Not all requirements prioritization criteria are equal at all times: A quantitative analysis

The Journal of Systems and Software 209 (2024) 111909

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Not all requirements prioritization criteria are equal at all times: A quantitative analysis☆

Richard Berntsson Svensson [a,*], Richard Torkar [a,b]

[a] Department of Computer Science and Engineering, Chalmers and University of Gothenburg, Gothenburg, Sweden
[b] Stellenbosch Institute for Advanced Study, Stellenbosch, South Africa

ARTICLE INFO

Keywords:
Requirements prioritization
Bayesian analysis
Empirical software engineering
Construct validity

ABSTRACT

Requirement prioritization is recognized as an important decision-making activity in requirements engineering . Requirement prioritization is applied to determine which requirements should be implemented and released. In order to prioritize requirements, there are several approaches/techniques/tools that use different requirements prioritization criteria, which are often identified by gut feeling instead of an in-depth analysis of which criteria are most important to use. Therefore, in this study we investigate which requirements prioritization criteria are most important to use in industry when determining which requirements are implemented and released, and if the importance of the criteria change depending on how far a requirement has reached in the development process. We conducted a quantitative study where quantitative data was collected through a case study of one completed project from one software developing company by extracting 32,139 requirements prioritization decisions based on eight requirements prioritization criteria for 11,110 requirements. The results show that not all requirements prioritization criteria are equally important, and this change depending on how far a requirement has reached in the development process. For example, for requirements prioritization decisions before iteration/sprint planning, having high Business value had an impact on the decisions, but after iteration/sprint planning, having high Business value had no impact.

Editor's note: Open Science material was validated by the Journal of Systems and Software Open Science Board.

1. Introduction

Requirements Prioritization (RP) is an important decision making task in software development (Herrmann and Daneva, 2008) where the objective is to determine, from a set of candidate requirements, which requirements are the most valuable and thus should be included in the product (Berander and Andrews, 2005), and in which order they should be implemented (Riegel and Doerr, 2015). Prioritizing requirements (i.e., determining the most valuable ones) involves making decisions based on one or several criteria, e.g., budget (Bukhsh et al., 2020), time constraints (Bukhsh et al., 2020), technical constraints (e.g., development cost and risk) (Riegel and Doerr, 2015; Shao et al., 2017; Pergher and Rossi, 2013), business aspects (e.g., market competition and regulations) (Pergher and Rossi, 2013), customer satisfaction (Pergher and Rossi, 2013; Shao et al., 2017), or business value (Riegel and Doerr, 2015; Daneva et al., 2013). The increasing number of requirements, both from internal (e.g., developers) and external (e.g., customers) sources, and from the availability of vast amount of data (big data) coming from digital networks connecting an increasing number of people, devices, services, and products (Berntsson Svensson et al., 2019), makes RP even more difficult.

Several RP techniques have been introduced in the literature (Pergher and Rossi, 2013; Achimugu et al., 2014; Hujainah et al., 2018; Bukhsh et al., 2020) to make RP accurate, efficient, and reliable (Bukhsh et al., 2020). For example, RP techniques based on new technologies such as machine learning and repository mining (Pergher and Rossi, 2013; Achimugu et al., 2016; Shao et al., 2017) (following the trend of big data in requirements engineering), or RP techniques based on established RP concepts such as Analytical Hierarchy Process, Numerical Assignment, Planning Game, and Cumulative Voting (Bukhsh et al., 2020; Riņkevičs and Torkar, 2013).

Regardless if the RP techniques are based on new technologies or established concepts, all use one or several criteria when prioritizing requirements. However, all techniques have limitations, not only related to, e.g., scalability and requirements dependencies (Achimugu et al., 2014; Shao et al., 2017), but also due to assumptions about

☆ Editor: Uwe Zdun.
* Corresponding author.
E-mail addresses: richard@cse.gu.se (R. Berntsson Svensson), richard.torkar@gu.se (R. Torkar).

https://doi.org/10.1016/j.jss.2023.111909
Received 3 December 2022; Received in revised form 23 July 2023; Accepted 20 November 2023
Available online 20 November 2023

project context (e.g., the order in which requirements that should be prioritized are presented to the stakeholders, available information during RP, and how the RP process looks like) (Riegel and Doerr, 2015; Bukhsh et al., 2020) and assumptions about which criteria should be used when prioritizing requirements, which is often decided based on gut feeling (Riegel and Doerr, 2015). Having a predefined set of criteria to be used in RP may lead to using misleading criteria (Riegel and Doerr, 2015), and thus making wrong/poor decisions. Hence, it is important to have flexible RP techniques where the used criteria are based on an in-depth analysis of which criteria are the most appropriate for a given context/project (Berander and Andrews, 2005). There are studies (e.g., Berntsson Svensson et al., 2011; Daneva et al., 2013; Jarzebowicz and Sitko, 2020) that have investigated which criteria are most commonly used in industry and/or most important/valuable when prioritizing requirements. Most (if not all) of these studies have investigated the RP criteria by asking industry practitioners for their subjective opinion concerning which criteria are most commonly used and/or most important when prioritizing requirements. However, the importance of various RP criteria and its content may be recalled differently among the practitioners due to memory bias (von Zedtwitz, 2002) and details may quickly be forgotten (Baird et al., 1999). Furthermore, reflection on purely experience-based memory recall (e.g., when asking practitioners about their subjective opinions) carries a high risk of drawing incorrect conclusions (Glass, 2002) and may result in emotional sessions rather than in constructive fact-based discussions. Therefore, to understand which RP criteria have an actual impact on RP decisions, an in-depth analysis (Berander and Andrews, 2005) based on actual RP decisions are needed to avoid practitioners' biases. To the best of our knowledge, no study has analyzed the actual outcome of RP decisions in industry to identify which RP criteria have an actual impact on RP decisions, or if the impact changes over time.

In this paper, we investigate which RP criteria have an actual impact in industry when determining which requirements are implemented and released to its customer and which ones are dropped. To this aim, we conducted a quantitative study of one completed project from one case company to investigate which criteria industry practitioners actually base their decisions on when prioritizing requirements, and if the criteria change depending on how far a requirement has reached in the development process. In order to investigate which RP criteria actually have an impact on RP decisions, we performed a quantitative study considering 32,139 RP decisions based on eight RP criteria for 11,110 features[1] from one completed project with 14 software development teams and five cross-functional teams. That is, we collected quantitative data through a case study. The extracted data was analyzed by designing, comparing, validating, and diagnosing ordinal Bayesian regression models employing a Sequential likelihood. In addition, ordered categorical predictors were modeled as category-specific effects. Finally, to better understand how these effects vary over time a conditional effects analysis was conducted.

The results of this quantitative study show that not all RP criteria have an equal impact on RP decisions, and that the impact of a criterion changes depending on how far a requirement has reached in the development process. For example, having a high business value has an actual impact on RP early in the development process, high customer value has an impact in the middle, while being a critical requirement only has an impact at the end of the development process. Moreover, one out of eight used RP criteria, namely number of key customers who believed the requirement is important, had no impact on RP. Although the criterion dependency to other requirements had a significant impact on RP at one point in time, it did not matter if the requirement had dependencies to other requirements or not.

The remainder of this paper is organized as follows. Section 2 presents related work, and an introduction to Bayesian Data Analysis. Section 3 describes the design of our quantitative study, while Section 4 presents the results. Section 5 discuss the findings and Section 6 discloses the threats to the validity of our study. Finally, Section 7 gives a summary of the main conclusions.

[1] The case company use the term feature for requirement.

2. Background and related work

This section presents related work on requirements prioritization. We conclude the section by providing a brief introduction to Bayesian data analysis.

2.1. Requirements prioritization

Several systematic literature reviews (SLRs) and systematic mapping studies have studied state-of-the-art in Requirements Prioritization (RP) (Herrmann and Daneva, 2008; Kaur and Bawa, 2013; Pergher and Rossi, 2013; Achimugu et al., 2014; Hujainah et al., 2018; Thakurta, 2017; Bukhsh et al., 2020). Herrmann and Daneva (2008) investigated RP techniques based on benefit and cost information and concluded that empirical validations of RP techniques where needed. Kaur and Bawa (2013) conducted an SLR and identified seven RP techniques that were compared and analyzed. The seven RP techniques were Analytic Hierarchy Process (AHP), value-oriented prioritization, cumulative voting, numerical assignment, binary search tree, planning game, and B-tree prioritization. The authors concluded that more work in the RP area is needed in order to improve the effectiveness—in terms of complexity and time consumption—of RP techniques. Pergher and Rossi (2013) performed a systematic mapping study focusing on empirical studies in RP. The authors identified that accuracy, time consumption, and ease of use were the most common criteria to use when evaluating RP techniques. Moreover, the results revealed that most studies in the RP area focus on RP techniques. Achimugu et al. (2014) conducted an SLR with the focus on RP techniques and their prioritization scales. The SLR identified 49 RP techniques that, in general, faced challenges related to time consumption, requirements dependencies, and scalability.

Later on, Hujainah et al. (2018) conducted an SLR to identify strengths and limitations of RP techniques. The results showed that RP is important for ensuring the quality of the developed system. In addition, 108 RP techniques were identified and analyzed based on, e.g., used RP criteria and limitations. In total, 84 RP criteria were used among the 108 RP techniques, where the criterion importance was the most frequently used. Moreover, the authors concluded that the existing RP techniques have limitations with regards to scalability, requirements dependencies, time consumption, and lack of quantification, which is in-line with the reported limitations in Achimugu et al. (2014). Thakurta (2017) performed a systematic mapping study focusing on understanding RP artifacts, which included the objective of RP and factors that influence the overall RP process. In a recent SLR, Bukhsh et al. (2020) evaluated the existing empirical evidence in the RP area, which did not only include empirical evidence related to RP techniques. The results show that AHP is the most accurate and commonly used RP technique in industry. Most of the focus in the RP literature is on proposing, developing, and evaluating RP techniques, and comparing the performance of existing RP techniques (Pergher and Rossi, 2013). The most common approach to evaluate RP techniques is by empirically evaluate two or more RP techniques, where AHP is commonly used as one of them (Bukhsh et al., 2020).

All RP techniques use one or several criteria for RP, where most of them use a fixed, predefined, set that are used during the RP process (Riegel and Doerr, 2015). However, the predefined criteria may not be suitable for all contexts. Thus, it is important to identify which criteria to use, and which ones are the most important to use given the context. Riegel and Doerr (2015) conducted an SLR to identify and categorize prioritization criteria. In total, about 280 prioritization criteria were extracted from the literature and categorized into six main

categories: benefits, costs, risks, penalties and penalty avoidance, business context, and technical context and requirements characteristics. The most frequently mentioned RP criteria in the literature were: implementation effort, resource availability, implementation dependencies, business value, customer satisfaction, and development effort. Hujainah et al. (2018) identified 84 RP criteria where importance was the most used criterion among the identified RP techniques, followed by cost, business value, value, and dependency. Thakurta (2017) identified several factors that influences RP, including requirements dependencies, software architecture, business value, and stakeholder roles.

Most of the identified RP criteria in the above literature comes from proposed RP techniques, and thus are selected based on gut feeling (Riegel and Doerr, 2015) and not importance. There are studies, e.g., Berntsson Svensson et al. (2011), Daneva et al. (2013), Jarzebowicz and Sitko (2020), that looked into which RP criteria are used/important in industry. Berntsson Svensson et al. (2011) investigated how RP is conducted in industry and which criteria are used when prioritizing requirements. The results show that cost, value, customer input, and/or no criterion are the most commonly used criteria in industry for RP. In another study, Daneva et al. (2013) found that the understanding of requirements dependencies is important for RP, and that the two most important RP criteria are business value and risk. Jarzebowicz and Sitko (2020) investigated agile RP in industry. The results show that business value is the most commonly used RP criterion, but other criteria such as complexity, stability, and interdependence are also used. However, these studies are based on the practitioners' subjective opinion about which RP criteria are important, and not on an in-depth analysis based on actual RP decisions.

The above indicates that the focus have been on comparing RP techniques and not on what we should measure, i.e., the criteria. Ultimately, in all analysis, what you measure and how you measure it, is more important than the actual analysis. To this end we focus on an analysis technique that allows us to take prior knowledge into account, handles disparate types of data, uses generative models, and quantifies uncertainty through probability theory, in order to investigate what effect different measurements have on RP.

