UTRECHT UNIVERSITY

Using Job Stories and Jobs-to-be-Done in software requirements engineering

by

Maxim van de Keuken

Supervisors: Sjaak Brinkkemper (UU), Garm Lucassen (UU) and Johan Schlingmann (Stabiplan)

A thesis submitted in partial fulfillment for the degree of
Master of Business Informatics

Faculty of Science
Department of Information and Computing Sciences

November 2017

Abstract

In the field of requirements engineering (RE) for software products, lightweight approaches such as User Stories have gained substantial adoption. While critics highlight the limitations of User Stories, Job Stories are emerging as an alternative that emphasizes situations, motivations and expected outcomes. Building on the broader theory of Jobs-to-be-Done, this new approach has not been studied in research yet. Scientific foundations are lacking for the Job Story artifact and there are no actionable methods for effectively applying Job Stories. Thus, practitioners end up creating their own variant of Job Stories that fail to deliver the promised value of the Jobs-to-be-Done theory. We integrate theories from multiple Job Story authors to create a conceptual model of Job Stories and to construct a generic method for Jobs-to-be-Done Oriented RE. Applying our Job Story Method to an industry case study, we highlight benefits and limitations. Our method aims to bring Job Stories from craft to discipline, and provide systematic means for applying Jobs-to-be-Done Oriented RE in practice and for assessing its effectiveness.

Keywords: Job Stories; Jobs-to-be-Done; Software Requirements Engineering.

Preface

Writing this section marks the end of my journey into the world of Jobs-to-be-Done, and the completion of this master thesis. From start to finish, this has been an interesting, challenging, and above all tremendously educational experience for me. Since I would not have been able to get this job done alone, those that supported me throughout this project deserve acknowledgment.

Thanks are due to Garm Lucassen and Sjaak Brinkkemper, who have introduced me to this interesting topic, provided much needed guidance throughout the project and gave me a taste of the academic world through our work on the paper. I am also grateful to Fabiano Dalpiaz for his help during the early phases of this project.

For making the case study possible and being involved in this project, there are many people at Stabiplan that deserve thanks. I would like to express special gratitude to Johan Schlingmann for all his time, support, insights and enthusiasm that went into this project.

Finally, I am grateful to Jacqueline and my family for their never-ending support during my studies.

Contents

Chapter 1

Introduction

In the context of software development, User Stories describe functionality that will be valuable to either a user or purchaser of a system or software [1]. With adoption of over 55%[2], User Stories are becoming widely used during Requirements Engineering (RE), and are especially popular in agile software development [3]. Overall, software professionals are predominantly positive about User Stories, the associated templates, and quality guidelines [4].

However, practitioners have found User Stories not to be without faults. The main criticisms they express are that User Stories are too much based on assumptions and do not acknowledge causality [5]. While each User Story should be associated with at least one user role or persona [1] and should express what that persona wants to achieve, some practitioners report missing information on motivation and context[6]. Because of this, the question of 'why', that underlines each requirement, remains unanswered.

It is the focus on personas that has proved to be a major source of criticism for User Stories. Practitioners argue that the accuracy of personas is difficult or impossible to verify [7], and that building personas is a process that is prone to prejudices[8]. To help address these issues, in 2013 Alan Klement introduced a new paradigm for requirements formulation: Job Stories [6]. Designed to overcome perceived weaknesses of User Stories, Job Stories capture requirements in an alternative format that emphasizes the motivational and situational context that drive customer behavior. The differences in format between User- and Job Stories are illustrated below in table 1.1.

Story	**Format**
User Story	**As a** <role>, **I want** <goal>, [**so that** <benefit>] [1]
Job Story	**When** <situation>, **I want to** <motivation>, **so that I** <expected outcome>[5]

Table 1.1: User- and Job Story formats

Job Stories are based on the foundational ideas of the Disruptive Innovation Theory's method *Jobs-to-be-Done* by Clayton Christensen [9], which is a collection of principles that helps you discover and understand the interactions between customers, their motivations and the products they use [10]. In 2005, Christensen formulated the central notion behind Jobs-to-be-Done: Customers do not buy products because they want them, they hire products to help them get a Job done that they are struggling with in their lives [11]. For innovators, it should therefore be the top priority to understand what Job customers are hiring their product to do [9].

To help explain Jobs-to-be-Done, Christensen often refers to the case of a fast-food restaurant chain that was looking to improve the sales of its milkshakes. In this case, researchers found that customers were hiring milkshakes for the Job of eating breakfast in their car. The milkshakes were the best solution for this Job because they are easy to drink whilst driving, and since they take a long time to consume, milkshakes provide some entertainment on a long commute to work [9].

1.1 Problem Statement

There is no consensus on what exactly constitutes Jobs-to-be-Done theory [10]. As a result, different authors have proposed different approaches to Jobs-to-be-Done. While grounded in the same concepts, these approaches provide different perspectives. Due to a general lack of well-documented examples of the application of these approaches, and strong disagreements between thought-leaders in the field, it can be confusing for practitioners looking to apply Job Stories and Jobs-to-be-Done in their work.

Since most of the examples that are available are for physical products, Jobs-to-be-Done is particularly difficult to apply in the context of software. In the few cases where it has been applied, practitioners rarely document their process. To summarize, the problem addressed in this study can be formulated as follows:

'The fragmented state of Jobs-to-be-Done literature and a lack of well-documented examples from practice, makes it difficult for practitioners in the software industry to determine whether and how to apply the theory.'

This problem is important because it limits the applicability and diffusion of Jobs-to-be-Done and Job Stories. Furthermore, without well-documented and independent case studies, it is difficult to judge the value that Jobs-to-be-Done and its artifacts such as Job Stories add in the software industry.

1.2 Objective, Scope, and Structure

The main objective of this research is to address this problem and help provide clarity on what Jobs-to-be-Done and Job Stories are, and especially how they can be applied in the software industry. We aim to address this problem by summarizing the current state of Jobs-to-be-Done literature, and integrating the different concepts and approaches into a practical method for use in the context of requirements engineering for software products. We apply this method in a case study at a product software company in the Netherlands.

This study follows the following structure: in chapter 2, the research approach is described. In chapter 3, Jobs-to-be-Done theory is explored and the different ways to apply it are discussed. In chapter 4 we analyze how Job Stories and Jobs-to-be-Done are positioned in the context of the software industry, and propose a concrete method for using Job Stories in that context. Chapter 5 introduces the case study, and in chapter 6 we present and analyze the results. Finally, in chapter 7 we answer the research question through a discussion of the results.

Chapter 2

Research Approach

2.1 Research Questions

As previously introduced, this research focuses on exploring the use of Job Stories and Jobs-to-be-Done in the context of software products. For practitioners looking to apply this new approach in their own environments, the fragmented body of literature on Jobs-to-be-Done can be confusing. Different authors present different approaches for applying it, and there is a general lack of well-documented examples of practical application, especially in the context of software.

Due to this situation, it is difficult to judge the added value of Job Stories and Jobs-to-be-Done when applied in the context of software products, and it is difficult for practitioners to put the theory into practice. Since overall goal of this work is to help address these problems, we arrive at the following main research question:

'What is the value of Job Stories and Jobs-to-be-Done in the context of requirements engineering for software products?'

A first step in answering this question, is gaining a solid understanding of Jobs-to-be-Done and its artifacts such as Job Stories. This is reflected in the first two sub-questions:

SRQ1: *'What are the principles of Jobs-to-be-Done theory?'*

SRQ2: ''What are the main concepts and artifacts related to Jobs-to-be-Done?'

To help build a shared understanding of what applying Jobs-to-be-Done entails, we need to explore the different approaches and analyze what makes them different. This brings us to the third and fourth sub-questions:

SRQ3: *'What are the existing approaches to applying Jobs-to-be-Done?'*

SRQ4: *'How do the different approaches to applying Jobs-to-be-Done relate to each other?'*

Through answering these first four sub-questions of Jobs-to-be-Done theory and the main approaches that exist for its application. However, Jobs-to-be-Done is usually described in the context of physical products and since this research focuses on the software industry, we need to explore how the theory can be used in this context. This brings us to the fifth and sixth sub-questions:

SRQ5: *'What are examples of the application of Job Stories and Jobs-to-be-Done in the context of software products?'*

SRQ6: *'What main usage scenarios can be identified for Job Stories and Jobs-to-be-Done in the context of software products?'*

To answer the main research question and draw a conclusion on the value of Job Stories and Jobs-to-be-Done in the software industry, it is not enough to investigate how they are used. It is also necessary to reflect on how it relates to existing techniques that are already used in the software industry for similar purposes. This brings us to our seventh sub-question:

SRQ7: *'How are Job Stories and Jobs-to-be-Done positioned in the landscape of requirements engineering for software products?'*

Through the previous sub-questions we have gained insight into the different ways Job Stories and Jobs-to-be-Done can be applied to add value over existing techniques used in the software industry. To help practitioners in the software industry conduct requirements engineering via Jobs-to-be-Done, we need to build on this knowledge and investigate how the available literature can be used to create a full fledged method for the use of Job Stories and Jobs-to-be-Done in that context. By formulating the method using a Process-Deliverable-Diagram (PDD), it can capture both the most important Jobs-to-be-Done and Job Story activities, as well as the utilized artifacts [12]. The creation of a method is reflected in the eighth sub-question:

SRQ8: *'What is a method for using Job Stories and Job-to-be-Done to perform requirements engineering for a software product?'*

To judge the value of the method, it should be applied in practice. We do this in a case study at Stabiplan B.V. A detailed overview of the case study is provided in chapter 5.

In short, the case revolves around the apps that Stabiplan is adding to its product portfolio. The apps are small and independent products that will be offered alongside Stabiplan's larger products, but are intended to serve a different, more global market. For Stabiplan, the challenge is to determine what functionalities should be encapsulated into an app, in order to incite users to adopt them.

In the case study, we apply the created method to perform the requirements engineering for a specific new app. The application of the method in the case study is described in chapter 6. In order to be able to answer the main research question, it is important to reflect on the experience of applying the method in the case study and perform an evaluation of both the project and the method. The goal of the evaluation is to get a sense of the value of the method, and by extension help come to a meaningful conclusion with regard the value of Job Stories and Jobs-to-be-Done in the context of requirements engineering for software products.

The approach to the evaluation is described in section 5.2, and the results of the evaluation are reported in section 6.2. Apart from a reflection on the method from a theoretical perspective, the main goal of the evaluation is to gauge the value of the method for Stabiplan. Since 'value' is a difficult construct to measure, we focus on investigating the impact of the application of the method on the business process at Stabiplan. We do this by performing interviews with key stakeholders at Stabiplan. This brings us to the 9th sub-question:

SRQ9: *'In what ways did the method positively or negatively impact business processes at Stabiplan?'*

Since the main research question is broadly oriented towards the software industry as a whole, it should be determined to what degree the results of the case study still apply in this broader context. Therefore, the 10th, and final, sub-question for this project is:

SRQ10 : *'What findings from the case study can be generalized outside the context of the case?'*

We reflect on this sub-question during the discussion which is included in chapter 7.

2.2 Research Design

The research methods used in this project consist of a literature review and a case study. The literature review is performed during the exploratory phase of the project, and is

used to answer the first seven sub-questions. Afterwards, the findings are applied in a case study aimed at answering sub-questions 8, 9 and 10.

2.2.1 Literature Study

As mentioned earlier, the first seven sub-questions for this research are to be answered using a literature study. Important to note is that due to a lack of scientific literature on the topic, a substantial amount of the literature that discusses the use of Jobs-to-be-Done in the context of software products is written by practitioners.

The snowballing technique has been used to explore both types of literature. As a starting point for these snowballing sessions influential works of Dr. Clayton Christensen [9, 13, 14], Anthony Ulwick [15–18], and Alan Klement [6, 8, 10, 19–21]. In addition to this, there are many other practitioners in the Jobs-to-be-Done "community" who write about their thoughts and experiences related to Jobs-to-be-Done on various online platforms [22–26].

2.2.2 Case Study

In his prominent work on case studies, Yin identifies three types of case studies: descriptive, exploratory and explanatory studies, but also argues that the boundaries between each type are not always clean and sharp [27]. This study focuses on exploring the use of Jobs-to-be-Done in the context of software products. The goal here is to produce some propositions that are to be explored in further research. Therefore, the case study conducted in this project can be classified as an exploratory case study that involves a single-case [27].

2.3 Relevance

By providing an exploration of Jobs-to-be-Done theory that is aimed at the software industry, and proposing a method to apply it in that context, this study is relevant from scientific- and practitioner perspectives.

2.3.1 Scientific Relevance

Since the introduction of Job Stories in 2013, practitioners in the software industry have been experimenting with Job Stories and Jobs-to-Done, and writing about their

experiences. However, this trend has not yet been picked-up by the scientific community, as there is barely any scientific literature on he use of Job Stories and Jobs-to-be-Done in the software industry.

Since Job Stories and Jobs-to-be-Done challenge extensively researched techniques from the field of software requirements engineering such as User Stories and Persona's, from a scientific perspective it is very relevant to investigate to what extend these claims from practitioners are justified and why. As such, this topic and study is very relevant from a scientific perspective. By providing an initial exploration of Job Stories and Jobs-to-be-Done and providing the explicitly documented case study, this work will hopefully facilitate scientific debate, and fuel further research on the subject.

2.3.2 Practitioner Relevance

Judging by the blogs and discussions that are posted on the internet, practitioners in the software industry are increasingly interested in Job Stories and Jobs-to-be-Done. Although still an upcoming approach, different authors propose different techniques and approaches for Jobs-to-be-Done. This is confusing for practitioners who are unsure what approach to Job Stories and Jobs-to-be-Done is best, and how to apply it in their own environments. This situation is especially relevant for practitioners in the software industry, as many of the approaches and examples are for physical products.

This study is relevant for practitioners since it aims to provide some clarity in regard to the fragmented and sometimes convoluted body of literature that is available on Jobs-to-be-Done. By proposing and evaluating a concrete method for the use of Job Stories in software requirements engineering, this study can be helpful for practitioners looking to use Job Stories in their own work.

2.3.3 Contributions

As an exploratory study on the use of Job Stories and Jobs-to-be-Done in the context of software products, this work makes the following contributions that are relevant from both a practitioner and scientific perspective:

1. In chapter 3, we explore the fragmented body of literature on Jobs-to-be-Done to provide an holistic and objective overview of what the theory entails, and build a conceptual model of Job Stories.

2. In chapter 4, we investigate the use of Job Stories and Jobs-to-be-Done in the context of software products. By doing so, we work towards positioning Jobs-to-be-Done and Job Stories in the landscape of techniques used to perform Requirements Engineering for software products.

3. Also in chapter 4, we introduce the Integrated Job Story Method that integrated the different views on Jobs-to-be-Done and Job Stories literature.

4. In chapters 5 and 6, we evaluate the applicability of the Integrated Job Story Method with a case study in which we define the high-level requirements for two apps,

5. In chapter 7, we reflect on the project and answer the research questions posited earlier.

Chapter 3

Jobs-to-be-Done Theory

In this section an overview of the theory on Jobs-to-be-Done is provided. As such, we will first discuss the background and core concepts of the theory. Afterwards we will dive into the different approaches to applying it, and see if we can identify some main usage scenarios.

3.1 Background of the Theory

The theory on Jobs-to-be-Done is based on a diverse set of underlying principles. In this section the most prominent of these principles are discussed.

Joseph Schumpeter – 'Creative Destruction'
In his book 'Capitalism, Socialism and Democracy', Schumpeter introduces 'Creative Destruction' [28]. This is a process of industrial mutation that revolutionizes the existing economic structure from within, destroys it, and creates a new structure. The 'Incumbent's Curse' describes a tendency for large, incumbent firms to focus on solidifying their market position with incremental innovations, instead of creating radical product innovation. As such, incumbents are vulnerable to 'Creative Destruction' at the hand of small firms that bring radical product innovations. However, in subsequent research the 'Incumbent's Curse' was found to not always be valid [29].

William Edwards Deming – 'System of Profound Knowledge'
In 1986 Deming proclaimed that the world had entered a new economic age where "quality management" was vital for a firm's survival. In his management theory "The system of profound knowledge", Deming outlined 14 points for management to thrive in this new age [30]. Deming is most famous for helping Japan go through a revolution in quality and economic production since 1950 [31].

For Jobs-to-be-Done, the most important point in 'The system of profound knowledge', is the notion that all organizations must be viewed as a system of processes. The goal should be to optimize the entire system, not individual components [30]. This focus on connectedness – referred to as interdependence – is an important part of innovation and Jobs-to-be-Done [6].

Amos Tversky and Daniel Kahneman – 'Cumulative Prospect Theory'
This psychological theory is a further development of prospect theory, and is a model for descriptive decisions that applies to uncertain, as well as risky prospects with any number of outcomes. The main point is that people generally have an attitude of loss aversion, where they care more about potential losses than potential gains [32].

Jobs-to-be-Done theory uses this psychological concept to help explain why customers make certain decisions when buying and using products. Irregularities in decision making are the result of emotional forces that influences the motivation of a potential customer to adopt a product [10].

Gary Klein – 'Naturalistic Decision Making'
Similarly to Cumulative Prospect Theory, the Naturalistic Decision Making (NDM) framework is used in Jobs-to-be-Done to help explain how customers make decisions [10]. The defining feature of the NDM framework is that it is focused on the role of experience in enabling people to quickly categorize situations in order to make effective decisions [33].

Bob Moesta, John Palmer and Rick Pied – 'Jobs-to-be-Done language'
Practitioners Bob Moesta, John Palmer and Rick Pied are credited for taking the previously described influential theories, and combining them with their own experience to create the first Jobs-to-be-Done principles. In doing this, they created the language that defines Jobs-to-be-Done: customers have "jobs" that they are trying to get "done" [10].

Theodore Levitt – "Marketing Myopia"
Theodore Levitt is proof that the core principles behind Jobs-to-be-Done are not new. Levitt is often referenced in Jobs-to-be-Done literature for making the point that customers do not want a product, they want help getting a job done. Levitt made this point by appropriating a quote from Leo McGinneva to create an analogy: "people do not want a quarter inch drill, they want a quarter inch hole" [34]. In 1960, he published "Marketing Myopia", in which he already argued that companies should not define themselves by what they produced, but instead by what customer need they were trying to address [35].

Michael T. Bosworth – "Solution selling: Creating buyers in difficult selling markets"
Another concept that seems very closely related to Jobs-to-be-Done is Solution Selling.

This is a sales methodology that prescribes to – instead of focusing on the promotion of an existing product – focus on the customer's pains, and addressing these with new products or services [36].

Following this approach is especially valuable in situations where products or services are hard sell because they are difficult to describe, intangible, have long sell cycles or expensive [37].

Bettencourt, Lush and Vargo – "A service lens on value creation"
Similar to Solution Selling, is 'Service-Dominant Logic' (SDL) which is a marketing theory focused on the demand side of marketing [38]. SDL dictates that everything a company sells is a service – even products. In this sense, a company that sells air conditioners is really selling the service of keeping the inside of a building cool [39].

The authors argue that SDL has emerged as a way of understanding value creation and is made actionable with Jobs-to-be-Done, using the Job Mapping technique. Job Mapping is discussed in more detail in section 3.4. In short, it is used to understand the customer's perspective of getting a job done in a way that cuts across different resources – different products for instance. The goal is to map – from a customer perspective – what must be done to get the overall job done, rather than mapping how things are currently being done [16].

3.2 Core Theory

Principles from the previously described theories and concepts come together in Jobs-to-be-Done. However, as stated by Alan Klement, the body of literature on Jobs-to-be-Done is fragmented and therefore there is no clear consensus on what exactly constitutes Jobs-to-be-Done theory. This is reflected in his definition of Jobs-to-be-Done: "a collection of principles that helps you discover and understand the interactions between customers, their motivations, and the products they use" [10]. In this context, a Job is the fundamental problem a customer needs to resolve in a given situation [13].

However, while there are certainly some ideas that are universally accepted as Jobs-to-be-Done, other ideas have given cause for significant disagreement between thought-leaders in the field. As a result, one could conclude that the landscape of ideas related to Jobs-to-be-Done has diverged into two roughly defined camps. On the one hand there are practitioners, such as the aforementioned Alan Klement and Dr. Clayton Christensen, who view Jobs-to-be-Done as a strictly qualitative approach. On the other hand there are practitioners, most notably Anthony Ulwick and his innovation consulting firm Strategy, who favor a more quantitative approach to Jobs-to-be-Done. These two approaches will

be discussed in more detail later in this section, but first we will shortly recap the common ground that defines Jobs-to-be-Done.

3.2.1 Common ground in Jobs-to-be-Done

It is important to note that the previously introduced approaches to Jobs-to-be-Done are mainly different from a practical perspective. From a theoretical perspective they are still grounded on roughly the same theoretical concepts.

Fundamentally, Jobs-to-be-Done is based on the notion that customers do not shop for products. Instead they browse the market, looking for the best solution for a problem or opportunity they experience in their lives.

This analogy between the product selection process and the job market very effectively illustrates the point of Jobs-to-be-Done. In the job market, a good hiring manager understands the job that he or she is hiring for, and has a list of criteria that are used to determine which candidate is the best fit for the Job. From the perspective of a job applicant, this means that the key to getting hired is to understand the Job and criteria are associated with it, and to convince the hiring manager that you are the best choice. For a candidate it is not only important to have the required skills, but also to present yourself in such a way that it is clear that you are the right person for the job.

As per this analogy we find that in the general market for products and services, we find that those that offer products of services are in the position of job applicants, vying for the attention of their prospective customers. The customers act out the role of the hiring managers, using their own criteria to determine which offering best suits their needs.

This perspective on the product selection process has a few implications. First of all, for companies, the traditional framing of the competition as products of the same category becomes questionable [40]. This is because for consumers, the search for the optimal solution for a Job is not limited to a certain product category. The consumer wants to get a Job done, and does not necessarily care what product he or she uses to achieve this. As a result, in Job-based market segmentation markets are defined on the Jobs that customers hire products for, instead of on product categories or demographic characteristics [9].

A second implication is that the demographic and psychographic segmentation techniques that many companies use to help shape their product strategy, could be misleading [40]. This is because these demographic and psychographic characteristics that are for instance encapsulated in personas, can only be used to target correlations in data.

Such techniques offer no explanation of the causal forces that drive the customer to behave in a certain way. We explore this issue further in section 4.3.5 on personas.

The third implication - and this is one that is especially relevant in the software industry - is related to the metrics that are generated when customer use a product. Making decisions based on these metrics is dangerous because while usage metrics can indicate differences in the popularity of different functionalities, they offer no explanation as to why that is the case. Of course, one can make educated guesses and assumptions as to what the explanation might be, but getting it wrong will most likely lead to missed opportunities. A practical example is provided by Intercom, who when trying to improve their popular new "map" feature, discovered that their customers did not want a map in the traditional sense of the word. Instead, they were hiring the map to help make their companies website or booth at a conference look impressive. Therefore improving the map meant making it worse from a cartographic perspective, but look more impressive and easier to share [5].

Now that we have discussed the concepts and notions that are at the common core of Jobs-to-be-Done, it is time to discuss the different ways in which they can be put into practice. This is where we will notice the chasm between proponents of the qualitative- versus quantitative approach. We will first discuss the qualitative side of the argument.

3.3 Qualitative Approach

A major proponent of Jobs-to-be-Done is the famous Harvard professor Dr. Clayton Christensen, who has adopted Jobs-to-be-Done as an extension to his theory of Disruptive Innovation [40, 41]. Building on the ideas of 'Creative Destruction' [28], disruptive innovation [41] and now Jobs-to-be-Done [40], share a common element: the importance of data in the innovation process is de-empathized.

Christensen argues that by focusing on the available data, companies risk losing track of the information related to the Jobs that customers are hiring their product for. According to Christensen - and Jobs-to-be-Done in general - this is the case because customer behavior is not driven only by the functional characteristics of a product. Customers hire products for Jobs that revolve around struggling moments in their lives, which can have powerful social and emotional dimensions [9]. This is not data that can be easily generated by a company and then analyzed in a spreadsheet. Therefore, Christensen - and others who follow his line of thought - can be regarded as proponents of a "qualitative approach" to Jobs-to-be-Done.

Being an influential thought-leader, Christensen has inspired others to work with- and write about Jobs-to-be-Done. Notably there is Alan Klement, who focuses on providing a practitioners perspective on the theory. In his contributions to the theory, Klement further emphasizes the importance of the emotional aspects of a Job. He argues that understanding the emotional journey that led to the customer "hiring" or "firing" a product is the way to uncover the Job that the product is doing for that customer. Because of this, Klement argues that all Jobs should be emotional instead of functional [10].

In his work - that is summarized in the book "When Coffee and Kale Compete" [10], Klement further explores the practical application of Jobs-to-be-Done. Most notable his philosophy on how to discover the Job(s) that a product is hired for. Klement makes the case that customers are not able to accurately explain what they want.

Even if they could, Klement argues that these stated needs are not static, but - due to certain psychological phenomena will change over time and are susceptible to outside influences [42]. Therefore, Klement points out that the only type of preference that is worth studying, is the preference that is revealed through customer behavior [10].

Because of this focus on revealed preference, those that follow the qualitative approach rely on performing interviews with customers that recently either stopped using a product or adopted a new product [10]. These "Jobs-to-be-Done interviews" or "Switch interviews" follow in the footsteps of 'Customer Case Research' (CCR), which is an exploratory research method aimed at discovering purchase drivers [43]. In other words, the goal of the interviews is to understand the journey the customer had to undertake in order to make the commitment to a product [44]. Questions that a researcher would ask in this situation include for instance: *"what started you on the road to making this purchase?"*, and *"when and how did you decide the price for this product was acceptable?"*[45]. Within JTBD theory there are various artifacts that relate to the interviews. An example of this are four forces that drive customer behavior, as described in the book: "Intercom on Jobs-to-be-Done" and included below in figure 3.1.

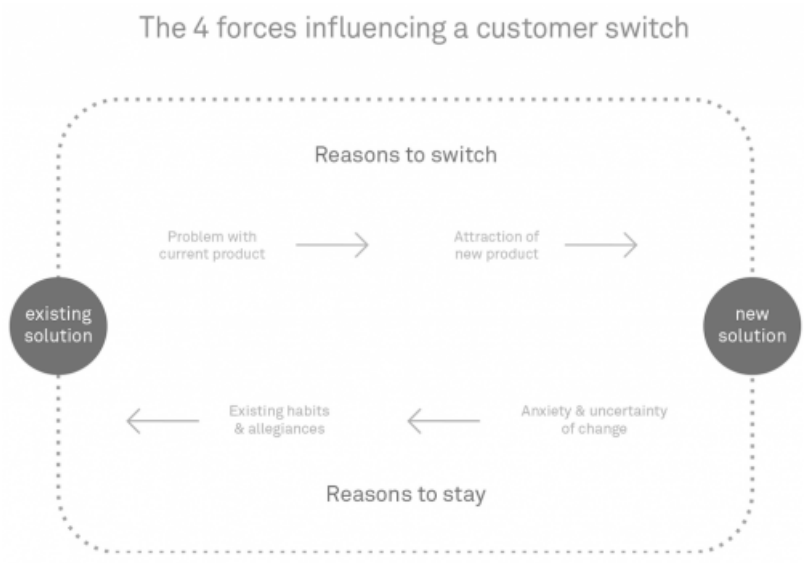

FIGURE 3.1: The four forces that influence customer behavior [5]

In the qualitative process, the results of the interviews are studied and used to formulate the core Job or Jobs that customers are hiring a product for [10].

3.3.1 Job Stories

To contextualize these high-level Jobs, Alan Klement and the software company Intercom invented the concept of Job Stories [5]. Following a similar format as User Stories, the two techniques are closely related. While a more detailed comparison of the two types of stories can be found in section 4.3.2, in this section we focus on Job Stories.

Job Stories are composed of three parts: (i) the triggering event or situation in which a problem arises, (ii) the motivation and goal to make a change to the situation, and (iii) the resulting expected outcome that improves the situation [10]. Below, some examples are included to illustrate the format.

JS1 - **When** I am looking for a new task on the task board, **I want** to filter all tasks to only see ones associated with my skills, **so I can** see open tasks that I am able to complete.

JS2 - **When** an item does not have an estimate or has an estimate I'm not happy with, **I want** to be able to restart the estimation process and notify everyone, **so that** the team knows a particular item needs to be estimated upon.

JS3 - **When** my friend tags me in an unflattering photo, **I want** my Facebook work colleagues to not see me in that photo, **so that** I can maintain my professional reputation.

