Highlights

A Practitioner Survey on Requirements Technical Debt Quantification

Judith Perera, Ewan Tempero, Yu-Cheng Tu, Kelly Blincoe

- RTD is fixed mostly during the implementation of software features
- Benefit of fixing RTD was the most agreed-upon and quantified concept
- Differences in practitioner and company perspectives
- The choice of concepts to quantify varied for the top three RTD instances
- Existing tools may not well support the quantification needs in the industry

A Practitioner Survey on Requirements Technical Debt Quantification

Judith Perera[a,*], Ewan Tempero[a], Yu-Cheng Tu[a], Kelly Blincoe[b]

[a]*School of Computer Science, The University of Auckland, Auckland, 1010, New Zealand*
[b]*Department of Electrical, Computer and Software Engineering, The University of Auckland, Auckland, 1010, New Zealand*

Abstract

Requirements Technical Debt (RTD) is a phenomenon borrowed from the Technical Debt literature that captures the consequences of sub-optimal decisions made concerning software requirements. We report on a survey conducted to understand industry practices and perceptions about RTD quantification.

The survey instrument was designed based on prior work, the RTD Quantification Model (RTDQM), which captures RTD quantification conceptually and serves as a reference point. Our survey employs the Critical Incident Technique (CIT), inquiring practitioners about what RTD instances they encountered. Follow-up questions focused on understanding whether practitioners fixed such RTD instances and whether they quantified model concepts such as the Cost of fixing, the Benefit of fixing, and the Consequences of not fixing. We also sought their opinions on whether quantification supports decision-making.

Our findings suggest that the Benefit of fixing RTD is the concept agreed by most practitioners that it supports decision-making, and is quantified in practice. Practitioners' preferences regarding the concepts to quantify seem to differ based on the different RTD instances. Survey findings also suggest that the company and individual perspectives regarding the quantification of the concepts differ. Our findings reveal future research avenues that warrant deeper conversations with the industry.

Keywords: , Technical Debt, Requirements Technical Debt, Quantification, Survey, Empirical Studies

1. Introduction

Requirements Technical Debt (RTD) is a phenomenon that is gaining the attention of both Requirements Engineering (RE) and Technical Debt (TD) research communities. The phenomenon captures the consequences of sub-optimal decisions made concerning software requirements either with intention (e.g., as a strategic decision) or unintentionally (Ernst, 2012; Abad and Ruhe, 2015; Lenarduzzi and Fucci, 2019; Perera et al., 2024a). It is similar to the notion of "debt" captured by the Technical Debt (TD) metaphor (Cunningham, 1992; Avgeriou et al., 2016), but the focus is on "Requirements".

As with TD, RTD can also have adverse consequences unless managed. Besker et al. (2017) surveyed software practitioners to investigate how practitioners perceive and estimate the impact of the negative consequences of TD (TD Interest). Their findings showed that, on average, 36% of all development time is estimated to be wasted due to TD and that RTD is one of the types of TD that generates the most negative effects. Therefore, RTD must be managed without letting it lead to negative consequences in the long run.

We hypothesize that quantifying or measuring RTD (i.e., quantifying or measuring costs, benefits, and consequences associated with sub-optimal decisions made concerning requirements), one of the TD management activities, can support informed decision-making for managing RTD. In this paper, we report on the methodology and results of a survey we conducted with the aim of understanding industry perceptions and practices of quantifying RTD. In particular, we wanted to understand the current practices of quantifying RTD and practitioners' perceptions of how quantifying RTD could support informed decision-making for eliminating or repaying RTD.

[*]Corresponding Author
Email address: `jper120@aucklanduni.ac.nz` (Judith Perera)

This study is part of a larger effort to conceptualize and operationalize the quantification and management of RTD. While our previous work (Perera et al., 2023a, 2024a) conceptualized RTD quantification, this study serves as a bridge to operationalizing the theory. The survey answered the following research questions.

- **RQ1: What are the current industry practices of quantifying RTD?**
- **RQ2: How could RTD quantification support informed decision-making for managing RTD?**

We employed the Critical Incident Technique (CIT) (Flanagan, 1954) in our survey, asking the respondents to describe a recent incident that created a significant impact on their project (or company) due to problems with requirements, that is, RTD instances (i.e., RTD items). Follow-up survey questions focused on the scenario of fixing (or eliminating or repaying) the RTD instances. Survey questions about RTD quantification were designed informed by the Requirements Technical Debt Quantification Model (RTDQM), which was developed in our previous work (discussed in Section 2.2). We asked practitioners whether they fixed the RTD instances, how often (or when) they fixed them and whether they quantified RTD quantification model concepts such as the Cost of fixing, the Benefit of fixing, and the Consequences of not fixing, to make the decision to fix (Perera et al., 2024a). We also sought practitioners' agreement about whether quantifying those concepts supports decision-making and later analyzed the differences in agreement and the quantification done in practice. We asked practitioners what existing tools or techniques supported quantification of RTD and what other factors may need to be considered for quantification. We analyzed the differences in the concepts quantified for the different RTD instances described by the practitioners. We also analyzed the differences in quantification practices based on practitioner and company demographics.

We found that most practitioners fixed RTD during the implementation of software features indicating that the RTD instances are attended to only later on in the software development process. Practitioners perceived model concepts to be useful in the decision-making to fix RTD when they were quantified. However, the most prominent concept agreed on and quantified in practice was the Benefit of fixing RTD instances. Another finding was that the choice of concepts to quantify for decision-making was not the same for different RTD instances fixed during a critical incident. Practitioners mentioned various other factors as important to quantify for decision-making apart from the Cost of fixing, the Benefit of fixing and the Consequences of not fixing. However, some factors could still be related to the same concepts, while some factors could be related to concepts not used in the design of this survey but captured in our RTD Quantification Model (RTDQM) (Perera et al., 2024a). We could identify a practitioner and a company profile based on their tendency to quantify RTD which indicate that there were differences in the individual and company perspectives. Although both proprietery and internal tools were used in the industry, it is unclear whether they support the quantification needs well enough.

Our work presents, to the best of our knowledge, the first study surveying software practitioners regarding the quantification of RTD. The main contributions of this paper are:

- A survey instrument designed for gathering industry practices and perceptions of RTD quantification
- Implications and future research directions
- A Replication Package[1] that allows researchers to replicate or extend this work

The rest of this paper is structured as follows. We discuss background and related work in Section 2. Section 3 reports on the Methodology employed to conduct this study. Section 4 describes the RTD instances gathered from the practitioners. Results obtained for our Research Questions are reported in section 5. Sections 6 and 7 discuss further analyses of our findings. Section 8 revisits the research questions and discusses implications to researchers and practitioners. Section 9 discusses threats to validity. The paper is concluded in Section 10 while mentioning the next steps of this work.

2. Background and Related Work

Below, we discuss the background needed to understand this work and the most closely related work found in the literature.

[1]Replication Package: `https://doi.org/10.5281/zenodo.14172401`

2.1. Background

2.1.1. Problems with Requirements

The Naming the Pain in Requirements Engineering (NaPiRE) initiative surveyed software practitioners worldwide to understand RE practices and problems (Fernández et al., 2017; Fernández and Wagner, 2013; Wagner et al., 2017, 2019; Kalinowski et al., 2017). Moving targets, communication flaws between the project team and the customer, communication flaws within the project team, incomplete or hidden requirements, and under-specified requirements, were among the top five requirements problems that caused project failures as identified in their work (Fernández et al., 2017). The authors analyzed the causes and effects of the identified top five problems. 'Incomplete or hidden requirements' was the most consistent problem, which was primarily caused by lack of experience of RE team members, time pressure, stakeholders lacking business vision, poor requirements elicitation techniques, specifying requirements too abstractly, and missing completeness checks. The most common effects (i.e., consequences) were time overrun, post-implementation rework, and poor product quality.

2.1.2. Requirements Technical Debt (RTD)

Requirements Technical Debt (RTD) captures the consequences of sub-optimal decisions made concerning requirements (Perera et al., 2024a). In other words, the phenomenon captures the consequences of problems with requirements caused either intentionally or unintentionally. An RTD Item (or an RTD Instance — we use this term throughout this paper) adopts the 'TD Item' from the TD literature (Avgeriou et al., 2016). An RTD Item could be introduced during the identification of requirements, e.g., inadequate capturing of user needs (Lenarduzzi and Fucci, 2019), or during the documentation of requirements, e.g., incomplete or outdated or inadequate requirements documentation, ambiguities in the Systems Requirements Specification (SRS), i.e., Requirement Smells (Charalampidou et al., 2017; Lenarduzzi and Fucci, 2019; Lenarduzzi et al., 2020; Mendes et al., 2016; Ojameruaye and Bahsoon, 2014). RTD could also be introduced during the implementation of the system, for example, through the poor or partial implementation of requirements (Lenarduzzi and Fucci, 2019; Kazman et al., 2001; Bonfim and Benitti, 2022; Sivzattian and Nuseibeh, 2001; Karlsson and Ryan, 1997). RTD may also concern non-functional requirements, for example, missed, outdated, or under-specified quality requirements (Costa et al., 2022) or how well a system architecture satisfies quality requirements (Kazman et al., 2001).

2.1.3. Negative consequences of RTD

Similar to TD, RTD could be harmful unless managed effectively. Failing to adequately capture user needs during RE could lead to building the wrong product (Barbosa et al., 2022; Ernst, 2012; Karlsson and Ryan, 1997). RTD Items such as Requirement Smells introduced during the documentation of requirements could also have a cascading impact on other software development activities such as having to re-design or re-implement or re-test parts of the system (Behutiye et al., 2022; Bonfim and Benitti, 2022; Karlsson and Ryan, 1997; Lenarduzzi et al., 2020). This could also potentially lead to the introduction of Code, Design, or Architecture TD. RTD could also lead to delivery delays, low external quality, low maintainability, financial loss, and impaired company image (Bonfim and Benitti, 2022; Barbosa et al., 2022; Frattini et al., 2023; Ojameruaye and Bahsoon, 2014). Therefore, we must pay attention to managing RTD. The TD management activities, such as *Identification, Monitoring, Prioritization, Measurement (or Quantification), Repayment, and Prevention* (Li et al., 2015; Junior and Travassos, 2022), also apply to RTD.

2.1.4. A definition for RTD and the RTD Quantification Model (RTDQM)

In our previous work reported in (Perera et al., 2023a, 2024a), we conducted a Systematic Mapping Study (SMS) in which we analyzed primary studies discussing this phenomenon with the aim to conceptualize RTD and its quantification. The studies found in our mapping study discussed various definitions for RTD, which we synthesized into a common formal definition: *"RTD captures the consequences of sub-optimal decisions made concerning requirements, either deliberately (for strategic gains) or inadvertently (due to changes in context), during the identification, documentation, and implementation of requirements as features or architectural design decisions"* (Perera et al., 2024a). This definition captures the notions of intentionality, the introduction of RTD during the different activities of software development, and the functional and non-functional aspects of requirements.

Furthermore, we developed a conceptual model, the *Requirements Technical Debt Quantification Model (RTDQM)*, that captures the concepts and relationships sufficient to model the quantification of RTD grounded in the literature found in our SMS. The model concepts were further categorized into high-level themes: *process or time, cost, benefit, probability, and priority*.

RTDQM is a theoretical foundation that allows one to understand and discuss RTD quantification. It serves as a reference point to compare and analyze existing RTD quantification approaches and as a guide to develop new approaches based on the model concepts. Our survey instrument is designed based on some of these concepts that are relevant to the scenario of fixing (or repaying or eliminating) RTD Items. This is further discussed in the Methodology (Section 3).

2.2. Related Work

2.2.1. Practitioner surveys on TD

The InsighTD project surveyed software practitioners to investigate TD prevention practices (Freire et al., 2020b), TD monitoring practices (Freire et al., 2024), and TD repayment practices (Freire et al., 2020a). Some work (Rios et al., 2019) was also done to investigate the causes and effects of the presence of TD in agile software projects. As part of the same project, Rios et al. (2020) surveyed practitioners on the causes, effects and practices of managing Documentation Debt. However, the InsighTD project did not survey practitioners regarding the quantification of TD. Our study focuses on TD quantification (or measurement), which is one of the TD management activities. We investigate TD quantification with respect to RTD, one of the TD types which was also not a specific focus of the InsighTD project. We hope that the findings of our study will contribute towards expanding the knowledge base developed through the series of surveys conducted via the InsighTD project. We provide directions for future research, in particular, concerning the quantification of RTD.

Codabux et al. (2017) gathered insights from practitioners about the following aspects of Technical Debt: definition, characterization, consequences, benefits, and how TD is communicated. They interviewed 17 practitioners and conducted a survey of 67 participants. They identified the most commonly adopted definition for TD (*"Technical debt is a tradeoff made due to schedule, scope, cost, and quality constraints."*), the most common method to measure TD as person-hours, and the most common method to identify debt as *allocating time in iterations to address debt.* The authors claim that an interesting outcome of their work is how to assess the risks of TD by evaluating liabilities beyond the cost of directly handling the debt. Their study focused on TD rather than RTD. Our study is different in that it focuses on RTD and especially on its quantification. However, in answers to their research question pertaining to notions of technical debt, they found RTD to be one of the aspects described by their respondents. As described by Codabux et al. (2017), examples from their survey respondents included de-scoping of non-functional requirements and *"what you built initially might no longer be right"* (Codabux et al., 2017), which the authors described as the instance where requirements change over time, and the software system no longer fits the needs of the customer. We complement the findings of Codabux et al. (2017)'s work by further investigating the management activity TD measurement (or quantification) with respect to a particular type of TD, that is, RTD. We utilized some of Codabux et al. (2017)'s findings when designing the pre-defined lists of options in our survey questions, for example, measuring TD in person-hours. One of the findings of Codabux et al. (2017)'s work was that the management of TD could become easier when the type of TD is known. Therefore, we believe that a better understanding of RTD quantification through our study will contribute to that goal.

2.2.2. Practitioner surveys on RTD

Frattini et al. (2023) developed an initial analytical theory to explore the debt metaphor in the context of Requirements Engineering (RE). They organized concepts related to practitioners' understanding and managing of Requirements TD into 23 falsifiable propositions and provided explanations to these propositions (Frattini et al., 2023). Their theory was derived from interview and survey data. We developed the RTD Quantification Model (RTDQM) in our previous work (Perera et al., 2023a, 2024a) grounded in the literature found via a Systematic Mapping Study (SMS) and with the use of Thematic Analysis (Braun and Clarke, 2006). The survey we report in this paper was designed based on some of our model concepts (Perera et al., 2024a), relevant to the scenario of fixing (or eliminating or repaying) RTD. We surveyed practitioners to understand their practices and perceptions about RTD quantification. Our findings validated that the model concepts were useful to quantify for decision-making. Even though the order of execution of the studies is different in ours and Frattini et al. (2023)'s work, they can complement each other — although executed independently, both studies found that the concepts of RTD are perceived to be similar to their TD counterpart but still may have unique aspects relevant to RTD. Some of the pre-defined options in our survey questions are based on our previous work as well as Frattini et al. (2023)'s work. Frattini et al. (2023) found that measuring and tracking requirements engineering debt is yet immature in practice, our work makes contributions towards filling this gap. Our findings offer insights into industry perceptions about RTD quantification, one step further from the theoretical findings.

2.2.3. Identification and Quantification of RTD

Melo et al. (2022), conducted a Systematic Literature Review (SLR) on 66 primary studies from 2010 to 2020. They identified causes for RTD, existing strategies for identification and measurement (i.e., quantification) and metrics to support the measurement. However, the 66 primary studies that resulted in their query did not exclusively focus on RTD. Hence, the metrics they were proposing for measurement were too based on prior work on other TD types rather than prior work on RTD. For example, one of the studies listed as 'quantification approach' in their study, the work by Nugroho et al. (2011), is not focused on RTD. Compared to their study, in our previous work, where we conducted Systematic Mapping Study (SMS) on RTD quantification (Perera et al., 2023a, 2024a), we specifically focused on previous works on RTD and the *quantification of RTD* in our search query. In the survey we report in this paper, we provide insights into the industry practices of quantifying RTD and practitioners' perceptions about its support for deicion-making, based on the theoretical findings from our previous work. The findings of our survey show the gap between the current tools in the industry and the needs of the practitioners in terms of RTD quantification. This confirms the gap identified by Melo et al. (2022), the absence of tools and software that can be used in the context of RTD.

In our previous work (Perera et al., 2023a), where we conducted an SMS on RTD quantification, we initially identified seven approaches discussing RTD quantification and proposed an initial conceptual model for RTD quantification based on the literature found in the SMS. The key observation from our work was that, although RTD is similar in many aspects to TD in the software code it has its own components. We then extended this work, adding 11 new approaches from the literature (Perera et al., 2024a) in a second round of conducting the SMS. The initial model for RTD quantification was updated to accommodate newly identified concepts. The initial model was also updated to include RTD quantification for the non-functional aspect of requirements. We then combined our model with our previous work (Perera et al., 2024b), presenting a combined model which shows the cascading impact of RTD on other development activities, such as implementation and design.

