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]

[1] RE-Lab, Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands
{g.lucassen,m.a.r.vandekeuken,f.dalpiaz,s.brinkkemper}@uu.nl
[2] Stabiplan, Bodegraven, The Netherlands
{g.w.sloof,j.schlingmann}@stabiplan.com

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. As 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 multiple approaches based on job stories 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 to 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
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 that only capture the 'what' and 'how' [1–5]. With the rise of agile software development, user stories have become increasingly popular with adoption up to 55% [6]. Although intended

E. Kamsties et al. (Eds.): REFSQ 2018, LNCS 10753, pp. 227–243, 2018.
https://doi.org/10.1007/978-3-319-77243-1_14

to focus on the problem space and not the solution space [7,8], user stories are often formulated with a specific solution in mind [9], also because some authors suggest their use for describing *features* [10].

Recognizing this problem for the development of innovative software products, Alan Klement introduced a new paradigm for requirements formulation called Job Stories [11], which relies on the following template:

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

A job story is written from the perspective of a customer who, in a given *situation*, expresses a *motivation* for the requirement to exist, aiming to attain an *expected outcome*. For example [11]: *When an important new customer signs up* (situation), *I want to be notified* (motivation), *so I can start a conversation with them* (expected outcome).

Job stories are based on the ideas of the Disruptive Innovation Theory's method *Jobs-to-be-Done* (JTBD) by Christensen [12], a collection of principles that help discover and understand the interactions between customers, their motivations and the products they use [13]. Despite its recency, there are many authors who propose different approaches to JTBD [14–17]. 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, JTBD is difficult to apply in the context of software, for many of the examples are for physical products. To improve on this situation, this paper makes the following contributions:

1. We collect Jobs-to-be-Done literature and position job stories among its different approaches and best practices (Sect. 2);
2. We conceptualize the notion of job story and analyze the syntax and semantics of 131 job stories gathered from public sources in Sect. 3;
3. We introduce the Integrated Job Story Method that reconciles the different views on JTBD and job story literature in Sect. 4;
4. We evaluate the applicability of the Integrated Job Story Method on a case study about app development in the computer-aided design field (Sects. 5 and 6).

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

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

Job stories originate from Jobs-to-be-Done (JTBD), and the major principles of Jobs-to-be-Done build upon a large body of literature from management science [12,13,18]. The most similar predecessor to JTBD 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 [15].

Ten years later, Christensen introduced the 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"* [19]. 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 [1,2]. To uncover the *real* driving force of customer behavior, innovators should investigate the jobs that customers are struggling to get done [12].

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 milkshake for kids in the afternoon [14].

The milkshake example shows the suitability of Jobs-to-be-Done for physical products with straightforward functionality [12,14,15,19,20]. Through a series of blog posts and a self-published free book, Alan Klement and Intercom Inc. propose a new paradigm for JTBD-based requirements called *Job Stories* [11,21]. 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** I am 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."* [11]

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"* [16]. However, a study of the relationship between user stories and job stories is beyond the purpose of this paper.

3 Conceptualizing Job Stories

The original template for job stories [11] has been customized by practitioners in different ways (see the collection of job stories in our dataset for an overview[1]), 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

[1] The dataset of this paper is available at http://dx.doi.org/10.17632/xpcv3jb6b4.1.

situation and (iii) the resulting *expected outcome* that improves the situation [13]. Consider the following illustrative examples of job stories (labeled JS1–JS3) that we use throughout this section:

JS1. When I am looking for a new task on the task board, **I want to** filter tasks to only see those 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 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 [22] resulting in a conceptual model of job stories. To do so, we conducted a thorough analysis of 131 job stories (available in the dataset) that we have identified through search engines in publicly accessible sources that include blogs, presentations, project code, theses, and books. Based on such analysis, and relying on frequency of occurrence and the authors' experience in conceptual modeling, we built a conceptual model of job stories (see Fig. 1).

We elaborate on the story components and discuss the adherence of the job stories with our conceptual model. The fully-fledged analysis is available in the dataset.