We attempt to build solid foundations for this RE artifact by performing a theory building exercise [46] resulting in a conceptual model of job stories. To do so, we conducted a thorough analysis of 131 Job Stories from publicly accessible sources and constructed a conceptual model of a valid Job Story as shown in Figure 3.2. In the following subsections, we elaborate on the decomposition of each part and report on the quality of the collected Job Stories by analyzing how well the collected Job Stories fulfill each part. Our analysis document is available online: https://goo.gl/2TUFhw.

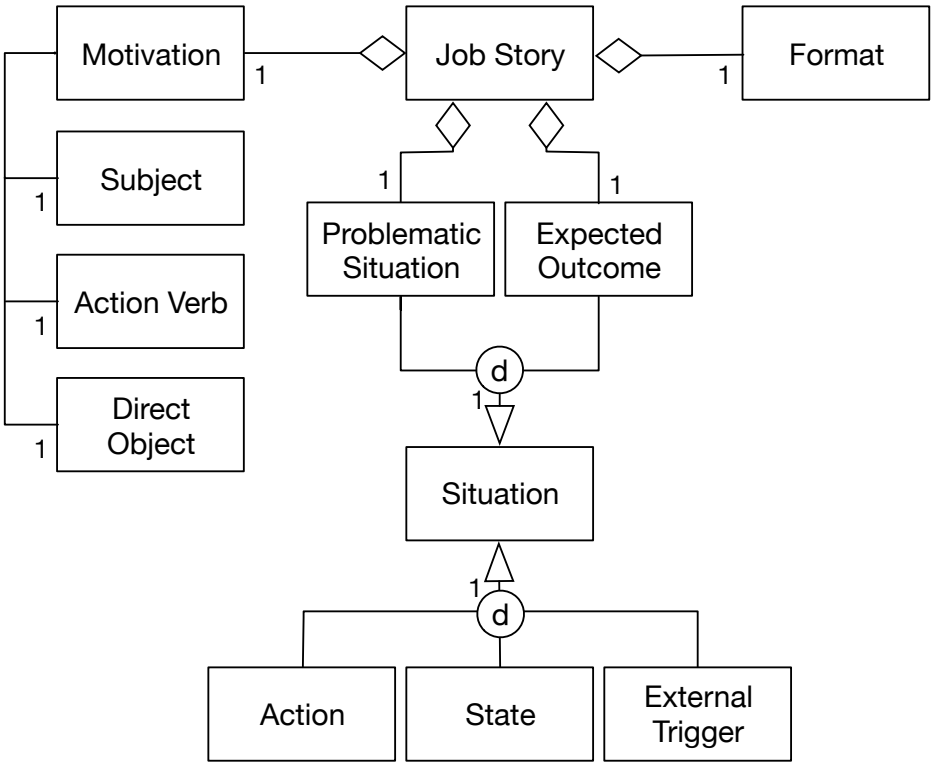

FIGURE 3.2: Conceptual model of Job Stories

3.3.1.1 Template

A Job Story should roughly follow the Job Story template introduced in section 3.3.1. For example, the I can be replaced with a different stakeholder or omitted entirely. This template is the syntactic structure within which the situation, motivation and expected outcome are interspersed.

Analysis: 113 of the 131 Job Stories adhere to the template. Out of the non-compliant stories, 6 added a role before the situation, 6 did not include an expected outcome, 4 did not use a template at all, 2 did not incorporate a motivation indicator.

3.3.1.2 Problematic Situation

The first part of a Job Story provides the contextual starting point for the reader to understand why the actor has a motivation and a goal. It should describe a situation which confronts the actor with a concrete problem [10]. This may be a generic need such as wanting something to eat, or a specific problem in the context of a product such as JS2. In our Job Story analysis, we identified three variants of a well-formed situation that correspond to one example Job Story each:

- **Action.** The problem lies in the action that an actor is executing as part of her job. This case is exemplified by JS1.
- **State.** An actor or an object may be in a problematic situation (state) because of their attributes. This can take the form of an object property such as in JS2, but can also be an human emotional state such as being bored or unfamiliar with something.
- **External event.** The problematic situation is experienced because of an external event, i.e., an action conducted by some other actor. Exemplified by JS3.

Analysis: Analyzing the 113 Job Stories that adhere to the Job Story template, two things stand out. First, most Job Stories (65) capture the situation as an action, 40 use a state and just 8 include an external event. Note that as an action or external event completes, the result is also a state. The difference between the variants is thus solely in the writing style. Second, only 36 Job Stories define a *problematic* situation. The other 77 Job Stories only describe a situation such as: "When I'm adding or deleting users" or "When I have multiple workspaces". While the original articles on Job Stories do not require the situation to be problematic [5, 6], follow-up blogs stress that a helpful situation defines a concrete struggle and includes as much context as possible [10, 19, 47].

3.3.1.3 Motivation

The second part of a Job Story is its central element of motivation. Capturing the change that needs to occur in order to reach the expected outcome, the motivation generally already implies a solution to alleviate the problematic situation. Theoretically, motivations can have different structures, for they can be used to represent different types of requirements. The majority of motivations, however, adhere to the same basic grammatical structure as the means part of a user story [48]: (1) they contain a subject with an intent verb such as "want" or "am able", (2) followed by an action verb that expresses the action related to the problem to resolve, and (3) a direct object on which

the subject of (1) executes the action. The motivations of JS1-3 adhere to the structure of a user story's means: <JS#,Subject+Intent,Verb,Direct Object> - <JS1, I want to, filter, tasks>, <JS2, I want to, restart, estimation process>, <JS3, I want, see, me>. Aside from these three parts, motivations are free form text and may contain any other linguistic construct such as adjectives and indirect objects.

Analysis: All 113 valid Job Stories adhere to the grammatical structure above to a greater or lesser extent: 83 Job Stories follow the structure exactly, including the *I* as the subject, 20 adhere to the user story structure but incorporate another actor as the subject as in "customers want to", and the remaining 10 do not include an action verb, but readers infer *to have* from the text structure like in "I want a filled in example".

3.3.1.4 Expected Outcome

While the first part of a Job Story presents the problematic as-is situation, the *expected outcome* sketches the desired to-be situation that the actor wants to achieve by satisfying the motivation. In this way, the expected outcome conveys additional context that is necessary to satisfy the motivation in the *right* way.

As the expected outcome also describes a situation, we encountered the same three possible variants in our Job Story analysis. Practitioners may define the expected outcome as: (1) an action the actor can now conduct as in JS1, (2) a desired state of an actor or non-sentient object such as *"so that proprietary intellectual property is secure"*, or (3) an external event for a different actor to conduct an action like JS2. The only difference is that the variants are often expressed in the negative form as in "so that there is no ambiguity" or "so that I don't have to refresh".

Analysis: The dominance of the action variety is less overwhelming with 51 Job Stories capturing the expected outcome with a new or improved action, while 42 define a state an 20 Job Stories refer to events.

3.4 Quantitative Approach

The main precursor to Jobs-to-be-Done can be considered to be the Outcome-Driven Innovation (ODI) methodology introduced by Anthony Ulwick in 1996. Making a case against the 'pivot and fail fast' mindset that is prevalent in Silicon Valley, ODI stresses the importance of identifying the unmet needs in a market before entering it with a product or service [49]. To do this, ODI emphasizes the use of data and quantitative analysis to identify opportunities to innovate and help customers get Jobs done better.

In this, ODI focuses on understanding the customers' *Desired Outcome*, which represents what they want to achieve, instead of giving customers what they think they want [16].

With his consultancy firm Stategyn, Ulwick has been applying ODI for over 25 years with a claimed success ratio of 85%, and is a pioneer in the field of Jobs-to-be-Done [50]. A major reason practitioners prefer to use ODI over other, more qualitative oriented techniques, is the use of data [51]. It is argued that only through data, it is possible to effectively structure to- and measure of a Job [52]. However, there are other important differences, such as the importance of emotional aspects of the Job, how causality is determined, and the artifacts used to analyze a Job.

First of all, instead of focusing on the emotional aspects of Jobs, in ODI, functional tasks are at the center of a Job. Ulwick does acknowledge the importance of the emotional dimension. However, he argues that this is mainly the case products that are very simple from a functional perspective, such as cosmetics or perfume, that are often purchased to get emotional Jobs done. However, for products that are complex from a functional perspective, such as electronics or software, the prime focus should be on function as these products have limited emotional appeal [17].

A second issue that distinguishes Ulwick from the likes of Christensen and Klement, is the issue of causality. As discussed earlier, all agree that causality is important, and that correlations in demographic and psychographic data are no replacement for causality [15]. However, Ulwick argues that there is other data that can still be used to determine causality [53].

Another difference is that while Klement argues that customers cannot tell you what they want [42], Ulwick makes the point that companies go about listening to customers all wrong. Customers should not be trusted to come up with new solutions, they should only be asked what a new product or service should do for them [15]. This is reflected in the ODI process, that - like the "qualitative" approach - revolves around customer interviews. However, instead of following the philosophy of CCR, ODI interviews are designed to explore a functional task, and uncover how customers measure success at each step of getting the Job done [54].

Ulwick argues that each Job follows roughly the same structure, therefore, the process of the activities related to getting a Job done can be represented in a uniform way using an artifact called a 'Job Map' [16]. As such, 'Job Mapping' involves exploring the customers' perspective of getting a Job done, not through focusing on the steps the customer is taking to get the Job done, but rather on what the customer is trying to achieve at each step [18]. The latter is described using the previously mentioned concept of *'Desired Outcomes'*, that describe what success means to a customer at each step of a

Job Map. Desired Outcomes are statements that describe a direction of improvement, a unit of measure, and a object of control with contextual clarification and examples [16]. An example of a Desired Outcome statement is as follows:

> "Minimize the time it takes to verify the accuracy of a desired outcome with a customer, e.g., its meaning, completeness, exactness, etc." [16].

To determine which desired Outcomes offer the largest opportunity for innovation and commercial success, Ulwick argues that the best opportunities come from the desired outcomes that are important to customers, but are not satisfied by existing products and services [15]. This is determined by asking customers in a survey to rate each outcome on a scale of 1 to 10 on the importance of each outcome and the satisfaction with current solutions. Using the ODI "Opportunity Algorithm", these metrics can be used to discover the most promising areas. The algorithm is simple, and looks like this:

$$[\text{Importance} + (\text{Importance} - \text{Satisfaction}) = \text{Opportunity}]$$

Using these ratings for Importance and Satisfaction, ODI suggests to plot the Desired Outcomes in an 'Opportunity Prioritization' graph. In this graph the y-axis indicates the satisfaction with current solutions, and the x-axis indicates the importance [55, 56]. Figure 3.3 included below, is an example of such a Opportunity Prioritization graph.

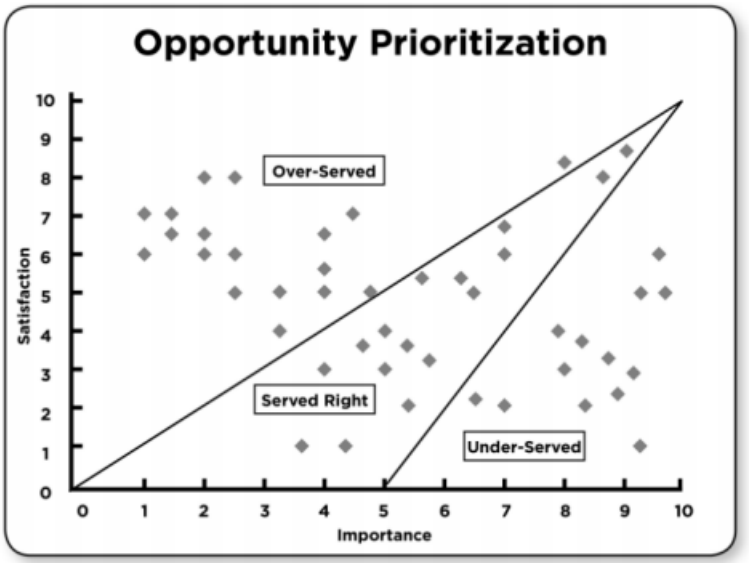

FIGURE 3.3: Example of ODI Opportunity Prioritization graph [56]

As illustrated in the example, an 'Opportunity Graph' can be used to identify which Desired Outcomes are probably over-served, served right or under-served. Ulwick and

Hamilton lay out some innovation strategies for targeting these different types of needs [57]:

1. A new solution that gets the Job done worse, but is cheaper will only appeal to customers who are over-served by existing solutions.
2. A new solution that gets the Job done better, but is also more expensive will only appeal to customers who are under-served by existing solutions.
3. A new solution that gets the Job done better, and is cheaper will appeal to all customers.

3.5 Conclusion

In this part of the literature review we explored Jobs-to-be-Done theory in an effort to answer the first four sub-research questions.

To answer the first sub-question: *'What are the principles of Jobs-to-be-Done theory?'*, we discussed the background of Jobs-to-be-Done, and the theoretical notions that constitute it. What we found was a fragmented body of literature, that yielded no clear consensus on what exactly constitutes Jobs-to-be-Done. In fact, Jobs-to-be-Done can be defined as: "a collection of principles that helps you discover and understand the interactions between customers, their motivations, and the products they use" [10]. However, a commonly accepted core principle of Jobs-to-be-Done is that instead of wanting to buy products, customers are looking for the best solution for a Job they need to get done in their lives. As a result, markets should not be segmented by product category, but by the Jobs they are hired for. The same logic applies to customer segmentation, customers should not be segmented by their demographic or psychographic characteristics, but by the Jobs they need to get done [40].

For the second sub-question - *'What are the main concepts and artifacts related to Jobs-to-be-Done?'* - we investigated the main concepts and artifacts related to Jobs-to-be-Done. We found that there are many different concepts belonging to the different approaches to Jobs-to-be-Done. For starters, depending on the preferred approach, *Job Statements* can be used to represent Jobs at different levels of granularity and from various perspectives, such as emotional or functional. To unpack a Job and describe the different aspects that make it important, practitioners commonly use *Job Stories* or *Desired Outcomes.* Job Stories are a variant to User Stories, that emphasize the motivational and situational context that drive customer behavior [10]. Furthermore, a *Job Map* is used to illustrate the Job from the customers perspective, by breaking it

down into discrete *Job Steps*, and an *Opportunity Graph* can be used to compare needs in the market based on how customers rate their importance and satisfaction with current solutions [54].

In our research we found that there is not a uniform approach to Jobs-to-be-Done. With the third and fourth sub-questions, namely: *'What are the existing approaches to applying Jobs-to-be-Done?'* and *'how do the different approaches to applying Jobs-to-be-Done relate to each other?'*, we investigate this in more detail. In doing so, we found that there are two main "camps" of Jobs-to-be-Done practitioners: those that favor a *'qualitative'* approach as championed by the likes of Christensen and Klement, and those that prefer the *'quantitative'* approach based on Ulwick's ODI methodology.

The qualitative approach de-emphasizes the role of data in the innovation process, and stresses the importance of the emotional dimensions of Jobs [9]. The quantitative approach on the other hand, as represented by Anthony Ulwick's Outcome-Driven Innovation methodology, favors exploring customer needs in the context of a core functional Job, and prioritizing these needs using quantitative techniques [16].

Chapter 4

Jobs-to-be-Done in the Software Industry

4.1 Application in Practice

In this section we will discuss different examples of Jobs-to-be-Done, and Job Stories in particular, in the context of software products. This is interesting because the literature on Jobs-to-be-Done mostly caters towards physical products. Simple physical products such as milkshakes are often used to to help explain the theory. However, applying a theory that was designed with physical products in mind on usually much more complex software products is not a straightforward process. Still, practitioners in the software industry have done just that. However, concrete and fleshed out examples of how they did this are few and far between, and the examples that are available do not incorporate an holistic perspective on the theory. Therefore, the aim of this section is to provide some insights in the different ways that Jobs-to-be-Done can be applied in the context of software products, and derive some conclusions with regard to the application in the case study.

4.1.1 Guiding IT Procurement

We will start this section with an example that offers a different and interesting perspective of the use of Jobs-to-be-Done. The example is provided by some American health care institutes, who used the theory to help them determine what IT systems to acquire, and how to organize their IT landscape [58]. Specifically there is the case of Bellin Health, a hospital that needed to invest in a new information and analytics platform that was to be used by different types of stakeholders. To determine the most

appropriate IT systems for each type of stakeholder, the Jobs-to-be-Done framework was used to identify and represent their high-level needs. In this context, a Job represented a set of problems that had to be solved by the new IT systems [59].

4.1.2 Software Product Development

Moving on to examples related to the development of software products, we notice that there are various different ways in which different parts of the theory are applied and often combined.

First of all there is Intercom, the previously mentioned company that has adopted Jobs-to-be-Done to guide many of their practices from marketing to product development, and has worked with Alan Klement to invent Job Stories [5]. Their connection with Klement indicates that Intercom subscribes to the qualitative approach to Jobs-to-be-Done. This is reflected in the unstructured way they apply Jobs-to-be-Done principles to help them design features for their CRM product. For instance, they describe the use of Jobs-to-be-Done to eliminate meaningless steps in a workflow. Furthermore, when improving features a Jobs-to-be-Done mindset is used to try and focus only on how the feature is used and the Job that it is hired for. Job Stories have been integrated into the work processes at Intercom. Before the start of each new project, a project brief document needs to be created. This is a single page document that uses Job Stories to help remind the team of the problem or opportunity that is being tackle [5].

A similar case as the one from Intercom, is provided by Thoughtbot, a software development firm. Fiedler, describes in little detail how Thoughtbot has moved away from User Stories towards Job Stories [60]. The transition was made in an effort to steer the development team away from assumptions about personas, and to promote empathy for the user. While providing little argumentation, Fiedler states that switching to Job Stories has proved beneficial for Thoughtbot.

Another example is provided by Tessmann, who proposes a way to use Job Stories to explore and describe work-flows, which can then be used as a source for UI needs in the design for a system [61]. In short, the proposed process involves focusing on the different expected outcomes that are associated with each work-flow. Then, Job Stories can be used to describe each step of incremental progress needed to achieve the different expected outcomes.

Next, we come to the work of Bloemberg, who - in his Bachelor thesis - describes the use of Job Stories to document requirements for a software system, based on workshops with different groups of stakeholders [62].

A final example to be discussed in this section is provided by Carpenter, who proposes a high level method for eliciting and prioritizing customers' Jobs [63]. This method is interesting because it combines elements from both the qualitative- and quantitative approach to Jobs-to-be-Done, and is also geared towards incremental innovation. The method relies on interviews with customers in which different topics are discussed from a functional perspective. Different Jobs will emerge from the discussion, which are then described using Job Stories. In turn, the Job Stories should be prioritized based on the customers' satisfaction with current solutions, and perceived importance of the Job. Because of this, Carpenter's method is essentially a scaled down version of ODI, that uses Job Stories instead of Desired Outcome statements.

4.2 Main Usage Scenarios

In this section we will reflect on the examples provided in the previous section, and see if we can identify distinctly different usage scenarios.

From the examples it becomes apparent that the theory is applied in many different ways. Practitioners use it freely, in ways that fit their needs, mostly to facilitate discussions and promote understanding and empathy for the users. When practitioners write about their experiences they usually only define the concepts and discuss a vague way of using them that is relevant from their viewpoint. As a result, the distinction between the qualitative- and quantitative approach, that seems so definite in theory, is not really reflected by the examples from practice.

Because of this, and because of the general lack of well-described examples from practice, it is very difficult to identify main usage scenarios. Therefore, to still be able to find an answer to the sub-question, we will provide a theoretical perspective on what parts of the theory are best suited for different usage scenario.

On a high level, we can identify three main usage scenarios, namely: improving existing products, designing new products, and promoting discussions and customer empathy among a team.

4.2.1 Improving Existing Products

In this situation a company has an existing product and customer base, that it wants to improve. Both the qualitative- and the quantitative approach, or a combination of the two, can be used in this scenario.

In general, applying Jobs-to-be-Done in this scenario will revolve around the idea that improvements to a product should be made based on an understanding of the job(s) that customers hire it to do. The focus should not be on what new features to implement, but on what can be changed to improve how customers are helped in their struggle [9]. The choice between the quantitative and qualitative approach, would have to depend on the amount of time and resources available, where the qualitative approach should be more lightweight than ODI. Additionally, due to the looser structure, the qualitative approach could be assumed to be more suited for radical innovation, since it is more likely to lead out of the beaten path. The quantitative approach on the other hand, has the rigor and focus that is seemingly better suited to incremental innovation.

4.2.2 Designing New Products

In this situation a company is designing a product that is entirely new for them, and that they do not have a customer base for. This hinders all approaches to Jobs-to-be-Done, making it more difficult to source the jobs, due to there being no east access to existing customers to interview. Still, a workaround would be to interview (ex-)customers from competitors, and develop a solution that better addresses the relevant jobs.

Again, both approaches, or a combination, could be applied and the trade-off between them is similar to the previously described usage scenario. The idea of a combination is very interesting as one could combine the strengths of the different approaches, using the quantitative approach for the functional dimension, and enriching this with additional emotional contextual information using the qualitative approach. To some degree, Carpenter - as discussed earlier - describes such a combination by combining an ODI like approach with Job Stories [63].

4.2.3 Promoting Customer Empathy and Discussions Among a Team

The third scenario that we can identify, is to promote customer empathy and discussions for entire product development teams. In this sense the goal is to have all team members be aware of the Jobs that are being addressed, and to have these insights guide design decisions.

For this purpose, the qualitative approach seems to be the most appropriate. This is because it is relatively lightweight to implement, and is specifically focused on connecting with the customer on an emotional level. Judged by their experience reports, the approach described by Intercom seems to be effective. Intercoms approach revolves around

a project brief document that uses Job Stories to describe the Jobs that are addressed for the customer.

4.3 Existing Techniques in the Software Industry

This project explores the use of Job Stories in the context of RE for software products. To get a sense of the added value it provides, it is important to study related techniques that are used in this context, and evaluate how they relate to Job Stories and Jobs-to-be-Done as a whole. The techniques that are discussed in this section differ in terms of scope, ranging from complete design methodologies to artifacts used in the RE processes. In this section, the following techniques are discussed: User-Centered Design (UCD), scenarios, Use Cases, User Stories, personas, and goal-oriented requirements engineering. For each technique, the underlying rationale, its strengths and weaknesses, as well as its relation with Jobs-to-be-Done and Job Stories is discussed.

4.3.1 User-Centered Design

A key principle from Jobs-to-be-Done is that it is necessary to have an understanding of the struggles that a customer experiences in their lives, in order to be able to design a solution that truly helps the customer [9]. This focus on putting the user first when designing software, is shared by a set of practices known as User-Centered Design [64].

There is no agreed upon definition for UCD and therefore it is being applied in a variety of ways [65]. However, Draper & Norman argue that the process of applying UCD involves focusing usability throughout the entire development process and further throughout the system life cycle [66].

Endsley and Jones offer three principles that focus on the role of technology in UCD [67]. The first principle is that technology should be organized around the user's goals, tasks, and abilities. The second principle dictates that technology should be organized around the way users process information and make decisions. The third and final principle emphasizes that technology must keep the user in control and aware of the state of the system.

An example of how UCD can be applied in practice is described by Hussain, et al. [68]. While designing a web based system, they evaluated the usability of the application in small iterative tests. The evaluation process revolved around the creation of prototypes of the user-interface throughout the design process. These prototypes were then evaluated by usability engineers and test-users.

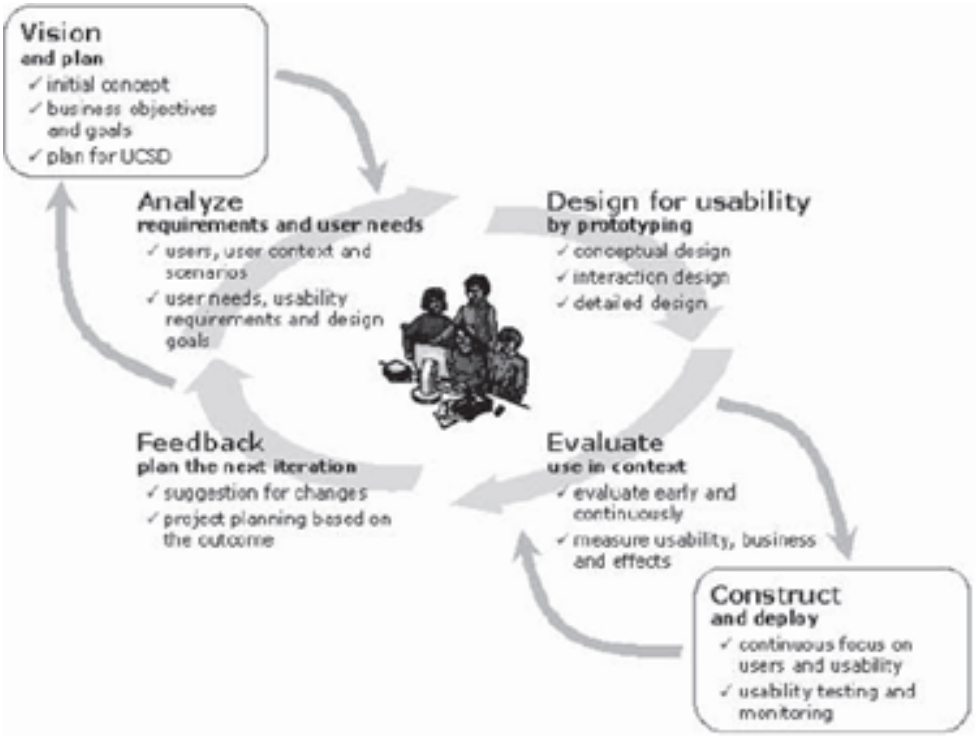

FIGURE 4.1: Process for User-Centered Systems Design [66]

Reflecting on the principles of UCD, it seems that organizing the software system around the user's goals, tasks, and abilities certainly sounds very similar to the philosophy of understanding and designing for the Job that customers are trying to do [13].

Another commonality between the two, is that both Jobs-to-be-Done and UCD typically involve the direct observation of users [64]. Due to their similarities, one might view Jobs-to-be-Done and UCD as complementary. Figure 4.1 taken from Draper & Norman [66] illustrates this quite nicely. The figure shows that UCD is focused on the whole process of software development. Although not explicitly researched, JTBD theory seems to have the potential to be used in a UCD oriented software development process.

4.3.2 User Stories

User Stories are short, natural-language statements that essentially reflect the customer's words. User Stories emerged as a part of the eXtreme Programming (XP) methodology. To keep up with the fast pace prescribed by XP, User Stories were not intended as fully fledged requirements. Instead, they serve as commitments for further conversations between the customer and developers [69].

Describing functionality that will be valuable to either a user or purchaser of a system or software, User Stories are composed of three aspects: a written description of the story, conversations about the story, and acceptance tests that convey what can be used to determine the completion of the story. The written description of the story follows the template: "As a <role>, I want <goal>, [so that <benefit>]", where the part between square brackets is optional [1].

As a part of an incremental, agile, approach to software development, User Stories have to facilitate the process of Release Planning (RP). RP is the process of selecting and assigning features to be developed in order to create a sequence of product releases that satisfies technical, resource, budget, and risk constraints [70].

To facilitate release planning, it is important to be able to gauge which requirements are most important given the previously described technical, resource, risk and budget constraints [71]. User Stories take this need into account by assigning 'story points' to each story. These indicate the size and complexity of the story relative to other stories. Large User Stories that can be split into two or more stories of smaller size, are sometimes referred to as Epics [1]. Ton [72], illustrates the difference between Epics and User Stories by stating that Epics represent a single high-level business objective, while User Stories represent the most granular piece of business value as deemed by the customer. An important practice in RP, is to determine which User Stories are to be included in a release by combining the story points with ratings on the priority of the Epics and User Stories [1].

When the software for a release has been developed, XP dictates that it is checked whether the software actually reflects the users' wishes [73]. This is where the previously mentioned User Story acceptance tests come in. Acceptance tests are specified by the users and delivered to the development team. The software is then developed with these user expectations in mind. After development, it is verified whether the software actually passes the acceptance tests [1].

Cohn [1], has identified the main reasons to use User Stories. First of all, User Stories emphasize verbal communication over written communication. Secondly, User Stories can be understood by both developers, and users or the business. Additionally, User Stories are the right size for planning and are suitable for iterative development.