Maldonado and Shihab (2015) examined source code comments from five open-source projects to identify what types of TD could be seen as self-admitted by developers. In Self-Admitted TD (SATD), developers know that the current implementation is not optimal and write comments in the code to indicate it. By classifying the code comments, the authors found that SATD could be classified into five different types of TD: design, defect, documentation, requirements and test. The most common type was design debt. The second most common type was requirements debt. The contribution of their work was a publicly available dataset of 33K comments categorized into the types of SATD. Our work further investigates RTD (i.e., requirements debt) and can complement their work by providing aspects that could be a starting point to further investigate with respect to self-admitted RTD.

2.2.4. Comparison against Related Work

The research area is quite new. There are no studies that specifically survey practitioners regarding RTD quantification. Therefore, to the best of our knowledge ours is the first study surveying software practitioners to learn about RTD quantification practices and how quantification could support decision-making for managing RTD. In our case, we considered the scenario of making the decision to fix RTD instances or in other words, repay the debt. We compared our contributions with the most closely related work discussed above in Section 2 in a summarised Table format as shown in Table 1 below.

Table 1: Overview of Contributions of Related Studies

Study	Focus	Contributions
InsighTD Project — Freire et al. (2020b), Freire et al. (2024), Freire et al. (2020a), Rios et al. (2019) and Rios et al. (2020)	TD	TD prevention practices, TD monitoring practices, TD repayment practices, Causes and effects of the presence of TD in agile software projects, Causes, effects and practices of managing Documentation Debt
Codabux et al. (2017)	TD	Findings on definition, characterization, consequences, benefits, and how TD is communicated
Frattini et al. (2023)	RTD	An initial analytical theory to explore the debt metaphor in the context of RE
Melo et al. (2022)	RTD Identification and Quantification	Causes for RTD, Existing strategies for identification and measurement, Metrics to support the measurement (based on metrics for TD)
Perera et al. (2023b)	RTD Quantification	A conceptual model for RTD quantification that serves as a theoretical foundation to compare and analyze existing quantification approaches and as a guidance to develop new quantification approaches
Maldonado and Shihab (2015)	SATD Identification and Quantification	A publicly available dataset of 33K comments categorized into the types of SATD
This Study	RTD Quantification	Survey instrument, Implications and future research directions for RTD quantification, Replication package

3. Methodology

3.1. Research Questions and Research Method

We aim to answer the following research questions through our study. The chosen research method was a survey, i.e., an anonymous online questionnaire. We followed the guidelines for conducting surveys in software engineering provided by Linåker et al. (2015), Kitchenham and Pfleeger (2008) and Ciolkowski et al. (2003) to conduct our survey. Furthermore, the knowledge we gained from developing our conceptual model reported in our previous work (Perera et al., 2024a) (and discussed briefly in Section 2.1) guided the development of the survey instrument. This is further discussed in Section 3.2.1.

Survey research allows for gathering insights into opinions, experiences, and needs with respect to practices and problems in software engineering (Mendez et al.) Our goal is to learn about current practices of quantifying RTD and perceptions about the support of RTD quantification for decision-making from software practitioners in the industry. Hence, a survey was deemed the useful approach to gathering data from a larger sample of software practitioners in the industry at a lower cost (Stol and Fitzgerald, 2020). Purposes of survey research can be either explanatory, exploratory or descriptive, according to Wohlin et al. (2012). Our survey takes the nature of both descriptive and exploratory. The first research question takes a descriptive nature, while the second one takes a descriptive as well as an exploratory nature.

- **RQ1: What are the current industry practices of quantifying RTD?**
- **RQ2: How could RTD quantification support informed decision-making for managing RTD?**

The goal of RQ1 is to investigate the current practices regarding RTD quantification in the industry (i.e., how often practitioners fx RTD, whether they formally or informally quantified RTD when fixing and reasons for not quantifying), while the goal of RQ2 is to investigate how RTD quantification could support informed decision-making according to practitioners' perceptions (i.e., what concepts could support decision-making and what tools, techniques and other factors could help).

We designed our survey instrument to revolve around the scenario of fixing problems with requirements gathering insights from practitioners about decisions made with respect to fixing the problems that have consequences down the line, i.e., fixing RTD Instances or RTD Items. This included, in particular, asking software practitioners about problems faced with requirements that created a significant impact, whether these instances were fixed and how often (or when) they were fixed, whether the cost of fixing and the benefit of fixing were quantified, whether there were consequences of not fixing the problems, and whether such consequences were quantified to help make the decision to fix.

Since we considered decision-making given the scenario of fixing RTD instances, the sub-research questions were designed to fit the scenario of fixing RTD. The detailed research questions are as follows. By 'concepts', we mean the RTDQM model concepts relevant to the scenario of fixing RTD (i.e., Cost of fixing, and Benefit of fixing and Consequences of not fixing RTD), as described later in Table 2 and discussed under Section 3.2.1.

- **RQ1: What are the current industry practices of quantifying RTD?**
  - **RQ1.1 How often do practitioners fix RTD?**
  - **RQ1.2 What concepts of RTD quantification were formally or informally quantified by practitioners when fixing RTD?**
  - **RQ1.3 What are the reasons for not quantifying RTD?**
- **RQ2. How could RTD quantification support informed decision-making for managing RTD?**
  - **RQ2.1 What concepts of RTD quantification support making informed decisions to fix RTD?**
  - **RQ2.2 What tools, techniques, and other factors may help?**

3.2. Study Design

We prepared an anonymous online questionnaire using the Qualtrics online platform[2]. The survey was self-administered. But each section of the survey contained an introduction to help the respondents familiarize themselves with the topic on which the questions of the section were based prior to answering the questions in that section. The estimated completion time for the questionnaire was approximately 25 minutes. The demographic questions were split into two sections and displayed at the beginning and the end of the survey. Participant demographics are reported in Section Appendix B. A summary of the characteristics of the dataset can be found in Section 3.4.2. We sought feedback from two experts in Empirical Software Engineering and TD outside our research project and made improvements to the survey prior to releasing it for data collection.

3.2.1. Question Design

The survey questions included primarily *Single-choice* and *Multiple-choice* questions. Five-point *Likert scale* questions sought agreement or disagreement for a given statement. A few *Open-ended* questions allowed the respondent to answer freely with their opinion. For every question where a list of options was given to choose from, the options also contained the option 'Other' with a free text field to give the respondents the opportunity to specify a different option than what was listed. The listed options for the questions were designed based on our previous work described in Perera et al. (2024a) and related work discussed earlier in Section 2.2 such as the works of Codabux et al. (2017) and Frattini et al. (2023). A simplified version of the survey questions can be found in Table A.10 in the appendix along with their question type and the survey section they belong to. The Table also lists the pre-requisite questions for a particular survey question. This means the respondents may have taken a different path depending on how they answered one question. The mapping between the Research Questions and Survey questions can also be found in Table A.10. The complete questionnaire can be found in our Replication Package[3].

[2]Qualtrics: `https://www.qualtrics.com/strategy/research/survey-software`
[3]Replication Package: `https://doi.org/10.5281/zenodo.14172401`

Table 2: RTDQM concepts that informed the question design of the survey instrument

Model Concept	**Description**	**Category**	**Term Used in the Survey**
RTD Item	A unit of TD pertaining to software requirements i.e., represents the consequences of sub-optimal decisions made concerning software requirements	Process/time	problems with requirements
RTD Rectification Step	A software development process step where developers spend time and effort on rectifying (or remediating or eliminating or repaying) RTD	Process/time	fixing RTD
Cost of Rectifying RTD	The cost to rectify (or remediate or eliminate or repay) RTD during a Rectification step.	Cost	Cost of fixing
Benefit of Rectifying RTD	Benefit gained by rectifying (or remediating or eliminating or repaying) RTD	Benefit	Benefit of fixing
RTD Interest	The additional costs or consequences incurred due to the presence of RTD	Cost	Consequences of not fixing
RTD Interest Constituent: Rework Code Costs	Rework code costs incurred due to RTD, e.g., refactoring software code	Cost	Consequences of not fixing, examples were provided in the survey
RTD Interest Constituent: New Code Costs	New code costs incurred due to RTD, e.g., writing new software code	Cost	Consequences of not fixing, examples were provided
RTD Interest Constituent: Rework Design Costs	Rework design costs incurred due to RTD, e.g., redesign of the software architecture	Cost	Consequences of not fixing, examples were provided
RTD Interest Constituent: New Design Costs	New design costs incurred due to RTD, e.g., adding new components to the software architecture	Cost	Consequences of not fixing, examples were provided
RTD Interest Constituent: Rework RE Costs	Rework Requirements Engineering costs due to RTD, e.g., refining the System Requirements Specification (SRS)	Cost	Consequences of not fixing, examples were provided
RTD Interest Constituent: New RE Costs	New Requirements Engineering costs due to RTD, e.g., having to conduct new user interviews	Cost	Consequences of not fixing, examples were provided

Demographics. Demographic questions were Q1 to Q3 and Q41 to Q44 (See Table A.10). These questions were related to, for example, the role of the respondent (e.g., business analyst, software developer), what activities they performed in their day-to-day work (e.g., gathering requirements, formalizing requirements, implementing features), how experienced they were in their role (e.g., 2-5 years, more than ten years), country of work (e.g., New Zealand, Australia), the size and context of the company they worked for (small businesses, small to medium enterprises, large businesses, domestic and international businesses), and what application domains they were involved in, for example, agriculture, manufacturing, and financials and what software development methodology they practised (e.g., agile, waterfall). This section helped us make observations based on the classification of the respondents according to their demographics.

Quantifying RTD. The goal of this survey section was to answer RQ1 and RQ2. This section included survey questions Q4 to Q22 (See Table A.10). The questions in this section were designed based on the concepts in our conceptual model (e.g., cost, benefit and process/time related concepts) developed in Perera et al. (2023a) and Perera et al. (2024a). The model concepts (and their descriptions) that informed the survey instrument can be found in Table 2. Those concepts were relevant to the scenario of fixing RTD instances. For example, the concept 'RTD Interest' refers to the notion of interest on the debt while 'Cost of rectifying RTD' refers to the notion of principal, the amount of effort required to eliminate or repay the debt. Table A.10 shows the mappings between model concepts and the survey questions.

The purpose of using the model concepts to design our survey was to gather industry perceptions based on our theoretical findings. But we did not show the model to our participants or use the model concepts as terminology in the survey questions since we did not want to bias our participants. An introduction screen was displayed at the beginning of the survey section so that the respondents would be familiar with the phenomenon of RTD to make sure that they interpreted the survey questions accurately. Figure 1, shows part of the introduction screen. From there onwards, we used the term 'problems with requirements' to refer to RTD instances throughout the survey. Given that most participants would not be familiar with the term RTD, we used the simplified term so that it is more understandable for our respondents when answering the survey questions. Furthermore, we used terms such as 'measure' and 'estimate', apart from 'quantification', to ensure that we described the questions in a way that they were well understood by our participants. The reason for using terms that could be more familiar to practitioners rather than using technical jargon was to mitigate the threat to construct validity. Construct validity refers to how well the survey instrument measures what it was designed to measure. We attempted to achieve the same by providing the introduction screen at the beginning of the survey section as well as by using consistent terms throughout the survey.

The respondents were asked about RTD instances (i.e., consequences of sub-optimal decisions made concerning requirements or 'problems with requirements' that had consequences) in Q4, Q5 and Q6 to set the scene for RQ1 and RQ2 prior to displaying the rest of the survey questions. Q4 asked about the most common RTD instances encountered by the survey respondents in their current role. A pre-determined list to select from and the option 'Other' was provided. Then Q5 used the Critical Incident Technique (CIT) (Flanagan, 1954) to ask practitioners about a recent incident that created a significant impact due to problems with requirements. The respondents had to briefly describe, in one to three sentences, what happened. Q6 asked about the RTD instances encountered in the incident described by the respondents who answered Q5. Again, a pre-determined list and the option 'Other' was provided. We discuss the RTD instances in Section 4.

Q7 to Q17 were follow-up questions. Q7 inquired how often practitioners generally fixed the RTD instances. Q8 to Q17 were asked to understand whether the RTD instances described with respect to the critical incident were fixed, if so, when they were fixed, and whether they quantified associated costs, benefits, and consequences.

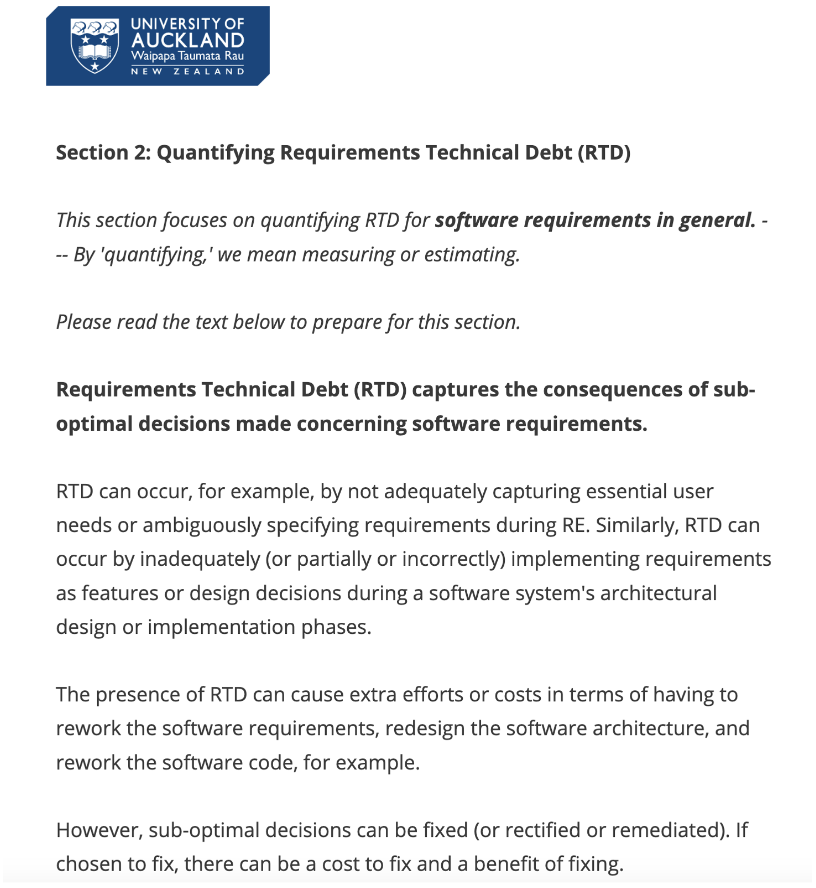
Q18 to Q20 were Likert scale questions that sought agreement among practitioners regarding the usefulness of quantifying different concepts of RTD quantification to make an informed decision regarding fixing RTD. Q21 and Q22 were open-ended questions that sought practitioners' perceptions on what existing tools and techniques support quantification and what other factors may be useful to quantify to support informed decision-making for RTD management.

3.2.2. Participant selection

The study was targeted toward gathering and analyzing data from software practitioners (e.g., business analysts, requirements engineers, software engineers, software developers, software architects, and project managers) engaged in software development, particularly in Requirements Engineering (RE), software design, and implementation activities.

We used *Convenience sampling* and *Snowball sampling* for obtaining our sample of participants (Linåker et al., 2015; Ralph et al., 2020). We advertised the invitation to participate in this study through the researchers' professional networks (e.g., LinkedIn and Twitter), the industrial connections of Veracity Lab[1], and the professional networks of

[1]Veracity Lab: `https://veracity.wgtn.ac.nz/`


UNIVERSITY OF AUCKLAND Waipapa Taumata Rau NEW ZEALAND

Section 2: Quantifying Requirements Technical Debt (RTD)

This section focuses on quantifying RTD for ***software requirements in general.*** *--- By 'quantifying,' we mean measuring or estimating.*

Please read the text below to prepare for this section.

Requirements Technical Debt (RTD) captures the consequences of sub-optimal decisions made concerning software requirements.

RTD can occur, for example, by not adequately capturing essential user needs or ambiguously specifying requirements during RE. Similarly, RTD can occur by inadequately (or partially or incorrectly) implementing requirements as features or design decisions during a software system's architectural design or implementation phases.

The presence of RTD can cause extra efforts or costs in terms of having to rework the software requirements, redesign the software architecture, and rework the software code, for example.

However, sub-optimal decisions can be fixed (or rectified or remediated). If chosen to fix, there can be a cost to fix and a benefit of fixing.

Figure 1: Screenshot showing part of the introduction to the RTD quantification section of the survey

the researchers in the Veracity Spearhead project, by which this work is funded. Participants were allowed to share the invitations with their colleagues who fit within the inclusion criteria for participation. Participation in the survey was entirely voluntary. Anyone who accepted the invitation and met the inclusion criteria was included in the online anonymous questionnaire.