2.2. Bayesian data analysis

Lately, many tools and probabilistic programming languages have been developed to tackle some of the challenges we face when designing more powerful statistical models. In our view, several things have improved. First, probabilistic programming languages, e.g., `Turing.jl` or `Stan`, have matured.[2] Second, resampling techniques based on Markov chain Monte Carlo (MCMC) have improved (Brooks et al., 2011). Third, principled and transparent procedures for conducting Bayesian inference using the mentioned techniques now exist (Talts et al., 2018; Gabry et al., 2019; Gelman et al., 2017; Betancourt, 2019), and are being continuously improved (Vehtari et al., 2021).

In our case we have decided to use the dynamic Hamiltonian Monte Carlo implementation that Stan provides (Brooks et al., 2011). Mainly because it is considered to be the gold standard in the field, which many research groups use when benchmarking their algorithms. Additionally, compared to other MCMC implementations, Stan provides a plethora of diagnostics to ensure validity and reliability of the findings.

If we further contrast the above with how analyses are done in computer science and software engineering research today (Gomes de Oliveira Neto et al., 2019), we feel that a principled Bayesian workflow serves us well. In short, issues such as the arbitrary $\alpha = 0.05$ cut-off, the usage of null hypothesis significance testing and the reliance on confidence intervals have been criticized (Ioannidis, 2005; Morey et al., 2016; Nuzzo, 2014; Woolston, 2015), and when analyzing the arguments, we have concluded that there is a need to avoid many of the issues plaguing other scientific fields.

In this paper we expect the reader to have knowledge regarding design of statistical models. In our particular case we will conduct linear regression, however our outcome (dependent variable) is of an ordered categorical nature (i.e., compared to a count the differences in value is not always equal), as are some of our predictors (independent variables) (Bürkner and Charpentier, 2020).[3] To this end, we will design, compare, validate, and diagnose ordinal Bayesian regression models with the purpose of propagating uncertainty and making probabilistic statements by using a posterior probability distribution as explained by, e.g., Bürkner and Vuorre (2019), Furia et al. (2021), Torkar et al. (2021), Furia et al. (2021).

To summarize the development of statistical models in this paper we refer interested readers to the replication package.[4] However, we will provide a rudimentary overview of the steps involved (the below is a summary of what an interested reader can find in Furia et al. (2021)).

First, model design begins with assumptions about the underlying data generation process (i.e., the likelihood). As the reader will see, the outcome in the data is ordered categorical, which leaves us with a number of options. We decide on the most appropriate likelihood by using ontological and epistemological arguments (see replication package).

Second, once a rudimentary model has been developed one needs to set prior probability distributions (priors) on all parameters that we want estimated. To ensure that we are not overfitting, i.e., learning too much about the data, it is important that we check what the combinations of priors imply on our outcome. This is called prior predictive checks.

Third, once the model has been sampled, we check diagnostics to ensure that we have reached a stationary (stable) posterior probability distribution. If this is the case, we then check how well the model fits the empirical data; this is called posterior predictive checks.

Fourth, we next repeat the steps above and design more models. We then use Kullback–Leibler divergence (Kullback and Leibler, 1951) to decide on which model is the best (relative to other models) concerning out-of-sample predictions (i.e., which model deals best with new data); this is ultimately a cross-validation approach.

However, just because we have a 'best' model, that does not mean that this will be true for all eternity. The approach above is iterative, and over time we learn more about the studied phenomenon and, hence, models will evolve over time when new evidence is added.

3. Study design

The aim of this study is to empirically evaluate the impact RP criteria have on decisions when prioritizing requirements, and if the criteria change depending on how far a requirement has reached in the development process. To address the aim of this study, a quantitative study where quantitative data was collected through a case study (Wohlin et al., 2003) was used. Data was collected from one completed software project from one software developing company. The following research questions (RQ) provided the focus for the empirical investigation:

- **RQ1:** Which of the used requirements prioritization criteria, by the case company, have an actual impact when determining which requirements should be implemented and released?
- **RQ2:** Does the impact of requirements prioritization criteria change depending on how far a requirement has reached in the development process?

[2] See https://turing.ml and https://mc-stan.org.

[3] Throughout the paper we use the terms variate, predicted variable, dependent variable, and outcome interchangeably. The same applies to the terms covariate, independent variable, and predictor. Since the paper's focus is on *features* in requirements engineering, we refrain from using that term in connection to our statistical analysis.

[4] https://github.com/torkar/feature-selection-RBS DOI: 10.5281/zenodo.4646845.

3.1. Project selection criteria

We conducted our analysis on one completed software project from one software developing company from our industry collaboration network. The software developing company has a large number of completed and ongoing projects. Thus, in order to select a project to be analyzed, four criteria were identified that needed to be satisfied:

- *Criterion 1: Completed project*. It was important for the studied project to be completed in order to analyze all requirements and decisions during the project's life cycle. Thus, we avoided projects with a short development time, e.g., projects that was only 50% completed, since these projects would have an incomplete number of requirements and decisions made.
- *Criterion 2: More than one criterion*. About 280 different RP criteria have been identified in the literature (Riegel and Doerr, 2015), while other studies, e.g. Berntsson Svensson et al. (2011), Thakurta (2017), Maalej et al. (2016), have identified different criteria that are considered important in RP. Therefore, in order to analyze which criteria actually have an impact on the decisions in industry, it was important to analyze a project that used several different RP criteria.
- *Criterion 3: Complete information*. We needed reliable data in order to produce a healthy dataset (the most important aspect in any statistical analysis is the data, not what approach one uses). To that end, all information and data about the requirements and the RP decisions needed to be documented and complete (i.e., no missing data/value/information about the requirements, RP criteria, or decisions made). This includes that all requirements' states should be documented, all used RP criteria including their values should be complete (i.e., no missing values), and all decisions (from RP) needed to be documented.
- *Criterion 4: Large number of requirements and decisions*. In order to fully understand which RP criteria have an impact in industry, our studied project could not be a too simple example with only a few requirements and decisions made. Therefore, it was important that the studied project had a large number of requirements and decisions made, which could be seen as representative of a project at larger software company.

These four criteria allowed us to (i) identify a project that the company identified as a representative project of the case company, i.e., purposive sampling (Baltes and Ralph, 2020) ensuring representativeness, (ii) discard projects having a short development time with few requirements and decisions, and (iii) discard projects with only one or a few RP criteria. A "gate-keeper" at the case company identified a suitable project that fulfilled all four criteria.

3.2. Characteristics of the case company and the selected project

The case company develops software for embedded products in a global consumer market. In the targeted organization of the case company (i.e., where the studied project belongs), there are about 1000 employees in software development. The software development model used by the targeted organization is a continuous development model influenced by Scrum to allow for coordination with hardware and product projects. Requirement engineering is partly handled by the business department (as part of the cross-functional teams) and partly by the software development teams. The targeted organization use cross-functional teams that include customer representatives for key customers, which is either a representative from the real key customer or a customer proxy assigned by the business department. The cross-functional teams have full responsibility for defining, prioritizing, implementing, and testing features. A feature is developed and prioritized by one cross-functional team. In the targeted organization, 10–15 software development teams work in a typical project in several cross-functional teams for a duration of 2–3 years. A typical project has between 10,000 and 15,000 features.

Table 1
Characteristics of the analyzed software project.

Characteristic	#
Features	11,110
Decisions	32,139
Requirements prioritization criteria	8
Development teams	14
Cross-functional teams	5

The project in focus for this study, i.e., the studied project, is one of the targeted organization's products. The studied project had a lead time close to three years from start to closure. In total, 14 software development teams in 5 cross-functional teams were involved in the development of the software for the embedded product. In total, the studied project had 11,110 features. Fig. 1 illustrates the structure of the features and what level of details the features had (written as user stories, natural language, and use cases). Note that the features in Fig. 1 are not the real features that were used in the analyzed project (due to confidentiality reasons, the used features are not allowed to be revealed). A feature could be in one of seven states, as shown in Fig. 3.

The different states are described in Section 3.3. The state of a feature shows how far the feature has reached in the development process. Before a feature was assigned to a state, a RP decision was made. All features were prioritized based on eight RP criteria by the responsible cross-functional team. The cross-functional team that developed and prioritized the feature was also responsible for collecting and recording the value for each RP criteria. In total, 32,139 RP decisions were based on the eight RP criteria. Table 1 provides a description of the studied software project. The eight RP criteria are described in Table 2 together with the recorded values for each RP criterion.

3.3. Data extraction

We extracted data from three databases from the case company of the studied project. The first database contained all features of the studied project, the second database contained all states for all features, and the third database contained all RP criteria and its values for each feature of the studied project. Fig. 2 provides an overview of our data extraction steps, which are described below.

(D1) Extract all features. The first step in the data collection and extraction phase was to extract all features that were ever considered from the completed project. For each feature, a unique ID (FeatureID) was extracted, which was used to link the feature to all RP decisions and state(s) the feature reached in the development process (see D2 below), and to all values it had for each requirement prioritization criteria (see D3 below). In total, 11,110 features were extracted.

(D2) Extract all states for each feature. When features are discovered it is not certain if the feature will be included in the product release. Available resources, scope, and lead-time limits the realization of any feature into the product. Therefore, to keep track of all features through the software development process, a feature can have one of seven states, namely: elicited, prioritized, planned, implemented, tested, released, or dropped (illustrated in Fig. 3). The state of a feature shows how far the feature has reached in the development process. Before a feature is given a state (the first state is elicited), a decision (i.e., RP decision) was made to include that feature in the project. All extracted features from D1 reached at least the state of elicited, and thus is considered to be included in the project. Then, before a feature changes its state, a new RP decision based on eight criteria (see D3 below and Table 2) was made. A feature could move (backward or forward) from one state to another, meaning a feature could have been in one or several states. When extracting all states from the second database, the FeatureID was used to link each feature to all its states in the project. In

As a user I want to protect my device so only authorized persons have access to the information. We need to comply with X.

Comments:
Local function which is enforced by central management. Time-out must be configureable.

Suggested solution:
Depending on interaction design a function may be prompted every time the user access sensitive information.

Affected features:
Feature A and Feature B

Use Case - Register X on agreement level

This Use Case is executed in an agreement. The purpose is to add agreement service X including Y and Z. The agreement service X generates transactions for activation on platforms. Activation is done in the window for the agreement of service X according to standard form. Y consists of at least two sub-Y's, and must be activated before the activation of Z. Y is selectable to add X. Each sub-Y is added in the platforms and invoicing will start at a given start date. Change: On the service X it should be possible to mark a sub-Y. When this is marked, the sub-Y will be available to connect from the service X

As an administrator I want to have the following functions:
- Configure device parameters
- Configure certificates
- Configure user settings

The system shall receive encrypted data and being able to decrypt and save the data.

The response time for function X shall not exceed Y ms

As a Customer the lead-time shall not be affected by differentiation of variants

Fig. 1. Example of structure and what level of details the features had in the analyzed project.

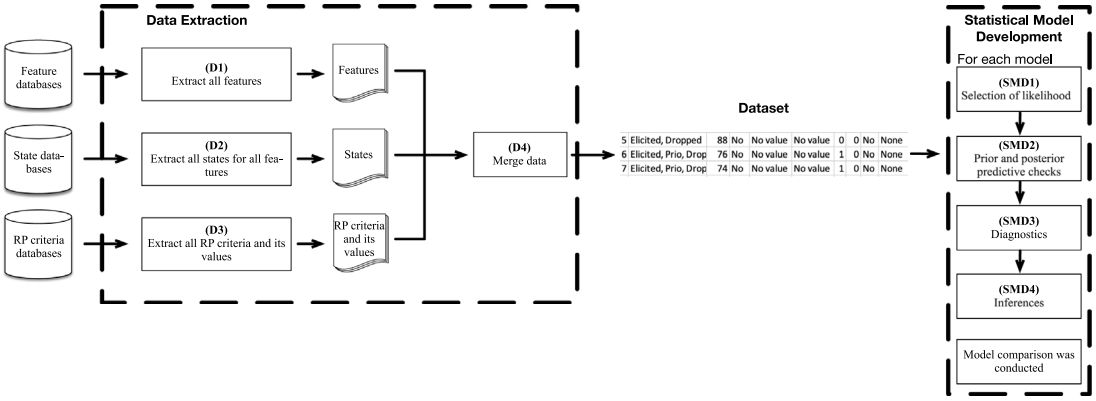

Fig. 2. Overview of the study design.