However, incorporating User Stories in the software development process will not guarantee success. Patel and Ramachandran have identified some problems that can be caused by having users as a source of requirements [74]. An important problem is that users rarely have a general picture of the requirements of system in their mind. In most

Characteristics	**Description**
I - Independent	User Stories should not overlap in concept, and it should be possible to schedule and implement them in any order.
N - Negotiable	User Stories should be co-created by customer and programmer during development.
V - Valuable	A User Story should represent value to the users, not only to the developers.
E - Estimable	It should be possible for the customer to rank and schedule a User Story's implementation.
S - Small	A good User Story typically does not represent more than a few person-weeks' worth of work.
T - Testable	A good User Story is clear enough so that it can be tested.

Table 4.1: INVEST characteristics of good User Stories [75]

cases, different users have different needs. This can lead to problems such as conflicts between requirements, missing requirements, ambiguous requirements, and the neglection of non-functional requirements.

Multiple frameworks have been proposed to help create User Stories that are not affected by these issues. Most popular is the framework proposed by Bill Wake [75], known under the acronym "INVEST", which is described below in table 4.1. There is little scientific evidence of the effectiveness of User Stories. However, practitioners do agree that User Stories can improve their productivity and quality of their work deliverables, when applied following the template described in the introduction, and quality guidelines such as INVEST [4].

Since Job Stories were created – at least in part – as a response to perceived flaws in User Stories [6], it is interesting to investigate how the two concepts relate. While very similar, Mike Cohn, a prominent author on User Stories, illustrates that they can still be very different: "As a technique to represent what a software system is supposed to do, User Stories are argued to be superficially the same as alternative solutions: requirement statements, use cases and scenarios [and now Job Stories]. However, the large amount of alternatives suggests, that the small differences between the solutions are in reality very important. The variation consists of what is written down, and when" [1].

To make a further comparison between User Stories and Job Stories, table 4.2 included below, contains a set of User Stories linked to their Job Story counterparts. Admittedly, it should be noted that the User Stories in the example are not necessarily representative of User Stories in general. However, it does illustrate what supporters of Job Stories see as the problem with the format of User Stories.

Nr.	Type	Story
1	User Story	**As** a moderator, **I want** to create a new game by entering a name and an optional description, **so that** I can start inviting estimators.
	Job Story	**When** I'm ready to have estimators bid on my game, **I want to** create a game in a format estimators can understand, **so that** the estimators can find my game and know what they are about to bid on.
2	User Story	**As** an estimator, **I want** to see the item we're estimating, **so that** I know what I'm giving an estimate for.
	Job Story	**When** I find an item I want to set an estimate for, **I want to** be able to see it, **so that** I can confirm that the item I'm estimating is actually the correct one.
3	User Story	**As** a moderator **I want** to select an item to be estimated or re-estimated, **so that** the team sees the item and can estimate it.
	Job Story	**When** an item does not have an estimate or has an estimate I'm not happy with, **I want to** be able to restart the estimation process and notify everyone, **so that** the team knows a particular item needs to be estimated upon.

TABLE 4.2: User Stories compared to Job Stories, adapted from "Replacing The User Story With The Job Story" [6]

Trivial or not, it is apparent that there are some differences between the User Stories and Job Stories in table 4.2. For starters, we see that in the transition from User Story to Job Story the personas are omitted, and replaced with contextual information on the situation that triggered the story.

Klement, argues that the personas do not add anything to the stories [6]. However, this is not something all practitioners agree on, as can be gleamed from the reactions to his article. It is not clear in the framing of the example, but it could for example be that 'moderator' and 'estimator' are security profiles and therefore do add meaningful information.

A clear difference between the two types of stories is that instead of the personas, Job Stories address an issue that is not represented in User Stories: "The customer goes about life as usual, and then a problem arises. What is the trigger or situation?" [10].

As mentioned before, Job Stories should be more independent from solutions than User Stories. This is reflected in the example. For instance, in the first User Story a moderator wants to start inviting estimators. The moderator suggests what needs to be done to achieve this goal: he should create a new game by entering a name and an optional description. A solution is specified but the User Story does not provide insight into why this is the best solution.

Klement, makes the argument that the Job Stories in table 4.2 contain a lot of information that is missing in the User Stories [6]. For instance, in what situation is it necessary to start inviting estimators? And, why is the solution suggested to create a new game by entering a name and an optional description? The latter question is especially interesting since it shows how causality is missing in the User Story.

The Job Story in the example provides insight in what is really important: a game should be created in some format estimators can understand. Now, the assumption made in the User Story that this format is a name and an optional description might very well be valid. However, it is not unthinkable that estimators might be able to understand the game better if the format also included information on, for instance, the type of game.

An issue to address here is that – apart from the issues related to personas and contextual information about the situation - the previously described differences between Job Stories and User Stories are relatively superficial. In their essence, the two types of stories are very similar. Therefore, a User Story could be written to address many of the same concerns as Job Stories. For instance, when User Story 1 from table 4.2 would be written from a Jobs-to-be-Done perspective, it would look something like this:

"**As a** moderator, **I want to** create a game in some format estimators can understand, **so that** estimators can find my game and know what they are about to bid on."

What we see here, is a User Story that includes motivational elements in its <goal> section, and a more elaborated expected outcome in its <benefit> section. Some practitioners argue that since User Stories can be written in this way, the need for Job Stories to exist is somewhat equivocal [76]. Another argument downplaying the importance of the differences between User Stories and Job Stories, is that the format of the story is not as important as the processes and interactions around it [77]. When Intercom introduced the concept of Job Stories, they did more than change the format of their stories, they also changed their processes and adopted a Jobs-to-be-Done mindset [77].

This brings us to a conclusion on the value of Job Stories and its relation to User Stories. When only looking at the formats, there is no apparent winner. Instead, let's

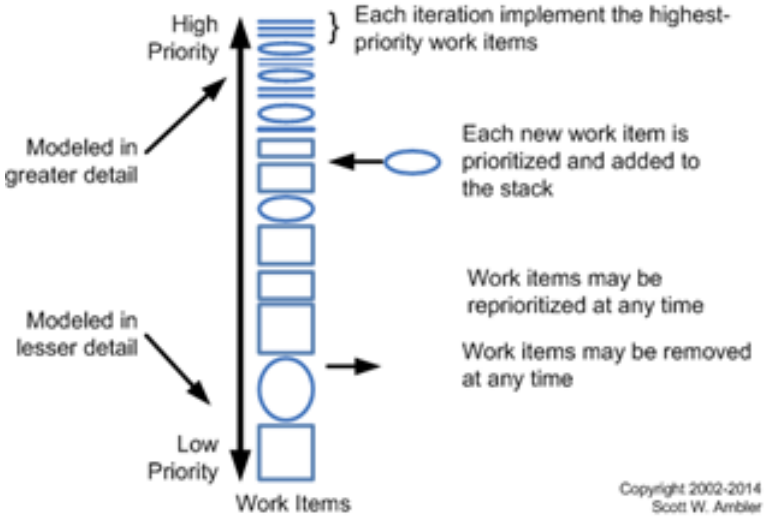

FIGURE 4.2: Disciplined agile change management process [79]

apply Jobs-to-be-Done for ourselves and consider the Job that Job Stories are doing in the broader Jobs-to-be-Done theory. From this perspective, Job Stories exist for the job of expressing Jobs-to-be-Done thinking in a format that is actionable in the context of software products. Job Stories exist because they are different enough from User Stories to be a better candidate for this job.

Following this line of thinking we should also consider the jobs that User Stories might be more suitable for. In fact, part of the appeal of User Stories is the close link with agile software development methodologies, as is illustrated by the widespread adoption by practitioners who employ those methodologies [3]. Under the condition that proper care is taken to ensure their quality, these practitioners recognize the value of User Stories [4].

This compatibility with agile processes has led to the creation of additional techniques and processes that involve them. An example of this is the technique 'planning poker', which can be used to improve User Story estimation performance [78]. An example of an agile process that revolves in part around User Stories, is the disciplined agile change management process, as proposed by Ambler & Lines [79], and illustrated in figure 4.2.

From this perspective, the value of User Stories is broader than the format of the story itself, it is also defined by the different ways practitioners implement them in their processes as a means to design better systems. Because of the similarities between Job Stories and User Stories, these lessons learned on how to effectively implement User Stories should not be forgotten, as they might very well also apply to Job Stories.

4.3.3 Use Cases

After discussing User Stories, it is natural to turn our attention to Use Cases. Since User Stories and Use Cases are closely related, it is interesting to investigate the relationship between Job Stories and Use Cases.

The close relationship between User Stories and Use Cases is illustrated by Imaz & Benyon [80], who argue that User Stories are the first artifacts used to describe interactions. For implementation purposes, something more formal is needed such as Use Cases. According to Bittner [81], Use Cases describe sequences of events that, taken together, lead to a system doing something useful. He argues that Use Cases are a powerful way to express the behavior of a system in a way that all stakeholders can understand.

An important concept related to Use Cases are actors. These are specific roles plated by a system user that represents similar behavior when using the system. Actors can be human beings, and external systems or devices that communicate with the system. In this sense, a use case is a typical system usage scenario for a specific actor [82]. Since there can be many different Use Cases for a system, a Use Case Diagram is used to show the interactions between actors outside the system and the different Use Cases inside the system [83]. An example of an UML Use Case Diagram is illustrated in figure 4.3 [84]. To conclude this section on Use Cases it is important to mention another argument provided by Imaz & Benyon [80]. They state that User Stories are not suitable for direct use in the implementation of a system. User Stories do not provide the gradual transformation from informal to semi-formal information that is necessary to get usable models in implementation. Use Cases on the other hand, can provide this semi-formal middle ground and are therefore more suitable for implementation. However, while this might be a valid point in theory, in practice User Stories are often used during the implementation of software systems.

4.3.4 Scenarios

On a high level, Carroll [85] defines scenarios as narratives describing what people do when engaged in particular activities. These narratives help ground the design in an understanding of the interrelation of tasks users carry out over time, and the capture and sharing of this information for design.

In this sense, scenarios are often used to help understand the interface between the environment and the system, as well as a means of eliciting and specifying software behavior [86]. Additionally, Potts argues that scenarios have some characteristic elements:

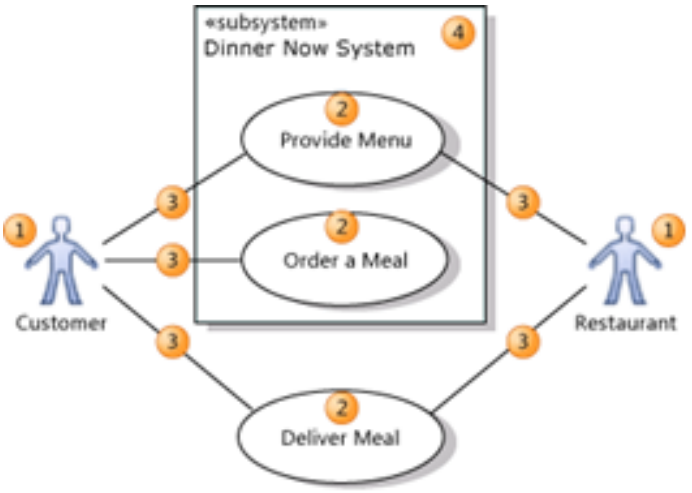

FIGURE 4.3: UML Use Case Diagram example [84]

there is a setting, there are actors with goals, and there is a plot shaped by sequences of actions and events [87].

This focus on the link between the environment and the system helps explain the close relationship between Scenarios and Use Cases. Scenarios are used to represent different paths of possible behavior through a Use Case, these paths are investigated to elaborate requirements [88].

Also worth mentioning is the link between scenarios and User Stories. This link lies in the close relationship between scenarios and the acceptance criteria for User Stories. While sometimes conflated, scenarios are concrete examples of acceptance criteria [89].

Turning our attention towards Job Stories, we see that the characteristics defined by Potts [87], indicate that there are some similarities between the two concepts. Even though scenarios offer a much more detailed description, both describe the situation an actor is in, and both describe a goal that an actor is trying to achieve.

While this requires further research, an approach where scenarios and Use Cases are constructed based on Job Stories seems feasible. Making this step from Job Stories to scenarios and Use Cases means moving from the *problem space* into the *solution space*.

4.3.5 Personas

Another prominent concept in RE is the persona method. This method creates an imaginary representation of a user, based on information about users' needs, behaviors, and preferences [90]. The persona should be given a name, a face, and enough relevant details to make them seem real to the project members [1].

In his book: "The inmates are running the asylum", Cooper argued that software products should not be designed to appeal to all people [90]. He made the point that it is impossible for a product to be perfect for everyone, since people are different and have different needs. Due to limits in available resources it is impossible to properly account for all these needs in a single product, a product designed for all people risks ending up as a product that is acceptable for all people. To prevent this from happening, Cooper suggests to design a product that is perfect for a specific type of person. Personas exist to facilitate this.

Using the persona method should yield three main advantages: firstly, personas engage teams to think about users; secondly, personas can be used to make design decisions; and thirdly, by not being based on large amount of user data, personas are not susceptible to problems related to dealing with large datasets, such as paralysis or inappropriate generalization [7]. Beside these points, Maness, Miaskiewics, & Sumner [91], state that the essential benefit of personas is that they help build empathy for the target users.

However, in Jobs-to-be-Done theory personas are criticized. Coopers argument that is impossible to design products for all people is brushed aside by proponents of Jobs-to-be-Done, who argue that the focus should not be on people at all. Instead, the focus should be on the Jobs that the people need done [6]. By focusing on the Jobs that different "types" of people have in common, a symptomatic problem with personas can be avoided. This problem that plagues persona-based segmentation is that there are too many typical users to focus on. To make matters worse, these typical users often do not have a lot in common [9].

Another argument against the use of personas that is echoed by supporters of Jobs-to-be-Done is that their validity is impossible to verify. Chapman & Millman [7], argue that this is the case because personas are fictional artifacts, which likely have no determinable relation to real data. Additionally, they present the point that personas cannot be validated using interviews and ethnographic research. This is because specific examples of data cannot be provided to prove their accuracy.

A counterpoint is offered by Faily & Flechais, who propose a methodology that makes it possible to create more grounded personas that can indeed be independently validated [92]. However, in practice most personas will be based on a combination of user studies and the designer's assumptions [93].

To close this section on personas, we can conclude that the value of personas lies in their ability to help build empathy for the target users [91]. However, there are reasons to doubt the validity of the personas' representation of the target users [7]. This makes the

concept of job based segmentation, as proposed in Jobs-to-be-Done theory, an interesting alternative that deserves to be investigated.

4.3.6 Goal-oriented Requirements Engineering

As the name suggests, goal-oriented RE involves the use of goals and related concepts in RE. Axel van Lamsweerde [94], a prominent author in this field, argues that within RE, goals can be used at different levels of abstraction to capture the various objectives the system under consideration should achieve. More specifically, goals are used in goal-oriented RE for eliciting, elaborating, structuring, specifying, analyzing, negotiating, documenting, and modifying requirements.

On a high level, the process of goal-oriented RE has three main steps. First, problems and opportunities are identified through an analysis of the current system. Secondly, high level goals are identified to address the problems, and meet the opportunities. Thirdly, requirements are then elaborated to meet the goals [95].

Van Lamsweerde [95], points out that one of the main advantages of goal-oriented RE is that technical artifacts from the software development process, such as object models and requirements, can be derived systematically from goals.

The strong link between goal-oriented RE and the rest of the software development process is illustrated by the work of Bresciani, Perini, Giorgini, Giunchiglia and Mylopoulos [96], who used concepts from goal-oriented RE to develop Tropos, an agent-oriented software development methodology. What defines Tropos as a software methodology, is that the concept of an agent, and all related mentalistic notions (such as goals and plans), are used in all phases of software development [96].

Another important characteristic of Tropos is that it covers the very early phases of requirements engineering. Here, the aim is to create a deeper understanding of the environment where the software must operate, and the kind of interactions that should occur between software and human agents. This is done by identify and modelling domain stakeholders as social actors, who depend on one another for goals to be achieved.

By focusing on the goals that a system should achieve, goal-oriented RE strives to understand the 'why' aspects behind software. Another perspective on this is offered by Eric Yu and John Mylopoulos [97]. Instead of investigating the relationships between actors and their goals in the context of the use of a system, the research of Yu and Mylopoulos focuses on the (re)design of software processes.

They argue that in order to improve or redesign a process, it is often necessary to have a deep understanding of that process. The authors define this deep understanding as "an

understanding that reveals the 'whys' behind the 'whats' and the 'hows'". A model is presented that captures the motivations, intents, and rationales that underlie a software process and its embedding organization, in terms of dependency relationships between actors.

One of the concepts proposed to help represent the dependency relationships between actors in this context, is an 'Actor Dependency' (AD) model. Interestingly, this model incorporates an issue that is also important in Jobs-to-be-Done theory: people do not always make rational decisions when faced with uncertainty [10]. To account for this issue, the AD model uses concepts from the field of organizational computing.

Reflecting on the field of goal-oriented RE as a whole, it must be noted that it is much more to it than the literature mentioned in this section. Since it is difficult to discern a uniform notion of a goal in RE, the field is also plagued by some ambiguity [98]. Still, when comparing goal-oriented RE to Jobs-to-be-Done, some commonalities and differences become apparent.

For starters, both focus on motivational aspects, but do it in very different ways. A goal represents something an actor wants to get done. While this seems very similar to a Job, the use of the broader term 'actor', instead of 'customer' or 'user', illustrates the difference in focus between goal-oriented RE and Jobs-to-be-Done.

In short, the difference is that while Jobs-to-be-Done is problem focused, goal-oriented RE employs a broader, more solution oriented perspective as it directly relates the problem with the system that is going to solve it. In Jobs-to-be-Done the ultimate goal is to understand the underlying problems that create Jobs. To gain this understanding one should talk to customers. While – as illustrated by Tropos' early phases of RE – goal-oriented RE does deem an understanding of the external environment to be important, it also plays an important role in developing a solution by for instance delivering technical artifacts that fuel the development process.

Still, due to their similarities, goal-oriented techniques like Tropos could be very interesting for further research. Either towards a possible broader role of Jobs-to-be-Done in software development, or towards integrating the two techniques.

4.4 Conclusions of the Literature Review

This part of the literature review was performed to find an answer to the fifth, sixth and seventh sub-question posed in section 2.1.

For the fifth sub-question: *'What are examples of the application of Job Stories and Jobs-to-be-Done in the context of software products?'*, we found that Jobs-to-be-Done is applied in various ways including: determining what IT system to procure; optimizing the workflow for software solutions; as a direct replacement for User Stories; and to document requirements for software systems, based on workshops with users.

The sixth sub-question focuses on identifying main usage scenarios for Job Stories and Jobs-to-be-Done: *'What main usage scenarios can be identified for Job Stories and Jobs-to-be-Done in the context of software products?'*. As discussed in section 4.2, practitioners seem to use Jobs-to-be-Done and Job Stories in three main scenarios: to improve existing products, to design new products, and in a more general way to promote customer empathy among a team.

In the seventh sub-question, we investigate how Job Stories and Jobs-to-be-Done relate to other techniques used in the software industry: *'How are Job Stories and Jobs-to-be-Done positioned in the landscape of requirements engineering for software products?'* This investigation serves two purposes: it helps provide perspective on the use of Jobs-to-be-Done in the software industry, and it helps determine in what ways the value it delivers is unique in comparison with existing techniques.

In section 4.3 it was discussed that while there is some overlap with most techniques, Jobs-to-be-Done does provide a fresh perspective and has its own merits. What sets it apart is the rejection of traditional segmentation methods in favor of job based segmentation. Also differentiating, is that while other techniques tend to be more solution- or system focused, Jobs-to-be-Done focuses on understanding the problems.

These differences in focus indicate that while Jobs-to-be-Done might not be a suitable replacement for most existing techniques, there could be a synergistic relationship between Jobs-to-be-Done and other techniques. While this relationship should be explored in further research, we can hypothesize that Jobs-to-be-Done could be used to gain an understanding of the problem that is being solved, and other techniques are applied to make the step to implementation. However, to make this possible, it is first necessary to address the gap in literature, and collect- and evaluate more explicitly documented methods for applying Jobs-to-be-Done in that context.

This brings us to the Integrated Job method that has been created to address sub-question 8, which is discussed in the next section.

4.5 Integrated Job Story Method

As discussed earlier, there is no universally accepted interpretation of Jobs-to-be-Done theory. As a result different variants of Jobs-to-be-Done oriented methods exists. Two main examples include the Outcome-Driven Innovation methodology proposed by Anthony Ulwick, that uses quantitative techniques to capture and prioritize customers' Desired Outcomes [16], and the more qualitative Job Stories based approach by Alan Klement [6].

As discussed in section 3, despite the common ground between both approaches, supporters of each approach reject many of the fundamental assumptions of the other and proclaim that theirs is the only valid approach to Jobs-to-be-Done. Due to the common ground between them, and due to both approaches being advertised as requirements approaches based on Jobs-to-be-Done, it is hard for practitioners to understand which Jobs-to-be-Done approach and format fits their situation best. As a result, practitioners apply the theory pragmatically, employing arbitrary combinations of the different methods [63, 99].

To answer sub-research question 8: *'What is a method for using Job Stories and Job-to-be-Done to perform requirements engineering for a software product?',* and help practitioners conduct RE via Jobs-to-be-Done, we employ a similarly pragmatic approach and combine the available Jobs-to-be-Done literature into a method. Described using a Process-Deliverable-Diagram (PDD), the method captures both the most important Jobs-to-be-Done and Job Story activities, as well as the utilized artifacts [100]. The resulting Integrated Job Story Method 4.4 has five phases.

P1 - Interview phase: exploratory interviews are conducted with (prospective) customers in order to uncover their goals, their current way of achieving these goals, and the problems that exist in their as-is situation;

P2 - Analysis phase: the workflow, context and motivations of the interviewees are analyzed to formulate initial Jobs and Job Stories;

P3 - Survey phase: in order to validate the results of the analysis phase, a survey is conducted with a larger sample of the target audience;

P4 - Prioritization phase: the survey results are analyzed to determine which Jobs present the largest opportunity for innovation, and;

P5 - Project definition phase: the Jobs and Job Stories for development are selected and a *project brief* is created that can facilitate the follow-up development project.

The Integrated Job Story incorporates the two variants two accommodate the two mainstream approaches to Jobs-to-be-Done that were discussed earlier this section: a *quantitative* approach based on ODI an a *qualitative* approach based on Klement's work. In principle, the Integrated Job Story Method is designed to utilize the structure from the quantitative approach and the concepts of the qualitative approach. However, in our illustration we do elaborate on concepts from the quantitative approach when possible.

Below, activities of the method are discussed in more detail. In chapter 5 we describe the case in which the method is applied, and in chapter 6.1 we report on the actual application.

P1. Interview phase

Perform interviews involves arranging, performing, recording and transcribing interviews with suitable participants. This results in `INTERVIEW TRANSCRIPTIONS` for each interviewee containing information on the `INTERVIEWEE BACKGROUND` and the `WORKFLOW` that describes how they currently try to reach their `GOALS`, in spite of the `PROBLEMS` that make this a struggle.

P2. Analysis phase

Define high-level functional Jobs by analyzing the `GOALS` and `WORKFLOWS` that were collected during the interviews. This results in a number of high-level `FUNCTIONAL JOBS` that the users are trying to get done.

Quantitative approach:

Create Job Maps by breaking down the identified `JOBS` into a series of discrete `JOB STEPS` that you capture in a `JOB MAP`. The goal of Job Mapping is to determine what the Job looks like for the customer, from beginning to end, and if different customers have different approaches to getting the Job done [54].

Create Desired Outcomes. For each Job Step in the Job Maps, the interview transcriptions are reviewed to identify the `DESIRED OUTCOMES`: metrics that document what success means to a customer for each `JOB STEP`. These statements describe a direction of improvement, a unit of measure, and a object of control with contextual clarification and examples [54]. An example in the context of angioplasty balloons is: 'minimize the amount of force that is required to cross the lesion with the balloon' [15].

Qualitative approach:

Create Job Stories for each high-level functional Job that, taken together, satisfy their goals. The resulting `JOB STORIES` should provide context about the user's struggle to get the functional Job done: the situation, motivation and expected outcome. A commonly used structure for a `JOB STORY` is: **When** <situation>, **I want to** <motivation>, **so I can** <expected outcome> [10].

P3. Survey phase

Create survey to validate the output of the analysis phase. The `SURVEY` asks the participants to rate each Desired Outcome or Job Story on two dimensions with a Likert scale: `IMPORTANCE` and `SATISFACTION` with current solutions [15].

Perform survey includes finding and reaching potential survey respondents as well as extracting the collected `SURVEY RESULTS` for analysis.

P4. Prioritization phase

Calculate Opportunity Scores for each of the Job Stories or Desired Outcomes by applying the following formula to the survey results:

`OPPORTUNITY SCORE` = `IMPORTANCE` + (`IMPORTANCE` - `SATISFACTION`) [15].

Identify Opportunity Segments involves analyzing the opportunity scores to identify the Job Stories with a high rating for importance and a medium to low rating for satisfaction [54]. To do this, plot the Job Stories or Desired Outcomes on an `OPPORTUNITY GRAPH`, with the satisfaction on the y-axis and importance on the x-axis. Next, divide the `OPPORTUNITY GRAPH` in three `OPPORTUNITY SEGMENTS` to classify needs into `UNDER-SERVED`, `SERVED-RIGHT`, or `OVER-SERVED` [57].

Select high-priority Job Stories or Desired Outcomes in the opportunity segment to target and classify these as `HIGH-PRIORITY JOB STORIES` or `DESIRED OUTCOMES`.

P5. Project definition phase

Quantitative approach: Create Job Stories. The requirements engineer first formulates Job Stories based on the selected high-priority Desired Outcomes.

Identify non-functional Jobs that capture the fundamental problems of the customer his functional Jobs. As fundamental problems are rarely about a straightforward, functional task [14], these `NON-FUNCTIONAL JOBS` are necessary to provide insight into the underlying forces that drive customer behavior [13].

Select Jobs for development from the non-functional Jobs to address with a concrete development project and redefine them as `PROJECT JOBS`.

Select Job Stories for development. Depending on the project scope, the requirements engineer selects all- or a sub-set of the high-priority Job Stories associated with the Project Job. The resulting set of `PROJECT JOB STORIES` define the scope of the software solution.

Establish problem. From the selected non-functional job and its associated job stories, establish one or more re-occurring and concrete `PROBLEM`(s) that development can address in a concrete project.

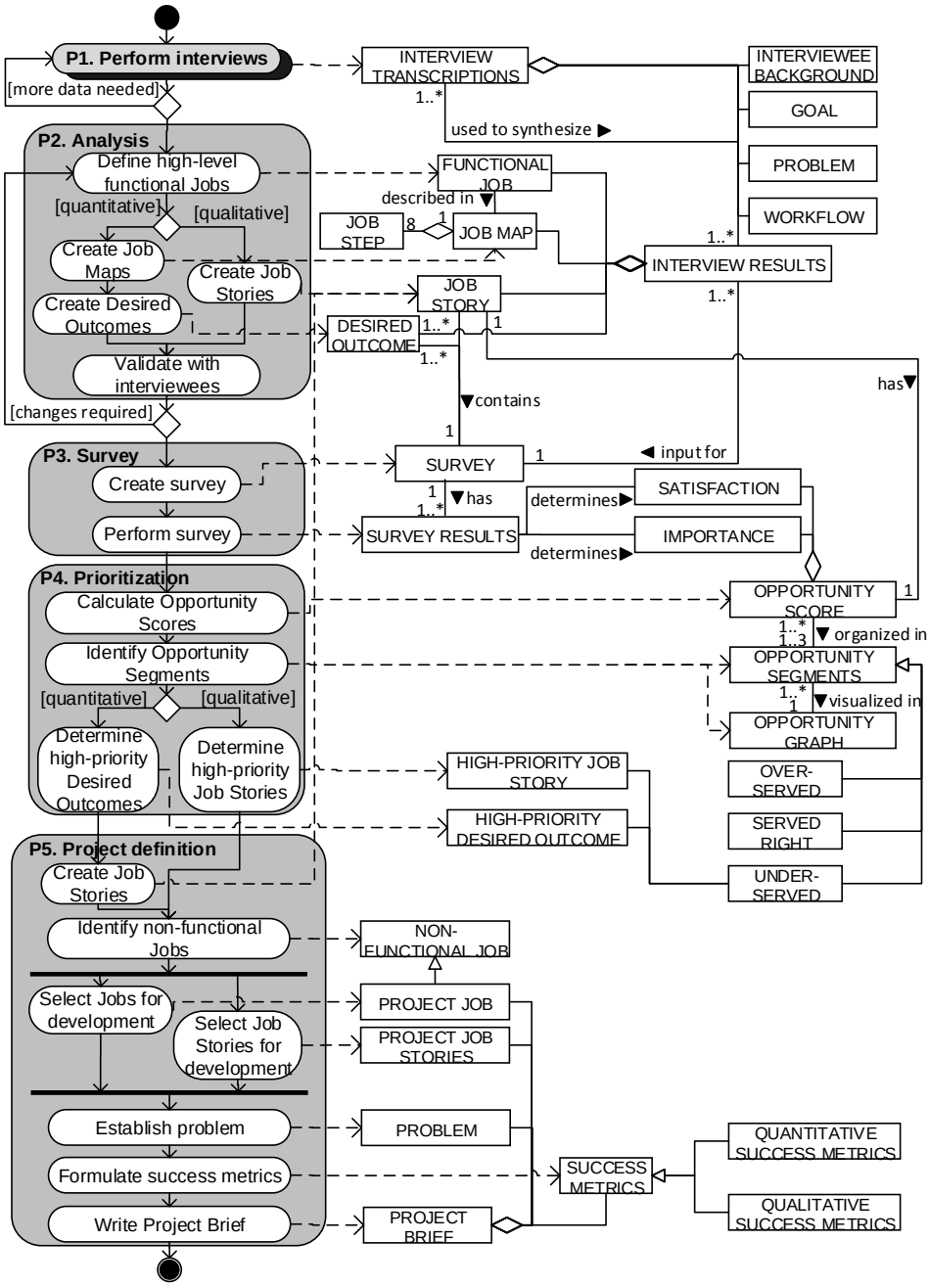

FIGURE 4.4: Integrated Job Story Method

Formulate success metrics by defining the `QUALITATIVE-` and/or `QUANTITATIVE SUCCESS METRICS` that determine whether the problem has been solved [5].