***Participant inclusion criteria*.**

- *Practitioners engaged in Requirements Engineering (RE) or software implementation activities in the technology sectors in New Zealand and overseas.*
- *Proficient in English.*

The study was approved by the University of Auckland Human Participants Ethics Committee on 13/11/2023 for three years (Reference Number UAHPEC26580).

3.3. Data collection

The survey invitation was sent to the participants on the 9th of February, 2024. We closed the survey on April 23, 2024, as the responses slowed dramatically and did not seem to increase anymore at that point. There was a total of 82 responses. Out of those responses, 52 responses were considered valid.

3.4. Data Analysis

3.4.1. Data Cleaning and Pre-processing

The survey responses were downloaded as a CSV file from Qualtrics. Afterward, the file was imported to a Python environment for data analysis. Empty responses and responses where the progress rate was less than 13% out of the overall progress of the survey were removed. Responses with less than 13% progress were responses where the participants did the demographic questions at the beginning of the survey, answered only the first RTD quantification question and exited the survey. Finally, 52 responses were considered valid. These 52 responses may not indicate a 100% progress since all the survey questions were optional to respond to and some questions were also displayed based on a pre-requisite question. We analyzed the data for each question based on the number of respondents who answered each question. Therefore, it was not a must to have complete responses where participants completed all the questions. Therefore, the 52 responses were considered as valid.

3.4.2. Dataset

The final dataset comprised 52 valid responses out of which 32 respondents mentioned the country of work while 20 did not. The highest number of respondents out of those who mentioned the country of work were from New Zealand (23.08%). Sri Lanka (11.54%), Sweden (5.77%) and Germany (5.77%) followed. Since demographic questions were split across two sections of the survey (displayed at the begining and the end of the survey), the 'Country of work' question was displayed at the end of the survey. Therefore, some respondents did not answer that question. At the same time, some respondents may have chosen not to reveal their country of work since the survey question was optional.

Based on the type of company, either domestic or international Small businesses, Small to Medium enterprises and Large Businesses, 25% was the highest number indicating Large International Businesses as the most common type among those who mentioned their type of company. Out of 52 respondents, 50% were in the Software Engineer/ Systems Engineer/ Developer category. Other roles (17.31%) and Systems Architect/ Software Architect/ Tech lead category (9.62%) followed. There were also respondents who were in non-technical roles such as the Project Manager/ Product Manager/ Team Lead category (7.69%) and the Business Analyst/ RE Engineer category (5.77%) and QA Manager/ QA Lead/ QA Engineer category (1.92%). However, the technical roles were the most common. Most respondents (30.77%) had 5-10 years of work experience followed by those who had 1-2 years of experience (23.08%) and more than 10 years of experience (17.31%). Most respondents followed a Hybrid approach of Agile and Waterfall in their companies (34.62%). However, there were also those who followed agile only (32.63%) and waterfall (i.e., traditional) only (1.92%). The most common application domain was Finance (12.36%). Most respondents spent their time implementing feature enhancements (11.86%) followed by Refactoring (11.22%) and Writing or modifying software code to implement features (10.9%).

In summary, our dataset comprises of respondents from different countries, different company sizes following mostly a hybrid process model but also other process models, having expert to novice levels of experience, and in different roles but mostly in technical roles. Therefore, our dataset represents various software development contexts. More details on the demographics of our dataset can be found in Appendix B.

3.4.3. Analyzing Closed-Ended Questions

We used descriptive statistics and calculated the percentage of respondents who chose each option in each closed-ended question.

3.4.4. Analyzing (Coding) Open-Ended Questions

We conducted a qualitative content analysis on the responses to the open-ended questions. We utilized the NVivo[1] software to code the responses. For this, the responses for the open-ended questions in the valid responses were imported to NVivo. The complete record of responses to the open-ended questions from one respondent was first assigned a label e.g., R1, R2. Afterward, the responses were coded within each open-ended question (e.g., R1_Q20).

The methodology we followed for the coding process was Thematic Analysis by Braun and Clarke (2012). We first coded the responses and then grouped them into common themes that emerged from the coding and then extracted

[1]NVivo: `https://techcenter.qsrinternational.com`

meaningful insights. Responses that discussed a similar theme were coded under one theme (i.e., code), and themes that belonged to a high-level theme (i.e., high-level code) were grouped together. For example, themes 'proprietary tools' and 'internal tools' belonged to the high-level theme 'tools'. A single response could be coded under multiple codes if it pertained to multiple themes. Coding was performed by the first author. At least two other authors reviewed the initial codes prior to the weekly consensus meetings where disagreements were discussed and resolved.

4. RTD Instances

Few survey questions were aimed at understanding the RTD Instances encountered by practitioners. Survey respondents were asked about the most common RTD instances they encountered at work generally (Q4) and w.r.t. to a critical incident described by them (Q6). Some examples were given for the respondents to select from and the option 'Other' was given additionally if the respondent preferred to give a different answer.

Generally, the most common RTD instance was 'Insufficient documentation' (63.46% of respondents out of those who answered this question thought so). The next most common RTD instances were 'Not capturing essential user needs when gathering requirements' (55.77%) and 'Changes in the requirements documentation' (51.92%). More than half of the respondents found the top three RTD instances to be common. Table 3 shows the percentages.

The most common RTD instance that had a significant impact in critical incidents described by respondents was 'Changes in the requirements documentation' (42.86% out of those who answered the question mentioned it). 'Lack of requirements validation with the user' (36.73%) and 'Insufficient documentation' (34.69%) followed. Table 4 shows the percentages of RTD instances in critical incidents.

Table 3: RTD instances (in general)— Options: Lenarduzzi and Fucci (2019), Bonfim and Benitti (2022), Perera et al. (2023b), Frattini et al. (2023)

Option (summarized)	**Number of respondents (n = 49)**	**Percentage**
Insufficient documentation of requirements	31	63.46
Not capturing essential user needs while gathering requirements	27	55.77
Changes in the requirements documentation	25	51.92
Inadequate (incorrect or partial) implementation of requirements	24	50.0
Trade-offs between what requirements to implement during prioritization of requirements	24	48.08
Ambiguities in requirements documentation	23	46.15
Lack of requirements validation with the user	19	38.46
Lack of requirements traceability	19	38.46
Requirements are not satisfied by the system design	14	28.85
Not documenting requirements at all	14	28.85
Other (Please specify)	5	9.62

Table 4: RTD instances in critical incident — Options: Lenarduzzi and Fucci (2019), Bonfim and Benitti (2022), Perera et al. (2023b), Frattini et al. (2023)

Option (summarized)	**Number of respondents (n = 49)**	**Percentage**
Changes in the requirements documentation	21	42.86
Lack of requirements validation with the user	18	36.73
Insufficient documentation of requirements	17	34.69
Not capturing essential user needs while gathering requirements	15	30.61
Requirements are not satisfied by the system design	14	28.57
Inadequate (incorrect or partial) implementation of requirements	11	22.45
Ambiguities in requirements documentation	10	20.41
Lack of requirements traceability	10	20.41
Trade-offs between what requirements to implement during prioritization of requirements	8	16.33
Not documenting requirements at all	5	10.2
Other (Please specify)	5	10.2

Although 'Changes in the requirements documentation' was the most common RTD instance in the critical incidents reported by practitioners, it is arguable whether this could be considered an RTD instance on every occasion, especially in the agile context, since change is expected in that context according to Wagner et al. (2017). Changes in requirements could also stem from other RTD instances, such as Requirement Smells or insufficient documentation of requirements. For example, a requirements change can be triggered when a Requirement Smell is corrected or when an initially specified requirement is split into multiple atomic requirements at a later stage. However, despite the various instances of requirements changes that could be described, we considered 'Changes in the requirements documentation' an RTD instance since, in the context of our survey, *it is a requirements problem that created a significant impact, i.e., had consequences due to sub-optimal decisions made concerning requirements*, which aligns with the definition of RTD discussed in Section 2.1.

Documentation of the requirements was found to be an important aspect since insufficient documentation was a common problem generally as well as in a critical incident. According to our survey responses, capturing essential user needs and validating with the end user is also important to reduce RTD instances from being introduced. This emphasizes the importance of user feedback loops, which was also one of our key findings reported in Perera et al. (2024a). For example, the problem of not capturing essential user needs would be solved by obtaining early feedback from the user. Changes in the requirements documentation would also be less likely to happen by having continuous user feedback.

5. Results

5.1. RQ1: What are the current industry practices of quantifying RTD?

RQ1 investigated the current practices of quantifying RTD in the industry. By this, we mean how often the practitioners quantified (RQ1.1) and whether they formally or informally quantified the costs, benefits and consequences of RTD relevant to the scenario of fixing the RTD instances (RQ1.2). There can be other scenarios where RTD management decisions can be made, for example, to prevent RTD instances or even to incur RTD prudently to gain a faster time to market with the intention of rectifying RTD in the next release(s). We focused on the scenario of fixing RTD instances and the quantification in that context. If practitioners did not quantify, the reasons for not doing so were inquired (RQ1.3).

5.1.1. How often practitioners fix RTD Instances (RQ1.1)

Survey question Q7 asked how often practitioners generally fixed RTD instances. Table 5 shows that practitioners fix them mostly during the implementation of features (68.75% out of those who answered the question). 54.17% of the respondents selected that they fix during the design of the software architecture, while 50% chose to fix only during testing. 47.92% chose to fix before designing or implementing the system. 29.17% respondents mentioned that the fixes were generally quick fixes just before delivering the software, while 4.17% respondents noted that it was not a common practice to fix RTD instances for them. Respondents who selected the choice 'Other,' however, did not specify anything.

Survey question Q8 asked when the RTD instances in the critical incidents described by the practitioners were fixed. Table 6 shows that the most common timing for fixing RTD instances encountered in the critical incident was also during the implementation of software features (32.65% out of those who answered the question). The second most common timing seems to be only when the end user complained after delivery (12.24%). However, in the critical incident, a similar percentage of respondents chose to fix RTD instances before designing or before implementing the software system (12.24%). A similar percentage chose to fix them during testing. Interestingly, 12.24% also did not fix the RTD instances in the critical incident. However, no reasons were given. According to the results, regardless of whether it is a critical incident or not, practitioners tend to fix RTD mostly when implementing software features.

5.1.2. Concepts formally or informally quantified by practitioners (RQ1.2) and Reasons for not quantifying them (RQ1.3)

Survey questions Q9 - Q15 helped answer the research questions RQ1.2 and RQ1.3 (See Table A.10). We asked practitioners whether they formally (e.g., in terms of person-hours, story points or dollars) or informally (e.g., had some expectation of extra costs in terms of effort spent) quantified the Cost of fixing (Q9), the Benefit of fixing (Q11), and the Consequences of not fixing (Q14) RTD when they fixed instances that led to a significant impact in the critical incident they described (Q8). Survey questions Q9 and Q11 were asked only if the practitioners responded to Q8 and said they fixed the RTD instances at some point. If the participants did not formally or informally quantify the cost, benefit, or consequences, the reasons for not doing so were inquired via the survey questions Q10, Q12, and Q15.

Table 5: How often practitioners fix RTD instances in general — Options: Frattini et al. (2023)

Option	**Number of respondents (n = 48)**	**Percentage**
During implementing software features	33	68.75
During implementing the software architecture	26	54.17
During testing software features	24	50.0
Before designing or implementing the software system	23	47.92
Quick fixes just before delivering the software	14	29.17
It's not a common practice to fix unless the end user complaints	2	4.17
Other (Please specify)	1	2.08

Table 6: When RTD instances in critical incident were fixed — Options: Frattini et al. (2023)

Option	**Number of respondents (n = 49)**	**Percentage**
During implementing software features	16	32.65
Fixed only when the end user complained after delivery	6	12.24
Before designing or implementing the software system	6	12.24
During testing software features	6	12.24
Did not fix them	4	8.16
A quick fix just before delivering the software	4	8.16
Other (Please specify)	1	2.04

Cost of fixing RTD Instances. Table 7 shows that most of the practitioners (46.88%) responded that they did not formally or informally quantify the Cost of fixing (i.e., Cost of Rectifying RTD). The most common reason for not quantifying was that the respondent was not the decision-maker and did not feel the need to quantify (See Table 8 — 40% of the respondents who answered this question said so). 26.67% said that some decisions are made by expert judgment rather than quantifying. 6.67% said that there was no reliable way to quantify the cost, while 13.33% said there were reliable ways, but they were time-consuming. Other reasons given by the respondents included not taking responsibility and the fact that the fix was a new project. Some examples are as follows. Participants' responses verbatim are within double quotes:

- Not taking responsibility: *"Since most of those are issues from the project team"*
- Fix being a new project: *"We considered the fix as a new project"*

However, some practitioners (34.38%) formally quantified the effort to fix, for example, in terms of person-hours or story points or in dollars or other currency. 18.75% respondents informally quantified, for example, had some expectation in terms of effort spent.

Benefit of fixing RTD Instances. According to Table 7, most practitioners informally quantified the Benefit of fixing (i.e., Benefit of Rectifying RTD) even though they did not quantify the Cost of fixing (64.52%). A percentage of 9.68% formally quantified. The most common reason for not quantifying was that the respondent was not the decision-maker and did not feel the need to quantify, similar to the reasons for not quantifying the Cost of fixing (See Table 8— 50% of the respondents who answered this question mentioned so). 25% said that some decisions are made by expert judgment rather than quantifying, while 12.5% said that there was no reliable way to quantify the benefit. The only other reason given by the respondents was that it was not a priority.

- Not a priority: *"It was not deemed a priority to do so."*

Survey question Q13 was asked, based on the RTDQM model concepts pertaining to the RTD Interest constituents, the new and rework costs associated with the RE, Implementation, and Design steps in the software development process that could be incurred as consequences of not fixing RTD instances. If the practitioners selected a choice that indicated that there were such costs, then Q14 asked if they formally or informally quantified such costs. Q15 asked about the reasons for not quantifying the consequences.

Table 7: Quantification practices

Option (Summarized)	Cost of Fixing (n = 32)		Benefit of Fixing (n = 31)		Consequences of Not Fixing (n = 37)	
	Number	Percentage	Number	Percentage	Number	Percentage
Formally quantified	11	34.38%	3	9.68%	13	34.84%
Informally quantified	6	18.75%	20	64.52%	12	32.43%
Did not quantify at all	15	46.88%	8	25.81%	11	29.73%

Table 8: Reasons for not quantifying

Option (Summarized)	Cost of Fixing (n = 15)		Benefit of Fixing (n = 8)		Consequences of Not Fixing (n = 11)	
	Number	Percentage	Number	Percentage	Number	Percentage
Not the decision-maker	6	40%	4	50%	4	36.36%
Decisions are made by experience or expert judgement	4	26.67%	2	25%	4	36.36%
Reliable but time consuming ways	2	13.33%	-	-	1	9.09%
Reliable but expensive ways	-	-	-	-	1	9.09%
No reliable ways	1	6.67%	1	12.5%	-	-
Other (Please specify)	2	13.33%	1	12.5%	1	9.09%

Table 9: Consequences of not fixing RTD (i.e., RTD Interest) — Options: Perera et al. (2023b)

Option	Number of respondents (n = 47)	Percentage
Rework code costs e.g., refactoring software code	23	48.94%
Rework design costs e.g., redesigning the software architecture	18	38.3%
New code costs e.g., writing new software code	17	36.17%
New design costs e.g., adding new components to the software architecture	16	34.04%
Rework RE costs e.g., refining the requirements specification	11	23.4%
New RE costs e.g., having to conduct new user interviews	8	17.02%
Other (Please specify)	5	10.64%
No consequences	4	8.51%

Consequences of not fixing RTD Instances. According to Table 9, 'rework code costs' associated with the software code was the most common consequence, i.e., RTD Interest constituent (48.94%). 'Rework design costs' was the next most common consequence (38.3%), and the 'new code costs' followed (36.17%). It is evident from the top three most common consequences that RTD interest has a cascading impact on later stages of software development.

Survey question Q14 asked whether the practitioners quantified these extra efforts or costs, i.e., consequences incurred in the project due to not fixing the RTD instances. According to Table 7, most practitioners formally quantified the consequences (37.84%). Some of them informally quantified the consequences (32.43%); for example, they had some expectation of the extra costs in terms of the effort spent. However, 29.73% practitioners did not quantify. The most common reason, according to Table 8, is again that the respondent is not the decision-maker — 36.36% of the respondents who answered this question said so. A percentage of 9.09% said that there was no reliable way to quantify the consequences, while a similar percentage said there were reliable ways, but they were time-consuming. Other reasons provided by the respondents included themes such as problems found only during the development of an MVP, the entire project being scrapped or discontinued, bugs being added to the backlog, and non-financial costs. Some examples are as follows:

- Problems were found only during developing a MVP: *"We pivoted out as the problem was found during building a MVP (Minimum Viable Product) to evaluate the Epic Hypothesis"*
- Project was discontinued: *"The entire project was scrapped "*

In conclusion, for all three concepts, costs, benefits, and consequences, the most common reason for not quantifying was that the respondent was not the decision-maker.

