and text-written use case model's sections. The content of each section is well explained. Both use case diagrams and text-written use case models to show every user's needs, every goal they have with respect to the system, and every business variant involved [2].

Use cases can be used for both traditional and agile development. Cockburn [124] notes five good reasons for using *use cases*:

1. They provide a short summary of what the system will contribute to the business and the users. They also provide a project planning skeleton, to be used in building initial priorities, estimates, team allocation, and timing.
2. The basic flow of events (also the main success scenario) of the use case provides everyone with an agreement on what the system will and will not do. It provides the context for each requirement. A scenario is a story about use [126].
3. The alternative flows of the use case model provide a framework for investigating all the little, niggling things that somehow take up 80% of the development time and budget. They provide a look-ahead mechanism so that the customer/product owner/business analyst can spot issues that are likely to take a long time to find answers for.
4. The alternative flows provide answers to the many detailed, tricky business questions programmers ask.
5. Text-written and graphical use case models show that the developers/analysts have thought through every user's needs, every goal they have with respect to the system, and every business variant involved.

Model's attribute [8]	Model: existing higher-level text-written use case models which were primarily made for development reasons [7]
Point of view	Behavior [2, 8]
Perspective on a certain focus	Who → subject of content – actor
	What → subject of content – event
	Why → subject of content – use case name
Specific level of detail	Flow of higher-level activities

Table 3: Attributes of text-written use case models

Use case name: CLOSE A BANK ACCOUNT

Basic flow of events

1. …
   Refer to the following alternative flows: A, B.
2. Clerk in the front office writes a notification letter about closure of the account to the customer.

Alternative flows of events

Alt.flow A:

1. Clerk in the front office fills in a request for cancelling authorizations.
2. Clerk in DEPT 2 deletes all the permissions.

Alt.flow B:

1. Clerk in the front office fills in a form to close e-banking
2. Clerk in DEPT 3 closes it.

Additional information:

After the clerk in DEPT 1 changes the status of customer's account to "passive" the clerk in the front office needs to close all open businesses, such as e-banking and authorizations. According to the characteristics of the account none, one (A or B) or two alternative flows (A and B) are triggered. No matter how many alternative flows are triggered, clerk in the front office always writes an email about the closure to the DEPT 4 who needs to send him/her all of the account's physical documents. The event (step 2 of the basic flow) can start after opened authorizations are closed, opened e-banking is closed and all the account's physical documents are handed to the clerk in the front office.

Figure 4: Example of text-written use case model

Leffingwell adds that, when applying uses cases to agile development, they need to be:

- lightweight, which means no design details, user interface aspects, etc.;
- not fixed, which means they are merely statements of intended system behavior;
- informal, which means they are presented on whiteboards, lightweight tools, etc.;
- thinking tools, which means the analyst should not worry about keeping them maintained.

We have already discussed text-written use cases in context of using them either for user story elicitation [7] or for understanding of execution order and integration dependencies among user stories [2]. However, as shown in Sections 2.2.2 and 2.2.3, empirical research on both aspects is limited. Table 3 presents the attributes of higher-level text-written use case models that we use in this thesis. Figure 4 shows the model's sections that we adopt from Rational Unified Process's use case specification.

2.3.3 Visual business process model – a BPMN model

Visual models can significantly contribute to the analytical process by enabling holistic communication [127]. A standard notation for the visual modeling of business processes is the Business Process Model and Notation (BPMN) [31, 128] which was specifically created to be used in the modeling of business processes [115]. A BPMN model describes a sequence of actions between multiple organizational roles that produces value for the customer.

Modeling projects which use BPMN are run for the multiple purposes discussed earlier: documentation, simplification, and optimization. If models are created for non-development purposes, they are typically captured on higher levels of abstraction and focused on an analysis of the organization's business processes (e.g. to meet the needs of an ISO certificate, to simplify) [30]. Existing high-level BPMN models are generally available in both large [118] and small- and medium-sized companies [119]. We argue that these business process models can be reused for a new purpose – information system support for a new application.

Business process models have already proven useful for requirement elicitation [129]. BPMN models have been promoted for use in software development in the past via the Service Oriented Architecture (SOA) concept. Nevertheless, those BPMN models are very detailed (low leveled) and modeled specifically to automatically generate some of the application's code [123]. We argue that for agile development with user stories high-level BPMN models can be beneficial for eliciting initial user stories. However, so far no empirical research on this matter has been performed. Moreover, the use of BPMN business process models for better understanding dependencies among user stories has not previously been proposed. The attributes of high-level BPMN models we are listed in Table 4. Figure 5 sets an example of a BPMN model that we work with in this thesis. The BPMN's elements that we consider are: activity, flow arrow, swimlane (pool, lane), OR and XOR gateways.

Model's attribute [8]	Model: existing higher-level business process models which were primarily made for non-development reasons
Point of view	Behavior [106]
Perspective on a certain focus	Who → BPMN element swimlane
	What → BPMN element activity
	Why → business process name, prior activity
Specific level of detail	Higher-level activities

Table 4: Attributes of existing higher-level business process models

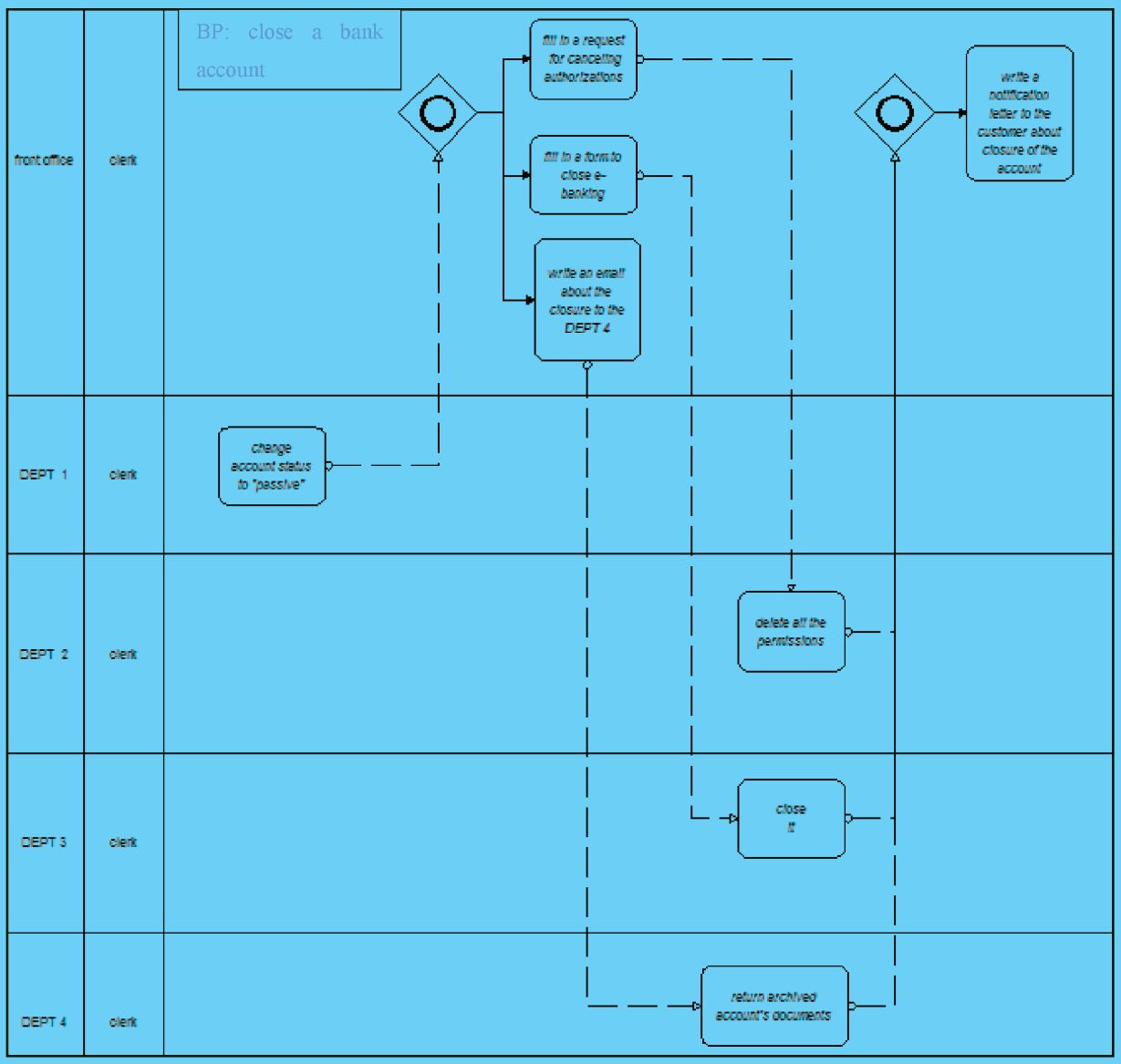


Figure 5: Example of BPMN model

2.4 Summary of Chapter 2

Over the years, the software development industry has tried out many different development approaches. The earliest approach is the waterfall approach, while (relatively) the newest is the agile development approach. These two approaches differ, among others, in how they deal with requirements analysis. Traditional systems analysis implies that there is a set of requirements that can be mostly determined up-front and that these requirements can be used as a basis to estimate the data and resources. This line of thinking has led many development projects to bad analysis which has been a frequent cause of failed projects [2, 52]. On the other hand, with agile requirements analysis the date and resources are fixed, while requirements present a variable. The relatively new agile analysis is a highly evolutionary and collaborative process where developers and

project stakeholders actively work together on a just-in-time basis to understand the domain, model the requirements, and estimate and prioritize them. Agile Model Driven Development (AMMD) is an agile modeling approach which allows eliciting requirements and understanding the project's domain [9].

A user story is a requirement model which is the most commonly used non-visual artifact in agile development [5], [7]. At the end of the initial requirement envisioning days, the user stories are presented in the form of clauses which are short, easy to read, and understandable to all stakeholders [2]. The development team makes them grow into more detailed requirements during the development iterations [7]. Approaches for eliciting user stories are lightweight and non-obtrusive so they are applied more or less continuously. We divided user story elicitation approaches into two groups according to their point of departure. The first group starts a meeting with a blank piece of paper, while the second starts with existing documentation. The first group of approaches is supported by the literature relatively well. However, the second group has very few proposals.

Once the list of user stories has been acquired/elicited, it also has to be understood how each individual user story fits into the big picture of the customer's business domain. It is a fact that a single user story does not convey the entire business, but only a part of it [15]. This means that while a user story is independent of the perspective of a developer it can still depend on other user stories from a business perspective [15]. In this context, insights into requirement dependencies are crucial for project success [15-21]. Similar to Martakis and Daneva [17], we emphasize the importance of understanding execution order and integration dependencies. The list of user stories in the initial requirements envisioning days requires an additional complementing model to clearly present the dependencies among user stories.

Business process models are the most widely used type of conceptual models [103] and capture how the business works and how value is created for different business stakeholders [104]. Business processes modeling starts by understanding what employees do and how activities complement each other to make added value for the customer [107]. To ensure good decisions are made about the design of an application, an essential skill is to understand process models of the business domain the system is intended to support [30]. Analysts are likely to uncover reasonably accurate visual and/or textual business process models within the customer's organization [7] which have potential to be used for user story elicitation and understanding of the dependencies among user stories.

3 Proposed method: Business Process User Story (BuPUS) method

In this chapter we present our Business Process User Story (BuPUS) method. In Section 3.1 we briefly describe the approach that we use to create the method. Next, in Section 3.2 we use ontological concepts to build generic data structure model. The constructs of our base method, which are presented in Section 3.2, are derived from generic data structure model's constructs. Furthermore, we present integration of the base method's constructs with general business process constructs in Section 3.4. Since business process models can be modeled in different configurations/notations we propose two project-specific methods which upgrade the base method. One is focused on BPMN model (Section 3.5) and another on text-written use case model (Section 3.6). For each configuration we propose its own integration with the base method (Sections 3.5.1 and 3.6.1), its own elicitation technique (Sections 3.5.2 and 3.6.2), and its own techniques for tracing execution order and integration dependencies (Sections 3.5.3-4 and 3.6.3-4). In Section 3.7 we present BuPUS process. Finally, in Section 3.8 we make a summary of the chapter.

3.1 Creating the method

Method engineering is an approach to create software development methods that are customized to the specifics of organizations or projects [44]. A specific type of method engineering is situational method engineering which deals with adapting existing methods to specific project situations [130]. Situational method engineering approaches differ in several aspects: the point of departure, the granularity of method fragments used to create situational methods, and the ability to execute. We use the Process Configuration Approach proposed by Bajec et al. [44]. This approach configures a new project-specific method by extending the base method. A base method encompasses instructions on how to handle various project types. It becomes richer every time new variations of handling projects are added. A new variation (a project-specific method) is added when a new project has a specific set of characteristics which cannot be correctly treated by any existing variation of the base method. Our proposed BuPUS method (which is our main artifact) is composed of multiple (sub-)artifacts:

- A generic data structure model presents ontological constructs, which are used to derive more specific constructs of base and project-specific methods.
- A base method's data structure model, which presents user story constructs.
- Model of general business process, which presents general business process constructs.

- A model of integration of the user story and general business process.
- A first project-specific method presents 1) a model of integration of the user story and BPMN, 2) a technique for eliciting user stories from BPMN models, and 3) two techniques for tracing both execution order and integration dependencies by using BPMN models.
- A second project-specific method presents 1) a model of integration of the user story with text-written use case model, 2) a technique for eliciting user stories from text-written use case models, and 3) two techniques for tracing both execution order and integration dependencies by using text-written use case models.
- A *BuPUS process* proposes a flow of activities which are needed to elicit user stories from business process models, and provide a better understanding of execution order and integration dependencies.

3.2 Generic data structure model

As suggested by Saghafi and Wand [28], we use ontological concepts to clarify the meaning of the constructs of the method. In general, ontological concepts organize and describe what exists in reality in terms of: the properties, structure and interactions between real-world things [131]. Bunge's ontology and its concepts are the most widely used ontology in the analysis and design of information systems [35]. We use ontology's concepts to define the constructs of a generic data structure: a *thing*, *composite thing*, and *property* [132]. The world is made up of *things*. A *composite thing* consists of multiple *things* which are then referred to as components. A *property* can be either inherited (a *property* of a *composite thing* is also a property of a *component*) or emergent (a property of a *composite thing* is not part of any of its components). A *composite thing* must possess emergent *properties* of its own next to the inherited properties. These three constructs (shown in Figure 6) are of the highest abstraction level (in later figures we refer to this level as level 1) and are used to derive more specific constructs of the base and the project-specific method.

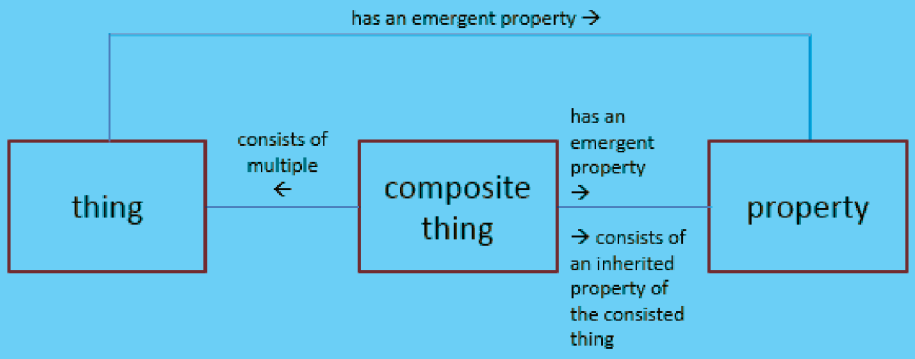


Figure 6: A generic data structure model

3.3 Base method

The base method supports a simple technique for creating user story clauses, which is: *I as a <user role> can <function>*. In line with this template, the constructs *user role* and *function* are defined. *User role* is an instance of *thing*. Since a user is a specific person who interacts with software, we define *user role* as a generic term, which is common to multiple persons in the organization who execute the same set of functions. *Function* is an instance of *property* which represents a business activity that needs to be supported by the new application [2]. *User story,* as an instance of *composite thing*, consists of *function* and *user role*. The list of user stories can also include other information, such as estimates and priorities of user stories [3]. We have not included them in our base method since they are not relevant for understanding execution order or integration dependencies or the elicitation.

Figure 7 shows the newly defined set of so-called user story constructs as instances of previously defined constructs of the generic data structure. The connection between constructs of a different level of abstraction is shown with a dotted line.

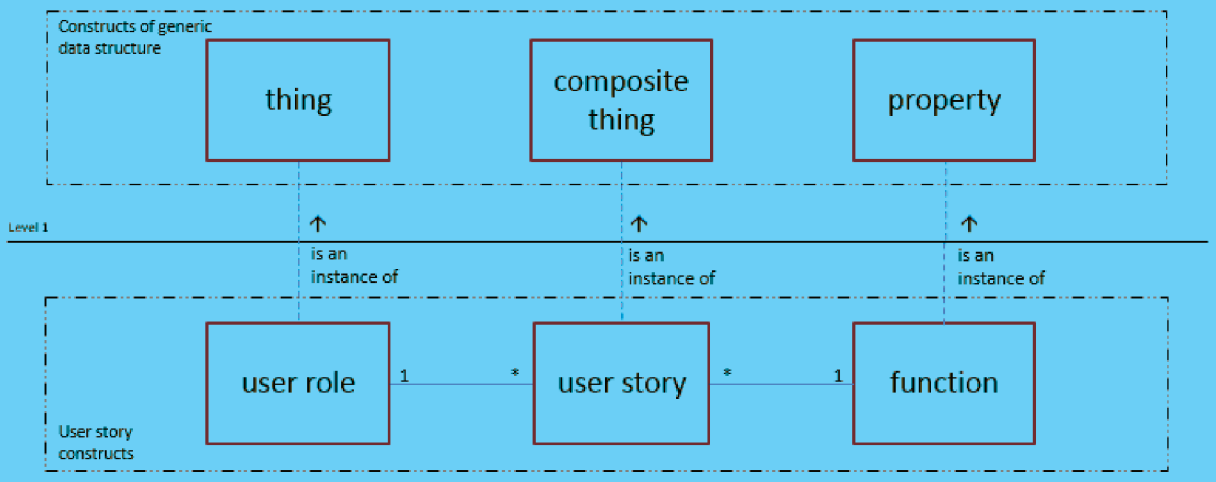


Figure 7: Base method's data structure model

In general, a method includes (beside the data structure) a process for enacting it [1]. We do not specify a base method process because our project-specific process is meant to be applied to any of the existing agile development methods (e.g. Scrum, XP) which utilize user stories.

3.4 Integration of the user story and general business process

In the following, we focus on projects with these characteristics: 1) a given list of user stories is available; 2) corresponding business process models are available; and 3) the execution order and integration dependencies among user stories need to be understood. In order to add a new variation to the base method, a new project-specific method has to be created [44]. We build the project-specific method in such a way that it can be used with any business process modeling notation. In this way, we leave flexibility for the future integration of other types of process models such as EPCs, UML Activity Diagrams, or Petri nets that all build on similar concepts [133]. In our paper, we focus on BPMN models.

A business process integrates information on what needs to be done, who is going to do it, and why [105]. In line with this definition, a set of so-called general business process constructs is depicted in Figure 8. A *who* construct is an organizational role (e.g. a clerk in the front office), a *what* construct is an activity performed by the organizational role for a specific purpose, and a *why* construct is a reason for performing the business process. *Who*, *what,* and *why* are constructs on abstraction level 2 and are instances of constructs on abstraction level 1. The reasons for a particular derivation are discussed later, after we define the constructs on abstraction level 3.

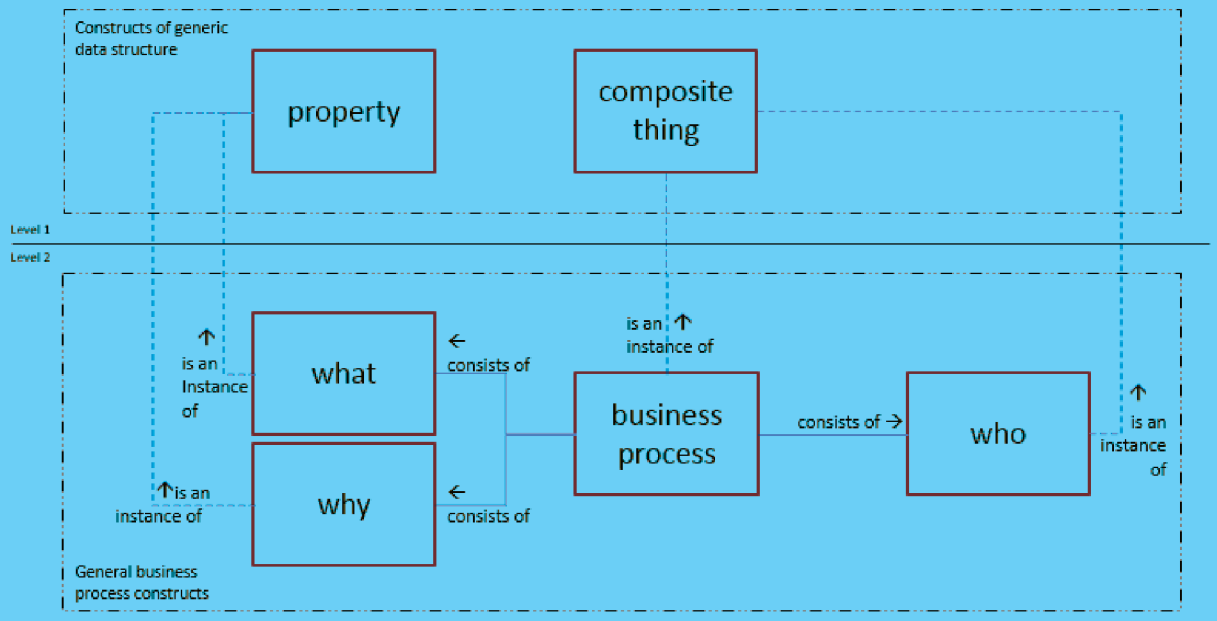


Figure 8: Model of general business process

Figure 7 and Figure 8 provide the foundation for integrating user story constructs with general business process constructs. We integrate *function* (Figure 7) with *what* (Figure 8), and *user role* (Figure 7) with *who* (Figure 8). A specific business process can be partially or fully supported by a new application. Thus, when the function of a user story is intended to support a business activity, a connection between *what* and *function* is made. Similarly, *who* is connected with *user role* if the work of the organizational role is to be supported by at least one requirement of the new application. The integration is presented in Figure 9.

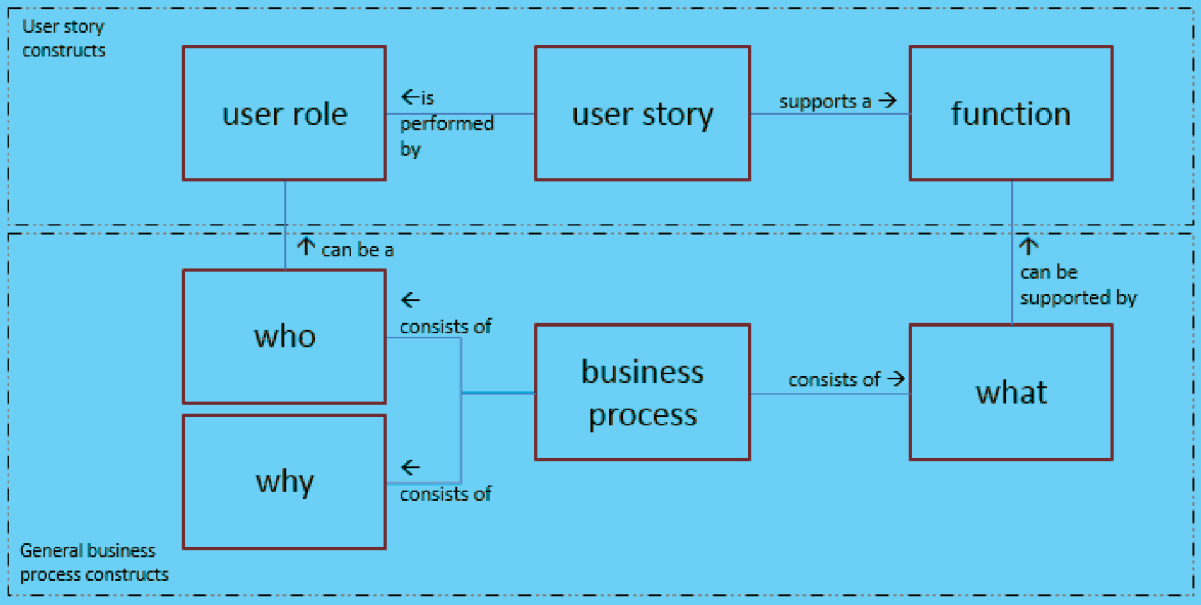


Figure 9: A model of integration of the user story and general business process

With the constructs in Figure 9 we cannot directly connect a specific pair of user story constructs (*user role* and *function*) with a specific pair of the general business process constructs (*who* and *what*) which is relevant for understanding the integration and execution order dependencies among user stories. Therefore, we decided to add constructs of a specific business process notation – BPMN. In addition, good user story conversations are also about the value of a function [27]. Conextra's user story template is used to this end: *I as a <user role> can <function> so that <added value>* [2, 3]. The added value should help to distinguish important user stories from frivolous ones [3]. Nevertheless, the <added value> is easily understood as the <*business value I receive*> where *I* refers to the user role [2]. In some cases, several functions need to be executed so the user role can gain a meaningful value from the application. We argue that a user story should add value to the activities of a business process. Further, the output of the business process provides value to the internal or external customer of the organization [117]. Therefore, we aim to: 1) connect the two pairs discussed at the beginning of the paragraph by using BPMN constructs and by proposing new constructs; and 2) show the value of a user story as proposed above via the additional constructs.

3.5 Project-specific method focused on BPMN model

3.5.1 Integration of the user story and BPMN model

The BPMN grammar includes a plethora of elements, but in practice an average model uses only nine different BPMN elements [134]. For the integration, we need only frequently used BPMN elements, which we use to define a set of so-called BPMN constructs. The derived BPMN constructs represent abstraction level 3 and are related to abstraction level 1 via abstraction level 2.

Swimlane is a graphical container for partitioning a set of activities from other activities [32]. There are two specifications of a swimlane: a pool and a lane [26]. *Pool* represents a participant in a collaboration (e.g. a department). *Lane* is a partition that is used to organize and categorize activities within the pool (e.g. a specific organizational role within a department). Therefore, *swimlane* provides insight into who performs the specific activity. Like Recker et al. [135], we categorize *pool* and *lane* as two things which makes *swimlane* an instance of *composite thing*. Since *swimlane* is an instance of *who*, they are both instances of the same construct on abstraction level 1 – *composite thing*.

Activity can be either an atomic or non-atomic (compound) unit of work that an organization performs in its business processes [32]. Any *activity* consists of multiple *thing* constructs (discussed later in Section 3.4.3) and that makes it an instance of *composite thing*. In addition, it has two properties: *activity description* and *activity type*. The *activity type* construct specifies if the activity is atomic (the activity is not broken down into more detailed activities) or non-atomic (the activity has a collapsed process). *Activity description* is a clause which shortly describes the work of the organizational role. Since *activity description* represents an instance of *what*, they are both instances of the same construct on abstraction level 1 – *property*.

BPMN business process model is a set of activities that represent the steps required to achieve a business objective [32]. *BPMN business process model* has a property called *BPMN name*. *BPMN name* is an instance of the *why* construct. They are both instances of the same construct on abstraction level 1 – *property*. Further, the *BPMN business process model* consists of multiple *thing* constructs (e.g. *activity*) which makes it an instance of *composite thing*. Since *BPMN business process model* is an instance of *business process*, it is an instance of the same construct on abstraction level 1 – *composite thing*.

The use of the discussed BPMN constructs (see Figure 10) facilitates the integration of a specific pair of BPMN constructs, namely *swimlane* (an instance of *who*) and *activity description* (an instance of *what*) with a specific pair of user story constructs, namely the *user role* and *function* constructs. We matched *user role* with *swimlane*, and *function* with *activity description*. Both *user role* and *swimlane* reflect organizational roles, such that they can be matched 1:1. On the other hand, matching *activity description* with *function* is not as simple since a granularity level of *function* in comparison to *activity description* needs to be defined. As suggested by Liskin et al. [25], we have different levels of a user story granularity. Therefore, we define a new set of so-called connecting constructs (which are not part of BPMN constructs): *association* (an instance of *thing*) and *granularity level* (an instance of *property*). *Association* holds information about which *user story* is connected to which *activity*. When associating a user story with a BPMN activity element, the association is attributed according to one of the following user story association granularity levels:

- Level 1: the user story function is more abstract than the business process activity description (e.g. US_299 in Figure 11);
- Level 2: the user story function is approximately equal to the business process activity description (e.g. US_99 in Figure 11); or

- Level 3: the user story function is more detailed than the business process activity description (e.g. US_189 in Figure 11).

Figure 10 depicts the integration of the user story constructs, connection constructs, and BPMN constructs. Only the BPMN constructs on level 3 are connected as instances of constructs on levels 2 and 1.

Figure 11 depicts a business process model enriched with objects of user story references. The objects are visually represented with a text-box object containing a user story code as a label.

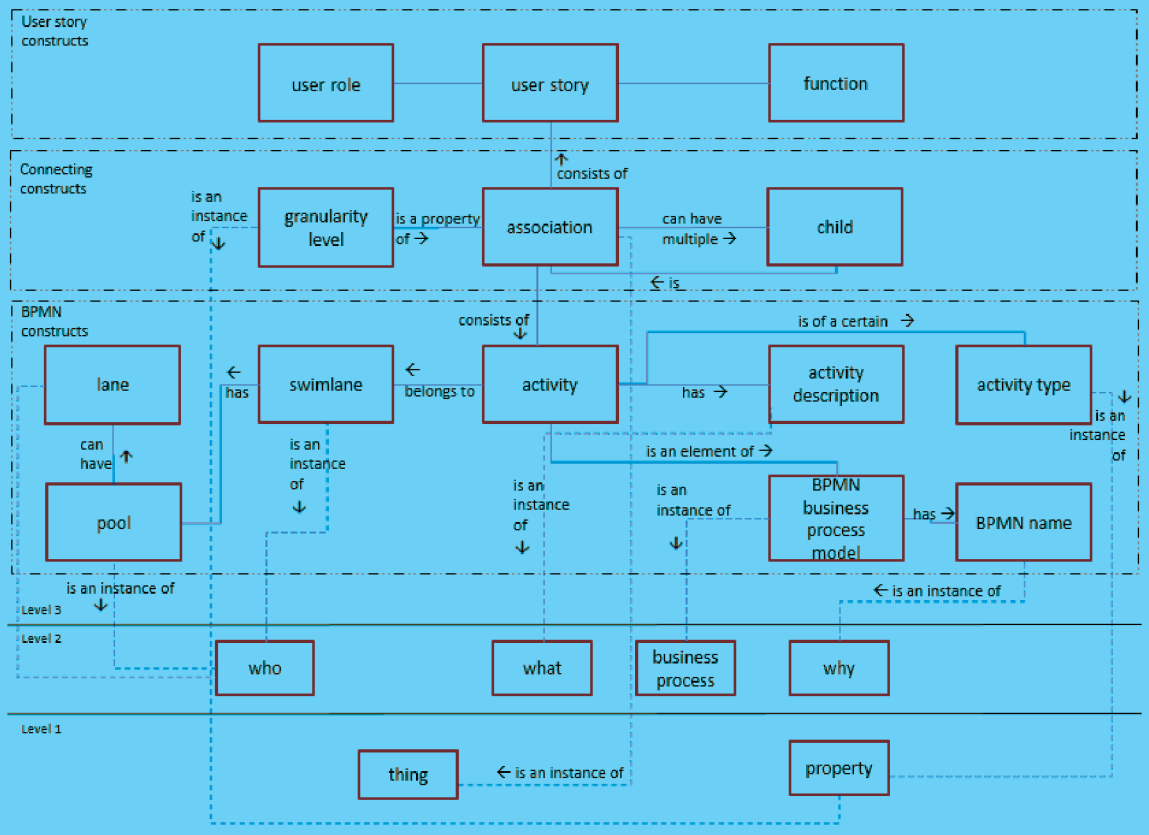


Figure 10: Integration of the user story constructs, connecting constructs and BPMN constructs

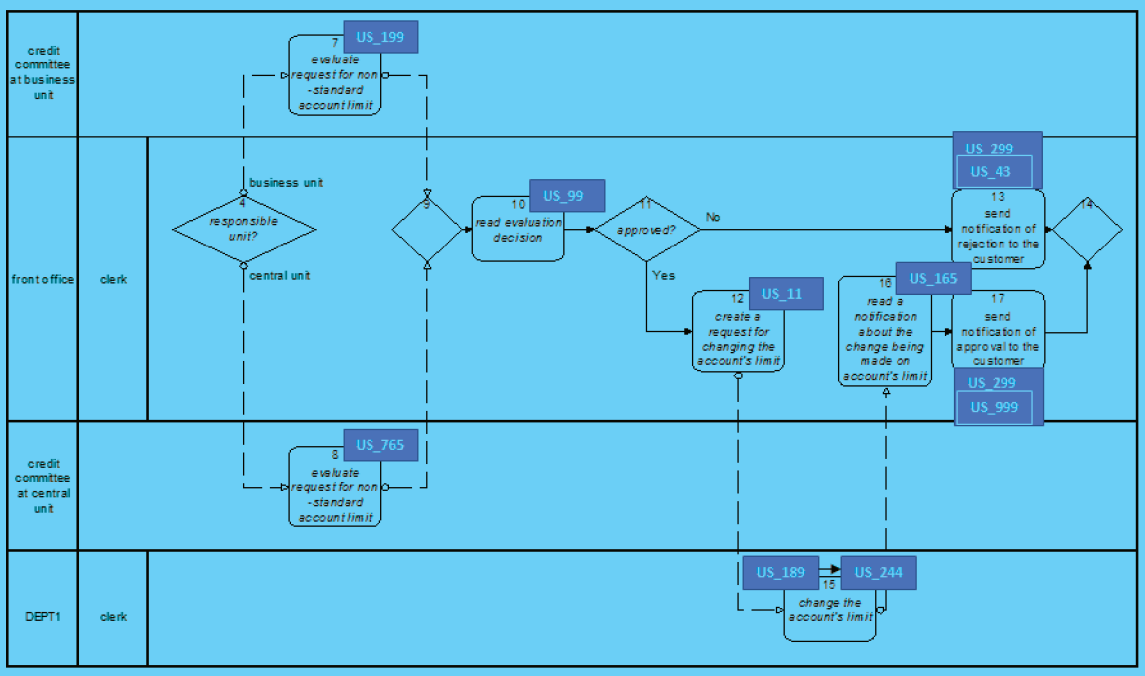


Figure 11: Business process model with associations with user stories

When business process activities are associated with user stories, the value of a specific user story can be analyzed. For example, the function of user story US_244 from Table 2 is: "send a notification about a change in the account's attribute". As mentioned, the association of US_244 is a representative of an association at granularity level 3. The value of the function is to support the performance of a business activity "change the account's limit" (see element number 15 in Figure 11). Moreover, the function adds value to a business process called "getting a non-standard account limit" since it supports the process execution.

3.5.2 Technique for eliciting user stories from a BPMN model

In order to create user stories, values for constructs *user role* and *function* are needed. A general business process can offer those values from constructs *who* and *what*. Since a specific business process model created with BPMN notation is available the values can be we obtained from the constructs *swimlane* and *activity description*. Our goal is to have a list of user story clauses generated by template: *I as <user role> can <function>*. Therefore, we need to create simple textual descriptions from BPMN models. Leopold et al. [136] have proposed an architecture for automatic generation of textual descriptions from visual business process models. The architecture is very comprehensive since if focuses on linguistic needs for automatic creation of larger amount of text such as text planning, sentence planning, surface realization and flexibility. Since we are creating individual user story clauses we need a simplified version of the architecture. So we have

adopted some of the architecture's elements in our proposed elicitation technique's steps. When eliciting user stories we distinguish three sub-elicitations: a basic sub-elicitation, advanced higher-level sub-elicitation, and advanced lower-level sub-elicitation.

BPMN model is the input to the basic sub-elicitation. Our assumption is that the model is already final and modeled on such level of detail that the new application is not expected to change the flow or description of activities. If this is not the case we need to make a to-be version of the business process. In order to elicit user stories from BPMN model we need to follow these steps:

Step 1: *Preparation of a BPMN model*. On the existing BPMN model (see example Figure 5) we need to mark those activities which are going to be influenced by the new application. Mark them for example with blue boxes (e.g. see blue boxes in Figure 11). In blue boxes write down US reference codes (e.g. US_16).

Step 2: *Preparation of user story's syntactic trees*. For each reference code create a user story clause by using the template: I as <user role> can perform <function>. In order to fill in the gaps <user role> and <function> we firstly need to perform *linguistic information extraction* from the BPMN model [136]. A goal in our technique is to annotate activities. For instance, in order to make a syntactic tree for user story reference code US_16, the associated activity "fill in a form to close e-banking" has to be annotated. We annotate it with the action (A) "fill in", and the business object (BO) "a form to close e-banking". Furthermore, we additionally annotate each activity with the line (L) (e.g. "clerk"), and a pool (P) (e.g. front office) that it belongs to. The result of this step is a syntactic tree for each user story reference code. The tree is used to represent the most significant aspects of the syntactic structure of the clause [136]. See Figure 12.

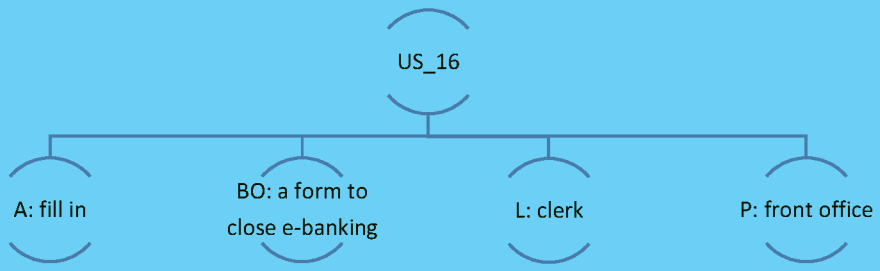


Figure 12: An example of a syntactic tree for user story UC_16 from BPMN model

Step 3: *Creation of user story clauses*. Each syntactic tree needs to be transformed into user story clause by using the mentioned template. The literature offers several

off-the-shelf realizers which directly transform trees into grammatically correct clauses (see e.g. [137, 138]). User story clauses need to follow the user story's template. Since in practice the activity descriptions sometimes refer to business objects with synonyms, shorten names etc. (e.g. US_12's function is "close it" where "it" refers to e-banking) we can choose to constantly use the same term for the same thing.

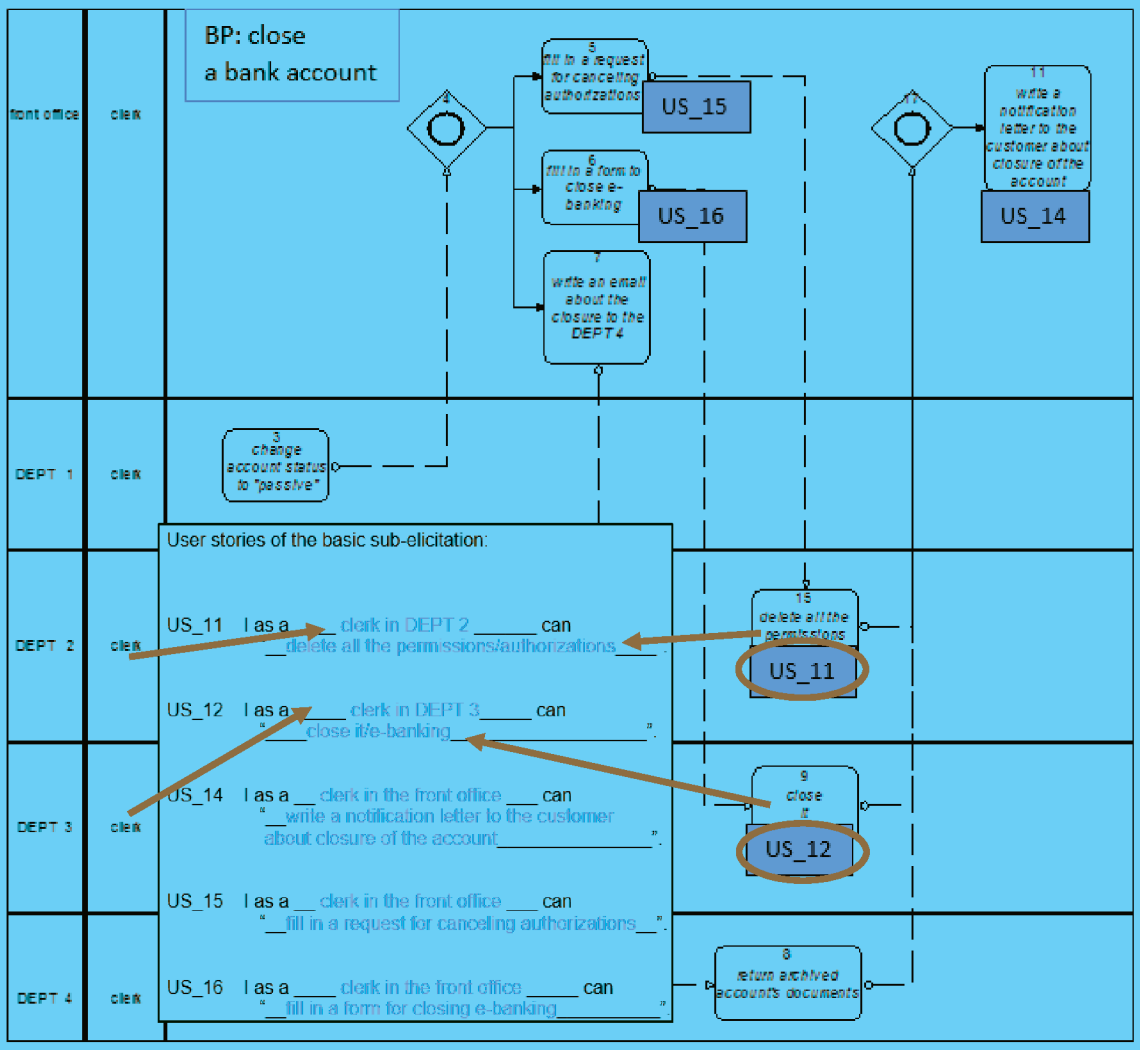


Figure 13: An example of basic sub-elicitation of user stories from BPMN model

The output there is a list of user stories and BPMN model associated with user stories. The associations are attributed with association granularity level 2.