3.1 Template

A job story should roughly follow the original job story template (when... I want to... so that...) proposed by Klement [11]. 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 (resembling user stories), 6 did not include an expected outcome, 4 did not use a template, 2 did not include a motivation indicator.

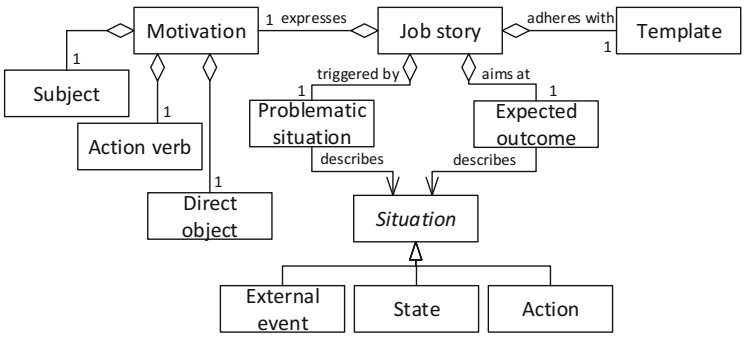

Fig. 1. Conceptual model of job stories

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 [13]. 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 fit 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 a new state. 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 [11,21], follow-up blog posts by Klement himself explain the importance of defining a concrete struggle and its context [13,23].

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 [9]: (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 and 20 job stories refer to events.

4 Integrated Job Story Method

Multiple flavors of JTBD orientation exist. For example, the creators of ODI have embraced JTBD, but their method to capturing requirements starts from defining customers' desired outcomes [24]. Despite being advertised as a requirements engineering method based on JTBD, ODI differs from job stories. It is therefore hard for practitioners to understand which JTBD approach and format suits their situation best, resulting in various combinations of different methods [17,25].

We take a systematic approach to reconcile the existing methods based on JTBD, and thereby assist practitioners in operationalizing such paradigm. We do so by applying situational method engineering to construct a Process-Deliverable-Diagram (PDD) (see Fig. 2); the resulting PDD describes our integrated method in terms of JTBD and job story activities, as well as the utilized artifacts [26].

Our method, which is named *Integrated Job Story Method*, was assembled pragmatically: we studied the literature on JTBD, identified fragments that could be consistently combined, and determined feasibility based on an application to a real-world case, described below. Note that we propose *a* method for

conducting RE based on JTBD, but other methods exist. Our method comprises five phases:

P1. Perform interviews: 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.

Case study. Stabiplan is a medium-sized (170 employees, 3,800+ clients) 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. We focus on a new addition to Stabiplan's product portfolio, i.e., independent products based on Revit called *apps* that have limited functionality and aim at new customer segments. Stabiplan's aims to use the method to determine what functionality, either existing 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 JTBD: a *quantitative* approach based on ODI—based on metrics—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 jobs, desired outcomes and job stories are in the dataset; due to space limitations, we show only snippets in the paper.

4.1 Perform Interviews (P1)

This phase, which is not decomposed into activities due to its generic nature, involves arranging, performing, recording and transcribing interviews with suitable participants. This results in `INTERVIEW TRANSCRIPTIONS` for each interviewee including their `INTERVIEWEE BACKGROUND`, and the `WORKFLOW` they currently follow trying to reach their `GOALS`, in spite of the `PROBLEMS` they struggle with. The practice of interviewing customers as a part of JTBD is described both by Christensen [14] and Ulwick [15];