Summary of Findings (RQ1):

- **RQ1.1:** RTD is fixed mostly during the implementation of software features, regardless of whether it is in general or in a critical incident.
- **RQ1.2:** Most practitioners did not formally or informally quantify the Cost of fixing, while they did informally quantify the Benefit of fixing. Most practitioners formally quantified the Consequences of not fixing. The most commonly experienced consequences were: rework code costs, rework design costs, and new code costs.
- **RQ1.3:** Most practitioners who did not quantify were not the decision-maker.

5.2. *RQ2. How could RTD quantification support informed decision-making?*

Survey questions Q16 - Q22 helped answer this research question (See Table A.10). Q16 - Q20 were Likert Scale questions, while Q21 and Q22 were Open-Ended questions. For Q16 - Q20, participants were asked to rate their level of agreement with a given statement (RQ2.1) based on the following scale: *Strongly Agree, Somewhat Agree, Neither Agree nor Disagree, Somewhat Disagree and Strongly Disagree.* The open-ended questions, Q21 and Q22, collected responses about tools and techniques currently used in the industry for RTD quantification and other factors that may be useful to consider in decision-making regarding fixing RTD (RQ2.2). One way of managing RTD is through the payment of debt by fixing RTD instances.

5.2.1. *Concepts that support making informed decisions to fix RTD (RQ2.1)*

Figure 2 shows the level of agreement among the survey respondents (from Strongly Agree to Strongly Disagree) about the usefulness of quantifying the 'Cost of fixing' (Q16), 'Benefit of Fixing' (Q17) and 'Consequences of not fixing' (Q18), quantifying both costs and benefits (Q19), and quantifying all, costs, benefits and consequences (Q20), to make informed-decisions regarding fixing RTD instances, for example, to decide whether to fix and when to fix the RTD instance.

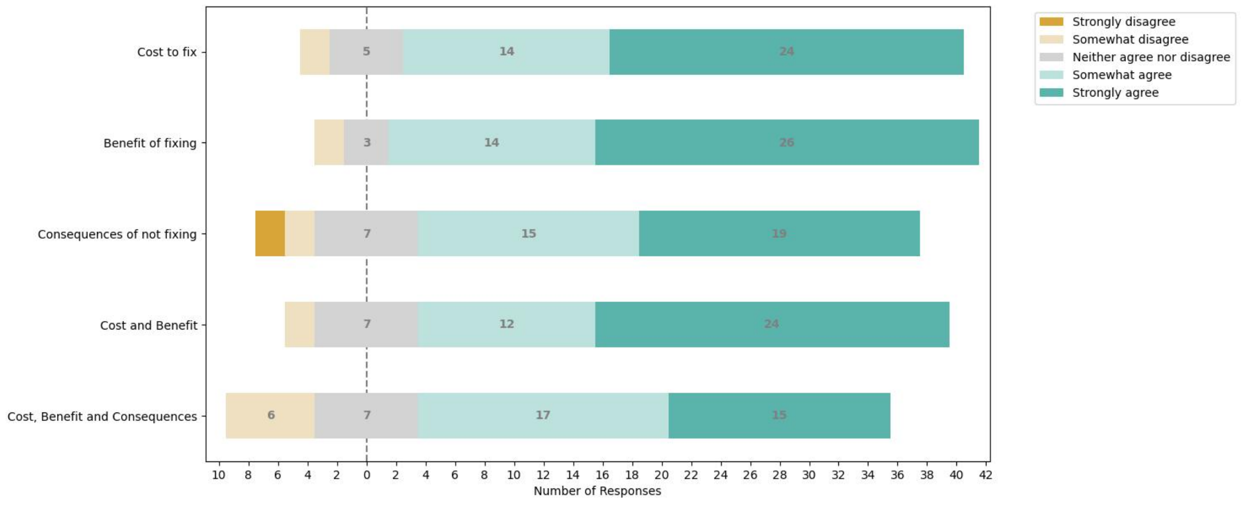


Figure 2: Level of Agreement for Likert Scale Questions Q16 - Q20 — Numbers are out of 45, 46, 46, 46 and 46 who rated Q16 - Q20, respectively.

According to the results obtained, more than half of the respondents strongly agree that quantifying the Cost of fixing and the Benefit of fixing supports informed decision-making (53.33% of the respondents agreed that quantifying the Cost of fixing supports and 56.52% agreed that the Benefit of fixing supports). Furthermore, more than half of the respondents strongly agree that both the cost of fixing and the Benefit of fixing must be quantified together for informed decision-making rather than the cost alone (52.17% of the respondents). However, this is less than the number of respondents who rated the Cost and Benefit concepts alone. Figure 2 illustrates the numbers of respondents according to the level of agreement.

We asked respondents for the reasons for their ratings. Common themes that could be identified among the reasons given by those who strongly agreed that quantifying the Cost of fixing is important for informed decision-making included: to avoid or reduce unnecessary expenses, to have more information, cost being an important factor to evaluate, and because estimating also costs effort. For example:

- Cost is an important factor to evaluate: *"Important for the PO to evaluate whether the issue should be fixed. Important for the dev team to check whether there is a common understanding about the scope to fix the issue"*

Common themes that could be identified among the reasons given by those who strongly agreed that quantifying the Benefit of fixing is important for informed decision-making were that the benefit is an important factor in deciding or prioritizing and it is helpful to understand the impact on the business. Some example quotes from the respondents are as follows:

- Benefit is an important factor to decide or prioritize: *"important for the PO to prioritize when it should be fixed"*
- Understand the impact on the business: *"This will show the impacts on business flow and help users to do the day-to-day business seamlessly"*

Those who strongly agreed that quantifying the Consequences of not fixing is important for informed decision-making gave the following reasons: to estimate product maintenance efforts, the effect of the use of numerical values, and to avoid work dependencies among teams. Some examples are as follows:

- Effect of using numerical values: *"Seeing a numerical value always makes things easier to decide"*
- Avoid dependencies among teams: *"other teams could be blocked by not fixing an issue"*

Those who strongly agreed that quantifying costs is not sufficient and that the benefits also must be quantified along with the costs gave reasons such as it is better to have more information and that the benefit is an important factor to decide or prioritize. Example quotes:

- Better to have more information: *"When cost is insufficient, the benefit must be quantified", "Definitely better to have more info"*
- Benefit is an important factor to decide or prioritize: *"the benefit is important to prioritize/deprioritize the fix"*

However, there were also some respondents who somewhat disagreed with this (nearly one-third of the respondents) but did not provide a reason for disagreeing.

Nearly one-third, 32.61% of the respondents strongly agreed that quantifying all three aspects, Cost of fixing, the Benefit of fixing, and the Consequences of not fixing, would be useful for informed decision-making. More than one-third of the respondents (39.13% of the respondents) somewhat agreed with the same. At the same time, some of them somewhat disagreed with this (13.04%).

Those who strongly agreed that quantifying the Consequences of not fixing alone is insufficient for making an informed decision and that the costs and benefits of fixing must also be quantified said that it is better to have more information to make decisions.

Those who somewhat disagreed with this gave reasons such as, not having expectations of severe consequences:

- No expectation of severe consequences: *"If no one expects any severe consequences an implmentation is good enough for now. This can still be later revisited when the consequences would change"*

In conclusion and according to Figure 2, the majority of the respondents generally strongly agreed or somewhat agreed with all the statements, while there were also neutral responses where the respondents did not agree or disagree with all the statements and a smaller proportion of the respondents somewhat disagreed with the statements. However, some respondents strongly disagreed that quantifying the consequences did not support informed decision-making (4.34%). Unfortunately, those who rated so did not provide any reasons for their strong disagreement.

5.2.2. *Tools, techniques, and other factors that could help (RQ2.2)*

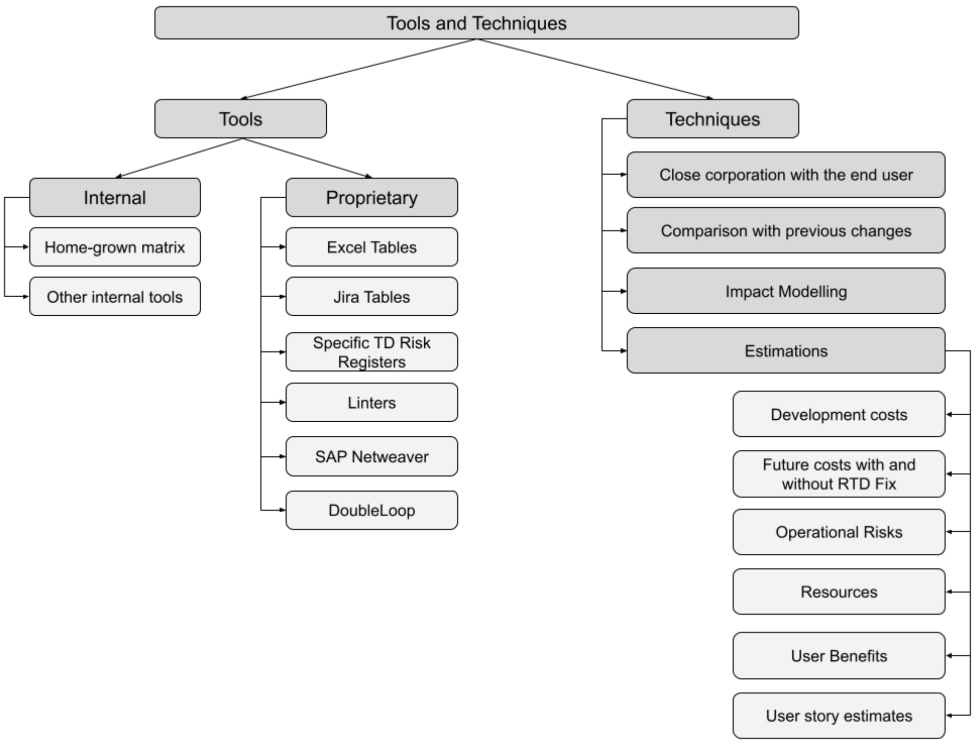


Figure 3: Tools and Techniques currently used in the Industry

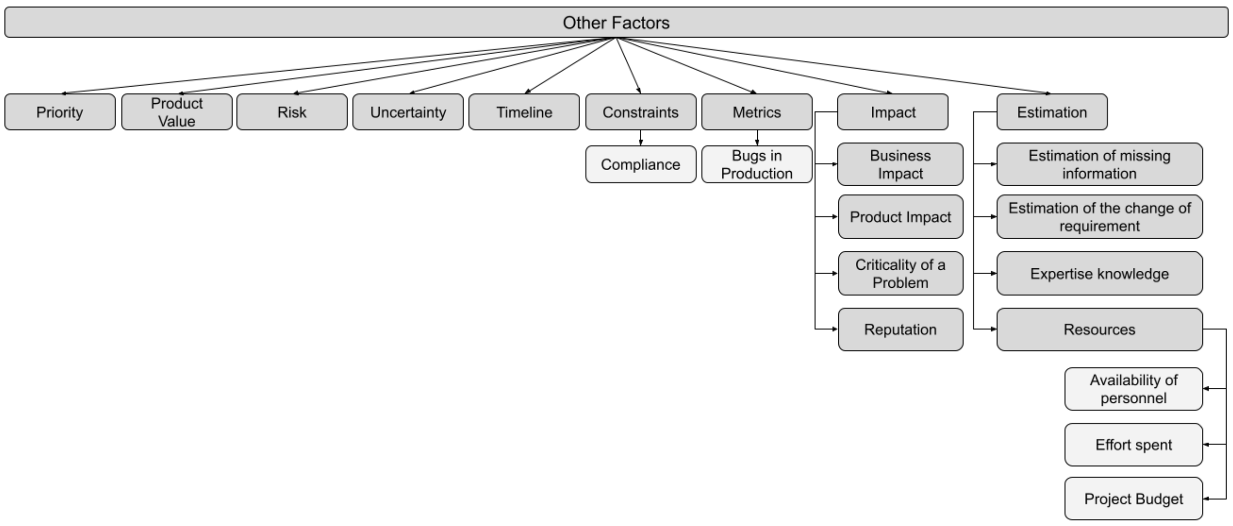


Figure 4: Other factors to consider in quantification

Insights from coding Open-Ended Questions. Figures 3 and 4 show insights gained by coding the answers to Q21 and Q22.

Figure 3 shows that existing tools in the industry could be categorized as internal and proprietary. Two examples of internal tools were mentioned by the survey respondents, home-grown metrics and other internal tools, while multiple proprietary tools were given as examples by the respondents. Proprietary tools included Excel tables, Jira tables, TD risk registers, Linters, SAP Netweaver, and DoubleLoop. DoubleLoop was used for Impact modeling, which is one of the techniques the respondents mentioned.

The four techniques that were identified as discussed by the practitioners were; Close cooperation with the end user, comparison with previous changes, impact modeling, and estimations. Practitioners estimated factors such as development costs, future costs with and without the RTD fix, operational risks, resources, user benefits, and user stories, apart from the costs, benefits, and consequences discussed in our survey.

Figure 4 shows that among other factors (apart from costs, benefits, and consequences) respondents thought would be useful to quantify were Priority, Product Value, Risk, Uncertainty, Timeline, Constraints, Metrics, Impact, and Estimations. According to the respondents, business impact, product impact, criticality of a problem, and reputation were considered important to quantify or measure in terms of impact. In terms of estimates, the missing information in the requirements, the change of a requirement, expert knowledge, and resources were considered important by the survey respondents. Some respondents mentioned 'late Bugs in production' as a metric to be considered. Compliance was categorized as a constraint based on the responses.

Figures 3 and 4 both point out that Estimations have been considered as a techniques as well as in other factors by our survey respondents.

Summary of findings (RQ2):

- **RQ2.1:** The ratings for the Likert Scale questions indicate agreement among practitioners that quantifying the Cost of fixing, the Benefit of fixing, the Consequences of not fixing, both costs and benefits, and all three (costs, benefits, and consequences) are useful for decision-making.
- **RQ2.2:** Both internal and proprietary tools were used in the industry. Four techniques were identified, which included close cooperation with the end-user. Respondents also emphasized the importance of estimations.

6. Current practices of quantifying RTD: Differences based on demographics and RTD instances

Findings discussed in Section 5 showed that most practitioners who responded to our survey fixed RTD during the implementation of software features. Most practitioners did not formally or informally quantify the cost to fix but rather quantified the consequences of not fixing and the benefit of fixing. Those who did not quantify said that they were not the decision-maker. We further analyzed quantification practices based on differences in demographics.

6.1. Quantification practices based on differences in demographics

6.1.1. Practitioner demographics

Figures 5, 6, and 7 illustrate the number of respondents who either formally or informally quantified RTD quantification concepts or did not at all quantify any concept based on their roles, work experience, and primary activities performed in their role, respectively.

According to Figure 5, the Software Engineer/ System Engineer/ Developer category has the highest response rate for those who quantified any concept. However, this is inevitable since the number of respondents who completed the survey was also highest for the same category. Only two respondents did not quantify any concept at all.