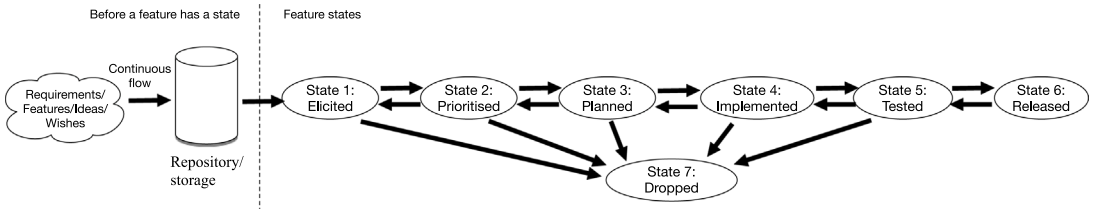

Fig. 3. Overview of feature states.

this study, we only extracted the forward transitions of each feature. In total, 32,139 decisions were extracted. Fig. 3 shows the different states for a feature, which are described below.

State Elicited: Each feature that has been through a pre-feasibility phase, and being prioritized (i.e., a decision is made to include the feature for the next step) reach the state Elicited.

State Prioritized: At regular intervals, features with the state of Elicited are being reviewed in a feasibility review for possible inclusion into the product. After the feasibility review, a decision (i.e., RP) is made. Features that are prioritized to be included get the state Prioritized.

State Planned: Features that have the state Prioritized are being reviewed in an analysis review. In the analysis review, each feature is analyzed based on, e.g., scope, adding details to the feature, estimations, and a more elaborate specification of the feature is created. After the analysis, the features are prioritized (i.e., a decision is made) to be included in the product or not. All features that are prioritized to be included in the product get the state Planned. These features are input for the design, coding, and iteration/sprint planning. Finally, these features are added to the product backlog.

State Implemented: From the product backlog, features are selected (i.e., prioritized) for development. When the features are developed, which includes technical design, coding, and unit tests, the features get the state Implemented.

State Tested: Although the implemented features include some testing, e.g., unit tests, a decision (RP) is made about which feature will be included for a more through testing process in order to ensure adequate level of quality before an implemented feature is released. Features that are selected for, and pass the testing, get the state Tested.

State Released: When all activities have been completed for the features with the state Tested, a decision (i.e., RP) is made about which features should be released. The features that are selected for being released get the state Released.

Table 2
Data dictionary. From left to right, variable name, encoding used in our model, possible values, description of the variable, and the type of variable. For type, $\mathbb{N}$ indicates a natural number, $\mathcal{O}$ indicates ordered data, and $\mathbb{Z}_2$ is binary data. The first row is our outcome variable, `State`.

Variable name	Code	Possible values	Description	Type
ID	n/a	1,...,11 110	Unique ID for each feature	$\mathbb{N}^+$
State	state	Elicited (1), Prioritized (2), Planned (3), Implemented (4), Tested (5), Released (6), Dropped (7)	A feature's state that shows how far the feature reached in the process	$\mathcal{O}$
Team priority	prio	0,...,1000	The relative priority a feature was assigned by the team	$\mathbb{N}^0$
Critical feature	crit	Yes/No	If a feature was considered to be critical for the product	$\mathbb{Z}_2$
Customer value	c_val	No value, Valuable, Important, Critical	How valuable the feature was considered to be for customers	$\mathcal{O}$
Business value	b_val	No value, Valuable, Important, Critical	The business value of the feature	$\mathcal{O}$
Stakeholders	sh	0,...,10	Number of key internal stakeholders who considered a feature important	$\mathbb{N}^0$
Key customers	kc	0,...,60	Number of key customers who considered a feature important	$\mathbb{N}^0$
Dependency	dep	Yes/No	If a feature has a dependency to other features	$\mathbb{Z}_2$
Architects' involvement	arch	None, simple, monitoring, active participation, joint design	The needed level of involvement from architects in order to design/implement a feature	$\mathcal{O}$

State Dropped: A feature can be rejected/dropped at any time in the process (until state Released). These features get the state Dropped. Dropped features are not deleted from the backlogs/repositories/storage to enable future analysis.

(D3) Extract all RP criteria and its values. The FeatureID and States were used to extract the RP criteria and its values for each feature and its state(s). We extracted data from all RP criteria that were used when prioritizing a feature in this project. In total, eight different RP criteria were used each time a feature was prioritized. The extracted RP criteria are: team priority, critical Feature, customer value, business value, stakeholders, key customers, dependency, and architects' involvement, which are described in Table 2. Note, we did not decide how many or which RP criteria were to be used in the analyzed project. This decision was made by the company before the project started.

(D4) Merge data. After D3 was completed, we merged all extracted data from all three databases (D1–D3) using FeatureID and State. This allowed us to remove incomplete information, e.g., features without a state and empty values for the RP criteria. Table 2 provides an overview of all extracted data from D1–D3.

Let us now examine the outcome variable in particular, i.e., State. A *State* is the state a feature has reached. Before a feature can reach the next higher state (e.g., moving from State Elicited to State Prioritized) or being dropped (i.e., moved to State Dropped), a RP decision is made. This RP decision is called a *cutpoint*. Meaning, for our six states (Elicited, Prioritized, Planned, Implemented, Tested, and Released) there are five cutpoints, cutpoint 1 is between states Elicited and Prioritized, cutpoint 2 between states Prioritized and Planned...and cutpoint 5 is between states Tested and Released. The result from the RP decision is either that a feature reached the next higher state or it is dropped. This result is called an *outcome*. Meaning, in our statistical model, RP happens at five different cutpoints (decision points) that control the exits of states 1–5, which decides whether a feature reaches the next higher state or being dropped.

Due to non-disclosure agreements, the empirical data, e.g., FeatureID, variable names, and values, are not allowed to be revealed. Hence, we generated a synthetic dataset with describing names and values. The modifications of the real data include, changing the real FeatureID to a random ID without replacement from 1 to 11,110. Moreover, all variable names (column Variable in Table 2) were changed to descriptive and generic names that described the purpose of the variable, inline with current literature. In addition, the values (column Value(s) in Table 2) have been modified to descriptive values. For example, the values for the variable State are changed to names that describes the state of a feature. Section 3.4 provides descriptive statistics of the merged data.

3.4. Descriptive statistics of the merged data

Table 2 provides a short description of the variables for each feature, while Fig. 4 presents their frequencies. For the variables `Dependency` and `Critical feature`, the answer was Yes/No and the ratio was 2004/9106 and 1948/9162, respectively.

If we examine our outcome `State` (Fig. 4(a)) we see that approximately 3000 (out of 11,110) features are dropped already in the first state (Elicited → Dropped) and approximately 3000 reach the final state (Elicited → … → Released). After the initial `State` 1, fewer and fewer are dropped up to, and including, `State` 5.

Concerning `Number of stakeholders` (Fig. 4(b)), the absolute majority of the features have only one, while for `Number of key customers` (Fig. 4(c)), most features have zero. Finally, concerning the variable `Priority` (Fig. 4(d)), most features have zero in priority, and are then, more or less, spread out to priority 1000, where there is another peak.

Looking at the variables that eventually will be modeled as category-specific effects (i.e., they being ordered categorical) one can see that for `Architects' involvement` (Fig. 4(e)) almost 90% of the features do not have any architects involved. Additionally, for `Business value` and `Customer value` (Figs. 4(f) –4(g)) the distributions are comparable, where the first step, 'No value', has been set 85%–95% of the time.

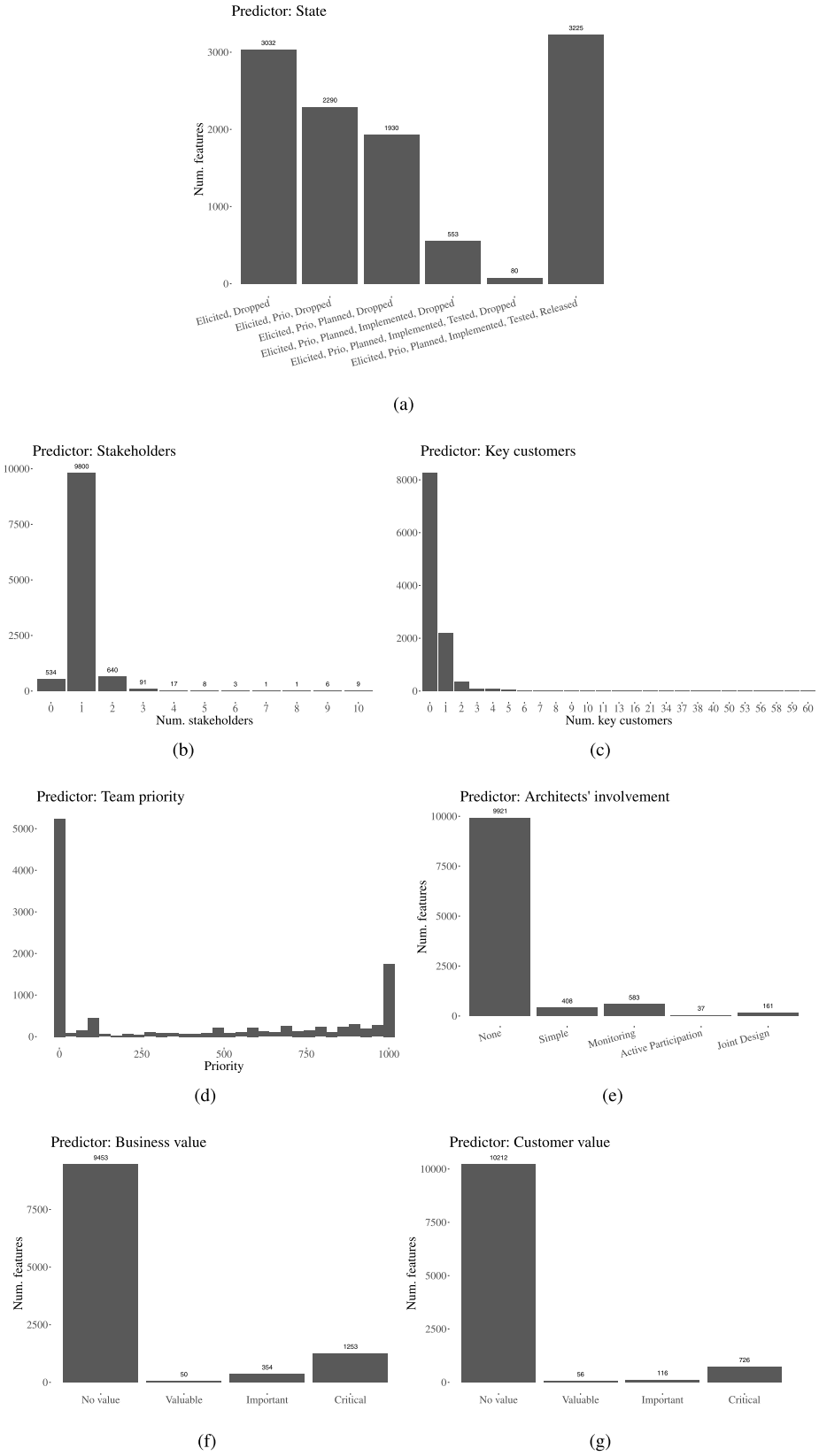

Fig. 4. Plots of outcome (a) and predictors (b–g), where e–g are ordinal independent variables. The *y*-axis represents the frequency. Two additional predictors are not plotted (`Dependency` and `Critical feature`).

After gaining some insight concerning the variables, we next turn our attention to statistical model design where we design, compare, validate, and diagnose statistical models to conduct inferences. (Fig. 2 provides an overview of our statistical model design.)

All steps in the analysis can be replicated by downloading the replication package, and preferably install `Docker`. The empirical data used in this manuscript is unfortunately not generally available due to an NDA. However, we have generated a synthetic dataset so anyone can follow the analysis step-by-step, and reach very similar results.

3.5. Model design

There are several ways to model ordered categorical (ordinal) data, but not until quite recently was it possible to use them easily in Bayesian data analysis. Software engineering, generally speaking, handles ordered categorical data by assuming that the conclusions do not depend on if a regression or ordinal model is used. The problem is, of course, that relying on an incorrect outcome distribution will lead to subpar predictive capabilities of the model (Bürkner and Vuorre, 2019). This, in combination with the fact that effect size estimates will be biased when averaging multiple ordinal items, and that data can be non-normal, is something a researcher should want to handle (Liddell and Kruschke, 2018).