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

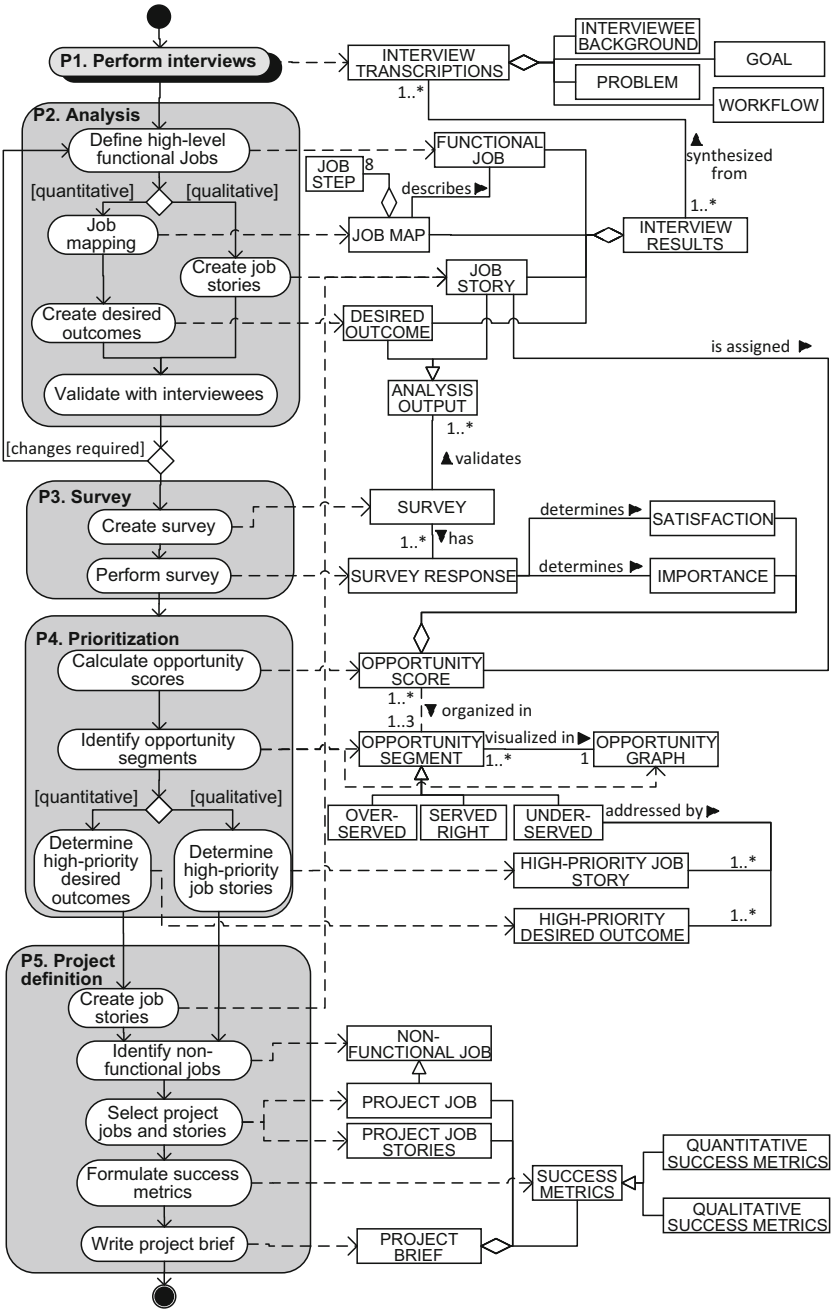

Fig. 2. Process-Deliverable Diagram [26] representing the activities (left-hand side) and the artifacts (right-hand side) of our integrated method for JTBD in software development.

4.2 Analysis Phase (P2)

Define high-level functional jobs. The interview transcriptions are analyzed, especially `GOALS` and `WORKFLOWS`; this results in a set of high-level `FUNCTIONAL JOBS` that the users are trying to get done. We use the term functional, drawing an analogy with the standard nomenclature in requirements engineering, to indicate that these jobs define *what* the user aims to achieve.

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[2]:

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

Next, the method splits into its two main paths that reflect the ongoing discussion among JTBD practitioners: the quantitative approach that is inspired by ODI, and the qualitative approach that reflects way of working of job story practitioners.

Quantitative Approach

Job mapping. The identified `JOBS` are broken down into a series of `JOB STEPS` that are represented in a `JOB MAP`. Job mapping establishes what the entire job looks like for the customer from beginning to end, providing a perspective that is not limited to the scope of the product of interest [20]. This includes identifying new/innovative ways for the product to help the different customers get more of the job done.

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 dataset.

Create desired outcomes. Each job step in the job maps, alongside the interview transcriptions, is turned into `DESIRED OUTCOMES`: metrics that explain what success means to a customer for each `JOB STEP`. These statements describe a direction of improvement, a unit of measure, and provide contextual clarification and examples [20].