By examining the different concepts quantified by the different roles, we can observe that the diversity between the concepts quantified by the Software Engineer/ System Engineer/ Developer category of respondents is higher compared to categories such as the Project Manager/ Product Manager/ Team Lead, QA Manager/ QA Lead/ QA Engineer and the Systems Architect/ Software Architect/ Tech lead. The Software Engineer/ System Engineer/ Developer category quantified the Cost of fixing the least and the Consequences of not fixing the most. It is also noteworthy that

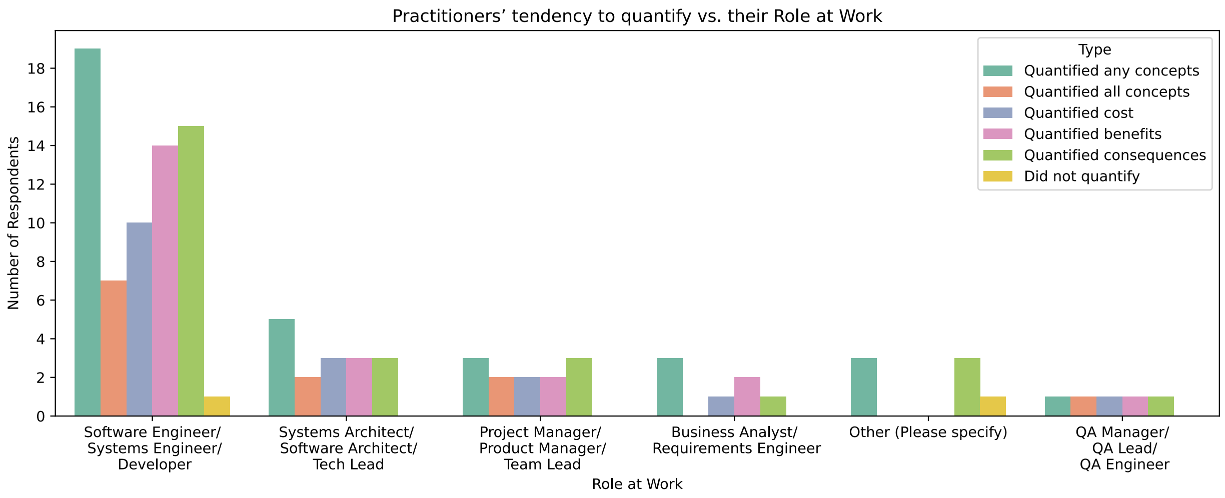


Figure 5: Practitioners' tendency to quantify vs. their Role at Work — A total of 35 respondents formally or informally quantified costs, benefits, or consequences, while only 2 did not quantify at all

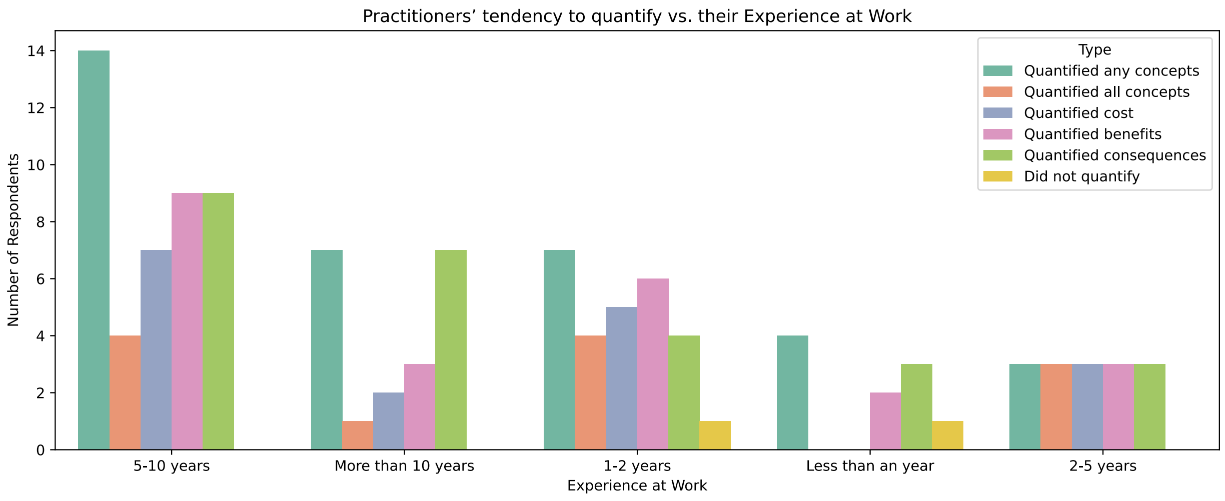


Figure 6: Practitioners' tendency to quantify vs. Work Experience — A total of 35 respondents formally or informally quantified costs, benefits, or consequences, while only 2 did not quantify at all

the manager roles such as Project Manager/ Product Manager/ Team Lead and QA Manager/ QA Lead/ QA Engineer categories seem to have more commonly quantified all concepts.

According to Figure 6, those who had 5-10 years of work experience had the highest response rate for quantifying any concept, which equals the number of respondents who belonged to that category of work experience. Respondents who did not quantify any concept at all had the least experience at work, i.e., less than two years of experience.

By examining the different concepts quantified by the different categories of work experience, we can observe that those who had two to five years of experience quantified all concepts equally. We can also see that the Consequences of not fixing RTD instances has been more commonly quantified by those who have more than five years of work experience. Those who had more than 10 years of work experience seem to quantify the consequences more frequently compared to the other concepts.

According to Figure 7, Implementing feature enhancements, fixing bugs, making improvements to the code (i.e.,

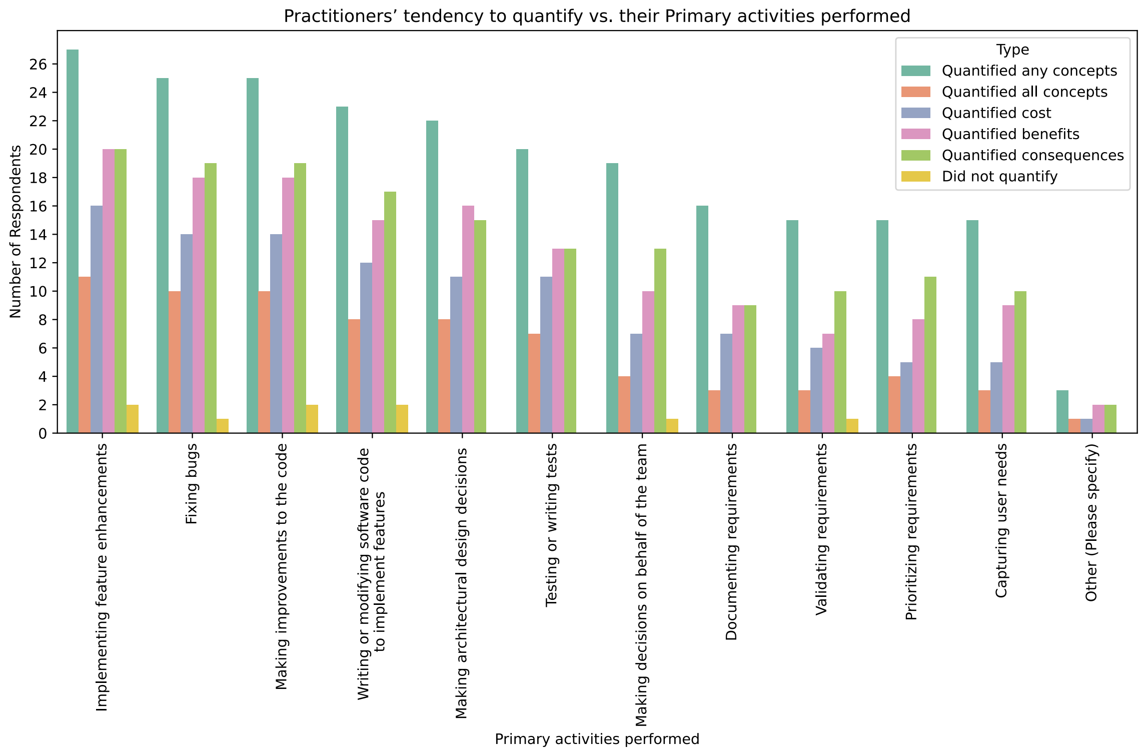


Figure 7: Practitioners' tendency to quantify vs. Primary activities they perform at Work — A total of 35 respondents formally or informally quantified costs, benefits, or consequences, while only 2 did not quantify at all

refactoring), and Writing or modifying software code to implement features were among the top four activities performed by those who mostly quantified Benefits of fixing or Consequences of not fixing. These activities were among the top four activities the majority of the survey respondents who completed the survey said they performed. However, it is noteworthy that all these activities are technical activities. Since the majority of the survey respondents also had technical roles such as Software Engineer/ Systems Engineer/ Developer and Software Architect/ Systems Architect/ Tech Lead, this result is not surprising.

For the activities, decision-making, validating requirements, and prioritizing requirements, the gap between those who quantified Consequences and those who quantified other concepts is higher compared to the other activities. This may indicate that quantification of consequences is more likely to be considered important by those who perform these activities.

In conclusion, we can derive a practitioner profile based on the tendency to quantify the most commonly quantified concept, that is, the Consequences of not fixing RTD instances.

The most common practitioner profile quantifying RTD: a Software Engineer/ Systems Engineer/ Developer or a Project Manager/ Product Manager/ Team lead with more than ten years of work experience engaged in activities such as decision-making, validating requirements, and prioritizing requirements.

6.1.2. Company demographics

Based on Figure 8, international SMEs and Domestic small businesses quantified all concepts equally. Large international businesses focused on quantifying the Benefits of fixing, while Large domestic businesses focused on

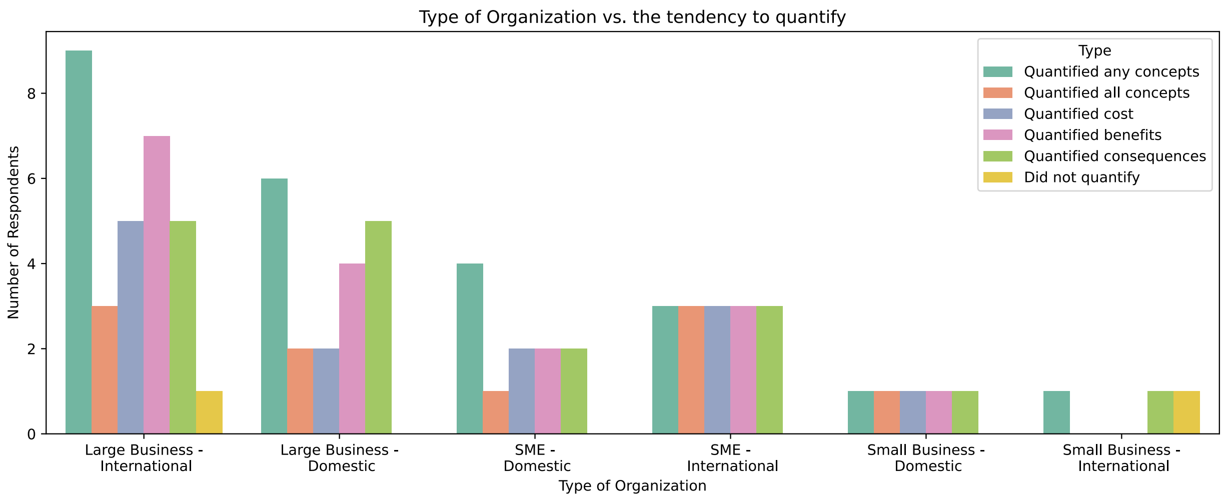


Figure 8: Type of Organization vs. the tendency to quantify — A total of 35 respondents formally or informally quantified costs, benefits, or consequences, while only 2 did not quantify at all

quantifying the Consequences of not fixing.

An interesting observation from Figure 8 is that the difference between those who quantified any concept and those who did not quantify at all was always positive or equal. Therefore, we can say that the encouragement for quantifying within all types of organizations is higher than for not doing so.

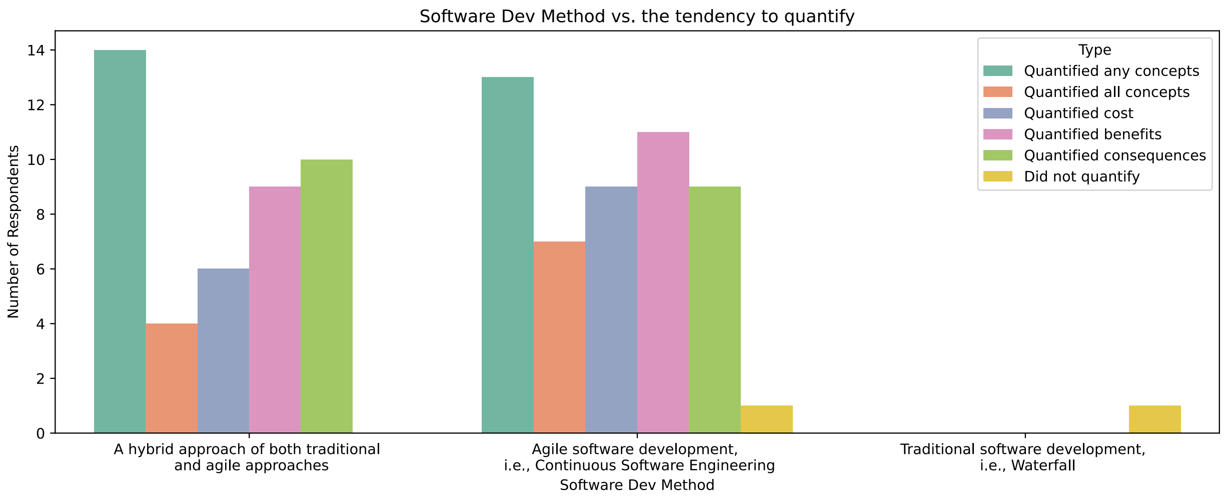


Figure 9: Software Development Methodology vs. the tendency to quantify — A total of 35 respondents formally or informally quantified costs, benefits, or consequences, while only 2 did not quantify at all

Based on Figure 9, those who followed a hybrid approach quantified Consequences or Benefits, while those who followed Agile approaches commonly quantified all concepts with a higher focus on Benefits.

Figure 10 shows that companies developing applications for Finance, Distribution/ Transportation/ Logistics, Energy, Telecommunication, Automotive, Manufacturing, and more focus on quantifying the Benefits of fixing RTD instances.

An interesting observation that could be made based on our data is that there were more respondents quantifying all concepts in the following application domains: Finance, Government Regulations/ Policies, Avionics, Audits and

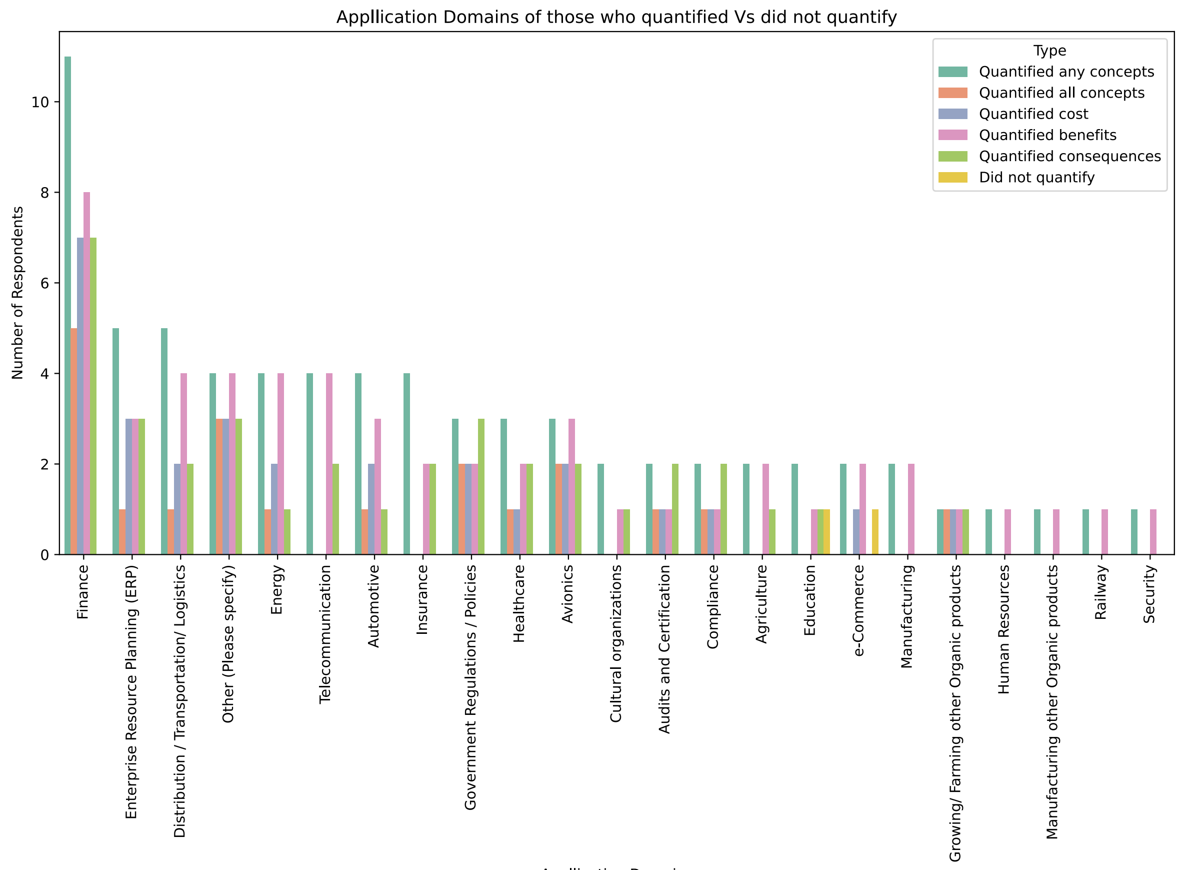


Figure 10: Application Domain vs. the tendency to quantify — A total of 35 respondents formally or informally quantified costs, benefits, or consequences, while only 2 did not quantify at all

Certification, Compliance, and Growing/Farming Organic Products.

In conclusion, we can derive a company profile based on the tendency to quantify RTD. The identified company profile mostly quantified the Benefits of fixing RTD instances.

The most common company profile quantifying RTD: a large international business that follows agile software development, and developing applications for domains such as finance, distribution/transportation/logistics, energy, telecommunication, automotive, and manufacturing.