Today, we have at least three principled ways to model ordinal data: Adjacent category (Bürkner and Vuorre, 2019), Sequential (Tutz, 1990), and Cumulative models (Walker and Duncan, 1967). These models have been developed and refined in a Bayesian framework mostly because of needs from other disciplines, such as psychology (Bürkner and Vuorre, 2019).

First, Adjacent category models can be used when predicting the number of correct answers to several questions in one category (think of a math module for the SAT or the PISA tests) (Bürkner and Vuorre, 2019). We could perceive that our underlying data-generation process could be modeled this way.

Sequential models, on the other hand, assume that the outcome results from a sequential process and that higher responses are only possible if they pass lower responses; which is very much the case for our outcome `State`.

Finally, Cumulative models assume that the outcome, e.g., observed Likert scale values, stems from a latent (not observable) continuous variable (Bürkner and Vuorre, 2019).

In the case of Sequential models, we can model ordinal predictors as category-specific effects, while in Cumulative models, predictors are modeled as monotonic effects, the latter in order to avoid negative probabilities (Bürkner and Charpentier, 2020).

The main reason for modeling predictors as category-specific is to gain a more fine-grained view of the effect a predictor has on the outcome (i.e., how much does the predictor affect each outcome, `State` $1, \ldots, 6$). In short, we want to model a predictor as an effect on 6 ordered categories we use as *outcomes*. The assumption that predictors have constant effects across all categories may be relaxed now, leading us to employ category-specific effects.

As an example, consider our predictor `Architects' involvement`; it is quite likely that this predictor affects the outcome (`State` $1, \ldots, 6$) differently. Without using category-specific effects, this pattern would not be seen. In our case, we will later see that some category-specific effects are 'significant'.

(SMD1) Selection of likelihood. The first step concerning model design is often to decide which likelihood to use for inference, the Cumulative, Sequential, or Adjacent-category. This can be done by designing six statistical models and approximating their pointwise out-of-sample prediction accuracy (a measure of the out-of-sample fit). Doing this will allow us to receive estimates of how well each model handles new data (this, we would claim, is state of the art concerning model comparison, as introduced by Vehtari et al. (2017)). In Table 3, the result of the model comparison is presented, and it is clear that the Sequential model with predictors modeled as category-specific effects (where possible), has relatively speaking better out of sample prediction accuracy.

Table 3
From left to right, the model names, a short description indicating what type of model this is, and then the difference in expected log pointwise predictive density and the standard error. The abbreviation 'cs' is short for category-specific effects, while 'mo' is short for monotonic effects. The below is a ranked list so the top model is considered to have the best relative out of sample prediction capabilities. Comparing the first and second model, since Δelpd is > 4x larger than the ΔSE, one could claim that the models do not have similar predictive performance (Magnusson et al., 2020). One can also see that using category-specific effects make a difference (1st vs. 2nd model) and that Sequential likelihood models are better than the alternatives (1st and 2nd vs. the other models).

Model	Description	Δelpd	ΔSE
$\mathcal{M}_{s[cs]}$	Sequential w/cs	0.0	0.0
$\mathcal{M}_{s}$	Sequential w/o cs	−43.6	10.8
$\mathcal{M}_{ac}$	Adjacent-category	−61.4	15.4
$\mathcal{M}_{c[mo]}$	Cumulative w/mo	−148.1	20.3
$\mathcal{M}_{c}$	Cumulative w/o mo	−148.6	20.2
$\mathcal{M}_{0}$	Cumulative w/o predictors	−3318.0	63.2

If we examine Table 3, there are several things it tells us. First, the model on the first row, $\mathcal{M}_{s[cs]}$, is different enough to warrant a first place. How do we know that? We can calculate the confidence interval (CI) between the first and second models, i.e., $-43.6 \pm 10.8 \cdot 2.576$ (2.576 is the z-score for the 99% CI), which leads to $\text{CI}_{99\%}[-74.42, -15.78]$. In summary, on the 99%-level, the first model is significantly better than the second (since the difference does not cross zero). The only difference between the two first models is that we model predictors, when possible, as category-specific effects.

Next, it is also notable that we do not see the same effect using a Cumulative model and modeling predictors as monotonic (rows 4–5). Finally, the last line is our null model ($\mathcal{M}_0$), which is a model that does not use any predictors and, thus, only models the mean. By looking at Δelpd, we see that adding predictors to our model (rows 1–5) has a clear effect compared to $\mathcal{M}_0$. Hence, the conclusion concerning the model comparison is that the Sequential model, using category-specific effects, is our target model for now $\mathcal{M} = \mathcal{M}_{s[cs]}$. Next, we need to set appropriate priors.

(SMD2) Prior and posterior predictive checks. For our candidate model, we have several parameters in need of appropriate priors. One way to decide on priors is to make sure that the combination of all priors should be nearly uniform on the outcome scale and that impossible values should not be allowed.

Using a Sequential(ϕ, κ) model we know that more probability mass could be set in the beginning (potentially all features could be dropped in `State` 1), and then we should assign less probability mass for each following level in our outcome; we have six categories in our outcome, i.e., `State` 1–6 (Fig. 4(a)).[5] The complete model design for $\mathcal{M}$ is thus,

$$\text{State}_i \sim \text{Sequential}(\phi_i, \kappa) \tag{1}$$

$$\text{logit}(\phi_i) = \beta_1 \cdot \text{prio}_i + \beta_2 \cdot \text{crit}_i + \beta_3 \cdot \text{cs}(\text{b\_val}_i) \tag{2}$$

$$+ \beta_4 \cdot \text{cs}(\text{c\_val}_i) + \beta_5 \cdot \text{sh}_i + \beta_6 \cdot \text{kc}_i \tag{3}$$

$$+ \beta_7 \cdot \text{dep}_i + \beta_8 \cdot \text{cs}(\text{arch}_i) \tag{4}$$

$$\beta_1, \ldots, \beta_8 \sim \text{Normal}(0, 1) \tag{5}$$

$$\kappa \sim \text{Normal}(0, 2) \tag{6}$$

[5] Conventions for writing mathematical forms of Sequential models vary somewhat, but we will use Sequential(ϕ, κ), where ϕ is our linear part and κ the intercepts we want to estimate, i.e., the cutpoints between each step in the outcome. For the six levels in the outcome (`State` $1, \ldots, 6$), we need $6 - 1 = 5$ cutpoints. The first cutpoint is the border between `State` 1 and 2, and the last cutpoint is the border between `State` 5 and 6. This way we can estimate the probability mass for each state.

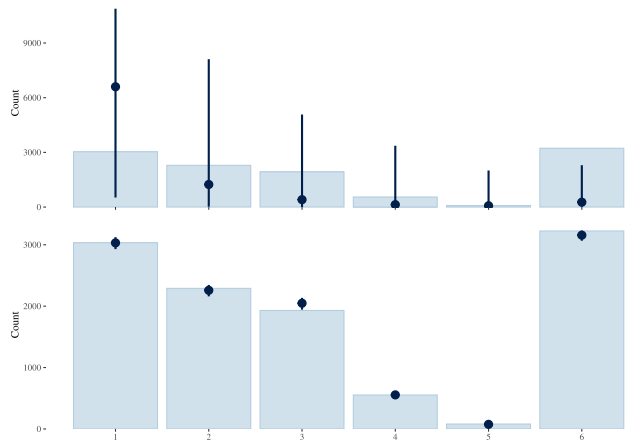

Fig. 5. The top plot is the prior predictive check (we only sample from the priors and use no empirical data), while the bottom plot is the posterior predictive check (when we have used empirical data). The dots indicate the mean while the lines indicate the 95% credible interval. The bars in both plots are our outcomes `State` $1, \ldots, 6$ (note the different scales on the vertical axis). The combination of our priors (top plot) shows that we are expecting a negative slope giving more probability mass to the first category and then less for each following category. After making use of the empirical data (bottom plot) one can see a good fit (i.e., there is very little uncertainty around each mean and the means are placed close to the top of each bar). If we would see the same pattern as in the top plot and more uncertainty around each mean, that could imply the priors had too strong influence on the empirical data.

Table 4
Summary of population-level (fixed) effects. From left to right, the name of the effect, estimate, estimation error (a parameter's posterior standard deviation), and lower and upper 95% credible intervals. Significant effects (95%) are in bold and left-aligned. The rows **bus. value [1,...,5]**, **cust. value [1,...,5]**, and **arch. involv. [1,...,5]**, are the five cutpoints which we use to estimate the deviation on each outcome (`State` $1, \ldots, 6$). Since we have six outcomes we have $6 - 1 = 5$ cutpoints (i.e., the borders between the six outcomes, `State` $1, \ldots, 6$).

Effect	Estimate	Est. Error	l-95% CI	u-95% CI
priority	**1.22**	**0.02**	**1.18**	**1.26**
criticality	**0.62**	**0.05**	**0.52**	**0.71**
stakeholders	**−0.05**	**0.02**	**−0.08**	**−0.02**
key customers	0.01	0.02	−0.02	0.04
dependency	**0.09**	**0.04**	**0.01**	**0.18**
bus. value [1]	**0.19**	**0.03**	**0.13**	**0.25**
bus. value [2]	−0.04	0.03	−0.10	0.02
bus. value [3]	**0.15**	**0.04**	**0.06**	**0.23**
bus. value [4]	**−0.18**	**0.06**	**−0.31**	**−0.06**
bus. value [5]	−0.09	0.14	−0.35	0.19
cust. value [1]	0.00	0.04	−0.07	0.08
cust. value [2]	0.05	0.04	−0.04	0.13
cust. value [3]	**0.13**	**0.06**	**0.02**	**0.24**
cust. value [4]	**0.19**	**0.08**	**0.04**	**0.35**
cust. value [5]	−0.01	0.16	−0.33	0.32
arch. involv. [1]	**0.13**	**0.05**	**0.03**	**0.22**
arch. involv. [2]	**0.09**	**0.05**	**0.00**	**0.18**
arch. involv. [3]	0.03	0.05	−0.06	0.13
arch. involv. [4]	−0.23	0.06	−0.34	0.11
arch. involv. [5]	**−0.29**	**0.13**	**−0.52**	**−0.03**

On the first line we assume that `State` is modeled using a `Sequential` likelihood. On Lines 2–4 we provide our linear model with all predictors and the parameters we want to estimate $(\beta_1, \ldots, \beta_8)$. Ordered categorical predictors are modeled as category-specific effects, i.e., `cs()`. As is evident, we model ϕ with a `logit` link function (Line 2), in order to translate back to the log-odds scale from the probability scale $(0, 1)$.

Finally, on Lines 5–6, we set priors on our parameters. The intercept (cutpoints) priors for κ are wider since we can expect them to vary more, while for our β parameters $\mathcal{N}(0, 1)$ might seem very tight, it still implies a prior variance of $\sigma^2 = (1 \cdot 8)^2 = 64$ for the model.

A visual view can be given by sampling from the priors only, i.e., prior predictive checks, and with priors and data, i.e., posterior predictive checks (see Fig. 5 for a comparison).

(SMD3) Diagnostics. When using dynamic Hamiltonian Monte Carlo we have a plethora of diagnostics, which we should utilize to ensure validity and efficiency of sampling (Brooks et al., 2011). Validity concerns the degree one can trust the results, while efficiency is an indicator that, while we might be able to trust the results, the results could be imprecise.

Here follows a short summary of the most common diagnostics and the outcome of these diagnostics for $\mathcal{M}$.

There should be no divergences since it is an indication that the posterior is biased (non-stationary); it mainly arises when the posterior landscape is hard for HMC to explore (a validity concern). No divergences were reported.

Tree depth warnings are not a validity concern but rather an efficiency concern. Reaching the maximum tree depth indicates that the sampler is terminating prematurely to avoid long execution time (Homan and Gelman, 2014). No warnings were reported.

Having low energy values (E-BFMI) is an indication of a biased posterior (validity concern). No warnings were reported.

The $\widehat{R}$ convergence diagnostics indicates if the independent chains converged, i.e., explored the posterior in approximately the same way (validity concern) (Vehtari et al., 2021). It should converge to 1.0 as $n \to \infty$. The $\widehat{R}$ diagnostics was consistently < 1.01, which is the current recommendation.