An input to advanced higher-level sub-elicitation is the list of additional user stories which are more general than those created from the basic sub-elicitation. Therefore, the input of the advanced higher-level sub-elicitation is a list of user stories created in the basic sub-elicitation. As suggested by Leffingwell [2] we aim to create higher-level (more abstract, more general) user stories in order to start forming a tree hierarchy of user

stories. This is believed to help to make a better initial design of the future application. In order to elicit higher-level user stories we need to follow these steps:

Step 1: *Creation of higher-level user stories.* For each user story created during the basic elicitation we generalize the activity description (inserted in <function>) either by generalizing verb or noun or both of them. The <user role> value is rewritten from the user story created by the basic sub-elicitation.

Step 2: *Group several user stories* (if possible) from the basic sub-elicitation under their common general user story, and archive each connection as integration dependency among two user stories of different granularities. Figure 14 shows an example of how to do advanced high-level sub-elicitation, while Figure 15 shows user story references from the basic (references colored in blue) and advanced higher-level (references colored in yellow) sub-elicitation in the BPMN model.

Step 3: *Make further higher-level user stories* from the list of user stories on the right hand side of Figure 14. At some point we can realize that several user stories share the same function (e.g. US_17 and US_18). It makes sense to make higher-level user story US_21 which unifies those two user stories with a list of user roles which share same function. This information can be valuable for application's design and for coding activities (e.g. less recoding which typically consumes a lot of development project's resources). The new higher-level user stories are:

US_21 We as clerk in DEPT 2 and clerk in DEPT 3 can "edit account's attribute".

US_22 I as a clerk in the front office can "write internal and external notifications".

US_23 I as a clerk in the front office can "fill in a form".

On the root of this upcoming tree hierarchy we put a BPMN model's name, which has a role of an epic:

US_epic1 We as clerk in the front office, clerk in DEPT 2 and clerk in DEPT 3 can "close a bank account".

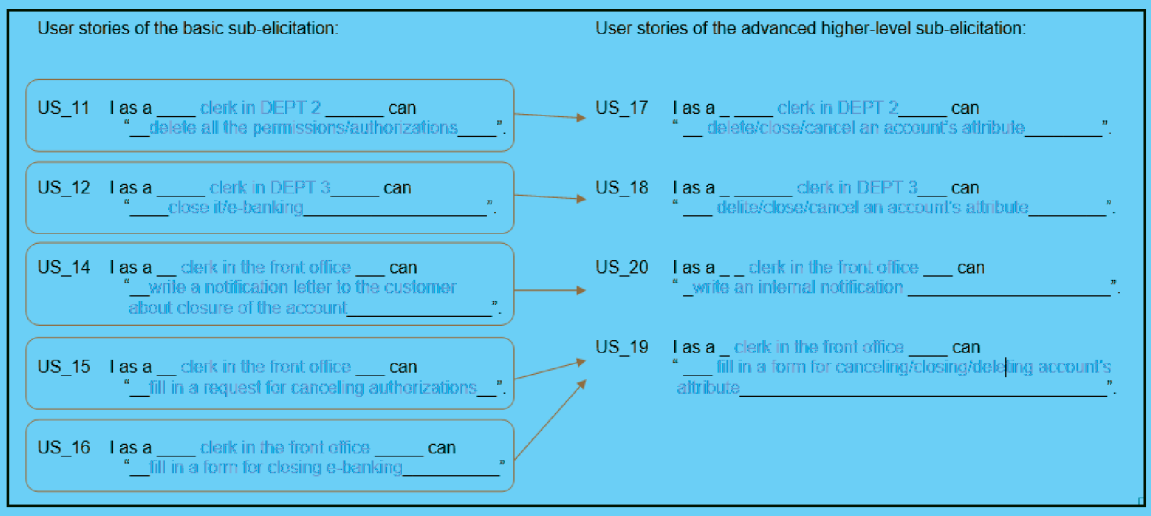


Figure 14: An example of advanced higher-level sub-elicitation of user stories from BPMN model

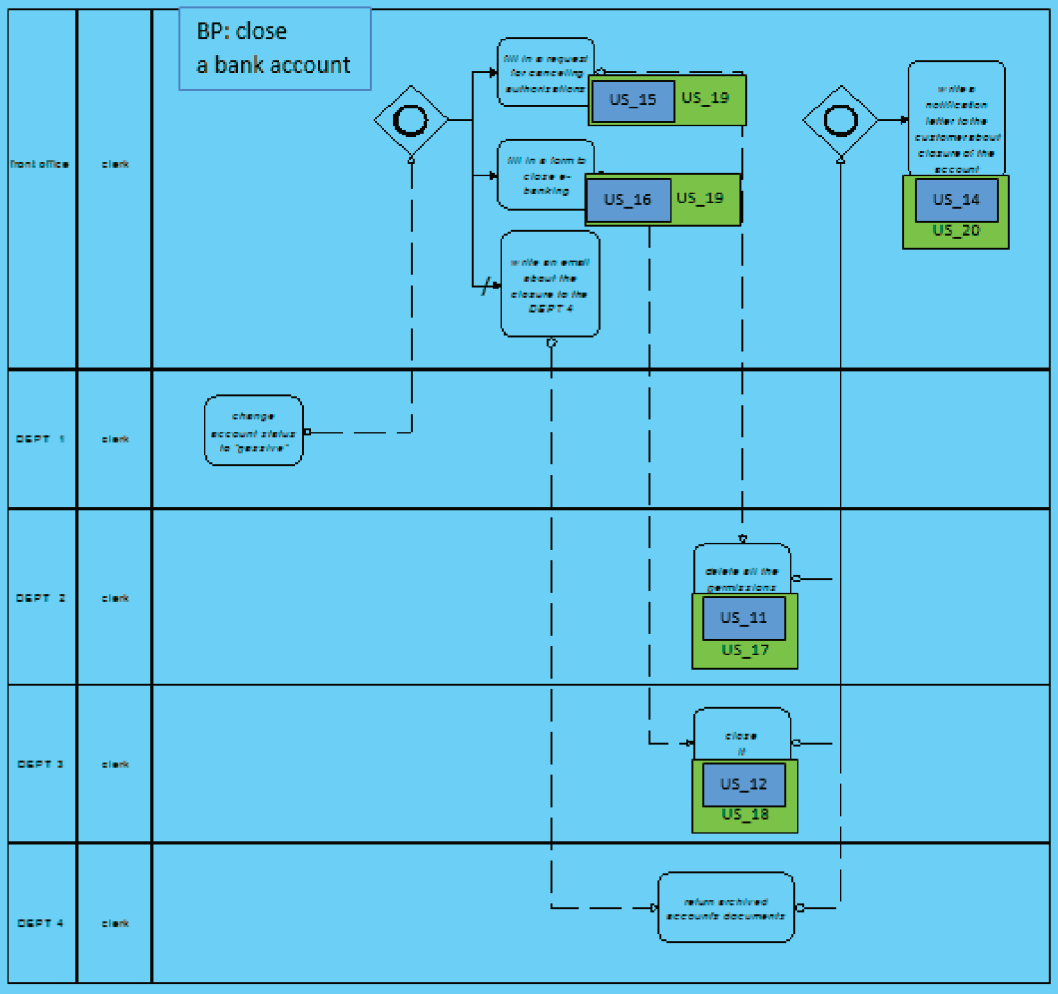


Figure 15: Results of basic and advanced higher-level sub-elicitation from BPMN model

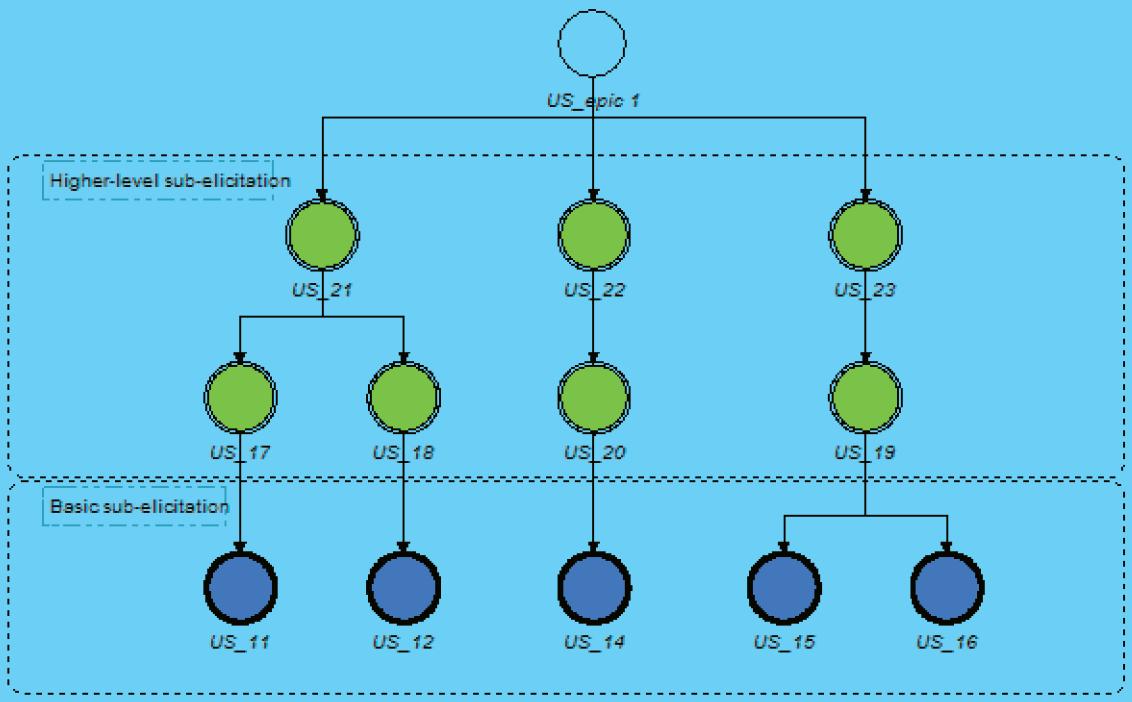


Figure 16: Tree-hierarchy of user stories created by basic and higher-level sub-elicitation

The output is a list of user stories and BPMN model associated with user stories. The associations are attributed with association granularity level 1 and 2.

We recommend that if user stories with associations of granularity level 1 occur in multiple sub-granularity layers as in Figure 16, we group them in sub-groups. For example, US_17 – US_20 are attributed with association granularity level 1.1, while US_21 – US_23 are attributed with association granularity level 1.2.

An input to advanced lower-level sub-elicitation is the list of additional user stories which are more detailed than those created from the basic sub-elicitation. Therefore, on the input of the advanced lower-level sub-elicitation is a list of user stories created in the basic sub-elicitation. As suggested by Leffingwell [2] we aim to create lower-level (more detailed) user stories in order to finish forming a tree hierarchy of user stories.

The advanced lower-level sub-elicitation starts with making estimates about resources needed for implementation of an individual user stories created in the basic-sub elicitation. In this thesis we choose not to specify how the estimates should be done. For this reason we make an assumption that the estimates are given from the development team. If the estimates show that a particular user story is too big to fit in one development iteration (e.g. US_12), its function should to be treated as an individual (to-be) sub-business process which needs to be modeled in a chosen notation. When having the new

sub-business process model (see e.g. Figure 17) available we can recursively start the basic and advanced higher-level sub-elicitation. The result is a sub-tree-hierarchy of user stories where the root is presented with user story which was previously declared as too big (e.g. US_12). The sub-tree hierarchy is integrated in the initial tree-hierarchy as presented in Figure 16. The development team makes the estimates of the new user stories from the basic-elicitation. If one of them is still too big to fit in one development iteration, we again need to create a business process model for it and start the elicitation.

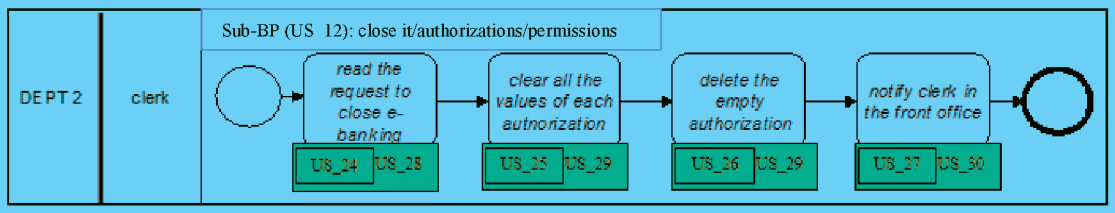


Figure 17: Sub-business process model for user story US_12

When all the estimates for user stories with bolded circles on Figure 18 are indicating that the user stories are of acceptable size to fit in development iterations, the customer side representatives are invited to set priorities to the user stories. We recommend that the priorities are firstly given to a chosen epic user story, because it represents the business value of a business process. If the estimate of the epic shows that the epic is too big then a sub-tree hierarchy is chosen because it represents a business value of a sub-business process. When there is no sub-tree hierarchies then priorities are given to the user stories which have a common higher-level user story.

The output is a list of user stories and BPMN model associated with user stories. The associations are attributed with association granularity level 1, 2 and 3.

We recommend that if user stories with associations of granularity level 3 occur in multiple sub-granularity layers as in Figure 18, we group them in sub-groups. For example, US_28 – US_30 are attributed with association granularity level 3.1, while US_24 – US_27 are attributed with association granularity level 3.2.

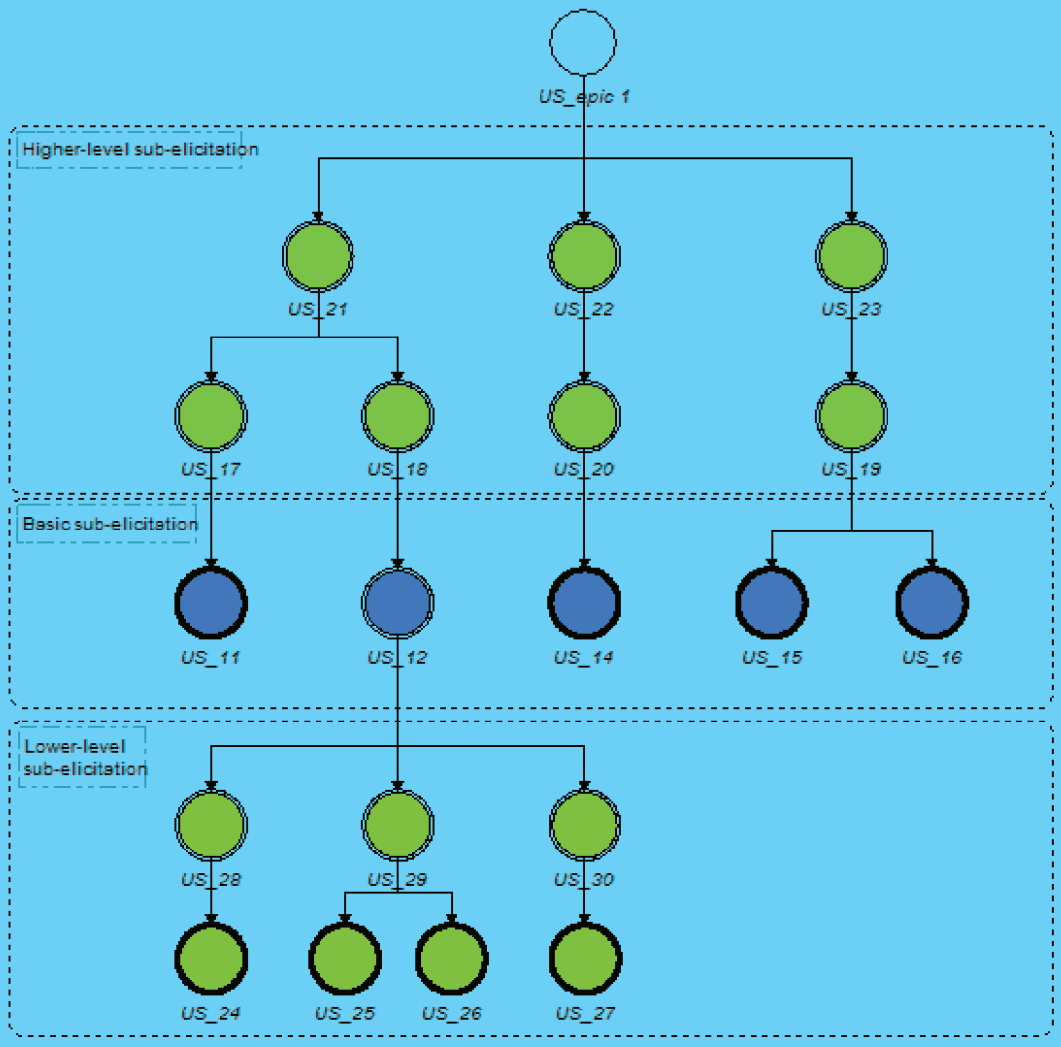


Figure 18: Tree-hierarchy of user stories created by basic, higher-level, and lower-level sub-elicitation

3.5.3 Technique for tracing execution order dependencies by using BPMN models

A user story can be influenced not just by a directly preceding user story but also by those which need to be executed before the previous one. This means that, to understand the execution order dependencies of a user story, one needs to be aware of different execution orders which lead to the execution of a particular user story. Execution orders are managed by flow and gateway elements and, therefore, the set of BPMN constructs needs to be extended with additional constructs.

The flow element has two specializations: a sequence flow (an arrow with a full line) and a message flow (an arrow with a dotted line). A sequence flow element represents the sequence of activities within the same lane, and a message flow element represents the transmission of a message between two activities which are in different lanes [32]. Therefore, a construct called *flow* represents a sequence of activities while its property

flow type shows to which of the two specializations a specific flow belongs. In addition, *flow* has two emergent properties: *start point BPMN element* and *end point BPMN element.* According to the modeling rules [139], each activity has exactly one input and one output flow. As such, *flow* is a composite of *activity*.

Gateways define the flow between activities [32]. Therefore, a construct called *gateway* represents information about possible directions of the flow. A BPMN model can contain different specializations of a gateway, such as XOR [32]. According to the modeling rules [139], each split gateway should have a corresponding join gateway. We have defined three properties of *gateway*, namely, *gateway type* to characterize the specialization, *gateway decision description* to characterize what is the decision about, and *split/join* to characterize the decision where necessary (namely, at split gateways). As such, *gateway* is an instance of *thing*, and also a composite of both *BPMN business process model* and *flow*. Since it is a composite of *flow*, this makes *flow* an instance of *composite thing*.

The following examples explain the process of *a technique for tracing execution order user stories by using BPMN models*. The example is focused on discovering execution orders that trigger the execution of US_299 (see list of user stories in Table 2). The business process model in Figure 11 shows that the directly preceding user story of US_299 can be either US_165 or US_11. US_165 is directly preceded by US_244 which is preceded by US_189. US_189 is directly preceded by US_11 which is directly preceded by either US_199 or US_765. If the preceded activity element is not associated with any user story, this indicates the end of the execution order. Therefore, the direct and indirect execution order dependencies of US_165 can be described by four different execution orders which are the output of the technique:

1) US_199 → US_11 → US_299
2) US_765 → US_11 → US_299
3) US_199 → US_11 → US_189 → US_244 → US_165 → US_299
4) US_765 → US_11 → US_189 → US_244 → US_165 → US_299

3.5.4 Technique for tracing integration dependencies by using BPMN models

Understanding integration dependencies requires information about the hierarchy relation of the user stories on the list; for example, which user story represents a composite of a higher-level (more abstract) user story. We can visually represent the composition in many ways. Figure 11 shows a technique nesting of association objects where the more detailed user story association object is visually smaller and within the bigger, more abstract user story association object. US_299 is more abstract than US_43 and US_999.

The two user stories represent specific execution variations of US_299. For managing the integration dependencies, we suggest using the two connecting constructs that we have already defined, namely, *association* and *granularity level*, and an additional property called *child*. This additional property shows the user stories that compose a specific higher-level user story. The output of the technique is a tree-hierarchy of user stories (e.g. Figure 18) and/or an associated business process model (e.g. Figure 15) which informs the reader about integration dependencies related to a specific business process model.

Figure 19 shows a current set of a BuPUS constructs at abstraction level 3. The constructs are grouped according to their relevance for understanding integration and execution order dependencies among user stories.

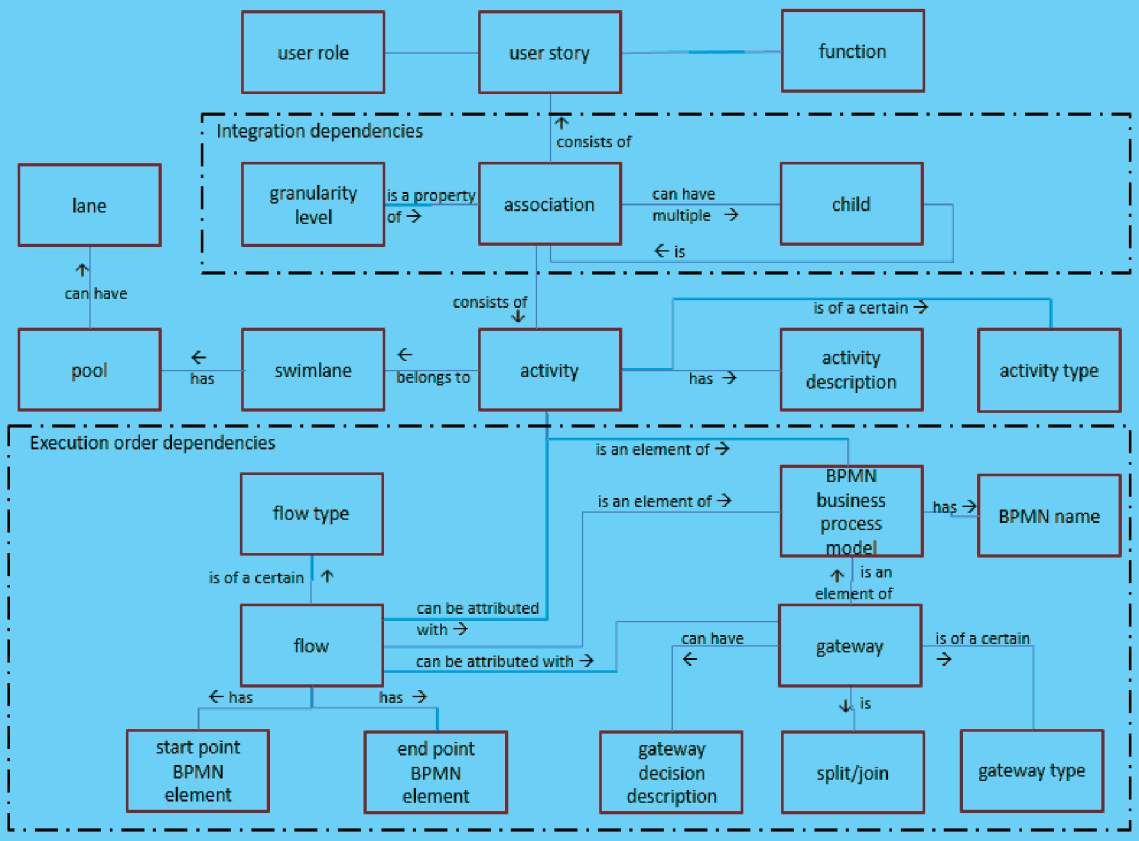


Figure 19: A model of integration of the user story and BPMN model

An overview of how the constructs on abstraction level 3 derive from the constructs on abstraction levels 2 and 1 is presented in Table 5.

Constructs from level 1	Constructs from level 2	Constructs from level 3
thing	-	user role
	-	association
	-	gateway
composite thing	-	flow
	who	swimlane
	-	activity
	business process	BPMN business process model
	-	user story
property	why	BPMN name
	-	granularity level
	what	activity description
	-	activity type
	-	flow type
	-	gateway type
	-	split/join
	-	gateway decision description
	-	start point BPMN element
	-	end point BPMN element
	-	child
	-	function

Table 5: Derivation relationships for project-specific method focused on BPMN model

3.6 Project-specific method focused on text-written use case model

3.6.1 Integration of the use case constructs with text-written use case constructs

Text-written use case model is a textual business process model [31]. Its comprehensive template is proposed by *Use-Case Specification* promoted by Rational Unified Process. Nevertheless, it can be captured at various levels of precision [8]. A typical detailed text-written use case model describes what the user does and what the application does in response. Since in our method we refer to a text-written use case model as a representative of textual business process models we assume that these models do not describe what the application does in response. Furthermore, we have adopted only those sections of the text-written use case model which are relevant for the elicitation of user stories and for the understanding of the dependencies, namely, Basic flow of events, Alternative flows of events, and Additional information.

Basic flow of events is a textual description of the events which are performed one after another [2]. When one event leads to a decision point where under certain conditions a

sub set of events is triggered, we call that sub-set of events an alternative flow. All alternative flows are gathered in the section called Alternative flow of events and triggered from the section Basic flow of events. The development team needs to think about all the "what ifs" that might affect the future flow of events [2]. While sections Basic flow of events and Alternative flow of events are needed for both the elicitation and understanding of the dependencies, the section Additional information needed only to support understanding of the dependencies. The information gives important details about when a specific alternative flow can/must be triggered.

Since we defined our text-written use case model as a textual business process model we propose an integration of the general business process constructs with new text-written use case constructs, namely, *use case model, use case name, event, event description, event type,* and *actor*. *Use case model* is an instance of *business process* which integrates information on what needs to be done, who is going to do it, and why [105],[31]. *Use case name* is defining a goal of events [2] and as such presents a reason for performing the business process. Furthermore, *use case name* is an instance of *why.* An *event* consists of multiple *thing* and *property* constructs and that makes it an instance of *composite thing*. Each *event* (either from basic or alternative flow section) is equipped with following properties: *use case name, event description*, *event type*, and *actor*. *Event description* construct is an instance of *what* and as such it is an activity performed by the organizational role. *Event type* is an instance of *property* which holds information about the flow of events the specific event belongs to (e.g. basic flow of events, or alternative flow A). *Actor* is a participant (a user role) who interacts with the application [2]. It is an instance of *who* and as such it presents an organizational role (e.g. a clerk in the front office).

Association integrates a *user story* with an *event* (see Figure 20). When associating a user story with an event, the association is attributed according to one of the following (previously mentioned) user story association granularity levels:

- Level 1: the user story function is more abstract than the business process activity description (e.g. yellow colored user story references in Figure 21);
- Level 2: the user story function is approximately equal to the business process activity description (e.g. blue colored user story references in Figure 21); or
- Level 3: the user story function is more detailed than the business process activity description.

Figure 20 shows the constructs and how they fit in existing BuPUS method third level data structure.

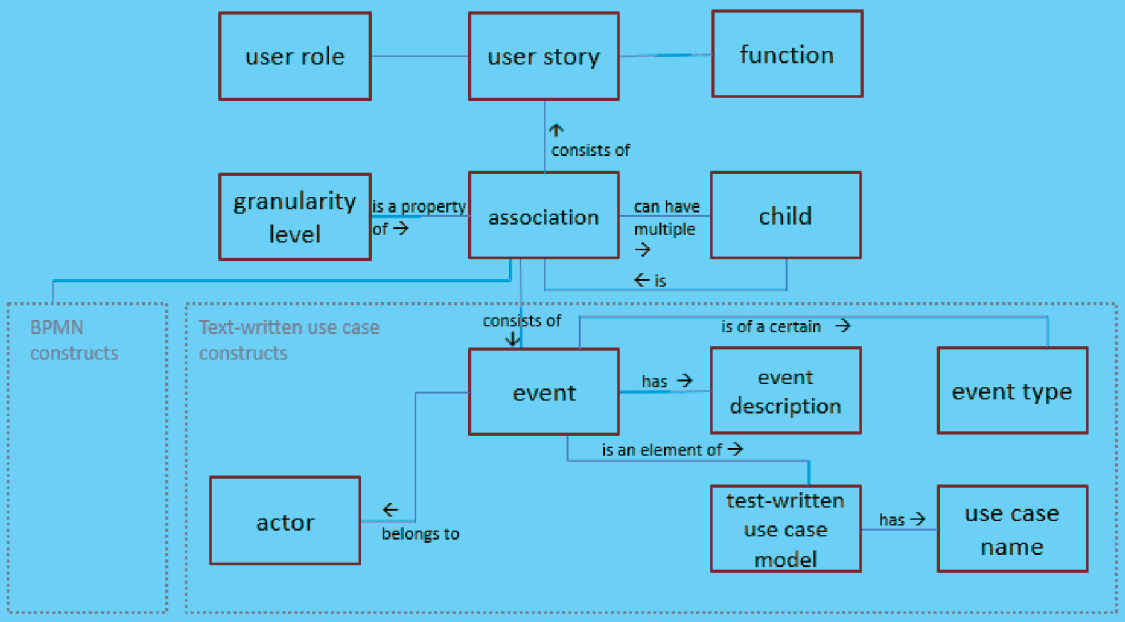


Figure 20: An integration of the user story constructs, connection constructs, and text-written use case constructs

Figure 21 gives an example of a text-written use case model enriched with user story references (see the last two columns in tables).

Each row in the Alternative flows of events is a candidate for user story. Each row in the section Basic flow of events is not though. We make an assumption that rows in section Basic flow of events can also contain note (e.g. Note 1) and decision description (e.g. Decision #1) references to text in section Additional information. We use these references to enable informational equivalence with the BPMN model where visual elements give information about the business environment which are important for understanding of user stories. Table 6 reports about reasons for referencing to section Additional information.

Use case name: CLOSE A BANK ACCOUNT

Basic flow of events

Actor	Event	US	US (+1)
Clerk in the front office	Note 1 Decision #1		
Clerk in the front office	Note 2 Note 3 Write a notification letter about closure of the account to the customer.	US_14	US_20

Alternative flows of events

Alt. flow	Actor	Event	US	US (+1)
A	Clerk in the front office	Fill in a request for cancelling authorizations	US_15	US_19
	Clerk in DEPT 2	Delete all the permissions	US_11	US_17
B	Clerk in the front office	Fill in a form to close e-banking	US_16	US_19
	Clerk in DEPT 3	Close it	US_12	US_18

Additional information:

Note 1: After the clerk in DEPT 1 changes the status of customer's account to "passive" the clerk in the front office needs to close all open businesses, such as e-banking and authorizations.

Decision #1: According to the characteristics of the account none, one (A or B) or two alternative flows (A and B) are triggered

Note 2: No matter how many alternative flows of Decision #1 are triggered, clerk in the front office always writes an email about the closure to the DEPT 4 who needs to send him/her all of the account's physical documents.

Note 3: The event can start after opened authorizations are closed, opened e-banking is closed and all the account's physical documents are handed to the clerk in the front office.

Figure 21: A text-written use case model associated with user story references

Informing the reader about …	BPMN model	text-written use case model
… the conditions in which a certain set of alternative flows can be started	OR split gateway element	Text in a decision note (e.g. Decision #1)
… the conditions in which the event can be started	OR merge gateway element	Text in a general note (e.g. Note 3)
	Manual activity element	Text in a general note (e.g. Note 2)
	Activity elements supported by other application	Text in a general note (e.g. Note 1)

Table 6: Supporting information equivalence of BPMN model and text-written use case model

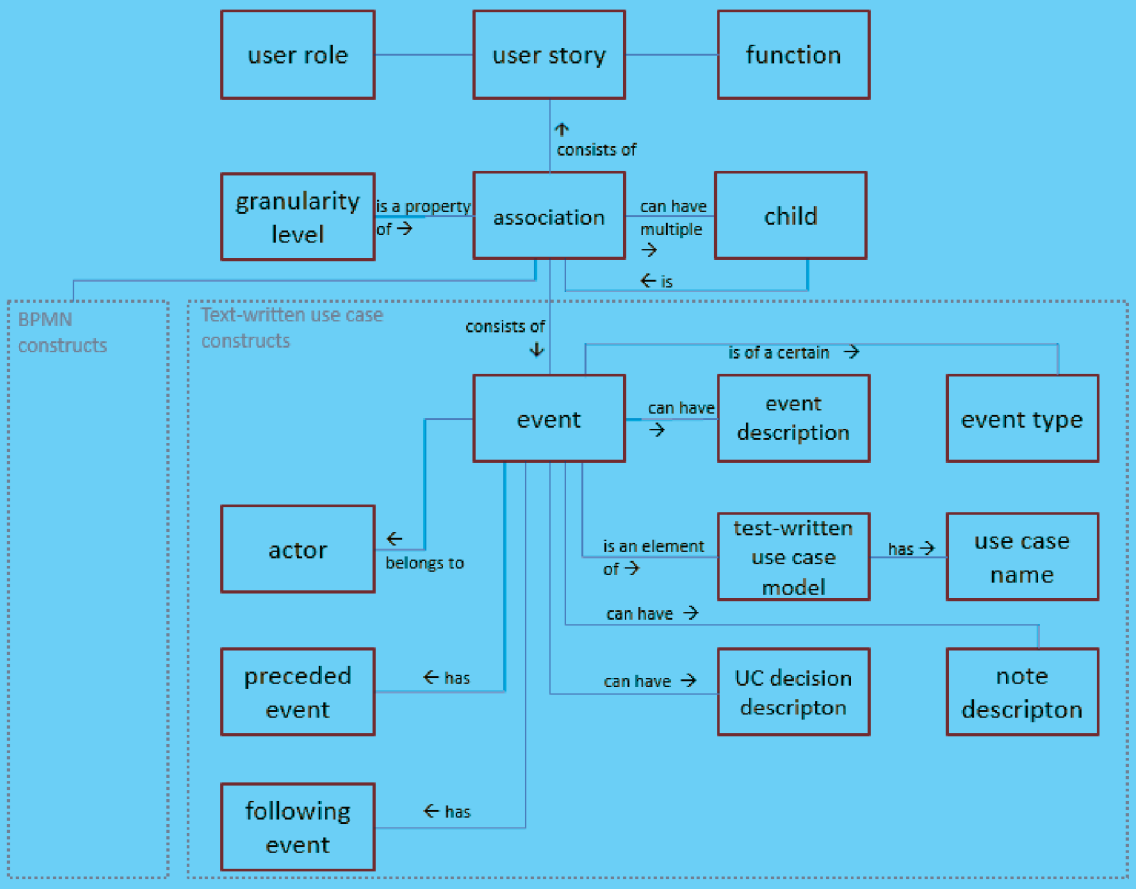


Figure 22: A model of integration of the user story and text-written use case model

In order to support references to section Additional information we propose two additional constructs, namely, *UC decision description* and *note description*. They are both properties of *event*. *UC decision description* is a textual description about which alternative flows can be started with certain conditions, while *note description* is a textual

description about the conditions of starting the event. The new constructs influence *event type* by adding its new "empty" value.

3.6.2 Technique for eliciting user stories from text-written use case models

In order to create user story clauses we need to get values for constructs *user role* and *function*. As mentioned, general business process can offer those values from constructs *who* and *what*. Since we have a text-written use case model we obtain the values from constructs *actor* and *event description*. A goal is to have a list of user story clauses generated by template: *I as <user role> can <function>*. Therefore, we need to create simple textual descriptions from rows in the tables present in sections Basic flow of events and Alternative flows of events. Again, we take advantage of Leopold et al. [136] architecture for automatic generation of textual descriptions. We adapted some of the architecture's elements in our proposed elicitation technique's steps. When eliciting user stories with a text-written use case model we distinguish three sub-elicitations (the same was done with BPMN model in Section 3.5.2): a basic sub-elicitation, advanced higher-level sub-elicitation, and advanced lower-level sub-elicitation.

A text-written use case model is the input to the basic sub-elicitation. We assume that the mandatory sections of the model, which we previously discussed, are available. We do not make any assumptions on how detailed the text-written use case model is. In order to elicit user stories from text-written use case model we need to follow these steps:

Step 1: *Preparation of a text-written use case model*. On the existing text-written use case model (see example Figure 4) we need to create tables in sections Basic flow of events and Alternative flows of events. The rows give information about: what needs to be performed (event description), who executes the event (actor) and which user story code will be used as a reference to a user story clause. Additionally, we use reference notes to section Additional information to clearly describe business environment of events. We propose to mark the user story references, for example, with blue text coloring (see blue boxes in Figure 11).

Step 2: *Preparation of user story's syntactic trees***.** For each reference code we need to create a user story clause by using the template: I as <user role> can perform <function>. In order to fill in the gaps <user role> and <function> we firstly need to perform *linguistic information extraction* from the text-written use case model [136]. A goal of our technique is to annotate event descriptions. For instance, in order to make a syntactic tree for user story reference code US_16, the associated

event description “fill in a form to close e-banking” has to be annotated. We annotate it with the action (A) “fill in”, and the business object (BO) “a form to close e-banking”. Furthermore, we additionally annotate each event description with the actor (AC) (e.g. clerk in the front office). The result of this step is a syntactic tree for each user story reference code. The tree is used to represent the most significant aspects of the syntactic structure of the clause [136]. See Figure 23.

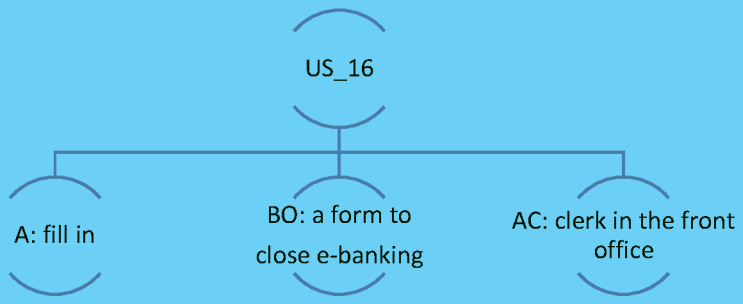


Figure 23: An example of a syntactic tree for user story UC_16 from text-written use case model

Step 3: *Creation of user story clauses*. Each syntactic tree needs to be transformed into user story clause by using the mentioned template.

The output of the basic sub-elicitation is a list of user stories and text-written use case model associated with user stories. The associations are attributed with association granularity level 2.

An input to advanced higher-level sub-elicitation is the list of additional user stories which are more general than those created from the basic sub-elicitation. Step 1 (creation of higher-level user stories) and 2 (group several user stories) from this elicitation are performed the same way as in previously described technique for eliciting user stories from the BPMN model (please refer to Section 3.4.3). Figure 21 shows visual representations of user story references from basic and advanced higher-level sub-elicitation placed in the text-written use case model. In the Step 3 we make further higher-level user stories. For demonstration we created user stories, namely, US_21 – US_23, and US_epic1, which are the same as those ones created in Step 3 of higher-level sub-elicitation from text-written BPMN model. Consequently, we created the same tree-hierarchy of user stories too (see Figure 16).

Use case name: CLOSE A BANK ACCOUNT

Basic flow of events

Actor	Event	US
Clerk in the front office	Note 1 Decision #1	
Clerk in the front office	Note 2 Note 3 Write a notification letter about closure of the account to the customer.	US_14

Alternative flows of events

Alt. flow	Actor	Event	US
A	Clerk in the front office	Fill in a request for cancelling authorizations	US_15
	Clerk in DEPT 2	Delete all the permissions	US_11
B	Clerk in the front office	Fill in a form to close e-banking	US_16
	Clerk in DEPT 3	Close it	US_12

Additional informati

Note 1: After the cler "passive" the clerk in the front nking and authorizations.

Decision #1: Accord B) or two alternative flows (A

Note 2: No matter h d, clerk in the front office alwa o needs to send him/her all of t

Note 3: The event ca banking is closed and all the ac the front office.

US_11 I as a ___clerk in DEPT 2_____ can "_____delete all the permissions/authorizations____".

US_12 I as a ___ clerk in DEPT 3_____ can "_____close it/e-banking_____".

US_14 I as a ____clerk in the front office _____ can "_____write a notification letter to the customer about cloasure of the account_____".

US_15 I as a __ clerk in the front office _____ can "_____fill in a request for canceling authorizations___".

US_16 I as a ___ clerk in the front office _____ can "_____fill in a form for closing e-banking _____"

Figure 24: An example of basic sub-elicitation of user stories from text-written use case model

An input to advanced lower-level sub-elicitation is a list of user stories created in the basic sub-elicitation. We create new lower-level user stories in the same way as described in advanced lower-level sub-elicitation from the BPMN model (please refer to Section 3.5.2). In Figure 25 we created a new text-written use case model which is informational equivalent to BPMN model in Figure 17. Also, tree-hierarchy of user stories created by basic, higher-level, and lower-level sub-elicitation is the same and presented in Figure 18.

Use case name: close it/authorizations/permissions

Basic flow of events

Actor	Event	US (-1.1)	US (-1.2)
Clerk in DEPT 2	Read the request to close e-banking	US_24	US_28
Clerk in DEPT 2	Clear all the values of each authorization	US_25	US_29
Clerk in DEPT 2	Delete the empty authorization	US_26	US_29
Clerk in DEPT 2	Notify clerk in the front office	US_27	US_30

Figure 25: Text-written use case model for user story US_12

3.6.3 Technique for tracing execution order dependencies by using text-written use case models

Same as in Section 3.5.4 we argue that a user story can be influenced not just by a directly preceding user story but also by those which need to be executed before the previous one. In order to trace execution order dependencies among user stories in the case of text-written use case model, we propose new constructs: a *preceding event*, and *following event. Preceding event* is an event which occurred just before, and the following event is the one which will happen immediately after a certain event. For example, in Figure 25 there is a reference to US_25. The user story is associated to the event description "clear all the values of each authorization". The preceding event has description "read the request to close e-banking". And the following event has description "delete the empty authorization." Another example: in Figure 21 there is a reference to user story US_12. The event it belongs to is preceded by an event with description "fill in a form to close e-banking" and followed by an event with description "write a notification letter about closure of the account to the customer".

The following example describes *a technique for tracing execution order user stories by using text-written use case models* thru an example. The example is focused on discovering execution orders that trigger the execution of US_14 (see Figure 24). The business process model shows that the directly preceding user story of US_14 can be either US_11 or US_12. US_11 is directly preceded by US_15, while US_12 is directly preceded by US_16. Therefore, the direct and indirect execution order dependencies of US_14 can be described by two different execution orders which are the output of the technique:

1) US_15 → US_11 →US_14
2) US_16 →US_12 → US_14

3.6.4 Technique for tracing integration dependencies by using text-written use case models

Understanding integration dependencies requires information about the hierarchy relation of the user stories on the list such as the one presented in Figure 18. We can visually represent the hierarchy in another way too – with columns added to the text-written use case model. Figure 21 shows two tables. The last two columns of the tables contain references to the user stories. The column with blue colored text has references to user stories with association granularity level 2, and the column with yellow colored text has references to user stories with granularity level 1. Figure 25 also shows two columns we marked with green to clearly show that they belong to the association granularity level 3, and that there is one group of user stories (US_28 - US_30) which is more general than the other (US_24 – US_27). For managing the integration dependencies, we suggest using the three connecting constructs that we have already defined, namely, *association, granularity level*, and *child*.

Figure 21 shows a set of a BuPUS constructs at abstraction level 3 focused on tracing integration and execution order dependencies among user stories with text-written use case model.

An overview of the text-written use case constructs on abstraction level 3 derive from the constructs on abstraction levels 2 and 1 as presented in Table 7.