6.2. *Quantification practices based on RTD instances*

The top three problems with requirements that had a significant impact (i.e., top three RTD instances) as described by participants in the critical incident were Changes in the requirements documentation, Lack of requirements validation with the user, and Insufficient documentation. We analyzed the RTD quantification concepts that were quantified in these instances.

According to Figure 11, all three concepts, Cost of fixing, Benefit of fixing, and Consequences (i.e., RTD Interest), were quantified in all three instances. The majority of the respondents have mostly quantified benefits. The quantification of consequences and costs follow.

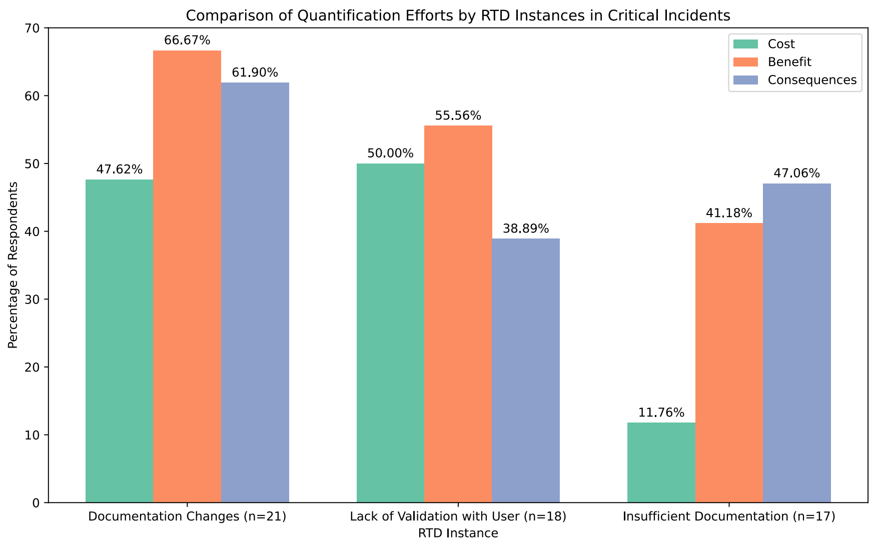


Figure 11: Comparison of quantification efforts for top three RTD instances fixed during the critical incident — Percentages are out of 21, 18, and 17 total respondents for Documentation Changes, Lack of Validation with User, and Insufficient Documentation, respectively.

For the RTD instance, Documentation changes, Benefits, and Consequences are quantified more frequently than Costs. According to this it seems that the decision to fix documentation changes is driven by the quantification of Benefits and Consequences more than the Cost of fixing.

For the RTD instance, Lack of validation with the user, Benefits, and Costs have been quantified more frequently than Consequences, indicating that Costs and Benefits are the decision-driving factors in this instance.

For the RTD instance, Insufficient documentation, the Consequences have been quantified by the majority of the respondents, followed by Benefits. Interestingly, only a very small number of respondents quantified the cost of fixing the instance of insufficient documentation. This implies that the cost of fixing insufficient documentation seems to be neglected during the decision-making process to fix such instances; rather, the Benefits and Consequences of not fixing them have been the decision-making factors.

Based on the top three RTD instances and the RTD quantification concepts quantified by the practitioners in these instances, we can conclude that the **choice of concepts to quantify has been different for the different RTD instances**.

7. RTD quantification for informed decision-making: Agreement vs Practices

Findings discussed in Section 5 showed that practitioners generally agree that quantifying the Cost of fixing, the Benefit of fixing, the Consequences of not fixing, the combination of both costs and benefits, and the combination of all three (costs, benefits, and consequences) can be useful for decision-making. We further analyzed the agreement and the actual quantification done in practice.

7.1. Comparison between agreement about support for decision-making and quantification practices

The left sub-figure of Figure 12 illustrates those who agreed that quantification of Cost, Benefit, and Consequences supports decision-making and those who did not agree so. In contrast, the right sub-figure illustrates those who quantified and those who did not quantify the Cost, Benefit, and Consequences in practice.

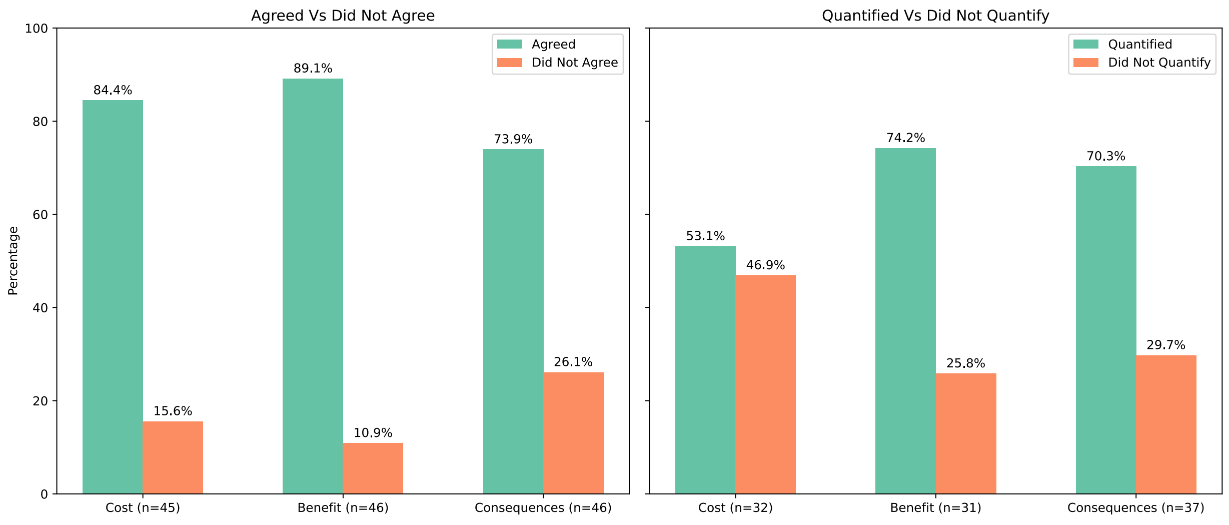


Figure 12: Those who Agreed Vs Did not Agree and Those who Quantified Vs Did not Quantify — Left: Percentages are out of 45, 46, and 46 total respondents for Cost, Benefit, and Consequences, respectively — Right: Percentages are out of 32, 31, and 37 total respondents for Cost, Benefit, and Consequences, respectively.

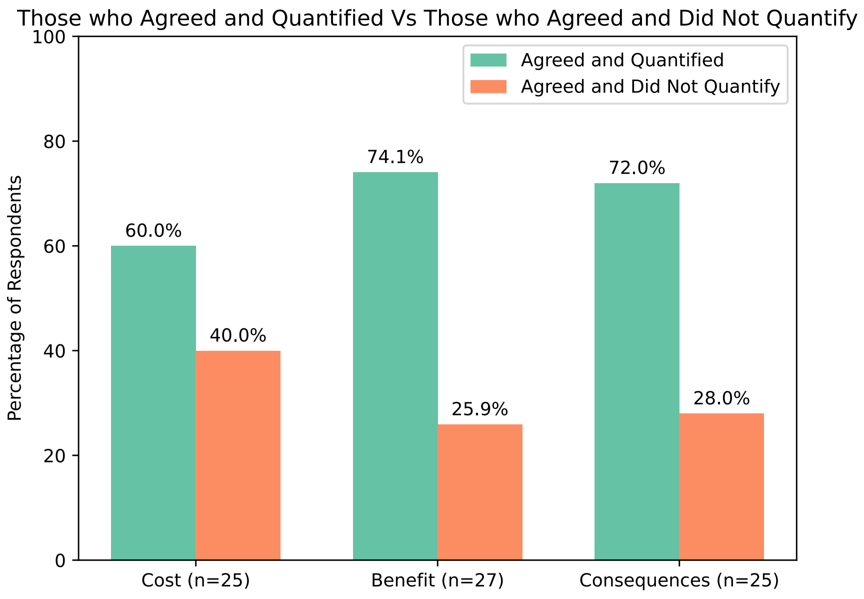


Figure 13: Those who Agreed and Quantified Vs Those who Agreed and Did not Quantify — Percentages are out of 25, 27, and 25 total respondents for Cost, Benefit, and Consequences, respectively.

By "agreed", we mean those who chose the options *'Strongly Agree,' 'Somewhat Agree.'* By "did not agree" we mean, those who chose the options *'Neither Agree nor Disagree', 'Somewhat Disagree', 'Strongly Disagree'*. We considered both options *'formally' and 'informally' quantified* in the category 'quantified'.

According to the left sub-figure in Figure 12, for all three categories, Cost, Benefit, and Consequences, the majority of the respondents agreed that quantification of these concepts supports decision-making for RTD management. Similarly, according to the right sub-figure in Figure 12, the majority of the respondents quantified the Cost, Benefit, and Consequences rather than not quantifying. However, the difference between those who quantified and those who did not quantify is less compared to the difference between those who agreed and those who did not agree, especially for Costs.

Figure 13 illustrates those who quantified Cost, Benefit, and Consequences in practice *while they agreed* that quantification supports decision-making and those who did not quantify in practice while they agreed so. The majority of those who agreed also quantified the RTD quantification concepts in practice. Among those who agreed and quantified, the most prominent concept agreed on and quantified in practice was the Benefit of fixing. Consequences of not fixing and the Cost of fixing followed. Among those who did not quantify, while they agreed that quantification supports decision-making, Cost was the most prominent concept. This implies that there have been respondents who did not actually quantify but who would have preferred to quantify the Cost of fixing.

Among those who agreed and quantified, quantifying the Benefits and Consequences were the most popular cases. **The most prominent concept agreed on and quantified in practice was the Benefit of fixing.** There have been respondents who did not quantify the Cost of fixing but who would have preferred to quantify it. However, the percentage of those who agreed that quantifying the cost could support decision-making was still less than those who agreed about the benefit.

8. Discussion

8.1. Revisiting the Research Questions

Regarding RTD quantification practices (RQ1), our aim was to know how often practitioners fixed RTD instances (RQ1.1), whether practitioners formally (e.g., in terms of person-hours, story points or dollars) or informally (e.g., had some expectation of extra costs in terms of effort spent) quantified RTD in practice, if so which concepts of RTD quantification were quantified (RQ1.2) and if practitioners did not quantify, what were the reasons (RQ1.3).

According to our findings, *RTD is fixed mostly during the implementation of software features*, regardless of whether it is in general or during a critical incident which implies that RTD is not given special attention but handled when it must be attended to. However, this could be due to the practitioners not being aware enough of the consequences of RTD early on or due to making a conscious decision to tackle it during implementation. The more likely case is that practitioners are unaware of the consequences at the beginning of the project.

Most practitioners did not formally or informally quantify the Cost of fixing problems with requirements leading to RTD instances. However, they *informally quantified the Benefit of fixing and formally quantified the Consequences of not fixing*. This again supports the claim that most probably practitioners are unaware of the consequences at the beginning of the project and would make an effort to fix the issues only when the consequences are known and if there is a benefit to fixing them. The most commonly experienced consequences were, according to our model concepts, Rework Code Costs, Rework Design Costs, and New Code Costs. This implies that *the consequences of RTD are mostly related to downstream activities such as design and implementation*. Therefore, it is worthwhile to investigate if the downstream impact of RTD could be predicted early on.

Among the reasons for not quantifying, the most common one was that the practitioner answering the survey was *not the decision-maker*, therefore, they did not see the need to quantify. Based on practitioner demographics, the observation was that those who were performing activities such as decision-making, validating requirements, and prioritizing requirements were more likely to quantify the consequences of not fixing the RTD than the other concepts.

Regarding how RTD quantification could support informed decision-making for managing RTD (RQ2), survey findings showed that practitioners generally agree that quantifying the model concepts (RQ2.1) Cost of fixing, the Benefit of fixing, and the Consequences of not fixing supported informed decision-making. The practitioners also generally agreed that the combination of both the Cost of fixing and the Benefit of fixing, and the combination of all three model concepts can be useful for decision-making. The common reasoning for the choice of a combination of the concepts was the perception that having more information is better.

Those who strongly agreed that quantifying the Cost to fix was important for decision-making wanted to avoid unnecessary expenses and considered the cost as an important factor to evaluate. Those who strongly agreed about quantifying the Benefit of fixing stated that the Benefit is an important factor in deciding to prioritize. Those who strongly agreed that quantifying the Consequences of not fixing is important to be quantified for decision-making wanted to estimate product maintenance efforts and avoid work dependencies among teams.

However, the *most prominent concept agreed on and actually quantified in practice was the Benefit of fixing*. Hence, it is worthwhile to investigate whether this stays true for another sample of practitioners in the industry and the reasons for this concept to be the prominent one. Even though we saw differences in the concepts preferred to be quantified were different *for the different RTD instances, the Benefit of fixing was still more commonly quantified*. Hence, it is worth to further investigate it with respect to different RTD instances and other decision-making scenarios.

Industry tools (RQ2.2) that help with the quantification, according to the practitioners who answered our survey included both internal and proprietary tools. Among the techniques (RQ2.2) identified as utilized by the practitioners, close cooperation with the end user emphasized *the need to have the end user in the loop*, while other techniques, such as comparison with previous changes, impact modelling, and estimations, highlighted the tools' focus on quantifying the consequences of RTD. This raises the question, *whether existing tools support industry needs in terms of what to quantify*.

Our previous work (the conceptual model discussed briefly in Section 2.1), captured some of the other important factors to quantify (RQ2.2) that were proposed by the practitioners without exposure to our model. For example, development costs and future costs with or without the RTD fix can be calculated by model concepts such as the *Development path cost* or by combining model concepts such as *RTD Interest*, *Implementation cost*, and *Development Path Cost*. User story estimates proposed by the practitioners can be considered similar to the Implementation Cost discussed in our model Perera et al. (2024a). Hence, these model concepts can be considered as validated through this practitioner survey. However, Resources and User Benefits were not explicit concepts in our model. But since resources can be interpreted in terms of time or effort to fix or person-hours, they could be considered implementation or development costs which are concepts captured in our model. User Benefits can be quantified by the End User Satisfaction Level, another concept in our model. We encourage researches and practitioners to *incorporate those model concepts in developing tools and techniques that could support RTD quantification*.

8.2. Implications and Future Research Directions

8.2.1. Difference in practitioner and company demographics for quantifying RTD

Among the practitioners, those who mostly quantified had roles such as engineers or developers. However, the diversity of the concepts quantified was also high in that category (quantified in the order of consequences, benefits, costs and all concepts) compared to more business-oriented and leadership roles, such as project managers, who tend to quantify all three concepts, Cost to fix, Benefit of fixing and the Consequences of not fixing more or less equally.

Those who generally had more than ten years of work experience seem to quantify the Consequences of not fixing more frequently. This implies that in more senior roles, they may take action based on the impact of not taking action. Another finding was that quantification of Consequences is more likely to be considered important by those who performed activities such as decision-making, validating requirements and prioritizing requirements.

Quantification was encouraged by most companies than not doing so. Companies who followed agile practices quantified all concepts with a higher focus on Benefits. Companies in the domains such as finance, government regulations or policies, avionics, audits and certification or compliance quantified all concepts, indicating that domains associated with high risk seem to quantify all concepts for decision-making.

Based on our data, we could identify a practitioner profile that mostly quantified consequences and a company profile that mostly quantified benefits. The observations made based on the most common practitioner profile and the most common company profile indicate that the company and the individual interests in quantifying RTD concepts differ. The company focuses on the benefits while the individuals focus on the consequences of not fixing or in other words, the consequences of not taking action. Therefore, it is worthwhile to investigate further whether individual and company perspectives differ for another sample set of practitioners.

8.2.2. Choice of concepts to quantify for different RTD instances

Based on the top three RTD instances (i.e., changes in the requirements documentation, lack of requirements validation with the end user and insufficient documentation) and the RTD quantification concepts quantified by the practitioners in those instances, we can conclude that the choice of concepts to quantify has been different for the different RTD instances. Therefore, it is worthwhile to invest in deeper conversations with practitioners to better understand their quantification behaviors related to different RTD instances and different scenarios of decision-making. This will help better understand what concepts can help practitioners make informed decisions in specific situations and thereby propose quantification approaches that support the quantification of the concepts fit for purpose. The RTD Quantification model in our previous work (Perera et al., 2024a) and the findings from this survey can provide a starting point.

8.2.3. Agreement vs practices of RTD quantification

Among those who agreed that quantification of an RTD quantification concept can support decision-making and, at the same time, quantified the concept in practice, Benefits and Consequences were the most popular concepts quantified. However, there were respondents who did not actually quantify the Cost of fixing but who would have preferred to quantify the Cost of fixing. But, the percentage of those who agreed that quantifying the cost could support decision-making was still less than those who agreed about the benefit. This is an interesting avenue to pursue, to learn whether practitioners' preferences differ from what they actually do in practice and whether addressing that gap would help in better decision-making.