Write Project Brief that summarizes the results from all preceding phases. The resulting single-page `PROJECT BRIEF` includes a succinct description of the problem to be solved by the project, the relevant project jobs and project stories, as well as the success metrics to determine whether the problem has been solved [5].

Chapter 5

Case Study

In this chapter we discuss the practicalities associated with putting the Integrated Job Story Method to the test in a case study. To do this, we introduce the company where the case study is performed, specify the context and goals of the case study, and describe the protocol for evaluating the application of the Integrated Job Story Method in the case study.

5.1 Introducing the case study

We first introduce Stabiplan, the company where the project is performed, and then dive into the details of the case study.

5.1.1 About Stabiplan

Stabiplan is a Dutch product software company that is headquartered in the Netherlands but a large portion of its software development teams are located in Romania. Stabiplan operates in the market for computer-aided design (CAD) software. Its products extend the AutoCAD and Revit products by AutoDesk for mechanical, electronics and plumbing (MEP) installations. Founded in 1990, Stabiplan now has more than 170 employees and with more than 3,800 clients, it is the market leader in the Netherlands and Belgium [101].

Within the world of CAD, Stabiplan's main focus is on the emerging trend of Building Information Modelling (BIM). Getting increasing support from the European Union, BIM is a process that aims to improve efficiency by implementing large scale ditalization in the construction sector [102]. To facilitate BIM, IT is used to create an

accurate virtual model of a building that includes all relevant information [103]. This model consists of drawings, calculations, estimates, material lists and planning [104]. As illustrated in figure 5.1, the knowledge encapsulated in a BIM model should form a reliable basis for decisions during its entire life-cycle, meaning: from earliest conception to demolition [105].

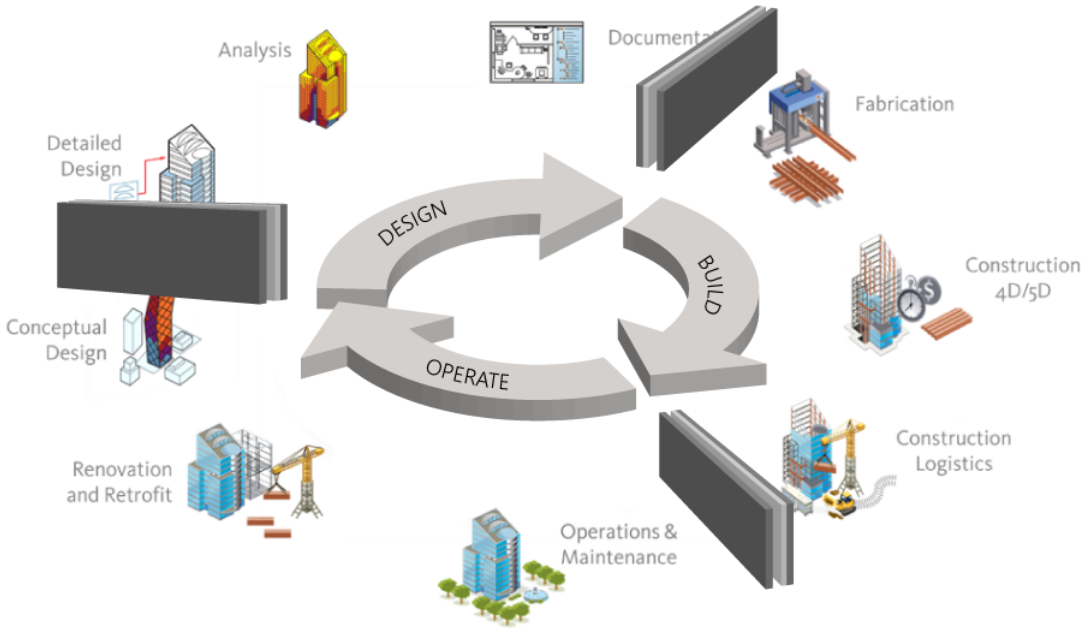

FIGURE 5.1: The BIM life cycle [106]

Stabiplan's core product is Stabicad. Available for both AutoCAD and Revit, Stabicad is used by engineers to design, visualize, control and optimize technical installations within the field of mechanical, electronics and plumbing (MEP). Of the two versions, Stabicad for Revit is the product that is tailored towards BIM, and with its three-dimensional modelling capabilities it is Stabiplan's main focus for the future [107].

Accompanying Stabicad, Stabiplan offers the MEPcontent library, which contains a collection of MEP elements that are use in BIM models. Due to BIM's focus on supporting the entire lifecylce of a building, in many cases the elements used in a model need to accurately represent their real-world counterparts. both in representation and in behavior. To make this possible, the elements in the MEPcontent library are created in close collaboration with manufactures and are linked with up-to-date, graphical and parametrical product information [108]. Through MEPcontent Stabiplan makes the elements available for BIM practitioners who do not use Stabicad. The elements in the MEPcontent library can be downloaded, however for non-Stabicad users that want more advanced feature a subscription model is in place.

5.1.2 The case at Stabiplan

Developing the MEPcontent platform plays an important part in Stabiplan's expansion strategy. With its market share of approximately 70 percent, opportunities for growth in the Dutch domestic market are limited. In pursuit of expansion, Stabiplan found Stabicad as a product, to be unsuitable for release on a more global market for three main reasons.

The first barrier that limits global expansion for Stabicad is the price of the product. Being expensive, customers have an extensive list of demands that need to be met for them to be able to justify adopting the product. As a result, sales advisors are often required to convince prospective customers that Stabicad meets their demands, and to push them to adopt the product. While Stabiplan has sales teams and offices for the most important European markets, covering a global market this way is not an efficient way to use resources.

Similarly, another barrier for global expansion is the technological complexity of the product. With its extensive portfolio of highly specialized functionalities, Stabicad is empowering for seasoned users, but also comes with a learning curve that is intimidating for new users. While Stabiplan is developing its e-learning resources, active support, training sessions and consultancy is often required to help customers implement and use the product.

The third barrier is related to localization in the MEP sector. Standards and naming schemes often vary region by region. Incorporating all localizations into Stabicad would be a difficult task and would likely complicate the user experience.

However, with its MEPcontent platform Stabiplan is reaching a global audience. To capitalize on this opportunity, Stabiplan is planning to extend its product portfolio with a set of Revit apps. These are independent products that like Stabicad extend the functionality of Revit, but at a much smaller scope. Due to their limited scope, the apps can be designed to be much more suitable for the global market. In doing this, Stabiplan is ideally looking to encapsulate functionality already present in Stabicad, but the apps can also bring new functionality to Stabiplan.

For Stabiplan to make the app strategy a success, the challenge is to create a portfolio of apps that will incite Revit users to adopt them. In doing this, the question is what functionality should be encapsulated into an app to maximize market success. For our research this is an interesting challenge, since it allows us to test our ability to use the Integrated Job Story Method to help come to an answer. From Jobs-to-be-Done perspective we can phrase Stabiplan's question as follows: what should a new app offer

to help customers make progress on the Jobs they are struggling with, so that as many customers as possible will decide to hire it. Due to the limited scope of the apps, it is particularly important to target a specific Job or Jobs that are important to as many customers as possible, and where the customers are also not already satisfied with existing solutions.

Now that we have introduced Integrated Job Story method, and the context and challenge of the case in which it is applied, we need to discuss how the method and its application in the case study will be evaluated.

5.2 Evaluation protocol

In this section we discuss the approach for evaluating the case study to help answer the research questions posited in section 2. The actual evaluation of the case study is reported on in section 6.2. The sub-research question that is relevant for this evaluation is:

SQ9: *"In what ways did the method positively or negatively impact business processes at Stabiplan?"*

To do this our evaluation approach is composed of two parts. First, we reflect on the method itself. Since the method attempts to integrate different aspects of Jobs-to-be-Done theory, it can be argued that there is a degree of arbitrariness to it. We therefore should discuss to what degree the method is an accurate representation of Jobs-to-be-Done, and if other approaches might be more suitable under different circumstances.

For the second and main part of the evaluation, we focus on the application of the method in the case study. Here, we reflect on what went right and wrong during the application of the method. To supplement our own reflection we also evaluate the method and its results with various stakeholders within Stabiplan. We discuss the stakeholder evaluation in more detail in the following sub-section.

5.2.1 Stakeholder evaluation

Since the primary goal of the project was to help Product Management identify interesting opportunities to pursue with an app related to radiators, Product Management is the main stakeholder and should evaluate whether the project has helped reach this goal, and whether the method helps Product Management in their practices.

However, in literature Jobs-to-be-Done is often described to have the ability to have an impact on an organization that is broader than Product Management. An example would be that it is said to help promote customer empathy throughout an organization [10]. To account for this, we also evaluate the method and project with stakeholders from Marketing and Development. By including Marketing we get a sense of the impact of the method on non-technical business functions, and the inclusion of Development helps us see the method from the perspective of those that build the solutions.

In the following sub-section, we discuss each stakeholder group and define a hypothesis that describes the value the method could deliver for that stakeholder. Sub-hypotheses are used to dive deeper and highlight different aspects that are relevant. In the evaluation sessions that are described in section 6.2, we provide each stakeholder with a form that contains the relevant sub-hypotheses and ask them to rate each on a Likert scale ranging from *'strongly disagree'* to *'strongly agree'*.

Product Management

Since the method includes the solicitation and prioritization of requirements, which are responsibilities of Software Product Management, PM is the most important stakeholder for the project. Given these responsibilities, to be valuable, the method should adhere to the following central hypothesis:

'The Integrated Job Story Method helps Product Management identify and address high level software requirements.'

To help answer this central hypothesis, we have defined several sub-hypotheses that unpack the potential benefits of the method into the different artifacts created by the method, and the method's practical applicability. These sub-hypotheses are included below.

We first explore the different artifacts created by the method. We are interested in the utility of the Project Brief for Product Management. In the following two sub-hypotheses we focus on the ability of the Project Brief to promote empathy and understanding of the needs of the customer, and its ability to promote the creative design of solutions by focusing on fully understanding the problem before diving in to the design process.

1. 'A Project Brief help Product Management understand how new software development should help the customer get Jobs done.'
2. 'A Project Brief can help me creatively design solutions that add value for the customer.'

Next we focus on the utility of Job Stories for Product Management. Here we are interested in whether Product Management thinks that Job Stories are indeed a good way to specify requirements.

1. 'The created Job Stories contain meaningful information on how to help the customer.'
2. 'The information contained in a Job Story can help Product Management design software solutions.'

For the 'Opportunity Prioritization' technique, we are interested in whether Product Management finds this a useful way to prioritize requirements, and whether prioritization that was performed during the project has practical value for PM.

1. 'Using customer data to classify needs as over-served, served right or under-served, can help PM determine how to improve our products.'
2. 'The Opportunity Graph created during the project has practical value for PM.'

With regard to the Job Portfolio we are interested in whether Product Management can use it as a practical tool to aid strategic decision making.

1. 'A Job Portfolio can help Product Management be aware of the Jobs that Stabiplan is hired for.'
2. 'A Job Portfolio can help Product Management formulate a strategy on how to improve our products to help the customer better in the future.'

For the Job Map we are interested in whether Product Management thinks it has practical value. If so, we want to know whether this value is more on a strategic or operational level, since operation value might call for a larger role of the Job Map in the method.

1. 'A Job Map helps me understand the customers' perspective of interacting with Stabiplan to get Jobs done.'
2. 'Product Management can use a Job Map to strategically determine how Stabiplan could add value for the customer.'
3. 'Product Management can use a Job Map to help me design software solutions.'

Finally, we are interested in how Product Management perceives the practical applicability of the method. Here, we should discuss the results of the method in the case study, whether Stabiplan can apply the method at a larger scale, and whether Product Management finds it worthwhile to apply the method.

1. 'The application of the Integrated Job Story Method in the case study has yielded valuable results for Product Management.'
2. 'In the future, Stabiplan can start applying the Integrated Job Story Method at a larger scale.'
3. 'Applying the Integrated Job Story Method is worthwhile from a cost/benefit perspective.'

Marketing

Evaluating the project and method with marketing is interesting because there is a close link between Jobs-to-be-Done and the marketing of products. The quintessential example of Jobs-to-be-Done is Clayton Christensen's milkshake marketing case [13]. In theory, Jobs-to-be-Done can help marketeers convince their target audience since speaking to the Job that a prospective customer is trying to get done, will help create better marketing material [10].

For the Sales department, the situation is not very different. Understanding the context and motivations that might drive a prospect to hire a new product can be very valuable when trying to convince them to hire your product [109]. However, at Stabiplan the Sales the department is not directly involved in selling the apps to customers. While the artifacts created by the method might also be useful for Sales, we therefore focus on evaluating them from the perspective of the Marketing department.

The central hypothesis for the marketing evaluation is:

'The artifacts created by the Integrated Job Story Method help Marketing be aware of how Stabiplan helps customer get Jobs done, and improve their marketing efforts.'

To be able to answer this hypothesis, we first need to verify whether Marketing can indeed benefit from adopting a Jobs-to-be-Done mindset in their work. Afterwards, we need to discuss the different artifacts created by the method to determine which are useful for Marketing.

For this evaluation we have identified two main parts in the work of a marketeer: determining the optimal marketing strategy and creating marketing material. To investigate

whether the marketeers at Stabiplan find Jobs-to-be-Done theory appealing for these different parts of their work, we have formulated the following sub-hypotheses:

1. 'Explicitly thinking about the customers' Jobs-to-be-Done and how Stabiplan delivers value helps Marketing formulate a marketing strategy.'
2. 'Explicitly thinking about the customers' Jobs-to-be-Done and how Stabiplan delivers value helps Marketing create powerful marketing material.'

The rest of the evaluation should focus on determining which of the artifacts created by the method the marketeers find valuable, and investigating how they would use them in practice. Below, sub-hypotheses are included that explore the Project Brief with the Job Stories, the Job Portfolio model, and the Job Map.

1. 'A Project Brief helps Marketing understand how a specific new software development project helps the customer get Jobs done.'
2. 'A Project Brief helps Marketing create powerful marketing material for the related solution.'
3. 'A Job Portfolio model helps Marketing be aware of the bigger picture of how Stabiplan helps the customer get Jobs done.'
4. 'Marketing can use a Job Portfolio model to help formulate a marketing strategy.'
5. 'Marketing can can use a Job Portfolio model to help create powerful marketing material.'
6. 'A Job Map helps Marketing understand the customers' perspective of interacting with Stabiplan to get Jobs done.'
7. 'Marketing can use a Job Map to help formulate a marketing strategy.'
8. 'Marketing can use a Job Map to help create powerful marketing material.'

Development

The third stakeholder that is involved in the evaluation is Development. Here, it is important to consider that at Stabiplan, the developers build software based on a fully specified design. This means that the developers work only in the solution space of product development. Since Jobs-to-be-Done is fully focused on the problem space [5], developers are likely to be not directly affected by the results of the method. For other

organizations this might be different, depending on the software development practices that are in place.

Due to the role of development, the only artifact that could be relevant is the Project Brief with the Job Stories. This is because the Project Brief relates to a specific project that development might be working on. As a result, our central hypothesis becomes:

Having a Project Brief available for new development projects helps Development build software solutions.

We further elaborate on this central hypothesis with three sub-hypotheses.

1. 'A Project Brief helps Development be aware of the context in which customers will use their work.'
2. 'Job Stories contain meaningful information that is useful for Development.'
3. 'A Project Brief and Job Stories will help Development think creatively when building software.'

Chapter 6

Analysis & Results

Now that both the case and the Integrated Job Story method have been introduced, in this chapter we dive into the process of actually applying it, and describe the analysis and the results. Afterwards, the method is evaluated according to the protocol described in section 5.2.

6.1 Applying the Integrated Job Story Method

First we discuss the application of the method. Following the same structure used in section 4.5, we discuss each phase of the method individually. To shortly recap section 5.1.2, the goal of applying the method was to determine what Jobs and Job Stories should be addressed by a new 'Stabicad for Revit' app related to radiators, to provide as much value to customers as possible and incite them to adopt it.

P1. Interview phase
Perform interviews. We conducted and transcribed four extensive (1.5 hour each) interviews with users of Stabiplan working in appliances and connectors, the product's target market. During the interviews, the focus was on uncovering:

1. The professional background of the interviewee.
2. The goals that the interviewee is trying to achieve when working with Revit.
3. The problems that make it difficult to achieve these goals.
4. The workflow of how the interviewee tries to overcome the problems and achieve their goals. Following principles from the 'Job Mapping' technique, the focus is on what the interviewee is trying to accomplish in their workflow, rather than on how they are doing this [54].

P2. Analysis phase

Define high-level functional Jobs. Analyzing what the interviewees are trying to achieve when working with radiators in Revit, we formulate a set of functional Jobs that are included in the appendix in section 1. Since there was no capacity to study all high-level functional Jobs, it was decided to limit the scope to the 4 Jobs that most closely relate to using Revit for radiators.

J1 - 'Help me configure radiators.'

J2 - 'Help me place radiators.'

J3 - 'Help me model piping systems.'

J4 - 'Help me create bills of materials.'

Create Job Maps. We applied the Job Mapping technique to analyze the customers' workflow and identify all the lower-level functional Jobs the customers are trying to get done. Later in the project we created a fully fledged Job Map which is included in section 1 of the appendix

Create Desired Outcomes. Analyzing the interviewees' struggle when trying to get their Jobs done, we defined 50 Desired Outcomes for the four Jobs above. The rationale for this approach is that since a Desired Outcome is a metric that customers use to determine to what degree they are able to get a Job done successfully [16], a customer that struggles is an indication of an unsatisfied Desired Outcome.

While the full collection of Desired Outcomes is included in section 1 of the appendix, for J1, two examples of Desired Outcomes are 'Minimize the chance that I need to check external documentation' and 'Minimize the chance that a fabrication specific bottom connection block I need is not available'.

Create Job Stories. Using the contextual information identified in the interviews, we created 21 Job Stories that relate to J1-4 such as '**When** I am configuring a radiator, **I want** the heating power of the radiator to be accurately represented in the flow and return temperatures, **so that** my model will correctly represent the produced heat.' The full collection of created Job Stories is included in section 1 of the appendix.

To create the Job Stories, we found it useful to start with the moments of struggle that were identified during the interviews, and then fill in the Job Story template: '**When** [the situation in which the struggling moment occurs], **I want** [the motivational forces that drive the customer], **so that** [the outcome the customer hopes to achieve].' [6]. Finally, it should be mentioned that creating the Job Stories was an iterative process, and changes were made based on feedback from stakeholders at Stabiplan.

Validate with interviewees. We took an iterative approach to this activity by validating draft Job Stories based on the interviewee's input. The outcome was considered satisfactory and no substantial changes to the Job Stories were necessary.

P3. Survey phase

Create survey. We created an online survey with 7-point Likert scale questions for each of the Job Stories which is available upon request here `https://goo.gl/rvbfZw`. While the interviews were performed with dutch users of Stabicad for Revit, the survey is aimed at the target audience for the new app: international MEPcontent users, who usually work with a 'plain' version of Revit combined with a few extensions. As such, the survey is designed to validate and prioritize the Jobs and Job Stories that were created based on the interviews.

Perform survey. We distributed the survey and obtained 10 responses from the target audience. This disappointing amount if respondents is due to the fact that the target audience falls outside of Stabiplan's main clientele. Without a (strong) relationship with Stabiplan, there is little to help motivate people to participate in the survey.

Initially, the survey was distributed among a set of 195 registered MEPcontent users that had shown interest in radiators by downloading content related to radiators. Only yielding five responses, the survey was also posted on several online forums such as the Revit- and RevitMEP subreddits, and the Revit Forum, through which five more responses were added.

P4. Prioritization phase

Calculate Opportunity Scores. Using the 'Opportunity Prioritization' formula, we calculated the mean opportunity score for all respondents for each Job Story [15]. For 'importance' and 'satisfaction', we used the average ratings for each Job Story. Ideally, we would have wanted to experiment with how differences in demographic characteristics and other contextual factors influenced the ratings. Unfortunately, this was not possible due to the limited sample size.

Identify Opportunity Segments. Applying the technique proposed by Anthony Ulwick, we plotted the Job Stories' Opportunity Scores on the Opportunity Graph as illustrated in Fig. 6.1 [55, 56].

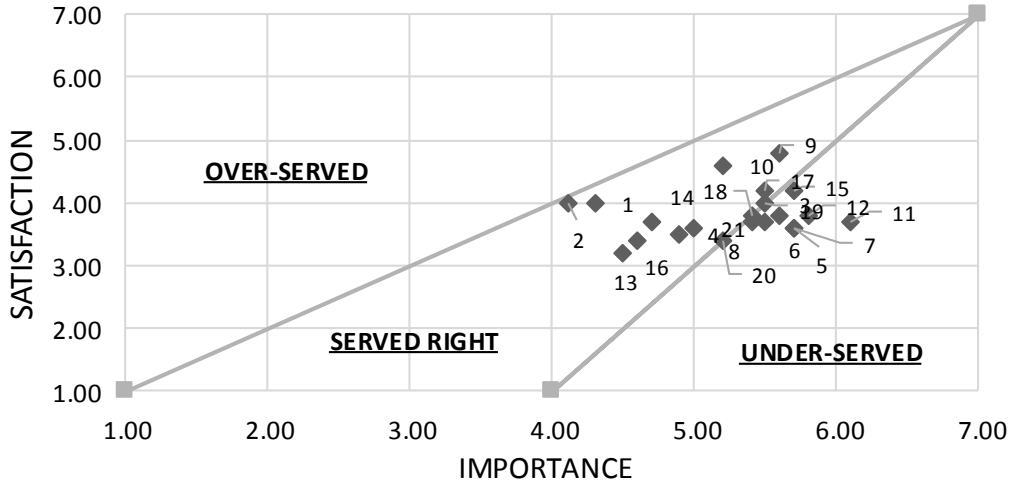

FIGURE 6.1: Stabiplan's Job Stories plotted on an Opportunity Graph

P5. Project definition phase

Identify non-functional Jobs. As there is no literature on how to categorize Job Stories based on the common non-functional Jobs, Stabiplan defined their own four step approach:

1. We define three high-level non-functional Jobs that drive customers to hire Stabiplan to help their organization excel at Building Information Modeling.
2. We decompose each high-level non-functional Job into two or more medium-level sub-non-functional Jobs.
3. We categorize all Job Stories according to the medium-level non-functional Job they belong to.
4. For each medium-level non-functional Job, we sub-categorize its associated Job Stories in low-level non-functional Jobs.

Our approach provides us with insight in the portfolio of non-functional Jobs that Stabiplan is hired for by customers. To illustrate this, we have created a Job Portfolio model, which is a variant of commonly used product portfolio diagrams. To save space the Job Portfolio is divided into two parts: figure 6.2 includes the high- and medium-level non-functional Jobs, and 6.3 includes the low-level non-functional Jobs. Upon request, a larger, full version of the Job Portfolio is available, that is also linked to the Job Stories.

Select Jobs for development. To select the most important Jobs for further development projects, we want to select the low-level Jobs whose Job Stories have the highest mean opportunity score. As the focus of this project is on improving *modeling* for radiators, we select low-level non-functional Jobs 1 and 4: *'Help me ensure that I deliver high quality work'* and *'Help me save time'*.

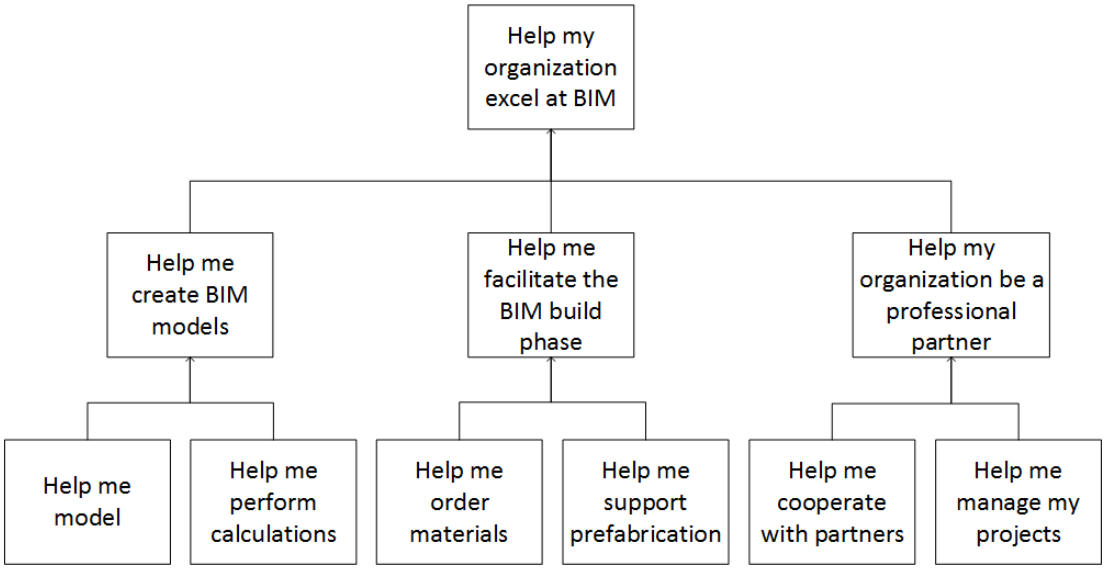

FIGURE 6.2: Stabiplan Job Portfolio: high- and medium level Jobs

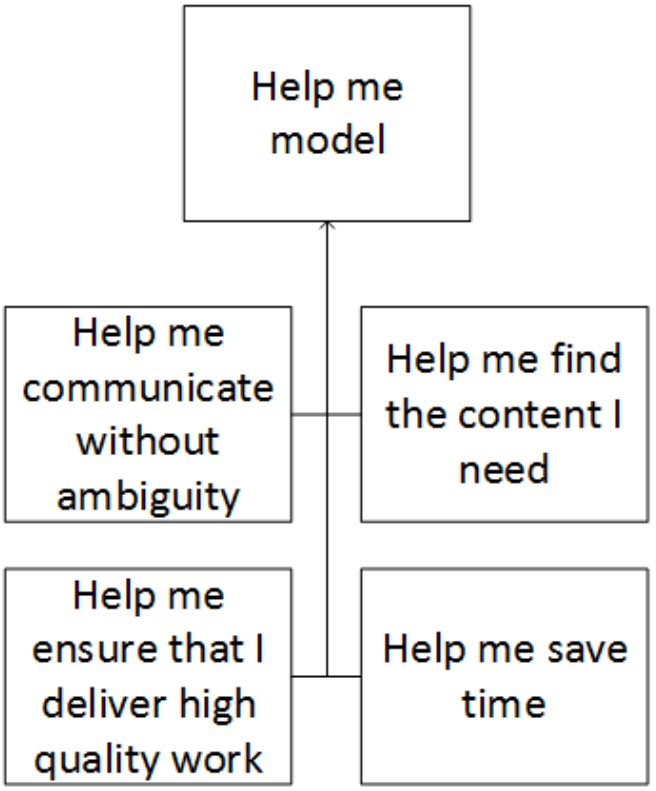

FIGURE 6.3: Stabiplan Job Portfolio: low level Jobs

Select Job Stories for development. Together, Jobs 1 and 4 comprise 10 Job Stories. To identify the most important Job Stories for inclusion in development, we select only those that are under-served, i.e. with an opportunity score of 7.0 or higher. Applying this exclusion criteria results in a total of 8 Job Stories as shown in Fig. 1.