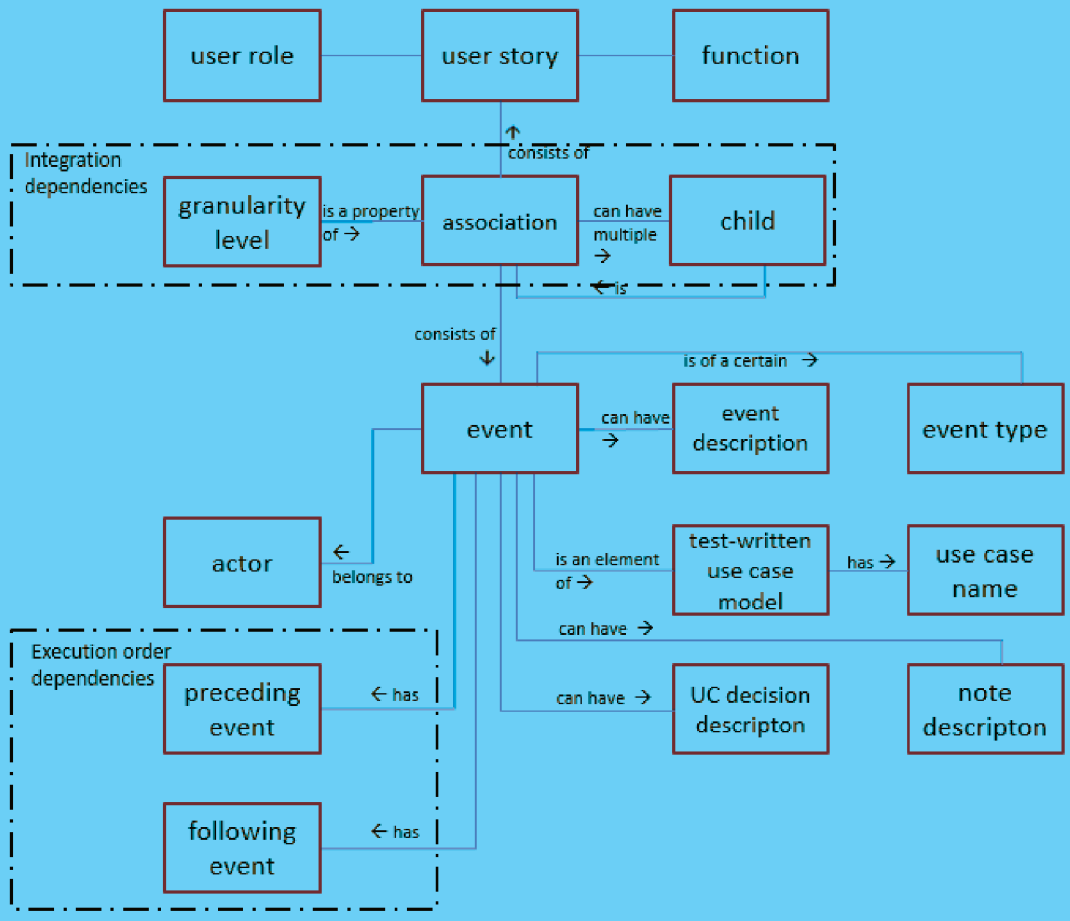


Figure 26: A model of integration of the user story and text-written use case

Constructs from level 1	Constructs from level 2	Constructs from level 3
thing	-	user role
	-	association
composite thing	who	actor
	-	event
	business process	text-written use case model
	-	user story
property	why	use case name
	-	granularity level
	what	event description
	-	event type
	-	UC decision description
	-	note description
	-	preceding event
	-	following event
	-	child
	-	function

Table 7: Derivation relationships for project-specific method focused on text-written use case model

3.7 BuPUS process

Figure 27 shows BuPUS process, which can be used in three situations:

1. for eliciting user stories from business process models, namely, BPMN model and text-written use case model,
2. for understanding the execution order and/or integration dependencies among user stories with either of the two models, and
3. for both, the elicitation and understanding.

The process starts with collecting available business process models and list of user stories. If the models are modeled in BPMN notation, the analyst needs to standardize the use of the BPMN elements by applying modeling guidelines suggested by Mendling et al. [139]. After all, not all process models in practice are of excellent quality [140]. Next, if the analyst observes, that the list of user stories is empty, a proposed technique for eliciting user stories from BPMN models is used. Its output consists of associated BPMN models (accompanied by a tree-hierarchy of user stories) and a list of user stories.

Next, if the analyst wishes to learn about execution order dependencies among user stories, he/she needs to apply our proposed technique for tracing execution order dependencies by using BPMN models. The input materials of the technique are associated BPMN models and a (not empty) list of user stories. Note that the models have not been associated yet if the list of user stories at the beginning of the BuPUS process was not empty. In this case, the analyst needs to create a tree-hierarchy of the given user stories and associate the BPMN activities with user story references. In BuPUS process we assume that an interest for understanding the dependencies rises from a certain problem-solving question which needs to be answered. Consequently, for understanding execution order dependencies of a user story we propose a problem-solving question generated with Template 1. Similarly, for understanding integration dependencies we propose a problem-solving question generated with Template 2. For more details about the templates see Sections 5.2.2 and 5.3.2. An output of the technique is the answer to that question. This makes the end of BuPUS process if the business process models we work with are BPMN models.

We could be also working with business process models modeled as text-written use case models. If the list of user stories is empty, a proposed technique for eliciting user stories from text-written use case models is used. Its output consists of associated text-written use case models (accompanied with a tree-hierarchy of user stories) and a list of user

stories. If the analysts can be interested in understanding execution order and integration dependencies he/she needs to apply our proposed technique for tracing execution order dependencies by using text-written use case models, and a problem-solving question generated with Template 1. Similarly, if he/she is interested in understanding integration dependencies then he/she needs to apply our proposed technique for tracing integration dependencies by using text-written use case models, and a problem-solving question generated with Template 2.

3.8 Summary of Chapter 3

We built BuPUS method by applying Process Configuration Approach proposed by Bajec et al. [44]. We have proposed the base method and two project-specific methods. The base method describes user story constructs and their relations. First project-specific method is focused on a BPMN model. It integrates the user story constructs and newly defined BPMN constructs. After that it proposes three techniques: a technique for eliciting user stories from a BPMN model, and techniques for tracing both execution order and integration dependencies by using BPMN models. The second project-specific method is focused on a text-written use case model. It integrates the user story constructs and newly defined text-written use case constructs. After that we propose three techniques: a technique for eliciting user stories from a text-written use case model, and techniques for tracing both execution order and integration dependencies by using text-written use case models.

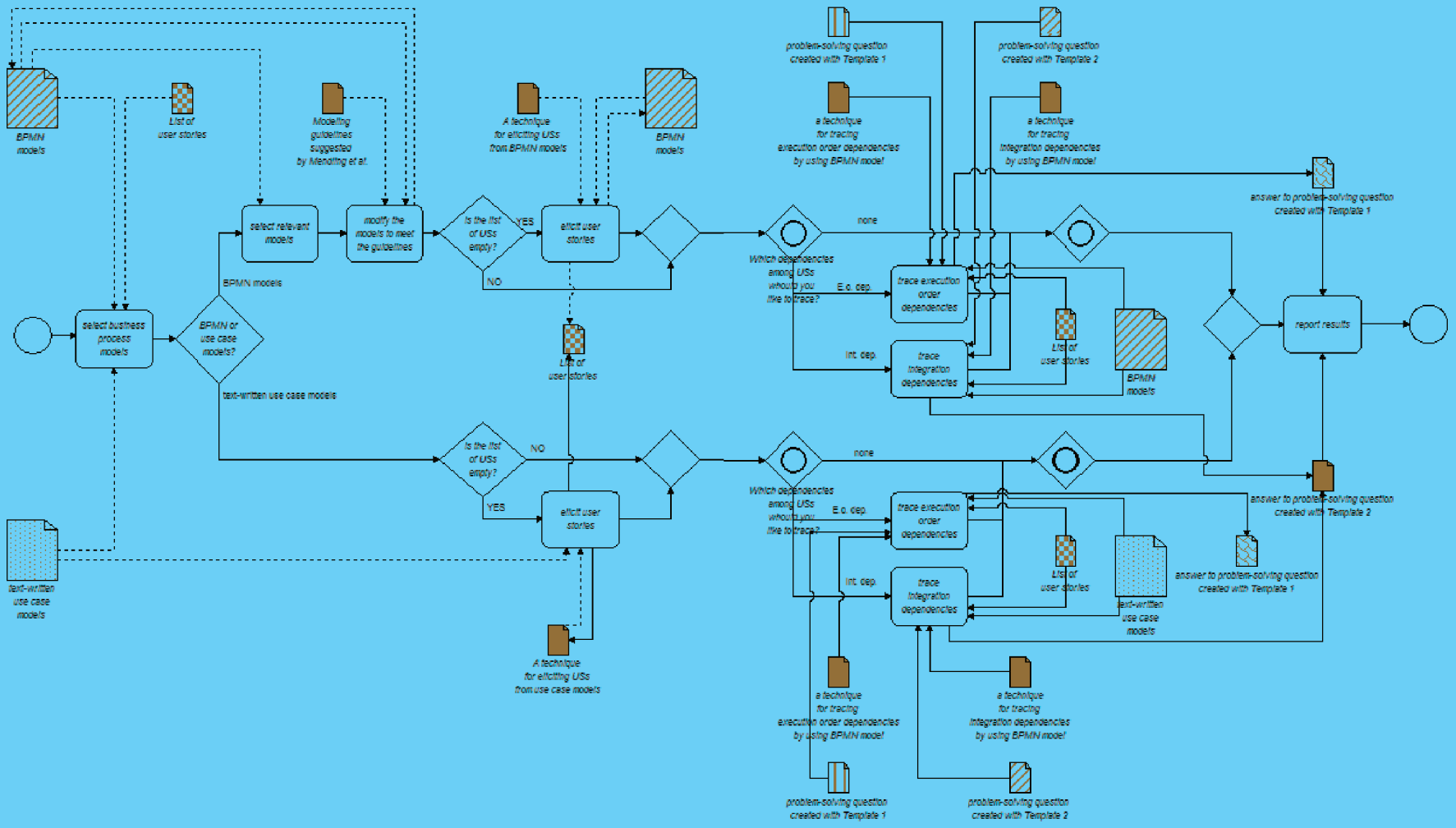


Figure 27: BuPUS process

4 Hypotheses

In this chapter, we present our hypotheses. In Section 4.1 we decompose two research questions into five prepositions. Next, we build three hypotheses for Proposition 1 and also three for Proposition 2 in Section 4.2. In Section 4.3 we build two hypothesis for Proposition 3 and two for Proposition 4. In Section 4.4 we build one hypothesis from Proposition 5. Finally, Section 4.5 summarizes the chapter.

4.1 Propositions

Our research question is: How can available business process models be exploited for agile software development with user stories? We argue that the method presented in Chapter 3 enables elicitation of user stories from business process models, and improves the understanding of the execution order and integration dependencies of a user story. We approached to our research gradually with two phases of evaluation, which complement each other. In first phase, we believe that associated business process models are a beneficial complementary model for understanding execution order and integration dependencies. We have chosen to work with BPMN models and evaluate whether they as business process models have a positive impact on the understanding. We formulated first two propositions, where we compared two configurations, namely, list of user stories alone and list of user stories with associated BPMN models (so-called BPMN material):

Proposition 1: Understanding of the execution order dependencies among user stories is greater when reading the BPMN material in comparison to reading the list of user stories.

Proposition 2: Understanding of the integration dependencies among user stories is greater when reading the BPMN material in comparison to reading the list of user stories.

After establishing that business process models have the positive impact we looked for, we argue that visual business process models are more beneficial for understanding the dependencies compared to textual business process models. For the visual business process model we continued to work with previously unexplored BPMN model, and for the textual business process model we have chosen to work with text-written use case model. The text-written use case model was previously discussed by Leffingwell [2] as a means with potential to support understanding of the dependencies. We decided to empirically evaluate his suggestion. Therefore, we compare two configurations, namely,

BPMN material and list of user stories with associated text-written use case models (so-called text-written use case material). We formulated next two propositions:

Proposition 3: Understanding of the integration dependencies of user stories is greater when reading the (visual) BPMN material in comparison to reading the text-written use case material.

Proposition 4: Understanding of the execution order dependencies of user stories is greater when reading the (visual) BPMN material in comparison to reading the text-written use case material.

The last proposition was formulated at the same time as Proposition 3 and 4 but focused on the elicitation. Similarly as before, we argue that for user story elicitation the visual business process models can elicit more of correct user stories than textual business process models. The text-written use case model was previously discussed by Ambler [7] as a means with potential to support elicitation of user stories. We decided to empirically evaluate his suggestion by comparing two elicitation techniques, namely, a technique for eliciting user stories from a BPMN model, and technique for eliciting user stories form text-written use case models. Thus, the fifth proposition is:

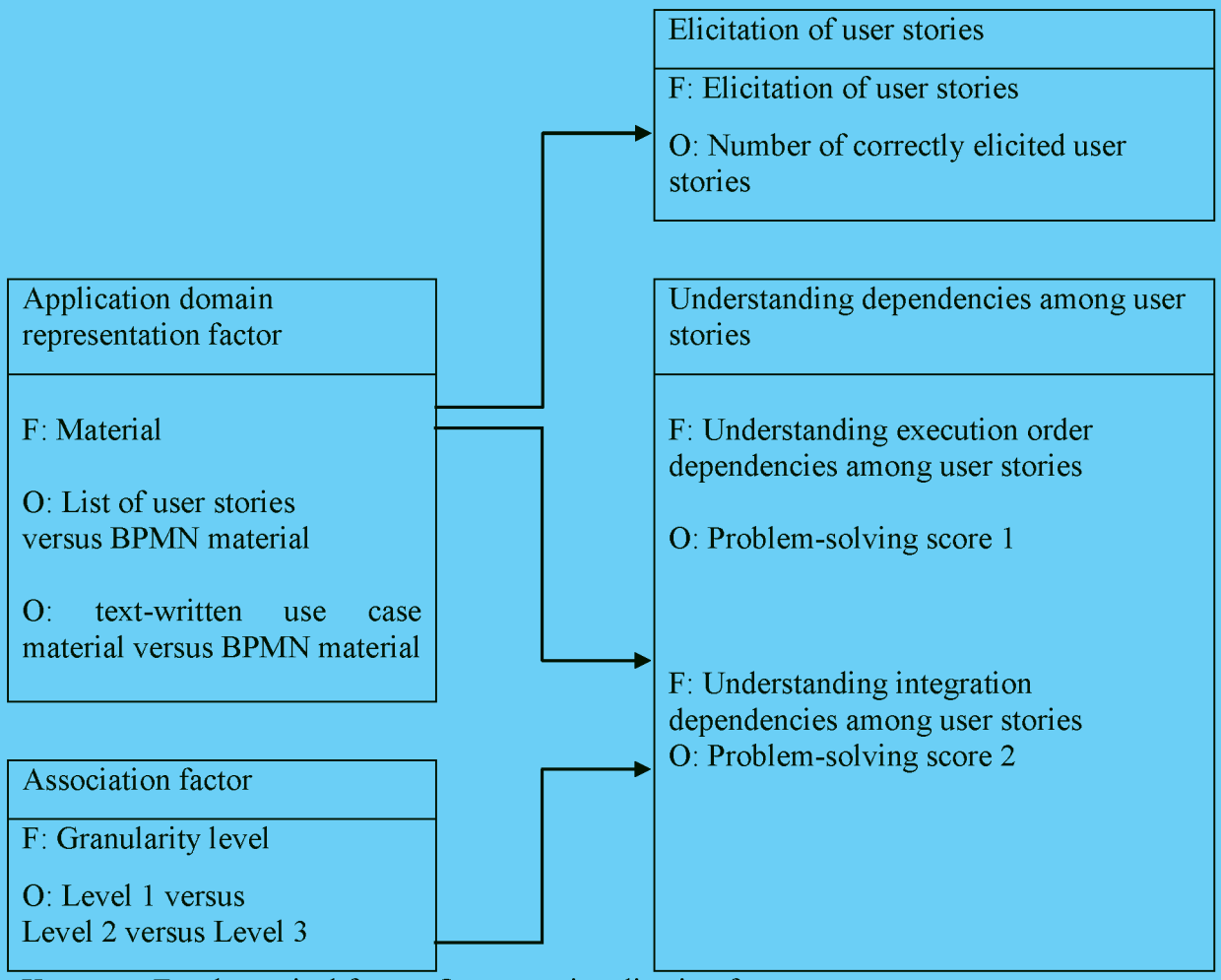
Proposition 5: Elicitation of user stories is more effective when using the (visual) BPMN material in comparison to the text-written use case material.

In order to decompose Proposition 1, 2, 3 and 4 into hypotheses, we discuss cognitive theory. The focus on cognitive theory is natural when considering conceptual modeling techniques [38]. We follow recent work [28, 31, 35] when discussing Mayer's cognitive theory of multi-media learning [39]. The theory has frequently been used to evaluate conceptual modeling grammars [28]. It is a theory of how people learn from words and pictures. Mayer's research relies on the experimental comparison in which an experimental group of learners receives a lesson that contains the to-be-tested feature while a control group of learners receives an otherwise identical lesson that lacks the to-be-tested feature. Subsequently, both groups take a problem-solving test.

4.2 Building hypotheses for Proposition 1 and 2

In further sub-sections we discuss: 1) Mayer's spatial contiguity principle applied to two techniques of project-specific method focused on BPMN model, namely, a technique for tracing execution order dependencies from a BPMN model, and a technique for tracing integration dependencies from a BPMN model 2) Mayer's extraneous processing

principle applied to two configurations, namely, a list of user stories, and BPMN material, 3) the expressiveness of two different grammars which support the two configurations, 4) information equivalence of the two configurations, and 5) hypothesis H1-H6 as part of our research model in Figure 28.



Key: F = theoretical factor, O = operationalization factor

Figure 28: Research model

4.2.1 Spatial contiguity principle

The technique for tracing execution order dependencies by using BPMN models (described in Section 3.5.3) and technique for tracing integration dependencies by using BPMN models (described in Section 3.5.4) adhere to the spatial contiguity principle of Mayer's theory which suggests placing essential words next to the corresponding graphical element. Also a recent study by Ottensooser et al. [31] demonstrate that when using textual and graphical notation at the same time the performance of domain understanding is the highest.

The first technique, technique for tracing execution order dependencies by using BPMN models, uses the user story code labels as user story references. The references are placed in a text box next to the corresponding BPMN activity element. By associating user stories to activity elements, a clear representation of the execution order among user stories is gained by following BPMN flow arrows. Since the activity elements follow a temporal sequence, the user stories associated to those activities also need to follow that sequence. The technique's output consists of words and pictures, while the list of user stories only has words. We believe this is one of the reasons that the technique's material should provide better support in understanding the execution order dependencies among user stories compared to having only the list of user stories.

The second technique, technique for tracing integration dependencies by using BPMN models, uses tree-hierarchy of user stories (see e.g. Figure 18) and/or a technique for nesting of association objects (see e.g. Figure 11) to enable clear representation of integration dependencies among user stories. So, one way to trace integration dependencies is by following the tree-hierarchy's branches. At one end, there is an epic user story. When moving to the other end, the user stories get more and more detailed. At the far end of the tree-hierarchy's branches, there are user stories which are of acceptable size to enter development iteration. And the other way to trace integration dependencies is by using a technique for nesting of association objects. Its text boxes with user story reference codes are placed directly into the BPMN model next to the corresponding activity element. The technique's output consists of words and pictures, while the list of user stories only has words. We believe this is one of the reasons that the technique's material should provide better support in understanding the integration dependencies among user stories compared to having only the list of user stories.

4.2.2 Extraneous processing

Extraneous processing is cognitive processing during learning that does not serve the learning goal and causes a cognitive overload [39]. The list of user stories has three characteristics which cause extraneous processing. First, it contains user story clauses which are from different business process models. Learning the execution order and integration dependencies from such a list causes extraneous processing. Second, the user story clauses use synonyms for the same thing. An assumption of our study is that no glossary is available. Third, the user stories are from different levels of abstraction. These three characteristics, which decrease the understanding of the dependencies among user stories, can be present in real-life projects. We took these characteristics into account

when preparing the list of user stories for our empirical evaluation. The list was available also with BPMN material.

4.2.3 Expressiveness of a grammar

The expressiveness of a grammar is its ability to generate scripts that capture information about a modeled domain [38]. Wand and Weber [141] propose evaluating modeling methods by analyzing their constructs. In Chapter 3, we grounded the expressiveness of two grammars, which we promoted in two models, namely, a base method data structure model (Figure 7), and a model of integration of the user story and BPMN model (Figure 19). Our goal in this section is to examine their ability to convey information about the execution order and integration dependencies. In our research, the grammar used by base method data structure model is not different from the grammar that is used for generating one single user story clause. This kind of user story clauses are commonly found in a phase of project preparation which occurs in initial requirements envisioning days. The user story template is simple since the customer side of the development project is responsible for delivering its list of user stories [3]. Consequently, the user story clauses are too coarse to promote understanding of the dependencies [25]. Later, in development iterations the knowledge about specific user stories grows with the communication between the development team and the customer representative [3]. Figure 7 shows the state of user stories in the project phase of preparation where there are no constructs holding information about the execution order or integration dependencies among user stories. In Figure 19 we obtain information about the execution order dependencies among user stories in an explicit way. The information is stored in properties of constructs *gateway* and *flow*.

We also focused on the integration dependencies among user stories. Similar to Patton [27], Lin et al. [16], Clarke and Kautz [24] and Leffingwell [2], we group user stories of different abstraction levels together and depict them in a specific structure. Figure 19 depicts the constructs that are used to hold information about the integration dependencies among user stories which are *association*, *granularity level* and *child*. We believe that the BPMN constructs are sufficiently grammatically expressive to share explicit knowledge about both the execution order and integration dependencies among user stories with the project's stakeholders since there are constructs which capture that information.

4.2.4 Informational equivalence

When comparing material for two experimental treatments while using two different grammars, the results might be confounded because one model might contain more information than the other or one may be easier to understand than the other [38]. As suggested by Gemino and Wand [38], we discuss this issue by analyzing informational equivalence. The term informational equivalent means that all information in one material is inferable from the other and vice versa [38, 142]. In our first experiment, the control group was given the list of user story clauses (see the example in Table 2), while the experimental group was given the list of user story clauses as well as the associated business process model (see the example of the model in Figure 11). When preparing the material, we strove to move close to informational equivalence. First, the set of user roles discussed on the list is the same as the set of BPMN swimlanes (e.g. user role "clerk in the front office" and pool "front office" with the lane "clerk"). Second, the functions of user stories are associated with activity elements where no activity element is left without a user story association. The difference in materials is in the additional elements of the associated business process model, namely, the decision and merge XOR gateways, and flow arrows. We use flow arrows to show the sequence of activity elements and consequently the execution order dependencies among user stories. We use XOR gateways to demonstrate how the integration dependencies among user stories can be shown in an associated business process model.

Next, when associating user stories with activity elements we would move closer to information equivalence if each association object was linked to exactly one user story. But since we wanted to investigate three different types of association granularity (namely, a user story is: a) approximately equally big as; b) more abstract than; or c) more detailed than its corresponding activity element), the association labels on the business process models were not equally distributed to the activity elements. As a result, some activities had more than one association and some user stories were associated to multiple activity elements.

4.2.5 Hypotheses H1-H6

Figure 28 shows our research model. Two factors have an influence on an understanding of the dependencies: an application domain representation factor and an association factor. First, the application domain is represented by one of the two materials: the list of user stories or the BPMN material. Second, each user story association is characterized by one of the three granularity levels: level 1, level 2, or level 3. If they were modeled in

an abstract manner compared to the associated activity description, we would obtain level 1 user story associations. If they were modeled at a similar abstraction level as the user stories, we would obtain a level 2 user story association. And if the given business process models activity elements were modeled in great detail compared to the associated activity description, we would obtain a level 3 user story association. With this setting, we can compare how each of the two application domain representations of the three granularity levels influences the understanding of the execution order and integration dependencies among user stories. From Proposition 1 which focuses on the understanding of the execution order dependencies, we created three hypotheses for each user story granularity level: H1, H2, and H3. Similarly, from our Proposition 2 which focuses on the integration dependencies we created hypotheses H4, H5, and H6.

H1: When a user story is more abstract than the corresponding activity in a business process model, the understanding of the *execution order dependencies* among user stories is greater reading the BPMN material than by reading the list of user stories.

H2: When a user story is approximately equally big as the corresponding activity in a business process model, the understanding of the *execution order dependencies* among user stories is greater reading the BPMN material than by reading the list of user stories.

H3: When a user story is more detailed than the corresponding activity in a business process model, the understanding of the *execution order dependencies* among user stories is greater reading the BPMN material than by reading the list of user stories.

H4: When a user story is more abstract than the corresponding activity in a business process model, the understanding of the *integration dependencies* among user stories is greater reading the BPMN material than by reading the list of user stories.

H5: When a user story is approximately equally big as the corresponding activity in a business process model, the understanding of the *integration dependencies* among user stories is greater reading the BPMN material than by reading the list of user stories.

H6: When a user story is more detailed than the corresponding activity in a business process model, the understanding of the *integration dependencies* among user stories is greater reading the BPMN material in comparison to reading the list of user stories.

4.3 Building hypotheses for Proposition 3 and 4

In further sub-sections we discuss: 1) Mayer's spatial contiguity principle applied to two techniques, namely, a technique for tracing execution order dependencies from a BPMN model, and a technique for tracing execution order dependencies from a text-written use case model 2) Mayer's spatial contiguity principle applied to two techniques, namely, a technique for tracing integration dependencies from a BPMN model, and a technique for tracing integration dependencies from a text-written use case model, 3) Mayer's extraneous processing principle which we apply to two configurations, namely, a text-written use case material, and BPMN material, 3) the expressiveness of two different grammars which supports the two configurations, 4) information equivalence of the two configurations, and 5) hypothesis H7-H10 as part of our research model.

4.3.1 Spatial contiguity principle

We compare two techniques in hypothesis H7: the technique for tracing execution order dependencies by using BPMN models (described in Section 3.5.3) and technique for tracing execution order dependencies by using text-written use case models (described in Section 3.6.3). We have already discussed how the product of the first technique, technique for tracing execution order dependencies by using BPMN models, adheres to the special contiguity principle in Section 4.2.1. The second technique, technique for tracing execution order dependencies by using text-written use case models produces textual material. The information is grouped in sections and also in tables of those sections. The user story references are placed in one or more columns of the tables so each event description (a row) is associated to multiple user stories but on different abstraction levels. By associating user stories to event descriptions, a clear representation of the execution order among user stories is gained by following the sequence of rows. Since the rows follow a temporal sequence, the user stories associated to those rows also need to follow that sequence. However, the technique's output consists of text only (no pictures), so we believe this is one of the reasons that the other technique which produces BPMN material should provide better support in understanding the execution order dependencies among user stories.

We compare another two techniques in hypothesis H8: the technique for tracing integration dependencies by using BPMN models (described in Section 3.5.4) and technique for tracing integration dependencies by using text-written use case models (described in Section 3.6.4). We have already discussed how the product of the first technique, technique for tracing execution order dependencies by using BPMN models,

adheres to the special contiguity principle in Section 4.2.1. The second technique, technique for tracing integration dependencies by using text-written use case models produces textual material. As mentioned, the user story references are placed in one or more columns of the tables so each event description (a row) is associated to multiple user stories but on different abstraction levels. In this technique we follow the rule: the column with user story references which is more on the right hand size is more abstract than the one on the left. By associating user stories to event descriptions, a representation of the integration dependencies among user stories is gained by following the columns with user story references from the left to the right. The last column of the far right present an epic user story. This representation of the integration dependencies among user stories can become overwhelming. For more complex integration dependencies we propose to complement the representation on the text-written use case model with a tree-hierarchy of user stories (see e.g. Figure 18). Nevertheless, in our experiment we choose to work with the visual representation of integration dependencies, which are added directly to the text-written use case model. Since this representation consists of text only (no pictures), we believe this is one of the reasons that the other technique, which produces BPMN material, should provide better support in understanding the integration dependencies among user stories.

4.3.2 Extraneous processing

The list of user stories which complements both techniques' products. It has three characteristics which cause extraneous processing. First, it contains user story clauses which are from different business process models. Second, the user story clauses use synonyms for the same thing. Third, the user stories are from different levels of abstraction. These three characteristics can be present in real-life projects. We took these characteristics into account when preparing both BPMN and text-written use case material for measuring understanding of execution order and integration dependencies among user stories. Therefore, the extraneous processing was the same either with BPMN or text-written use case material.

4.3.3 Expressiveness of a grammar

In Chapter 3, we grounded the expressiveness of two grammars presented in two models, namely, the model of integration of the user story and BPMN (Figure 19), and the model of integration of the user story and text-written use case constructs (Figure 26). Our goal in this section is to examine their ability to convey information about the execution order and integration dependencies. We have already discussed expressiveness of the BPMN

constructs for supporting explicit knowledge about both execution order and integration dependencies in Section 4.2.3. Nevertheless, we have not yet discussed expressiveness of the text-written use case constructs. Figure 26 depicts constructs that are used to hold information about the integration dependencies among user stories which are *association*, *granularity level* and *child*. This constructs are the same as for BPMN material. On the other hand, text-written use case constructs which supporting execution order dependencies are different from those used with BPMN models. For tracing execution order dependencies the knowledge about temporal sequence is crucial. In text-written use case models *preceding* and *following event* hold that information. We conclude that both grammars, namely, the grammar used in model of integration of the user story and BPMN model, and grammar used in the model of integration of the user story and text-written use case model, can hold explicit knowledge about both the execution order and integration dependencies among user stories.

4.3.4 Informational equivalence

In our second experiment, one group was given the BPMN material, while another group was given text-written use case material. We were interested in which material supports the understanding of execution order and integration dependencies among user stories better. For this reason we aimed for informational equivalence of the two materials as presented in Table 8.

4.3.5 Hypotheses H7-H10

As mentioned, in Figure 28 two factors have an influence on an understanding of the dependencies: an application domain representation factor and an association factor. First, the application domain is represented by one of the two materials: the text-written use case material or the BPMN material. Second, each user story association is characterized by one of the two granularity levels: level 1, or level 2. If they were modeled in an abstract manner compared to the associated activity description, we would obtain level 1 user story associations. And if they were modeled at a similar abstraction level as the user stories, we would obtain a level 2 user story association. With this setting, we compare how each of the two application domain representations influences the understanding of the execution order and integration dependencies among user stories. From Proposition 3 which focuses on the understanding of the execution order dependencies, we created two hypotheses for each user story granularity level: H7, and H8. Similarly, from our Proposition 4 which focuses on the integration dependencies we created hypotheses H9, and H10.

H7: When a user story is approximately equally big as the corresponding activity in a business process model, the understanding of the *execution order dependencies* among user stories is greater reading the BPMN material than by reading the text-written use case material.

H8: When a user story is more abstract than the corresponding activity in a business process model, the understanding of the *execution order dependencies* among user stories is greater reading the BPMN material than by reading the text-written use case material.

H9: When a user story is approximately equally big as the corresponding activity in a business process model, the understanding of the *integration dependencies* among user stories is greater reading the BPMN material than by reading the text-written use case material.

H10: When a user story is more abstract than the corresponding activity in a business process model, the understanding of the *integration dependencies* among user stories is greater reading the BPMN material than by reading the text-written use case material.

	BPMN material	Text-written use case material
User story's function	Activity description	Event description
User story's user role	Swimlane	Actor
What is the model about?	Business process name	Use case name
Associating user stories	Placing user story references next to the activity descriptions	Placing user story references in same row as event descriptions
Conditions in which a certain set of alternative flows can be started	OR split gateway element	Note in section Additional information
Conditions in which the event can be started	OR merge gateway element	Note in section Additional information
	Manual activity element	Note in section Additional information
	Activity elements supported by other application	Note in section Additional information
Level 2 association	Yellow text box with a user story reference in it	Yellow text coloring of a column in tables
Level 1 association	Blue text box with a user story reference in it	Blue text coloring of a column in tables

Table 8: Informational equivalence of BPMN and text-written use case material

4.4 Building hypothesis for Proposition 5

We compare the following two techniques: the technique for eliciting user stories from BPMN models (described in Section 3.5.2), and technique for eliciting user stories from text-written use case models (described in Section 3.6.2). The first technique adheres to the spatial contiguity principle of Mayer's theory which suggests placing user story references next to the corresponding activity element, and the second one does not since it does not operate with visual elements. In both treatments the subjects needed to perform basic sub-elicitation, and first two steps of advanced hither-level sub-elicitation. We believe visual BPMN elements are the reason why should the first technique perform better in elicitation of correct user stories. The grammars which support the two techniques are both equally expressive. They both support the creation of same user story clauses. The first uses *swimlane* and *activity description*, and the second uses *actor* and *event description*. The advanced hither-level sub-elicitation is not influenced by the given material/treatment. The materials for both treatments are informationally equivalent as argued in Table 6 and 8.

In our research model (Figure 28) there is one factor that has an influence on user story elicitation performance: an application domain representation factor. This factor is represented by one of the two materials: the text-written use case material or the BPMN material. We compare how each of the two application domain representations influences the elicitation of correct user stories. From Proposition 5 we created one hypothesis:

H11: Elicitation of user stories results in more of correct user stories when using the (visual) BPMN model in comparison to the text-written use case model.

4.5 Summary of Chapter 4

Our main research question was broken down to five propositions, where two of them were focused on the understanding of execution order dependencies, another two for the understanding of integration dependencies, and one on the elicitation of user stories. Each of the propositions was decomposed to one or more hypothesis. In total, we proposed eleven hypotheses.

5 Research methodology

In this chapter we describe how we planned, operationalized and analyzed two experiments. Section 5.1 describes experimental design for both of them. Afterwards, Section 5.2 focuses on the first experiment, and Section 5.3 focuses on the second experiment. Finally, Section 5.4 summarizes Chapter 5.

5.1 Experimental design

Experiment is a research method promoted in the design science research methodology [41]. We conducted two experiments and used a one-factor experiment with two treatments with a completely randomized design for both of them.

For the *first experiment* we conducted approximately 10 pilot experiments between January and March 2015 in order to test subject's understanding of our instructions and prepared material. During the pilot phase, we made sure that the questionnaire was comprehensible and possible to be completed by majority in 50 minutes. The feedback gained from the pilot experiments also influenced the BuPUS development (e.g. the set of constructs that we presented as relevant). For the experiment we created two workbooks, namely workbook 1 and workbook 2 as depicted in Table 9. In both workbooks, the first treatment was *guided* which means that an additional corresponding business process model associated with the user stories was available, and that we gave instructions about its use. The second treatment was *unguided* which means that solely the list of user stories was available. We gave no instructions. Each participant was exposed to both treatments from two cases, namely, Case 1 and Case 2. Both cases were from the banking sector. Data collected from the guided and the unguided treatment of the same subject were treated independently. Therefore, the method of data collection used was a between-subject design. The experiment was conducted in April 2015. Workbook 1 is shown in Appendix A.

Workbook	**Treatment 1: guided (BPMN material)**	**Treatment 2: unguided (list of user stories)**
Workbook 1	Case 1	Case 2
Workbook 2	Case 2	Case 1

Table 9: A composition of workbooks for experiment 1

For the *second experiment* we conducted 3 pilot experiments between October and November 2015 in order to test subject's understanding of our instructions and prepared BPMN and text-written use case material. During the pilot phase, we made sure that the questionnaire was comprehensible and possible to be completed by majority in 65

minutes. The feedback gained from the pilot experiments also influenced the BuPUS development (e.g. the set of constructs that we presented as relevant). For the experiment we created four workbooks, namely workbook A, B, C and D as depicted in Table 10. In workbooks A and B, the subjects firstly work with BPMN material and secondly with UC material (shorten for text-written use case material). BPMN material is consisted of a list of user stories and an associated BPMN model. The UC material is consisted of a list of user stories and associated text-written use case model. In workbooks C and D, the subjects firstly work with UC material and secondly with BPMN material. Both BPMN and UC material are informational equivalent as discussed in Chapter 4. Each participant was exposed to both treatments from two cases. Case "OPEN" and Case "CLOSE" are both from the banking sector. Data collected from the guided and the unguided treatment of the same subject were treated independently. Therefore, the method of data collection used was a between-subject design. The experiment was conducted in November 2015. Workbook A is shown in Appendix B.

Workbook	**Treatment: BPMN material**	**Treatment: list of user stories**	**The first treatment in the workbook**
Workbook A	Case "OPEN"	Case "CLOSE"	BPMN material
Workbook B	Case "CLOSE"	Case "OPEN"	BPMN material
Workbook C	Case "CLOSE"	Case "OPEN"	UC material
Workbook D	Case "OPEN"	Case "CLOSE"	UC material

Table 10: A composition of workbooks for experiment 2

5.2 Experiment 1: list of user stories versus BPMN material

5.2.1 Task

The task for our subjects was to interpret the given material. The subjects read the material provided and gave answers to the questions about user stories. The questions were focused on measuring the understanding of the user stories. Mayer [39] promotes problem-solving tests to evaluate how well the learner understands the material. A measurement instrument that Mayer uses in his experiments is a questionnaire and the measures are its scores. As suggested by Gemino and Wand [38], we measured the subjects' performance in comprehension, problem-solving and recall tests.

5.2.2 Questionnaire

Workbooks 1 and 2 are composed of background questions and two treatments. The first treatment is the one with BPMN material. Each treatment is composed of four tests: comprehension, problem-solving, recall test, and perceived ease of use and usefulness test. The first test for each treatment was a comprehension test on the given material. The

test was used for control purposes since we wanted to ensure that the subjects made an effort and developed their cognitive model. Afterwards, the subjects took the problem-solving test. The problem-solving questions were used to support the hypotheses. Next, a second comprehension test was conducted, but this time with the material removed (called a recall test in the continuation). The recall test was also used for control purposes only. The last set of questions for each treatment included self-assessment items. They were about the (perceived) ease of use and usefulness of the given material.

By scoring a *comprehension test* we measure how well a subject can understand the given material. The test questions can be answered by either looking at the list of user stories or by looking at the BPMN material (the product of our BuPUS). BPMN material shows the user story dependencies visually while in the list of user stories shows them textually. We used the following comprehension questions:

- Which user role executes the user story US_*? (radio buttons)
- What are the two "card limit types" that the customer can ask for? (radio buttons)
- What are the two "customer types"? (open question)
- Is the execution of US_* optional? (Yes / No / Do not know)
- Which user stories have the potential to be executed when the bank's customer wants to get a new MasterCard? (check boxes)
- Which user roles need to collaborate when the bank's customer wants to get a new MasterCard? (check boxes)

The *problem-solving test* consists of two types of questions where one type evaluates the understanding of execution order dependencies and the other of integration dependencies. Each type has its own template. The first problem-solving template question focuses on the understanding of the execution order of a specific set of user stories.

Template 1: What was previously done wrong (or not done at all) so the activity <specific user story function> cannot get started?

We narrowed down the possible answers by providing additional guidance. We broke question Template 1 down into three specific sub-questions. As shown in Figure 29, under sub-question: a) the subject needs to fill in the user story codes about the execution order (with the subjects we used a word *sequence*) of user stories which can lead to the execution of user story US_299. The subjects can recognize that there are four different execution orders (also referred to as sequences) from the question itself. Subjects with guided material are able to see the execution orders by following the flow arrows between

the associated business process activities. On the other hand, subjects with unguided material need to read the user story clauses and understand the execution orders from that. Sub-question b) refers to the execution orders of a). In b) the subjects need to find out where in the execution order the work is passed to another user role. The answer requires checking if the first and the second user story in the execution order have different user roles, then the second and the third, then the third and the fourth and so forth. Subjects with guided material could see the shift by observing flow arrows which cross to another swimlane, while subjects with unguided material needed to read the user story clauses and understand the information from that. Lastly, sub-question c) also refers to the execution orders of a). The subjects need to understand if there are any business decisions taken which influenced the further execution order. Subjects with guided material could recognize the decision points by understanding the XOR gateway elements, while subjects with unguided material need to understand these by reading the user story clauses. Points are given for each correctly filled gap, for choosing the right radio button and for ticking the correct check boxes (see Figure 29).

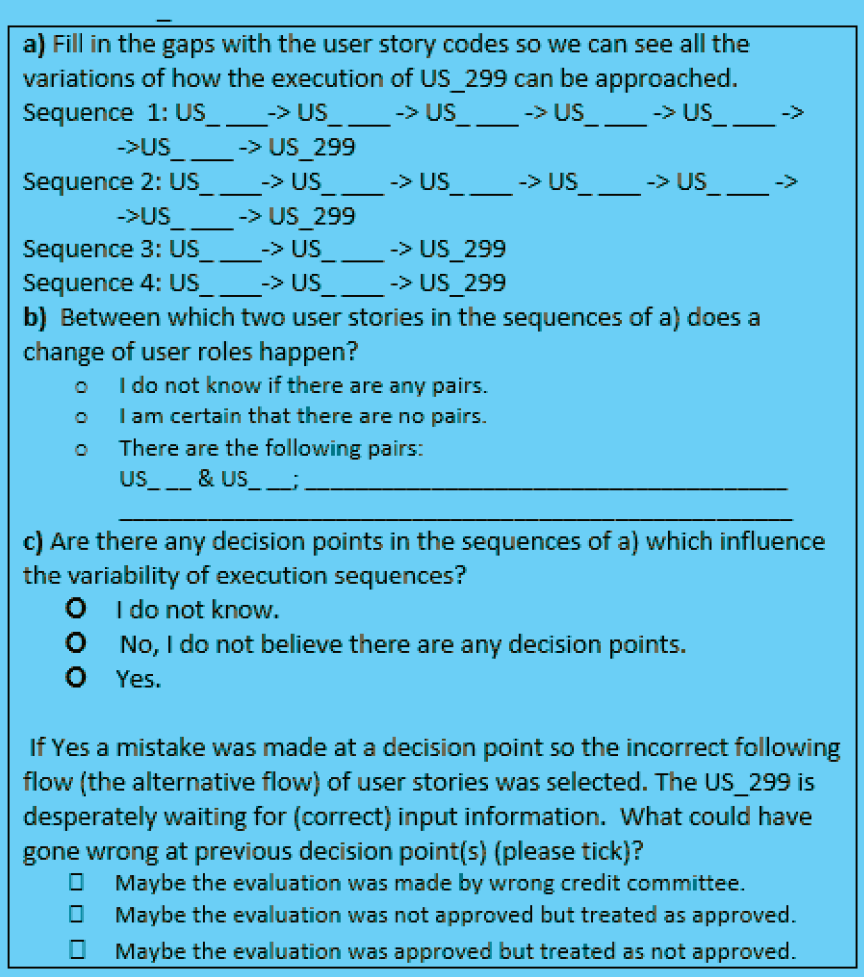

a) Fill in the gaps with the user story codes so we can see all the variations of how the execution of US_299 can be approached.
Sequence 1: US_ ___-> US_ ___ -> US_ ___ -> US_ ___ -> US_ ___ ->
->US_ ___ -> US_299
Sequence 2: US_ ___-> US_ ___ -> US_ ___ -> US_ ___ -> US_ ___ ->
->US_ ___ -> US_299
Sequence 3: US_ ___-> US_ ___ -> US_299
Sequence 4: US_ ___-> US_ ___ -> US_299
b) Between which two user stories in the sequences of a) does a change of user roles happen?
o I do not know if there are any pairs.
o I am certain that there are no pairs.
o There are the following pairs:
US_ __ & US_ __; _____
_____
c) Are there any decision points in the sequences of a) which influence the variability of execution sequences?
O I do not know.
O No, I do not believe there are any decision points.
O Yes.
If Yes a mistake was made at a decision point so the incorrect following flow (the alternative flow) of user stories was selected. The US_299 is desperately waiting for (correct) input information. What could have gone wrong at previous decision point(s) (please tick)?
☐ Maybe the evaluation was made by wrong credit committee.
☐ Maybe the evaluation was not approved but treated as approved.
☐ Maybe the evaluation was approved but treated as not approved.