8.2.4. Efficacy of tools and techniques used in the industry

Both internal and proprietary tools were used in the industry, while practitioners also seem to use techniques such as close cooperation with the end user, comparison with previous changes, impact modeling, and estimations. Estimations proposed as useful to be done by the practitioners included development costs, future costs with and without a fix for RTD, risks, user benefits, resources, and impact. Furthermore, concepts such as development costs and future costs with and without a fix for RTD were found to be considered by practitioners. The research direction that could be taken here is to investigate the efficacy of existing tools and techniques. The tools and techniques mentioned by the practitioners who responded to this survey serves as a starting point. However, deeper conversations with practitioners may reveal more information.

8.2.5. Other factors to consider for quantification and decision-making

The survey results highlighted the importance of quantification and estimations. This also included other factors to consider apart from the cost and benefit of fixing and consequences of not fixing such as, development costs, future costs with and without a fix for RTD, risks, user benefits, resources, and impact need to be further investigated. However, some of these factors (e.g., impact) could still be related to the same concepts we discussed in this survey, while some factors could be related to concepts not used in this survey but captured in our RTD Quantification Model (RTDQM) (Perera et al., 2024a). It will be worthwhile to investigate especially whether these factors are important to be considered for the decision-making scenario of fixing RTD (i.e. repayment) or whether they would be helpful in other scenarios such as prevention or prioritization.

8.2.6. The cascading impact of RTD

It was evident from the top three most common consequences practitioners mentioned that RTD has a cascading impact on later stages of software development, such as design and implementation. Furthermore, our survey findings indicated that practitioners fixed the RTD instances they encountered mostly during the implementation of software features. Therefore, it is worthwhile to investigate whether the impact of RTD on the downstream activities could be identified and predicted early on so that software development teams are more aware of the consequences of RTD and would be able to take action early enough to reduce the associated costs and risks.

9. Threats to validity

This section discusses the limitations and threats to the validity of this study. According to the guidelines for conducting surveys in software engineering provided by Linåker et al. (2015), validity and reliability are two main concerns in evaluating the threats to the validity of surveys. According to Kitchenham and Pfleeger (2008), obtaining a valid sample and construct validity are considered important for validity, while Linåker et al. (2015), state that there needs to be a certain level of reliability in order to make any final claims from the results of a survey. Therefore, we evaluate these with respect to our study below.

9.1. *Internal Validity*

Obtaining valid data depends on sampling. We defined a target population for our study. — *"The target population is the group or the individuals to whom the survey applies"* (Kitchenham and Pfleeger, 2008). In our case, this is software practitioners who are engaged in Requirements Engineering (RE) or software implementation activities (e.g., business analysts, requirements engineers, software engineers, software developers, architects, and project managers) in the industry. However, they had to be proficient in English since the researchers could conduct the survey only in English. See the participant inclusion criteria in Section 3.2.2.

"A valid sample is a representative subset of the target population" according to (Kitchenham and Pfleeger, 2008). Therefore, we must ensure that our sample is representative. In our case, we used *Convenience sampling* and *Snowball sampling* for obtaining our sample of participants (Linåker et al., 2015; Ralph et al., 2020). This means that our participants were those who willingly chose to participate in our study, and they could forward the survey invitation to their connections who also volunteered to participate in the study. These methods of sampling were chosen due to ethical concerns, including all participants who fit within the inclusion criteria and are willing to participate in our study. Furthermore, these two chosen methods of sampling allow us to gather more participants at a lower cost, e.g., by advertising the study via social media channels such as LinkedIn and Twitter. Out of the responses that we considered valid responses (52) for our survey, some of the respondents did not mention their country of work since the survey question was optional to respond to. However, the number of respondents who did mention a country was still higher than those who did not (61.52%). Similarly, the unknown percentage for the role at work was only 5.77%. A percentage of 94.23% specified roles that were representative of our target population. However, most of our participants were in technical roles than non-technical roles (more than 50%). Therefore, our findings may be more biased towards the perceptions of practitioners in technical roles.

9.2. *Construct Validity*

Construct validity refers to how well an instrument measures the construct it is designed to measure. This means, in our case, it refers to how well our survey questions actually measure the construct that was intended to be measured by the designer of the question. Our survey instrument was designed based on our conceptual model, but we carefully worded the questions so as not to use technical terms. Instead, we used terms that were more understandable to the participants. Which survey questions were informed by which RTDQM concepts can be found in Table A.10. For example, the survey question *Q13: Were there consequences (or negative impacts) in terms of extra efforts or costs later in the project due to not fixing or delaying to fix those problems with requirements? Select all that apply.*, was informed by the RTD Interest constituents in RTDQM. The following option values were provided in the survey question: *Yes, there were rework design costs, e.g., redesigning the software architecture, Yes, there were new design costs, e.g., adding new components to the software architecture, Yes, there were rework code costs, e.g., refactoring software code, Yes, there were new code costs, e.g., writing new software code, Yes, there were rework Requirements Engineering (RE) costs, e.g., refining the System Requirements Specification (SRS), Yes, there were new Requirements Engineering (RE) costs, e.g., having to conduct new user interviews, Other (Please specify), No, there were no consequences*. The options were clearly explained with examples, so the respondent was well informed. Furthermore, we used the terms measure and estimate, apart from quantification, to make sure that we describe the questions in a way that they are understood, even if the participants used different terms to mean the same thing. An example survey question where these terms were used is Q11. Table A.10 shows a simplified version of the survey questions. Furthermore, an introduction screen was displayed at the beginning of the survey section so that the respondents would be familiar with the phenomenon of RTD to make sure that they interpreted the survey questions accurately. The complete questionnaire can be found in our Replication Package[4]. Given the efforts we have made, we believe that the threat to construct validity is sufficiently mitigated.

9.3. *External Validity*

External validity refers to whether the findings and conclusions of our study can be generalized to the target population (Wright et al., 2010; Ralph et al., 2020). The number of respondents who answered each question in

[4]Replication Package: `https://doi.org/10.5281/zenodo.14172401`

our survey is different per question due to two reasons: all survey questions were optional to respond to, and some questions were displayed to a respondent based on a pre-requisite question that they answered. However, we analyzed the results of our questions based on the number of respondents who answered them. We mentioned the number of respondents who answered the question in the figures corresponding to the results of the survey question. We acknowledge that one could conclude that the survey results may not be generalizable globally since most of our respondents are from New Zealand, Sri Lanka, Sweden and Germany. But our dataset comprises respondents from different company sizes following mostly a hybrid process model but also other process models, having expert to novice levels of experience, and in different roles but mostly in technical roles. Hence, our dataset represents various software development contexts. Therefore, we think that having more responses from other countries will not be likely to change the conclusions made from our results. However, we acknowledge that our sample may not be representative of "all of the software practitioner population worldwide" despite having practitioners who had various roles and came from different sizes of companies, domains, and countries since most of our survey participants had technical roles. Our findings and conclusions may be more generalizable to a population of technical stakeholders rather than to both technical and non-technical stakeholders.

10. Conclusion and Future Work

Our work presents the first study to survey software practitioners on the quantification of RTD. We found that practitioners addressed RTD during feature implementation, suggesting RTD instances are handled later in development. Practitioners found that model concepts were useful for decision-making when quantified, with the Benefit of fixing RTD being the most commonly agreed upon and quantified in practice. Differences in individual vs. company perspectives emerged — companies focused on benefits while individuals focused on consequences. Furthermore, the choice of concepts to quantify varied for different RTD instances fixed during critical incidents. While both proprietary and internal tools are used, their effectiveness in supporting quantification remains unclear. Practitioners also suggested other factors for decision-making apart from the costs, benefits, and consequences discussed in our survey. However, they were captured in the Requirments Technical Debt Quantification Model (RTDQM) (Perera et al., 2024a). Future work will explore tool efficacy, individual vs. company perspectives, and RTD quantification across various instances and decision-making scenarios apart from repayment through practitioner interviews.

CRediT authorship contribution statement

Judith Perera: Writing – original draft, Writing – review & editing, Software, Data Curation, Methodology, Validation, Conceptualization, Investigation, Formal analysis, Visualization, Project administration, Funding Acquisition. **Ewan Tempero**: Writing – review & editing, Supervision, Methodology, Conceptualization, Project administration. **Yu-Cheng Tu**: Writing – review & editing, Supervision. **Kelly Blincoe**: Writing – review & editing, Supervision, Methodology, Resources, Project administration.

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

The source code and evaluation results are available at `https://doi.org/10.5281/zenodo.14172401`.

Acknowledgements

This work was supported by the New Zealand Ministry of Business, Innovation, and Employment via The Science for Technological Innovation (SfTI) National Science Challenge Veracity Technology Spearhead [2021-SfTI-SpH10-VUW].

References

Abad, Z.S.H., Ruhe, G., 2015. Using real options to manage technical debt in requirements engineering, in: 2015 IEEE 23rd International Requirements Engineering Conference, IEEE. pp. 230–235.

Avgeriou, P., Kruchten, P., Ozkaya, I., Seaman, C., 2016. Managing Technical Debt in Software Engineering (Dagstuhl Seminar 16162). Dagstuhl Reports 6, 110–138. URL: `http://www.dagstuhl.de/16162{\%}0Ahttp://drops.dagstuhl.de/opus/volltexte/2016/6693{\%}0Ahttp://www.dagstuhl.de/16162`.

Barbosa, L., Freire, S., Rios, N., Ramač, R., Taušan, N., Pérez, B., Castellanos, C., Correal, D., Pacheco, A., López, G., et al., 2022. Organizing the td management landscape for requirements and requirements documentation debt. UMBC Faculty Collection .

Behutiye, W., Rodríguez, P., Oivo, M., Aaramaa, S., Partanen, J., Abhervé, A., 2022. Towards optimal quality requirement documentation in agile software development: A multiple case study. Journal of Systems and Software 183, 111112.

Besker, T., Martini, A., Bosch, J., 2017. The pricey bill of technical debt: When and by whom will it be paid?, in: 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME), IEEE. pp. 13–23.

Bonfim, V.D., Benitti, F.B.V., 2022. Requirements debt: causes, consequences, and mitigating practices., in: SEKE, pp. 13–18.

Braun, V., Clarke, V., 2006. Using thematic analysis in psychology. Qualitative research in psychology 3, 77–101.

Braun, V., Clarke, V., 2012. Thematic analysis. American Psychological Association.

Charalampidou, S., Ampatzoglou, A., Chatzigeorgiou, A., Avgeriou, P., 2017. Assessing code smell interest probability: a case study, in: Proceedings of the XP2017 Scientific Workshops, pp. 1–8.

Ciolkowski, M., Laitenberger, O., Vegas, S., Biffl, S., 2003. Practical experiences in the design and conduct of surveys in empirical software engineering. Springer.

Codabux, Z., Williams, B.J., Bradshaw, G.L., Cantor, M., 2017. An empirical assessment of technical debt practices in industry. Journal of software: Evolution and Process 29, e1894.

Costa, A.F.F., Marques, A.B.D.S., Santos, I.S., Andrade, R.M.D.C., 2022. Towards a process to manage usability technical debts, in: Proceedings of the XXXVI Brazilian Symposium on Software Engineering, pp. 241–246.

Cunningham, W., 1992. The WyCash portfolio management system. Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA Part F1296, 29–30. doi:`10.1145/157709.157715`.

Ernst, N.A., 2012. On the role of requirements in understanding and managing technical debt, in: 2012 Third International Workshop on Managing Technical Debt, IEEE. pp. 61–64.

Fernández, D.M., Wagner, S., 2013. Naming the pain in requirements engineering: design of a global family of surveys and first results from germany, in: Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering, pp. 183–194.

Fernández, D.M., Wagner, S., Kalinowski, M., Felderer, M., Mafra, P., Vetrò, A., Conte, T., Christiansson, M.T., Greer, D., Lassenius, C., et al., 2017. Naming the pain in requirements engineering: Contemporary problems, causes, and effects in practice. Empirical software engineering 22, 2298–2338.

Flanagan, J.C., 1954. The critical incident technique. Psychological bulletin 51, 327.

Frattini, J., Fucci, D., Mendez, D., Spinola, R., Mandić, V., Taušan, N., Ahmad, M.O., Gonzalez-Huerta, J., 2023. An initial theory to understand and manage requirements engineering debt in practice. Information and Software Technology 159, 107201.

Freire, S., Rios, N., Gutierrez, B., Torres, D., Mendonça, M., Izurieta, C., Seaman, C., Spínola, R.O., 2020a. Surveying software practitioners on technical debt payment practices and reasons for not paying off debt items, in: Proceedings of the 24th International Conference on Evaluation and Assessment in Software Engineering, pp. 210–219.

Freire, S., Rios, N., Mendonça, M., Falessi, D., Seaman, C., Izurieta, C., Spínola, R.O., 2020b. Actions and impediments for technical debt prevention: results from a global family of industrial surveys, in: Proceedings of the 35th Annual ACM Symposium on Applied Computing, pp. 1548–1555.

Freire, S., Rios, N., Pérez, B., Castellanos, C., Correal, D., Ramač, R., Mandić, V., Taušan, N., López, G., Pacheco, A., et al., 2024. Hearing the voice of software practitioners on technical debt monitoring: Understanding monitoring practices and the practices' avoidance reasons. Journal of Software Engineering 12, 11.

Junior, H.J., Travassos, G.H., 2022. Consolidating a common perspective on technical debt and its management through a tertiary study. Information and Software Technology , 106964.

Kalinowski, M., Curty, P., Paes, A., Ferreira, A., Spínola, R., Fernández, D.M., Felderer, M., Wagner, S., 2017. Supporting defect causal analysis in practice with cross-company data on causes of requirements engineering problems, in: 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP), IEEE. pp. 223–232.

Karlsson, J., Ryan, K., 1997. A cost-value approach for prioritizing requirements. IEEE software 14, 67–74.

Kazman, R., Asundi, J., Klein, M., 2001. Quantifying the costs and benefits of architectural decisions, in: Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001, IEEE. pp. 297–306.

Kitchenham, B.A., Pfleeger, S.L., 2008. Personal opinion surveys, in: Guide to advanced empirical software engineering. Springer, pp. 63–92.

Lenarduzzi, V., Fucci, D., 2019. Towards a holistic definition of requirements debt, in: 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, IEEE. pp. 1–5.

Lenarduzzi, V., Fucci, D., Mendéz, D., 2020. On the perceived harmfulness of requirement smells: An empirical study, in: Joint 26th International Conference on Requirements Engineering: Foundation for Software Quality Workshops, Doctoral Symposium, Live Studies Track, and Poster Track, Pisa; Italy, CEUR-WS.

Li, Z., Avgeriou, P., Liang, P., 2015. A systematic mapping study on technical debt and its management. Journal of Systems and Software 101, 193–220. URL: `http://dx.doi.org/10.1016/j.jss.2014.12.027`, doi:`10.1016/j.jss.2014.12.027`.

Linåker, J., Sulaman, S.M., de Mello, R.M., Höst, M., 2015. Guidelines for conducting surveys in software engineering. Department of Computer Science, Lund University.

Maldonado, E.d.S., Shihab, E., 2015. Detecting and quantifying different types of self-admitted technical debt, in: 2015 IEEE 7Th international workshop on managing technical debt (MTD), IEEE. pp. 9–15.

Melo, A., Fagundes, R., Lenarduzzi, V., Santos, W.B., 2022. Identification and measurement of requirements technical debt in software development: A systematic literature review. Journal of Systems and Software , 111483.

Mendes, T.S., de F. Farias, M.A., Mendonça, M., Soares, H.F., Kalinowski, M., Spínola, R.O., 2016. Impacts of agile requirements documentation debt on software projects: a retrospective study, in: Proceedings of the 31st Annual ACM Symposium on Applied Computing, pp. 1290–1295.

Mendez, D., Avgeriou, P., Kalinowski, M., Ali, N.B., . Handbook on teaching empirical software engineering.

Ministry of Foreign Affairs and Trade, New Zealand, . New zealand trade and smes. `https://www.mfat.govt.nz/assets/Trade-agreements/UK-NZ-FTA/Trade-and-small-medium-enterprises-.pdf`. Accessed: 2023-09-30.

Nugroho, A., Visser, J., Kuipers, T., 2011. An empirical model of technical debt and interest, in: Proceedings of the 2nd workshop on managing technical debt, pp. 1–8.

Ojameruaye, B., Bahsoon, R., 2014. Systematic elaboration of compliance requirements using compliance debt and portfolio theory, in: Requirements Engineering: Foundation for Software Quality: 20th International Working Conference, REFSQ 2014, Essen, Germany, April 7-10, 2014. Proceedings 20, Springer. pp. 152–167.