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'.

[2] We adopt Klement's syntax for the jobs: the customer hires a product to get jobs done.

Qualitative Approach

Create job stories. Each high-level functional job is refined into job stories; together, the stories should satisfy the interviewees' 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 [13].
Case: Using the contextual information identified in the interviews, we created 21 job stories that relate to the jobs 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. The resulting desired outcomes or job stories of the analysis phase are checked to determine if they correspond to the stated 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.

4.3 Survey Phase (P3)

Create survey. This activity validates the output of the analysis phase with a larger group. 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]. This survey technique and the prioritization described in the next phase are based on Anthony Ulwick's ODI methodology [15];
Case: We created an online survey with 7-point Likert scale questions for each of the job stories which is available in our online materials.

Perform survey. This activity involves finding and reaching potential survey respondents as well as extracting the collected `SURVEY RESPONSES` for analysis.
Case: We distributed the survey and obtained 10 responses from the target audience.

4.4 Prioritization Phase (P4)

Calculate opportunity scores. For each of the job stories or desired outcomes, the following formula, taken from Ulwick's work [15], is applied to the survey response:
`OPPORTUNITY SCORE` = `IMPORTANCE` + max(`IMPORTANCE` − `SATISFACTION`, 0).
Case: We calculated the mean opportunity score for all respondents for each job story.

Identify Opportunity Segments. This activity analyzes the opportunity scores to identify the job stories with a high rating for importance and a medium to low rating for satisfaction [20]. These are the job stories that are currently under-served and should present a significant opportunity for improvement. 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`. This approach is explained in [27].
Case: We plotted the job stories' Opportunity Scores on the Opportunity Graph in Fig. 3.

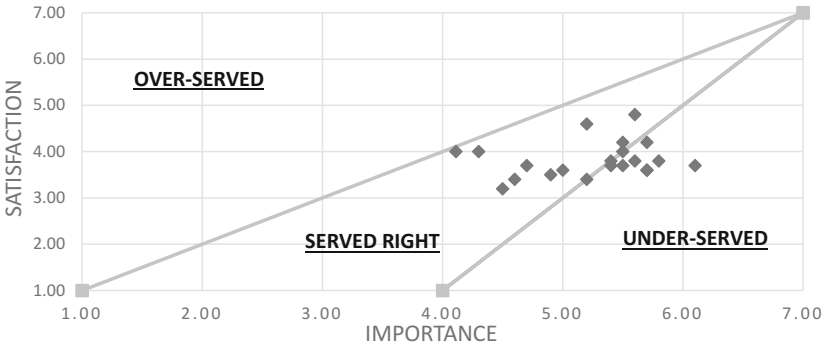

Fig. 3. Opportunity graph for ModelComp's job stories

Select high-priority job stories or desired outcomes. Analyze the opportunity segment to target and identify the `HIGH-PRIORITY JOB STORIES` or `DESIRED OUTCOMES`.
Case: Based on the ratings of P3, 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.

4.5 Project Definition Phase (P5)

This final phase of the method is based on the experiences reported by Alan Klement and the company Intercom, of working with job stories and creating project briefs [21]. We adapt those ideas for our integrated method.

Create job stories. In the quantitative approach, the requirements engineer formulates a set of job stories based on the selected high-priority desired outcomes.

Identify non-functional jobs that capture the fundamental problems the customer faces with her functional jobs. As fundamental problems are rarely about a straightforward, functional task [28], these `NON-FUNCTIONAL JOBS` are necessary to provide insight into the underlying forces that drive customer behavior [14]. Such non-functional jobs give insights on *how* the customer wants the function to be delivered.
Case: As no literature exists that explains how to categorize job stories based on the common non-functional jobs, a specific four step approach was defined for ModelComp:

1. Three high-level non-functional jobs are defined that drive customers to hire ModelComp to help their organization excel in business information modeling, the sub-field of CAD supported by ModelPlatform;
2. Each high-level non-functional job is decomposed into two or more medium-level sub-non-functional jobs;
3. All job stories are classified according to the medium-level non-functional job they belong to;
4. For each medium-level non-functional job, its associated job stories are classified in low-level non-functional jobs.