The effective sample size (ESS) captures how many independent draws contain the same amount of information as the dependent sample obtained by the HMC algorithm, for each parameter (efficiency concern). The higher, the better. When ESS ≈ 0.1 one should start to worry, and in absolute numbers we should be in the hundreds for the Central Limit Theorem to hold. The ESS diagnostics was consistently > 0.2.

Finally, Monte Carlo Standard Error (MCSE) was checked for all models. The MCSE is yet another diagnostic that reflects effective accuracy of a Markov chain by dividing the standard deviation of the chain with the square root of its effective sample size (validity concern).

In the replication package accompanying this paper the statistical validity and efficiency was checked for all models (see Sects. 2.*.3 in the replication package). All models passed all checks.

Having reached some confidence that the target model is representing the data generation process adequately, while assuring the validity and efficiency concerning the sampling, we next turn our attention to model validation (as opposed to validation of the output from the sampling algorithm).

First, for each model we conducted posterior predictive checks to see that the model captured the regular features of the data. Second, all model comparison was conducted using LOO (Vehtari et al., 2017), which relies on approximate leave-one-out cross-validation. LOO was selected since it has good diagnostics warning the user of suspect results; compared to other techniques (i.e., WAIC, BIC, AIC), that lack such diagnostics.

(SMD4) Inferences. The next section will provide results from the model by listing all parameter estimates (in our case the cutpoints, κ, are not relevant, but we shall focus on $\beta_1, \ldots, \beta_8$) and plot them.

In particular, we will analyze the category-specific effects that were modeled. Does the fine-grained view, which the category-specific modeling of predictors provides us with, tells us a story about how each predictor, affects each outcome, i.e., `State` $1, \ldots, 6$?

Finally, we will present a number of conditional effects. The latter concept is an excellent way to better understand the effect a specific predictor has on the six outcomes. Not only the size of the effect will be visible, but also how it varies depending on a number of factors.

The analysis follows the guidelines we present in previous work (Furia et al., 2021; Furia et al., 2021), but for brevity, we do not discuss here the application of the guidelines, but refer the interested readers to the replication package for details.

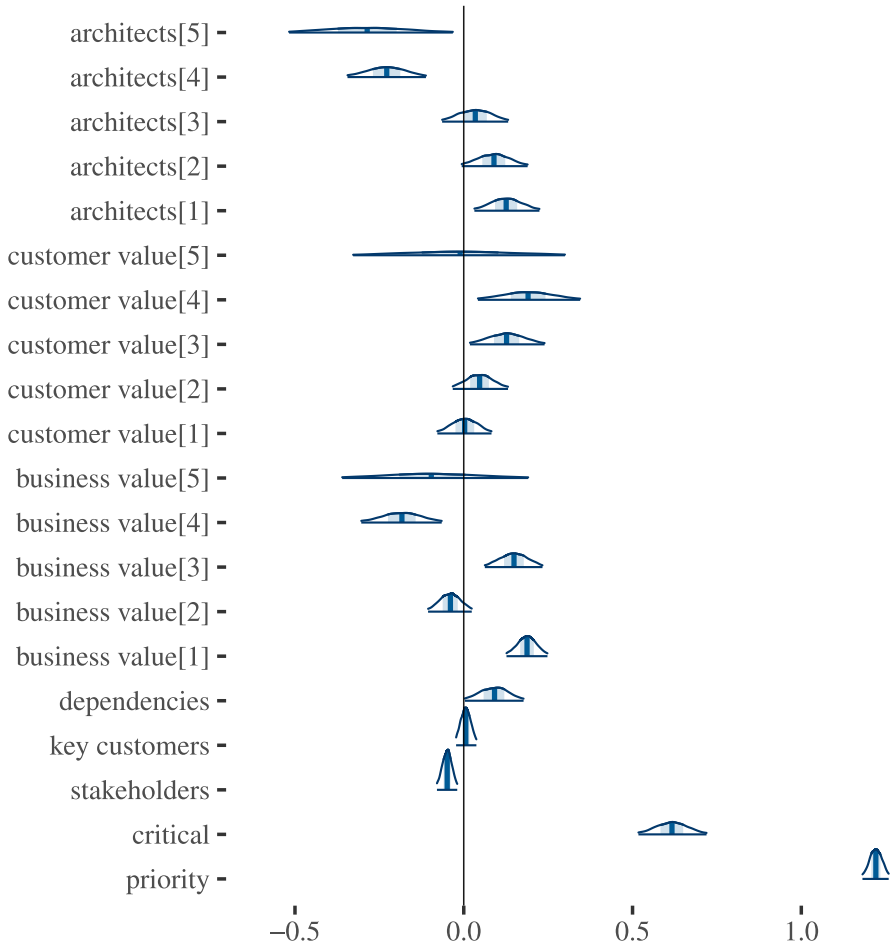

Fig. 6. Density plot of all population-level (fixed effects). Examining the above plot, from bottom to top, we can claim that the first three parameters are significant and their 95% CI do not cross zero (each density is cut off at 95%). The fourth parameter, `Key customers`, is not significant. The fifth parameter, `Dependencies`, is significant $CI_{95\%} = [0.01, 0.18]$. For our three parameters modeled as category-specific: `Business value`, `Customer value`, and `Architects' involvement`, some categories are visibly not crossing zero. The rows `bus. value[1,...,5]`, `cust. value[1,...,5]`, and `arch. involv.[1,...,5]`, are the five cutpoints which we use to estimate the deviation on each outcome (`State` $1, \ldots, 6$). Since we have six outcomes we have $6 - 1 = 5$ cutpoints (i.e., the borders between the six outcomes, `State` $1, \ldots, 6$).

4. Results

Before we explain the concept of conditional effects, we will investigate the model's results as-is.

First, Table 4 consists of all parameters we are interested in (i.e., the β's). All rows in bold indicate a significant effect, that is, the 95% credible interval of an effect's distribution, does not cover zero (l-95% and u-95% columns in the table).

What can we tell from Table 4? First, `Priority`, `Criticality`, and `Dependency` have a positive effect (the higher the more likely to end up in a higher state), while the opposite is true for `Stakeholders`. Second, the predictor `Key customer` has very little predictive power. Third, for the predictors that were modeled as category-specific the picture is not that clear (i.e., they were modeled separately for each of the categories in the outcome `State`).

Looking at the first such predictor, i.e., `Business value` (**bus. value[1,..,5]**), one can find three effects that are considered 'significant': **bus. value[1]**, **bus. value[3]**, and **bus. value[4]**. First, the higher the `Business value` in `State` 1 and 3, the likelier it is that it reaches those states, while the opposite holds for `State` 4. In the latter case, a higher `Business value` leads, probabilistically speaking, more often to a requirement that will not reach `State` 4. This indicates that the predictor affects the outcome (our six states) differently.

Fig. 6 provides a visualization of the table that is, perhaps, more straightforward to understand. In Fig. 6, each parameter's posterior probability distribution, with 95% credible intervals, is plotted. This, perhaps, allows the reader to gain more insight, compared to only looking at Table 4.

Next, even though we are not very interested in an effect's point estimate *per se*, let us take `Stakeholders`, as an example, i.e., $\mu = -0.05$ $CI_{95\%}[-0.08, -0.02]$. Recall from Section 3.4 that `Stakeholders` could vary $(0, \ldots, 10)$ and was used to indicate how many stakeholders a particular feature had. First, we transform the value using inverse logit, since the model used a logit() link function, i.e., $\exp(-0.05)/(\exp(-0.05) + 1) = 0.49$.

In order to improve sampling, all variables, where appropriate, were centered and scaled, i.e., for all values, we removed the variable's mean ($\mu_x = 1.05$), and then scaled each value by dividing it with the variable's standard deviation ($\sigma_x = 0.53$). Hence, to receive the original scale we do the opposite, i.e., $0.49 \cdot 0.53 + 1.05 = 1.31, CI_{95\%}[1.306, 1.314]$. In short, the model estimates that, on average, we have 1.31 stakeholders per feature, with a 95% credible interval of $[1.306, 1.314]$.

If we next take customer value as an example, we can claim that customer value has a positive effect on the third and fourth cutpoints, i.e., the cutpoints between `States` 3/4 and between 4/5, are pushed up, leading to more probability mass being assigned to the lower levels, i.e., `State` ≤ 4 (`Elicited, Prio, Planned, Implemented,`

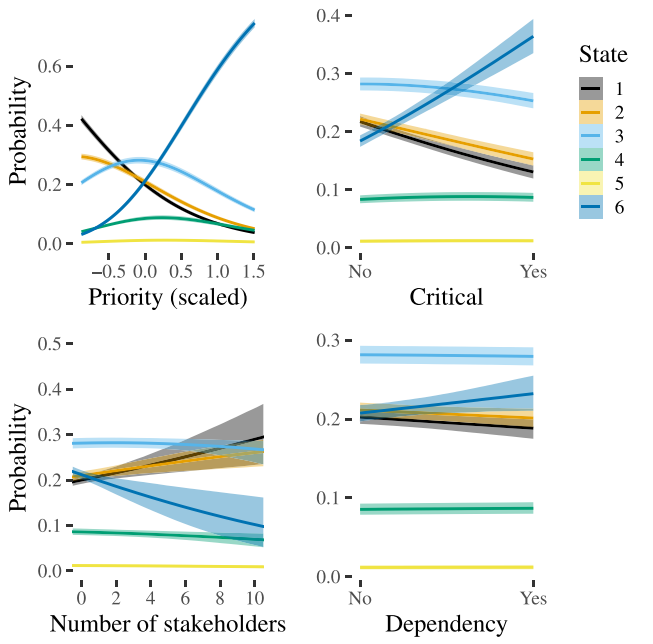

Fig. 7. Conditional effects for four of the predictors. On the *y*-axis we have the estimated probability (which can differ between the plots). The top-left plot has a scaled *x*-axis, while for the bottom left plot it is on the outcome scale. The two plots to the right have a dichotomous outcome 'No'/'Yes' on the *x*-axis. The colored bands show the 95% credible interval.

`Dropped`, and below). This detail would have been impossible to notice without modeling the effect as category specific.[6] However, what is even more interesting is the fact that we have a joint posterior probability distribution for all effects, i.e., it is possible to see how each effect varies when fixing all other covariates to their mean or reference category.

4.1. Conditional effects

Conditional effects allow us to fix all predictors to their mean, or reference category for factors, except for the one we want to understand better.[7] If we plot our significant effects for our continuous covariate, one can see how the effect varies depending on `State` (Fig. 7).

Let us now go through these plots one by one and make notes about the particular characteristics of an effect. The top-left plot in Fig. 7 presents the effect `Priority`. What is evident, compared with the other plots, is that the uncertainty is low since the bands surrounding each line are tightly following the line. If we look at `State` 6 (a feature is released), we can see that there is a much higher probability (*y*-axis) for `State` 6, as we move to the right (the priority increases). Also, as expected, for `States` 1 and 2 to have a high `Priority` is uncommon.

Examining the next plot (clockwise), one can see that for `State` 6, there is a clear change when moving from 'No' to 'Yes' (albeit with a slight increase in uncertainty). Next, in `Dependency`, which is also a dichotomous variable, there is a difference between `State` 6 and `State` 1 and 2. In short, if we have a dependency, there is a greater probability that the feature will end up in `State` 6, while the opposite holds for `State` 1 and 2. Also worth noting is that `State` 3 has the highest probability of having a dependency (and then it does not matter if it moves from 'No' to 'Yes').

Finally, for `Stakeholders`, one can see that `State` 1 and 2 implies that the more stakeholders, the higher the probability that it will end up in those states, which might sound counterintuitive; however, we also have an increase in uncertainty. The opposite holds for `State` 6, but once again, greater uncertainty when increasing the number of stakeholders. If we examine Fig. 4(b), we also receive an answer for why the uncertainty increases in this way, i.e., we have less data (evidence) when the number of stakeholders increases.

We will refrain from plotting the last three significant category-specific effects, and simply conclude by saying that all three effects contain categories that affect the outcome positively or negatively.

To summarize this section, we have seen that analyzing estimates drawn from the posterior probability distribution provides us with indications of significant effects (by looking at the standard deviations and credible intervals). The conditional effects analysis, i.e., fixing all other variables except for one, provided insight into how an effect varies (an effect's size is, by itself, not always exciting, but rather how it varies depending on context).