Establish problem. Combining the selected Job Stories and our growing insights from the

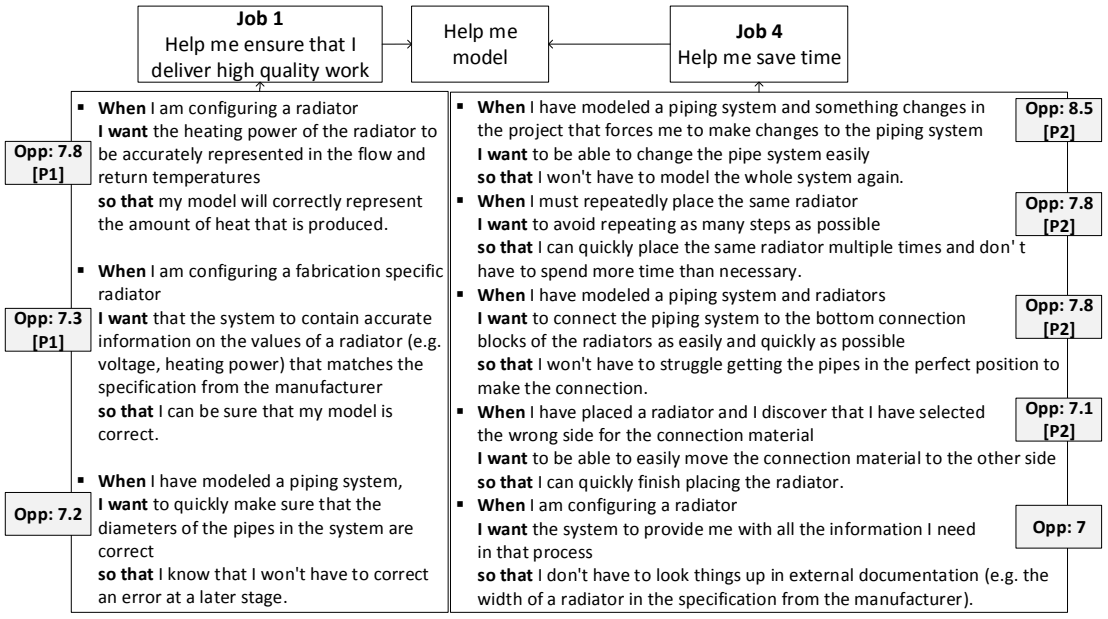

FIGURE 6.4: Snippet of Stabiplan Job Portfolio for radiators

interviews, we identify a concrete and distinct problem for each Project Job as shown in Fig. 1. For Job 1, we identify that modelers struggle to ensure that the radiators used in a Revit model are linked to the correct parametric information (P1). For Job 2, we identify that modelers struggle to change an existing piping system to incorporate changing requirements (P2).

Formulate success metrics. We defined high-level success metrics that do not presuppose a solution, such as *'the solution helps the modeler feel confident that the appliances are configured correctly'* for Job 1 and *'the time it takes to change the types of appliances that are used in a piping system is reduced by X%'* for Job 2. Note that these may be made more specific later on.

Write Project Brief. Using the Jobs, Job Stories, problem and success metrics defined earlier, we wrote one Project Brief for each of the two Jobs. To help tie the different elements together and provide more context, we also defined a high-level Job Story for each Job.

Job 1: '**When** I am working on a complicated model for an important project and I cannot afford to make mistakes, **I want** to be able to identify and fix possible errors, **so that** I can be confident that the work I deliver is of high quality.'

Job 2: '**When** I have been working on a model for a while but suddenly the requirements change, **I want** to be able to quickly modify my existing model to address the new requirements, **so that** we can prevent major delays to the project.'

6.2 Evaluation

In this section we evaluate the project and method as was described in section 5.2. First we reflect on the method from a theoretical perspective, then we discuss our experience of applying it in the case study, and finally we evaluate the method and it's results with stakeholders at Stabiplan.

6.2.1 Reflection from a theoretical perspective

As mentioned in section 5.2, it can be argued that our method is based on a somewhat arbitrary selection of concepts and principles from Jobs-to-be-Done. While the method does not represent a universally accepted approach to Jobs-to-be-Done (since there is none), we can assert that the different phases of the method are grounded in Jobs-to-be-Done theory. However, it is possible that in some cases, a different configuration would be more appropriate. Below, we discuss some relevant parts of Jobs-to-be-Done theory that have not been included in the method, explain why, and present some scenario's in which it might be appropriate to use them.

Emotional Jobs. Some Jobs-to-be-Done practitioners argue that Jobs should not be defined from a functional perspective. The rationale is that by defining Jobs from an emotional perspective, you stay far away from specific implementations and are more likely to convey the deeper reasons a Job is important [10]. However in the method, emotional Jobs are not used because it seems to be (near) impossible to design functionally complex products with emotional needs as a primary design input [17]. Our experience indicates that this is indeed the case. However, there are reasons to believe that for innovators that operate in the context of B2C products, it could very well be valuable to pay closer attention to the emotional dimensions of Jobs-to-be-Done [10].

Desired Outcomes. While we have included Desired Outcomes in the method by means of variability, the method primarily relies on Job Stories. This was done for two main reasons: Job Stories were found to be more suitable for use in the survey than Desired Outcomes, since they include more contextual information and unlike Desired Outcomes do not suffer from issues related to negative framing. The second reason is that due to the contextual information they convey, and the fact that they are currently already used by practitioners in the software industry, they are more suited to support design processes in this context.

However, in a large scale application involving many interviews and survey participants, it could be advisable to use the alternate 'quantitative' path illustrated in the method, and use Desired Outcomes until P5. where they are used as a basis for defining Job

Stories. This is because significant effort is required to create a set of Job Stories that accurately represents a large set of Desired Outcomes. Therefore, to save time it could be advisable to use the Desired Outcomes in the survey, create a prioritization based on the results of the survey, and create Job Stories based on the Desired Outcomes that have been assigned high-priority status.

Switch Interviews. By some considered an integral part of Jobs-to-be-Done, Switch Interviews involve the practice of interviewing (ex-)customers who have recently switched from one solution to another, to uncover the contextual and motivational forces that made them switch [10]. While this might indeed be a good way to uncover the Jobs-to-be-Done, the Integrated Job Story method relies on the 'Job Mapping' technique because it is less restrictive in which participants are suitable and can therefore be applied more widely [54].

Statistical analysis. In ODI the prioritization of needs based on the survey, involves various statistical analyses to identify segments of market opportunities [110]. For the Integrated Job Story Method we decided exclude this, since the optimal process is not disclosed in ODI, and it would add complexity to the method. Practitioners looking to apply the method should be aware of this, and consider whether thoughtful statistical analysis of their data is worthwhile.

6.2.2 Reflection on the application of the method

In this section, we reflect on our experience applying the method in the case study. Following a similar structure as used earlier, we shortly discuss each phase of the method separately.

P1. Interview phase

For the interview phase, it should be noted that the process of arranging interviews with suitable participants proved to be a bottleneck for the project. Initially, the goal was to perform interviews with members of the target audience, the registered MEPcontent users - via video conferencing. However, unfortunately these users were not interested in participating. We therefore decided to compromise and conduct the interviews with existing Stabicad for Revit users in the Netherlands. Even still, finding willing participants that worked with radiators in Stabicad for Revit was a challenge.

Reflecting on the interviews themselves, in hindsight it was for the best that the they were performed in-person, instead of via video conferencing. Conducting the interviews with Revit running so that the interviewees could show exactly what they were talking about was helpful in gaining a correct understanding.

While the number of interviews that were performed (4) was limited, they were conducted at a sufficient level of detail to capture the needed information. Here it helped that focus was on a small part of Revit.

P2. Analysis phase

Defining the high-level functional Jobs was a straightforward process. These were the same activities many of the interviewees were performing in their workflow. Expanding on these high-level functional Jobs, initially our strategy was to create a Job Map for each Job. However, after some experimentation we concluded that this was a too low level of analysis. Initially deciding to move on with the project, we later created a full Job Map that aimed at the entire process of working with radiators in Revit.

Creating the Desired Outcomes was also a relatively straightforward process. For each functional Job, we had asked the interviewee's what was important for them and what problems they struggled with. With this information, we were able to define the Desired Outcomes.

Creating the Job Stories was more difficult. While the information to be included in the three parts of each Job Story had become clear through the interviews, determining the best way to formulate a Job Story was not easy. After exploring many variants with different levels of granularity and formulations, the created Job Stories are a compromise between the very high level Job Stories Klement describes, and a more practical focus on the context of software products.

P3. Survey phase

When creating the survey, we quickly found that although preferable, it would not be feasible to include all Job Stories in the survey. To find a good balance between having a sufficiently broad scope and ensuring that respondents were going to finish the survey, it was decided to focus on the Job Stories related to the four non-functional Jobs mentioned in section 6.1.

While we adopted the survey technique from ODI, it was decided to use a 7-point Likert scale for the questions instead of the 10-point scale used in ODI. This was done in an effort to avoid possible biases associated with rating scales without a mid-point [111].

Finally, it should be discussed that performing the survey proved to be another bottleneck for the project. Finding suitable participants who were willing to fill in the survey was a challenge. This was likely due to the target audience not being a part of Stabiplan's core customer base, and the still considerate length of the survey. Resorting to online forums we eventually managed to get 10 Revit users who worked with radiators to fill in the survey. In reflection, it would have been better to have designed the project

around a target audience that is easier to reach and motivate to participate in both the interviews and survey.

P4. Prioritization phase

The prioritization technique from ODI relies on the results of the survey to determine which needs - which we described using Job Stories - present the largest opportunity for innovation. Being a quantitative technique it would have been preferable to have a larger sample size for the survey. This would have increased the reliability of the prioritization, and allowed us to do perform more interesting analyses.

With more data, we would have been able to investigate the impact of the professional context of the modelers on the opportunity scores for the Job Stories. We would for instance have liked to investigate the difference in opportunity scores between modelers who mostly work on residential housing projects, and those who specialize in utility projects.

P5. Project definition phase

In this phase, the main challenge was to go from the prioritized Job Stories to a concretely defined scope for an app or apps. The level of detail on which the Job Stories are defined do not allow us to simply take the Job Stories with the highest opportunity score, and presume that each should be addressed by an app. This is not feasible because depending on the implementation, more Job Stories are likely to be relevant for an app.

Organizing the Job Stories based on the non-functional Jobs helped to work-around this issue, since they should convey the different aspects that drive customer behavior. Here, the under-served Job Stories indicate the parts of that Job that the Revit user should be interested in getting done better.

To define the non-functional Jobs, it was helpful to analyze Stabiplan's product portfolio, and consider what deeper Job each offering helps its users get done. Illustrating the Jobs that Stabiplan is hired for in a 'Job Portfolio' diagram helped us categorize the Job Stories according to the non-functional Jobs they contributed to. While this does not solve our scoping problem entirely, since at this stage we cannot be sure how many apps are needed to adequately address a Job, it seems to be a useful tool to incrementally build a portfolio of apps that are focused on addressing the different Jobs.

We used the Job Portfolio to select the two non-functional Jobs whose Job Stories have the highest mean opportunity score. For this project we wanted to define a possible app for each Job. As a first step in defining the scope of the apps, we eliminated the Job Stories that were not under-served. Since there are many different ways the apps could help Revit users get the relevant Jobs done better, we had to define a concrete problem each app was going to address. At this stage it was challenging to define the problems

at a level that is both actionable as input for the design process, but does not push the design in the direction of a certain solution. To formulate the problems, we were able to use the transcriptions of the interviews.

Following the format from Intercom, the last steps were to define success metrics for each app, and summarize the previous steps into two project briefs [5]. At this stage, formulating the success metrics proved difficult, since we found that is was very easy to presuppose a solution in a success metric. To account for this, we defined high-level success metrics that are to be made more specific after a solution has been defined.

6.2.3 Stakeholder evaluation

In this section we evaluate the project from the perspective of three groups of stakeholders: Product Management, Marketing and Development.

Product Management

The project was evaluated in detail with the lead Product Manager at Stabiplan, who was closely involved with this project. Below, we discuss the opinion of the lead Product Manager on the artifacts created during the project, the method itself, and the lessons learned for Stabiplan.

On a high level, the product manager likes Jobs-to-be-Done theory for the emphasis it puts on understanding how and why customers use products. An integral part of applying Jobs-to-be-Done is interviewing customers and distilling requirements, which the product manager finds useful and important practice. While the Project Briefs and Job Stories that were created as a result did not present revolutionary new insights to the product manager, he did find them effective for scoping a project and conveying their customers' priorities to internal stakeholders. Furthermore, the product manager thinks that due to their focus on describing the problem, the Project Briefs can help with the creative design of solutions.

"On their own these Job Stories do not provide me with deep insights. I think that they are a fine way of writing requirements but the real value is in framing them in a Project Brief, and then designing a creative solution."

However, the most valuable part of the method for PM was the prioritization of the Job Stories in the Opportunity Graph. The product manager likes that it is based on

customer research and provides a clear overview of the importance and current satisfaction of needs in a market. Through experience, the product manager has gained a solid understanding on how to address the current market. He therefore thinks that the opportunity prioritization technique is most valuable when investigating how to best address a new market segment, as is the case with the apps.

"Exploring what Job Stories are under-served, served right and over-served is very valuable to target apps to a specific country or market. Within the context of an app, we can use this prioritization to determine what needs to emphasize in our solution."

Building upon the opportunity prioritization, the product manager found it useful to think about its offerings in terms of the Jobs they are hired for by customers, as expressed in the Job Portfolio.

"I like this overview and I think that the categorization is correct. Analyzing our offerings and explicitly linking them to the customers' needs or Jobs are very useful activities for product management."

The utility of the Job Map was less obvious to the product manager. He noted that it could indeed be a nice way to visualize the results of the customer research. However, it does not add much value over the Job Portfolio and Project Brief.

"This is another visualization of how to present the results, I think that this could be a useful approach but I do not necessarily see us applying it in practice."

To summarize we can say that the product manager liked the Project Briefs and the opportunity prioritization, while the Job Portfolio and the Job Map can be considered as 'nice to haves'. Reflecting on the method as a whole, the product manager concluded that while it includes useful activities and delivers valuable results, there are some problems that limit its practical usability.

First of all, the method is heavily dependent on the participation of existing or prospective customers. During the project, we struggled to find suitable participants for both the interviews and survey. It was difficult to motivate customers to participate and the the limited scope of the project meant that only a smaller portion of the customer base was suitable. When applying the method at a larger scale and repeatedly recruiting customers to participate in the process, there is a danger of overburdening the customers. The second problem is that as a whole, performing the method is very time-consuming. Therefore, the product manager will not be implementing the method in its current form.

However, Stabiplan will be doing research on various ways to use its online community to make it easier and quicker to collect the data that is needed for the method.

"For our app strategy we are looking at ways to activate online communities in countries where we currently do not have a presence. If we can establish a community that can use to collect data, and we can use some automation to process the data, I think it will be very useful to work this way."

Marketing

Three marketeers participated in the hour long evaluation session. During which, the findings from the project were presented and their utility for Marketing was discussed. The session was recorded and the marketeers filled out an evaluation form that featured the propositions listed in section 6.2.3. Below we shortly discuss the results of the evaluation.

When discussing the appeal of Jobs-to-be-Done, the marketeers found that adopting a Jobs-to-be-Done mindset could help them in their work. Explicitly thinking about the Jobs customers' are trying to get done and how Stabiplan delivers value, can both help formulate the optimal marketing strategy, and help create powerful material.

"This mindset helps to validate assumptions about customer needs and software requirements that are incorporated in the marketing strategy. For our strategy we might assume that everyone uses a certain feature the same way. However, Jobs-to-be-Done could help in finding out that this is not the case and that we should change our marketing in Germany or France."

To put Jobs-to-be-Done into practice, the marketeers found the Project Brief to be the most valuable artifact that was created during the project. Since it concretely defines a specific project, they found that they could easily incorporate it in their work. In particular, the marketeers consider the contextual information in the Project Brief to be useful when creating marketing material. Due to the highly technical and specialized market Stabiplan operates in, it can be difficult for marketeers to convey what makes a certain improvement or feature important for the customer. To support them in doing this, the marketeers find the Project Brief to be much more useful than the feature lists that are currently used.

"A feature list only shows what a product can do, it does not necessarily help highlight the advantages. With Jobs-to-be-Done you can connect with a target audience by

showing that you understand the problems they are dealing with and incorporate this into the marketing material."

Less relevant for the marketeers, were the Job Portfolio and Job Map that were created during the project. For the Job Portfolio, the marketeers did find that the provided contextual information on the Jobs that Stabiplan is hired for by the customer and how Stabiplan addresses these Jobs, could be useful when determining marketing strategy. Insights from the Job Portfolio could for instance be used to create a marketing campaign aimed at a particular job. However, the marketeers did not see a use for the Job Portfolio when creating marketing material, which severely limits its utility.

"When creating marketing material we need to convey a message to the target audience, this Job Portfolio does not really help me explain why someone should download a particular app. The Job or Job Stories are more relevant for creating marketing material."

For the Job Map, the marketeers did appreciate its ability to show what is important to a customer when interacting with a product. While this can be useful for marketing, the marketeers found it difficult to determine concrete ways to implement the Job Map in their workflow that would justify the amount of work that is required to create one.

"The Job Map seems to be less directly relevant for us, but it could be a reason for us to do some more in-depth research on one of the Job Steps." "I think that the Job Map would mainly be useful to help link the functionality of a product to the whats in it for them for the customer. What we could do is pick one, or multiple, Job Step from a Job Map to highlight in the marketing message."

Development

A smaller evaluation was performed with members of the development team after a presentation about the project. At Stabiplan, developers receive a fully fleshed specification of the software they need to build. Therefore, the developers are not very involved in the problem space where Jobs-to-be-Done is focused on. As a result, the developers consider the project brief and its associated Job Stories to be defined on too high a level to be useful for actual software development. However, the developers did believe that the information in the project brief is useful to familiarize new hires with the non-technical context of Stabiplan's products.

"By showing how our work affects the customer, the Project Briefs could help new hires understand the context of our products and enable them to become more pro-active in the development process."

6.3 Conclusions

In this chapter we have described the results of applying the Integrated Job Story method in the case study, discussed our experience applying it, and performed an evaluation with stakeholders at Stabiplan. Ultimately, the goal of this chapter was to investigate whether the Integrated Job Story Method could realize the theoretical benefits of Jobs-to-be-Done and Job Stories in practice, by being valuable for stakeholders at Stabiplan. Since 'value' is a difficult concept to measure, sub-question 9 focuses on the impact of the method on the business processes at Stabiplan:

'In what ways did the method positively or negatively impact business processes at Stabiplan?'

We answer this question by reflecting reflecting on the evaluation that was performed with key stakeholders at Stabiplan, and weigh the positive effects of the method against the negative. Overall, we can conclude that the case study yielded positive results but that there are also some issues that limit its practical applicability.

Starting with Product Management, we have discussed that the prioritization of Job Stories based on their 'Opportunity Score' was found to be a very effective way to identify what needs to address in a market. Furthermore, Product Management found using the Job Stories and Project Briefs to be valuable. While not delivering groundbraking insights, due to the contextual and problem oriented nature of the information they convey, they are supportive in the design process.

While the Integrated Job Story method proved to be not directly beneficial for Development, the artifacts of the method had real value for Marketing. Namely the Project Brief, which they found to be very helpful when creating marketing material, since it conveys how a feature or solution adds value for the customer.

However, where the method has had a negative impact is in the fact that it proved to be much more time-consuming than Stabiplan's traditional processes. Finding suitable participants for the interviews and survey, and performing the various analyses can be a significant bottleneck. As a result, for Stabiplan the method is not suitable for every

project, and is best saved for exploring new market segments. To help increase its applicability and cut down on the time required for its execution, Stabiplan is investigating new ways to utilize its online community to help automate data-collection.

Chapter 7

Discussion

In this project, we performed an exploratory study on the use of Job Stories and Jobs-to-be-Done in the context of requirements engineering for software products. As an emergent technique, Job Stories are increasingly popular among practitioners in the software industry. As discussed in section 1.1, Jobs-to-be-Done is described in a fragmented body of literature that features different conflicting approaches that are mainly attuned to physical products.

In the context of software products, there are very few well-documented examples of the application of the theory available. Combine this with the conflicting approaches all marketed under the label of "Jobs-to-be-Done" and we have a confusing situation for practitioners in the software industry looking to apply Job Stories and Jobs-to-be-Done in their work. Furthermore, a lack of scientific literature on Job Stories makes it difficult to come to an objective judgment of the value it adds to the software industry.

We address this problem by performing an exploratory study that aims to answer the following main research question:

'What is the value of Job Stories and Jobs-to-be-Done in the context of requirements engineering for software products?'

Describing the value of Job Stories and Jobs-to-be-Done in software requirements engineering is a multi-faceted endeavour that we address through a set of sub-questions that were introduced in section 2.1. In the following section, we discuss these sub-questions in three parts that are based on the main phases of this project.

7.1 Answering the Sub-Research Questions

For the first part of the discussion that is described in section 7.1.1, we reflect on the theoretical exploration of Job Stories and Jobs-to-be-Done. Through answering the first four sub-questions we gain a solid understanding of the theory and what general approaches exist for its application.

SRQ1: *'What are the principles of Jobs-to-be-Done theory?'*

SRQ2: *'What are the main concepts and artifacts related to Jobs-to-be-Done?'*

SRQ3: *'What are the existing approaches to applying Jobs-to-be-Done?'*

SRQ4: *'How do the different approaches to applying Jobs-to-be-Done relate to each other?'*

In the second part, which is described in section 7.1.2, we work towards establishing the value Job Stories and Jobs-to-be-Done add to the software industry, by investigating how they are applied in this context. By doing so, we answer the fifth, sixth and seventh sub-question:

SRQ5: *'What are examples of the application of Job Stories and Jobs-to-be-Done in the context of software products?'*

SRQ6: *'What main usage scenarios can be identified for Job Stories and Jobs-to-be-Done in the context of software products?'*

SRQ7: *'How are Job Stories and Jobs-to-be-Done positioned in the landscape of requirements engineering for software products?'*

For the third part, described in section 7.1.3, we answer the final three sub-questions by proposing a concrete method for the use of Job Stories and Jobs-to-be-Done in software requirements engineering, which we apply and evaluate in a case study at Stabiplan. By providing a well-documented report of our experience we gain insight into the value of the method and theory in the case study, which we then use to come to broader conclusions about the value of Job Stories and Jobs-to-be-Done in the context of software requirements engineering.

SRQ8: *'What is a method for using Job Stories and Job-to-be-Done to perform requirements engineering for a software product?'*

SRQ9: *'In what ways did the method positively or negatively impact business processes at Stabiplan?'*

SRQ10 : *'What findings from the case study can be generalized outside the context of the case?'*

7.1.1 A Theoretical Exploration of Job Stories and Jobs-to-be-Done

When investigating the different principles related to Jobs-to-be-Done to answer the first sub-question, we found that opinions on what can be considered as "Jobs-to-be-Done" vary widely. Due to the fragmented literature there is no clear clear consensus on what exactly constitutes Jobs-to-be-Done, and practitioners approach the theory in different ways. Due to this situation, a definition offered by Alan Klement seems fitting: "a collection of principles that helps you discover and understand the interactions between customers, their motivations, and the products they use" [10].

However, as described in section 3.2, for the first sub-question we can conclude that the main fundamental principle that defines Jobs-to-be-Done, is the idea that customers do not shop for a product, they are searching for the optimal solution for a Job they need to get done in their lives. Accepting this principle has some further implications for Jobs-to-be-Done practitioners. For a consumer, a market is not segmented by product categories, but as solutions for different Jobs. For example, a consumer looking for a solution to her unruly backyard, might consider a lawnmower, a gardening service, genetically modified grass or perhaps even concrete. For companies this means that to excel in the market, they should strive to understand the Jobs they want- or are addressing in the market, what their competition looks like, and aim to design a solution that helps the customer get that Job done optimally.

According to Jobs-to-be-Done, understanding a Job means understanding the situational and motivational contexts that drive customer behavior. Christensen argues that this is not easily uncovered through the analysis of data [40]. To put these principles into practice practitioners employ different approaches. To answer the second and third sub-questions, we first discuss these approaches and then reflect on the related concepts and artifacts.

As described in sections 3.3 and 3.4, we have identified two main high-level approaches to Jobs-to-be-Done: a *'qualitative approach'* and a *'quantitative approach'*. While these are not necessarily concretely defined methodologies, the split between qualitative and quantitative does represent the main disagreement between Jobs-to-be-Done thought leaders, and this has major implications for how the theory is put into practice.

Here, we consider the *'qualitative approach'* to be represented by the likes of Clayton Christensen and Alan Klement, who argue that the role of data in the innovation process should be de-emphasized and that customers cannot accurately express what they want. To investigate a product's Jobs-to-be-Done, proponents of the qualitative approach suggest interviewing customers who have recently started or stopped using the product, and investigate what drove them to make the switch. The qualitative approach emphasizes the importance of the emotional- over the functional dimension of Jobs. A core concept of the qualitative approach are *Job Stories*, which are used to convey the different motivational and situational contexts that relate to a Job.

The *'quantitative approach'* revolves around the Outcome-Driven-Innovation (ODI) methodology by Anthony Ulwick. While ODI acknowledges the risks associated with asking customers what they want, it operates from the assumption that when done correctly, quantitative techniques can be used to identify opportunities to innovate and help customers get Jobs done better. While recognizing the importance of the emotional dimensions of Jobs, a functional Job is central in ODI. A core concept in ODI are *Desired Outcomes*, which are the metrics that document what customers want to achieve at each step of getting a Job done [16].

As to which of these approaches is best suited for use in software requirements engineering, we see that each approach has its own benefits and disadvantages. Starting with the qualitative approach, its focus on emotional Jobs makes it more likely to find deep insights. However, in the context of functionally complex software products the quantitative approach with its focus on a core functional Job seems to be more appropriate. As Ulwick argues, this is because it is difficult to design an complex software product based on emotional needs [17].

However, many practitioners in the software industry are very interested in Job Stories, which can be considered a part of the 'qualitative' approach. This interest is likely due to Job Stories being specifically designed with software products in mind, and likely also partly due to the similarity to User Stories, which can help practitioners conceptualize how to apply it in their work. In our experience, the Desired Outcomes from ODI proved to be less appealing than Job Stories. Stakeholders at Stabiplan found that the repetitive use of 'minimize' gives the statements a negative feel. Furthermore, compared to Job Stories the contextual information that is conveyed through Desired Outcomes is limited.

Still, one advantage of ODI over the techniques from the qualitative approach is that it offers a much more comprehensive methodology. While one could hire Ulwick's consultancy firm Strategyn for help with applying ODI, the entire approach is described

in literature. Being a more recent development, the qualitative approach lacks such a cohesive structure and focus on how to put the concepts and techniques into practice.

Reflecting on this comparison we see that each approach has its own advantages and disadvantages. This brings us to the suggestion that an optimal approach to Jobs-to-be-Done is a combination of the two approaches, possibly relying on ODI for the structure and on Job Stories for the link with software development. However, thought-leaders from both approaches claim that theirs is the only valid way to practice Jobs-to-be-Done. This indicates that from a theoretical perspective, combining different approaches might not be a good idea.

In the second part of this discussion we will come back to this issue and investigate whether the behavior of practitioners reflects this theoretical chasm between the two approaches. For the second part we discuss how practitioners in the software industry apply the theory in their work. Our primary focus for this part is on the positioning of Job Stories and Jobs-to-be-Done amongst other techniques used to perform requirements engineering for software products.

7.1.2 Job Stories and Jobs-to-be-Done in the Software Industry

While not represented in scientific literature, practitioner literature does include various examples of the use of Job Stories and Jobs-to-be-Done in the context of software products. It is unfortunate that these examples are often lacking in quality, as for most, a detailed documentation of the actual process of documentation is missing. As a result, they often introduce an interesting concept or technique, but fail to provide the depth needed to support practitioners looking to put it in practice.