Figure 29: Example of Case 1 sub-questions used for answering a Template 1 question (caption from Workbook 1)

For example, in Case 1 there is a problem-solving question: *What was previously done wrong (or not done at all) so the activity "send a notification about a change in the account's attribute (US_244)" cannot get started?* The answer to a Template 1 question is a combination of answers to sub-questions a), b), and c). Therefore, the answer to the question refers to Figure 24 and is composed like this:

- Maybe one of the preceding user stories was not executed correctly – see the user stories listed under a).
- Maybe there was a communication failure between two user stories when passing information from one user role to another – see the pairs of user stories listed under b).
- Maybe a wrong decision about a sub-flow was made – see the boxes under c) which are ticked.

The second problem-solving template question requires understanding of the integration dependencies of a specific user story.

Template 2: Sometimes the activity <specific user story function> has to be executed in a different way. When does that happen?

To answer the Template 2 question, less abstract version(s) of a particular user story need to be available on the list of user stories. An example question is: *Sometimes the activity "notify the customer about the credit committee's decision (activity from US_299)" can be executed in different variations. Which user stories (codes) present those variations?* Figure 11 shows the nesting of association objects: an association object with label US_299 (I as a clerk in the front office can notify the customer about the credit committee's decision) has two specializations: US_43 (I as a clerk in the front office can give information about the rejection) and US_999 (I as a clerk in the front office can send a letter about the approval). The answer to the question in Figure 30 consists of choosing the second radio button and filling in the keywords on the line: rejection (or US_43) and approval (or US_999).

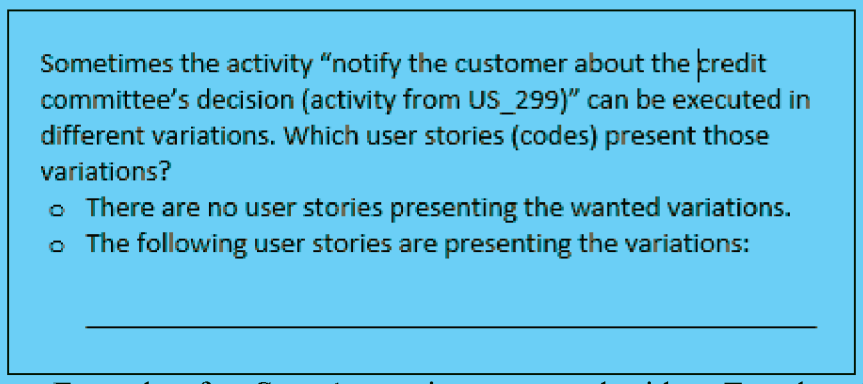


Figure 30: Example of a Case 1 question generated with a Template 2 question (caption from Workbook 1)

Each problem-solving template question is used three times per case in three situations when a user story is: 1) more abstract than; 2) approximately equally big as; and 3) more detailed than the corresponding activity in a business process model. Table 11 presents an overview of the questions for Case 1 and their relationship to the hypotheses.

Question	**Hypothesis**
Sub-questions of Template 1 addressed to US_299 (see Figure 29)	H1
Sub-questions of Template 1 addressed to US_11	H2
Sub-questions of Template 1 addressed to US_244	H3
Template 2 addressed to US_299 (see Figure 30)	H4
Template 2 addressed to US_11	H5
Template 2 addressed to US_244	H6

Table 11: Problem-solving questions of Case 1 and their relationship to the hypotheses

The *recall test* measures how well the subjects remembered the information about the domain (see e.g. in Figure 31).

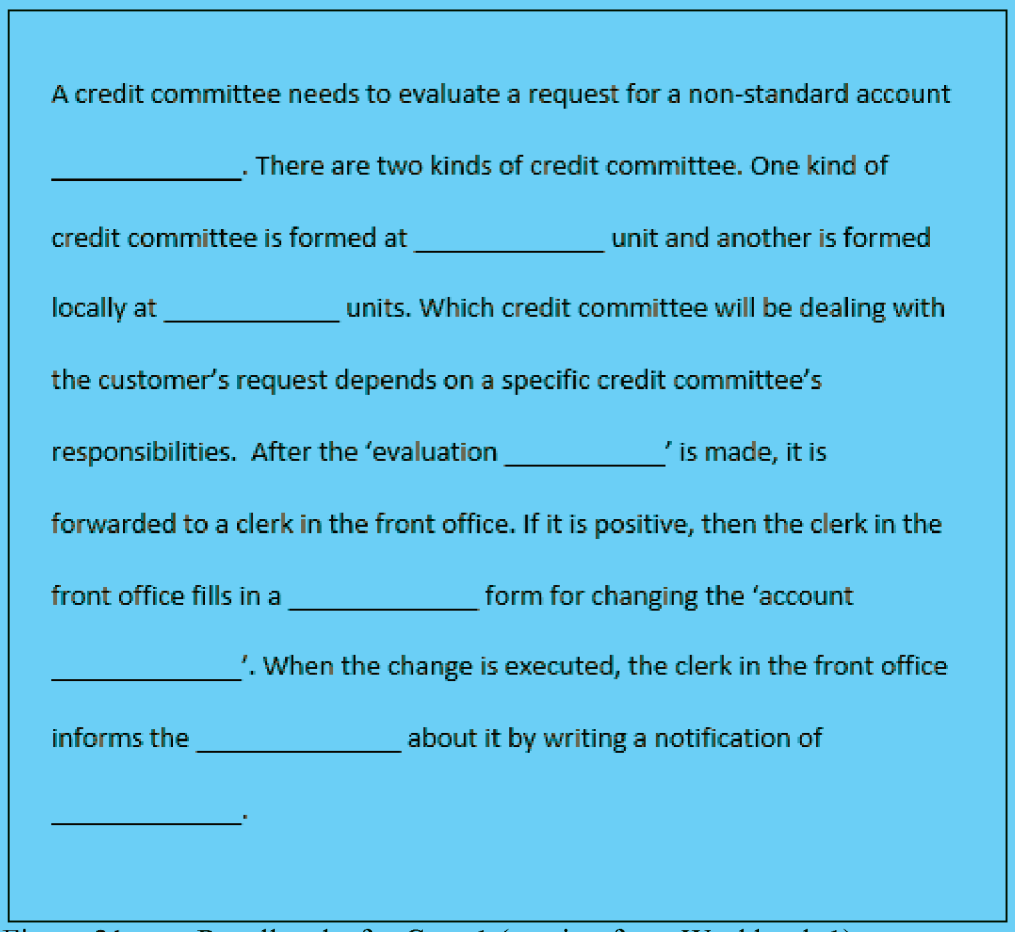


Figure 31: Recall tasks for Case 1 (caption from Workbook 1)

Questions from Maes and Poels [143] are used to measure perceived *ease of use and usefulness* on a 7-point Likert scale with response options ranging from "strongly disagree" to "strongly agree". The questions are asked separately for the guided treatment and the unguided treatment. The statements were:

- It was easy for me to understand what the material was trying to show.
- Using the material was often frustrating.
- Overall, the material was easy to use.
- Learning how to read the material was easy.

5.2.3 Materials

We used business process models which were developed for a European bank for analyzing and optimizing the business processes and not for IT development. We chose an 'account management' topic, which had 24 business process models available. Out of those 24 models, we chose two of them for our experiment: "getting a new MasterCard", and "getting a non-standard account limit". We made modifications to the two models by following the seven process modeling guidelines [139]: we consistently used message flow arrows when the following activity was in another pool, we added merge gateways

to every XOR decision gateway, we translated all labels to English and used verb-object activity labels. We have also hidden some business process parts to protect business secrecy (e.g. we used invented names for departments). When making these modifications, we were careful not to change the information of the business process models but only its visual presentation.

The BPMN models were created in iGrafx. For our research, the activity element is the most important element since it is associated with a user story. Awareness of its surrounding elements such as swimlanes, XOR gateways, and flow arrows is also important for understanding the dependencies. For each case, we provided a business process model with these elements. An example of a business process model for Case 1 is shown in Figure 11.

After the business process models were prepared, we created two lists of user stories. Our assumption is that all user stories on the list meet the INVEST quality aspects and that they are selected to enter in a common development iteration. An example of the list of user stories for Case 1 is shown in Table 2. In our material we incorporated the three characteristics of the list of user stories discussed in Section 2.2.3. The first characteristic is "the naming of the user story functions can refer to different levels of abstraction". We associated the function to a business process activity element with one of the three levels of the association granularity in the following way:

- Level 1: A situation when a user story is more abstract than the corresponding activity in a business process model. This was done by describing the user story function on a higher level of abstraction than the associated activity element description. An example of a level 1 association is shown in Figure 11 where US_299 is associated to multiple activity elements.
- Level 2: A situation where a user story is approximately of equal abstraction as the corresponding activity in a business process model. This was done by describing the user story function and activity element description in a very similar way or even exactly in the same way. An example of a level 2 association is shown in Figure 11 where US_11 is included.
- Level 3: A situation where a user story is more detailed than the corresponding activity in a business process model. This was done by presenting a user story function as a very detailed task of the associated activity element. An example of a level 3 association is shown in Figure 11 where multiple user stories (US_189 and US_244) are associated to one activity element.

The second characteristic "different synonyms can refer to the same artifact" is integrated into our material as described in Section 2.2. And the third characteristic "User stories can take part in different execution orders" means that not all user stories from the list of user stories in Table 2 (see US_300 and US_100) are associated to the activities of business process model presented in Figure 11.

For preparing an associated business process model for Case 2, we followed the specific set of rules described in Table 12.

Association level	Association level's characteristic
Level 1	The user story is associated to multiple activity elements. AND The associated activity elements are between the same XOR decision and XOR merge gateway. AND The user story function is more abstract than the associated activity elements. AND The associated activity elements are each in their own parallel alternative path.
Level 2	The user story is associated to one activity element. AND The user story function is relatively equally big as an activity element.
Level 3	The user story is associated to one activity element. AND The activity element has multiple associations. AND The user story function is more detailed than the associated activity element. AND The sequence of user stories associated to the same activity element is easily recognizable.

Table 12: Rules for generating additional cases

5.2.4 Operation

The workbooks were printed in landscape orientation on both sides of the paper. In this way, two pages were seen at the same time (e.g. pages 2 and 3, pages 4 and 5, etc.). The page on the left-hand side was used to present the material. The workbooks were given to the students so that two students who were neighbors did not have the same workbook. We gave them instructions not to flip forward unless they had been told to do so and to work from the beginning through to the end of the workbook. They were not allowed to flip back the pages.

The experiment was executed with seven repetitions in approximately 70 minutes, 50 minutes of which was used for solving the workbooks. Three students returned their material untouched. Table 13 presents the experimental protocol.

Step	Description of the step	Timing
1.	The researcher explained the purpose, structure and rules of the experiment.	App. 5 min
2.	Students solved questions about their background.	Max. 2 min
3.	The researcher gave an oral explanation of the example material.	App. 7 min
4.	The students solved the comprehension test of the guided treatment's case.	Max. 7 min
5.	The researcher gave a verbal explanation of the example questions.	App. 7 min
6.	Students solved the problem-solving test of the guided treatment's case.	Max. 12 min
7.	Students solved the recall test of the guided treatment's case.	Max. 5 min
8.	Students solved questions about the perceived ease of use and usefulness of the guided treatment's case.	Max. 5 min
9.	The students solved the comprehension test of the unguided treatment's case.	Max. 7 min
10.	Students solved the problem-solving test of the unguided treatment's case.	Max. 12 min
11.	Students solved the recall test of the guided treatment's case.	Max. 5 min
12.	Students solved questions about the perceived ease of use and usefulness of the guided treatment's case.	Max. 5 min

Table 13: Experimental protocol for experiment 1

5.2.5 Subjects

The subjects in our experiment were 127 undergraduate students attending the Business Information Systems II course at Vienna University of Economics and Business in Austria in April 2015. Each student was attending one of seven classes. The classes were managed by six different lecturers. Given that the students had learned about business process management before, that the understanding of the concept of a user story is relatively simple, and that knowledge about the guided case was given on the spot, the subjects in our experiment were not required to be experts in either BPMN modeling or

agile development with user stories. The students were familiar with another modeling notation so reading simple BPMN models was no trouble for them. Further, they were informed about the INVEST qualities that should be met by a user story, and about our assumption that all the user stories in the given material meet those qualities. The selected students are likely to be good enough representatives of development project stakeholders who want to communicate about the execution order and integration dependencies among defined user stories [144].

Background questions were asked about gender, nationality, age, perceived knowledge of the English language and familiarity with user stories, business process models, BPMN and bank-specific processes. The group included 85 male and 41 female students while one subject did not specify his or her gender. More than 90% of the students were under the age of 26. Our experiment was in English but less than 1% of our students were native English speakers. Nevertheless, 90.5% marked their perceived familiarity with the English language on a Likert scale (1=poor, 7=excellent) with a grade of 4 or more. Moreover, 37.8% of students had never heard about user stories before while 58.3% marked grades 2 to 4 (Likert scale: 1=not at all familiar, 7=very familiar). Further, 85% of the students marked their perceived familiarity with business process models with a grade of 4 and less (Likert scale: 1=poor, 7=excellent). In addition, 42.5% of the students had never learned about the BPMN notation before, while another 48.8% of the students graded their knowledge with scores between 2 to 4 (Likert scale: 1=poor, 7=excellent). Moreover, 78.7% of the students scored their perceived familiarity with the specific business process “getting a new MasterCard” with 4 or less (Likert scale: 1=not at all familiar, 7=very familiar), and 92.9% of the students gave a grade of 4 or less for perceived familiarity with the business process “getting a non-standard account limit” (Likert scale: 1=not at all familiar, 7=very familiar).

5.2.6 Results

As Table 14 shows, the maximum scores of Case 1 and Case 2 were not equal. That is why we normalized scores for each of the problem-solving answers, the sum of comprehension answers and the sum of recall master answers to 100 for both Case 1 and Case 2.

Variable	Case 1 original max score	Case 2 original max score	Recalculated max score used for statistical analysis
Template1_ABSTRus	15	3	100
Template1_EQUALus	10	9	100
Template1_DETAILus	25	10	100
Template2_ABSTRus	3	4	100
Template2_EQUALus	1	1	100
Template2_DETAILus	1	1	100
Comprehension_sum	20	16	100
Recall_sum	8	8	100

Table 14: Original and recalculated maximum scores of questions

Table 15 shows the descriptive statistics for the problem-solving questions generated with Template 1 for both guided and unguided treatment. To provide answers to the questions generated with Template 1, the subjects needed to have an understanding of the execution order dependencies. We generated three problem-solving questions which were set for three different associations: first, for the association characterized with granularity level 1 (question Template1_ABSTRus), second, for associations characterized with granularity level 2 (question Template1_EQUALus) and third, for associations characterized with granularity level 3 (question Template1_DETAILus). For the guided treatment, the mean score for Template1_ABSTRus is 74.60 and for the unguided treatment it is 18.14. The mean score for Template1_EQUALus is 71.92 for the guided treatment and 12.50 for the unguided treatment. Finally, the mean score for the Template1_DETAILus is 74.28 for the guided treatment and 18.74 for the unguided treatment.

Treatment / factor	Dependent variable	N	Max score	Mean	Median
Guided	Template1_ABSTRus	127	100	74.60	80.00
	Template1_EQUALus	127	100	71.92	77.80
	Template1_DETAILus	127	100	74.28	73.30
Unguided	Template1_ABSTRus	127	100	18.14	10.00
	Template1_EQUALus	127	100	12.50	0.00
	Template1_DETAILus	127	100	18.74	6.70

Table 15: Descriptive statistics for the understanding of the execution order dependencies

Similarly, Table 16 shows descriptive statistics of the answers to problem-solving questions generated with Template 2 which required an understanding of the integration dependencies. With Template 2, we generated three problem-solving questions which were set for three different associations: first, for the association characterized with granularity level 1 (question Template1_ABSTRus), second, for associations

characterized with granularity level 2 (question Template1_EQUALus) and, third, for associations characterized with granularity level 3 (question Template2_DETAILus). For the guided treatment the mean score for Template2_ABSTRus is 64.90 and for the unguided treatment it is 21.00. The mean score for Template2_EQUALus is 64.57 for the guided treatment and 47.24 for the unguided treatment. Finally, the mean score for Template2_DETAILus is 74.80 for the guided treatment and 49.61 for the unguided treatment.

Treatment / factor	Dependent variable	N	Max score	Mean	Median
Guided	Template2_ABSTRus	127	100	64.90	100.00
	Template2_EQUALus	127	100	64.57	100.00
	Template2_DETAILus	127	100	74.80	100.00
Unguided	Template2_ABSTRus	127	100	21.00	0.00
	Template2_EQUALus	127	100	47.24	0.00
	Template2_DETAILus	127	100	49.61	0.00

Table 16: Descriptive statistics for the understanding of the integration dependencies

Table 17 shows the descriptive statistics for the comprehension (Comprehension_sum) and recall tasks (Recall_sum). The mean score of the Comprehension_sum is 65.07 for the guided treatment and 51.74 for the unguided treatment. Next, the mean score for the Recall_sum is 49.41 for the guided treatment, and 46.85 for the unguided treatment.

Treatment	Control variable	N	Max score	Mean
Guided	Comprehension_sum	127	100	65.07
	Recall_sum	127	100	49.41
Unguided	Comprehension_sum	127	100	51.74
	Recall_sum	127	100	46.85

Table 17: Descriptive statistics for the comprehension and recall tests

Table 18 shows descriptive statistics for measures for the perceived ease of use and usefulness.

Informative variable	Guided treatment		Unguided treatment	
	Mean	Std. dev.	Mean	Std. dev.
EaseOfUnderstand (1-7)	4.13	1.67	2.5	1.51
Frustrating (1-7)	3.76	1.67	4.91	1.91
EaseOfUse (1-7)	3.98	1.54	2.31	1.20
EaseOfLearning (1-7)	4.28	1.59	2.49	1.33

Table 18: Descriptive statistics for measures of ease of use and usefulness

We used the Kolmogorov-Smirnov test to investigate the normality of our data. The Kolmogorov-Smirnov test showed that the data for all variables listed in Table 8 are non-parametric. The distribution scores for all dependent variables are significantly different from a normal distribution. Therefore, we use non-parametric test in the continuation.

The Mann-Whitney test is a non-parametric equivalent to the independent t-test. Table 19 summarizes the data after the scores for Template1_ABSTRus, Template1_EQUALus, and Template1_DETAILus were ranked. The mean rank for Template1_ABSTRus exposed to the guided treatment is 184.70 and for the unguided treatment it is 70.30. The mean rank for Template1_EQUALus exposed to the guided treatment is 185.67 and for the unguided treatment it is 69.33. The mean rank for Template1_DETAILus exposed to the guided treatment is 180.44 and for the unguided treatment it is 74.56.

	Treatment	N	Mean rank	Sum of ranks
Template1_ABSTRus	Guided	127	184.70	23456.50
	Unguided	127	70.30	8928.50
	Total	254		
Template1_EQUALus	Guided	127	185.67	23580.00
	Unguided	127	69.33	8805.00
	Total	254		
Template1_DETAILus	Guided	127	180.44	22915.50
	Unguided	127	74.56	9469.50
	Total	254		

Table 19: Mann-Whitney test ranks for execution order dependency understanding

Table 20 shows the Mann-Whitney test statistics for the three questions generated with Template 1. The grouping variable is TreatmentCode. Scores for Template1_ABSTRus exposed to the guided treatment (Mdn=80.00) were significantly higher than those exposed to the unguided treatment (Mdn=10.00) with U=800.50, z=-12.44, p=<0.05, r=-0.78. The effect accounts for 60.8% of the variance, which is generally treated as a large effect size. These results support hypothesis H1.

Scores for Template1_EQUALus exposed to the guided treatment (Mdn=77.8) were significantly higher than those exposed to the unguided treatment (Mdn=0.00) with U=677.00, z=-12.78, p=<0.05, r=-0.80. The effect accounts for 64% of the variance, which is generally treated as a large effect size. These results support hypothesis H2.

Scores for Template1_DETAILus exposed to the guided treatment (Mdn=73.3) were significantly higher than those exposed to the unguided treatment (Mdn=6.70) with U=1341.50, z=-11.61, p=<0.05, r=-0.73. The effect accounts for 53.3% of the variance, which is generally treated as a large effect size. These results support hypothesis H3.

	Template1_ABSTRus	Template1_EQUALus	Template1_DETAILus
Mann-Whitney U	800.50	677.00	1341.50
Wilcoxon W	8928.50	8805.00	9469.50
Z	-12.44	-12.78	-11.61
R	-0.78	-0.80	-0,73
Asymp.Sig (2-tailed)	0.000	0.000	0.000
Exact Sig (2-tailed)	0.000	0.000	0.000
Exact Sig (1-tailed)	0.000	0.000	0.000
Point Probability	0.000	0.000	0.000

Table 20: Mann-Whitney test statistics for execution order dependency understanding

Table 21 summarizes the data after the scores for Template2_ABSTRus, Template2_EQUALus, and Template2_DETAILus were ranked. The mean rank for Template2_ABSTRus exposed to the guided treatment is 163.93 and for the unguided treatment it is 91.07. The mean rank for Template2_EQUALus exposed to the guided treatment is 138.50 and for the unguided treatment it is 16.50. The mean rank for Template2_DETAILus exposed to the guided treatment is 143.50 and for the unguided treatment it is 111.50.

	Treatment	N	Mean rank	Sum of ranks
Template2_ABSTRus	Guided	127	163.93	20819.00
	Unguided	127	91.07	11566.00
	Total	254		
Template2_EQUALus	Guided	127	138.50	17589.00
	Unguided	127	116.50	14795.50
	Total	254		
Template2_DETAILus	Guided	127	143.50	18224.50
	Unguided	127	111.50	14160.50
	Total	254		

Table 21: Mann-Whitney test ranks for integration dependency understanding

Table 22 shows the Mann-Whitney test statistics for three questions generated with Template 2. The grouping variable is TreatmentCode. Scores for Template2_ABSTRus exposed to the guided treatment (Mdn=100.00) were significantly higher than those exposed to the unguided treatment (Mdn=0.00) with U=3438.00, z=-8.28, p=<0.05, r=-0.52. The effect accounts for 27% of the variance, which is generally treated as a large effect size. These results support hypothesis H4.

Scores for Template2_EQUALus exposed to the guided treatment (Mdn=100.00) were significantly higher than those exposed to the unguided treatment (Mdn=0.00) with U=6667.00, z=-2.77, p=<0.05, r=-0.17. The effect accounts for 2.9% of the variance, which is generally treated as a small effect size. These results support hypothesis H5.

Scores for Template2_DETAILus exposed to the guided treatment (Mdn=100.00) were significantly higher than those exposed to the unguided treatment (Mdn=0.00) with U=6032.50, z=-4.13, p=<0.05, r=-0.26. The effect accounts for 6.8% of the variance, which is generally treated as a small effect size. These results support hypothesis H6.

	Template2_ABSTRus	Template2_EQUALus	Template2_DETAILus
Mann-Whitney U	3438.00	6667.50	6032.50
Wilcoxon W	11566.00	14795.50	14160.50
R	-0.52	-0.17	-0.26
Z	-8.28	-2.77	-4.13
Asymp.Sig (2-tailed)	0.000	0.006	0.000
Exact Sig (2-tailed)	0.000	0.008	0.000
Exact Sig (1-tailed)	0.000	0.004	0.000
Point Probability	0.000	0.002	0.000

Table 22: Mann-Whitney test statistics for integration dependency understanding

5.2.7 Validity of results

External validity threats are conditions that limit the ability to generalize the results of the experiment to industrial practice [145]. It is analyzed according to three aspects. The first aspect is concerned with how generalizable the results are to the wider population [145]. According to Gemino and Wand [38], two dimensions reflect several types of subjects in the conceptual modeling phase of scope definition: prior domain knowledge and prior modeling knowledge. We perceive our subjects as experts in modeling knowledge and as novices in prior domain knowledge. Similar to Batra et al. [144], we generalize the subjects to the population of typical information systems users who are involved in the analysis and design process, and need to communicate about the execution order and integration dependencies among previously defined user stories.

The second aspect of the external validity of the presented research relates to the extent to which the material used is representative of real-world material [145]. The business process models used for Case 1 and Case 2 are part of a real business process model taken

from a repository of a bank, which increases the external validity. The business process models were made for the purpose of business analysis and not for development reasons. This makes them perfect for studying the possibilities of reusing the existing documentation. On the other hand, our user story models are fictional. Their creation was inspired by the chosen business process models which reduces the external validity.

Finally, the third important aspect of the external validity relates to the time of carrying out the experiment [145]. All seven repetitions of the experiment were conducted in the same week. Nevertheless, some classes were scheduled in the mornings (starting at 8:00 at the earliest) and some in the afternoons (starting at 17:00 at the latest) which could have influenced how tired the students were when participating in the experiment. We did not collect any information about their feelings at the moment the experiment was conducted.

Internal validity demands that the treatment causes the effect [145]. We considered two aspects. The first aspect of internal validity relates to maturation which has the effect that the subject reacts differently as time passes [145]. Our subjects were given verbal explanations during the guided treatment. On the other hand, they were affected negatively when taking the unguided treatment since it was always scheduled as the second one and it did not contain any verbal explanations. Therefore, the students could have felt tired by the time they engaged in the unguided treatment. Nevertheless, the maximum time for each treatment was set at 24 minutes and all subjects in the class started both the first and second case together. They were encouraged to work through to the end of the workbook and were additionally motivated with bonus points for good performances in both treatments.

The second aspect of internal validity relates to resentful demoralization [145]. This means that a student exposed to less desirable treatment may give up on finishing the workbooks or not perform as well as they could have. All students provided at least one entry for the unguided treatment. Further, the number of subjects who provided no entries for any the six problem-solving questions of the unguided treatment is 10. In order to investigate how much effect on the overall results those 10 subjects would have we recalculated all statistics with them excluded. The results showed that the data with 117 subjects support hypotheses H1, H2, H3, and H4 with a large effect size just like the data with 127 subjects do. Similarly, the data with 117 subjects support hypotheses H5 and H6 with a small effect size just like the data with 127 subjects do. We thus conclude that the bias in the perception of the subjects is insignificant.

Conclusion validity is concerned with the relationship between the treatment and the results, and the conclusions drawn from the results [145]. We discuss three aspects. The first aspect concerns the appropriateness of the statistical tests. A Kolmogorov-Smirnov test confirmed that the normality assumption did not hold for any of the measures. Therefore, we used a non-parametric Mann-Whitney test [146].

The second aspect concerns the reliability of the experimental replications [145]. As mentioned, the experiment was conducted in seven classes which were managed by one of six different lecturers. Table 23 shows descriptive statistics for the seven repetitions of the experiment. In addition, we report the results of Levene's test of homogeneity of variance in Table 24. Therefore, we summed up all the scores of the variables listed in Table 14 for the guided and unguided treatments of each workbook. We grouped the scores of the subjects' guided and unguided treatments by the seven classes. For the variable Guided_sum, the variances were not significantly different for all seven classes, namely C1, C2, C3a, C3b, C4, C5, C6, $F(6,120)=1.62$; $p>0.05$. Also for the variable Unguided_sum, the variances for all seven classes were not significantly different, $F(6,120)=1.43$; $p>0.05$. We conclude that there are no significant statistical differences between the classes. Therefore, the replications are reliable.

Class' code	No. of subj.	Guided treatment (Guided_sum)		Unguided treatment (Unguided_sum)	
		Mean	Std. Dev.	Mean	Std. Dev.
C1	25	556.30	151.02	247.50	134.63
C2	11	593.75	169.59	215.64	141.72
C3a	21	488.43	183.56	206.34	137.83
C3b	20	526.01	138.67	282.08	145.60
C4	19	492.07	163.10	298.78	114.72
C5	18	622.54	123.19	340.87	153.30
C6	13	519.13	154.77	262.48	92.10

Table 23: Descriptive statistics for the experiment's seven repetitions

The third aspect concerns the reliability of the experimental measures [145]. The idea is that measures which can be repeated with the same results are more reliable. Our workbooks consisted of two different cases managed by two different treatments. Table 25 shows descriptive statistics for Case 1 and Case 2. For Case 1, the average score for the guided treatment is higher (Guided_sum=528.25) than for the unguided treatment (Unguided_sum=224.76). For Case 2, the average score for the guided treatment is higher (Guided_sum=550.63) than for the unguided treatment (Unguided_sum=307.53). We conducted Levene's test of homogeneity of variances for Case 1 and Case 2, see Table 26. For the Guided_sum, the variances were not significantly different for Case 1 and

Case 2, $F(1,125)=1.9$; $p>0.05$. Therefore, the homogeneity of variances suggests that both Case 1 and Case 2 gave similar results when the subjects were exposed to the guided treatment. On the other hand, for the unguided treatment we cannot draw a similar conclusion. For Unguided_sum, the variances were significantly different for Case 1 and Case 2, $F(1,125)=10.52$; $p<0.01$, which increases the threat to the reliability of the measures and, therefore, represents a limitation of our research.

		Levene statistic	df1	df2	Sig.
Guided_sum	Based on Mean	1.623	6	120	0.147
	Based on Median	1.075	6	120	0.381
	Based on Median and with adjusted df	1.075	6	105.024	0.382
	Based on trimmed mean	1.565	6	120	0.163
Unguided_sum	Based on Mean	1.435	6	120	0.207
	Based on Median	0.936	6	120	0.472
	Based on Median and with adjusted df	0.936	6	102.301	0.473
	Based on trimmed mean	1.374	6	120	0.231

Table 24: Test of the homogeneity of variances for the seven repetitions of the experiment

Case	Guided treatment (Guided_sum)			Unguided treatment (Unguided_sum)		
	Mean	Std. dev.	N	Mean	Std. dev.	N
Case 1	528.25	171.31	63	224.76	110.57	64
Case 2	550.63	145.48	64	307.53	150.64	63

Table 25: Descriptive statistics for the different cases the subjects took

		Levene Statistic	df1	df2	Sig.
Guided_sum	Based on Mean	1.902	1	125	0.170
	Based on Median	1.903	1	125	0.298
	Based on Median and with adjusted df	1.903	1	112	0.298
	Based on trimmed mean	1.731	1	125	0.191
Unguided_sum	Based on Mean	10.521	1	125	0.002
	Based on Median	10.278	1	125	0.002
	Based on Median and with adjusted df	10.278	1	121	0.002
	Based on trimmed mean	10.526	1	125	0.002

Table 26: Test of the homogeneity of variances for Case 1 and Case 2

Construct validity threats relate to the design of the experiment and to the social factors [145]. We discuss three aspects. The first aspect is about the risk of a measurement bias [145]. In our experiment we used different types of measures (e.g. comprehension instruments, recall instruments, background questions) that were cross-checked against each other. A second aspect relates to a threat where the subject is involved in more than one study since treatments from the different studies may interact [145]. In our study, the subjects were involved in both guided and unguided treatments but with different cases. A third aspect is related to a threat called experimenter expectancies [145]. Experimenters can bias the results of a study both consciously and unconsciously based on what they expect from the experiment. The scoring of our workbooks was done by one of the authors, which increases the threat. Nevertheless, the scoring was based on a previously prepared list of master answers for both Case 1 and Case 2. The answers were set to be short and straightforward.

5.3 Experiment 2: BPMN material versus text-written use case material

5.3.1 Tasks

There were two task prepared for our subjects. The first one was to elicit user stories from the given material. Subjects were given instructions to perform all steps of the basic sub-elicitation process, and first two steps of the higher-level sub-elicitation. Our focus was on measuring how many correct user stories were elicited. We encouraged subjects not to leave the test empty. For control purposes we counted wrong (that is with user role or function entry empty; or with one or both of the entries incorrect) and empty user stories (that is without both user role and function entries).

After the first task was performed, subjects got an updated version of the material for the second task which was to interpret the given material and show understanding about the execution order and integration dependencies of a specific user story. Mayer [39] promotes problem-solving tests to evaluate how well the learner understands the material. As in experiment 1, measurement instrument is a questionnaire and the measures are its scores. As suggested by Gemino and Wand [38], we measured the subjects' performance in comprehension, problem-solving and recall tests.

5.3.2 Questionnaire

Workbooks A, B, C and D are composed of background questions and two treatments. Each treatment is composed of five tests: elicitation, comprehension, problem-solving, recall, and perceived ease of use and usefulness test. The first test for each treatment was an elicitation test on the given material. The second test was a comprehension test on given upgraded material. This test was used for control purposes since we wanted to ensure that the subjects made an effort and developed their cognitive model. Afterwards, the subjects took the problem-solving test. The problem-solving questions were used to support the hypotheses. Next, a recall test was used for control purposes only. The last set of questions for each treatment included questions to measure (perceived) ease of use and usefulness of the material available in the treatment.

By scoring a *comprehension test* we measure how well a subject can read the given material. The test questions can be answered by either looking at the list of user stories or looking at the BuPUS product. In the associated business process model, the user story dependencies can be visually interpreted while from the list of user stories they can only be textually interpreted. We used the following questions which we adapted from experiment 1:

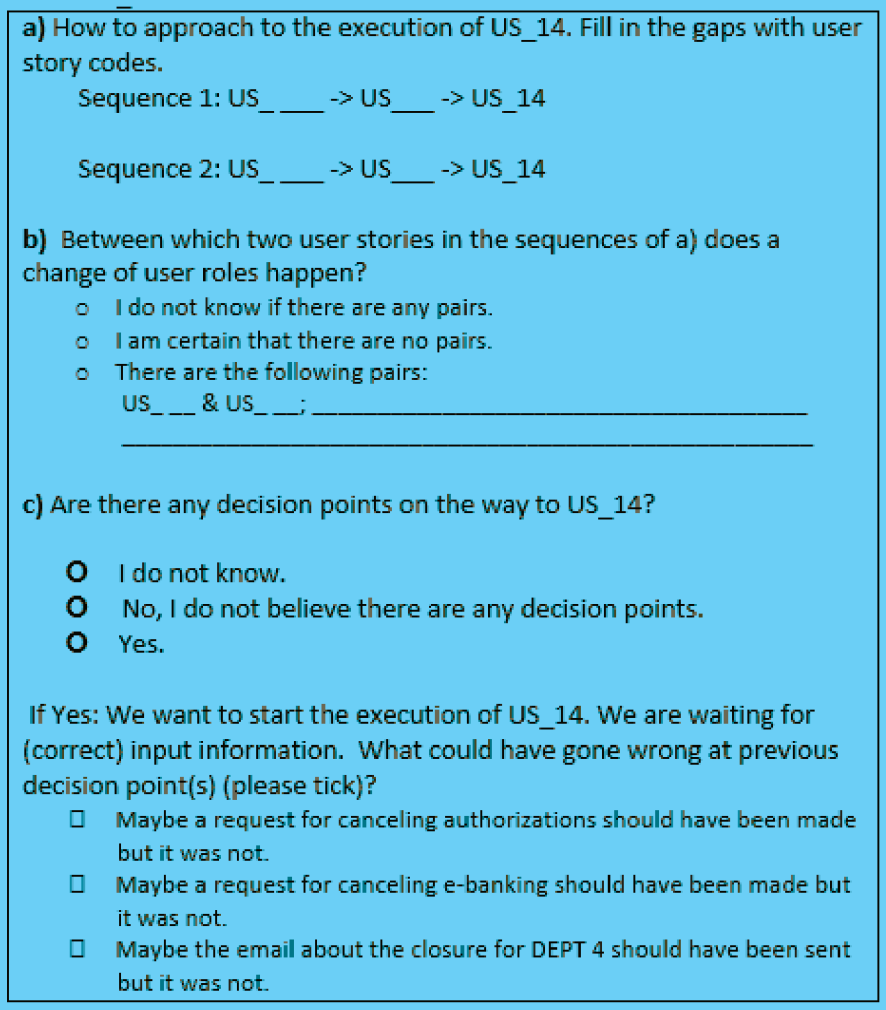
- Who participates in activities in the model (tick)?
- What is the sequence of activities about (tick)?
- How many user stories are in the model? Write down the number.

Similar to experiment 1, the *problem-solving test* consists of two types of questions where one type evaluates the understanding of execution order dependencies and the other of integration dependencies. Each type has its own template. The first problem-solving template question focuses on the understanding of the execution order of a specific set of user stories.

Template 1: What was previously done wrong (or not done at all) so the activity <specific user story function> cannot get started?

We adapted Template 1 and broke it down into three specific sub-questions. As shown in Figure 27, under sub-question: a) the subject needs to fill in the user story codes about the execution order (with the subjects we used a word *sequence*) of user stories which can lead to the execution of user story US_14. The subjects can recognize that there are two different execution orders. Subjects with BPMN material are able to see the execution orders by following the flow arrows between the associated business process activities.

On the other hand, subjects with UC material need to read the execution orders by following the sequence of rows. Next, sub-question b) refers to the execution orders of a). In b) the subjects need to find out where in the execution order the work is passed to another user role. The answer requires checking if the first and the second user story in the execution order have different user roles, then the second and the third, then the third and the fourth and so forth. Subjects with BPMN material should focus on reading swimlanes, while subjects with UC material should focus on reading the actor column. Lastly, sub-question c) also refers to the execution orders of a). The subjects need to understand if there are any business decisions taken which influenced the further execution order. Subjects with BPMN material could recognize the decision points by understanding the OR gateway elements, while subjects with UC material needed to understand these from the section Additional information. Points are given for each correctly filled in gap, for choosing the right radio button and for ticking the correct check boxes (see Figure 32).


a) How to approach to the execution of US_14. Fill in the gaps with user story codes.

Sequence 1: US_ ___ -> US___ -> US_14

Sequence 2: US_ ___ -> US___ -> US_14

b) Between which two user stories in the sequences of a) does a change of user roles happen?

- o I do not know if there are any pairs.
- o I am certain that there are no pairs.
- o There are the following pairs:
  US_ __ & US_ __; _____
  _____

c) Are there any decision points on the way to US_14?

- O I do not know.
- O No, I do not believe there are any decision points.
- O Yes.

If Yes: We want to start the execution of US_14. We are waiting for (correct) input information. What could have gone wrong at previous decision point(s) (please tick)?

- ☐ Maybe a request for canceling authorizations should have been made but it was not.
- ☐ Maybe a request for canceling e-banking should have been made but it was not.
- ☐ Maybe the email about the closure for DEPT 4 should have been sent but it was not.

Figure 32: Example of Case "CLOSE" sub-questions used for answering a Template 1 question (caption from Workbook A)

For example, in Case "CLOSE" there is a problem-solving question: *What was previously done wrong (or not done at all) so the activity "send a notification about a change in the account's attribute (US_14)" cannot get started?* The answer to a Template 1 question is a combination of answers to sub-questions a), b), and c) in Figure 32. The answer is composed the same way as the answer in Figure 24.

The second problem-solving template question requires understanding of the integration dependencies of a specific user story. We adopted the Template 2 that we used in the first experiment. An example question is: *Sometimes the activity "write a notification letter about closure of the account to the customer (activity from US_14)" can be executed in different variations. Which user stories (codes) present those variations?* Figures 10 and 16 both show that neither BPMN nor UC material present the wanted variations of US_14. The answer to the question in Figure 33 consists of choosing the first radio button. For choosing the correct radio button the subject was given one point.

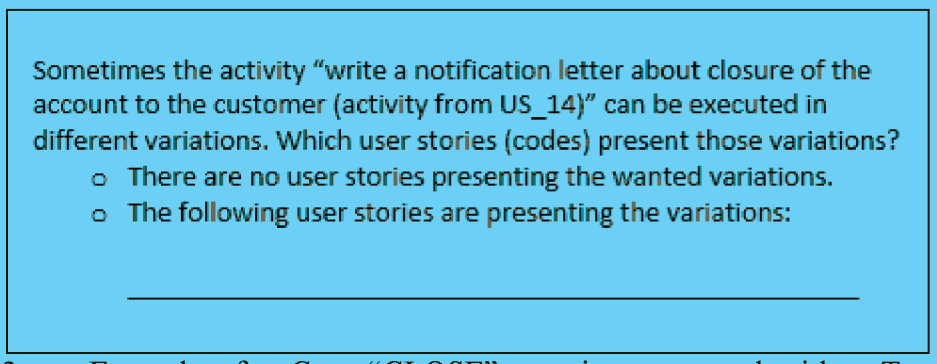


Figure 33: Example of a Case "CLOSE" question generated with a Template 2 question (caption from Workbook A)

Each problem-solving template question is used two times per case. The two times applies to two situations when a user story is: 1) more abstract than; and 2) approximately equally big as the corresponding activity description (or event description) in a business process model. Table 27 presents an overview of the questions for Case "CLOSE" and their relationship to the hypotheses.

Question	Hypothesis
Sub-questions of Template 1 addressed to US_14 (see Figure 32)	H7
Sub-questions of Template 1 addressed to US_17	H8
Template 2 addressed to US_14 (see Figure 33)	H9
Template 2 addressed to US_17	H10
Elicitation of user stories from a business process model (see Figure 15)	H11

Table 27: Problem-solving questions of Case "CLOSE" and their relationship to the hypotheses

The *recall test* measures how well the subjects remember the information about the domain. The subjects needed to fill in six empty gaps to get a maximum of six points. Finally, the last test was used to measure perceived *ease of use and usefulness* on a 7-point Likert scale with response options ranging from "strongly disagree" to "strongly agree". We adopted four question from Maes and Poels [143] as we did in experiment 1. This test was done twice for each of the treatments.

5.3.3 Materials

In our design for the second experiment, we used BPMN models from the same bank's repository as for experiment 1. Nevertheless, out of a set of available 24 models, we chose two new business process models for our second experiment: "close a bank account" (also referred as case "CLOSE"), and "open a bank account" (also referred to as case "OPEN"). We made modifications to this two models by following the seven process modeling guidelines [139]: we consistently used message flow arrows when the following activity was in another pool, we added merge gateways to every OR decision gateway, we translated all labels to English and we used verb-object activity labels. We have also hidden some business process parts to protect business secret. When making these modifications, we were careful not to change the information of the business process models but only its visual presentation. In both BPMN models "OPEN" and "CLOSE" we have chosen the same set of BPMN elements: activity, pool, lane, OR gateway, and flow arrow. An example of a business process model for Case "CLOSE" is available in Figure 15.