5. Discussion

In this section, the results are discussed and related to previously published findings. Section 5.1 discusses the first research question, while the second research question is discussed in Section 5.2. Section 5.3 discusses general findings and, finally, Section 5.4 discusses implications for practitioners.

5.1. RP criteria with actual impact (RQ1)

In analyzing RQ1, this section examines which RP criteria the company deemed most important to use in their project (i.e., which ones are actually used), and which ones have an actual impact when deciding which features should be implemented and released. However, we did not investigate if the used RP criteria are the most *appropriate* ones to use. This decision was made by the company and is not part of this study.

Looking into which RP criteria the company deemed most important to use in practice, eight RP criteria were used when prioritizing requirements, namely:

- **Team priority** - the teams subjective/expert opinion of the importance of a feature,
- **Critical Feature** - if the feature was critical or not,
- **Customer value** - how valuable the feature was considered to be for customers,
- **Business value** - the business value of the feature,
- **Stakeholders** - number of key internal stakeholders who considered a feature important,
- **Key customers** - number of key customers who considered a feature important,
- **Dependency** - if a feature has a dependency to other features, and
- **Architects' involvement** - the needed level of involvement from a software architect in order to design/implement a feature.

All eight RP criteria used by the case company have already been identified in the literature (e.g., in Berntsson Svensson et al., 2011; Riegel and Doerr, 2015; Thakurta, 2017; Hujainah et al., 2018; Daneva et al., 2013; Zhang et al., 2014; Eckstein, 2004). Business and customer value are native to agile (Eckstein, 2004), and used in industry when prioritizing requirements (Berntsson Svensson et al., 2011; Daneva et al., 2013). Although expert opinion (peoples' previous experiences, opinions, intuitions, various criteria, arguments, or a combination of

[6] In the replication package one can see that not modeling category-specific effects would miss that architects' involvement actually has some significant effects.

[7] For unordered categorical variables, which we do not use, the first category is the reference category. This can however be changed if needed. All other categories are deviations from this category (as is the case of many model designs, whether frequentist or Bayesian).

one or several of these information sources) is not identified as an RP criterion in the literature, it is often used when prioritizing requirements (Berntsson Svensson et al., 2011; Maalej et al., 2016; Holmström Olsson and Bosch, 2014). However, there is a difference between how expert opinion is used in the analyzed project and what is reported in the literature. The difference is that the team's expert opinion (called Team priority) is an explicitly specified criterion for RP where the teams decide on a value (between 0 and 1000) that represents their expert opinion, while in the literature expert opinion is not stated as an explicit RP criterion and it is not quantified.

One interesting finding is related to the importance of a requirement. Hujainah et al. (2018), indicate that importance is the most frequently used RP criterion in the identified RP techniques/tools. According to Hujainah et al., importance refers to how important a requirement is to the stakeholders. This definition is in line with (Thakurta, 2017), who defines importance as the subjective evaluation of a requirement by stakeholders. However, stakeholders include several different types of stakeholders, e.g., users, customers, the project team, marketing/business department, and competitors; thus it is not clear in Hujainah et al. (2018), Thakurta (2017) which perspective is used. On the other hand, Riegel and Doerr (2015), report on several different perspectives of importance identified as RP criteria, e.g., project importance with regards to overall project goal, importance to business goals, and importance to customers. In the analyzed project, three different perspectives of importance were used when prioritizing requirements as three separate criteria, namely: (i) from the project's perspective (Critical feature), (ii) from an internal stakeholder perspective (Stakeholders), and (iii) from a customer perspective (Key customers). The analyzed project's different perspectives of importance is in line with the view of Riegel and Doerr (2015). In the literature, importance is often used in pair-wise comparisons, to produce an ordered list of requirements based on importance, or from a cost-value perspective. However, in the analyzed project, the importance from stakeholder and customer perspective were simply used by counting how many internal stakeholders considered the feature/requirement to be important and counting how many key customers (customer perspective) consider a feature/requirement to be important.

One surprising finding, when comparing the used RP criteria in the analyzed project with the literature, is that implementation/development effort/cost was not used at all in the analyzed project, despite being frequently mentioned in the literature (e.g., in Hujainah et al., 2018; Thakurta, 2017), and being the most frequently mentioned criterion in Riegel and Doerr (2015). Moreover, there are several RP techniques/tools in the literature (e.g., in Bukhsh et al., 2020; Hujainah et al., 2018) that are based on cost/effort, and it has been reported to be used in industry when prioritizing requirements, e.g., in Berntsson Svensson et al. (2011), Daneva et al. (2013). Despite that various cost/effort estimations are performed at the case company and for the analyzed project, it is not deemed as an important criterion to be used for RP. One possible explanation may be that cost/effort estimations were considered by the team when setting their own priority (called Team priority), but not explicitly used when prioritizing requirements. However, it is not possible to confirm or reject this explanation based on the extracted data. We can only conclude, based on the extracted data, that implementation/development effort/cost was not considered an important RP criterion at the company when determining which requirements should be implemented and released, which is not in line with the literature.

The case company used eight RP criteria in the analyzed project, but just because they are used it does not mean that they have an actual impact when determining which requirements should be implemented and released. Therefore, we analyzed 32,139 decisions for 11,110 features to see which of the eight RP criteria have an actual impact. Based on the results in Section 4 (see Table 4 and Fig. 6), seven out of the eight RP criteria used in the analyzed project have an actual impact on RP. The only criterion that did not have an actual impact was Key customers.

When comparing the RP criteria that do have an actual impact on the RP decisions made, there is a difference between the seven RP criteria in how strong of an impact each criterion had, as shown in Table 4 (column Estimate) and in the replication package. Two criteria had a strong impact on the RP decisions, namely Team priority and Critical Feature. Team priority had the strongest impact on the RP decisions (with an estimate of 1.22), while Critical Feature had an estimate of 0.62. The remaining five RP criteria had a small impact with an estimate between −0.29 and 0.19. The finding that Team priority (i.e., the teams' expert opinion/experiences/subjective opinion) had the strongest impact on RP decisions is in line with the literature (Holmström Olsson and Bosch, 2014; Maalej et al., 2016) which suggest that RP decisions are commonly based on previous experiences and opinions.

Although the RP criterion Dependency was significant, meaning it had an actual impact on RP, it had a low impact on deciding which requirements that were implemented and released, which is shown in Table 4 and Fig. 6. This result is not in line with the literature (Daneva et al., 2013; Riegel and Doerr, 2015; Thakurta, 2017; Hujainah et al., 2018; Shao et al., 2017; Zhang et al., 2014). This is surprising since requirement dependencies are important when prioritizing requirements and deciding the order in which the requirements can be implemented (Zhang et al., 2014). Some requirement needs to be satisfied according to conditions of other requirements, while others may have to be implemented together (Li et al., 2012). According to Shao et al. (2017), requirement prioritization results that do not consider requirements dependency can rarely be used, which is not in line with the results from this study. In addition, requirement dependencies are used as an RP criterion in industry (Daneva et al., 2013), is frequently mentioned as an important RP criterion in the literature (Riegel and Doerr, 2015; Thakurta, 2017), and used in several RP techniques/tools (Hujainah et al., 2018). However, just because dependency is used as a RP criterion, it may not have a large impact on the RP decisions made, as shown in this study. One possible explanation for the difference between this study and the literature is that we have not asked industry practitioners what they consider (i.e., their subjective opinion) to be important when prioritizing requirements, nor have we used our own opinion or previous studies to decide which criteria have an actual impact on RP. Instead, we investigated the actual outcome of 32,139 RP decisions for one completed project at one software developing company. To the best of our knowledge, no other study has analyzed the actual outcome of RP decisions in industry to identify which RP criteria have an actual impact, and definitely not with such a large sample.

5.2. Impact of RP criteria depending on the state (RQ2)

As shown in Table 5 (the conclusions in Table 5 are based on the results in Section 4 and in the replication package), different RP criteria had different impact on RP depending on the state of the requirement, i.e., depending on how far the requirement has reached in the development process. Meaning, some criteria had a high impact on RP early in the development process, others in the middle, while some had a high impact at the end.

When moving from State 1: *Elicited* to State 2: *Prioritized* (cutpoint 1 in Table 5), five RP criteria had an impact on the RP decision. The lower team priority, the higher business value (i.e., to be considered valuable for the company), not being considered a critical feature, the more internal stakeholders that consider a feature to be important, and the more architects are involved, the higher probability that a requirement reach State 2. For cutpoint 2 (when a requirement is moving from State 2: *Prioritized* to State 3: *Planned*), the RP criteria stakeholders and architect's involvement have the same impact as in cutpoint 1. In addition, a medium Team priority (i.e., not too low and not too

Table 5
Requirements prioritization criterion's impact depending on a requirements state- A '-' means no impact.

Cutpoint	Team priority	Bus. value	Cust. value	Critical	Depend.	Stakeholders	Key cust.	Arch. involv.
1	Low	High	–	No	–	More	–	More
2	Medium	–	–	No	–	–	–	More
3	–	High	High	–	–	–	–	–
4	–	Low	High	–	–	–	–	Less
5	High	–	–	Yes	–	Less	–	Less

high) had an impact on the RP in cutpoint 2. When a requirement moves from State 3: *Planned* to State 4: *Implemented* (cutpoint 3 in Table 5), high customer and business value had an impact on the requirement prioritization, i.e., the higher customer and business value a requirement have, the more likely it is that it will reach State 4. High customer value, low business value, and less involvement from the architects are important for a requirement to move from State 4: *Implemented* to State 5: *Tested* (cutpoint 4 in Table 5). Finally, in cutpoint 5 (moving from State 5: *Tested* to State 6: *Released*), being a critical feature and considered important for the team (i.e., having high Team priority), and having less internal stakeholders interested in the requirement and less involvement from the architects, increases the probability of the requirement to be released.

Looking into the RP criterion Stakeholders, it is only in cutpoint 1 where more internal stakeholders that consider a feature to be important means that a feature is more likely to reach the next state. For all other cutpoints (i.e., RP decisions), Stakeholders had either no impact on the decisions or it is a lower probability for a feature to be included (i.e., prioritized) with an increasing number of internal stakeholders who consider the feature to be important, which is not in line with the literature (Riegel and Doerr, 2015; Thakurta, 2017; Hujainah et al., 2018).

That the software architects need to be more involved in the beginning of the project (cutpoints 1 and 2) makes sense since it is important to analyze if the included requirements have any negative impact on the current architecture, and/or if the technical debt would increase. However, just because a requirement may have a negative impact on the current architecture and/or the technical debt, it does not mean it will not be included in the product. It means that it is important to get this information/knowledge from the experts (i.e., the software architects) before making decisions about the requirement.

One interesting, and surprising finding is that only internal value (Business value and Stakeholders) and not external value (Customer value and Key customers) had an actual impact on deciding which features reach State 3: *Planned* (up until cutpoint 2). That is, among all features that were prioritized to be included in the product until State 3, only internal value was considered while the customer perspective was ignored. The criterion Customer value only starts having an actual impact when a feature moves from State 3 to State 4 (cutpoint 3), and from State 4 to State 5 (cutpoint 4), while Key customers did not have an impact when prioritizing features. This means that features in the early phases in the development process with high customer value may not be prioritized to be included if the business value is low. This is not in line with Daneva et al. (2013) where the focus is on combining value-creation for the vendor (i.e., Business value) with value-creation for the customer (i.e., Customer value).

The findings in this study show that the team's expert opinion/experiences/subjective opinion etc. only had a positive impact on RP at the very end of the development process (in cutpoint 5). This is not in line with the literature (Holmström Olsson and Bosch, 2014; Maalej et al., 2016), which suggests that the decisions and selection of what to include are commonly based on previous experiences, opinions, intuitions, arguments, or a combination of one or several of these information sources. Instead, up until cutpoint 5, the decisions (i.e., RP) were based on the internal stakeholders view of the importance of the feature, business value and finally customer value. However, in cutpoint 5, Team priority had a very large effect (probability mass close to 70%) on which features should be released.