Select project jobs and stories. A subset of the non-functional jobs is assigned to the development project (`PROJECT JOBS`), alongside all or a sub-set of the high-priority job stories associated with the project jobs (`PROJECT JOB STORIES`). The selection may take into account synergies between jobs and job stories, e.g., stories that can be bundled in the same project because of their interdependencies at development time.
Case: We choose 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"*. Together, these jobs comprise 10 job stories, out of which we select only those that are under-served, i.e., with an opportunity score of 7.0 or higher. This results in 8 job stories as shown in Fig. 4.

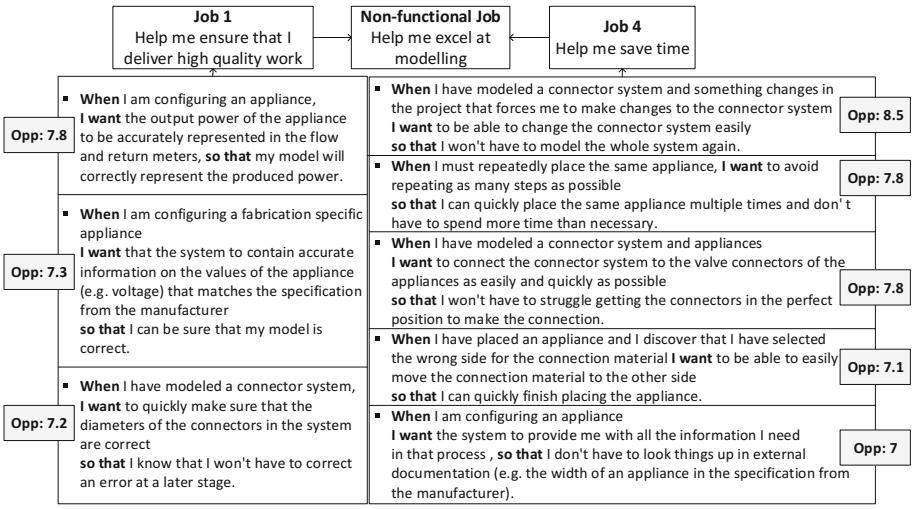

Fig. 4. Snippet of ModelComp job portfolio for appliances. The label "Opp." indicates the opportunity score for a given job story.

Formulate success metrics. To enable assessing whether the development is effective, `QUALITATIVE`- and/or `QUANTITATIVE SUCCESS METRICS` are establish that test whether the problem has been solved [21].
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.

Write project brief. This artifact 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 [21]. The stakeholders can check the project brief to validate the requirements before starting the development.

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.

Lead Product Manager. He considers the process of interviewing customers and distilling requirements a useful and important practice. Although the project brief and job stories did not present revolutionary new insights to 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"*. Two major drawbacks of the method were identified. First, too many activities require active participation of (prospective) customers, who may be difficult to reach in a timely fashion. Second, the method is very time-consuming and the product manager considered the project to take too long by modern (agile) development standards.

Marketeers. JTBD fits well with this role, for marketing and sales should be convincing when it directly engages the jobs one is trying to complete [13,14]. 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, they consider the contextual information in the project brief a 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. JTBD might help in finding out that this is not the case and that we should change our marketing in Germany or France."*

Developers. They consider the project brief and its associated job stories too high a level to be useful for actual software development. However, the developers considered the information in the project brief useful to familiarize new hires with the non-technical context of ModelComp'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 Discussion

Since JTBD 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. This is illustrated by the example about Creatlr above.
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 variety of Jobs-to-be-Done approaches. Unfortunately, JTBD 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 [28]. 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.

Missing guidelines on how to write 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. Similarly, there are no methods available to validate that the formulated job stories are appropriate and applicable.

7 Related Literature

The Jobs-to-be-Done theory is grounded in management and innovation science. A key input is Schumpter's theory on 'Creative Destruction': as firms grow and age, they lose some of their ability to innovate, and often end up losing their market share to radical innovations from new firms [29]. To avoid creative destruction, Levitt argues

that companies should not define themselves by what they produced, but by what customer needs they are addressing [18]. 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 [19].