After the business process models were prepared, we created two lists of user stories for each case. Our assumption is that all user stories on the list meet the INVEST quality aspects and that they are selected to enter in the same development iteration. Same as for experiment 1, we incorporated the three characteristics of the list of user stories discussed in Section 2.2.3. The first characteristic is "the naming of the user story functions can

refer to different levels of abstraction". We associated the function to a business process activity element with one of the two levels of the association granularity in the following way:

- Level 1: A situation when a user story is more abstract than the corresponding activity in a business process model. This was done by describing the user story function on a higher level of abstraction than the associated activity element description. An example of a level 1 association is shown in Figure 15 where US_19 is associated to multiple activity elements.
- Level 2: A situation where a user story is approximately of equal abstraction as the corresponding activity in a business process model. This was done by describing the user story function and activity element description in a very similar way or exactly in the same way. An example of a level 2 association is shown in Figure 15 where US_14 is associated to one activity element.

The second characteristic "different synonyms can refer to the same artifact" is also integrated into our material, for example, on Figure 15 we use terms permissions and authorizations as synonyms. The third characteristic "user stories can take part in different execution orders" means that not all user stories from the list are associated to its activities elements in the BPMN model.

For preparing an associated business process model for case "OPEN", we followed the specific set of rules described in Table 12. Furthermore, when preparing text-written use case models for case "OPEN" and case "CLOSE" we aimed for informational equivalency with the BPMN models as discussed in Section 4.3.4.

5.3.4 Operation

The workbooks were printed in landscape orientation on both sides. In this way, two pages were seen at the same time (e.g. pages 2 and 3, pages 4 and 5, etc.). The page on the left-hand side was used to present the material. The workbooks were given to the students so that two students who were neighbors were not solving the same workbook. We gave them instructions not to flip forward unless they had been told to do so and to work from the beginning through to the end of the workbook. They were not allowed to flip the page back.

The experiment was executed in six repetitions which took between 67 and 74 minutes, maximum 48 minutes of which was used for solving the workbooks. About ten workbooks were returned untouched. Table 28 presents the experimental protocol. Steps

3-11 were performed when executing the first case. These steps were repeated as steps 12-20 for solving the second case.

Step	Description of the step	Timing
1.	A researcher makes an introduction to the experiment.	App. 5 min
2.	Students solve the background questions.	Max. 2 min
3.	A researcher gives instruction to the material.	App. 2 min
4.	Students solve the comprehension test.	Max. 2 min
5.	A researcher gives instruction about how to perform the elicitation of user stories from the given material.	App. 6 min
6.	Students solve the elicitation test.	Max. 8 min
7.	A researcher gives instruction about how to work with problem-solving questions created with Template 1.	App. 5 min
8.	Students solve the first half of the problem-solving test.	Max. 7 min
9.	A researcher gives instruction about how to work with problem-solving questions created with Template 2.	App. 1 min
10.	Students solve second half of the problem-solving test.	Max. 2 min
11.	Students solve the recall test and the perceived ease of use and usefulness.	Max. 4 min
12.	A researcher gives instruction to the new material.	App. 2 min
13.	Students solve the comprehension test.	Max. 2 min
14.	A researcher gives instruction about how to perform the elicitation of user stories from the given material.	App. 6 min
15.	Students solve the elicitation test.	Max. 8 min
16.	A researcher gives instruction about how to work with problem-solving questions created with Template 1.	App. 5 min
17.	Students solve the first half of the problem-solving test.	Max. 7 min
18.	A researcher gives instruction about how to work with problem-solving questions created with Template 2.	App. 1 min
19.	Students solve second half of the problem-solving test.	Max. 2 min
20.	Students solve the recall test and the perceived ease of use and usefulness.	Max. 4 min

Table 28: Experimental protocol for experiment 2

We collected 130 solved workbooks. We searched for outliers by looking at scores of subjects' comprehension text. We argue that the subjects should gain at least 9 points out

of 15 to give credible results. This way we eliminated 7 subjects because of their bad performance on the comprehension test with UC material, and additional 4 subjects for their bad performance on the comprehension test with BPMN material. Thus the statistical analysis included 119 workbooks.

5.3.5 Subjects

The subjects in our experiment were 119 undergraduate students attending the Business Information Systems II course at Vienna University of Economics and Business in Austria in November 2015. Each student was attending one of six classes. The classes were managed by five different lecturers. Given that the students had learned about business process management before, that the understanding of the concept of a user story is relatively simple, and that knowledge about both treatments was given on the spot, the subjects in our experiment were not required to be experts in either BPMN modeling, text-written use case modeling or agile development with user stories. Most of the students were familiar with another modeling notation so reading simple BPMN models or simple text-written use case models was no trouble for them. The selected students are likely to be good enough representatives of development project stakeholders who want to elicit user stories and communicate about the execution order and integration dependencies among user stories.

Background questions were asked about gender, nationality, age, perceived knowledge of the English language and familiarity with user stories, business process models, BPMN and bank-specific processes. The group included 76 male and 43 female students. More than 96.6% of the students were under the age of 26. Our experiment was in English. Nevertheless, 92.4% marked their perceived familiarity with the English language on a Likert scale (1=poor, 7=excellent) with a grade of 4 or more. Further, 98.3% of students had never heard about user stories before (Likert scale: 1=not at all familiar, 7=very familiar). Further, 86.5% of the students marked their perceived familiarity with business process management with a grade of 4 and less (Likert scale: 1=poor, 7=excellent). In addition, 47% of the students had never learned about the BPMN notation before while another 47.9% of the students graded their knowledge with scores between 2 to 4 (Likert scale: 1=poor, 7=excellent). Similarly, 39.5% of the students had never learned about the text-written use case models before, while another 52.1% of the students graded their knowledge with scores between 2 to 4 (Likert scale: 1=poor, 7=excellent). Further, 47% of the students scored their perceived familiarity with the specific business process “OPEN” with 4 or less (Likert scale: 1=not at all familiar, 7=very familiar), and 73.1%

of the students gave a grade of 4 or less (Likert scale: 1=not at all familiar, 7=very familiar) for perceived familiarity with the business process "CLOSE".

5.3.6 Results

As Table 29 shows, the maximum scores of Case "OPEN" and Case "CLOSE" were equal.

Variable	Case "OPEN"	Case "CLOSE"
Template1_EQUALus	7	7
Template1_ABSTRus	13	13
Template2_EQUALus	1	1
Template2_ABSTRus	2	2
Elicitation_correctUS	9	9
Elicitation_wrongUS	9	9
Elicitation_emptyUS	9	9
Comprehension_sum	15	15
Recall_sum	6	6

Table 29: Maximum scores of questions in Experiment 2

Table 30 shows descriptive statistics for the problem-solving questions generated with Template 1 for both treatments. To provide answers to the questions generated with Template 1, the subjects needed to have an understanding of the execution order dependencies. We generated two problem-solving questions which were set for two different associations: first, for the association characterized with granularity level 1 (question Template1_ABSTRus) and second for associations characterized with granularity level 2 (question Template1_EQUALus). For the treatment with BPMN material, the mean score for Template1_ABSTRus is 3.454 and for the treatment with UC material it is 2.479. The mean score for Template1_EQUALus is 8.201 for the treatment with BPMN material and 5.151 for the treatment with UC material.

Treatment	Dependent variable	N	Mean	Median
BPMN material	Template1_EQUALus	119	8.201	8.000
	Template1_ABSTRus	119	3.454	4.000
UC material	Template1_EQUALus	119	5.151	4.000
	Template1_ABSTRus	119	2.479	2.000

Table 30: Descriptive statistics for the understanding of the execution order dependencies for Experiment 2

Similarly, Table 31 shows descriptive statistics for answers to the problem-solving questions generated with Template 2. To provide those answers the subjects needed to have an understanding of the integration dependencies. With Template 2, we generated two problem-solving questions which were set for two different associations: first, for the association characterized with granularity level 1 (question Template1_ABSTRus), and

second, for the associations characterized with granularity level 2 (question Template1_EQUALus). For the treatment with BPMN material the mean score for Template2_ABSTRus is 0.933 and for the treatment with UC material it is 0.941. The mean score for Template2_EQUALus is 0.639 for the treatment with BPMN material and 0.655 for the treatment with UC material.

Treatment	Dependent variable	N	Mean	Median
BPMN material	Template2_EQUALus	119	0.639	1.000
	Template2_ABSTRus	119	0.933	1.000
UC material	Template2_EQUALus	119	0.655	1.000
	Template2_ABSTRus	119	0.941	1.000

Table 31: Descriptive statistics for the understanding of the integration dependencies for Experiment 2

Table 32 shows descriptive statistics for measurements taken for the elicitation task. To elicit user stories the subjects needed to follow one of the two elicitation techniques. For treatment with BPMN material, they used the technique for eliciting user stories from BPMN model, and for treatment with UC material, they used the technique for eliciting user stories from UC model. For both treatments we scored three variables. With variable Elicitation_correctUS we counted correctly defined user stories, with variable Elicitation_wrongUS we counted wrongly defined user stories, and finally, with Elicitation_emptyUS we counted user stories which were left completely empty. The subjects knew exactly how many user stories were expected to be elicited. For the treatment with BPMN material the mean score for Elicitation_correctUS is 5.882 and for the treatment with UC material it is 5.765. The mean score for Elicitation_wrongUS is 2.605 for the treatment with BPMN material and 2.639 for the treatment with UC material. And finally, for the treatment with BPMN material the mean score for Elicitation_emptyUS is 0.500 and for the treatment with UC material it is 0.600.

Treatment	Dependent variable	N	Mean	Std. dev.
BPMN material	Elicitation_correctUS	119	5.882	2.233
	Elicitation_wrongUS	119	2.605	2.030
	Elicitation_emptyUS	119	0.500	1.065
UC material	Elicitation_correctUS	119	5.765	2.016
	Elicitation_wrongUS	119	2.639	2.066
	Elicitation_emptyUS	119	0.600	1.159

Table 32: Descriptive statistics for elicitation measures (Experiment 2)

Table 33 shows the descriptive statistics for the comprehension (Comprehension_sum) and recall tasks (Recall_sum). The mean score for the Comprehension_sum is 13.462 for the treatment with BPMN material, and 12.235 for the treatment with UC material. Next,

the mean score for the Recall_sum is 3.689 for the treatment with BPMN material and 3.185 for the treatment with UC material.

Treatment	Control variable	N	Mean	Std. dev.
BPMN material	Comprehension_sum	119	13.462	1.364
	Recall_sum	119	3.689	1.494
UC material	Comprehension_sum	119	12.235	1.614
	Recall_sum	119	3.185	1.584

Table 33: Descriptive statistics for the comprehension and recall tests for Experiment 2

Table 34 shows descriptive statistics for measures for the perceived ease of use and usefulness.

Informative variable	BPMN material treatment		UC material treatment	
	Mean	Std. dev.	Mean	Std. dev.
EaseOfUnderstand (1-7)	3.874	1.571	2.840	1.396
Frustrating (1-7)	3.756	1.573	4.538	1.770
EaseOfUse (1-7)	3.924	1.445	2.823	1.363
EaseOfLearning (1-7)	4.244	1.501	3.151	1.527

Table 34: Descriptive statistics for measures of ease of usefulness for Experiment 2

We used the Kolmogorov-Smirnov test to investigate the normality of our data. The test shows that the data for all variables listed in Table 29 are non-parametric ($p<0.05$). The distribution scores for all dependent variables are significantly different from a normal distribution. Therefore, we use a non-parametric test in the continuation.

We conducted the Mann-Whitney test just like in the experiment 1. Table 35 summarizes the data after the scores for Template1_ABSTRus and Template1_EQUALus were ranked. The mean rank for Template1_EQUALus exposed to the treatment with BPMN material is 1845.97 and for the treatment with UC material it is 70.09. The mean rank for Template1_ABSTRus exposed to the treatment with BPMN material is 133.56 and for the treatment with UC material it is 105.44.

	Treatment	N	Mean rank	Sum of ranks
Template1_EQUALus	BPMN material	119	1845.97	17371.00
	UC material	119	90.09	11070.00
	Total	238		
Template1_ABSTRus	BPMN material	119	133.56	15893.50
	UC material	119	105.44	12547.50
	Total	238		

Table 35: Mann-Whitney test ranks for execution order dependency understanding (Experiment 2)

Table 36 shows the Mann-Whitney test statistics for the three questions generated from the Template 1. The grouping variable is TreatmentCode. Scores for Template1_EQUALus exposed to the BPMN material treatment (Mdn=8) were significantly higher than those exposed to the UC material treatment (Mdn=4) with U=3930.00, z=-5.951, p=<0.05, r=-0.39. The effect accounts for 15.2% of the variance, which is generally treated as a medium effect size. These results support hypothesis H7.

Scores for Template1_ABSTRus exposed to the BPMN material treatment (Mdn=4) were significantly higher than those exposed to the UC material treatment (Mdn=2) with U=5407.50, z=-3.191, p=<0.05, r=-0.25. The effect accounts for 6.2% of the variance, which is generally treated as a small effect size. These results support hypothesis H8.

	Template1_EQUALus	Template1_ABSTRus
Mann-Whitney U	3930.00	5407.50
Wilcoxon W	11070.00	12.547.50
Z	-5.951	-3.191
R	-0.39	-0.25
Asymp.Sig (2-tailed)	0.000	0.001
Exact Sig (2-tailed)	0.000	0.001
Exact Sig (1-tailed)	0.000	0.001
Point Probability	0.000	0.000

Table 36: Mann-Whitney test statistics for execution order dependency understanding (Experiment 2)

Table 37 summarizes the data after the scores for Template2_ABSTRus and Template2_EQUALus were ranked. The mean rank for Template2_EQUALus exposed to the treatment with BPMN material is 118.50 and for the treatment with UC material it is 120.50. The mean rank for Template2_ABSTRus exposed to the treatment with BPMN material is 119.26 and for the treatment with UC material it is 119.47.

	Treatment	N	Mean rank	Sum of ranks
Template2_EQUALus	BPMN material	119	118.50	14101.50
	UC material	119	120.50	14339.50
	Total	238		
Template2_ABSTRus	BPMN material	119	119.26	14191.50
	UC material	119	119.47	14249.50
	Total	238		

Table 37: Mann-Whitney test ranks for integration dependency understanding (Experiment 2)

Table 38 shows the Mann-Whitney test statistics for the two questions generated with Template 2. The grouping variable is TreatmentCode. Scores for Template2_EQUALus exposed to the BPMN material treatment (Mdn=1) were not significantly lower than those exposed to the UC material treatment (Mdn=1) with U=6961.50, z=-0.271, p>0.05, r=-0.018. These results do not support hypothesis H9.

Scores for Template2_ABSTRus exposed to the BPMN material treatment (Mdn=1) were not significantly lower than those exposed to the UC material treatment (Mdn=1) with U=7051.50, z=-0.058, p>0.05, r=-0.004. These results do not support hypothesis H10.

	Template1_EQUALus	Template1_ABSTRus
Mann-Whitney U	6961.50	7051.50
Wilcoxon W	14101.50	14191.50
Z	-0.271	-0.058
R	-0.018	-0.004
Asymp.Sig (2-tailed)	0.787	0.954
Exact Sig (2-tailed)	0.892	0.981
Exact Sig (1-tailed)	0.446	0.491
Point Probability	0.104	0.011

Table 38: Mann-Whitney test statistics for execution order dependency understanding (Experiment 2)

Table 39 summarizes the data after the scores for Evaluation_correctUS were ranked. The mean rank for Evaluation_correctUS exposed to the treatment with BPMN material is 123.88 and for the treatment with UC material it is 115.12.

	Treatment	N	Mean rank	Sum of ranks
Evaluation_correctUS	BPMN material	119	123.88	14742.00
	UC material	119	115.12	13699.00
	Total	238		

Table 39: Mann-Whitney test ranks for elicitation (Experiment 2)

Table 40 shows the Mann-Whitney test statistics for the elicitation task. Scores for Elicitation_correctUS exposed to the BPMN material treatment (Mdn=1) were not significantly higher than those exposed to the UC material treatment (Mdn=1) with U=6559.00, z=-0.999, p>0.05, r=-0.064. These results do not support hypothesis H11.

	Elicitation_correctUS
Mann-Whitney U	6559.00
Wilcoxon W	13699.00
Z	-0.999
R	-0.064
Asymp.Sig (2-tailed)	0.787
Exact Sig (2-tailed)	0.892
Exact Sig (1-tailed)	0.446
Point Probability	0.104

Table 40: Mann-Whitney test statistics for execution order dependency understanding (Experiment 2)

5.3.7 Validity of results

External validity threats are conditions that limit the ability to generalize the results of the experiment to industrial practice [145]. We have analyzed it according to three aspects as we have done it in experiment 1. The first aspect is concerned with how generalizable the results are to a wider population [145]. According to Gemino and Wand [38], two dimensions reflect several types of subjects in the conceptual modeling phase of scope definition: prior domain knowledge and prior modeling knowledge. We perceive our subjects as experts in modeling knowledge and as novices in prior domain knowledge. We generalize the subjects to the population of typical information system users who are involved in the analysis and design process, and need to communicate about the execution order and integration dependencies among previously defined user stories.

The second aspect of the external validity of the presented research relates to the extent to which the material used is representative of real-world material [145]. The business process models used for Case "OPEN" and Case "CLOSE" are part of a real business process model from a repository of a bank which increases the external validity. The business process models were made for the purpose of business analysis and not for development reasons. This makes them perfect for studying the possibilities of reuse of existing documentation. On the other hand, our text-written use case models are derived from the BPMN models as informational equivalent. Furthermore, our user story models are fictional. Their creation was inspired by the chosen business process models which reduces the external validity.

Finally, the third important aspect of the external validity relates to the time of carrying out the experiment [145]. All six repetitions of the experiment were conducted in the same week. Nevertheless, some classes were scheduled in the mornings (starting at 9:00 at the

earliest) and some in the afternoons (starting at 18:00 at the latest) which could have influenced how tired the students felt when participating in the experiment. We did not collect any information about their feelings at the moment the experiment was conducted.

Internal validity demands that the treatment causes the effect [145]. We considered two aspects. The first aspect of internal validity relates to maturation which has the effect that the subject reacts differently as time passes [145]. Our subjects were exposed to both treatments. Since the order of treatments is important, we prepared Workbooks A and B where the treatment with BPMN material was the first treatment taken, and Workbooks C and D where the treatment with UC material was the first. This way we made an effort to neutralize both the tiredness and the learning effect. Furthermore, the maximum time for each treatment was set at 23 minutes and all subjects in the class started both the first and second case together. They were encouraged to work through to the end of the workbook and were additionally motivated with bonus points for good performance in both treatments.

The second aspect of internal validity relates to resentful demoralization [145]. This means that a subject exposed to a less desirable treatment may give up on finishing the workbooks or not perform as well as he or she could have. We eliminated some of the workbooks in three steps. In step 1 the workbooks of subjects who did not work from the beginning until the end of the workbook were removed. In step 2 we removed those subjects who did not solve the workbooks seriously (e.g. making jokes etc.). In step 3 we removed those subjects who did not pass the comprehension test of one or both of the treatments. We ended up with 119 subjects which we included in our analysis.

Conclusion validity is concerned with the relationship between the treatment and the results, and the conclusions drawn from the results [145]. We discuss three aspects. The first aspect concerns the appropriateness of the statistical tests. A Kolmogorov-Smirnov test confirmed that the normality assumption did not hold for any of the measures. Therefore, we used a non-parametric Mann-Whitney test [146] just like in experiment 1.

The second aspect concerns the reliability of the experimental replications [145]. As mentioned, the experiment was conducted in six classes which were managed by one of five different lecturers. Table 41 shows descriptive statistics for the six repetitions of the experiment. In addition, we report the results of Levene's test of homogeneity of variance in Table 42. Therefore, we summed all scores of the variables, namely, Comprehension_sum, Template1_EQUALus, Template1_ABSTRus,

Template2_EQUALus, Template2_ABSTRus, Recall_sum, Elicitation_correctUS for both treatments and grouped the sums by the six classes. For the treatment with BPMN material (variable BPMN_sum), the variances were not significantly different for all six classes, namely L1, L2, L3, L4a, L4b, L5, $F(5,113)=1.01$; $p>0.05$. For the variable UC_sum, the variances for all six classes were also not significantly different, $F(5,113)=1.49$; $p>0.05$. We conclude that there are no significant statistical differences between the classes. Therefore, the replications are reliable.

Class' code	No. of subj.	Treatment with BPMN material (BPMN_sum)		Treatment with UC material (UC_sum)	
		Mean	Std. Dev.	Mean	Std. Dev.
L1	23	33.56	6.99	28.70	8.22
L2	22	35.22	7.69	27.72	6.00
L3	19	33.74	8.74	28.79	9.38
L4a	24	39.67	6.52	32.08	8.99
L4b	17	37.00	7.21	35.00	9.17
L5	14	39.00	9.81	31.21	8.43

Table 41: Descriptive statistics for the experiment's six repetitions

		Levene statistic	df1	df2	Sig.
BPMN_sum	Based on Mean	1.015	5	113	0.412
	Based on Median	0.569	5	113	0.724
	Based on Median and with adjusted df	0.569	5	84	0.724
	Based on trimmed mean	0.862	5	113	0.509
UC_sum	Based on Mean	1.495	5	113	0.197
	Based on Median	0.776	5	113	0.569
	Based on Median and with adjusted df	0.776	5	99	0.569
	Based on trimmed mean	1.477	5	113	0.203

Table 42: Test of the homogeneity of variances for the six repetitions of the experiment 2

The third aspect concerns the reliability of the experimental measures [145]. The idea is that measures which can be repeated with the same results are more reliable. Our workbooks consisted of two different cases managed by two different treatments. Table 43 shows descriptive statistics for Case "OPEN" and Case "CLOSE". For Case "OPEN", the average score for the treatment with BPMN material is higher (BPMN_sum=37.93) than for the treatment with UC material (UC_sum=31.74). For Case "CLOSE", the average score for the treatment with BPMN material is higher (Guided_sum=34.73) than for the treatment with UC material (UC_sum=28.96). We conducted Levene's test of homogeneity of variances for Case "OPEN" and Case "CLOSE", see Table 44. For the UC_sum, the variances were not significantly different for both cases, $F(1,117)=1.25$;

p>0.05. Therefore, the homogeneity of variances suggests that both cases gave similar results when the subjects were exposed to the treatment with UC material. On the other hand, for the treatment with BPMN material we cannot draw a similar conclusion. For BPMN_sum, the variances were significantly different for Case "OPEN" and Case "CLOSE", $F(1,117)=5.67$; $p<0.05$, which increases the threat to the reliability of the measures and, therefore, represents a limitation on our research.

Case	Treatment with BPMN material (BPMN_sum)			Treatment with UC material (UC_sum)		
	Mean	Std. dev.	N	Mean	Std. dev.	N
Case "OPEN"	37.93	6.93	62	31.74	8.93	62
Case "CLOSE"	34.73	8.55	57	28.96	7.98	57

Table 43: Descriptive statistics for the different cases the subjects took

		Levene Statistic	df1	df2	Sig.
BPMN_sum	Based on Mean	5.668	1	117	0.019
	Based on Median	3.662	1	117	0.058
	Based on Median and with adjusted df	3.662	1	98	0.059
	Based on trimmed mean	4.814	1	117	0.030
UC_sum	Based on Mean	1.251	1	117	0.266
	Based on Median	1.113	1	117	0.294
	Based on Median and with adjusted df	1.113	1	117	0.294
	Based on trimmed mean	1.194	1	117	0.277

Table 44: Test of the homogeneity of variances for Case 1 and Case 2

Construct validity threats relate to the design of the experiment and to the social factors [145]. We discuss three aspects. The first aspect is about the risk of a measurement bias [145]. In our experiment we used different types of measures (comprehension instruments, recall instruments, background questions) that were cross-checked against each other. A second aspect relates to a threat where the subject is involved in more than one study since treatments from the different studies may interact [145]. In our study, the subjects were involved in both guided and unguided treatments but with different cases. A third aspect is related to a threat called experimenter expectancies [145]. Experimenters can bias the results of a study both consciously and unconsciously based on what they expect from the experiment. The scoring of the workbooks was done by two independent evaluators, who were not aware of our hypotheses. Their scoring was based on a previously prepared list of master answers for both Case "OPEN" and Case "CLOSE".

Even though the answers were set to be short and straightforward, there were some differences in scoring between the two. One of the authors and one of the evaluators sat together on meetings which lasted approximately six hours in total. They discussed the final scores of the workbooks where the initial evaluations differ. They wrote down additional scoring rules which were then followed equally for both treatments.

5.4 Summary of Chapter 5

We ran two experiments. The first one's purpose was to find out whether the use of business process models is beneficial for the understanding of execution order and integration dependencies among user stories. The research confirmed our hypotheses. In the following experiment, we compared textual and visual business process models and evaluate their performance in understanding of the dependencies and in elicitation of user stories. We did not find significant differences either the elicitation or for the understanding of integration dependencies. However, there are significant differences in the understanding of execution order dependencies.

6 Conclusion

In this chapter, we discuss the results of the thesis. First, Section 6.1 summarizes the main results of experiment 1 and 2. Next, in Section 6.2 we discuss implications for both practice and research. And finally, in Section 6.3 we propose further work.

6.1 Summary of the results

The results are summarized of is in Table 45. Hypotheses H1, H2, H3, H7 and H8 are focused on understanding the execution order dependencies among user stories. The confirmation of hypotheses H1, H2, and H3 shows that the use of associated business process models together with the list of user stories increases the understanding of the execution order dependencies and that the effect size is large. With hypotheses H7 and H8 we demonstrated that the use of BPMN models increases the understanding of execution order dependencies more than the use of text-written use case models.

Hypotheses H4, H5, H6, H9 and H10 are focused on understanding the integration dependencies among user stories. The understanding happens in two steps: first, when the subject understands correctly if the list of user stories contains user stories, which are related on different levels of granularity, and, second, when the subject understands correctly how those user stories are related. The statistics supported four out of the five hypotheses. The confirmation of H4, H5, and H6 shows that the use of associated business

process models together with the list of user stories increases the understanding of the integration dependencies but with different effect sizes. For H4 there is a large effect size, while for H5 and H6 there is a small effect size. On the other hand, with hypotheses H9 and H10 we did not statistically support the idea that the use of BPMN models increases the understanding of integration dependencies more than the use of text-written use case models.

Hypothesis H11 is focused on the elicitation of user stories. We compared two techniques, where one used BPMN model and the other text-written use case model during the elicitation. We did not statistically support that the use of BPMN models produces more of the correct user stories than the use of text-written use case models.

Alternative hypothesis	Result	Effect size r
H1	Supported	-0.78 (large effect)
H2	Supported	-0.80 (large effect)
H3	Supported	-0.73 (large effect)
H4	Supported	-0.52 (large effect)
H5	Supported	-0.17 (small effect)
H6	Supported	-0.26 (small effect)
H7	Supported	-0,39 (medium effect)
H8	Supported	-0.25 (small effect)
H9	Not supported	/
H10	Not supported	/
H11	Not supported	/

Table 45: Summary of the results

6.2 Contributions

Our research has direct *contributions to the practice* of agile development. We addressed two important problems:

- First problem arises from a lack of user story elicitation approaches which would be based on reusing customer's existing documentation. Reusing existing documentations is one of the agile principles [7]. Our proposed BuPUS method reuses existing business process models to provide information for the elicitation of user stories. We suggest to practitioners to use business process models for eliciting user stories. According to our results, we perceive BPMN models and text-written use case models as equally good business process models for this task.
- Second problem arises from managing user stories and finding their context. One way to investigate the context is by understanding dependencies of a user story. Our proposed BuPUS method reuses existing business process models to provide

information about execution order and integration dependencies. We suggest to augment agile development projects that use user stories with business process models for explicitly defining the user stories' context.

The findings hold four important *contributions to research*:

- Agile practices for eliciting user stories from existing documentation are a rather unexplored topic in the literature and empirical studies of it are scarce. We have contributed an empirical study on user story elicitation by comparing *the technique for eliciting user stories from text-written use case models* and *the technique for eliciting user stories from BPMN models*. Text-written use case models were previously suggested to be used for elicitation of user stories by Ambler [7]. On the other hand, BPMN models were not yet proposed to be used in an approach for user story elicitation even though they often exist in the organization and are therefore available as existing documentation. Our findings show that elicitation of user stories from text-written use case models is approximately equally good as eliciting user stories from BPMN models. This is an important finding for agile development research community. The two techniques for eliciting user stories are part of our proposed BuPUS method.
- Agile practices on understanding requirements dependencies are a rather unexplored topic in the literature and empirical studies of it are scarce [17]. We have contributed an empirical study on requirements dependencies. We provide evidence of the importance of execution order and integration dependencies for the understanding of user stories in our first experiment. Findings from our first experiment underline the need to augment user stories with a business process perspective in order to provide the required context. Furthermore, in our second experiment we have contributed another empirical study by making two comparisons. The first comparison included *the technique for tracing execution order dependencies by using text-written use case models*, and *the technique for tracing execution order dependencies by using BPMN models*. While the second comparison included *the technique for tracing execution order dependencies by using text-written use case models*, and *the technique for tracing execution order dependencies by using BPMN models*. Text-written use case models were previously suggested to be used for understanding of the dependencies among user stories by Leffingwell [2]. On the other hand, the use of BPMN models was not yet suggested for this. Our research results show, that the understanding of

execution order dependencies is greater when using BPMN models than in the case of text-written use case models. We believe that this is because BPMN presents business decisions for executing alternative flows in a standardized way with a set of gateway elements. On the other hand, the understanding of integration dependencies is approximately equally good when using either BPMN model or text-written use case model. All the four techniques for understanding the dependencies are part of our proposed BuPUS method.

- Our research complements research into project-specific software development methods. BuPUS demonstrates the feasibility of respective proposals in the specific agile development context. In this way, our empirical evidence lends further support for the situational method engineering approach proposed by Bajec et al. [44].
- Our research also contributes to the ongoing research stream on empirical software engineering. We adopt task types to measure problem-solving, which were originally defined by Mayer to study his theory of multimedia learning [39]. These types of tasks are specifically useful in this context since they are highly relevant in practice for software requirements ("what can go wrong").

6.3 Limitations and future work

Here we discuss limitations and how future research can address them. The BPMN models that we used in our research were from real bank's repository and were initially made for the purpose of gaining an ISO certificate. We did not clearly specify initial granularity of our models. Moreover, we made an assumption that the model's activity descriptions were detailed enough for eliciting user stories with INVEST qualities. This assumption was made for research purposes but in real development projects it may not always be true. For future work we propose a research question how to evaluate the existing business process models whether they meet the needs of each INVEST qualities?

The second further research question is whether it would be possible to automatically generate a list of user stories from business process models and how. The recent research that developed methods to automatically generate textual descriptions from business process models [136] could help in this regard. Indeed, user stories of Level 2 have a potential to be automatically generated from the activity descriptions. Nevertheless, the activities' descriptions sometimes refer to the same object with different synonyms. In this context, it would be interesting to conduct empirical research on usefulness of an additional document - the dictionary.

Our third research question is how to use enterprise architecture models (especially models from the application and technology layer) to better understand other kind of dependencies among user stories. In this context, it would be interesting to conduct empirical research on dependencies beyond the process-related ones as in this thesis, for example by including architectural dependencies [147]. All of this research questions call for further experiments and an empirical research agenda.

Our fourth research question is how to use the grammar presented in BuPUS to effectively support the techniques with a new application. It would be interesting to make application's ER models and algorithms. In this way we could find out if our grammar misses any constructs or maybe has too many of them. Once having the application available it would be exciting to find an agile team who would be willing to use our BuPUS method. This research question calls for a case study.

7 References

[1] Avison, D. and Fitzgerald, G., Information systems development - methodologies, techniques, and tools, 4th edition, McGraw-Hill Education, Berkshire, UK, 2006, p. 645, ISBN: 0077096266.

[2] Leffingwell, D., Agile software requirements: lean requirements practices for teams, programs, and the enterprise. Agile software fevelopment series, ed. Cockburn, A. and J. Highsmith, Addison-Wesley, Boston, USA, 2011, ISBN: 9780321635846.

[3] Cohn, M., User stories applied for agile software development. The Addison-Wesley signature series, ed. Beck, K., M. Cohn, and M. Flower, Addison-Wesley, Boston, USA, 2004, p. 268, ISBN: 0321205685.

[4] Ambler, S. *Agile analysis*. 2013; Available from: http://www.agilemodeling.com/essays/agileAnalysis.htm.

[5] Lindström, H. and Malmsten, M., User-centred design and agile development: rebuilding the Swedish national union catalogue, Cataloging and indexing: challenges and solutions, ed. McIntosh, J., CRC Press, 2011, pp. 252-266, ISBN: 1466562064, 9781466562066.

[6] Dybå, T. and Dingsøyr , T., Empirical studies of agile software development: a systematic review, Information and software technology, 50 (9-10), 2008, pp. 833-859, DOI: 10.1016/j.infsof.2008.01.006.

[7] Ambler, S., Agile modeling: effective practices for extreme programming and the Unified Process, ed. Hudson, T., Wiley computer publishing, New York, 2002, p. 402, ISBN: 0-471-20282-7.

[8] Gottesdiener, E., Requirements by collaboration, Workshops for defining needs, Addison-Wesley, Boston, USA, 2011.

[9] Ambler, S. and Lines, M., Disciplined agile delivery. A practitioner's guide to agile software delivery in the enterprise, IBM Press, United States, 2014, ISBN: 978-0-13-281013-5.

[10] Caughlan, J. and Macredie, R.D., Effective communication in requirements elicitiation: a comparison of methodologies, Requirements Engineering, 11 (12), 2002, pp. 47-60.

[11] Kassel, N. and Malloy, B.A., An approach to automate requirements elicitation and specification, Proceedings of the 7th IASTED international conference on software engineering and application, CA, USA, 2003, pp. 544-549.
[12] Nuseibeh, B. and Easterbrook, S., Requirements engineering: a roadmap. The future of software engineering., ACM Press, New York, USA, 2000, pp. 37-46.
[13] Pitts, M.G. and Browne, G.J., Improving requirments elicitation: an empirical investigation of procedural prompts, Information systems journal, 17 (1), 2007, pp. 89-110, DOI: 10.1111/j.1365-2575.2006.00240.x.
[14] Van Buren, J. and Cook, D., Experiences in the adoption of requirements engineering technologies, J Def soft engineering, 11 (12), 1998, pp. 3.10.
[15] Kulshreshtha, V., Boardman, J., and Verma, D., The emergence of requirements networks: the case for requirements inter-dependencies, International journal of computer applications in technology, 45 (1), 2012, pp. 28-41, DOI: 10.1504/IJCAT.2012.050130.
[16] Lin, J., Yu, H., Shen, Z., and Miao, C., Using goal net to model user stories in agile software development, IEEE/ACIS international conference on software engineering, artificial intelligence, networking and parallel/distributed computing, IEEE, Las Vegas, NV 2014, pp. 1-6, DOI: 10.1109/snpd.2014.6888731.
[17] Martakis, A. and Daneva, M., Handling requirements dependencies in agile projects: a focus group with agile software development practitioners, International conference on research challenges in information science, IEEE, Paris, 2013, pp. 1-11, DOI: 10.1109/RCIS.2013.6577679.
[18] Li, J., Liming, Z., Ross, J., Liu, Y., Zhang, H., Wang, Q., and Li, M., An initial evaluation of requirements dependencies types in change propagation analysis, International conference on evaluation & assesment in software engineering, IET, Ciudad Real 2012, pp. 62-71, DOI: 10.1049/ic.2012.0009.
[19] Yan, Y., Li, S., and Sun, W., Dependency based technique for identifying the ripple effect on requirements evolution, Journal of software, 7 (3), 2012, pp. 544-550, DOI: 10.4304/jsw.7.3.544-550.
[20] Gupta, C., Singh, Y., and Chauhan, D., Dependency based process model for impact analysis: a requirement engineering perspective, International journal of computer applications, 6 (6), 2010, pp. 28-33.
[21] Harrmann, A. and Daneva, M., Requirements prioritization based on benefit and cost predistion: an agenda for future research, International requirements engineering, IEEE, Catalunya, 2008, pp. 125-134, DOI: 10.1109/RE.2008.48
[22] Cockburn, A., Agile software development. The agile software development series, ed. Cockburn, A. and J. Highsmith, Pearson education, Boston, USA, 2000.
[23] Strode, D.E., A dependency taxonomy for agile software development projects, Information systems frontiers, * (*), 2015, pp. 1-24, DOI: 10.1007/s10796-015-9574-1.
[24] Clarke, R.J. and Kautz, K., What's in a user story: IS development methods as communication, Information systems development: transforming organisations and society through information systems, ed. Strahonja, V., N. Vrček., D. Plantak Vukovac, C. Barry, M. Lang, H. Linger, and C. Schneider, Faculty of organization and informatics, Varaždin, Croatia, 2014, ISBN: 978-953-6071-43-2.
[25] Liskin, O., Pham, R., Kiesling, S., and Schneider, K., Why we need a granularity concept for user stories, Agile processes in software engineering and extreme programming, Springer international publishing, 2014, pp. 110-125, ISBN: 978-3-319-06861-9.
[26] Milicic, A., El Kadiri, S., Perdikakis, A., Ivanov, P., and Kiritsis, D., Toward the definition of domain concepts and knowledge through the application of the user

story mapping method International journal of product lifecycle management, 7 (1), 2014, pp. 3-16.
[27] Patton, J., User story mapping: discover the whole story, build the right product, ed. Economy, P., O'Reilly, 2014, ISBN: 978-1-491-90490-9.
[28] Saghafi, A. and Wand, Y., Do ontological guideines improve understandability of conceptual models? A meta-analysis of empirical work, Hawaii international conference on system sciences, IEEE, Waikoloa, HI 2014, DOI: 10.1109/HICSS.2014.567
[29] Lubke, D., Schneider, K., and Weidlich, M., Visualizing use case sets as BPMN processes, Requirements engineering visualization, IEEE, Barcelona, Catalunya 2008, pp. 21-25, DOI: 10.1109/REV.2008.8
[30] Mendling, J., Strembeck, M., and Recker, J., Factors of process model comprehension - findings from a series of experiments, Decision support systems, 53 (1), 2012, pp. 195-206, DOI: 10.1016/j.dss.2011.12.013.
[31] Ottensooser, A., Fekete, A., Reijers, H.A., Mendling, J., and Menictas, C., Making sense of business process descriptions: an experimental comparison of graphical and textual notations, Journal of systems and software, 85 (3), 2012, pp. 596-606, DOI: 10.1016/j.jss.2011.09.023.
[32] OMG, *Business process model and notation (BPMN), v2.0*. http://www.omg.org/spec/BPMN/2.0 2011, p. 550.
[33] Recker, J., Empirical investigation of the usefulness of Gateway constructs in process models, European journal of information systems, 22 (2013), 2013, pp. 673-689, DOI: 10.1057/ejis.2012.50.
[34] Reijers, H.A. and Mendling, J., A study into the factors that influence the understandability of business process models, IEEE transactions on systems, man and cybernetics, 41 (3), 2011, pp. 449-462, DOI: 10.1109/tsmca.2010.2087017.
[35] Bera, P., Burton-Jones, A., and Wand, Y., Research note - How semantics and pragmatics interacts in understanding conceptual models, Information systems research, 25 (2), 2014, pp. 401-419, DOI: 10.1287/isre.2014.0515
[36] Mendling, J., Reijers, H.A., and Recker, J., Activity labeling in process modeling: empirical insights and recommendations, Information systems, 35 (2010), 2010, pp. 467-482, DOI: 10.1016/j.is.2009.03.009.
[37] Mendling, J., Metrics for process models: empirical foundations of verification, error prediction, and guidelines for correctness. Lecture Notes in Business Information Processing, Springer Berlin Heidelberg, 2008, ISBN: 978-3-540-89223-6.
[38] Gemino, A. and Wand, Y., A framework for empirical evaluation of conceptual modeling techniques, Requirements engineering, 9 (4), 2004, pp. 248-260, DOI: 10.1007/s0O766-004-0204-6.
[39] Mayer, R.E., Multi-media learning, Second edition, Cambridge university press, USA, 2009, ISBN: 9780521735353.
[40] March, S.T. and Storey, V.C., Design science in information systems discipline: an introduction to the special issue on design science research, MIS Quarterly, 32 (4), 2008, pp. 725-730.
[41] Wieringa, R.J., Design science methodology for information systems and software engineering, Springer, Heidelberg, 2014, p. 332, ISBN: 9783662438398.
[42] Hevner, A., March, S., Park, J., and Ram, S., Design science in information systems research, MIS Quarterly, 28 (1), 2004, pp. 75-105.
[43] Iivari, J., A paradigmatic analysis of information systems as a design science, Scandinavian journal of information systems, 19 (2), 2007.

[44] Bajec, M., Vavpotic, D., and Krisper, M., Practice-driven approach for creating project-specific software development methods, Information and software technology, 49 (2007), 2007, pp. 343-365, DOI: 10.1016/j.infsof.2006.05.007.
[45] Object Management Group (OMG), *Business process model and notation (BPMN), version 2.0*. 2011, p. 538.
[46] Ludewing, J., Models in software engineering - an introduction, Software and systems modeling, 2 (1), 2003, pp. 5-14.
[47] Trkman, M., Mendling, J., and Krisper, M., The use of business process models for better understanding of integration and execution order dependencies of user stories, Information and software technology, 71 (March2016), 2016, pp. 58-76, DOI: doi:10.1016/j.infsof.2015.10.006.
[48] Trkman, M. and Krisper, M., BPMN - modeli procesov za strukturiran zajem uporabniških zgodb, Uporabna informatika, 19 (2), 2011, pp. 100-105.
[49] Trkman, M. and Mahnic, V., Structured approach for gathering user stories, International multiconference information society, Institut Jožef Stefan, Ljubljana, Slovenia, 2010, pp. 223-226.
[50] Pollard, C.E., Gupta, D., and Satzinger, J.W., Teaching Systems Development: A Compelling Case for Integrating the SDLC with the ITSM Lifecycle, Information Systems Management, 27 (2), 2010, pp. 113-122, DOI: 10.1080/10580531003684959.
[51] Shelly, G.B., Cashman, T.H., and Rosenblatt, H.J., Systems analysis and design, Fifth edition. Shelly cashman series, ed. Ouellette, C., Thomson, Boston, Massachusetts, 2003.
[52] Sutcliffe, A., User-Centred Requirements Engineering: Theory and Practice, ed. Martin, G., Springer, Kent, Great Britain, 2002.
[53] Turk, T., Analiza stroškov in koristi naložb v informatiko, Uporabna informatika, 13 (3), 2005, pp. 153-169.
[54] Lapalme, J., Gerber, A., Van der Merwe, A., Zachman, J., Vries, M.D., and Hinkelmann, K., Exploring the future of enterprise architecture: A Zachman perspective, Computers in Industry, 2015, pp. In press, DOI: http://dx.doi.org/10.1016/j.compind.2015.06.010.
[55] Thalheim, B., Entity-Relationship Modeling: Foundations of Database Technology, Springer, 2000.
[56] Varshney, G., Database Management System, Global Vision Publishing House, 2012, p. 314, ISBN: 8182203201, 9788182203204.
[57] Endsley, M.R. and Jones, D.G., Designing for situation awareness, An approach to user-centered design, Second Edition, SRC Press, Boca Raton, 2004, p. 396, ISBN: 13:978-1-4200-6358-5.
[58] Irani, Z., Themistocleous, M., and Love, P.E.D., The impact of enterprise application integration on information system lifecycles, Information & Management, 41 (2), 2003, pp. 177-187, DOI: http://dx.doi.org/10.1016/S0378-7206(03)00046-6.
[59] Krisper, M., Rupnik, R., Rožanec, A., Zrnec, A., Vavpotič, D., Osojnik, R., and Tomažič, R., eds. *EMRIS - enotna metodologija razvoja informacijskih sistemov*. Uvod, 1. zvezek, 2. izdaja, ed. Silič, M. and M. Krisper. 2003, CIP - Vlada republike Slovenije, Center Vlade RS za informatiko, Ljubljana, Slovenia. p. 149.
[60] Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., and Thomas, D. *Manifesto for Agile Software Development*. [cited september 2015; Available from: http://agilemanifesto.org/.