As described in section 4.1, notable examples include a hospital using Jobs-to-be-Done to guide its IT procurement, practitioners using Job Stories to analyze and optimize workflow for software solutions, implementing Job Stories as a direct replacement for User Stories, and using Job Stories as means to document requirements for software systems based on workshops with users. While we have not been able to find examples of the use of ODI in the context of software products, there are example where parts of the ODI method are applied, for instance the 'Opportunity Prioritization' technique [99]. Reflecting on these examples it should be noted that with the rising popularity of Job Stories and Jobs-to-be-Done in general, new experience reports from practitioners are frequently added. As such, it is possible that after performing this study, new insights have been developed.

For the fifth sub-question, we wanted to look at a higher level at what main usage scenarios can be identified for Job Stories and Jobs-to-be-Done in the context of software products. As described in section 4.2, we identified three main scenarios in which Job Stories and Jobs-to-be-Done are applied: to create entirely new products, to improve existing products, and in a more general way to promote customer empathy among a team. Originally, our intention was to compare these main usage scenarios and determine which of the previously discussed approaches was most suitable for each scenario, and present different methods for using Job Stories for the scenarios. However, while there are some differences that make each approach more suitable for a certain usage scenario, we found that these are relatively minor and not necessarily restrictive differences.

Our analysis of the examples from practitioners seems to supports this finding, as practitioners do not necessarily stick to one of the approaches described in the first part of this discussion.

While thought-leaders present a polarized perspective on Jobs-to-be-Done, where only their approach will lead to success, practitioners are more pragmatic and combine elements from each approach to fit their situations. In part, this is likely due to the differences between the two approaches not necessarily being clear without an extensive study of the literature. It does however set a precedent, that the distinction between the two main approaches does not need to be as strict as suggested by thought-leaders. This will come back to this observation in the third part of this discussion, which covers the proposing, applying and evaluating of a concrete method.

To gain insight into how such a method could add the most value, we investigated how Job Stories and Jobs-to-be-Done are positioned in the landscape of requirements engineering for software products. This is a relevant question because it explores what makes Job Stories and Jobs-to-be-Done unique compared to other existing techniques. To maximize the added value of our method, it should reflect those aspects that make the theory unique.

As discussed in section 4.3, Jobs-to-be-Done is not an entirely unique addition to the software industry, since there is some overlap with most related techniques such as 'User-Centered Design' or 'Goal-Oriented Requirements Engineering'. However, this does not mean that Job Stories and Jobs-to-be-Done are not valuable additions to the field of software requirements engineering.

Overall, we see that Jobs-to-be-Done is mainly differentiated by how far it goes in only focusing on the "problem space" of innovation. The previously mentioned User-Centered Design or Goal-Oriented Requirements Engineering also stress the importance

of focusing on the users problems or goals, but are more broadly oriented at supporting the development process, which naturally includes the building of solutions.

On a lower level, we have seen that Job Stories are pitched as an replacement or alternative for User Stories. As discussed in section 4.3.2, Alan Klement argues that User Stories are flawed because they do not convey information about the situational and motivational contexts behind the story [6]. Because of this lack of context and a focus on persona's, practitioners have expressed concerns that User Stories could induce incorrect assumptions about the true motivation of the user in question. Furthermore, practitioners report that User Stories are too often formulated with a specific solution.

Regarding the question of which story format is better, it is important to consider that each format can result in both good and poorly written stories. In fact, it is theoretically possible to write a User Story in such a way that it conveys most of the contextual information a Job Story focuses on. However, to write a story that contains this information a Jobs-to-be-Done approach is more effective than the persona based research that is often associated with User Stories. This is because while we cannot dismiss the benefits that can be achieved by using personas correctly, unlike Jobs-to-be-Done it does not focus on identifying the causality that drives customer behavior. Furthermore, incorrect use of persona's can surely perpetuate invalid assumptions.

Reflecting on the role of Job Stories and Jobs-to-be-Done relative to existing techniques used to perform software requirements engineering, we find that they add value as tools to uncover and represent the problems targeted by new software development. Due to this focus, Job Stories can be considered more of an addition to the portfolio of existing techniques instead of a replacement. While this should be investigated in further research we can hypothesize that synergistic effects can be achieved by using Job Stories to describe the problem, and other techniques to support the design and development of a solution.

7.1.3 Proposing the Integrated Job Story Method

For this third part of the discussion, we reflect on sub-questions 8, 9 and 10, related to the Integrated Job Story method, its application and evaluation.

As described in section 4.5, the Integrated Job Story method is proposed as an answer for the eighth sub-question. Instead of adhering to a single "approach" to Jobs-to-be-Done, we pragmatically combined available Jobs-to-be-Done literature into a method that captures both the most important Jobs-to-be-Done and Job Story activities, as well as the utilized artifacts. As such, the method uses the structure from ODI and

relies on Job Stories for analysis and presenting the results for further development. To accommodate practitioners that want to work with Job Stories but would also like to stick to ODI as much as possible, the method contains an alternative path that emphasizes Desired Outcomes whilst still using Job Stories to convey the results.

Proponents of either the qualitative of quantitative approach might regard the method as a somewhat arbitrary collection of different elements from Jobs-to-be-Done theory. With some thought-leaders denouncing other approaches and essentially arguing that theirs is the only valid way to practice Jobs-to-be-Done, this is a somewhat understandable point of view. However, given the novelty of Job Stories and Jobs-to-be-Done and the lack of comprehensive methods for its use in the software industry, we find it appropriate to take this experimental approach.

To find an answer for sub-question 9, the Integrated Job Story method was applied in a case study at Stabiplan, which was described in chapters 5 and 6, to help Stabiplan determine what high-level requirements to address with new Revit apps related to radiators. Through exploratory interviews with Stabicad customers, we uncovered a set of (functional) Jobs that we further described using Job Stories. We validated and prioritized these Jobs and Job Stories using an online survey we conducted with Revit users that make up the target audience for the new apps. The final phase of the project was centered around determining which of the high-priority Job Stories to address with the new apps. To do this, we categorized the prioritized Job Stories based on the deeper, non-functional Jobs they belong to, and decided to target each of the two non-functional Jobs who's Job Stories had the highest 'Opportunity Score' with an app. For each of these apps, we then created 'Project Brief' documents.

From a research perspective, our goal for the case study was to gauge the value of the method for Stabiplan, and use this to come to broader conclusions on the value of the method and of Job Stories and Jobs-to-be-Done for the field of software requirements engineering. To answer sub-question 9 and get a sense of the value for Stabiplan, we investigated the impact of the method on the business processes at Stabiplan. Since Jobs-to-be-Done is often touted to deliver organizational benefits that are broader than "just" Product Management, we investigated the impact on the business processes through interviews with various key stakeholders at Stabiplan, namely: Product Management, Development and Marketing. The results of the stakeholder evaluation are described in section 6.2.3. In short, the method yielded positive results but has some inherent problems that limit its practical applicability.

Product Management found the method to be useful for exploring and prioritizing the needs in new market segments, and to creatively design a solution to address these needs. Furthermore, Product Management found Job Stories to be effective for scoping a

project and conveying their customers' priorities to internal stakeholders. Being problem oriented instead of solution oriented, the results of the method were too high level to be relevant for Development. For Marketing, the Project Briefs were very useful as it helps them create marketing material that conveys why a product is valuable for its target audience.

However, practical applicability is limited by the time required to perform the method, and the fact that the method heavily relies on customer participation. In the case study we experienced great difficulty to find suitable participants for both the interviews and the survey. While this makes the method unsuitable for use at a larger scale, Stabiplan is investigating ways to use its online communities to aid with the data collection and make the method more feasible.

7.2 Answering the Main Research Question

In this section we formulate an answer for the following main research question:

'What is the value of Job Stories and Jobs-to-be-Done in the context of requirements engineering for software products?'

Based on the answers to the sub-questions provided above, we can conclude that in the context of software requirements engineering, Jobs-to-be-Done should be seen as a technique for investigating and understanding the problems a new software solution should solve for the customer. While this is not a foreign concept in the field of software requirements engineering, Jobs-to-be-Done is differentiated in three main ways:

1. Its focus is *solely* on the problem side of innovation and should not be used to design solutions.
2. It requires looking beyond the scope of a single system at the bigger picture of what the customer is trying to achieve. As such, the customer is the unit of analysis when investigating a problem, not a system.
3. It stresses that input for requirements should come from customers themselves, as most data cannot be used to uncover what drives customer behavior, and existing preconceptions and assumptions about what a customer wants are likely wrong.

As such, a Jobs-to-be-Done project that can be considered as "proper", always involves an extensive research process that includes interviews with customers to be able to

extract and describe the Jobs to be done. Therefore, properly applying Jobs-to-be-Done can be time-consuming and expensive.

This resource intensity means that for most organizations, it is not feasible to apply Jobs-to-be-Done for every project. However, there are some scenarios in which the benefits are likely to outweigh the costs. As discussed in section 6.2.3, one such scenario is when a company is looking to enter a new market or market segment. Here, Jobs-to-be-Done can help build an understanding of the needs in a market, and strategize on how to best address these needs.

this does not mean that for companies that are experienced and familiar with their market segment, it is not valuable to periodically take a step back and investigate their customers' Jobs to be done, and how their products help them make progress. The field of management science offers many examples of companies that got too comfortable as the producer of a certain product, only to be beaten by a disruptive innovation that gets the Job done better [41].

Reflecting on Job Stories specifically, we found that to be most valuable, Job Stories should be combined with proper Jobs-to-be-Done research to provide them with appropriate content. While User Stories can technically be written to contain mostly the same information as a Job Story, due to its focus on situations and motivations, the Job Story format is much more appropriate for highlighting different aspects of a Job.

However, this does not mean that Job Stories are generally better than User Stories. While the difference is subtle and it matters greatly how the stories are written, with its focus on user roles and concrete actions, the User Story format is better suited to describe specific behaviors of a system. While this is an interesting subject for further research we can conclude that Job Stories should not be seen as a replacement for User Stories. Instead, Job Stories should be seen as a tool for the problem space of innovation, while User Stories are best used in the solution space.

7.3 Conclusions

Jobs-to-be-Done is an innovation theory that is based on the fundamental principle that customers hire products and services to help them get Jobs done. This means that to create successful products, innovators should understand the Jobs they are addressing. To help put the theory into practice in the context of software products, Job Stories were pitched in 2013 as an alternative or replacement to User Stories. With its format of: '**When** [*situation*], **I want** [*motivation*], **so that** [*expected outcome*]', Job Stories

emphasize the situational and motivational contexts that are behind a customer requirement and help convey the Job they need to get done.

In this study we propose an *Integrated Job Story method* that describes how Job Stories and the theory on Jobs-to-be-Done can be used to define high-level requirements for software development projects. The Integrated Job Story method consists of five phases:

P1 - Interview phase: exploratory interviews are conducted with (prospective) customers in order to uncover their goals, their current way of achieving these goals, and the problems that exist in their as-is situation;

P2 - Analysis phase: the workflow, context and motivations of the interviewees are analyzed to formulate initial Jobs and Job Stories;

P3 - Survey phase: in order to validate the results of the analysis phase, a survey is conducted with a larger sample of the target audience;

P4 - Prioritization phase: the survey results are analyzed to determine which Jobs present the largest opportunity for innovation, and;

P5 - Project definition phase: the Jobs and Job Stories for development are selected and a *project brief* is created that can facilitate the follow-up development project.

To illustrate and evaluate the method, we apply it in a case study at the product software company Stabiplan, where it is used to define the high-level requirements for new apps. An evaluation has indicated that the method is useful as a means to explore and prioritize customer needs, and that Job Stories are useful artifacts for both product designers and marketeers. An important disadvantage of the method was identified in its time-consuming nature, and its reliance on customer participation.

Reflecting on the case study, we conclude that in the context of the Integrated Job Story method, Job Stories and Jobs-to-be-Done are most valuable when exploring new market segments as this is where its application is likely to yield the highest return on investment.

Through its theoretical exploration of Job Stories and Jobs-to-be-Done and its practical application of the Integrated Job Story method, this study has made important steps towards objectively assessing the role and value Job Stories and Jobs-to-be-Done add in the software industry. However, due to the limited scope of this study, more research is needed to come to a more comprehensive and generalizable value judgment.

7.3.1 Generalizability

Reflecting on the generalizability of the results of this study, it is important to consider that this was an exploratory study that does not make claims that should necessarily be generalized to a broader context like the software industry as a whole. While the main research question does refer to the software industry in general, we present our findings from the case study as a contribution to answering this question, not as a comprehensive answer.

While the case study has surely identified some fundamental strengths and weaknesses of the Integrated Job Story method, we recognize that relative weight of these strengths and weaknesses will differ based on the context of each case in which it can be applied, and do not necessarily mean to extrapolate our findings to Job Stories and Jobs-to-be-Done as a whole.

From a practitioner perspective, this means that those that might be looking to apply the Integrated Job Story method in their work, should consider how their context influences the efficacy and efficiency of the method, and determine whether- or what variant of the method to apply. From a scientific perspective, it means that this study does not propose an universal solution or conclusion. Instead, it should be seen as context-driven research aimed at building a body of knowledge that helps practitioners determine what to use in their context, and helps close the gap between academic research and industry [112].

7.3.2 Future Work

Since Job Stories and Jobs-to-be-Done are a new and relatively unexplored topic in the field of Software Requirements Engineering, there are several aspects that should be explored in future research.

First of all, to acquire a broader perspective on the value of Job Stories and Jobs-to-be-Done in the context of software products, there is a general need for more well-documented case studies. These future studies should not only feature the Integrated Job Story method but also experiment with alternative methods for applying Job Stories and Jobs-to-be-Done. This is important to help facilitate an holistic evaluation of the technique, and identify the optimal method.

A second topic for further research is the integration of Job Stories with existing techniques that are used in the software industry. It would be interesting to investigate whether it would be valuable to use Job Stories and Jobs-to-be-Done in tandem with problem oriented software development approaches such as User-Centered-Design and

Goal-Oriented Requirements Engineering. Particularly interesting is the relationship between Job- and User Stories. Originally pitched as a replacement or alternative for User Stories, Job Stories are now often considered to be entirely different in scope: Job Stories are used to investigate and describe problems, while User Stories are used to design a solution. Further research needs to be done on how this relationship would manifest itself in practice.

More generally, we see that practitioner literature on Job Stories and Jobs-to-be-Done is still very much under development. It should therefore be valuable for the scientific community to watch these developments and if relevant incorporate them in future scientific work.

Bibliography

[1] Mike Cohn. *User stories applied: For agile software development.* Addison-Wesley Professional, 2004.

[2] Mohamad Kassab. The changing landscape of requirements engineering practices over the past decade. In *Empirical Requirements Engineering (EmpiRE), 2015 IEEE Fifth International Workshop on*, pages 1–8. IEEE, 2015.

[3] Xinyu Wang, Liping Zhao, Ye Wang, and Jie Sun. The role of requirements engineering practices in agile development: an empirical study. In *Requirements Engineering*, pages 195–209. Springer, 2014.

[4] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn EM van der Werf, and Sjaak Brinkkemper. The use and effectiveness of user stories in practice. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 205–222. Springer, 2016.

[5] Intercom. *Intercom on Jobs-to-be-Done.* 2016. ISBN 978-0-9861392-3-9.

[6] Alan Klement. Replacing the user story with the job story, November 2013. URL `https://jtbd.info/replacing-the-user-story-with-the-job-story-af7cdee10c27#.w6zfetkf8`.

[7] Christopher N Chapman and Russell P Milham. The personas' new clothes: methodological and practical arguments against a popular method. In *Proceedings of the human factors and ergonomics society annual meeting*, volume 50, pages 634–636. SAGE Publications Sage CA: Los Angeles, CA, 2006.

[8] Alan Klement. Focus on causality, skip personas, March 2013. URL `http://alanklement.blogspot.nl/2013/03/focus-on-relationships-skip-personas.html`.

[9] Clayton M Christensen, Taddy Hall, Karen Dillon, and David S Duncan. Know your customers' "jobs to be done". *Harvard Business Review*, (September):1,

2016. ISSN 00178012. URL http://search.proquest.com/docview/1817078151?accountid=14797%0Ahttps://sfx.nelliportaali.fi/nelli07b/?url_ver=Z39.88-2004&rft_val_fmt=info:ofi/fmt:kev:mtx:journal&genre=article&sid=ProQ:ProQ%3Aabiglobal&atitle=Know+Your+Customers%27+%22Jobs+to+Be+Done%22.

[10] Alan Klement. *When Coffee and Kale Compete*. NYC Publishing, 2016. ISBN 1534873066.

[11] Clayton M Christensen, Scott Cook, and Taddy Hall. Marketing malpractice. *Make Sure AllYour Products Are Profitable*, page 2, 2005.

[12] Sjaak Brinkkemper. Method engineering: engineering of information systems development methods and tools. *Information and software technology*, 38(4): 275–280, 1996.

[13] Clayton M Christensen, Scott D Anthony, Gerald Berstell, and Denise Nitterhouse. Finding the right job for your product. *MIT Sloan Management Review*, 48(3):38, 2007. ISSN 15329194. URL http://search.proquest.com.library.capella.edu/docview/224962902?accountid=27965.

[14] Clayton M Christensen, Karen Dillon, Taddy Hall, and David S Duncan. Competing against luck: The story of innovation and customer choice. 2016.

[15] Anthony W Ulwick. Turn customer input into innovation. *Harvard business review*, 80(1):91–7, 2002.

[16] Anthony W. Ulwick and Lance A. Bettencourt. Giving customers a fair hearing. *MIT Sloan Management Review*, 49(3):62–68, 2008. ISSN 15329194.

[17] Anthony Ulwick. Emotional vs functional jobs: The basics of messaging, August 2016. URL http://www.marketingjournal.org/emotional-vs-functional-jobs-the-basics-of-messaging/.

[18] Anthony Ulwick. Mapping the job-to-be-done, January 2017. URL https://medium.com/@Ulwick/mapping-the-job-to-be-done-45336427b3bc.

[19] Alan Klement. 5 tips for writing a job story, November 2013. URL https://jtbd.info/5-tips-for-writing-a-job-story-7c9092911fc9#.zb5xnlq8q.

[20] Alan Klement, 2014. URL http://jobstobedone.org/radio/alan-klement-on-jobs-stories/.

[21] Alan Klement. Designing features using job stories, 2013. URL https://blog.intercom.com/using-job-stories-design-features-ui-ux/.

[22] 2017. URL `https://medium.com/tag/jobs-to-be-done/`.

[23] 2017. URL `https://www.jtbd.info`.

[24] 2017. URL `https://wwww.jobstobedone.xyz`.

[25] 2017. URL `https://www.jtbdinstitute.com`.

[26] 2017. URL `https://www.jobstobedone.org`.

[27] Robert K Yin. Case study research: design and methods. applied social research methods series, 5. *Biography, Sage Publications, London*, 1994.

[28] Joseph Alois Schumpeter. *Socialism, capitalism and democracy*. Harper and Brothers, 1942.

[29] Rajesh K Chandy and Gerard J Tellis. The incumbent's curse? incumbency, size, and radical product innovation. *Journal of marketing*, 64(3):1–17, 2000.

[30] Lisa D McNary. The system of profound knowledge: a revised profile of managerial leadership. *Leadership & Organization Development Journal*, 18(5): 229–235, 1997.

[31] W Edwards Deming. Improvement of quality and productivity through action by management. *Global Business and Organizational Excellence*, 1(1):12–22, 1981.

[32] Amos Tversky and Daniel Kahneman. Advances in prospect theory: Cumulative representation of uncertainty. *Journal of Risk and uncertainty*, 5(4):297–323, 1992.

[33] Gary Klein. Naturalistic decision making. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 50(3):456–460, 2008.

[34] Theodore Levitt. *Marketing Imagination: New*. Simon and Schuster, 1968.

[35] Theodore Levitt. Marketing myopia. *Harvard business review*, 38(4):24–47, 1960.

[36] Wikipedia. Solution selling, October 2011. URL `https://en.wikipedia.org/wiki/Solution_selling`.

[37] Michael T Bosworth. *Solution selling: Creating buyers in difficult selling markets*. McGraw-Hill, Inc., 2002.

[38] Arun Sharma, Gopalkrishnan R Iyer, and Heiner Evanschitzky. Personal selling of high-technology products: The solution-selling imperative. *Journal of Relationship Marketing*, 7(3):287–308, 2008.

[39] Lance A Bettencourt, Robert F Lusch, and Stephen L Vargo. A service lens on value creation. *California Management Review*, 57(1):44–66, 2014.

[40] Clayton M. Christensen, Karen Dillon, Taddy Hall, and David S. Duncan. *Competing Against Luck: The Story of Innovation and Customer Choice*. Harper Business, 2016. ISBN 0062435612, 9780062435613.

[41] Clayton M. Christensen. *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail*. Harvard Business School Press, Boston, MA, USA, 1997. ISBN 0-87584-585-1.

[42] Alan Klement. The illusion of measuring what customers want, May 2017. URL `https://jtbd.info/the-illusion-of-measuring-what-customers-want-3672a7892eb`.

[43] Gerald Berstell and Denise Nitterhouse. Looking " outside the box " customer cases help researchers predict the unpredictable. 9:1–11, 1997.

[44] Andrej Balaz. A template for jtbd interviews, 2015. URL `https://jtbd.info/jobs-to-be-done-interview-template-30421972ab2a#.iga3bvo3z`.

[45] Gerald Berstell and Denise Nitterhouse. Asking all the right questions: Exploring customer purchase stories can yield surprising insights. *Marketing Research*, 13:14–21, 2001.

[46] Kathleen M Eisenhardt and Melissa E Graebner. Theory building from cases: Opportunities and challenges. *Academy of management journal*, 50(1):25–32, 2007.

[47] Alan Klement. Your job story needs a struggling moment. `goo.gl/vsRC1d`, 2016.

[48] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn E. M. van der Werf, and Sjaak Brinkkemper. Improving agile requirements: the quality user story framework and tool. *Requirements Engineering*, 21(3):383–403, Sep 2016. ISSN 1432-010X. doi: 10.1007/s00766-016-0250-x. URL `https://doi.org/10.1007/s00766-016-0250-x`.

[49] Anthony Ulwick and Des Traynor. Strategyn's tony ulwick on jobs-to-be-done. `https://blog.intercom.com/podcast-tony-ulwick-on-jobs-to-be-done/#pivot-fail`, 2015.

[50] Strategyn. Innovation track record. URL `https://strategyn.com/our-results/`.

[51] Mike Boysen. The data-driven job story, September 2016. URL https://medium.com/effective-crm/the-data-driven-job-story-7b80eb8ac97.

[52] Kimon Paxinos. Re: Anything goes, relevancy and why i struggle with christensen's jobs theory, April 2017. URL https://www.linkedin.com/pulse/anything-goes-relevancy-why-i-struggle-christensens-jobs-derek-blair#pulse-comments-comment-urn:li:comment:(article:9131736068719516449,6258225835170500608).

[53] Anthony Ulwick. Customer segmentation is soured by milkshake marketing, January 2013. URL https://strategyn.com/2013/01/16/market-segmentation-is-soured-by-milkshake-marketing/.

[54] Lance A Bettencourt and Anthony W Ulwick. The customer-centered innovation map. *Harvard Business Review*, 86(5):109, 2008.

[55] Strategyn, 2017. URL https://strategyn.com/outcome-driven-innovation-process/.

[56] David Silverstein, Phil Samuel, and Neil DeCarlo. The innovator's toolkit: Technique 1 - jobs to be done, 2017. URL http://innovatorstoolkit.com/content/technique-1-jobs-be-done.

[57] Anthony Ulwick and Perrin Hamilton. The jobs-to-be-done growth strategy matrix. Technical report, Strategyn, 2016.

[58] Jacquelyn S Hunt, Richard F Gibson, John Whittington, Kitty Powell, Brad Wozney, and Susan Knudson. Guide for developing an information technology investment road map for population health management. *Population health management*, 18(3):159–171, 2015.

[59] Pete Knox, Jacquelyn Hunt, Brad Wozney, James Jerzak, and Kathy Kerscher. Bellin health: A model for population health. *The Journey Never Ends: Technology's Role in Helping Perfect Health Care Outcomes*, page 217, 2016.

[60] Kyle Fiedler. Converting to jobs stories, January 2015. URL https://robots.thoughtbot.com/converting-to-jobs-stories.

[61] Meag Tessman. Job stories in practice, December 2016. URL https://medium.theuxblog.com/job-stories-in-practice-f833265afd13#.59umklwgx.

[62] MA Bloemberg. Exploring the explore page: redesigning through reflective transformative design process. B.S. thesis, University of Twente, 2016.

[63] Hutch Carpenter. A method for applying jobs-to-be-done to product and service design, January 2013. URL `https://bhc3.com/2013/01/15/a-method-for-applying-jobs-to-be-done-to-product-and-service-design/`.

[64] Alan Dix. *Human-computer interaction*. Springer, 2009.

[65] Jan Gulliksen, Bengt Göransson, Inger Boivie, Stefan Blomkvist, Jenny Persson, and Åsa Cajander. Key principles for user-centred systems design. *Behaviour and Information Technology*, 22(6):397–409, 2003.

[66] Donald A Norman and Stephen W Draper. User-centered design. *New Perspectives on Human-Computer Interaction*, 1986.

[67] Mica R Endsley. *Designing for situation awareness: An approach to user-centered design*. CRC press, 2016.

[68] Zahid Hussain, Martin Lechner, Harald Milchrahm, Sara Shahzad, Wolfgang Slany, Martin Umgeher, and Peter Wolkerstorfer. Agile user-centered design applied to a mobile multimedia streaming application. In *Symposium of the Austrian HCI and Usability Engineering Group*, pages 313–330. Springer, 2008.

[69] Laurie Williams. The xp programmer: the few-minutes programmer. *IEEE Software*, 20(3):16, 2003.

[70] Gunther Ruhe and Moshood Omolade Saliu. The art and science of software release planning. *IEEE software*, 22(6):47–53, 2005.

[71] Günther Ruhe. Software release planning. *Handbook of software engineering and knowledge engineering*, 3:365–394, 2005.

[72] Hai Ton. A strategy for balancing business value and story size. In *Agile Conference (AGILE), 2007*, pages 279–284. IEEE, 2007.

[73] Roger S Pressman. *Software engineering: a practitioner's approach*. Palgrave Macmillan, 2005.

[74] Chetankumar Patel and Muthu Ramachandran. Story card based agile software development. *International Journal of Hybrid Information Technology*, 2(2): 125–140, 2009.

[75] Bill Wake. Invest in good stories and smart tasks, August 2003. URL `http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/`.

[76] Gianpi. Job stories are great, but they only replace badly written user stories, February 2015. URL `https://medium.com/@_gianpi_`

/job-stories-are-great-but-they-only-replace-badly-written-user-stories-8a8eea7
.dy8tsgu6s.

[77] Kamil Nicieja. The most common user story mistake, Augustus 2015. URL https://medium.com/@kamil/the-most-common-user-story-mistake-68f1d7dabc43#.sqqs50vmj.

[78] Nils Christian Haugen. An empirical study of using planning poker for user story estimation. In *Agile Conference, 2006*, pages 9–pp. IEEE, 2006.

[79] Scott Ambler and Mark Lines. *Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise*. IBM Press, 2012.

[80] Manuel Imaz and David Benyon. How stories capture interactions. In *International Conference on Human-Computer interaction*, pages 321–328. IOS Press, 199.

[81] Kurt Bittner. *Use case modeling*. Addison-Wesley Longman Publishing Co., Inc., 2002.

[82] Björn Regnell, Kristofer Kimbler, and Anders Wesslen. Improving the use case driven approach to requirements engineering. In *Requirements Engineering, 1995., Proceedings of the Second IEEE International Symposium on*, pages 40–47. IEEE, 1995.

[83] I Jacobsen. The use-case construct in objectoriented design. *Scenario-based Design: Envisioning Work and Technology in System Development. Wiley, NY*, 1995.

[84] Microsoft. Uml use case diagrams: Guidelines, 2005. URL https://msdn.microsoft.com/en-us/library/dd409432.aspx#Details.

[85] John M Carroll. *Scenario-based design: envisioning work and technology in system development*. John Wiley & Sons, Inc., 1995.

[86] Julio Cesar Sampaio do Prado Leite, Gustavo Rossi, Federico Balaguer, Vanesa Maiorana, Gladys Kaplan, Graciela Hadad, and Alejandro Oliveros. Enhancing a requirements baseline with scenarios. *Requirements Engineering*, 2(4):184–198, 1997.

[87] Colin Potts. Using schematic scenarios to understand user needs. In *Proceedings of the 1st conference on Designing interactive systems: processes, practices, methods, & techniques*, pages 247–256. ACM, 1995.