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 [30]. 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.

JTBD is similar to User-Centered Design (UCD), which involves focusing on usability throughout the entire development process and further throughout the system life cycle [31]. To do this, UCD advises designers to directly observe how users work [32], and use the insight to ensure that technology is organized around the user's goals, tasks, and abilities [33]. However, while UCD is a broad method focused on the whole process of software development, and focuses on usability [31], Jobs-to-be-Done focuses on the problem space and allows to consider multiple software qualities, not only usability.

A similar approach from the RE literature is Goal-Oriented Requirements Engineering (GORE), which revolves around the use of goals at different levels of abstraction, to capture the various objectives the system should achieve [34]. As such, like JTBD, GORE strives to uncover the 'why' behind software [2]. We consider JTBD as an industry-oriented adaptation of the principles behind goal orientation, which favors pragmatism (using text and simple diagrams) over conceptual soundness (e.g., the distinction between actor types, goals and tasks, goal types, and the distinctions that allow for formal reasoning over goal graphs).

Task oriented RE [4] proposes to express functional requirements as tasks that should be jointly achieved by humans and systems, but without assigning specific responsibilities. Job stories share the focus on the task/job, but focus on analyzing the motivations behind the job, rather than drilling down into the task details.

Job stories were conceived as a response to perceived problems with user stories and personas [11]. Experiences of Intercom illustrate that the implementation of job stories involves more than changing the template of the stories that are used. It also requires that processes are adopted to a JTBD mindset [21].

JTBD practitioners echo the concerns voiced by Christensen who argued that personas can not be used to explain why customers buy products [14], and Chapman and Millman, who argue that the validity of personas is impossible to verify [35]. JTBD posits that by focusing on the jobs that different 'types' of people have in common, these problems can be prevented [11].

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. Indeed, the lightweight template does not prevent practitioners from writing job stories that are highly solution-oriented.

Furthermore, because of job stories' similarity to user stories we would like to investigate how to extract concepts from job stories with natural language processing, similar to our earlier work on user stories [36]. 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 JTBD and job stories in software development. This will require thorough empirical research to determine the usefulness of the various activities and artifacts. An interesting follow-up study concerns how practitioners use job stories—and our method—to express nonfunctional requirements.

References

1. Potts, C., Bruns, G.: Recording the reasons for design decisions. In: Proceedings of the International Conference on Software Engineering, pp. 418–427. IEEE Computer Society (1988)
2. Yu, E.S., Mylopoulos, J.: Understanding "why" in software process modelling, analysis, and design. In: Proceedings of the International Conference on Software Engineering, pp. 159–168. IEEE (1994)
3. Zave, P., Jackson, M.: Four dark corners of requirements engineering. ACM Trans. Softw. Eng. Methodol. **6**(1), 1–30 (1997)
4. Lauesen, S.: Task descriptions as functional requirements. IEEE Softw. **20**(2), 58–65 (2003)
5. Dalpiaz, F., Franch, X., Horkoff, J.: iStar 2.0 language guide (2016). arXiv:1605.07767 [cs.SE]
6. Kassab, M.: The changing landscape of requirements engineering practices over the past decade. In: Proceedings of the International Workshop on Empirical Requirements Engineering, pp. 1–8 (2015)
7. Jeffries, R.: Essential XP: card, conversation, and confirmation. XP Magazine, August 2001
8. Cohn, M.: User Stories Applied: For Agile Software Development. Addison-Wesley Professional, Boston (2004)
9. Lucassen, G., Dalpiaz, F., van der Werf, J.M.E.M., Brinkkemper, S.: Improving agile requirements: the Quality User Story framework and tool. Requir. Eng. **21**(3), 383–403 (2016)
10. Paetsch, F., Eberlein, A., Maurer, F.: Requirements engineering and agile software development. In: Proceedings of the IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. 308–313. IEEE (2003)
11. Klement, A.: Replacing the user story with the job story (2013). goo.gl/fZ1iqe
12. Christensen, C.M., Hall, T., Dillon, K., Duncan, D.S.: Know your customers' "Jobs to Be Done". Harv. Bus. Rev. **94**(9), 54–62 (2016)
13. Klement, A.: When Coffee and Kale Compete. NYC Publishing, New York (2016)
14. Christensen, C.M., Anthony, S.D., Berstell, G., Nitterhouse, D.: Finding the right job for your product. MIT Sloan Manag. Rev. **48**(3), 38–47 (2007)
15. Ulwick, A.W.: Turn customer input into innovation. Harv. Bus. Rev. **80**(1), 91–97 (2002)
16. Klement, A.: Designing features using job stories (2013). goo.gl/NS889V