When discussing RQ1, which RP criteria have an actual impact on RP, we saw that Dependency had a significant impact on RP; however, it was weak due to high uncertainty. When analyzing RQ2, if the impact change depending on which state a requirement is in, we see, in particular in cutpoint 2 (i.e., to reach State 3), that Dependency had an impact with a probability mass of close to 30%. However, not much changes when the dependency moves from 'No' to 'Yes', as shown in Fig. 7. We see a similar pattern, although with a lower portability mass, for all other states. One possible explanation may be that all features, regardless if any dependencies to other features have been identified when the RP decisions are made, are treated in the RP decision process as if they have dependencies to other features. Meaning, the practitioners did not consider if the value for the Dependency criterion is 'Yes' or 'No', it was viewed as if there are dependencies, and if the dependencies are discovered later in the software development process they will be able to handle it without any delays. This is supported by Martakis and Daneva (2013) who found that practitioners in agile software development projects indicated that they were able to deal with dependencies without too much effect for the project, and whether the dependencies were discovered early or late did not have an effect or impact on the project.

5.3. General discussion of results

The results from RQ2 (see Section 5.2) show that not all RP criteria have an equal impact on which requirements are prioritized to be included, implemented, and eventually released, and that the impact of a criterion changes depending on where in the software development process a requirement is (refers to the six different states, as described in Table 2). For example, Business value had an impact on RP in the early phase, Customer value in the later phases, while being a critical requirement (i.e., Critical feature is 'YES') only had an impact in the last phase (State 6). These findings are not in line with how RP techniques/tools in the literature are developed (Hujainah et al., 2018; Thakurta, 2017). Most, if not all, RP techniques/tools select, based on expert opinion, which criteria should be used in the developed RP technique/tool (Riegel and Doerr, 2015). Hence, the RP technique/tool in the literature cannot be used in a flexible way with different criteria depending on the development phase, and thus may not be so useful in practice. This may be one reason why gut-feeling, subjective opinion, and expert judgement are frequently reported to be used in RP (Berntsson Svensson et al., 2011; Maalej et al., 2016; Holmström Olsson and Bosch, 2014) and it may be explained by the representativeness heuristic, which is a mental shortcut to lessen the cognitive load (Gren et al., 2017). Meaning, when the needed information (i.e., RP criteria with an actual impact) is not available/cannot be used in the current tools/techniques when making decisions, practitioners use similar or previous experience (e.g., their gut-feeling or subjective opinion) instead. The importance of having flexible RP techniques/tools is supported by Berander and Andrews (2005).

The findings in this study show the importance for customizing RP criteria, not only to a specific context/project (Riegel and Doerr, 2015), but also to specific development phases. Thus, when developing RP techniques/tools, or other decision support systems (e.g., AI-based or data-driven decision support systems), it is important to identify which criteria are important to use (i.e., which ones have an actual impact on RP decisions), and when to use them. Not identifying which criteria

that have an impact may lead to other consequences. One consequence may be that unnecessary criteria (information that is not important for the decision) are presented to the decision makers. That is, unnecessary decision criteria are visible for the decision makers, which may lead to poor or wrong decisions. Having an extra irrelevant option set, e.g., RP criteria that have no impact on the decision, visible to the decision maker should not affect the choice, but in some contexts it does (Huber et al., 1982). The importance of presenting correct information to the decision maker is shown in (Gren et al., 2017) where the presence of obsolete requirements negatively affected the cost/effort estimations of the requirements. Thus, it may have a similar affect on unnecessary RP criteria.

5.4. Implications for practitioners

In this section, we discuss the findings most important to industry practitioners. There are two aspects that could be of interest to industry practitioners in RP contexts: (1) not all RP criteria have an equal impact on RP decisions made, and that the impact of a criterion changes depending on where in the software development process a requirement is, and (2) to fully understand what RP criteria have an actual impact on RP, a detailed statistical analysis of used RP criteria in previous projects is needed.

First, we found strong evidence that not all RP criteria are equal (in terms of impact on RP decisions), and that the impact of an RP criterion changes depending on how far a requirement has reached in the software development process, as presented in Sections 5.1 and 5.2. Based on the results from our statistical analysis, we identified the need to oversee how RP criteria are decided/selected (i.e., which one to use) for RP decisions in projects in industry. Since our results show that different RP criteria have different impact depending on when the RP criterion is used (i.e., where in the software development process), it is recommended to consider development phase-specific RP criteria instead of sticking to the same criteria from the beginning to the end. Hence, our recommendation to industry practitioners is to at least consider that a RP criterion may not have an actual impact on RP decisions throughout the entire software development process, and thus select RP criteria to be used, e.g., in the beginning of the project, in the middle, and at the end of the project.

Which RP criteria that are better to use in the beginning, middle, and at the end of the project is not possible for us to recommend based on the analysis in this paper. There are several reasons for this. We have not analyzed or collected the needed data to see if the eight RP criteria used in the analyzed project are the most appropriate ones to use. Different companies and projects may, or perhaps should use different RP criteria than the ones used in the analyzed project. The selection of which RP criteria to use should depend on the specific project and development phase. For example, our results show that only internal value (business value and stakeholders) and not external value (customer value and key customers) have an actual impact on deciding which requirements reach state 3. However, this may not be what is important for a project or a company. Perhaps external value should first have an impact on the RP decisions (e.g., in the beginning) and then internal value should have an impact (e.g., in the middle of the software development process). If this is the case, the project and the company should not include internal value as RP criteria in the beginning of the project. Therefore, we do not recommend specific RP criteria for specific development phases. Instead, we recommend the practitioners to consider that not all RP criteria are equal, and that there may be a need to oversee how they select RP criteria where it is recommended to consider development phase-specific RP criteria instead of sticking to the same criteria from the beginning to the end.

Second, our work highlights the importance of conducting a detailed statistical analysis of RP criteria used in previous projects to identify which RP criteria have an actual impact on RP decisions made. Not identifying which RP criteria that have an actual impact on RP may lead to negative consequences. For example, if an RP criterion with no actual impact is presented to the decisions makers, it may affect the decisions in a negative way (e.g., poor or wrong decisions are made); and time is wasted on collecting/recording values to RP criteria with no actual impact on the decisions. Thus, by conducting a statistical analysis of historical RP criteria's actual impact on RP decisions, a knowledge base can be built to identify which ones have a strong impact on RP decisions and which ones are less important. This knowledge base can then be used for future projects within a company when deciding which RP criteria should be used and where in the software development process they have the strongest impact.

6. Validity threats

Construct validity (Ralph and Tempero, 2018) is concerned with the relation between theories behind the research and the observations. A construct in this case is a latent concept we are trying to measure. Ultimately we want to measure if the concept is real and if, the way we measure it indirectly, is appropriate to better understand the concept.

Concerning this study, the variables (e.g., RP criteria) used in the statistical analysis are all constructs and, hence, try to measure an underlying latent concept; this served our purposes well. By investigating literature and then contrast this with our statistical analysis we uncovered several cases where one could question if appropriate constructs are used in RP. First, the effect between constructs vary, which might not be a problem by itself; however, the fact that they vary over time is a bit more worrying. This could be a sign of inappropriate constructs (Section 5.2) and we have in this study taken a first systematic step to analyze these constructs using a principled approach to statistical analysis (see, e.g., early work by Furia et al. (2021)).

In summary, for our constructs, there might be face validity (does it makes sense?); however, content validity (do the constructs include all dimensions?) is most likely lacking for some constructs. This indicates that predictive validity can be questioned.

Internal validity concerns whether causal conclusions of a study are warranted or if overlooked phenomena are involved in the causation. We assume that the eight used RP criteria in the analyzed project are considered to be the most important ones to use when prioritizing requirements. However, other RP criteria than the ones in the database may have been used and, thus, affected the results. Moreover, another factor that may have affected the results is incorrect/missing data/value for the different RP criteria. There were no NAs in the dataset; however, that does not necessarily mean that there are no NAs. Some of the coding can be a representation of NA, e.g., 'No Value'. In this case, we know that 'No value' and 'None' in the dataset actually are values and not a representation of NAs since we asked the "gate-keeper" from the analyzed project about the correctness of the data.

External validity is concerned with the ability to generalize the results, i.e., in this case the applicability of the findings beyond the studied project and company. Analytical generalization enables drawing conclusions and, under certain conditions, relating them to other cases. This means that the context of the study needs to be compared to the context of interest for the findings to be generalized to. Therefore, we describe the case company and the studied project in as much details as possible considering confidentiality (see Section 3.2). However, the results of which RP criteria have an actual impact on RP decisions, and if their impact changes depending on how far a requirement has reached in the software development process, is specific for the studied project and the case company.

Even though we analyzed 11,110 features and 32,139 RP decision based on 8 RP criteria, we only analyzed one completed project. Thus, it is possible that the results would have been different if we studied other projects, RP decisions, and RP criteria. Our study was not designed to develop theories that applies to all projects, RP criteria and decisions, but rather to identify trends that could be a first step towards new knowledge and theories. However, it is not possible to

generalize the results from this study; although from a transferability perspective, the results may provide an overview that not all RP criteria have an equal impact on RP decisions, and that the impact changes depending on where in the software development process RP decisions are made. Meaning, for projects with a similar context, i.e., projects with thousands of requirements to be prioritized using several RP criteria in many different prioritization points by several practitioners/development teams, we would expect to see similar results in terms of: (1) that not all used RP criteria have an actual impact on the decisions throughout the entire software development process, (2) that some RP criteria have a higher impact than others in the beginning while others have higher impact at the end, and (3) that some RP criteria may not have an impact in one or several phases in the development process. However, we do expect to see changes in which specific RP criteria that have an impact on the decisions, both overall and for the specific development phases (beginning, middle, and end). One reason for this is that not all projects in all contexts will use exactly the same eight RP criteria as the analyzed project in this study. Even if some of the eight RP criteria would be used in other projects, they may be used differently.

7. Conclusion

In conclusion, we conducted a quantitative study where quantitative data was collected through a case study to analyze which RP criteria have an actual impact on RP decisions, and if the impact of a criterion changes depending on how far a requirement has reached in the development process. To this aim, we extracted 32,139 RP decisions based on eight RP criteria for 11,110 requirements (features) from one completed project at a case company. The extracted data was analyzed by designing, comparing, validating, and diagnosing ordinal Bayesian regression models. We showed how to model ordinal data in a principled way, how to use category-specific effects to get a more nuanced view, and how to report results using conditional effects. The results from this study highlights the following key findings:

1. Not all used RP criteria have an actual impact on RP decisions, e.g., Key customers had a very slight positive effect, which was not significant according to the 95% credible interval.
2. Not all RP criteria have an equal impact, and this changes depending on how far a requirement has reached in the development process. For example, for RP decisions before iteration/sprint planning, having high Business value had an impact on RP decisions, but after iteration/sprint planning having high Business value had no impact. Moreover, high Team priority (i.e., the teams' subjective opinion) and being a critical feature (i.e., Critical feature is 'YES') only had an impact at the very end of the development process.
3. Internal value (Business value and Stakeholders) is more important (i.e., have an actual impact on decisions) than external value (Customer value and Key customers) when prioritizing requirements in the beginning of the project. That is, among all requirements that are prioritized to be included in the project until the iteration/sprint planning meeting, only internal value is considered, while the customer perspective is ignored.
4. Although Dependency was found to have a significant impact on RP decisions, in particular in the middle of the development process, not much changes, in terms of actual impact in decisions, when the dependency moves from 'NO' to 'YES'. Meaning, if a requirement has dependencies to other requirements has no impact on requirement prioritization decisions.

The findings in this paper confirm the need for analyzing and identifying which RP criteria are important in order to develop flexible RP techniques/tools (Berander and Andrews, 2005). That is, the importance for customizing requirement prioritization criteria, not only for specific contexts/projects, but also for specific development phases.

Finally, the findings in this study highlights the need for conducting more quantitative studies (preferable in combination with qualitative data) on different projects and contexts, and with different RP criteria in order to get a more complete understanding of which RP criteria have an actual impact in RP decisions, and when in the development process they should be used. Although we only studied RP decisions and criteria, the results that different criteria (data/information) have different impact on the decisions depending on where in the development process the decisions are made, may be applicable to other types of decisions within software development. Therefore, it would be interesting to study other types of decisions using other support systems, e.g., AI-based, machine learning, or data-driven decision support systems, to identify which criteria/data/information have an actual impact on the decisions, and when in the development process.

CRediT authorship contribution statement

Richard Berntsson Svensson: Conceptualization, Methodology, Validation, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Richard Torkar:** Methodology, Validation, Formal analysis, Data curation, Writing – original draft, Writing – review & editing, Visualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Link to data/code is included in the submitted paper.

Acknowledgments

The computations were enabled by resources provided by the Swedish National Infrastructure for Computing (SNIC) at Chalmers Centre for Computational Science and Engineering (C3SE), partially funded by the Swedish Research Council through grant agreement no. 2018–05973.