[61] Schwaber, K., Agile project management with Scrum. Developer Best Practices, Microsoft Press, Redmond, 2004, ISBN: 978-0735619937
[62] Beck, K. and Andres, C., Extreme programmin explained: embrace change, Addison-Wesley, Boston, 2005, ISBN: 978-0321278654
[63] Beck, K., Extreme programming explained: embrace change, Addison-Wesley, Boston, 1999, ISBN: 978-0201616415
[64] Hailpern, B. and Tarr, P., Model-driven development: The good, the bad, and the ugly, IBM Systems Journal, 45 (3), 2006, pp. 451-461, DOI: 10.1147/sj.453.0451.
[65] Cohn, M., Agile estimating and planning. The Robert C. Martin Series, Prentice Hall, NY, USA, 2011, p. 330.
[66] Buchman, J. and Ekadharmawan, C.H., Barriers to sharing domain knowledge in sotware domevelopment practice in SMEs, International workshop on knowledge collaboration in sotware development, 2009, pp. 2-16.
[67] Gottesdiener, E., Requirements by collaboration: getting it right the first time, IEEE Software, 20 (2), 2011, pp. 52-55.
[68] Vessey, I. and Conger, S.A., Learning to specify information requirements: the relationsip between application and methodology, Res comp sci, 1993, 1993, pp. 177-202.
[69] Hadar, I., Soffer, P., and Kenzi, K., The role of domain knowledge in requirements elicitation via interviews: an exploratory study Requirements engineering, 19 (2), 2014, pp. 143-159, DOI: 10.1007/s00766-012-0163-2.
[70] Rosemann, M., Vessey, I., Weber, R., and Raduescu, C., Aligning organizational requirements with enterprise systems capabilities: the role of domain-specific knowledge, Americas conference on information systems, Keystone, Colorado, 2007.
[71] Kahrti, V., Vessey, I., Ramesh, V., Clay, P., and Park, S., Understanding conceptual scemas: exploring the role of application and IS domain knowledge, Information systems research, 17 (1), 2006, pp. 81-99.
[72] Anton, A.I. and Potts, C., *The use of goals to surface requirements for evolving systems*, in *International conference on software engineering*. IEEE, 1998, p. 157-166.
[73] Burton-Jones, A. and Weber, R., Understanding relationships with attributes in entity-relationship diagrams, International conference of information systems, Atlanta, GA, 1999, pp. 214-228.
[74] Shaft, T.M. and Vessey, I., The relevance of application domain knowledge: characterizing the comuter program comprehension process, Journal of Management Information Systems, 15 (1), 1998, pp. 51-77.
[75] Shaft, T.M. and Vessey, I., The relevance of application domain knowledge: the case of computer program comprehension, Information systems research, 6 (3), 1995, pp. 286-299.
[76] Curtis, B., Krasner, H., and Iscoe, N., A field study on the software design process for large systems, Communications of the ACM, 31 (11), 1988, pp. 81-99.
[77] Pitts, M.G. and Browne, G.J., Stopping behavior of systems analysts during information requirements elicitation J Mag inf syst, 21, 2004, pp. 203-226.
[78] Kaiya, H. and Saeki, M., Using Domain Ontology as Domain Knowledge for Requirements Elicitation, Requirements Engineering, 14th IEEE International Conference, 2006, pp. 189-198, DOI: 10.1109/re.2006.72.
[79] Wagstrom, P. and Herbsleb, J., Dependency forecasting in the distributed agile organization, Communications of the ACM, 49 (10), 2006, pp. 55-56, DOI: 10.1145/1164394.1164420.
[80] Daneva, M., van der Veen, E., Amrit, C., Ghaisas, S., Sikkel, K., Kumar, R., Ajmeri, N., Ramteerthkar, U., and Wieringa, R., Agile requirements prioritization

in large-scale outsourced system projects: an empirical study, The journal of systems and software, 86 (2013), 2013, pp. 1333-1353, DOI: 10.1016/j.jss.2012.12.046.
[81] Cataldo, M., Mockus, A., Roberts, J.A., and Herbsleb, J.D., Software dependencies, work dependencies, and their impact on failures, IEEE transactions on software engineering, 35 (6), 2009, pp. 864-878, DOI: 10.1109/TSE.2009.42
[82] Coyle, S. and Canboy, K., A case study of risk management in agile systems development, European conference on information systems NUI Galway, Verona, Italy, 2009, pp. 2567-2578.
[83] Cervone, H.F., Project risk management, OCLC systems & services: international digital library perspectives, 22 (4), 2006, pp. 256 - 262, DOI: 10.1108/10650750610706970.
[84] Dimitrijević, S., Jovanović, J., and Devedžić, V., A comparative study of software tools for user story management, Information and software technology, 57, 2015, pp. 352-368, DOI: http://dx.doi.org/10.1016/j.infsof.2014.05.012.
[85] Rees, M.J., A feasible user story tool for agile software development?, Ninth Asia-Pacific Software Engineering Conference, 2002, pp. 22-30, DOI: 10.1109/apsec.2002.1182972.
[86] Maiden, N., Exactly How Are Requirements Written?, Software, IEEE, 29 (1), 2012, pp. 26-27, DOI: 10.1109/ms.2012.6.
[87] Bower, G.H., Balck, J.B., and Turner, T.J., Scripts in memory for text, Cognitive psychology, 11, 1979, pp. 177-220.
[88] daSilva, T.S., Martin, A., Maurer, F., and Silveira, M., *User-centered design and agile methods: a systematic review*, in *Agile conference (AGILE)*. IEEE: Salt Lake City, UT, USA, 2011, p. 77-86.
[89] Williams, L., Guest editor's introduction: the XP programmer: the few-minutes programmer, IEEE software, (3), 2003, pp. 16-20.
[90] Mishra, D., Mishra, A., and Yazici, A., Successful requirement elicitation by combining requirement engineering techniques, First International Conference on the Applications of Digital Information and Web Technologies, 2008. , 2008, pp. 258-263, DOI: 10.1109/icadiwt.2008.4664355.
[91] Garg, N., Agarwal, P., and Khan, S., Recent advancements in requirement elicitation and prioritization techniques, 2015 International Conference on Advances in Computer Engineering and Applications (ICACEA), 2015, pp. 237-240, DOI: 10.1109/icacea.2015.7164702.
[92] Davis, A., Dieste, O., Hickey, A., Juristo, N., and Moreno, A.M., Effectiveness of requirements elicitation techniques: empirical results derived from a systematic review, International requirements engineering conference, 2006.
[93] Maiden, N. and Rugg, G., ACRE: selecting methods fro requirements acquisition, software engineering j, 11 (3), 1996, pp. 183-192.
[94] Moody, J.W., Blanton, J.E., and Cheney, P.H., A theoretically grounded approach to assist memory recall during information requirements determination, Journal of management information systems, 15 (1), 1998, pp. 79-98, DOI: http://www.jstor.org/stable/40398373
[95] Browne, G.J. and Rogich, M.B., An empirical investigation of user requirements elicitation: comparing the effectiveness of prompting techniques, J Manag Inf Syst, 17, 2001, pp. 223-249.
[96] Browne, G.J. and Ramesh, V., Improving information requirements determination: a cognitive perspective, inf management, 21, 2002, pp. 279-304.
[97] LaFrance, M., The knowledge acquisition grid: a method for training knowledge engineers, Internationa journal of man-machine studies, 26 (2), 1987, pp. 245-255, DOI: 10.1016/S0020-7373(87)80094-9.

[98] Farinha, C. and da Silva, M., Requirements Elicitation with Web-Based Focus Groups, Information Systems Development, ed. Pooley, R., J. Coady, C. Schneider, H. Linger, C. Barry, and M. Lang, Springer New York, 2013, pp. 443-455, ISBN: 978-1-4614-4950-8.
[99] daSilva, T.S., Selbach Silveira, M., Maurer, F., and Hellmann, T., User experience design and agile development: from theory to practice, Journal of software engineering and applications, 5 (x), 2012, pp. 743-751.
[100] Gayatri, V. and Pammi, K., *Agile User Stories: The Building Blocks for Software Project Development Success*. Scrum Alliance, 2013.
[101] Paasivaara, M., Heikkila, V.T., and Lassenius, C., Experiences in scaling the product owner role in large-scale globally distributed Scrum, International conference on global software engineering, IEEE, Porto Alegre, 2012, pp. 174-178, DOI: 10.1109/ICGSE.2012.41
[102] Mahnič, V. and Hovelja, T., On using planning poker for estimating user stories, Journal of systems and software, 85 (9), 2012, pp. 2086-2095, DOI: 10.1016/j.jss.2012.04.005.
[103] Davies, I., Green, P., Rosemann, M., Indulska, M., and Gallo, S., How do practitioners use conceptual modeling in practice?, Data & knowledge engineering, 58 (3), 2006, pp. 358-380, DOI: 10.1016/j.datak.2005.07.007.
[104] DaSilva, C.M. and Trkman, P., Business model: what it is and what it is not, Long range planning, 47 (6), 2014, pp. 379-389, DOI: 10.1016/j.lrp.2013.08.004.
[105] Loucopoulos, P. and Kavakli, E., Enterprise modelling and the teleological approach to requirements engineering, International Journal of Cooperative Information Systems, 4 (1), 1995, pp. 45-79, DOI: 10.1142/S0218843095000032
[106] Dumas, M., La Rosa, M., Mendling, J., and Reijers, H., Fundamentals of business process management, Springer-Verlag Berlin Heidelberg, Berlin, Germany, 2013, p. 399, ISBN: 978-3-642-33142-8.
[107] Indulska, M., Green, P., Recker, J., and Rosemann, M., Business process modeling: perceived benefits, Conceptual modeling - ER 2009, ed. Castano, S., U. Dayal, and A.H.F. Laender, Springer, Gramado, Brazil, 2009, pp. 458-471.
[108] Wang, H.J., Zhao, J.L., and Zhang, L.-J., Policy-driven process mapping (pdpm): discovering process models from business policies, Decision support systems, 48 (1), 2009, pp. 267-281.
[109] Dijkman, R., Dumas, M., van Dongen, B., Käärik, R., and Mendling, J., Similarity of business process models: metrics and evaluation, Information systems, 36 (2), 2011, pp. 498-516, DOI: http://dx.doi.org/10.1016/j.is.2010.09.006.
[110] Kovačič, A. and Bosilj Vukšič, V., Management poslovnih procesov: Prenova in informatizacija poslovanja, GV Založba, Ljubljana, 2005, p. 487.
[111] Dreiling, A., Rosemann, M., van der Aalst, W.M.P., and Sadiq, S., From conceptual process models to running systems: a holistic approach for the configuration of enterprise system processes, Decision support systems, 45 (2), 2008, pp. 189-207, DOI: 10.1016/j.dss.2007.02.007.
[112] Trkman, P., Kovačič, A., and Popovič, A., SOA adoption phases: a case study, Business and information systems engineering, 3 (4), 2011, pp. 211-220, DOI: 10.1007/s12599-011-0168-2.
[113] Erl, T., Service-oriented architecture: concepts, technology, and design, Prentice Hall Professional Reference, Upple Saddle River, New Jersey, USA, 2005, ISBN: 0-13-185858-0.
[114] Ferris, C., What are web services?, Communications of the ACM, 46 (6), 2003, pp. 31-32.

[115] Jurič, M. and Pant, K., Business Process Driven SOA using BPMN and BPEL: From Business Process Modeling to Orchestration and Service Oriented Architecture, Packt Publising, Birmingham, England, 2008, p. 311.
[116] Liu, D. and Shen, M., Business-to-business workflow interoperation based on process views, Decision support systems, 38 (3), 2004, pp. 399-419, DOI: 10.1016/S0167-9236(03)00116-7.
[117] Trkman, P., The critical success factors of business process management, International journal of information management, 30 (2), 2010, pp. 125-134, DOI: 10.1016/j.ijinfomgt.2009.07.003.
[118] Dijkman, R.M., La Rosa, M., and Reijers, H.A., Managing large collections of business process models - current techniques and challenges, Computers in industry 63 (2), 2012, pp. 91-97, DOI: 10.1016/j.compind.2011.12.003.
[119] Tam, A.S.M., Chu, L.K., and Sculli, D., Business process modelling in small- to medium-sized enterprises, Industrial management & data systems, 101 (4), 2001, pp. 144-152, DOI: 10.1108/02635570110390107.
[120] Mendling, J., Event-driven process chains (EPC), Lecture notes in business information process, Metrics for process models, ed. van der Aalst, W.M.P., J. Mylopoulos, N.M. Sadeh, M.J. Shaw, and C. Szyperski, Springer, Berlin, Germany, 2009.
[121] van der Aalst, W.M.P. and Hofstede, A.H.M., Yawl: yet another workflow language, Information systems, 30 (4), 2005, pp. 245-275.
[122] Wohed, P., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M., and Russell, N., Pattern-based analysis of the control-flow perspective of UML activity diagrams, Conceptual modeling - ER 2005, Lecture notes in computer science, Springer, Berlin, 2005, pp. 63-78, ISBN: 978-3-540-29389-7.
[123] Jurič, M. and Pant, K., Business process driven SOA using BPMN and BPEL: from business process modeling to orchestration and service oriented architecture, ed. Jennings, F., Packt Publishing, Birmingham, England, 2008, p. 328, ISBN: 9781847191465
[124] Cockburn, A., Writing effective use cases, Addison-Wesley, Upper Saddle River, NJ, 2001, p. 270, ISBN: 978-0201702255.
[125] Kruchten, P., The rational unified process: an introduction, Addison-Wesley Professional, 2004, ISBN: 0321197704.
[126] Carroll, J.M., Making use. Scenario-based design of human-cumputer interactions, MIT Press, 2000.
[127] Bagnoli, A., Beyond the standard interview: the use of graphic elicitation and arts-based methods, Qualitative Research, 9 (5), 2009, pp. 537-570.
[128] Dijkman, R.M., Dumas, M., and Ouyang, C., Semantics and analysis of business process models in BPMN, Information and software technology, 50 (12), 2008, pp. 1281-1294, DOI: http://dx.doi.org/10.1016/j.infsof.2008.02.006.
[129] Monsalve, C., April, A., and Abran, A., *On the expressivneness of business process modeling notations for software requirements elicitation*, in *Industrial electronics society (IECON)*. IEEE: Montreal, QC, Canada, 2012, p. 3132-3137.
[130] Hermsen, A.F., Brinkkemper, S., and Oei, H., Situational method engineering for information system project approaches, IFIP WG 8.1 working conference, ed. Verrijn Stuart, A.A. and T.W. Olle, IFIP Transactions A-55, Maastricht, Netherlands, 1994, pp. 169-194, ISBN: 0-444-82074-4.
[131] Shanks, G., Tansley, E., and Weber, R., Using Ontology To Validate Conceptual Models, Communications of the ACM - Service-oriented computing, 46 (10), 2003, pp. 85-89, DOI: 10.1145/944217.944244

[132] Wand, Y., Ontology as a foundation for meta-modelling and method engineering, Information and software technology, 38 (4), 1996, pp. 281-287, DOI: 10.1016/0950-5849(95)01052-1.
[133] Lohmann, N., Verbeek, E., and Dijkman, R., Petri Net transformations for business processes – a survey, Transactions on Petri Nets and other models of concurrency II, Lecture notes in computer science, ed. Jensen, K. and W. van der Aalst, Springer Berlin Heidelberg, Berlin, Germany, 2009, pp. 46-63, ISBN: 978-3-642-00898-6.
[134] zur Muehlen, M. and Recker, J., How much language is enough? Theoretical and practical use of the business process modeling notation, Advance information systems engineering, Lecture notes in computer science, ed. Bellahsène, Z. and M. Léonard, Springer Berlin Heidelberg, Berlin, Germany, 2008, pp. 456-479, ISBN: 978-3-540-69533-2.
[135] Recker, J.C., Indulska, M., Rosemann, M., and Green, P., Do process modelling techniques get better? A comparative ontological analysis of BPMN, Australasian Conference on Information Systems, QUT ePrints, Sydney, Australia, 2013.
[136] Leopold, H., Mendling, J., and Polyvyanyy, A., Supporting process model validation through natural language generation IEEE transactions on software engineering, 40 (8), 2013, pp. 818-840, DOI: 10.1109/TSE.2014.2327044
[137] Mcroy, S.W., Channarukul, S., and Ali, S.S., *Text realization for dialog*, in *International conference on intelligent technologies*. 2000.
[138] Lavoie, B. and Rambow, O., *A fast and portable realizer for text generation systems*, in *Applied natural language processing*. 1997, p. 265-268.
[139] Mendling, J., Reijers, H.A., and van der Aalst, W.M.P., Seven process modeling guidelines, Information and software technology, 52 (2), 2010, pp. 127-137, DOI: 10.1016/j.infsof.2009.08.004.
[140] Leopold, H., Mendling, J., and Gunther, O., What we can learn from quality issues of BPMN models from industry IEEE software, PP (99), 2015, pp. 1-9, DOI: 10.1109/MS.2015.81
[141] Wand, Y. and Weber, R., On the ontological expressiveness of information systems analysis and design grammars, Information systems journal, 3 (4), 1993, pp. 217-237, DOI: 10.1111/j.1365-2575.1993.tb00127.x.
[142] Larkin, J.H. and Simon, H.A., Why a diagram is (sometimes) worth ten thousand words Cognitive science, 11 (1), 1987, pp. 65-100, DOI: 10.1016/S0364-0213(87)80026-5.
[143] Maes, A. and Poels, G., Evaluating quality of conceptual modelling scripts based on user perceptions, Data & knowledge engineering, 63 (2007), 2007, pp. 701-724, DOI: 10.1016/j.datak.2007.04.008.
[144] Batra, D., Hoffler, J.A., and Bostrom, R.P., Comparing representations with relational and EER models, Communications of the ACM, 33 (2), 1990, pp. 126-139, DOI: 10.1145/75577.75579
[145] Wohlin, C., Runeson, P., Host, M., Ohlsson, M.C., Regnell, B., and Wesslen, A., Experimentation in software engineering, an introduction. The Kluwer international series in software engineering, ed. Basili, V.R., Springer, Boston, 2000, ISBN: 978-3-642-29043-5.
[146] Field, A., Discovering statistics using SPSS, ed. Carmichael, M., SAGE publications, Canada, 2009, ISBN: 987-1-4462-4917-8.
[147] Gomez, A., Rueda, G., and Alarcón , P.P., A systematic and lightweight method to identify dependencies between user stories, Agile processes in software engineering and extreme programming, Lecture notes in business information processing Springer Berlin, 2010, pp. 190-195, ISBN: 978-3-642-13053-3.

EXPERIMENT:
MEASURING DOMAIN UNDERSTANDING OF A USER STORY

Document: Workbook 1 – List vs. BP – C2P2 and C1P1 _v12

Authors:
Ph.D. student Marina Trkman (marina.trkman@fri.uni-lj.si)
prof. dr. Jan Mendling (jan.mendling@wu.ac.at)

INTRODUCTION TO EXPERIMENT'S WORKBOOK 1

Dear Participant,
This is a research study on user stories, which are a way that users present their requirements for the new software. Development experts have reported that understanding user stories can be difficult in practice. In this experiment, we aim to investigate user story understand in different configurations

If you have never seen user stories before, do not worry about it. Participation involves completing two cases which will take not more than 60 minutes to complete in total.

Confidentiality
We pay very high attention to treating all data, comments and responses from this experiment confidentially and strictly anonymous. If you would like to obtain additional information you may contact us at marina.trkman@fri.uni-lj.si.

Please consider the following while answering the survey questions:

- Answer all questions to the best of your knowledge.
- Only use the material that is presented for a question.
- Do not guess if the material does not provide the answer.
- Work from the start of the questionnaire to the end. **Do not flip back the page!**

Workbook's structure:

- Part 1: Demographical questions and questions about the familiarity with experiment's domain and models used.
- Part 2: Case 1
  - Read the models provided: List of user stories, and a corresponding business process model.
  - Answer comprehension questions about the models while looking at the models.
  - Answering problem solving questions while looking at the models.
  - Fill-in-the blank gaps without seeing the models.
  - Questions measuring perceived ease of use
- Part 3: Case 2
  - Read the model provided: List of user stories.
  - Answer comprehension questions about the list while looking at the list.
  - Answering problem solving questions about the list while looking at the list.
  - Fill-in-the blank gaps without seeing the list.
  - Questions measuring perceived ease of use
- Ending

PART 1: Background

Personal background

1. Gender: O Male O Female

2. Nationality: _____

3. What is your age? _____ years

4. What is your university? O WU O Univ. v Ljubljani

5. How good is your English?
   Poor 1 O 2 O 3 O 4 O 5 O 6 O 7 O Excellent

6. What is your preferred learning style? Write down number from 1 (most preferred) to 3 (least preferred).
   __ auditory (hearing)
   __ visual (seeing)
   __ kinesthetic (doing it)

Familiarity with user stories

7. Estimate the level of your prior familiarity with user story models.

	1	2	3	4	5	6	7	
Not at all familiar	O	O	O	O	O	O	O	Very familiar

Familiarity with business process models

8. Estimate the level of familiarity with business process model theory.

	1	2	3	4	5	6	7	
Not at all familiar	O	O	O	O	O	O	O	Very familiar

9. Estimate the level of familiarity with BPMN notation.

	1	2	3	4	5	6	7	
Not at all familiar	O	O	O	O	O	O	O	Very familiar

Familiarity with bank's domain

10. Estimate the level of familiarity about activities performed by an employee in the bank if a customer wants a new credit card (a MasterCard).

	1	2	3	4	5	6	7	
Not at all familiar	O	O	O	O	O	O	O	Very familiar

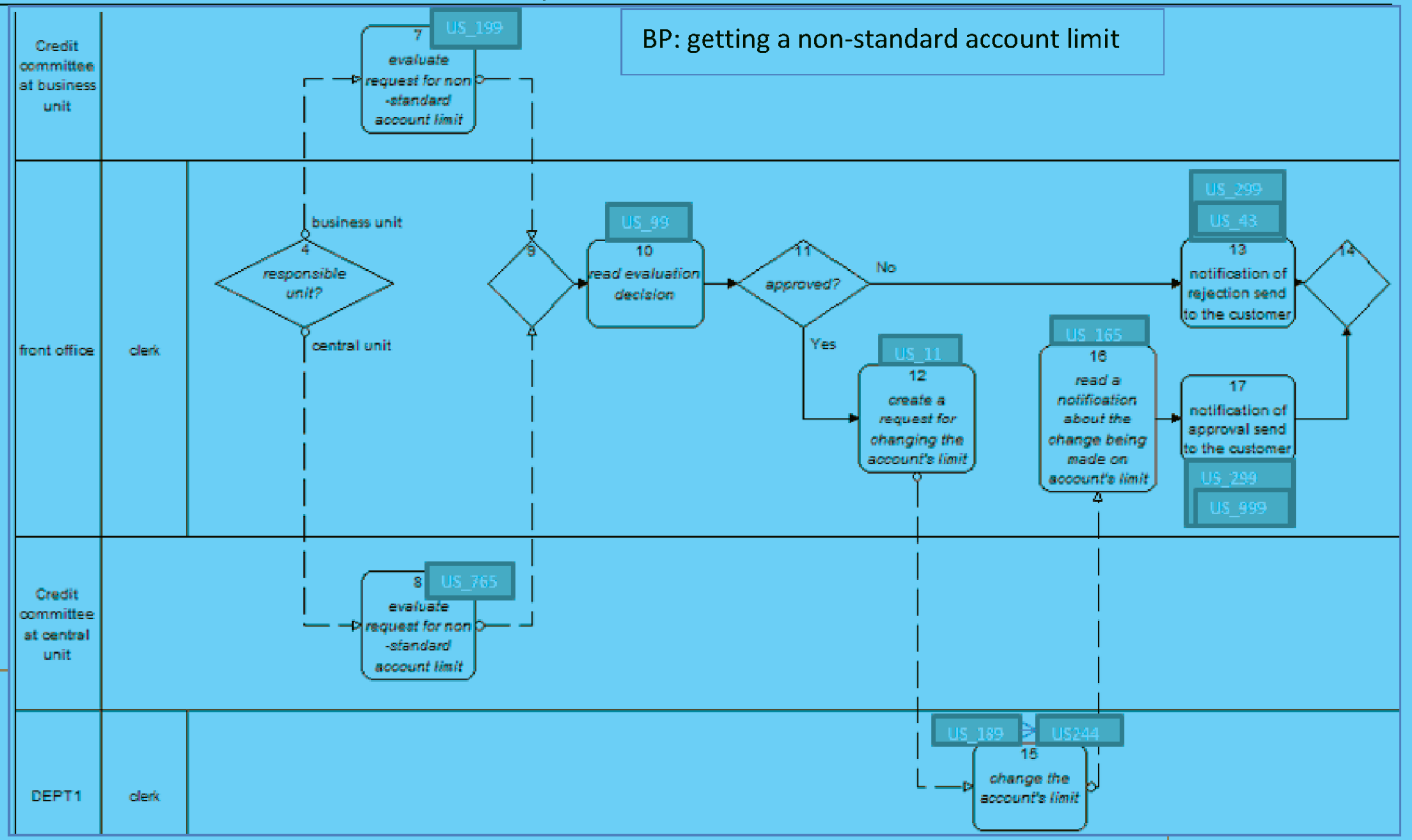
11. Estimate the level of familiarity about activities performed by an employee in the bank if a customer requests a non standard account limit.

	1	2	3	4	5	6	7	
Not at all familiar	O	O	O	O	O	O	O	Very familiar

Appendix A:
Workbook 1 from the Experiment 1

Material available for solving

the questions of Case 1



List of
user stories
for Case 1:

US_11 I as a clerk in the front office can "create a request to modify the account".
US_165 I as a clerk in the front office can "read about the account's limit".
US_189 I as a clerk in DEPT1 can "modify the account's attribute".
US_199 I as a credit committee (member) at central unit can "make an assessment report for non-standard limit".
US_244 I as a clerk in DEPT1 can "send a notification about the change of account's attribute".
US_299 I as a clerk in the front office can "notify the customer about credit committee's decision".
US_43 I as a clerk in the front office can "give information about the rejection".
US_765 I as a credit committee (member) at business unit can "evaluate non-standard account limit".
US_300 I as a clerk in the front office can "notify the customer".
US_99 I as a clerk in the front office can "read about the decision".
US_999 I as a clerk in the front office can "send the letter about the approval".
US_100 I as a credit committee (member) can "evaluate the proposal".

Part 2: Case 1 → comprehension questions about the model
Please make sure you see pages 153 and 154 at the same time.

Comprehension questions
Think about dependencies between user stories when answering the following comprehension questions.

12. Is the execution of US_11 optional?

O Yes O No O Do not know

13. Is the execution of US_299 optional?

O Yes O No O Do not know

14. Is the execution of US_43 optional?

O Yes O No O Do not know

15. Which user stories have a potential to be executed when the bank's customer wants to get a non-standard account limit (please tick)?

☐ US_11	☐ US_199	☐ US_43	☐ US_99
☐ US_165	☐ US_244	☐ US_765	☐ US_999
☐ US_189	☐ US_299	☐ US_300	☐ US_100

16. Which user roles need to collaborate when the bank's customer wants to get a non-standard account limit (tick)?

☐ Credit committee at business unit	☐ Clerk in the front office
☐ Credit committee at central unit	☐ Clerk in DEPT1

17. What are the two notification types that are created by the clerk in the front office in order to inform the customer about the change being made on the account limit attribute?

_____

18. Which user role executes the user story US_199 ?

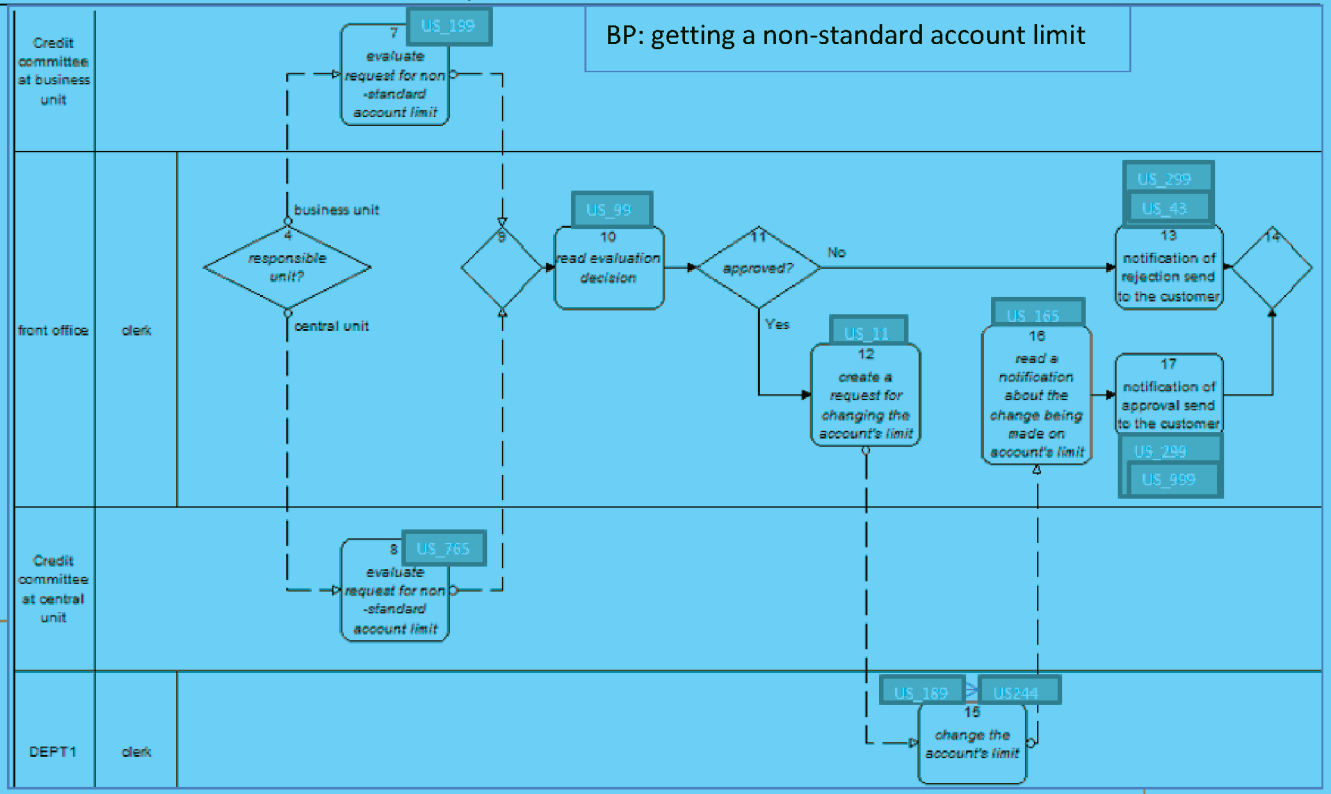
O Credit committee at business unit	O Clerk in the front office
O Credit committee at central unit	O Clerk in DEPT1

19. Who makes the change of the account's limit?

O Credit committee at business unit	O Clerk in the front office
O Credit committee at central unit	O Clerk in DEPT1

Material available for solving

the questions of Case 1



List of
user stories
for Case 1:

US_11 I as a clerk in the front office can "create a request to modify the account".
US_165 I as a clerk in the front office can "read about the account's limit".
US_189 I as a clerk in DEPT1 can "modify the account's attribute".
US_199 I as a credit committee (member) at central unit can "make an assessment report for non-standard limit".
US_244 I as a clerk in DEPT1 can "send a notification about the change of account's attribute".
US_299 I as a clerk in the front office can "notify the customer about credit committee's decision".
US_43 I as a clerk in the front office can "give information about the rejection".
US_765 I as a credit committee (member) at business unit can "evaluate non-standard account limit".
US_300 I as a clerk in the front office can "notify the customer".
US_99 I as a clerk in the front office can "read about the decision".
US_999 I as a clerk in the front office can "send the letter about the approval".
US_100 I as a credit committee (member) can "evaluate the proposal".

Part 2: Case 1 ->the problem solving questions

Please make sure you see pages 155 and 156 at the same time.

20. About US_244

a) Fill in the gaps with the user story codes so we could see all the sequences of how the execution of US_244 can be approached.

Sequence 1: US_ ___-> US_ ___ -> US_ ___ -> US_ ___ -> US_244

Sequence 2: US_ ___-> US_ ___ -> US_ ___ -> US_ ___ -> US_244

b) Between which two user stories in the sequences of a) does a change of user roles happen?

- ○ I do not know if there are any pairs.
- ○ I am certain that there are no pairs.
- ○ There are the following pairs:
  US_ __ & US_ __; _____
  _____

c) Are there any decision points in the sequences of a) which influence the variability of execution sequences?

- O I do not know.
- O No, I do not believe there are any decision points.
- O Yes.

If Yes: There was a mistake done at a decision point so the incorrect following flow (the alternative flow) of user stories was selected. The US_244 is desperately waiting for (correct) input information. What could have gone wrong at previous decision point(s) (please tick)?

- ☐ Maybe the evaluation was made by wrong credit committee.
- ☐ Maybe the evaluation was not approved but treated as approved.
- ☐ Maybe the evaluation was approved but treated as not approved.

21. About US_11

a) Fill in the gaps with the user story codes so we could see all the sequences of how the execution of US_11 can be approached.

Sequence 1: US_ ___ -> US___ -> US_ 11

Sequence 2: US_ ___ -> US___ -> US_11

b) Between which two user stories in the sequences of a) does a change of user roles happen?

- ○ I do not know if there are any pairs.
- ○ I am certain that there are no pairs.
- ○ There are the following pairs:
  US_ __ & US_ __; _____
  _____

c) Are there any decision points in the sequences of a) which influence the variability of execution sequences?

- O I do not know.
- O No, I do not believe there are any decision points.
- O Yes.

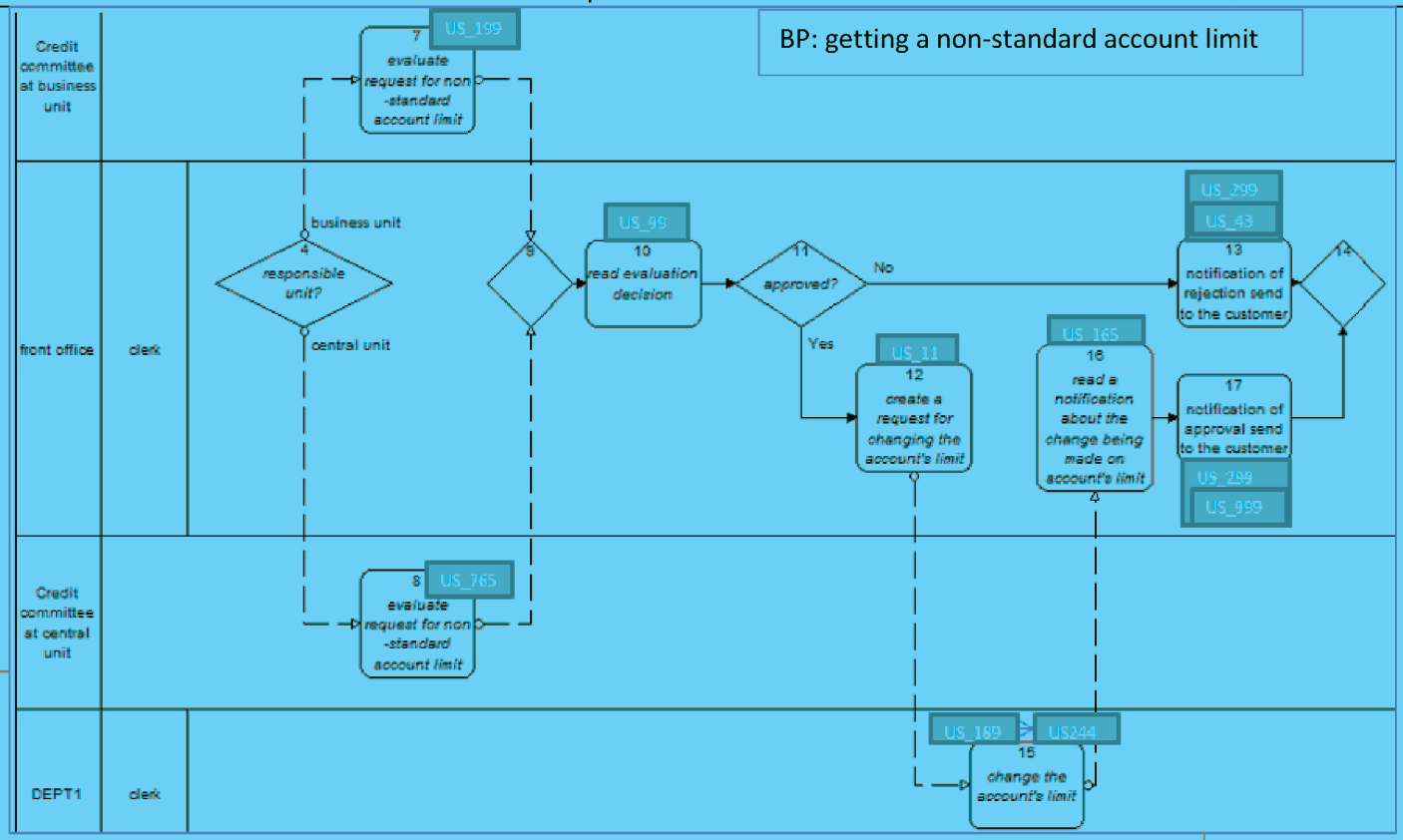
If Yes: There was a mistake done at a decision point so the incorrect following flow (the alternative flow) of user stories was selected. The US_11 is desperately waiting for (correct) input information. What could have gone wrong at previous decision point(s) (please tick)?

- ☐ Maybe the evaluation was made by wrong credit committee.
- ☐ Maybe the evaluation was not approved but treated as approved.
- ☐ Maybe the evaluation was approved but treated as not approved.

Appendix A:
Workbook 1 from the Experiment 1

Material available for solving

the questions of Case 1



List of
user stories
for Case 1:

US_11 I as a clerk in the front office can "create a request to modify the account".
US_165 I as a clerk in the front office can "read about the account's limit".
US_189 I as a clerk in DEPT1 can "modify the account's attribute".
US_199 I as a credit committee (member) at central unit can "make an assessment report for non-standard limit".
US_244 I as a clerk in DEPT1 can "send a notification about the change of account's attribute".
US_299 I as a clerk in the front office can "notify the customer about credit committee's decision".
US_43 I as a clerk in the front office can "give information about the rejection".
US_765 I as a credit committee (member) at business unit can "evaluate non-standard account limit".
US_300 I as a clerk in the front office can "notify the customer".
US_99 I as a clerk in the front office can "read about the decision".
US_999 I as a clerk in the front office can "send the letter about the approval".
US_100 I as a credit committee (member) can "evaluate the proposal".

Part 2: Case 1 ->the problem solving questions

Please make sure you see pages 157 and 158 at the same time.

22. About US_299

a) Fill in the gaps with the user story codes so we could see all the variations of how the execution of US_299 can be approached.

Sequence 1: US_ ___-> US_ ___ -> US_ ___ -> US_ ___ -> US_ ___ ->
->US_ ___ -> US_299

Sequence 2: US_ ___-> US_ ___ -> US_ ___ -> US_ ___ -> US_ ___ ->
->US_ ___ -> US_299

Sequence 3: US_ ___-> US_ ___ -> US_299

Sequence 4: US_ ___-> US_ ___ -> US_299

b) Between which two user stories in the sequences of a) does a change of user roles happen?

- ○ I do not know if there are any pairs.
- ○ I am certain that there are no pairs.
- ○ There are the following pairs:
  US_ __ & US_ __; _____
  _____

c) Are there any decision points in the sequences of a) which influence the variability of execution sequences?

- O I do not know.
- O No, I do not believe there are any decision points.
- O Yes.

If Yes: There was a mistake done at a decision point so the incorrect following flow (the alternative flow) of user stories was selected. The US_299 is desperately waiting for (correct) input information. What could have gone wrong at previous decision point(s) (please tick)?

- ☐ Maybe the evaluation was made by wrong credit committee.
- ☐ Maybe the evaluation was not approved but treated as approved.
- ☐ Maybe the evaluation was approved but treated as not approved.

23. Sometimes the activity "notify the customer about credit committee's decision (activity from US_299)" can be executed in different variations. Which user stories (codes) present those variations?
    - ○ There are no user stories presenting the wanted variations.
    - ○ The following user stories are presenting the variations:

      _____

24. Sometimes the activity "create a request to modify the account (activity from US_11)" can be executed in different variations. Which user stories (codes) present those variations?
    - ○ There are no user stories presenting the wanted variations.
    - ○ The following user stories are presenting the variations:

      _____

25. Sometimes the activity "send a notification about the change of account's attribute (activity from US_244)" can be executed in different variations. Which user stories (codes) present those variations?
    - ○ There are no user stories presenting the wanted variations.
    - ○ The following user stories are presenting the variations:

      _____

Part 3: Case 1 → post test

Introduction to the post test

On the right hand side of this page you see a text with some important words removed. Fill in the blanks using the knowledge you have gained from the model provided earlier. Each blank should be filled with only ONE word

Do not flip back the page! Fill in the blank gaps by using your memory.