[88] Alistair G Sutcliffe, Neil AM Maiden, Shailey Minocha, and Darrel Manuel. Supporting scenario-based requirements engineering. *IEEE Transactions on Software Engineering*, 24(12):1072–1088, 1998.

[89] Chris Parsons. Scenarios are not acceptance criteria, November 2012. URL `http://chrismdp.com/2012/11/scenarios-are-not-acceptance-criteria/`.

[90] Alan Cooper. *The Inmates Are Running the Assylum*. Macmillan Publishing Co., 1999.

[91] Jack M Maness, Tomasz Miaskiewicz, and Tamara Sumner. Using personas to understand the needs and goals of institutional repository users. *D-Lib Magazine*, 14(9/10):1082–9873, 2008.

[92] Shamal Faily and Ivan Flechais. Persona cases: a technique for grounding personas. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2267–2270. ACM, 2011.

[93] Yen-ning Chang, Youn-kyung Lim, and Erik Stolterman. Personas: from theory to practices. In *Proceedings of the 5th Nordic conference on Human-computer interaction: building bridges*, pages 439–442. ACM, 2008.

[94] Inge van de Weerd and Sjaak Brinkkemper. Meta-modeling for situational analysis and design methods. In *Handbook of research on modern systems analysis and design technologies and applications*, pages 35–54. IGI Global, 2009.

[95] Axel Van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, pages 249–262. IEEE, 2001.

[96] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004. ISSN 1387-2532. doi: 10.1023/B:AGNT.0000018806.20944.ef. URL `http://link.springer.com/10.1023/B:AGNT.0000018806.20944.ef`.

[97] Eric SK Yu and John Mylopoulos. Understanding" why" in software process modelling, analysis, and design. In *Software Engineering, 1994. Proceedings. ICSE-16., 16th International Conference on*, pages 159–168. IEEE, 1994.

[98] Eric Yu and John Mylopoulos. Why goal-oriented requirements engineering. In *Proceedings of the 4th International Workshop on Requirements Engineering: Foundations of Software Quality*, volume 15, pages 15–22, 1998.

[99] Luke Johnson. Jobs to be done: A case study in the nhs. goo.gl/gpaBpx, September 2017.

[100] Inge Van De Weerd, Sjaak Brinkkemper, Richard Nieuwenhuis, Johan Versendaal, and Lex Bijlsma. Towards a reference framework for software product management. In *Requirements Engineering, 14th IEEE International Conference*, pages 319–322. IEEE, 2006.

[101] Roderick van Cann, Slinger Jansen, and Sjaak Brinkkemper. Stabiplan. In *Software Business Start-up Memories*, pages 76–87. Springer, 2013.

[102] EU BIM Task Group. About the eu bim task group, 2016.

[103] Salman Azhar. Building information modelling (bim): Trends, benefits, risks, and challenges for the aec industry. *Management in Engineering*, 11(3):241–252, 2011.

[104] Stabiplan. Bim faq, 2017. URL https://www.stabiplan.com/en-us/page/bim-faqs.

[105] National BIM Standard. National bim standard - united states, October 2014. URL https://web.archive.org/web/20141016190503/http://www.nationalbimstandard.org:80/faq.php#faq7.

[106] Stabiplan. Life cycle management, January 2017. URL https://www.stabiplan.com/en-us/products/bim-consultancy/life-cycle-management/.

[107] Stabiplan. Stabicad, 2016. URL https://www.stabiplan.com/en-us/products/bim-software/.

[108] Stabiplan. Mepcontent subscription, November 2016. URL https://www.stabiplan.com/en-us/products/mepcontent/mepcontent-subscription/faq/.

[109] Josh Braun. What i learned about why people buy after interviewing 30 basecamp customers, March 2017. URL https://medium.com/sell-without-selling-your-soul/what-i-learned-about-why-people-buy-from-interviewing-30-customers-at-basecamp-

[110] Anthony Ulwick. Outcome-based segmentation. Technical report, 2014. URL https://strategyn.com/outcome-driven-innovation-process/market-segmentation/#form-outcome-based-segmentation.

[111] Ron Garland. The mid-point on a rating scale: Is it desirable. *Marketing bulletin*, 2(1):66–70, 1991.

[112] Lionel Briand, Domenico Bianculli, Shiva Nejati, Fabrizio Pastore, and Mehrdad Sabetzadeh. The case for context-driven software engineering research: Generalizability is overrated. *IEEE Software*, 34(5):72–75, 2017.

Appendices

Functional Jobs and Desired Outcomes

In this section the different JTBDs and related Desired Outcomes are included that were uncovered during the exploratory interviews.

1. Help me configure radiators:
   (a) Minimize the chance that I need to check external documentation.
   (b) Minimize the differences between the values that the radiator produces according to Revit and the manufacturer.
   (c) Minimize the amount of extra information that I need to put in.
   (d) Minimize the chance that a fabrication specific radiator I need is not available.
   (e) Minimize the chance that a fabrication specific bottom connection block I need is not available.
   (f) Minimize the chance that a fabrication specific thermostatic valve I need is not available.
   (g) Minimize the discrepancy between the radiator's heating power in Watts, and the flow and return temperatures.
   (h) Minimize the amount of times I need to click when configuring a radiator.
   (i) Minimize the time it takes to determine which radiators will produce the required heat.
2. Help me place radiators:
   (a) Minimize the effort it takes to repeatedly place the same radiator.
   (b) Minimize uncertainty when determining at which side of the radiator connection material should be placed.
   (c) Minimize difficulties that occur when moving connection material to one side of the radiator to another.

(d) Minimize the differences between the representation of a radiator in Revit and in reality.

3. Help me model piping systems:

   (a) Minimize complications associated with making changes to the pipe system at a later stage.

   (b) Minimize the chance that pipes will not connect automatically.

   (c) Minimize the amount of generic fittings needed to model a piping system.

   (d) Minimize the chance of errors in the diameters of the pipes in a piping system.

   (e) Minimize the effort associated with getting the pipes of a piping system into the correct positions.

   (f) Minimize the differences between the piping in a model, and how the pipes will actually be installed.

   (g) Minimize the amount of tries it takes to connect a fittings to an connection block/bottom block.

4. Help me create bills of materials:

   (a) Minimize the chance that the names of the articles in bill of material list, do not match the names used by manufacturers or suppliers.

   (b) Minimize the amount of unnecessary information included in article names.

   (c) Maximize the legibility of bills of materials when they are included on a drawing.

   (d) Minimize the length of article names.

   (e) Minimize the chance that the bill of materials generated from a model does not contain the actual materials that need to be ordered.

   (f) Maximize the possibility to generate a material list based on self-defined selections.

   (g) Minimize the effort required to process my material list into an ERP system.

5. Help me perform calculations with my model:

   (a) Minimize the chance that problems occur due to the calculations made based on my model not meeting certain norms.

6. Help me enrich my models with additional information:

   (a) Minimize the chance that tags are placed when this is not intended.

   (b) Minimize the chance that tags are placed at a place where this is not intended.

(c) Minimize the amount of unnecessary information included in tags.

(d) Minimize the degree to which tags are difficult to read when the scale of the model decreases.

(e) Minimize the amount of tags needed to convey the required information, e.g. by combining related tags.

(f) Minimize the effort required to add tags to a model.

(g) Minimize the effort required to organize tags in a model.

(h) Minimize the chance that it is not possible to incorporate the required information into a tag.

7. Help me adhere to standards and agreements in my industry:

(a) Minimize complications associated with exporting my model in the Industry Foundation Classes (IFC) file format.

(b) Minimize the chance that the assembly codes from the articles in my model do not adhere to industry information delivery agreements, e.g. NL-SfB.

(c) Minimize doubt caused by the symbolic representation of a model.

8. Help me find the right content for my model:

(a) Minimize the chance that the links provided by MEPcontent to manufacturer websites are not active.

(b) Minimize the chance that the specific content I need is not available.

(c) Minimize the chance that the generic content I need is not available.

(d) Minimize the chance that a specific radiator I need is not available.

(e) Minimize the chance that a specific connection block/bottom block I need is not available.

(f) Minimize the chance that a specific thermostatic valve I need is not available.

(g) Minimize the time it takes to determine whether the content I need is available.

9. Help me work better:

(a) Minimize the time it takes to learn or become re-acquainted with how the software works.

(b) Minimize the amount of times the software interrupts my work with unwanted tips.

(c) Minimize the amount of errors I make that could be prevented.

(d) Minimize the feeling I have that I am not using the software to its full potential.

Job Map

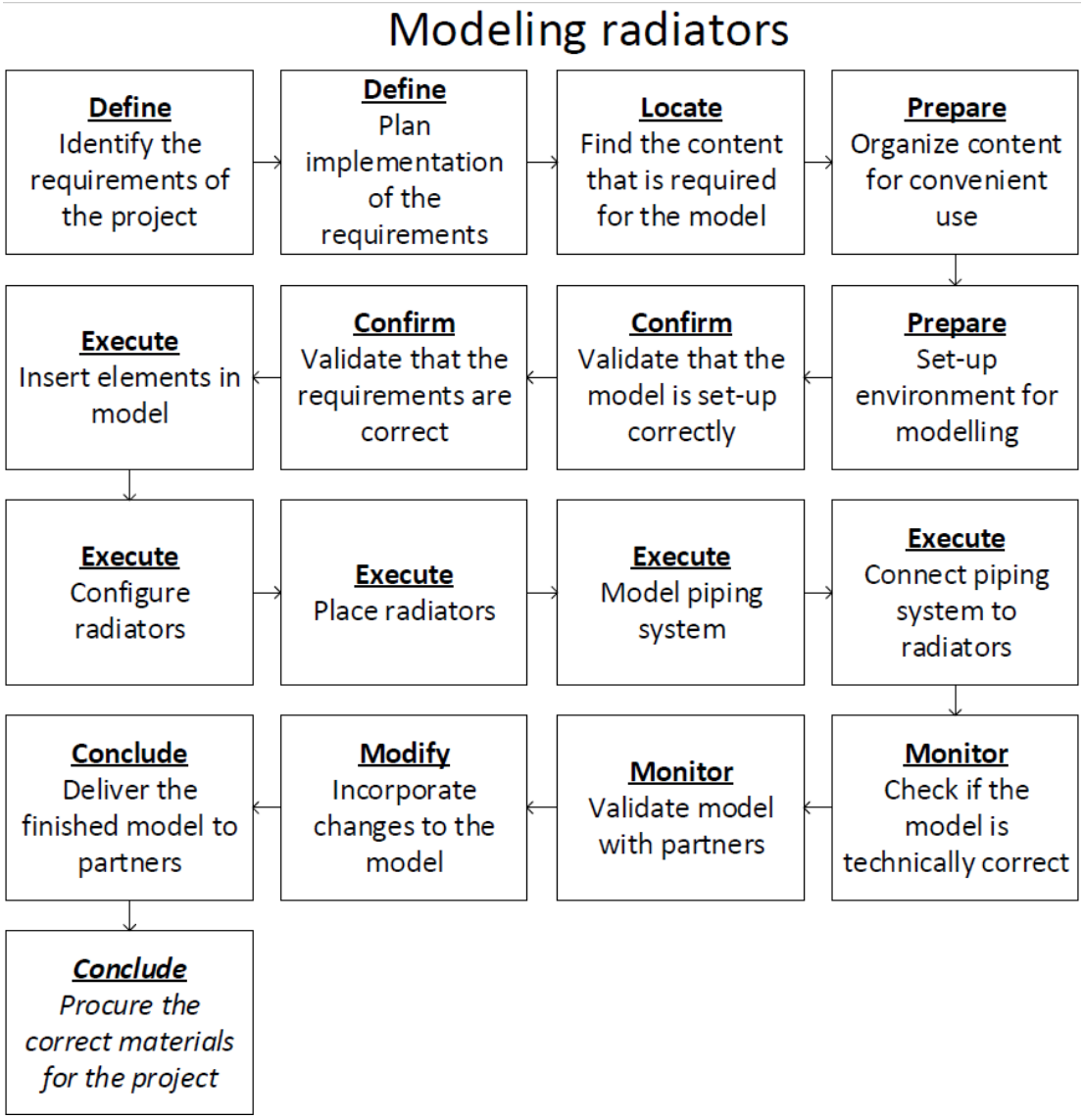

FIGURE 1: Job Map modeling radiators

Job Stories

1. Help me configure radiators:

   (a) When I am configuring a radiator, I want to have access to a complete library of fabrication parts (e.g. radiators, bottom connection blocks, thermostatic valves), so that my model can include the correct elements.

   (b) When I am configuring a radiator, I want to have access to a complete library of generic elements (e.g. radiators, bottom connection blocks, thermostatic valves), so that my model can include the correct elements.

(c) When I am configuring a radiator, I want the system to provide me with all the information I need in that process, so that I don't have to look things up in external documentation (e.g. the width of a radiator in the specification from the manufacturer).

(d) When I am configuring a radiator and I am trying to produce a specific amount of heating power, I want to quickly determine what configuration of radiators will produce the required heat, so that I won't have to waste time looking for the optimal configuration.

(e) When I am configuring a radiator, I want the heating power of the radiator to be accurately represented in the flow and return temperatures, so that my model will correctly represent the amount of heat that is produced.

(f) When I am configuring a fabrication specific radiator, I want that the system to contain accurate information on the values of a radiator (e.g. voltage, heating power) that matches the specification from the manufacturer, so that I can be sure that my model is correct.

2. Help me place radiators:

(a) When I must repeatedly place the same radiator, I want to avoid repeating as many steps as possible, so that I can quickly place the same radiator multiple times and don' t have to spend more time than necessary.

(b) When I have placed a radiator and I discover that I have selected the wrong side for the connection material, I want to be able to easily move the connection material to the other side, so that I can quickly finish placing the radiator.

(c) When I have placed a radiator, I want it to have a clear symbolic representation in the model, so that it is always clear where the radiators are positioned.

(d) When I have placed a radiator, I want it to be correctly represented in my model (e.g. size, connection points), so that I get an accurate view of what it will look like in the real-world and can spot potential problems in advance.

3. Help me model piping systems:

(a) When I have modeled a piping system and something changes in the project that forces me to make changes to the piping system, I want to be able to change the pipe system easily, so that I won't have to model the whole system again.

(b) When I have modeled a piping system and radiators, I want to connect the piping system to the bottom connection blocks of the radiators as easily and

quickly as possible, so that I won't have to struggle getting the pipes in the perfect position to make the connection.

(c) When I am modeling a piping system and I need to use flexible pipes, I want to be able to easily place flexible pipes in my model, so that my model shows exactly how the pipes will be installed.

(d) When I am modeling a piping system, I want to be able to fine-tune the model at a level of detail that I am satisfied with, so that I feel I am delivering quality work.

(e) When I have modeled a piping system, I want to quickly make sure that the diameters of the pipes in the system are correct, so that I know that I won't have to correct an error at a later stage.

(f) When I am modeling a piping system and need to use a fitting, I want to have access to the fabrication specific fitting I need, so that it can be included in the bill of materials.

(g) When I am modeling a generic piping system, I want to have access to the generic fittings I need, so that I can use the correct fittings in the model.

4. Help me create bills of materials:

(a) When I have created a bill of materials, I want the names of the fabrication parts on the bill to match the names used by the manufacturers or suppliers, so that they can be ordered directly from the bill without resulting in the delivery of the wrong items.

(b) When I have created a bill of materials, I want the names of the articles on the bill to be as concise as possible, so that it is easier to read, especially when printed.

(c) When I must create a bill of materials for a large and complex project, I want to be able to create separate bills for different self-defined selections within the model, so that we can easily organize our ordering process into multiple smaller orders.

(d) When my model will be used for prefabricated construction (prefab), I want to be able to easily create the prefab sheets I need, so that I can save time and move on to the next task.

Paper

Jobs-to-be-Done Oriented Requirements Engineering: A Method For Defining Job Stories

Garm Lucassen[1], Maxim van de Keuken[1], Fabiano Dalpiaz[1], Sjaak Brinkkemper[1], Gijs Willem Sloof[2], and Johan Schlingmann[2]
{g.lucassen, m.a.r.vandekeuken, f.dalpiaz, s.brinkkemper}@uu.nl,
{g.w.sloof, j.schlingmann}@stabiplan.com

[1] Department of Information and Computing Sciences, Utrecht University, The Netherlands
[2] Stabiplan, Bodegraven, The Netherlands

Abstract. [Context and motivation] Goal orientation is an unrealized promise in the practice of requirements engineering (RE). Conversely, lightweight approaches such as user stories have gained substantial adoption. While critics highlight the limitations of user stories, *Job Stories* are emerging as an alternative that embeds goal-oriented principles by emphasizing situation, motivation and expected outcome. This new approach has not been studied in research yet. **[Question/Problem]** Scientific foundations are lacking for the Job Story artifact and there are no actionable methods for effectively applying Job Stories. Thus, practitioners may end up creating their own flavor of Job Stories that may fail to deliver the promised value of the Jobs-to-be-Done theory. **[Principal ideas/results]** We integrate theories from multiple Job Story authors to create a conceptual model of Job Stories and to construct a generic method for Jobs-to-be-Done Oriented RE. Applying our Job Story Method to an industry case study, we highlight benefits and limitations. **[Contribution]** Our method aims to bring Job Stories from craft to discipline, and provide systematic means for applying Jobs-to-be-Done orientation in practice and for assessing its effectiveness.

Keywords: Job Stories; Requirements Engineering; Agile Development; Jobs to be Done; Problem Orientation; Empirical Case Study

1 Introduction

In Requirements Engineering (RE), problem-orientation is a long-standing research domain that emphasizes the importance of defining problems and capturing the 'why' as opposed to defining solutions and capturing the 'what' [29,37,38]. Alongside the rise of agile software development, user stories have become increasingly more popular with adoption up to 55% [17]. Although user stories are intended to focus on the problem space and not the solution space [15,8], authors and practitioners report that user stories are too often formulated with a specific solution [25,13].

Recognizing this problem for new software product development, Alan Klement introduced a new paradigm for requirements formulation in 2013: Job Stories [20]. Job Stories capture requirements in an alternative format that emphasizes the motivational and situational context that drive customer behavior:

When *<situation>*, ***I want (to)*** *<motivation>*, ***so that (I can)*** *<expected outcome>*.

It is based on the foundational ideas of the Disruptive Innovation Theory's method *Jobs-to-be-Done* by Clayton Christensen [7], which is a collection of principles that helps you discover and understand the interactions between customers, their motivations and the products they use [21]. Although still an upcoming approach, there are many authors who propose different approaches to Jobs-to-be-Done. This is confusing for practitioners, who are unsure how to apply this new approach in their own environment and struggle to formulate relevant and useful Job Stories. In particular, Jobs-to-be-Done is difficult to apply in the context of software; as many of the examples are for physical products. To ameliorate this situation, this paper makes the following contributions:

1. We collected Jobs-to-be-Done literature and position Job Stories among its different approaches and best practices in the Background in Section 2
2. We conceptualize the Job Story and evaluate the quality of 131 Job Stories gathered from public sources in Section 3
3. We introduce the Integrated Job Story Method that integrates the different views on Jobs-to-be-Done and Job Story literature in Section 4
4. We evaluate the applicability of the Integrated Job Story Method with a case study on developing an app for a single Job in Sections 5 and 6

We review related literature in Sec. 7 and present conclusions and future work in Sec. 8

2 Background: Job Stories and Jobs-to-be-Done

Although Job Stories originate from Jobs-to-be-Done (JTDB), the major principles of Jobs-to-be-Done build upon a large body of literature from management science [21,7,23]. The most similar predecessor to Jobs-to-be-Done is the 1996 Outcome-Driven Innovation method (ODI) which focuses on understanding customers' *Desired Outcome*, i.e., what they want to achieve, instead of giving customers what they think they want [32].

Ten years later, Christensen introduced the central notion of Jobs-to-be-Done: *"When people find themselves needing to get a job done, they essentially hire products to do that job for them"* [4]. To design products that customer segments want to hire for their jobs, traditional methods for market segmentation based on customer type or product type are inappropriate because they do not explain the 'why' of customer behavior. This aligns with fundamental theories in RE on the importance of the 'why' for software (process) analysis [29,37]. To uncover the *real* driving force of customer behavior, innovators should investigate the Jobs that customers are struggling to get done [7].

For example, fast food restaurant customers buy milkshakes to solve two distinct Jobs: (i) to keep themselves occupied during a long early commute to work in the morning, or (ii) to satisfy their kids' appetite for sugar in the afternoon. To tailor the milkshakes to better satisfy these Jobs, the fast food restaurant decided to differentiate the types of milkshakes they sell: thicker, longer-lasting milkshakes to commuters in the morning, and a thinner, easier-to-consume milk shake for kids in the afternoon.

The milk shake example shows the suitability of Jobs-to-be-Done for physical products with straightforward functionality [1,4,5,7,32]. Alan Klement and Intercom.io propose a new paradigm for JTBD-based requirements called *Job Stories* [20]. Building upon Christensen's rejection of demographic segmentation, Klement argues that user

stories' emphasis on roles and their actions inhibit the discovery of the 'why' due to the many assumptions a personified role implies. Instead of putting the role central, Job Stories emphasize the motivational and situational context that drive customer behavior:

When *<situation>,* ***I want (to)*** *<motivation>,* ***so that (I can)*** *<expected outcome>.*

Despite the initial idea of Job Stories as an alternative to user stories, practitioners have started adopting Job Stories alongside user stories. As Klement reports: *"You can write a Job Story to define a problem and then write user stories as potential solutions to that Job Story"* [19]. Arguably, this makes the Job Story format particularly well suited for capturing problem-oriented epics from Scrum, which the product team further decomposes into solution-oriented user stories.

3 Conceptualizing Job Stories

The original template for Job Stories [20] has been customized by practitioners in different ways, yet they all comprise three main parts: (i) the triggering event or *situation* in which a problem arises, (ii) the *motivation* and goal to make a change to the situation and (iii) the resulting *expected outcome* that improves the situation [21]. Consider the following illustrative examples that we use throughout this section:

JS1 - **When** I am looking for a new task on the task board, **I want to** filter all tasks to only see ones associated with my skills, **so I can** see open tasks that I am able to complete.

JS2 - **When** an item does not have an estimate or has an estimate I'm not happy with, **I want to** be able to restart the estimation process and notify everyone, **so that** the team knows a particular item needs to be estimated upon.

JS3 - **When** my friend tags me in an unflattering photo, **I want** my Facebook work colleagues to not see me in that photo, **so that** I can maintain my professional reputation.

We attempt to build solid foundations for this RE artifact by performing a theory building exercise [10] resulting in a conceptual model of job stories. To do so, we conducted a thorough analysis of 131 Job Stories from publicly accessible sources and constructed a conceptual model of a valid Job Story as shown in Figure 1. In the following subsections, we elaborate on the decomposition of each part and report on the quality of the collected Job Stories by analyzing how well the collected Job Stories fulfill each part. Our analysis document is available online: https://goo.gl/2TUFhw.

3.1 Template

A Job Story should roughly follow the Job Story template introduced in Section 2. For example, the *I* can be replaced with a different stakeholder or omitted entirely. This template is the syntactic structure within which the situation, motivation and expected outcome are interspersed.

Analysis: 113 of the 131 Job Stories adhere to the template. Out of the non-compliant stories, 6 added a role before the situation, 6 did not include an expected outcome, 4 did not use a template at all, 2 did not incorporate a motivation indicator.

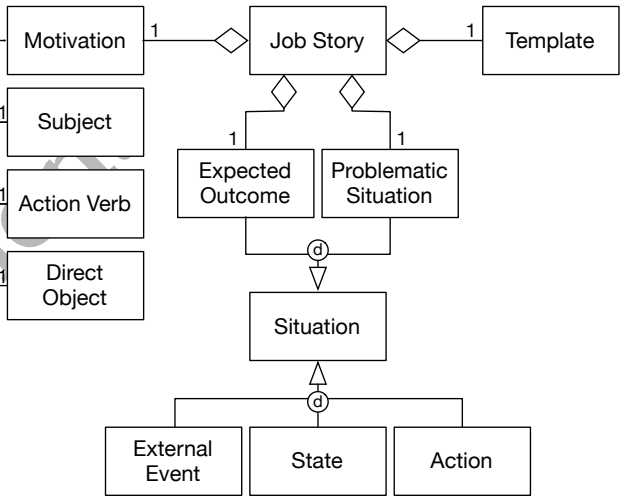

Fig. 1. Conceptual model of Job Stories

3.2 Problematic Situation

The first part of a Job Story provides the contextual starting point for the reader to understand why the actor has a motivation and a goal. It should describe a situation which confronts the actor with a concrete problem [21]. This may be a generic need such as wanting something to eat, or a specific problem in the context of a product such as JS2. In our Job Story analysis, we identified three variants of a well-formed situation that correspond to one example Job Story each:

- **Action.** The problem lies in the action that an actor is executing as part of her job. This case is exemplified by JS1.
- **State.** An actor or an object may be in a problematic situation (state) because of their attributes. This can take the form of an object property such as in JS2, but can also be an human emotional state such as being bored or unfamiliar with something.
- **External Event.** The problematic situation is experienced because of an external event, i.e., an action conducted by some other actor. Exemplified by JS3.

Analysis: Analyzing the 113 Job Stories that adhere to the Job Story template, two things stand out. First, most Job Stories (65) capture the situation as an action, 40 use a state and just 8 include an external event. Note that as an action or external event completes, the result is also a state. The difference between the variants is thus solely in the writing style. Second, only 36 Job Stories define a *problematic* situation. The other 77 Job Stories only describe a situation such as: "When I'm adding or deleting users" or "When I have multiple workspaces". While the original articles on Job Stories do not require the situation to be problematic [20,14], follow-up blogs stress that a helpful situation defines a concrete struggle and includes as much context as possible [18,21,22].

3.3 Motivation

The second part of a Job Story is its central element of motivation. Capturing the change that needs to occur in order to reach the expected outcome, the motivation generally already implies a solution to alleviate the problematic situation. Theoretically, motivations can have different structures, for they can be used to represent different types of requirements. The majority of motivations, however, adhere to the same basic grammatical structure as the means part of a user story [25]: (1) they contain a subject with an intent verb such as "want" or "am able", (2) followed by an action verb that expresses the action related to the problem to resolve, and (3) a direct object on which the subject of (1) executes the action. The motivations of JS1-3 adhere to the structure of a user story's means: <JS#,Subject+Intent,Verb,Direct Object> - <JS1, I want to, filter, tasks>, <JS2, I want to, restart, estimation process>, <JS3, I want, see, me>. Aside from these three parts, motivations are free form text and may contain any other linguistic construct such as adjectives and indirect objects.

Analysis: All 113 valid Job Stories adhere to the grammatical structure above to a greater or lesser extent: 83 Job Stories follow the structure exactly, including the *I* as the subject, 20 adhere to the user story structure but incorporate another actor as the subject as in "customers want to", and the remaining 10 do not include an action verb, but readers infer *to have* from the text structure like in "I want a filled in example".

3.4 Expected Outcome

While the first part of a Job Story presents the problematic as-is situation, the *expected outcome* sketches the desired to-be situation that the actor wants to achieve by satisfying the motivation. In this way, the expected outcome conveys additional context that is necessary to satisfy the motivation in the *right* way.

As the expected outcome also describes a situation, we encountered the same three possible variants in our Job Story analysis. Practitioners may define the expected outcome as: (1) an action the actor can now conduct as in JS1, (2) a desired state of an actor or non-sentient object such as *"so that proprietary intellectual property is secure"*, or (3) an external event for a different actor to conduct an action like JS2. The only difference is that the variants are often expressed in the negative form as in "so that there is no ambiguity" or "so that I don't have to refresh".

Analysis: The dominance of the action variety is less overwhelming with 51 Job Stories capturing the expected outcome with a new or improved action, while 42 define a state an 20 Job Stories refer to events.

4 Integrated Job Story Method

Multiple variations of Jobs-to-be-Done oriented methods exist. For example, the creators of Outcome-Driven Innovation have embraced Jobs-to-be-Done, but they propose an alternate approach to capturing requirements that requires first defining customers' Desired Outcomes [33]. Despite being advertised as a requirements approach based on Jobs-to-be-Done, it is different from Job Stories. It is therefore hard for practitioners to understand which Jobs-to-be-Done approach and format suits their situation best, resulting in arbitrary combinations of different methods [2,16].

To help practitioners conduct RE via Jobs-to-be-Done, we combine the available Jobs-to-be-Done literature into a Process-Deliverable-Diagram (PDD) that captures both the most important Jobs-to-be-Done and Job Story activities, as well as the utilized artifacts [35]. The resulting Integrated Job Story Method (Fig. 2) has five phases.