References

Achimugu, P., Selamat, A., Ibrahim, R., 2016. ReproTizer: A fully implemented software requirements prioritization tool. In: Nguyen, N.T., Kowalczyk, R. (Eds.), Transactions on Computational Collective Intelligence XXII. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 80–105.

Achimugu, P., Selamat, A., Ibrahim, R., Mahrin, M., 2014. A systematic literature review of software requirements prioritization research. Inf. Softw. Technol. 56 (6), 568–585.

Baird, L., Holland, P., Deacon, S., 1999. Learning from action: Imbedding more learning into the performance fast enough to make a difference. Organ. Dyn. 27 (4), 19–32.

Baltes, S., Ralph, P., 2020. Sampling in software engineering research: A critical review and guidelines. arXiv e-prints, arXiv:2002.07764, arXiv:2002.07764.

Berander, P., Andrews, A., 2005. Requirements prioritization. In: Aurum, A., Wohlin, C. (Eds.), Engineering and Managing Software Requirements. Springer, Berlin, Heidelberg, pp. 69–94.

Berntsson Svensson, R., Feldt, R., Torkar, R., 2019. The unfulfilled potential of data-driven decision making in agile software development. In: Agile Processes in Software Engineering and Extreme Programming. pp. 69–85.

Berntsson Svensson, R., Gorschek, T., Regnell, B., Torkar, R., Shahrokni, A., Feldt, R., Aurum, A., 2011. Prioritization of quality requirements: State of practice in eleven companies. In: 2011 IEEE 19th International Requirements Engineering Conference. pp. 69–78.

Betancourt, M., 2019. The convergence of Markov chain Monte Carlo methods: From the Metropolis method to Hamiltonian Monte Carlo. Ann. Phys. 531 (3), 1–6. http://dx.doi.org/10.1002/andp.201700214.

Brooks, S., Gelman, A., Jones, G., Meng, X.L., 2011. Handbook of Markov Chain Monte Carlo. CRC Press.

Bukhsh, F.A., Bukhsh, Z.A., Daneva, M., 2020. A systematic literature review on requirement prioritization techniques and their empirical evaluation. Comput. Stand. Interfaces 69.

Bürkner, P.C., Charpentier, E., 2020. Modelling monotonic effects of ordinal predictors in Bayesian regression models. Br. J. Math. Stat. Psychol. http://dx.doi.org/10.1111/bmsp.12195.

Bürkner, P.C., Vuorre, M., 2019. Ordinal regression models in psychology: A tutorial. Adv. Methods Pract. Psychol. Sci. 2 (1), 77–101. http://dx.doi.org/10.1177/2515245918823199.

Daneva, M., van der Veen, E., Amrit, C., Ghaisas, S., Sikkel, K., Kumar, R., Ajmeri, N., Ramteerthkar, U., Wieringa, R., 2013. Agile requirements prioritization in large-scale outsourced system projects: An empirical study. J. Syst. Softw. 86 (5), 1333–1353.

Eckstein, J., 2004. Agile Software Development in the Large: Diving Into the Deep. Dorset House Publishing Co., Inc., USA.

Furia, C.A., Feldt, R., Torkar, R., 2021. Bayesian data analysis in empirical software engineering research. IEEE Trans. Softw. Eng. 47 (9), 1786–1810. http://dx.doi.org/10.1109/TSE.2019.2935974.

Furia, C.A., Torkar, R., Feldt, R., 2021. Applying Bayesian analysis guidelines to empirical software engineering data. Trans. Software Eng. Methodol. http://dx.doi.org/10.1145/3490953, Accepted for publication.

Gabry, J., Simpson, D., Vehtari, A., Betancourt, M., Gelman, A., 2019. Visualization in Bayesian workflow. J. R. Stat. Soc.: Ser. A (Stat. Soc.) 182 (2), 389–402. http://dx.doi.org/10.1111/rssa.12378.

Gelman, A., Simpson, D., Betancourt, M., 2017. The prior can often only be understood in the context of the likelihood. Entropy 19 (10), http://dx.doi.org/10.3390/e19100555.

Glass, R.L., 2002. Project retrospectives, and why they never happen. IEEE Software 19 (5), 112–113.

Gomes de Oliveira Neto, F., Torkar, R., Feldt, R., Gren, L., Furia, C.A., Huang, Z., 2019. Evolution of statistical analysis in empirical software engineering research: Current state and steps forward. J. Syst. Softw. 156, 246–267. http://dx.doi.org/10.1016/j.jss.2019.07.002.

Gren, L., Svensson, R.B., Unterkalmsteiner, M., 2017. Is it possible to disregard obsolete requirements? An initial experiment on a potentially new bias in software effort estimation. In: Proceedings of the 10th International Workshop on Cooperative and Human Aspects of Software Engineering. IEEE Press, pp. 56–61.

Herrmann, A., Daneva, M., 2008. Requirements prioritization based on benefit and cost prediction: An agenda for future research. In: 16th IEEE International Requirements Engineering Conference. pp. 125–134.

Holmström Olsson, H., Bosch, J., 2014. From opinions to data-driven software R&D: A multi-case study on how to close the 'open loop' problem. In: 40th Euromicro Conference on Software Engineering and Advanced Applications. pp. 9–16.

Homan, M.D., Gelman, A., 2014. The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. J. Mach. Learn. Res. 15 (1), 1593–1623.

Huber, J., Payne, J.W., Puto, C., 1982. Adding asymmetrically dominated alternatives: Violations of regularity and the similarity hypothesis. J. Consum. Res. 90–98.

Hujainah, F., Bakar, R.B.A., Abdulgabber, M.A., Zamli, K.Z., 2018. Software requirements prioritisation: A systematic literature review on significance, stakeholders, techniques and challenges. IEEE Access 6, 71497–71523.

Ioannidis, J.P.A., 2005. Why most published research findings are false. PLOS Med. 2 (8), http://dx.doi.org/10.1371/journal.pmed.0020124.

Jarzebowicz, A., Sitko, N., 2020. Agile requirements prioritization in practice: Results of an industrial survey. Procedia Comput. Sci. 176, 3446–3455, Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 24th International Conference KES2020.

Kaur, G., Bawa, S., 2013. A survey of requirement prioritization methods. Int. J. Eng. Res. Technol. 2 (5), 958–962.

Kullback, S., Leibler, R.A., 1951. On information and sufficiency. Ann. Math. Stat. 22 (1), 79–86. http://dx.doi.org/10.1214/aoms/1177729694.

Li, J., Zhu, L., Jeffery, R., Yan Liu, He Zhang, Qing Wang, Mingshu Li, 2012. An initial evaluation of requirements dependency types in change propagation analysis. In: 16th International Conference on Evaluation Assessment in Software Engineering. EASE 2012, pp. 62–71.

Liddell, T.M., Kruschke, J.K., 2018. Analyzing ordinal data with metric models: What could possibly go wrong? J. Exp. Soc. Psychol. 79, 328–348. http://dx.doi.org/10.1016/j.jesp.2018.08.009.

Maalej, W., Nayebi, M., Johann, T., Ruhe, G., 2016. Toward data-driven requirements engineering. IEEE Softw. 33 (1), 48–54.

Maalej, W., Nayebi, M., Johann, T., Ruhe, G., 2016. Toward data-driven requirements engineering. IEEE Softw. 33 (1), 48–54.

Magnusson, M., Vehtari, A., Jonasson, J., Andersen, M., 2020. Leave-one-out cross-validation for Bayesian model comparison in large data. In: Chiappa, S., Calandra, R. (Eds.), Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics. In: Proceedings of Machine Learning Research, vol. 108, PMLR, pp. 341–351.

Martakis, A., Daneva, M., 2013. Handling requirements dependencies in agile projects: A focus group with agile software development practitioners. In: IEEE 7th International Conference on Research Challenges in Information Science. http://dx.doi.org/10.1109/RCIS.2013.6577679.

Morey, R.D., Hoekstra, R., Rouder, J.N., Lee, M.D., Wagenmakers, E.J., 2016. The fallacy of placing confidence in confidence intervals. Psychon. Bull. Rev. 23 (1), 103–123. http://dx.doi.org/10.3758/s13423-015-0947-8.

Nuzzo, R., 2014. Scientific method: Statistical errors. Nature 506 (7487), 150–152. http://dx.doi.org/10.1038/506150a.

Pergher, M., Rossi, B., 2013. Requirements prioritization in software engineering: A systematic mapping study. In: 2013 3rd International Workshop on Empirical Requirements Engineering. EmpiRE, pp. 40–44.

Ralph, P., Tempero, E., 2018. Construct validity in software engineering research and software metrics. In: Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018. EASE '18, Association for Computing Machinery, New York, NY, USA, pp. 13–23. http://dx.doi.org/10.1145/3210459.3210461.

Riegel, N., Doerr, J., 2015. A systematic literature review of requirements prioritization criteria. In: Fricker, S.A., Schneider, K. (Eds.), Requirements Engineering: Foundation for Software Quality. Springer International Publishing, pp. 300–317.

Riņkevičs, K., Torkar, R., 2013. Equality in cumulative voting: A systematic review with an improvement proposal. Inf. Softw. Technol. 55 (2), 267–287. http://dx.doi.org/10.1016/j.infsof.2012.08.004.

Shao, F., Peng, R., Lai, H., Wang, B., 2017. DRank: A semi-automated requirements prioritization method based on preferences and dependencies. J. Syst. Softw. 126, 141–156.

Talts, S., Betancourt, M., Simpson, D., Vehtari, A., Gelman, A., 2018. Validating Bayesian inference algorithms with simulation-based calibration. arXiv e-prints, arXiv:1804.06788, arXiv:1804.06788.

Thakurta, R., 2017. Understanding requirement prioritization artifacts: A systematic mapping study. Requir. Eng. 22 (4), 491–526.

Torkar, R., Furia, C.A., Feldt, R., Gomes de Oliveira Neto, F., Gren, L., Lenberg, P., Ernst, N.A., 2021. A method to assess and argue for practical significance in software engineering. IEEE Trans. Softw. Eng. 1. http://dx.doi.org/10.1109/TSE.2020.3048991.

Tutz, G., 1990. Sequential item response models with an ordered response. Br. J. Math. Stat. Psychol. 43 (1), 39–55. http://dx.doi.org/10.1111/j.2044-8317.1990.tb00925.x.

Vehtari, A., Gelman, A., Gabry, J., 2017. Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. Stat. Comput. 27, 1413–1432. http://dx.doi.org/10.1007/s11222-016-9696-4.

Vehtari, A., Gelman, A., Simpson, D., Carpenter, B., Bürkner, P.C., 2021. Rank-normalization, folding, and localization: An improved $\hat{R}$ for assessing convergence of MCMC. Bayesian Anal. 1–28. http://dx.doi.org/10.1214/20-BA1221.

von Zedtwitz, M., 2002. Organizational learning through post-project reviews in R&D. R D Manag. 21 (3), 255–268.

Walker, S.H., Duncan, D.B., 1967. Estimation of the probability of an event as a function of several independent variables. Biometrika 54 (1–2), 167–179. http://dx.doi.org/10.1093/biomet/54.1-2.167.

Wohlin, C., Höst, M., Henningsson, K., 2003. Empirical research methods in software engineering. In: Conradi, R., Wang, A.I. (Eds.), Empirical Methods and Studies in Software Engineering: Experiences from ESERNET. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 7–23.

Woolston, C., 2015. Psychology journal bans P values. Nature 519 (7541), 9. http://dx.doi.org/10.1038/519009f.

Zhang, H., Li, J., Zhu, L., Jeffery, R., Liu, Y., Wang, Q., Li, M., 2014. Investigating dependencies in software requirements for change propagation analysis. Inf. Softw. Technol. 56 (1), 40–53.

Richard Berntsson Svensson is an Associate Professor in Software Engineering at Chalmers and university of Gothenburg, Sweden. His research interests include data-driven decision making, agile and lean software development, requirements engineering, creativity and innovation, and human aspects of software engineering. He received his Ph.D. from Lund University, Sweden, 2011.

Richard Torkar is a professor of software engineering and Head of Computer Science and Engineering department at Chalmers and University of Gothenburg. His main interest lay in the area of quantitative analysis, in particular the application of Bayesian data analysis and computational statistics. For more information, please visit: https://www.torkar.se.