26. Fill-in-the blank gaps

A credit committee needs to evaluate a request for a non-standard account _____. There are two kinds of credit committee. One kind of credit committee is formed at _____ unit and another is formed locally at _____ units. Which credit committee will be dealing with the customer's request depends on specific credit committee's responsibilities. After the 'evaluation _____' is made, it is forwarded to clerk in the front office. If it is positive, then the clerk in the front office fills in a _____ form for changing the 'account _____'. When the change is executed, the clerk in the front office informs the _____ about it by writing a notification of _____.

Perceived ease of use and usefulness of Case 1 material

27. It was easy for me to understand what the material was trying to show.

	1	2	3	4	5	6	7	
Strongly Disagree	O	O	O	O	O	O	O	Strongly Agree

28. Using the material was often frustrating.

	1	2	3	4	5	6	7	
Strongly Disagree	O	O	O	O	O	O	O	Strongly Agree

29. Overall, the material was easy to use.

	1	2	3	4	5	6	7	
Strongly Disagree	O	O	O	O	O	O	O	Strongly Agree

30. Learning how to read the material was easy.

	1	2	3	4	5	6	7	
Strongly Disagree	O	O	O	O	O	O	O	Strongly Agree

You have finished Case 1. The following Case 2 has similar set of questions but no business process model available.

Material available for solving the questions of Case 2

List of user stories for Case 2:

US_100	I as a credit committee (member) can “evaluate the proposal”.
US_132	I as a credit committee (member) can "assess the eligibility for a non-standard limit".
US_155	I as a clerk in the front office can “list all the customers who are eligible to have a new MasterCard but they do not have one”.
US_765	I as a credit committee (member) at business unit can "evaluate non-standard account limit".
US_165	I as a clerk in the front office can "read about the account’s limit".
US_231	I as a clerk in the front office can “create a promotion letter”.
US_260	I as a credit committee (member) can "rate individual's financial position if its appropriate for a standard card limit".
US_311	I as a credit committee (member) can "analyze the corporate customer’s suitability for having a credit card".
US_67	I as a clerk in the front office can “fill in a proposal”.

Part 3: Case 2 → comprehension questions about the model
Please make sure see pages 161 and 162 at the same time.

Comprehension questions
Think about dependencies between user stories when answering the following comprehension questions.

31. Is the execution of US_231 optional?

O Yes O No O Do not know

32. Is the execution of US_100 optional?

O Yes O No O Do not know

33. Is the execution of US_132 optional?

O Yes O No O Do not know

34. Which user stories have a potential to be executed when the bank's customer wants to get a new MasterCard?

☐ US_100 ☐ US_763 ☐ US_260
☐ US_132 ☐ US_165 ☐ US_311
☐ US_155 ☐ US_231 ☐ US_67

35. Which user roles need to collaborate when the bank's customer wants to get a new MasterCard?

☐ Credit committee at business unit ☐ Clerk in the front office
☐ Credit committee at central unit ☐ Credit committee

36. What are the two 'customer types'?

_____

37. Which user role executes the user story US_231?

O Credit committee at business unit O Clerk in the front office
O Credit committee at central unit O Credit committee

38. What are the two 'card limit types' that the customer can ask for?

O individual and corporate
O accepted and rejected
O modified and non-modified
O standard and non-standard

Material available for solving the questions of Case 2

List of user stories for Case 2:

US_100	I as a credit committee (member) can “evaluate the proposal”.
US_132	I as a credit committee (member) can "assess the eligibility for a non-standard limit".
US_155	I as a clerk in the front office can “list all the customers who are eligible to have a new MasterCard but they do not have one”.
US_765	I as a credit committee (member) at business unit can "evaluate non-standard account limit".
US_165	I as a clerk in the front office can "read about the account’s limit".
US_231	I as a clerk in the front office can “create a promotion letter”.
US_260	I as a credit committee (member) can "rate individual's financial position if its appropriate for a standard card limit".
US_311	I as a credit committee (member) can "analyze the corporate customer’s suitability for having a credit card".
US_67	I as a clerk in the front office can “fill in a proposal”.

Part 3: Case 2 ->the problem solving questions

Here you need to see pages 163 and 164 at the same time.

39. About US_231

a) Fill in the gaps with the user story codes so we could see all the variations of how the execution of US_231 can be approached.

Sequence 1: US_ ___ -> US_231

b) Between which two user stories in the sequences of a) does a change of user roles happen?

- ○ I do not know if there are any pairs.
- ○ I am certain that there are no pairs.
- ○ There are the following pairs:
  US_ __ & US_ __; _____
  _____

c) Are there any decision points in the sequences of a) which influence the variability of execution sequences?

- O I do not know.
- O No, I do not believe there are any decision points.
- O Yes.

If Yes: There was a mistake done at a decision point so the incorrect following flow (the alternative flow) of user stories was selected. The US_231 is desperately waiting for (correct) input information. What could have gone wrong at previous decision point(s) (please tick)?

- ☐ Somebody send a promotion letter to the customer when he/she shouldn't have done it.
- ☐ The evaluation for a new MasterCard was tagged with a wrong customer type.
- ☐ The evaluation for a new MasterCard was tagged with a wrong card limit type.

40. About US_311

a) Fill in the gaps with the user story codes so we could see all the variations of how the execution of US_311 can be approached.

Sequence 1: US_ ___ -> US_ ___ -> US_ ___ -> US_311

Sequence 2: US_ ___ -> US_311

b) Between which two user stories in the sequences of a) does a change of user roles happen?

- ○ I do not know if there are any pairs.
- ○ I am certain that there are no pairs.
- ○ There are the following pairs:
  US_ __ & US_ __; _____
  _____

c) Are there any decision points in the sequences of a) which influence the variability of execution sequences?

- O I do not know.
- O No, I do not believe there are any decision points.
- O Yes.

If Yes: There was a mistake done at a decision point so the incorrect following flow (the alternative flow) of user stories was selected. The US_311 is desperately waiting for (correct) input information. What could have gone wrong at previous decision point(s) (please tick)?

- ☐ Somebody send a promotion letter to the customer when he/she shouldn't have done it.
- ☐ The evaluation for a new MasterCard was tagged with a wrong customer type
- ☐ The evaluation for a new MasterCard was tagged with a wrong card limit type

Material available for solving the questions of Case 2

List of user stories for Case 2:

US_100 I as a credit committee (member) can “evaluate the proposal”.
US_132 I as a credit committee (member) can "assess the eligibility for a non-standard limit".
US_155 I as a clerk in the front office can “list all the customers who are eligible to have a new MasterCard but they do not have one”.
US_765 I as a credit committee (member) at business unit can "evaluate non-standard account limit".
US_165 I as a clerk in the front office can "read about the account’s limit".
US_231 I as a clerk in the front office can “create a promotion letter”.
US_260 I as a credit committee (member) can "rate individual's financial position if its appropriate for a standard card limit".
US_311 I as a credit committee (member) can "analyze the corporate customer’s suitability for having a credit card".
US_67 I as a clerk in the front office can “fill in a proposal”.

Part 3: Case 2 ->the problem solving questions

Here you need to see pages 165 and 166 at the same time.

41. About US_100

a) Fill in the gaps with the user story codes so we could see all the variations of how the execution of US_100 can be approached.

Sequence 1: US_ ___ -> US_ ___ -> US_ ___ -> US_100

Sequence 2: US_ ___ -> US_100

b) Between which two user stories in the sequences of a) does a change of user roles happen?

- ○ I do not know if there are any pairs.
- ○ I am certain that there are no pairs.
- ○ There are the following pairs:
  US_ __ & US_ __; _____
  _____

c) Are there any decision points in the sequences of a) which influence the variability of execution sequences?

- O I do not know.
- O No, I do not believe there are any decision points.
- O Yes.

If Yes: There was a mistake done at a decision point so the incorrect following flow (the alternative flow) of user stories was selected. The US_100 is desperately waiting for (correct) input information. What could have gone wrong at previous decision point(s) (please tick)?

- ☐ Somebody send a promotion letter to the customer when he/she shouldn't have done it.
- ☐ The evaluation for a new MasterCard was tagged with a wrong customer type
- ☐ The evaluation for a new MasterCard was tagged with a wrong card limit type

42. Sometimes the activity "evaluate the proposal (activity from US_100)" can to be executed in different variations. Which user stories (codes) present those variations?
    - ○ There are no user stories presenting the wanted variations.
    - ○ The following user stories are presenting the variations:

      _____

43. Sometimes the activity "analyze the corporate customer's suitability for having a credit card (activity from US_311)" can to be executed in different variations. Which user stories (codes) present those variations?
    - ○ There are no user stories presenting the wanted variations.
    - ○ The following user stories are presenting the variations:

      _____

44. Sometimes the activity "create a promotion letter (activity from US_231)" can to be executed in different variations. Which user stories (codes) present those variations?
    - ○ There are no user stories presenting the wanted variations.
    - ○ The following user stories are presenting the variations:

      _____

Part 3: Case 2 → post test

Introduction to the post test

Fill in the blanks using the knowledge you have gained from the model provided earlier. Each blank should be filled with only ONE word

Fill in the blank gaps by using your memory.

45. Fill-in-the blank gaps

_____ in the front office receives a customer who wants a new _____. Sometimes it is advertised by a _____ letter. When the customer makes a request for a new card, the bank's employee fills in the form '_____ for a new MasterCard'. Next, the bank's employee needs to hand in the filled in form for an _____, which is done by _____ committee members. It is very important for them to know two things: firstly, what is the 'customer _____' and secondly, whether the customer wants a standard or a non-standard card _____.

You have finished Case 2.

Perceived ease of use and usefulness of Case 2 material

46. It was easy for me to understand what the material was trying to show.

	1	2	3	4	5	6	7	
Strongly Disagree	O	O	O	O	O	O	O	Strongly Agree

47. Using the material was often frustrating.

	1	2	3	4	5	6	7	
Strongly Disagree	O	O	O	O	O	O	O	Strongly Agree

48. Overall, the material was easy to use.

	1	2	3	4	5	6	7	
Strongly Disagree	O	O	O	O	O	O	O	Strongly Agree

49. Learning how to read the material was easy.

	1	2	3	4	5	6	7	
Strongly Disagree	O	O	O	O	O	O	O	Strongly Agree

You have finished the workbook.

Thank you.

EXPERIMENT:

ELICITING AND UNDERSTANDING USER STORIES

Document: Workbook A_v9

Authors:
Ph.D. student Marina Trkman (marina.trkman@fri.uni-lj.si)
prof. dr. Jan Mendling (jan.mendling@wu.ac.at)
prof. dr. Peter Trkman (peter.trkman@ef.uni-lj.si)

INTRODUCTION TO EXPERIMENT'S WORKBOOK A

Dear Participant,
this is a research study about creating user stories from business process models. User stories are a way in which users present their requirements for the new software. In this experiment we compare two techniques for creating user stories. Additionally, we investigate user story understanding in different configurations

If you have never seen user stories before, do not worry about it. Participation involves completing two cases which will take not more than 70 minutes to complete in total.

Confidentiality
We pay very high attention to treating all data, comments and responses from this experiment confidentially and strictly anonymous. If you would like to obtain additional information you may contact us at marina.trkman@fri.uni-lj.si

Please consider the following while answering the survey questions:

- Answer all questions to the best of your knowledge.
- Only use the material that is presented for a question.
- Do not guess if the material does not provide the answer.
- Work from the start of the questionnaire to the end. **Do not flip back the page!**

Workbook's structure:

- Part 1: Demographical questions and questions about the familiarity with experiment's domain and models used.
- Part 2: Case 1 – eliciting user stories from a visual business process model (BPMN model)
  - Read the material provided: labeled BPMN model.
  - Answer comprehension questions.
  - Create your list of user stories from the model.
  - Read the material provided: labeled BPMN model and the (given) list of user stories.
  - Answer problem-solving questions about user stories.
  - Solve the recall test.
  - Answer questions about perceived ease of usefulness of the Case 1 material.
- Part 3: Case 2 – eliciting user stories from a text-written business process model (use case model)
  - Read the material provided: labeled use case model.
  - Answer comprehension questions.
  - Create your list of user stories from the model.
  - Read the material provided: labeled use case model and the (given) list of user stories.
  - Answer problem solving questions about user stories.
  - Answer recall test.
  - Answer questions about perceived ease of usefulness of the Case 2 material.
- Ending

Background

Personal background

1. Gender: O Male O Female
2. Nationality: _____
3. Age? _____ years
4. University? O WU O Univ. of Ljubljana
5. How good is your English?

Poor 1 O 2 O 3 O 4 O 5 O 6 O 7 O Excellent

Familiarity with user stories

6. Have you ever studied about user stories?
   O Yes O No
7. Have you ever been part of user story elicitation activities?
   O Yes O No

Other

8. Do you percieve yourself to be visual or more textual type of learner?
   O Textual O Visual O Cannot specify

Familiarity with business process models

9. Estimate the level of familiarity with business process modeling theory.

	1	2	3	4	5	6	7	
Not at	O	O	O	O	O	O	O	Very
all familiar								familiar

10. Estimate the level of familiarity with BPMN models.

	1	2	3	4	5	6	7	
Not at	O	O	O	O	O	O	O	Very
all familiar								familiar

11. Estimate the level of familiarity with use case models.

	1	2	3	4	5	6	7	
Not at	O	O	O	O	O	O	O	Very
all familiar								familiar

Familiarity with bank's domain

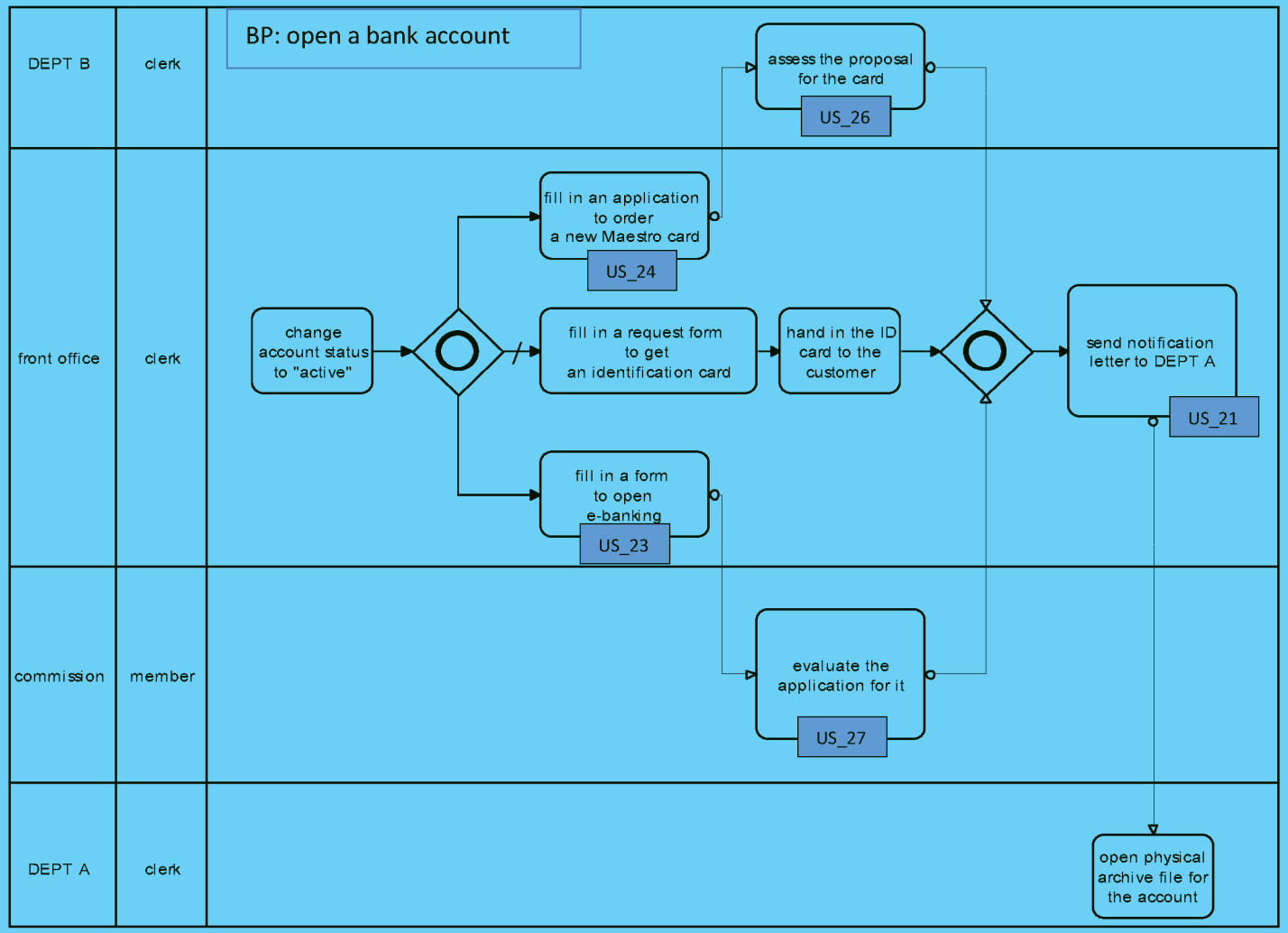
12. Estimate the level of familiarity about activities performed by an employee in bank if a customer wants to **open** a bank account.

	1	2	3	4	5	6	7	
Not at	O	O	O	O	O	O	O	Very
all familiar								familiar

13. Estimate the level of familiarity about activities performed by an employee in bank if a customer wants to **close** a bank account.

	1	2	3	4	5	6	7	
Not at	O	O	O	O	O	O	O	Very
all familiar								familiar

Case 1 material



Part 2: Case 1 → comprehension test

Please make sure you see pages with the material and with the questions at the same time.

Who participates in activities in the model (tick)?

- ☐ Clerk in DEPT 1
- ☐ Clerk in DEPT 2
- ☐ Commission member
- ☐ Clerk in DEPT B
- ☐ Clerk in the front office
- ☐ Clerk in DEPT 3
- ☐ Clerk in DEPT A
- ☐ Clerk in DEPT C

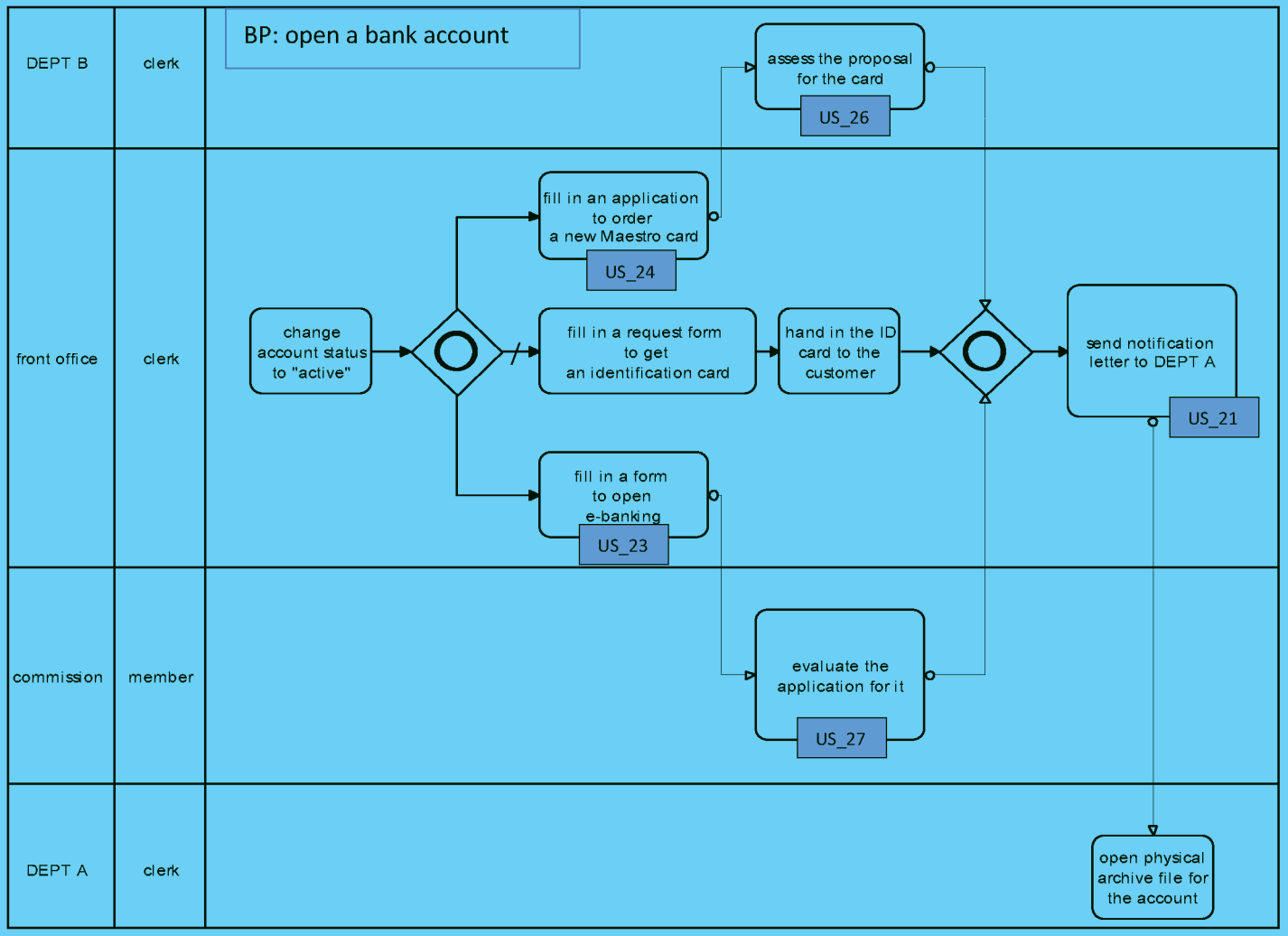
What is the sequence of activities about?

- ☐ About closing a bank account
- ☐ About getting a credit card
- ☐ About getting a debit card
- ☐ About getting a new non-standard account limit
- ☐ About getting a new non-standard credit limit
- ☐ About opening a bank account

How many user stories are present in the model?

Write down the number: _____ .

Case 1 material



Part 2: Case 1 -> elicitation test

Please make sure you see pages with the material and with the questions at the same time.

WAIT FOR INSTRUCTIONS for basic elicitation:

User story (US) template: I as a <user role> can "<activity>".

Detailed user stories:

US_21 I as a _____ can "_____".

US_23 I as a _____ can "_____".

US_24 I as a _____ can "_____".

US_26 I as a _____ can "_____".

US_27 I as a _____ can "_____".

WAIT FOR INSTRUCTIONS for advanced elicitation:

Here you need to make more general user stories out of the detailed ones. If possible, try to group at least two detailed user stories under the more general one that you define below. Remember: only one user role can be present in one user story.

More general user stories:

US_a I as a _ _____ can "_____".

US_b I as a _ _____ can "_____".

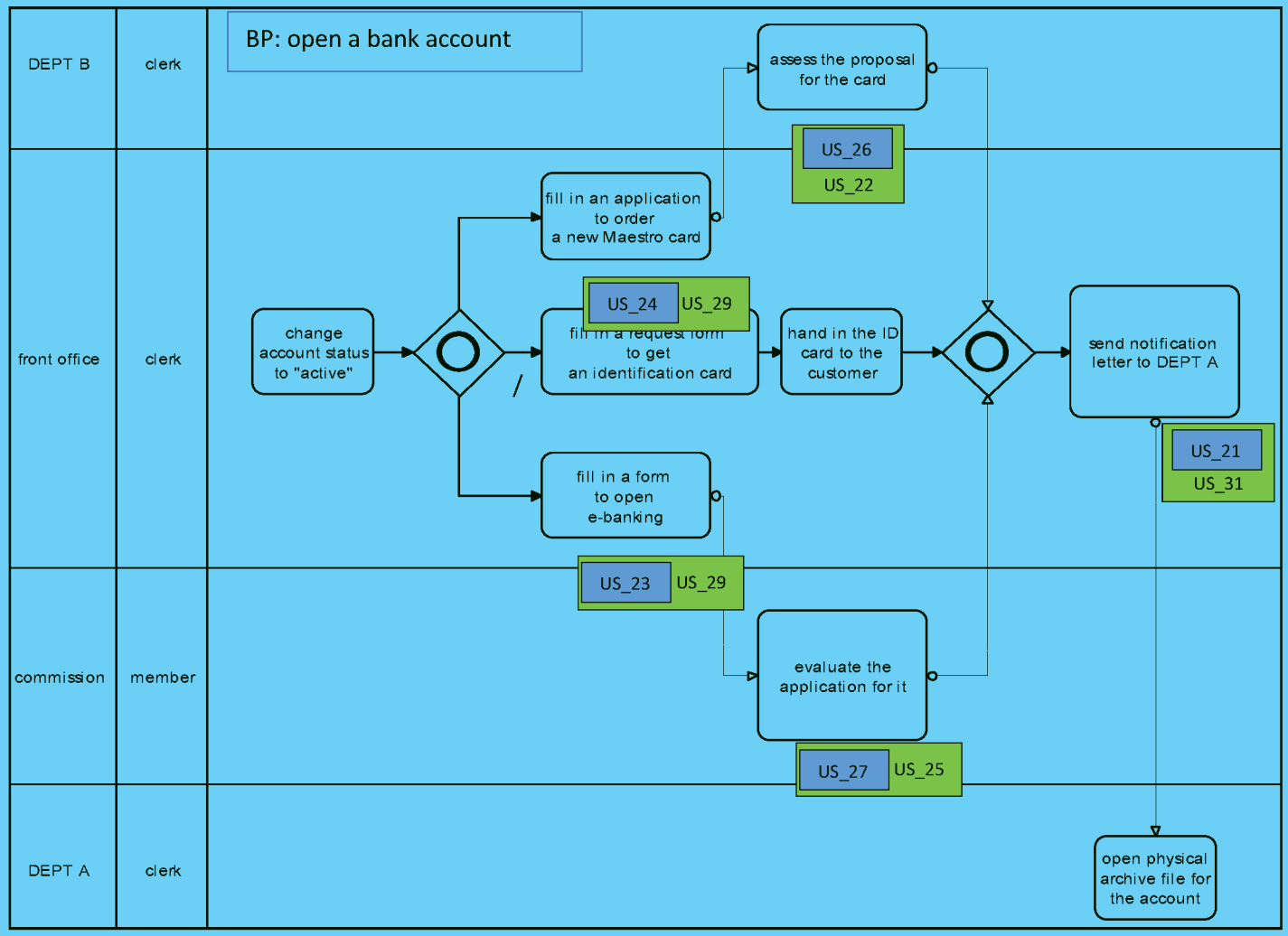
US_c I as a _ _____ can "_____".

US_d I as a _ _____ can "_____".

Case 1 material (upgraded)

List of user stories for Case 1:

US_20 I as a commission member can "evaluate an application for a MasterCard."
US_21 I as a clerk in in the front office can "send notification letter to DEPT A".
US_22 I as a clerk in DEPT B can "evaluate the application".
US_23 I as a clerk in the front office can "fill in a form to open e-banking".
US_24 I as a clerk in the front office can "fill in an application to order a new Maestro card".
US_25 I as a commission member can "evaluate the application".
US_26 I as a clerk in DEPT B can "assess the proposal for the card".
US_27 I as a commission member can "evaluate the application for it".
US_29 I as a clerk in the front office can "fill in a form".
US_31 I as a clerk in in the front office can "send internal notification letter".



Part 2: Case 1 ->problem solving test

Please make sure you see pages with the material and with the questions at the same time.

About US_21

a) How to approach to the execution of US_21. Fill in the gaps with user story codes.

Sequence 1: US_ ___ -> US_ ___ -> US_21

Sequence 2: US_ ___-> US_ ___ -> US_21

b) Between which two user stories in the sequences of a) does a change of user roles happen?

- I do not know if there are any pairs.
- I am certain that there are no pairs.
- There are the following pairs:
  US_ __ & US_ __; _____
  _____

c) Are there any decision points on the way to US_21?

- I do not know.
- No, I do not believe there are any decision points.
- Yes.

If Yes: We want to start the execution of US_21. We are waiting for (correct) input information. What could have gone wrong at previous decision point(s) (please tick)?

- ☐ Maybe a request for a new Maestro card should have been made but it was not.
- ☐ Maybe a request for opening e-banking should have been made but it was not.
- ☐ Maybe a request for new identification should have been made but it was not.

About US_22

a) How to approach to the execution of US_22. Fill in the gaps with user story codes.

Sequence 1: US_ ___ -> US_22

b) Between which two user stories in the sequences of a) does a change of user roles happen?

- I do not know if there are any pairs.
- I am certain that there are no pairs.
- There are the following pairs:
  US_ __ & US_ __; _____
  _____

c) Are there any decision points on the way to US_22?

- I do not know.
- No, I do not believe there are any decision points.
- Yes.

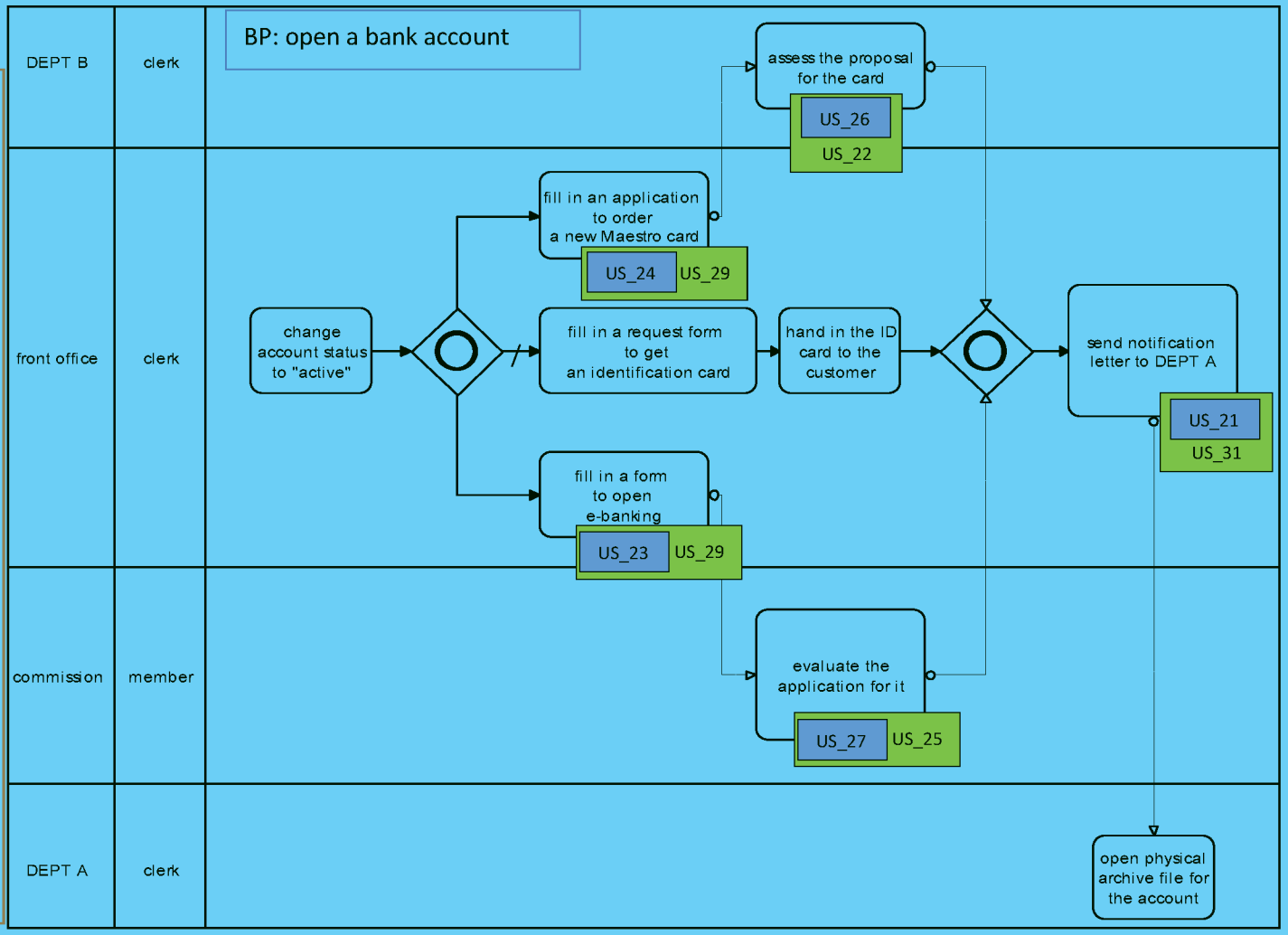
If Yes: We want to start the execution of US_22. We are waiting for (correct) input information. What could have gone wrong at previous decision point(s) (please tick)?

- ☐ Maybe a request for a new Maestro card should have been made but it was not.
- ☐ Maybe a request for opening e-banking should have been made but it was not.
- ☐ Maybe a request for new identification should have been made but it was not.

Case 1 material (upgraded)

List of user stories for Case 1:

US_20 I as a commission member can "evaluate an application for a MasterCard."
US_21 I as a clerk in in the front office can "send notification letter to DEPT A".
US_22 I as a clerk in DEPT B can "evaluate the application".
US_23 I as a clerk in the front office can "fill in a form to open e-banking".
US_24 I as a clerk in the front office can "fill in an application to order a new Maestro card".
US_25 I as a commission member can "evaluate the application".
US_26 I as a clerk in DEPT B can "assess the proposal for the card".
US_27 I as a commission member can "evaluate the application for it".
US_29 I as a clerk in the front office can "fill in a form".
US_31 I as a clerk in in the front office can "send internal notification letter".



Part 2: Case 1 –> problem-solving test

Please make sure you see pages with the material and with the questions at the same time.

Sometimes the activity "send notification letter to DEPT A (activity from US_21)" can be executed in different variations. Which user stories (codes) present those variations?

- There are no user stories presenting those variations.
- The following user stories are presenting those variations:

_____

Sometimes the activity "evaluate the application (activity from US_22)" can be executed in different variations. Which user stories (codes) present those variations?

- There are no user stories presenting those variations.
- The following user stories are presenting those variations:

_____

Part 2: Case 1 -> recall test

Do not flip back the page! Fill in the blank gaps by using your memory.

A customer wants to _____ a bank account. “Clerk in the front office” offers her additional services such are: e-banking and cards.

The customer has made decisions so “clerk in the front office” needs to prepare a/an _____ for opening e-banking. The “commission _____” will have to _____ it. Additionally, “clerk in the front office” needs to order a _____ card. “Clerk in _____” will have to give his approval about it.

Perceived ease of use and usefulness of Case 1 material

It was easy for me to understand what the material was trying to show.

	1	2	3	4	5	6	7	
Strongly Disagree	O	O	O	O	O	O	O	Strongly Agree

Using the material was often frustrating.

	1	2	3	4	5	6	7	
Strongly Disagree	O	O	O	O	O	O	O	Strongly Agree

Overall, the material was easy to use.

	1	2	3	4	5	6	7	
Strongly Disagree	O	O	O	O	O	O	O	Strongly Agree

Learning how to read the material was easy.

	1	2	3	4	5	6	7	
Strongly Disagree	O	O	O	O	O	O	O	Strongly Agree

Case 1 ends here. Case 2 starts when flipping the page.

Wait here for further instructions.

Case 2 material

Use case name: CLOSE A BANK ACCOUNT

Basic flow of events

Actor	Event	US
Clerk in the front office	Note 1 Decision #1: according to the characteristics of the account none, one (A or B) or two alternative flows (A and B) are triggered. See additional Note 2.	
Clerk in the front office	Note 3 Write a notification letter about closure of the account to the customer.	US_14

Alternative flows of events

Alt. flow	Actor	Event	US
A	Clerk in the front office	Fill in a request for cancelling authorizations	US_15
	Clerk in DEPT 2	Delete all the permissions	US_11
B	Clerk in the front office	Fill in a form to close e-banking	US_16
	Clerk in DEPT 3	Close it	US_12

Additional information:

Note 1: After the clerk in DEPT 1 changes the status of customer's account to "passive" the clerk in the front office needs to close all open businesses, such as e-banking and authorizations.

Note 2 on Decision #1: No matter how many alternative flows are triggered, clerk in the front office always writes an email about the closure to the DEPT 4 who needs to send him/her all of the account's physical documents.

Note 3: The event can start after opened authorizations are closed, opened e-banking is closed and all the account's physical documents are handed to the clerk in the front office.

Part 3: Case 2 → comprehension test

Please make sure you see pages with the material and with the questions at the same time.

Who participates in activities in the model (tick)?

- ☐ Clerk in DEPT 1
- ☐ Clerk in DEPT 2
- ☐ Commission member
- ☐ Clerk in DEPT B
- ☐ Clerk in the front office
- ☐ Clerk in DEPT 3
- ☐ Clerk in DEPT A
- ☐ Clerk in DEPT C

What is the sequence of activities about?

- ☐ About closing a bank account
- ☐ About getting a credit card
- ☐ About getting a debIt card
- ☐ About getting a new non-standard account limit
- ☐ About getting a new non-standard credit limit
- ☐ About opening a bank account

How many user stories are in the model?

Write down the number: _____ .

Case 2 material

Use case name: CLOSE A BANK ACCOUNT

Basic flow of events

Actor	Event	US
Clerk in the front office	Note 1 Decision #1: according to the characteristics of the account none, one (A or B) or two alternative flows (A and B) are triggered. See additional Note 2.	
Clerk in the front office	Note 3 Write a notification letter about closure of the account to the customer.	US_14

Alternative flows of events

Alt. flow	Actor	Event	US
A	Clerk in the front office	Fill in a request for cancelling authorizations	US_15
	Clerk in DEPT 2	Delete all the permissions	US_11
B	Clerk in the front office	Fill in a form to close e-banking	US_16
	Clerk in DEPT 3	Close it	US_12

Additional information:

Note 1: After the clerk in DEPT 1 changes the status of customer's account to "passive" the clerk in the front office needs to close all open businesses, such as e-banking and authorizations.

Note 2 on Decision #1: No matter how many alternative flows are triggered, clerk in the front office always writes an email about the closure to the DEPT 4 who needs to send him/her all of the account's physical documents.

Note 3: The event can start after opened authorizations are closed, opened e-banking is closed and all the account's physical documents are handed to the clerk in the front office.

Part 3: Case 2 -> elicitation test
Please make sure you see pages with the material and with the questions at the same time.

WAIT FOR INSTRUCTIONS for basic elicitation:

User story (US) template: I as a <user role> can "<activity>".

Detailed user stories:

US_11 I as a _____ can
"_____".

US_12 I as a _____ can
"_____".

US_14 I as a _____ can
"_____".

US_15 I as a _____ can
"_____".

US_16 I as a _____ can
"_____"

WAIT FOR INSTRUCTIONS for advanced elicitation:

Here you need to make more general user stories out of the detailed ones. If possible, try to group at least two detailed user stories under the more general one that you define below. Remember: only one user role can be present in one user story.

More general user stories:

US_a I as a _ _____ can
"_____".

US_b I as a _ _____ can
"_____".

US_c I as a _ _____ can
"_____".

US_d I as a _ _____ can
"_____".

Case 2 material (upgraded)

List of user stories for Case 2:

US_9 I as a clerk in DEPT6 can "sign the document".
US_11 I as a clerk in DEPT2 can "delete all the permissions".
US_12 I as a clerk in DEPT3 can "close it".
US_14 I as a clerk in the front office can "write a notification letter about closure of the account to the customer",
US_15 I as a clerk in the front office can "fill in a request for cancelling authorizations".
Us_16 I as a clerk in the front office can "fill in a form to close e-banking".
US_17 I as a clerk in DEPT 2 can "edit customer's account".
US_18 I as a clerk in DEPT 3 can "edit customer's account".
US_19 I as a clerk in the front office can "fill in a form".
US_20 I as a clerk in the front office can "write an informative letter to the customer".

Use case name: CLOSE A BANK ACCOUNT

Basic flow of events

Actor	Event	US	US (+1)
Clerk in the front office	Note 1 Decision #1: according to the characteristics of the account none, one (A or B) or two alternative flows (A and B) are triggered. See additional Note 2.		
Clerk in the front office	Note 3 Write a notification letter about closure of the account to the customer.	US_14	US_20

Alternative flows of events

Alt. flow	Actor	Event	US	US (+1)
A	Clerk in the front office	Fill in a request for cancelling authorizations	US_15	US_19
	Clerk in DEPT 2	Delete all the permissions	US_11	US_17
B	Clerk in the front office	Fill in a form to close e-banking	US_16	US_19
	Clerk in DEPT 3	Close it	US_12	US_18

Additional information:

Note 1: After the clerk in DEPT 1 changes the status of customer's account to "passive" the clerk in the front office needs to close all open businesses, such as e-banking and authorizations.

Note 2 on Decision #1: No matter how many alternative flows are triggered, clerk in the front office always writes an email about the closure to the DEPT 4 who needs to send him/her all of the account's physical documents.

Note 3: The event can start after opened authorizations are closed, opened e-banking is closed and all the account's physical documents are handed to the clerk in the front office.

Part 3: Case 2 ->problem solving test 1

Please make sure you see pages with the material and with the questions at the same time.

About US_17

a) How to approach to the execution of US_17. Fill in the gaps with user story codes.

Sequence 1: US_ ___ -> US_17

b) Between which two user stories in the sequences of a) does a change of user roles happen?

- I do not know if there are any pairs.
- I am certain that there are no pairs.
- There are the following pairs:
  US_ __ and US_ __; _____
  _____

c) Are there any decision points on the way to US_17?

- I do not know.
- No, I do not believe there are any decision points.
- Yes.

If Yes: We want to start the execution of US_17. We are waiting for (correct) input information. What could have gone wrong at previous decision point(s) (please tick)?

- ☐ Maybe a request for canceling authorizations should have been made but it was not.
- ☐ Maybe a request for canceling e-banking should have been made but it was not.
- ☐ Maybe the email about the closure for DEPT 4 should have been sent but it was not.

About US_14

a) How to approach to the execution of US_14. Fill in the gaps with user story codes.

Sequence 1: US_ ___ -> US___ -> US_14

Sequence 2: US_ ___ -> US___ -> US_14

b) Between which two user stories in the sequences of a) does a change of user roles happen?

- I do not know if there are any pairs.
- I am certain that there are no pairs.
- There are the following pairs:
  US_ __ & US_ __; _____
  _____

c) Are there any decision points on the way to US_14?

- I do not know.
- No, I do not believe there are any decision points.
- Yes.

If Yes: We want to start the execution of US_14. We are waiting for (correct) input information. What could have gone wrong at previous decision point(s) (please tick)?

- ☐ Maybe a request for canceling authorizations should have been made but it was not.
- ☐ Maybe a request for canceling e-banking should have been made but it was not.
- ☐ Maybe the email about the closure for DEPT 4 should have been sent but it was not.