17. Johnson, L.: Jobs to be Done: a case study in the NHS, September 2017. goo.gl/gpaBpx
18. Levitt, T.: Marketing myopia. Harv. Bus. Rev. **38**(4), 24–47 (1960)
19. Christensen, C., Cook, S., Hall, T.: Marketing malpractice: the cause and the cure. Harv. Bus. Rev. **83**(12), 74–83 (2005)
20. Bettencourt, L.A., Ulwick, A.W.: The customer-centered innovation map. Harv. Bus. Rev. **86**(5), 109–114 (2008)
21. Intercom Inc.: Intercom on Jobs-to-be-Done (2017). ISBN 978-0-9861392-3-9
22. Eisenhardt, K.M., Graebner, M.E.: Theory building from cases: opportunities and challenges. Acad. Manag. J. **50**(1), 25–32 (2007)
23. Klement, A.: Your job story needs a struggling moment (2016). goo.gl/vsRC1d
24. Ulwick, A.W., Bettencourt, L.A.: Giving customers a fair hearing. MIT Sloan Manag. Rev. **49**(3), 62–68 (2008)
25. Carpenter, H.: A method for applying Jobs-to-be-Done to product and service design, January 2013. goo.gl/5NUVwh
26. van de Weerd, I., Brinkkemper, S.: Meta-modeling for situational analysis and design methods. In: Syed, M.R., Syed, S.N. (eds.) Handbook of Research on Modern Systems Analysis and Design Technologies and Applications, pp. 35–54. IGI Global, Hershey (2009)
27. Ulwick, A., Hamilton, P.: The Jobs-to-be-Done growth strategy matrix. Technical report, Strategyn (2016)
28. Christensen, C.M., Dillon, K., Hall, T., Duncan, D.S.: Competing Against Luck: The Story of Innovation and Customer Choice. Harper Business, New York (2016)
29. Schumpeter, J.A.: Socialism, Capitalism and Democracy. Harper and Brothers, New York (1942)
30. Lucassen, G., Dalpiaz, F., Werf, J.M.E.M., Brinkkemper, S.: The use and effectiveness of user stories in practice. In: Daneva, M., Pastor, O. (eds.) REFSQ 2016. LNCS, vol. 9619, pp. 205–222. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30282-9_14
31. Norman, D.A., Draper, S.W.: User-centered design. New perspectives on human-computer interaction (1986)
32. Dix, A.: Human-Computer Interaction. Pearson education, Prentice Hall Europe (1998). ISBN: 9780132398640. https://books.google.nl/books/about/Human_computer_Interaction.html?id=tNxQAAAAMAAJ&source=kp_cover&redir_esc=y
33. Endsley, M.R.: Designing for Situation Awareness: An Approach to User-Centered Design. CRC Press, Boca Raton (2016)
34. van Lamsweerde, A.: Goal-oriented requirements engineering: a guided tour. In: Proceedings of the IEEE International Requirements Engineering Conference, pp. 249–262. IEEE (2001)
35. Chapman, C.N., Milham, R.P.: The personas' new clothes: methodological and practical arguments against a popular method. Proc. Hum. Factors Ergon. Soc. Ann. Meet. **50**, 634–636 (2006)
36. Lucassen, G., Robeer, M., Dalpiaz, F., van der Werf, J.M.E.M., Brinkkemper, S.: Extracting conceptual models from user stories with visual narrator. Requir. Eng. **22**(3), 339–358 (2017)