- **P1** - **Interview phase**: exploratory interviews are conducted with (prospective) customers in order to uncover their goals, their current way of achieving these goals, and the problems that exist in their as-is situation;
- **P2** - **Analysis phase**: the workflow, context and motivations of the interviewees are analyzed to formulate initial Jobs and Job Stories;
- **P3** - **Survey phase**: in order to validate the results of the analysis phase, a survey is conducted with a larger sample of the target audience;
- **P4** - **Prioritization phase**: the survey results are analyzed to determine which Jobs present the largest opportunity for innovation, and;
- **P5** - **Project definition phase**: the Jobs and Job Stories for development are selected and a *project brief* is created that can facilitate the follow-up development project.

We illustrate our method using a running example that explains how we applied each activity in a real-world case: Stabiplan, a product software company in the market of computer-aided design (CAD) software. Stabiplan's products extend the AutoCAD and Revit products by AutoDesk for mechanical, electronics and plumbing (MEP) installations. Founded in 1990, Stabiplan now has more than 170 employees and with more than 3,800 clients, it is the market leader in the Netherlands and Belgium [34].

For the case study, we focus on a new addition to Stabiplan's product portfolio, independent products based on Revit called *apps* that have limited functionality and aim at new customer segments. Stabiplan's goal is to use the method to determine what functionality, either existing in the core product or new, should be included in a highly specific Revit app for appliances and connectors to incite customers to adopt it. From here on, we refer to Stabiplan as *ModelComp* and Revit as *ModelPlatform*.

The Integrated Job Story Method incorporates two variants to accommodate two mainstream approaches to Jobs-to-be-Done: a *quantitative* approach based on ODI and a *qualitative* approach based on Klement's work. Our illustration adheres to the qualitative approach, but we also elaborate on the quantitative approach where possible. All these activities resulted in many materials of which we only show snippets, but all Jobs, Desired Outcomes and Job Stories are available on request.

P1. Interview phase

Perform interviews involves arranging, performing, recording and transcribing interviews with suitable participants. This results in `INTERVIEW TRANSCRIPTIONS` for each interviewee containing information on the `INTERVIEWEE BACKGROUND` and the `WORKFLOW` that describes how they currently try to reach their `GOALS`, in spite of the `PROBLEMS` that make this a struggle.

Case: We conducted and transcribed four extensive (1.5 hour each) interviews with users of ModelComp working in appliances and connectors, the product's target market.

P2. Analysis phase

Define high-level functional Jobs by analyzing the `GOALS` and `WORKFLOWS` that were collected during the interviews. This results in a number of high-level `FUNCTIONAL JOBS` that the users are trying to get done.

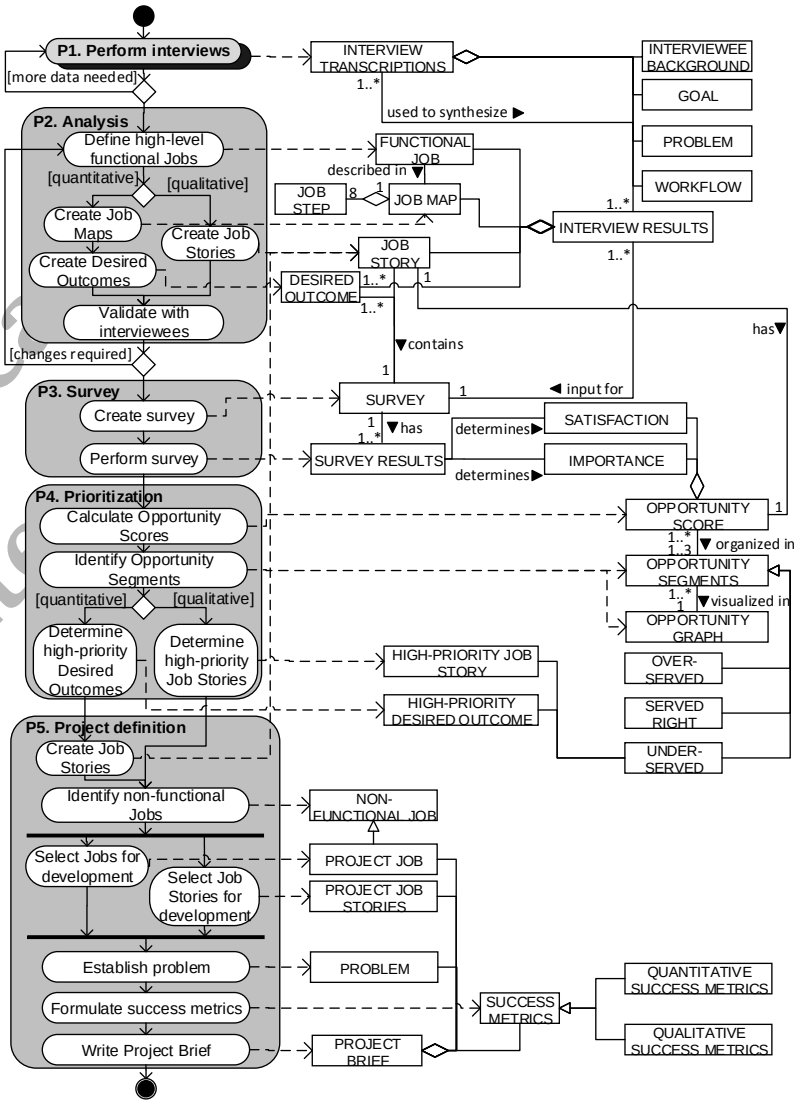

Fig. 2. Integrated method for Jobs-to-be-Done in Software Development

Case: Analyzing what the interviewees are trying to achieve when working with appliances in ModelPlatform, we formulate 4 high-level functional Jobs for using ModelPlatform for appliances:

- **J1** - Help me configure appliances.
- **J2** - Help me place appliances.
- **J3** - Help me model connector systems.
- **J4** - Help me create bills of materials.

Quantitative approach:
Create Job Maps by breaking down the identified JOBS into a series of discrete JOB STEPS that you capture in a JOB MAP. The goal of Job Mapping is to determine what the Job looks like for the customer, from beginning to end, and if different customers have different approaches to getting the Job done [1].
Case: We applied the Job Mapping technique to analyze the customers' workflow and identify all the lower-level functional Jobs the customers are trying to get done. Later in the project we created a fully fledged Job Map which is available in the online materials.

Create Desired Outcomes For each Job Step in the Job Maps, the interview transcriptions are reviewed to identify the DESIRED OUTCOMES: metrics that document what success means to a customer for each JOB STEP. These statements describe a direction of improvement, a unit of measure, and a object of control with contextual clarification and examples [1]. An example in the context of angioplasty balloons is: 'minimize the amount of force that is required to cross the lesion with the balloon' [32].
Case: Analyzing the interviewees' struggle when trying to get their Jobs done, we defined 50 Desired Outcomes for the four Jobs above. For J1, two examples are 'Minimize the chance that I need to check external documentation' and 'Minimize the chance that a fabrication specific thermostatic connector I need is not available'.

Qualitative approach:
Create Job Stories for each high-level functional Job that, taken together, satisfy their goals. The resulting JOB STORIES should provide context about the user's struggle to get the functional Job done: the situation, motivation and expected outcome [21].
Case: Using the contextual information identified in the interviews, we created 21 Job Stories that relate to J1-4 such as '**When** I am configuring an appliance, **I want** the output power of the appliance to be accurately represented in the flow and return meters, **so that** my model will correctly represent the produced power.'

Validate with interviewees whether the resulting Desired Outcomes or Job Stories of the analysis phase correspond to their problematic situation.
Case: We took an iterative approach to this activity by validating draft Job Stories based on the interviewee's input. The outcome was considered satisfactory and no substantial changes to the Job Stories were necessary.

P3. Survey phase
Create survey to validate the output of the analysis phase. The SURVEY asks the participants to rate each Desired Outcome or Job Story on two dimensions with a Likert scale: IMPORTANCE and SATISFACTION with current solutions [32].
Case: We created an online survey with 7-point Likert scale questions for each of the Job Stories which is available upon request here https://goo.gl/rvbfZw.
Perform survey includes finding and reaching potential survey respondents as well as extracting the collected SURVEY RESULTS for analysis.
Case: We distributed the survey and obtained 10 responses from the target audience.

P4. Prioritization phase
Calculate Opportunity Scores for each of the Job Stories or Desired Outcomes by applying the following formula to the survey results:

OPPORTUNITY SCORE = IMPORTANCE + (IMPORTANCE - SATISFACTION) [32].
Case: We calculated the mean opportunity score for all respondents for each Job Story.

Identify Opportunity Segments involves analyzing the opportunity scores to identify the Job Stories with a high rating for importance and a medium to low rating for satisfaction [1]. To do this, plot the Job Stories or Desired Outcomes on an OPPORTUNITY GRAPH, with the satisfaction on the y-axis and importance on the x-axis. Next, divide the OPPORTUNITY GRAPH in three OPPORTUNITY SEGMENTS to classify needs into UNDER-SERVED, SERVED-RIGHT, or OVER-SERVED [31].
Case: We plotted the Job Stories' Opportunity Scores on the Opportunity Graph in Fig. 3.

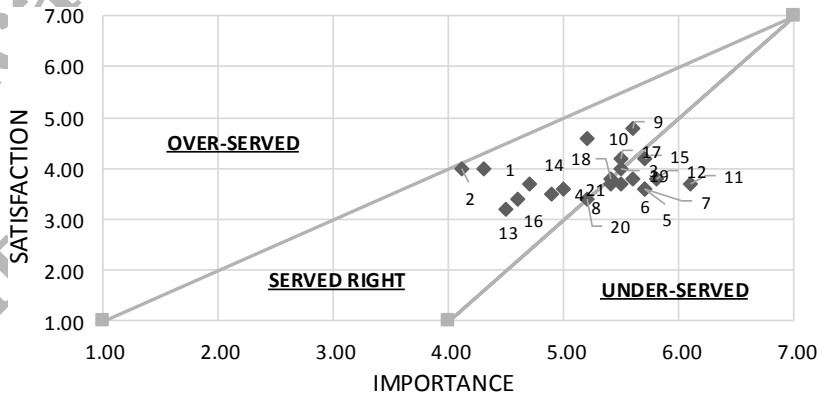

Fig. 3. Opportunity Graph for ModelComp's Job Stories

Select high-priority Job Stories or Desired Outcomes in the opportunity segment to target and classify these as HIGH-PRIORITY JOB STORIES or DESIRED OUTCOMES.
Case: None of the Job Stories in Fig. 3 are within the over-served segment, 10 Job Stories are served-right, 8 Job Stories are under-served and 3 are borderline cases. As ModelComp wants to improve upon an existing product, we select the 8 under-served Job Stories as well as the 3 borderline cases.

P5. Project definition phase
Quantitative approach: Create Job Stories The requirements engineer first formulates Job Stories based on the selected high-priority Desired Outcomes.

Identify non-functional Jobs that capture the fundamental problems of the customer his functional Jobs. As fundamental problems are rarely about a straightforward, functional task [6], these NON-FUNCTIONAL JOBS are necessary to provide insight into the underlying forces that drive customer behavior [5].
Case: As there is no literature on how to categorize Job Stories based on the common non-functional Jobs, ModelComp defined their own four step approach:

1. We define three high-level non-functional Jobs that drive customers to hire ModelComp to help their organization excel at Building Information Modeling
2. We decompose each high-level non-functional Job into two or more medium-level sub-non-functional Jobs

3. We categorize all Job Stories according to the medium-level non-functional Job they belong to
4. For each medium-level non-functional Job, we sub-categorize its associated Job Stories in low-level non-functional Jobs

Select Jobs for development from the non-functional Jobs to address with a concrete development project and redefine them as `PROJECT JOBS`.
Case: To select the most important Jobs for further development, we want to select the low-level Jobs whose Job Stories have the highest mean opportunity score. As the focus of this project is on improving *modeling* for appliances, we select low-level non-functional Jobs 1 and 4: *"Help me ensure that I deliver high quality work"* and *"Help me save time"*.

Select Job Stories for development - Depending on the project scope, the requirements engineer selects all- or a sub-set of the high-priority Job Stories associated with the Project Job. The resulting set of `PROJECT JOB STORIES` define the scope of the software solution.
Case: Together, Jobs 1 and 4 comprise 10 Job Stories. To identify the most important Job Stories for inclusion in development, we select only those that are under-served, i.e. with an opportunity score of 7.0 or higher. Applying this exclusion criteria results in a total of 8 Job Stories as shown in Fig. 4.

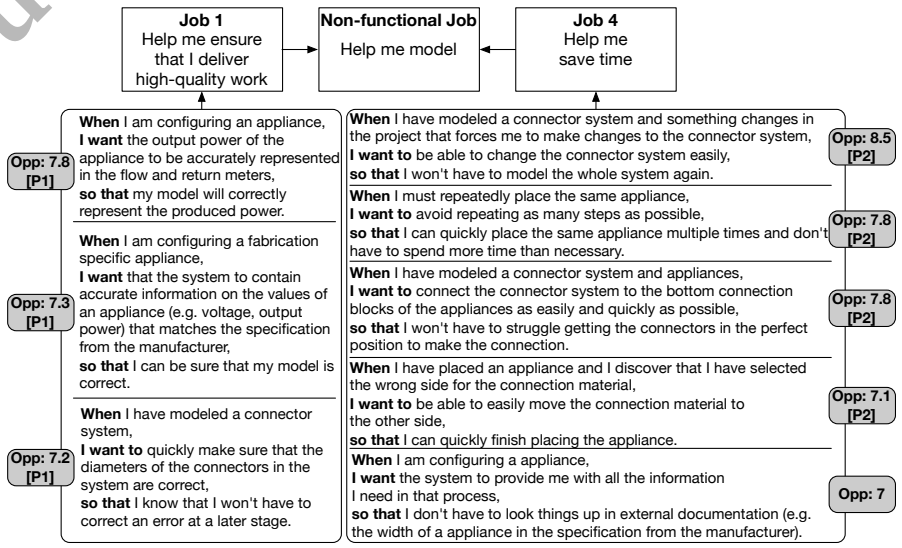

Fig. 4. Snippet of ModelComp Job Portfolio for appliances

Establish problem - From the selected non-functional job and its associated job stories, establish one or more re-occurring and concrete `PROBLEM`(s) that development can address in a concrete project.
Case: Combining the selected project job stories and our growing insights from the interviews, we identify a concrete and distinct problem for each Project Job as shown

in Fig. 4. For Job 1, we identify that modelers struggle to ensure that the appliances used in a ModelPlatform model are linked to the correct parametric information (P1). For Job 2, we identify that modelers struggle to change an existing connector system to incorporate changing requirements (P2).

Formulate success metrics by defining the `QUALITATIVE`- and/or `QUANTITATIVE SUCCESS METRICS` that determine whether the problem has been solved [14].
Case: We defined high-level success metrics that do not presuppose a solution, such as *'the solution helps the modeler feel confident that the appliances are configured correctly'* for Job 1 and *'the time it takes to change the types of appliances that are used in a piping system is reduced by X%'* for Job 2. Note that these may be made more specific later on.

Write Project Brief that summarizes the results from all preceding phases. The resulting single-page `PROJECT BRIEF` includes a succinct description of the problem to be solved by the project, the relevant project jobs and project stories, as well as the success metrics to determine whether the problem has been solved [14].
Case: Using the Jobs, Job Stories, problem and success metrics defined earlier, we wrote one Project Brief for each of the two Jobs. To help tie the different elements together and provide more context, we also defined a high-level Job Story for each Job.
Job 1: '**When** I am working on a complicated model for an important project and I cannot afford to make mistakes, **I want** to be able to identify and fix possible errors, **so that** I can be confident that the work I deliver is of high quality.'
Job 2:'**When** I have been working on a model for a while but suddenly the requirements change, **I want** to be able to quickly modify my existing model to address the new requirements, **so that** we can prevent major delays to the project.'

5 Evaluating the Integrated Job Story Method

Upon completing the project brief and delivering it to ModelComp stakeholders, we evaluated the utility of the method and its artifacts. We conducted an in-depth evaluation with the lead product manager, discussed the project brief with three marketeers and two developers. Below, we discuss the major findings and remarks by each stakeholder.

The Lead Product Manager considers the process of interviewing customers and distilling requirements to be a useful and important practice. Although the resulting Project Brief and Job Stories did not present revolutionary new insights to the the product manager himself, he found them effective for scoping a project and conveying their customers' priorities to internal stakeholders. Even more interesting, however, are the supporting artifacts Job Portfolio and Opportunity Graph, which can help identify new high-level opportunities for apps that emphasize specific Jobs: *"Exploring what Job Stories are under-served, served right and over-served is very valuable to target apps to a specific country or market"*. There are, however, two major drawbacks to the Integrated Job Story Method . First, too many activities require active participation of (prospective) customers, who may be difficult to persuade and/or reach in a timely fashion. Second, the method itself is very time-consuming and the product manager considered the project to take too long by modern (agile) development standards.

Marketeers and Jobs-to-be-Done are a natural fit, as marketing and sales should be more convincing when it directly engages the Jobs you are trying to complete [5,21]. Indeed, the marketeers found the project brief to be the most valuable artifact as it concretely defines a specific project that they can easily incorporate in their own work. In particular, the marketeers consider the contextual information in the project brief to be more useful input for creating marketing material than a feature list. A feature list only shows the 'how' of a product, not the benefits for the customer (the 'why'): *"For our marketing strategy we assume that everyone uses a certain feature the same way. Jobs-to-be-Done might help in finding out that this is not the case and that we should change our marketing in Germany or France."*

Developers consider the project brief and its associated Job Stories to be defined on too high a level to be useful for actual software development. However, the developers did believe that the information in the project brief is useful to familiarize new hires with the non-technical context of ModelComp's products, as the following quote illustrates: *"By showing how our work affects the customer, the Project Briefs could help new hires understand the context of our products and enable them to become more pro-active in the development process."*

6 Discussion

Since Jobs-to-be-Done and Job Stories are young approaches, our work is still exploratory. Nevertheless, as part of our conceptualization, method construction and illustrative application in a real-world case, we could identify some interesting findings:

Balanced Problematization of Job Stories - Although 113 Job Stories syntactically adhere to the Job Story template, only 31.9% (36) define a *problematic* situation. This may seem mostly harmless, but in fact often has a substantial negative impact: the problem definition shifts to another part of the Job Story, leaving no room for the expected outcome. Consider for example *"When I am searching for tools, I want Creatlr to filter the options to my demand, so I have less choice."*, here the problem, having too many tools to choose from, is implicitly captured in the expected outcome and the actual expected outcome is not captured: finding the right option quicker. A better way to write this Job Story is *"When a search for tools gives me too many options, I want Creatlr to filter the options to my demand, so I can quickly find the right tool"*. The practice of defining a Job Story poorly signifies the need for a more clear-cut definition of a Job Story and concrete guidelines for how to write a high-quality Job Story.

As a quality criterion, we propose that Job Stories should have a *balanced problematization*: the situation should be problematic, while the expected outcomes should *not* be problematic. Job Stories can thus be in three states of *unbalanced problematization*:

1. Job Stories whose situation is not problematic should either be *problematized*, or *discarded* when no problem can be identified.
2. Job Stories whose expected outcome is problematic should be *de-problematized*.
3. Job Stories whose situation is not problematic and whose expected outcome is problematic should be *re-problematized*.

The Jobs-to-be-Done Confusion - Unfortunately, Jobs-to-be-Done is not a unique and clear framework with delineated activities and deliverables. Instead, it is a 'set of

principles' from which a wide variety of best practices, approaches and techniques have emerged [6]. As there are multiple, distinct approaches in literature, the Integrated Job Story Method integrates multiple sources that we selected based on our expertise. Additional validation and experimentation is necessary to establish the optimal approach.

Magically Emerging Job Stories - Despite the variety in Jobs-to-be-Done sources and approaches, there are no concrete guidelines for how to define Jobs, Job Stories or Desired Outcomes from the goals, problems and workflows identified through the exploratory interviews. Nor are there any methods available to validate that the Job Stories you formulated are appropriate and applicable.

7 Related Literature

At its core, the Jobs-to-be-Done theory is shaped by certain key principles from management and innovation science. A noteworthy input is Schumpter's theory on 'Creative Destruction': as firms grow and get older, they seem to lose some of their capacity to innovate, and often end up losing their market share to radical innovations from new firms [30]. To avoid creative destruction, Levitt argues that companies should not define themselves by what they produced, but by what customer needs they are addressing [23]. Jobs-to-be-Done builds on these insights through its framing of the customer needs as the Jobs that drive them to 'hire' products or services [4].

In the software industry, many different techniques are being used to perform RE, ranging from classical approaches such as use cases and scenarios to more recent techniques such as user stories [24]. Although not an entirely unique proposition, Jobs-to-be-Done is differentiated in its genuine problem orientation that is achieved through an understanding of the problematic context that drives customer behavior.

As such, Jobs-to-be-Done is similar to User-Centered Design (UCD), which involves focusing on usability throughout the entire development process and further throughout the system life cycle [28]. To do this, UCD advises designers to directly observe how users work [9], and use the insight to ensure that technology is organized around the users goals, tasks, and abilities [11]. However, While the role of Jobs-to-be-Done is limited to performing RE in the problem space, UCD is focused on the whole process of software development and has an explicit focus on the usability quality aspect [28].

A similar approach from RE literature is Goal-Oriented Requirements Engineering, which revolves around the use of goals at different levels of abstraction, to capture the various objectives the system should achieve [36]. As such, like Jobs-to-be-Done, Goal-Oriented RE strives to uncover the 'why' behind software [37]. In fact, we consider Jobs-to-be-Done as a pragmatic, simplified, industry-oriented adaptation of the principles behind goal orientation.

Job Stories were conceived as a response to perceived problems with User Stories and personas [20]. Job Stories are designed to convey situational and motivational contextual information, and should be entirely focused on the problem, not the solution [20]. Job Story critics argue that User Stories can be written to convey similar information [12], and that the template of the story is not as important as the underlying processes and interactions [27]. Experiences of Intercom illustrate that the implemen-

tation of Job Stories involves more than changing the template of the stories that are used. It also requires that processes are adopted to a Jobs-to-be-Done mindset [14].

Jobs-to-be-Done practitioners echo the concerns voiced by Christensen who argued that personas can not be used to explain why customers buy products[5], and Chapman and Millman, who argue that the validity of personas is impossible to verify [3]. Jobs-to-be-Done posits that by focusing on the Jobs that different 'types' of people have in common, these problems can be prevented [20].

8 Conclusion and Future Research

We have explored the notion of Job Stories: a new paradigm for agile RE based on Jobs-to-be-Done that focuses on capturing the motivational and situational context that drive customer behavior. Our constructed conceptual model and Integrated Job Story Method are a first theory building attempt, aimed at supporting practitioners apply this paradigm in a more systematic manner than the current ad-hoc practices.

Our work is exploratory and paves the way for future directions. Considering the varying quality of Job Stories created by practitioners, there is a need for concrete quality criteria for Job Stories. Similarly, we intend to develop problematization techniques for expressing problems in Job Stories and for coping with Job Stories that need to be (re-)problematized. Furthermore, because of Job Stories' similarity to user stories we would like to investigate how to extract concepts from Job Stories with NLP, similar to our earlier work on user stories [26]. Finally, we want to further validate, evaluate and improve the Integrated Job Story Method in order to establish a rigorous and reliable approach to working with Jobs-to-be-Done and Job Stories in software development.

References

1. L. A. Bettencourt and A. W. Ulwick. The customer-centered innovation map. *Harvard Business Review*, 86(5), 2008.
2. H. Carpenter. A method for applying jobs-to-be-done to product and service design. goo.gl/5NUVwh, January 2013.
3. C. N. Chapman and R. P. Milham. The personas' new clothes: Methodological and practical arguments against a popular method. In *Proceedings of the human factors and ergonomics society annual meeting*, volume 50, pages 634–636, 2006.
4. C. Christensen, S. Cook, and T. Hall. Marketing malpractice: The cause and the cure. *Harvard Business Review*, 83(12), 2005.
5. C. M. Christensen, S. D. Anthony, G. Berstell, and D. Nitterhouse. Finding the right job for your product. *MIT Sloan Management Review*, 48(3), 2007.
6. C. M. Christensen, K. Dillon, T. Hall, and D. S. Duncan. *Competing Against Luck: The Story of Innovation and Customer Choice*. Harper Business, 2016.
7. C. M. Christensen, T. Hall, K. Dillon, and D. S. Duncan. Know your customers' "Jobs to Be Done". *Harvard Business Review*, 2016.
8. M. Cohn. *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.
9. A. Dix. *Human-computer interaction*. Springer, 2009.
10. K. M. Eisenhardt and M. E. Graebner. Theory building from cases: Opportunities and challenges. *Academy of Management Journal*, 50(1):25–32, 2007.

11. M. R. Endsley. *Designing for situation awareness: An approach to user-centered design*. CRC press, 2016.
12. Gianpi. Job stories are great, but they only replace badly written user stories. goo.gl/7pxy6o, February 2015.
13. T. Higgins. 5 classic mistakes made while writing user stories. goo.gl/LdzX3H, 2016.
14. Intercom. *Intercom on Jobs-to-be-Done*. 2016.
15. R. Jeffries. Essential XP: Card, conversation, and confirmation. *XP Magazine*, August 2001.
16. L. Johnson. Jobs to be done: A case study in the NHS. goo.gl/gpaBpx, September 2017.
17. M. Kassab. The Changing Landscape of Requirements Engineering Practices over the Past Decade. In *Proc. of EmpiRE*, pages 1–8, 2015.
18. A. Klement. 5 tips for writing a job story. goo.gl/dMWTQd, November 2013.
19. A. Klement. Designing features using job stories. goo.gl/NS889V, 2013.
20. A. Klement. Replacing the user story with the job story. goo.gl/fZ1iqe, 2013.
21. A. Klement. *When Coffee and Kale Compete*. NYC Publishing, 2016.
22. A. Klement. Your job story needs a struggling moment. goo.gl/vsRC1d, 2016.
23. T. Levitt. Marketing myopia. *Harvard Business Review*, 38(4):24–47, 1960.
24. G. Lucassen, F. Dalpiaz, J. M. E. van der Werf, and S. Brinkkemper. The use and effectiveness of user stories in practice. In *Proc. of REFSQ*, pages 205–222. Springer, 2016.
25. G. Lucassen, F. Dalpiaz, J. M. E. M. van der Werf, and S. Brinkkemper. Improving agile requirements: the quality user story framework and tool. *Requirements Engineering*, 21(3):383–403, Sep 2016.
26. G. Lucassen, M. Robeer, F. Dalpiaz, J. M. E. M. van der Werf, and S. Brinkkemper. Extracting conceptual models from user stories with visual narrator. *Requirements Engineering*, 22(3):339–358, Sep 2017.
27. K. Nicieja. The most common user story mistake. goo.gl/tf4aCH, 2015.
28. D. A. Norman and S. W. Draper. User-centered design. *New Perspectives on Human-Computer Interaction*, 1986.
29. C. Potts and G. Bruns. Recording the reasons for design decisions. In *Proc. of ICSE*, pages 418–427. IEEE Computer Society, 1988.
30. J. A. Schumpeter. *Socialism, capitalism and democracy*. Harper and Brothers, 1942.
31. A. Ulwick and P. Hamilton. The jobs-to-be-done growth strategy matrix. Technical report, Strategyn, 2016.
32. A. W. Ulwick. Turn customer input into innovation. *Harvard business review*, 80(1):91–7, 2002.
33. A. W. Ulwick and L. A. Bettencourt. Giving customers a fair hearing. *MIT Sloan Management Review*, 49(3):62–68, 2008.
34. R. van Cann, S. Jansen, and S. Brinkkemper. *Software Business Start-up Memories*. Springer, 2013.
35. I. van de Weerd and S. Brinkkemper. Meta-modeling for situational analysis and design methods. In *Handbook of research on modern systems analysis and design technologies and applications*, pages 35–54. IGI Global, 2009.
36. A. Van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Proc. of RE*, pages 249–262. IEEE, 2001.
37. E. S. Yu and J. Mylopoulos. Understanding "why" in software process modelling, analysis, and design. In *Proc. of ICSE*, pages 159–168. IEEE, 1994.
38. P. Zave and M. Jackson. Four dark corners of requirements engineering. *ACM Trans. Softw. Eng. Methodol.*, 6(1):1–30, Jan. 1997.