Case 2 material (upgraded)

List of user stories for Case 2:

US_9 I as a clerk in DEPT6 can "sign the document".
US_11 I as a clerk in DEPT2 can "delete all the permissions".
US_12 I as a clerk in DEPT3 can "close it".
US_14 I as a clerk in the front office can "write a notification letter about closure of the account to the customer",
US_15 I as a clerk in the front office can "fill in a request for canceling authorizations".
Us_16 I as a clerk in the front office can "fill in a form to close e-banking".
US_17 I as a clerk in DEPT 2 can "edit customer's account".
US_18 I as a clerk in DEPT 3 can "edit customer's account".
US_19 I as a clerk in the front office can "fill in a form".
US_20 I as a clerk in the front office can "write an informative letter to the customer".

Use case name: CLOSE A BANK ACCOUNT

Basic flow of events

Actor	Event	US	US (+1)
Clerk in the front office	Note 1 Decision #1: according to the characteristics of the account none, one (A or B) or two alternative flows (A and B) are triggered. See additional Note 2.		
Clerk in the front office	Note 3 Write a notification letter about closure of the account to the customer.	US_14	US_20

Alternative flows of events

Alt. flow	Actor	Event	US	US (+1)
A	Clerk in the front office	Fill in a request for cancelling authorizations	US_15	US_19
	Clerk in DEPT 2	Delete all the permissions	US_11	US_17
B	Clerk in the front office	Fill in a form to close e-banking	US_16	US_19
	Clerk in DEPT 3	Close it	US_12	US_18

Additional information:

Note 1: After the clerk in DEPT 1 changes the status of customer's account to "passive" the clerk in the front office needs to close all open businesses, such as e-banking and authorizations.

Note 2 on Decision #1: No matter how many alternative flows are triggered, clerk in the front office always writes an email about the closure to the DEPT 4 who needs to send him/her all of the account's physical documents.

Note 3: The event can start after opened authorizations are closed, opened e-banking is closed and all the account's physical documents are handed to the clerk in the front office.

Part 3: Case 2 -> problem-solving test
Please make sure you see pages with the material and with the questions at the same time.

Sometimes the activity “write a notification letter about closure of the account to the customer (activity from US_14)” can be executed in different variations. Which user stories (codes) present those variations?

- There are no user stories presenting the wanted variations.
- The following user stories are presenting the variations:

_____

Sometimes the activity “edit customer’s account (activity from US_17)” can be executed in different variations. Which user stories (codes) present those variations?

- There are no user stories presenting the wanted variations.
- The following user stories are presenting the variations:

_____

Part 3 – Case 2 -> Recall test

Do not flip back the page! Fill in the blank gaps by using your memory.

Fill-in-the blank gaps

"Clerk in the front office" needs to _____ the account.

The clerk needs to check if the account has any _____

authorizations and if it does he/she needs to write a form about

_____ them. Next, he or she also needs to check whether

there is an e-banking service to be _____. If there is, "clerk

in the front office" notifes a "clerk in _____" about it.

When everything mentioned is done and when all account's phyical

documents are _____ to the "clerk in the front office", the

"clerk in the front office" informs the customer about the state of his/her

account.

Perceived ease of use and usefulness of Case 2 material

It was easy for me to understand what the material was trying to show.

	1	2	3	4	5	6	7	
Strongly Disagree	O	O	O	O	O	O	O	Strongly Agree

Using the material was often frustrating.

	1	2	3	4	5	6	7	
Strongly Disagree	O	O	O	O	O	O	O	Strongly Agree

Overall, the material was easy to use.

	1	2	3	4	5	6	7	
Strongly Disagree	O	O	O	O	O	O	O	Strongly Agree

Learning how to read the material was easy.

	1	2	3	4	5	6	7	
Strongly Disagree	O	O	O	O	O	O	O	Strongly Agree

You have finished the workbook. Thank you for your participation.

Metoda za uporabo modelov poslovnih procesov pri zajemu zahtev uporabniških zgodb

1 Uvod

Podjetja, ki razvijajo programsko opremo, uporabljajo pri tem veliko različnih metodologij in pristopov [2]. Ne glede na uporabljeno metodologijo je velik izziv, kako učinkovito ugotavljati in upravljati uporabniške zahteve [3]. Pri tem obstaja velika razlika med kaskadnim pristopom in novejšimi agilnimi razvojnimi pristopi. Med drugim se razlikujejo v tem, kako izvajajo analizo zahtev. Kaskadni pristop predpostavlja, da je končni rezultat analize zahtev seznam podrobnih zahtev, ki jih je potrebno določiti pred začetkom razvoja. Agilni pristopi pa upoštevajo, da ni mogoče zabeležiti vseh zahtev na začetku. Kljub temu pa poskušajo zajeti večje število uporabniških zahtev že na začetku projekta, vendar na višji ravni podrobnosti [3]. Agilne metode zajemanje zahtev dojemajo kot razvijajoč in sodelovalni proces, v katerem razvijalci programske opreme in drugi deležniki neprestano in aktivno sodelujejo v želji po boljšem razumevanju problemske domene, opredelitvi uporabniških zahtev, oceni njihove zahtevnosti in določanju prioritet [4]

Uporabniške zgodbe so najpogosteje uporabljene način zajemanja zahtev v agilnih razvojnih projektih [5-7]. Cilj uporabniških zgodb ni dokumentirati vsako podrobnost o želeni funkciji, ampak zapisati nekaj kratkih stavkov, ki bodo v pomoč tako razvijalcem kot naročniku programske rešitve pri dogovarjanju v prihodnje [3]. Uporabniške zgodbe so ključne pri komunikaciji skozi celoten proces razvoja [3] in pomagajo pri premostitvi komunikacijske vrzeli med razvijalci in naročnikom [2, 8]. Pomembno je, da razvojna ekipa razume kontekst, v katerem uporabniške zgodbe podpirajo poslovno domeno. Le tako lahko razvojna ekipa zagotovi boljše ocene potrebnih virov (npr. časa, potrebnega za razvoj neke funkcionalnosti). Zato se na začetku projekta, ki uporablja agilni pristop, osredotočimo na pridobivanje uporabniških zgodb, ki so dovolj dobre za oceno zahtevnosti projekta [4] in za razumevanje konteksta pridobljenih uporabniških zgodb v poslovanju naročnika [2, 9].

Pridobivanje oziroma zajem uporabniških zgodb zahteva intenzivno komunikacijo, ki mora premagati kulturne in pomenske razlike, ki lahko obstajajo med uporabniki in razvojno ekipo [10]. Uporabniške zgodbe so pogosto napisani s strani naročnika (denimo uporabnikov ali managerjev v podjetju, ki naroča programsko opremo), ki ustrezno pozna poslovanje podjetja. Naročnik je odgovoren za zagotavljanje čim več uporabniških zgodb čim bolj zgodaj in za

ustrezno vključenost vseh uporabnikov pri zbiranju teh zgodb [2, 3]. Izbira pristopa za pridobivanje uporabniških zgodb je odvisna od tipa projekta in razpoložljivih virov [11, 12]. Več avtorjev predlaga različne pristope (na primer [3, 13]), le redki (na primer [7]) pa se osredotočajo na to, kako in katero obstoječo dokumentacijo (torej tako, s katero stranka že razpolaga) izrabiti za pridobivanje uporabniških zgodb.

Pri zajemu uporabniških zgodb je ključno njihovo razumevanje, ki ga je treba doseči pri vseh deležnikih in to že tekom priprave na razvojni projekt. Ker pa pridobivanje uporabniških zgodb vključuje zelo različne deležnike projekta, zahteva intenzivno komunikacijo [10], ki pa lahko zelo obremeni udeležence in posledično ovira razumevanje. Komunikacijsko pregrado med naročnikom in razvojno ekipo (ki vključuje denimo programerje, testerje in vzdrževalce sistema) lahko povzroči pomanjkljivo razumevanje poslovne domene na obeh straneh [14]. Člani razvojne ekipe morajo zelo dobro razumeti poslovanje naročnika, da lahko zagotovijo njegovo ustrezno podporo z novo aplikacijo. Pri tem je sicer zelo pomembno razumevanje posameznih uporabniških zgodb naročnika, vendar pa uporabniška zgodba sama po sebi ne odraža celotnega poslovanja naročnika, ki ga je treba podpreti z novo aplikacijo, ampak samo del. S poslovnega vidika je posamezna uporabniška zgodba odvisna od drugih zgodb naročnika [15]. Zato je za uspeh projekta ključno razumevanje odvisnosti med posameznimi zgodbami [15-21]. Nerazumevanje medsebojnih odvisnosti lahko privede do slabšega razumevanje konteksta posamezne uporabniške zgodbe projekta [2, 22]. Martakis in Daneva [17] sta zato poudarila pomen integracijskih in izvedbenih odvisnosti. Integracijske odvisnosti kažejo, kako nekatere uporabniške zgodbe zahtevajo predhodni razvoj drugih uporabniških zgodb. Izvedbene odvisnosti pa kažejo, kako zaključek ene zgodbe neposredno vpliva na drugo uporabniško zgodbo. Strode [23] sicer ti dve vrsti odvisnosti imenuje aktivnostne in tehnične odvisnosti.

V splošnem so konceptualni modeli ključen artefakt za razumevanje domene aplikacij in poslovnih potreb [28]. Ker je cilj večine agilnih razvojnih projektov čim boljša podpora poslovnih potreb [29], je nujno razumeti poslovne procese naročnika, ki naj bi jih podpirala nova aplikacija [30, 31]. Proces je zaporedje (ali tok) dejavnosti v organizaciji s ciljem opravljanja dela [32]. Procese lahko predstavimo z modelu poslovnih procesov, ki se zato pogosto uporabljajo pri razvoju nove programske rešitve [33, 34]. Zato je smiselno raziskati možnost ponovne uporabe obstoječih modelov poslovnih procesov za pridobivanje uporabniških zgodb in razumevanju integracijskih in izvedbenih odvisnosti med uporabniškimi zgodbami. Te možnosti v preteklosti še niso bile ustrezno raziskane.

Raziskovalno vprašanje, s katerim se ukvarjamo v disertaciji, je: Kako lahko uporabimo obstoječe modele poslovnih procesov za razvoj programja z uporabniškimi zgodbami? Cilj disertacije je predlagati in eksperimentalno potrditi nov pristop, imenovan BuPUS (Business Process User Story) metoda, ki omogoča pridobivanje uporabniških zgodb in razumevanje izvedbenih in integracijskih odvisnosti med njimi. Predlagana metoda povezuje/integrira uporabniške zgodbe z modeli poslovnih procesov, kot sta BPMN model in model primera uporabe.

2 Ozadje

Za pridobivanje uporabniških zgodb obstaja več pristopov. Najpogosteje uporabljen pristop je intervju [3, 13]. Cohn [3] predlaga delavnico za pisanje zgodb, ki združuje elemente možganskega viharjenja in prototipiranje. Ambler [7] predlaga, da se pridobivanje začne z uporabo obstoječe dokumentacije, in okvirno opisuje, kako pridobiti uporabniške zgodbe iz tekstovnega primera uporabe. Obstajajo tudi številne druge tehnike pridobivanja uporabniških zgodb, kot so vprašalniki, opazovanja [3] in fokusne skupine [2].

Za izboljšanje razumevanja integracijskih odvisnosti med uporabniškimi zgodbami obstaja nekaj pristopov. Na primer, Lin et al. [16] predlagajo metodo za razdelitev zapletenih procesov v zaporedne cilje in združevanje podrobnih uporabniških zgodb v nekaj ključnih ciljev. Clarke in Kautz [24] sta razvila metodo za vključevanje epov v uporabniške zgodbe. Leffingwell [2] predlaga model, ki hierarhično razgradi zahteve projekta. Liskin [25] zaključi, da je upravljanje ravni razdrobljenosti uporabniških zgodb področje, ki zahteva nadaljnje raziskave.

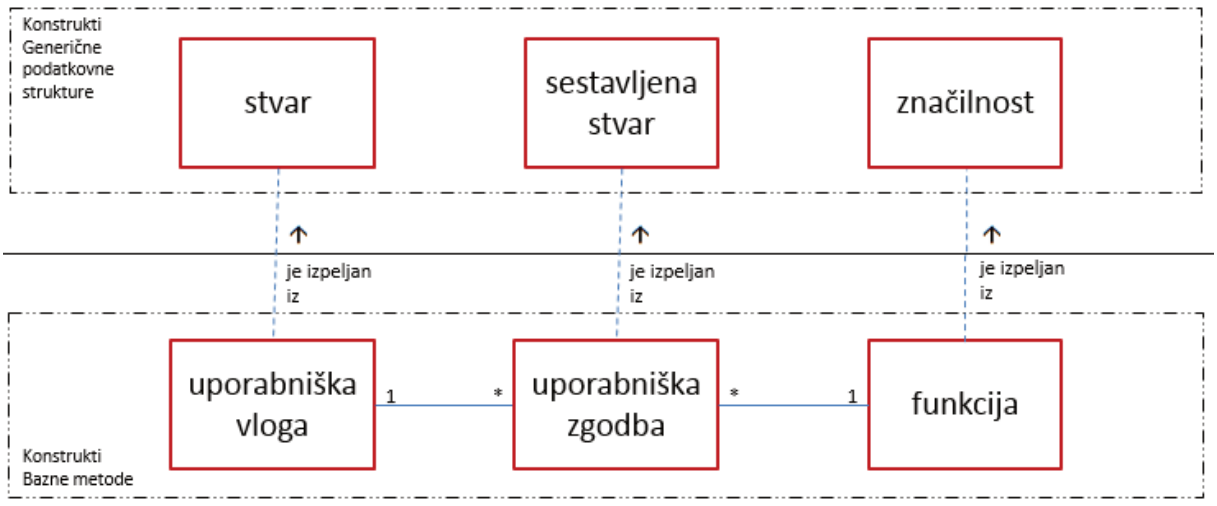
Za pridobivanje razumevanja izvedbenih odvisnosti med uporabniškimi zgodbami obstaja nekaj pristopov. Miličić et al. [26] predlagajo na uporabnike osredotočeno metodo, ki organizira uporabniške zgodbe v posamezne scenarije in po posameznih uporabnikih. Leffingwell [2] predlaga, da se izvedbene odvisnosti prikaže z diagrami primerov uporabe in s poročili, ki vključujejo specifikacijo primerov uporabe. Patton [27] predlaga tehniko za razbitje velike zgodbe v več manjših, pri čemer se še vedno ohranja velika slika projekta. Splošni namen omenjenih rešitev je, da ustvarijo konceptualni model, ki dopolnjuje seznam uporabniških zgodb z informacijami o integracijskih in izvedbenih odvisnostih.

3 Predlagana metoda

Metodo BuPUS smo razvili s pristopom, ki so ga predlagali Bajec et al. [44]. Njihov pristop temelji na razširitvi prej poznane metode, ki jo avtorji imenujejo bazna metoda. Razširitev se imenuje projektna metoda. Ta vključuje elemente bazne metode in dopolnitve, ki so prilagojene

posameznih situaciji/projektu. V okviru BuPUS predlagamo bazno metodo in dve projektni metodi. Prva projektna metoda temelji na uporabi BPMN modelov in druga na uporabi modelov primerov uporabe.

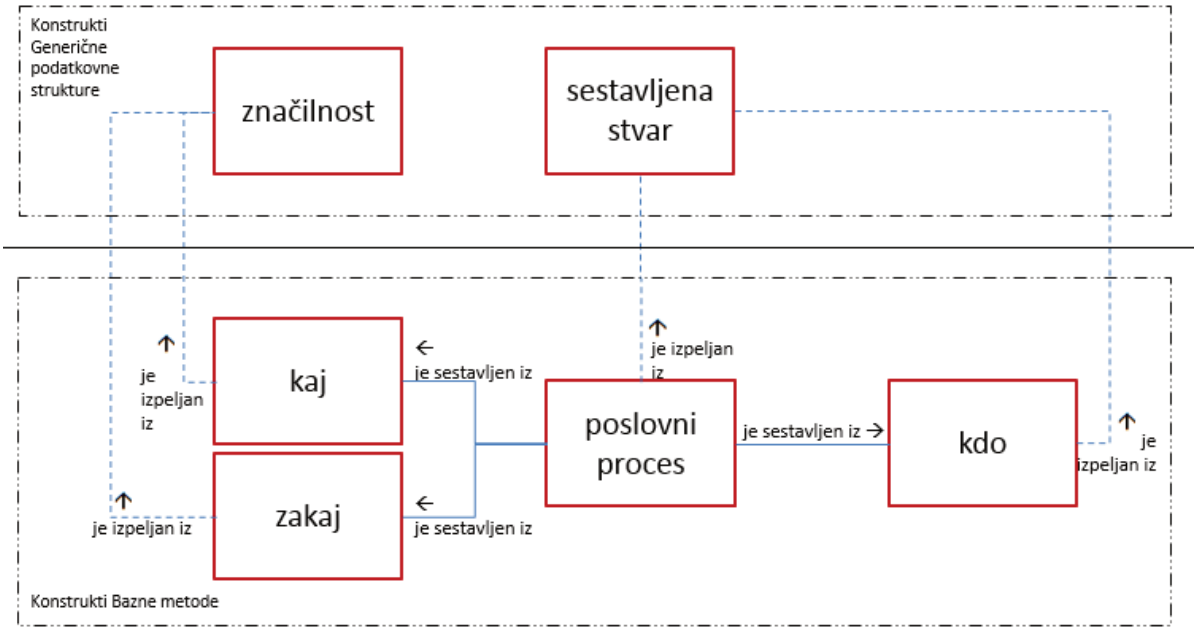
Bazna metoda je osredotočena na sestavne dele stavka uporabniške zgodbe: Jaz kot <uporabnik> lahko opravim <funkcijo>. Iz stavka smo izluščili tri konstrukte: uporabnik, funkcija in uporabniška zgodba. Uporabnik je organizacijska vloga v organizaciji naročnika, ki izvaja poslovne aktivnosti. Funkcija je neka poslovna aktivnost, ki bo podprta z novo aplikacijo. Definicija obeh konstruktov je podprta z definicijami višje nivojskih konstruktov iz tako imenovane generične podatkovne strukture. Generična podatkovna struktura je sestavljena iz treh med seboj povezanih konstruktov. Konstrukti, imenovani stvar, sestavljena stvar in značilnost, so definirani tako, kot so definirani istoimenski koncepti Bungejeve ontologije [132]. Svet je sestavljen iz stvari. Stvari imajo svoje značilnosti. Nekatere stvari so bolj kompleksne, tako da so sestavljene iz značilnosti in drugih stvari. Slednje imenujemo sestavljene stvari. Potemtakem je funkcija izpeljana kot značilnost ter uporabnik kot stvar. Uporabniška zgodba pa je sestavljena stvar, saj je sestavljena iz uporabnika in funkcije. Seznam uporabniških zgodb je v praksi opremljen tudi z informacijami o prioritetah in ocenah resursov, ki so potrebni za razvoj posamezne uporabniške zgodbe. Na te informacije se v naši metodi ne oziramo, ker verjamemo, da niso v pomoč pri nalogi pridobivanja uporabniških zgodb ali razumevanju medsebojnih odvisnosti med njimi.



Slika 1: Izpeljava konstruktov Bazne metode

Uporabniške zgodbe želimo podpreti z elementi poslovnega procesa in si s tem zagotoviti informacije potrebne za pridobivanje uporabniških zgodb in za razumevanje medsebojnih odvisnosti. V splošnem imajo poslovni procesi naslednje elemente: kaj mora biti narejeno, kdo

mora to narediti in zakaj [105]. Iz te definicije izhajajo štirje konstrukti: kdo, kaj, zakaj in poslovni proces. Konstrukt kdo predstavlja organizacijsko vlogo, konstrukt kaj predstavlja aktivnost, ki jo izvaja ta organizacijska vloga, in konstrukt zakaj predstavlja razlog, zakaj se skupek aktivnosti izvaja. Ker je nek poslovni proces lahko polno ali delno podprt z aplikacijo, bi lahko direktno povezali konstrukta kdo in uporabnik. Podobno bi lahko povezali konstrukta funkcija in kaj.



Slika 2: Izpeljava konstruktov poslovnega procesa

Želeli smo povezati konstrukta uporabniška zgodba in poslovni proces, vendar neuspešno. Sestavljena stvar poslovni proces je preveč splošna, zato smo se osredotočili na konkretne modele poslovnih procesov: BPMN model in model primerov uporabe.

V prvi projektni metodi se osredotočamo na grafične elemente BPMN modela [32], ki so v praksi pogosto v uporabi, na primer aktivnost, plavalna steza ipd. Konstrukt plavalna steza je okvir, s katerim ločujemo eno skupino aktivnosti od druge glede na to, katera organizacijska vloga jih izvaja. Plavalna steza je sestavljena stvar, ki opredeljuje organizacijsko vlogo (torej tip zaposlenega) z oddelkom, v katerem je ta vloga prisotna (konstrukt bazen), in z nazivom delovnega mesta v tem oddelku (konstrukt steza). Plavalna steza je izvedena iz konstrukta kdo.

Konstrukt aktivnost je enota dela, ki jo organizacija izvaja v svojem poslovnem procesu. Vsaka aktivnost ima dve značilnosti: opis aktivnosti in tip aktivnosti. Opis aktivnosti je stavek, ki na kratko opiše vsebino enote dela. Izveden je iz konstrukta kaj. Tip aktivnosti pa pove, ali je aktivnost definirana kot podproces ali ne.

Konstrukt BPMN model poslovnega procesa je skupek aktivnosti, ki so potrebne, da se ustvari poslovna dodana vrednost. Izveden je iz višje nivojskega konstrukta poslovni proces. Omenjeni konstrukt ima značilnost imenovano BPMN ime, ki predstavlja opis poslovne dodane vrednosti in je izveden iz konstrukta zakaj.

Omenjeni BPMN konstrukti omogočajo integracijo posameznega para konstruktov uporabniške zgodbe, ki sta uporabnik in funkcija, s posameznim parom konstruktov poslovnega procesa, ki sta opis aktivnosti in plavalna steza. Pri integraciji funkcije z opisom aktivnosti upoštevamo tri nivoje podrobnosti:

- nivo 1: funkcija je bolj abstraktna kot opis aktivnosti;
- nivo 2: funkcija je približno enako velika kot opis aktivnosti;
- nivo 3: funkcija je bolj podrobna kot opis aktivnosti.

Zaradi teh nivojev podrobnosti definiramo dva dodatna konstrukta: asociacija (izvedena iz svari) in nivo podrobnosti (izveden iz značilnosti). Asociacija daje informacijo o referenci na neko uporabniško zgodbo.

V naši drugi projektni metodi se osredotočamo na tekstualne elemente modela primera uporabe. RUP (Rational Unified Process) predlaga predlogo, ki podrobno in vseobsegajoče opisuje tekstualne elemente modela primera uporabe. Predloga se osredotoča na to, da bralec dobi informacije o tem, kaj uporabnik počne z aplikacijo in kako se aplikacija odziva na to početje. Ker je lahko model primera uporabe uporabljeni tudi kot manj vseobsegajoč [8], smo se odločili, da uporabimo samo tiste tekstualne elemente, ki so nujno potrebni tako za pridobivanje uporabniških zgodb kot za razumevanje njihovih medsebojnih odvisnosti. Naš model primerov uporabe opisuje, kaj uporabnik počne z aplikacijo in ne kako se aplikacija odziva na to početje. Za nas pomembna poglavja modela primera uporabe so: Glavni tok dogodkov, Alternativni tokovi dogodkov in Dodatne informacije. Glavni tok dogodkov predstavlja zaporedje glavnih aktivnosti [2]. Vsakič, ko so v poslovnem procesu možne odločitve, se v glavnem toku dogodkov pojavi sklic na poglavje Alternativni tokovi dogodkov [2]. V poglavje Dodatne informacije pa se sklicujemo v primerih, ko je potrebno dodatno pojasniti pogoje uporabe aplikacije, ki izhajajo iz poslovnega okolja.

Predlagamo sledeče konstrukte: model primera uporabe, ime primera uporabe, dogodek, opis dogodka, tip dogodka, akter, opis opombe in opis odločitve primera uporabe. Model primera uporabe je izveden iz višjenivojskega konstrukta poslovni proces. Ime primera uporabe definira cilj [2] in kot tak predstavlja razlog izvedbe poslovnega procesa. Izhaja iz višjenivojskega

konstrukta zakaj. Dogodek je izveden iz sestavljene stvari. Opis dogodka opisuje aktivnost zaposlenega. Izveden je iz višje nivojskega konstrukta kaj. Tip dogodka pove, ali nek dogodek pripada glavnemu ali alternativnemu toku dogodkov. Akter je nekdo, ki opravlja dogodek. Izveden je iz višjenivojskega konstrukta kdo. Opis opombe predstavlja pogoje, pod katerimi se lahko nek dogodek prične. Opis odločitve primera uporabe pa predstavlja pogoje, pod katerimi se lahko nek alternativni tok prične. Podobno kot pri BPMN modelu konstrukt asociacija povezuje/integrira uporabniško zgodbo z dogodkom. Integracija je opredeljena z enim od treh prej omenjenih nivojev podrobnosti.

S predlaganimi konstrukti BPMN modela in modela primera uporabe lahko podpremo delovanje predlaganih tehnik pridobivanja uporabniških zgodb. Tehnika pridobivanja uporabniških zgodb iz BPMN modela črpa vrednosti za konstrukta uporabnik in funkcija iz konstruktov plavalna steza in opis aktivnosti. Tehnika pridobivanja uporabniških zgodb iz modela primera uporabe pa črpa vrednosti za omenjena konstrukta iz konstruktov akter in opis dogodka. Iz omenjenih konstruktov BPMN modela in modela primera uporabe lahko s pomočjo predloge »jaz kot <uporabnik> lahko opravim <funkcijo>« ustvarjamo stavke, ki predstavljajo model uporabniške zgodbe. Pri BPMN modelu se tako osredotočimo na aktivnost, ki je povezana s kodo uporabniške zgodbe, in v predlogo te uporabniške zgodbe prepišemo opis aktivnosti ter ime steze, iz katere je aktivnost . Pri modelu primera uporabe pa se osredotočimo na dogodek in v predlogo prepišemo opis aktivnosti ter ime akterja.

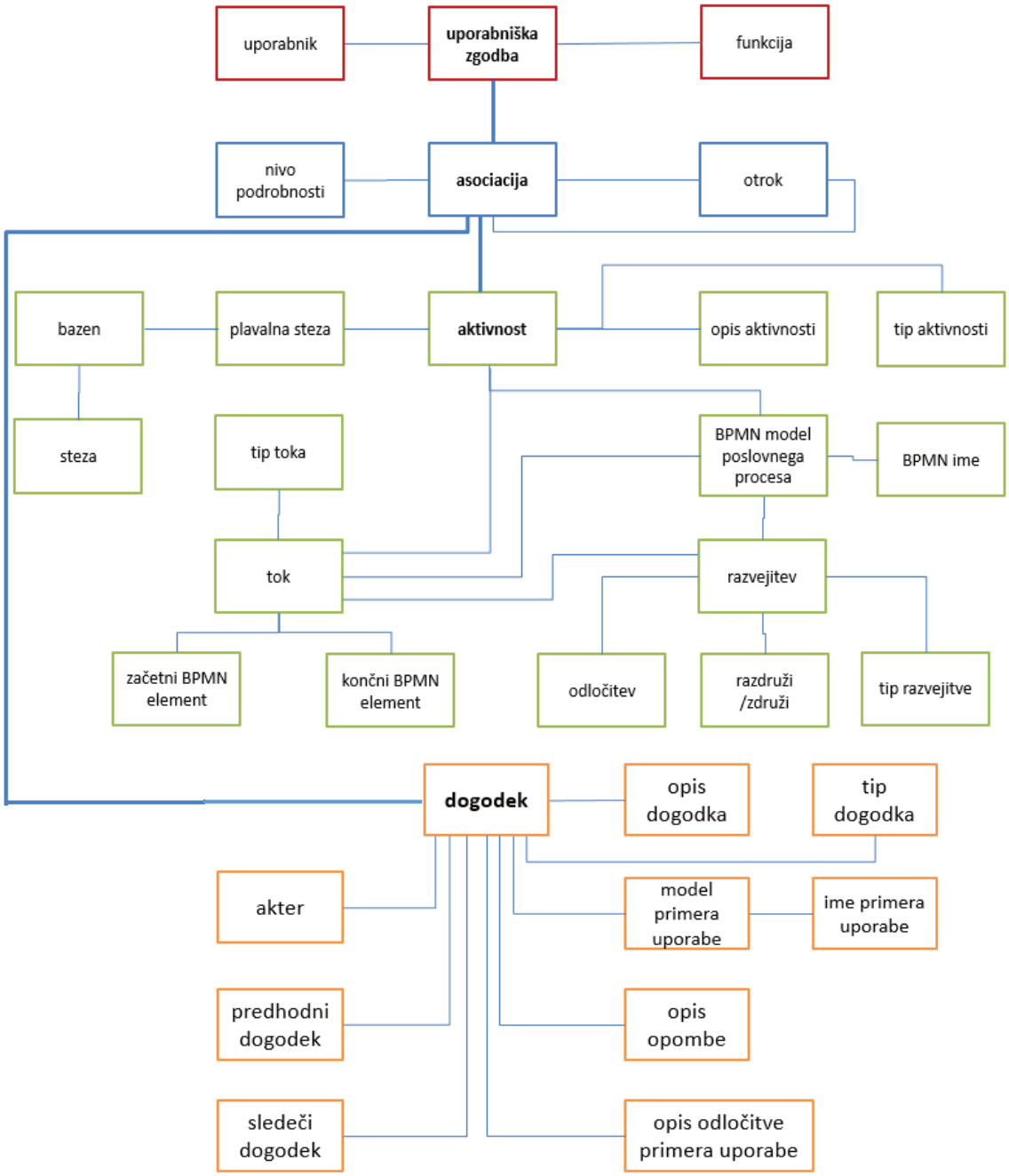
V okviru BuPUS metode predlagamo tudi tehniki za ugotavljanje izvedbenih odvisnosti uporabniške zgodbe iz BPMN modela in iz modela primera uporabe. Za ugotavljanje iz BPMN modela potrebujemo nove konstrukte: razvejitev in tok. Razvejitev pove, pod katerimi poslovnimi pogoji se določeni alternativni tokovi, ki sledijo iz razvejitve, sprožijo. Opis poslovnega pogoja je skrit v konstruktu odločitev. Vsaka razvejitev ima svoj začetek in konec. Samo začetni element razvejitve ima (lahko) opis poslovnega pogoja, medtem ko ga končni nima. Tako potrebujemo konstrukt razdruži/združi, ki si zabeleži, ali gre za začetni ali končni element razvejitve. BPMN nudi velik nabor različnih tipov razvejitev, kot sta OR in XOR, kar zabeležimo v konstruktu tip razvejitve. Prej omenjeni konstrukt tok predstavlja puščico, ki povezuje dve aktivnosti in nakazuje, katera se izvede prej in katera kasneje. Konstrukt tip toka pove, ali sta povezani aktivnosti opravljeni s strani različnih oddelkov/uslužbencev. Konstrukt začetni BPMN element in končni BPMN element povesta, kateri BPMN element je na začetku puščice in kateri na koncu. Z omenjenimi konstrukti lahko definiramo izvedbene tokove uporabniških zgodb, ki pripeljejo do izvedbe neke specifične uporabniške zgodbe. Najprej se

osredotočimo na asociirano aktivnost, nato pa potujemo po puščicah nazaj proti začetku modela. Vsakič, ko se na poti pojavi tip končne razvejitve, se za ena poveča število možnih poti oziroma različnih tokov, po katerih se pri izvajanju procesa proži zahteva po izvedbi neke določene uporabniške zgodbe.

Za delovanje tehnike za ugotavljanje izvedbenih odvisnosti s pomočjo modela primera uporabe prav tako definiramo nove konstrukte: predhodni dogodek in sledeči dogodek. Nova konstrukta predstavljata značilnost konstrukta dogodek in omogočata definicijo izvedbenih tokov, ki vodijo do zahteve po proženju neke izbrane uporabniške zgodbe. Najprej se osredotočimo na asociiran dogodek (njegovo vrstico), nato pa potujemo po tabeli navzgor. Tabela je v poglavju Alternativni tokovi dogodkov ali v poglavju Glavni tok dogodkov. Ko pridemo do na vrh nekega alternativnega toka, preskočimo na glavni tok in sicer tja, od koder je bil alternativni tok klican. V tabeli glavnega toka prav tako potujemo navzgor do prvega dogodka, kjer se ustavimo.

Kot zadnji dve tehniki BuPUS metode predlagamo tehniki za ugotavljanje integracijskih odvisnosti uporabniške zgodbe iz BPMN modela in iz modela primera uporabe. Za ugotavljanje teh odvisnosti potrebujemo obstoječa konstrukta asociacija in nivo podrobnosti ter nov konstrukt otrok. Otrok nam poda informacijo o integracijski odvisnosti izbrane uporabniške zgodbe od neke višje nivojske uporabniške zgodbe. Integracijsko odvisnost v BPMN modelu prikažemo z gnezdenjem objektov referenc uporabniških zgodb. Torej, če je referenca uporabniške zgodbe znotraj objekta (množice) neke druge reference uporabniške zgodbe, je prva uporabniška zgodba bolj podrobna od druge ter integracijsko odvisna od nje.

Slika 3 predstavlja BuPUS konstrukte iz najbolj podrobnega nivoja. V rdečem okvirju so konstrukti uporabniške zgodbe, v zelenem BPMN konstrukti, v oranžnem konstrukti primera uporabe in v modrem so konstrukti asociacije.



Slika 3: BuPUS konstrukti

4 Hipoteze

Našega raziskovalnega vprašanja smo se lotili v dveh zaporednih raziskovalnih fazah. V sklopu prve faze trdimo, da so povezani modeli poslovnih procesov koristni za boljše razumevanje integracijskih in izvedbenih odvisnosti med uporabniškimi zgodbami. Dokazujemo naslednji trditvi.

> Trditev 1: Razumevanje izvedbenih odvisnosti med uporabniškimi zgodbami je večje pri branju BPMN materiala kot pri branju seznama uporabniških zgodb.

Trditev 2: Razumevanje integracijskih odvisnosti med uporabniškimi zgodbami je večje pri branju BPMN materiala kot pri branju seznama uporabniških zgodb.

V sklopu druge faze trdimo, da so vizualni modeli poslovnih procesov bolj koristni za razumevanje odvisnosti ter za pridobivanje uporabniških zgodb kot pa tekstovni modeli poslovnih procesov. Postavimo naslednje trditve:

Trditev 3: Razumevanje integracijskih odvisnosti med uporabniškimi zgodbami je večje pri branju vizualnih BPMN materialov kot pri branju tekstovnih primerih uporabe.

Trditev 4: Razumevanje izvedbenih odvisnosti med uporabniškimi zgodbami je večje pri branju vizualnih BPMN materialov kot pri branju tekstovnih primerih uporabe.

Trditev 5: Pridobivanje uporabniških zgodb je bolj učinkovito pri uporabi vizualnih BPMN materialov kot pri branju tekstovnih primerih uporabe.

Pri izgradnji hipotez se sklicujemo na Mayerjevo teorijo [39], ki pravi, da z obogatitvijo teksta z vizualnimi elementi pridobimo večje razumevanje o izbrani tematiki. Z definicijo mnogih BuPUS konstruktov, ki zajemajo značilnosti uporabniške zgodbe, BPMN modela in modela primerov uporabe, smo predstavili slovnico predlagane metode. Izraznost slovnice, ki se nanaša na BPMN model, je za razumevanje odvisnosti med uporabniškimi zgodbami večja kot pri tisti, ki se nanaša na seznam uporabniških zgodb. Tako pričakujemo boljše rezultate z BPMN materialom. Izraznost slovnice, ki se na nanaša na BPMN model, in tiste slovnice, ki se nanaša na model primera uporabe, je različna vendar potencialno enako močna. Ne glede na to pričakujemo, da bodo imeli vizualni elementi BPMN modela večji vpliv na razumevanje integracijskih in izvedbenih odvisnosti med uporabniškimi zgodbami kot pa tekstualni elementi model primera uporabe.

V prvi raziskovalni fazi se želimo prepričati, ali lahko pričakujemo večje razumevanje odvisnosti med uporabniškimi zgodbami, če seznamu uporabniških zgodb na poseben način dodamo model poslovnega procesa. Pri tem nas zanima, kako je s tremi nivoji povezljivosti med uporabniško zgodbo in aktivnostjo. Iz Trditve 1 izhajajo prve tri hipoteze, medtem ko iz Trditve 2 naslednje tri:

H1: Ko je uporabniška zgodba bolj abstraktna kot asociirana aktivnost v modelu poslovnega procesa, je razumevanje izvedbenih odvisnosti med uporabniškimi zgodbami večje, če si pomagamo z BPMN materialom kot pa samo s seznamom uporabniških zgodb.

H2: Ko je uporabniška zgodba približno enako velika kot asociirana aktivnost v modelu poslovnega procesa, je razumevanje izvedbenih odvisnosti med uporabniškimi zgodbami večje, če si pomagamo z BPMN materialom kot pa samo s seznamom uporabniških zgodb.

H3: Ko je uporabniška zgodba bolj podrobna kot asociirana aktivnost v modelu poslovnega procesa, je razumevanje izvedbenih odvisnosti med uporabniškimi zgodbami večje, če si pomagamo z BPMN materialom kot pa samo s seznamom uporabniških zgodb.

H4: Ko je uporabniška zgodba bolj abstraktna kot asociirana aktivnost v modelu poslovnega procesa, je razumevanje integracijskih odvisnosti med uporabniškimi zgodbami večje, če si pomagamo z BPMN materialom kot pa samo s seznamom uporabniških zgodb.

H5: Ko je uporabniška zgodba približno enako velika kot asociirana aktivnost v modelu poslovnega procesa, je razumevanje integracijskih odvisnosti med uporabniškimi zgodbami večje, če si pomagamo z BPMN materialom kot pa samo s seznamom uporabniških zgodb.

H6: Ko je uporabniška zgodba bolj podrobna kot asociirana aktivnost v modelu poslovnega procesa, je razumevanje integracijskih odvisnosti med uporabniškimi zgodbami večje, če si pomagamo z BPMN materialom kot pa samo s seznamom uporabniških zgodb.

V drugi raziskovalni fazi se želimo prepričati, ali so vizualni modeli procesov boljši od tekstualnih pri razumevanju odvisnosti med uporabniškimi zgodbami. Za vizualni model procesov smo si izbrali BPMN model, ki je pogosto uporabljen pri analizi poslovanja, in model primera uporabe, ki je pogosto uporabljen pri razvijalnih projektih. Pri tem nas zanima, kako je z dvema nivojema povezljivosti med uporabniško zgodbo in aktivnostjo. Iz Trditve 3 izhajata prvi dve hipotezi, iz Trditve 4 pa naslednji dve:

H7: Ko je uporabniška zgodba približno enako velika kot asociirana aktivnost v modelu poslovnega procesa, je razumevanje izvedbenih odvisnosti med uporabniškimi zgodbami večje, če si pomagamo z BPMN materialom kot pa z materialom primera uporabe.

H8: Ko je uporabniška zgodba bolj abstraktna kot asociirana aktivnost v modelu poslovnega procesa, je razumevanje izvedbenih odvisnosti med uporabniškimi zgodbami večje, če si pomagamo z BPMN materialom kot pa z materialom primera uporabe.

H9: Ko je uporabniška zgodba približno enako velika kot asociirana aktivnost v modelu poslovnega procesa, je razumevanje integracijskih odvisnosti med uporabniškimi zgodbami večje, če si pomagamo z BPMN materialom kot pa z materialom primera uporabe.

H10: Ko je uporabniška zgodba bolj abstraktna kot asociirana aktivnost v modelu poslovnega procesa, je razumevanje integracijskih odvisnosti med uporabniškimi zgodbami večje, če si pomagamo z BPMN materialom kot pa z materialom primera uporabe.

V okviru druge raziskovalne faze želimo preveriti, pri katerem materialu je pridobljenih več pravilnih uporabniških zgodb. Iz Trditve 5 izhaja ena hipoteza:

H11: Več pravilno pridobljenih uporabniških zgodb dobimo, z BPMN modelom kot pa z modelom primera uporabe.

5 Raziskava

Pri merjenju razumevanja uporabniških zgodb se zgledujemo po predhodnih člankih na povezane teme [34-37]. Podobno kot ti avtorji, gradimo na teoretičnem okvirju za empirično ocenjevanje konceptualnih metod za modeliranje [38]. Razumevanje uporabniških zgodb merimo s testi reševanja problemov [35, 39], ki obravnavajo izvedbene in integracijske odvisnosti.

Pripravili smo dva eksperimenta. V pripravah na prvi eksperiment smo izvedli približno deset pilotnih testov. Oblikovali smo dva delovna zvezka z eksperimentalnimi nalogami: Zvezek 1 in 2. V obeh zvezkih smo obravnavali tako primer z BPMN materialom kot primer (samo) s seznamom uporabniških zgodb vendar na dveh različnih študijah primera. V pripravah na drugi eksperiment smo izvedli tri pilotne teste. Oblikovali smo štiri delovne zvezke z eksperimentalnimi nalogami: Zvezek A, B, C in D. V vseh zvezkih smo obravnavali tako primer z BPMN materialom kot primer z materialom primera uporabe, vendar na dveh novih študijah primera. V Zvezkih A in B so študenti delali najprej z BPMN materialom in nato z materialom primera uporabe. V Zvezkih C in D pa najprej z materialom primera uporabe in šele nato z BPMN materialom.

Eksperimente smo izvedli aprila in novembra 2015. Prvi eksperiment smo ponovili v sedmih razredih in drugi v šestih. Ponovitve so sledile natančno določenemu protokolu. Pri prvem eksperimentu smo pridobili 127 neoporečnih rešenih delovnih zvezkov in pri drugem 119.

Analiza pridobljenih podatkov je pokazala različne rezultate. S testom Kolmogorov-Smirnov smo najprej preverili, ali so vrednosti spremenljivk normalno porazdeljene in ugotovili, da niso. Tako smo pri potrjevanju hipotez iz obeh eksperimentov uporabili ne-parametričen test imenovan Mann-Whitney. Hipoteze od H1 do H8 smo potrdili, medtem ko hipotez od H9 do H11 nismo.

6 Zaključek

Naša raziskava vsebuje doprinos k znanju tako za razvijalce kot za raziskovalce. Osredotočili smo se na dva pomembna problema iz prakse. Prvi se osredotoča na pomanjkanje pristopov pridobivanja uporabniških zgodb iz obstoječe dokumentacije, drugi na pomanjkanje pristopov za razumevanje integracijskih in izvedbenih odvisnosti med uporabniškimi zgodbami. V okviru metode BuPUS smo tako predstavili šest tehnik, ki vse izhajajo iz ponovne uporabe obstoječih modelov poslovnih procesov. Delovanje tehnik smo utemeljili s slovnico, ki smo jo sestavili iz različnih konstruktov in jo preverili z eksperimentoma.

7 Literatura

Z referenčnimi številkami se sklicujemo na seznam literature, ki je na voljo v poglavju 7 References.