Perera, J., Tempero, E., Tu, Y.C., Blincoe, K., 2023a. Quantifying requirements technical debt: A systematic mapping study and a conceptual model, in: 2023 IEEE 31st International Requirements Engineering Conference (RE), pp. 123–133. doi:`10.1109/RE57278.2023.00021`.

Perera, J., Tempero, E., Tu, Y.C., Blincoe, K., 2023b. Quantifying technical debt: A systematic mapping study and a conceptual model. arXiv preprint arXiv:2303.06535 doi:`https://doi.org/10.48550/arXiv.2303.06535`.

Perera, J., Tempero, E., Tu, Y.C., Blincoe, K., 2024a. Modelling the quantification of requirements technical debt. Requirements Engineering , 1–38URL: `https://doi.org/10.1007/s00766-024-00424-3`, doi:`10.1007/s00766-024-00424-3`.

Perera, J., Tempero, E., Tu, Y.C., Blincoe, K., 2024b. A systematic mapping study exploring quantification approaches to code, design, and architecture technical debt. ACM Transactions on Software Engineering and Methodology URL: `https://doi.org/10.1145/3675393`, doi:`10.1145/3675393`.

Ralph, P., Ali, N.b., Baltes, S., Bianculli, D., Diaz, J., Dittrich, Y., Ernst, N., Felderer, M., Feldt, R., Filieri, A., et al., 2020. Empirical standards for software engineering research. arXiv preprint arXiv:2010.03525 .

Rios, N., Mendes, L., Cerdeiral, C., Magalhães, A.P.F., Perez, B., Correal, D., Astudillo, H., Seaman, C., Izurieta, C., Santos, G., et al., 2020. Hearing the voice of software practitioners on causes, effects, and practices to deal with documentation debt, in: Requirements Engineering: Foundation for Software Quality: 26th International Working Conference, REFSQ 2020, Pisa, Italy, March 24–27, 2020, Proceedings 26, Springer. pp. 55–70.

Rios, N., Mendonça, M.G., Seaman, C., Spínola, R.O., 2019. Causes and effects of the presence of technical debt in agile software projects .

Sivzattian, S., Nuseibeh, B., 2001. Linking the selection of requirements to market value: A portfolio-based approach, in: Proceedings of 7th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ 2001).

Stol, K.J., Fitzgerald, B., 2020. Guidelines for conducting software engineering research, in: Contemporary Empirical Methods in Software Engineering. Springer, pp. 27–62.

Wagner, S., Fernández, D.M., Felderer, M., Kalinowski, M., 2017. Requirements engineering practice and problems in agile projects: results from an international survey. arXiv preprint arXiv:1703.08360 .

Wagner, S., Fernández, D.M., Felderer, M., Vetrò, A., Kalinowski, M., Wieringa, R., Pfahl, D., Conte, T., Christiansson, M.T., Greer, D., et al., 2019. Status quo in requirements engineering: A theory and a global family of surveys. ACM Transactions on Software Engineering and Methodology (TOSEM) 28, 1–48.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., et al., 2012. Experimentation in software engineering. volume 236. Springer.

Wright, H.K., Kim, M., Perry, D.E., 2010. Validity concerns in software engineering research, in: Proceedings of the FSE/SDP workshop on Future of software engineering research, pp. 411–414.

Appendix A. Survey Questions

Table A.10: Survey questions, Question types, and Model Concepts that guided the development of the survey questions — SC - Single Choice, MC - Multiple Choice — Survey sections 1, 4 - Demographics, Survey Section 2 - RTD Quantification

No.	Question	Type	Options Ref.	RQs	Survey Section	Model Concepts
Q1	Primary role in company	SC	Codabux et al. (2017)	-	1	-
Q2	Primary activities performed in the current role	MC	Perera et al. (2024a)	-	1	-
Q3	Years of experience in the current role	SC	-	-	1	-
Q41	Country of work	SC	-	-	4	-
Q42	Scale of the organization (domestic, international, Small, SMEs, Large)	SC	Ministry of Foreign Affairs and Trade, New Zealand	-	4	-
Q43	Primary application domain(s) of the products and services respondents work on	MC	Frattini et al. (2023)	-	4	-
Q44	Software development methodology practiced	SC	-	-	4	-
Q4	Problems with requirements encountered within the current role	MC	Perera et al. (2024a), Frattini et al. (2023), Lenarduzzi and Fucci (2019), Bonfim and Benitti (2022)	RQ1	2	RTD Item
Q5	Most recent incident that created a significant impact on the project (or company) due to problems with requirements	Open-ended	based on CIT	RQ1	2	RTD Item

Continued on next page

Table A.10 – continued from previous page

No.	Question	Type	Options Ref.	RQs	Survey Section	Model Concepts
Q6	Most significant problems with requirements that created a significant impact on the project (or company), based on the critical incident	MC	Perera et al. (2024a), Frattini et al. (2023), Lenarduzzi and Fucci (2019), Bonfim and Benitti (2022)	RQ1	2	RTD Item
Q7	Generally, how often did they fix problems with requirements	MC	Frattini et al. (2023)	RQ1.1	2	RTD Rectification Step
Q8	When did they fix the problems with requirements in the critical incident	SC	Frattini et al. (2023)	RQ1.1	2	RTD Rectification Step
Q9	When fixing, did they formally or informally quantify the effort or cost to fix	SC	Codabux et al. (2017)	RQ1.2	2	RTD Rectification Step, Cost of Rectifying RTD
Q10	Primary reason for not quantifying the effort or cost to fix	SC	Frattini et al. (2023)	RQ1.3	2	Cost of Rectifying RTD
Q11	When fixing, did you formally or informally quantify the benefit of fixing	SC	-	RQ1.2	2	RTD Rectification Step, Benefit of Rectifying RTD
Q12	Primary reason for not quantifying the benefit of fixing	SC	Frattini et al. (2023)	RQ1.3	2	Benefit of Rectifying RTD
Q13	Consequences (or negative impacts) in terms of extra efforts or costs due to NOT fixing or delaying to fix problems with requirements	MC	Perera et al. (2024a)	RQ1.2	2	RTD Interest and Interest constituents
Q14	Did you formally or informally quantify such consequences, i.e., extra efforts or costs incurred in the project, due to NOT fixing or delaying to fix	MC	Codabux et al. (2017)	RQ1.2	2	RTD Interest and Interest constituents
Q15	Primary reason for not quantifying the consequences of NOT fixing or delaying to fix	MC	Frattini et al. (2023)	RQ1.3	2	RTD Interest and Interest constituents
Q16	Quantifying the cost to fix supports making an informed decision if and when to fix	Likert Scale	-	RQ2.1	2	Cost of rectifying RTD
Q17	Quantifying the benefit of fixing supports making an informed decision if and when to fix	Likert Scale	-	RQ2.1	2	Benefit of rectifying RTD

Continued on next page

Table A.10 – continued from previous page

No.	Question	Type	Options Ref.	RQs	Survey Section	Model Concepts
Q18	Quantifying the consequences of NOT fixing or delaying to fix supports making an informed decision to fix early	Likert Scale	-	RQ2.1	2	RTD Interest and Interest constituents
Q19	Quantifying the cost to fix is insufficient for making an informed decision; the benefit also must be quantified	Likert Scale	-	RQ2.1	2	Cost of rectifying RTD, Benefit of rectifying RTD
Q20	Quantifying the consequences of not fixing is insufficient for making an informed decision; the costs and benefits of fixing must also be quantified.	Likert Scale	-	RQ2.1	2	Cost of rectifying RTD, Benefit of rectifying RTD, RTD Interest and Interest constituents
Q21	What tools or techniques may help in quantifying costs, benefits and consequences related to RTD	Open-ended	-	RQ2.2	2	-
Q22	What other factors apart from costs, benefits and consequences are useful to consider when deciding to fix problems with requirements	Open-ended	-	RQ2.2	2	-

Appendix B. Demographics of Survey Respondents

Appendix B.1. Country of work

Most respondents out of those who mentioned their country of work were from New Zealand (12 respondents out of a total of 32 respondents who mentioned the country of work, and a percentage of 23.08% out of 52 valid responses). Sri Lanka, Sweden, and Germany followed. See Figure B.14.

Appendix B.2. Type of company

Small to Medium Enterprises (SMEs) are firms with 6 to 49 employees, and large businesses are firms with 50+ employees[2]. We considered both domestic and international companies under this categorization. See Figure B.15. However, most participants chose not to reveal the type of company. From those who revealed this information, large international businesses were the most common type of company.

Appendix B.3. Role at work

We asked about the respondent's role at work. This allowed us to interpret whether they had a technical or non-technical role. See Figure B.16. The most common type of role of the respondents was technical; they were Software Engineers, Systems Engineers, Developers, System Architects, Software Architects, and Tech Leads, and they added up to 59.62% of the valid responses. 17.31% selected their role as 'Other.' Respondents were given the option to specify their role if they selected 'Other.' Roles specified by the respondents who selected 'Other' were: Director of Product, General Manager, Data Scientist, Machine Learning Engineer, Researcher, and PhD candidate. We assumed that those who said they were researchers chose to answer the survey since they had previous work experience but currently were pursuing research.

[2]New Zealand Trade and SMEs: `https://www.mfat.govt.nz/assets/Trade-agreements/UK-NZ-FTA/Trade-and-small-medium-enterprises-.pdf`

Appendix B.4. Experience at work

See Figure B.17. Most of the respondents who responded to the survey mentioned that they had 5-10 years of work experience (a percentage of 30.77% out of 52 valid responses). Those who had less than a year of work experience were a percentage of 13.46%, the least number of participants were in this category. Those who had more than 10 years of work experience were a percentage of 17.31%.

Appendix B.5. Software development methodology

We asked the participants about the software development methodology used in their current organization. See Figure B.18. According to respondents who chose to answer this question, a hybrid approach of Traditional and Agile approaches was the most common (a percentage of 34.62% out of 52 valid responses).

Appendix B.6. Application domains

We asked respondents about the primary application domain (s) of the products and services they worked on in their current company or organization. The most common application domain was Finance. Energy, Automotive, and 'Other' followed. See Figure B.19. The responses under 'Other' were AI Software Company, Real State, and Research.

Appendix B.7. Primary activities performed

Participants were also asked about the primary activities they performed in their current role. According to Figure B.20, 11.86% of the respondents were implementing feature enhancements, while a percentage of 11.22% respondents were making improvements to the code (or refactoring), 10.9% were writing or modifying software code to implement features that satisfy the requirements and 10.9% were fixing bugs. Therefore, the top four most common activities performed by the survey respondents were mostly associated with the software code.

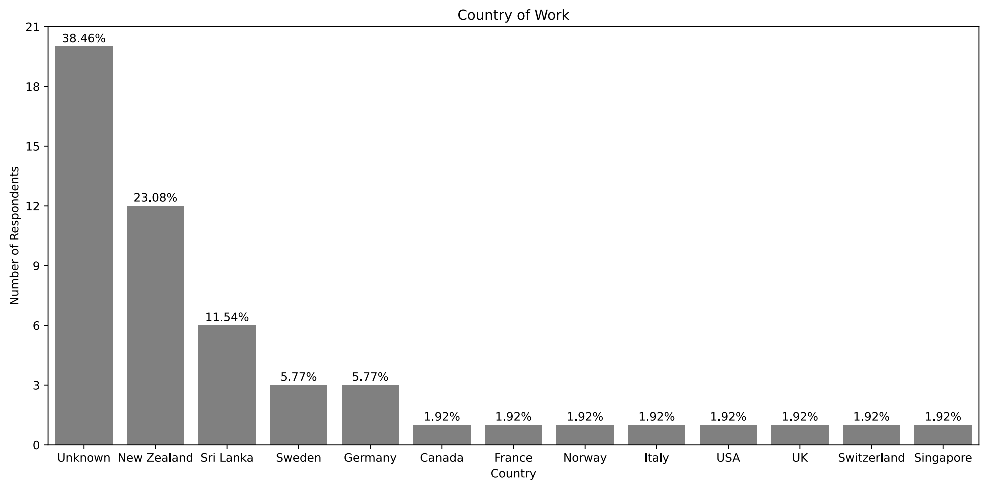


Figure B.14: Respondent's Country of Work

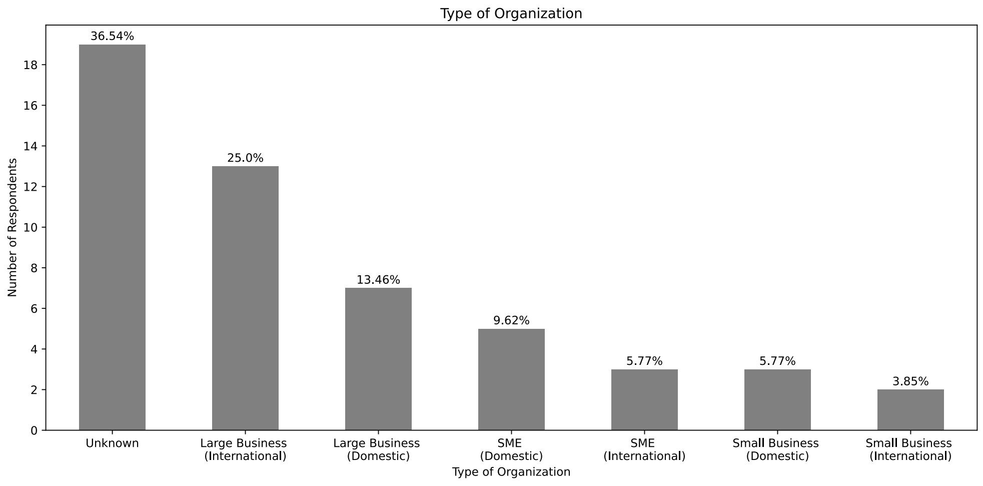


Figure B.15: Type of Company

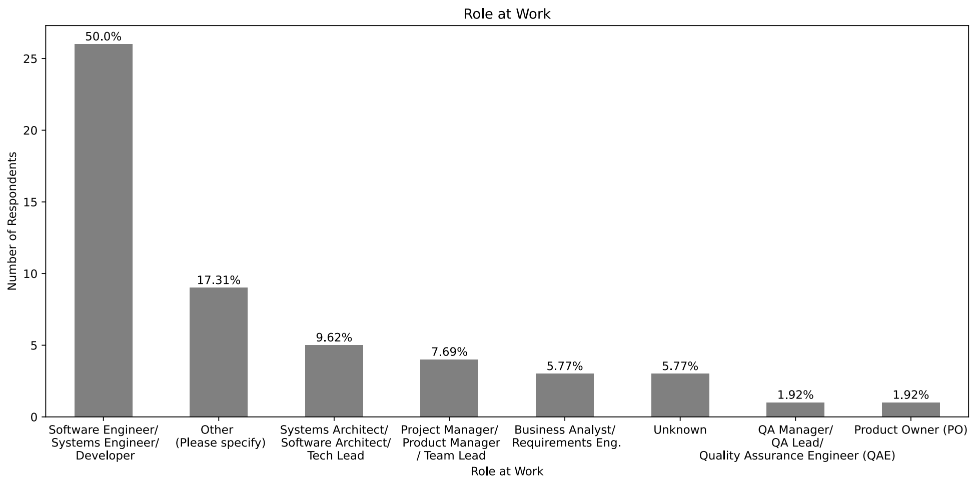


Figure B.16: Respondents' Role at Work

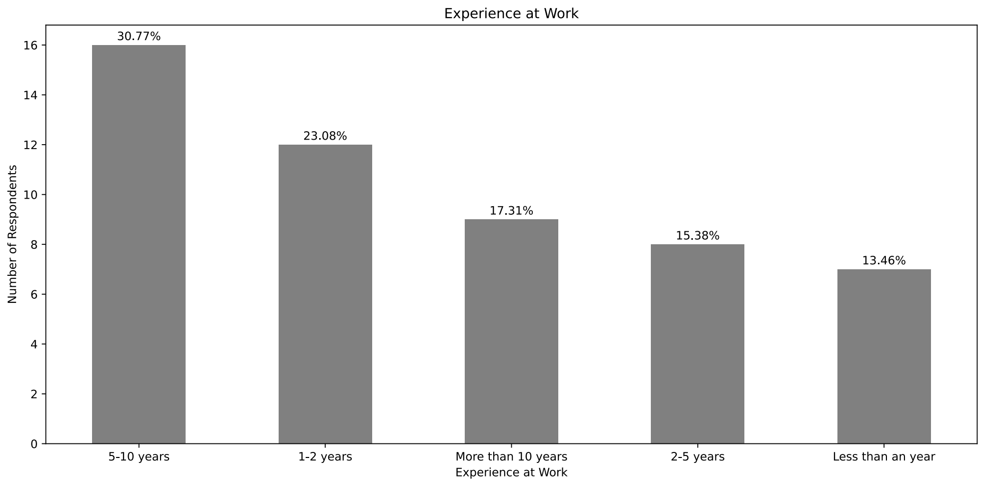


Figure B.17: Respondents' Experience at Work

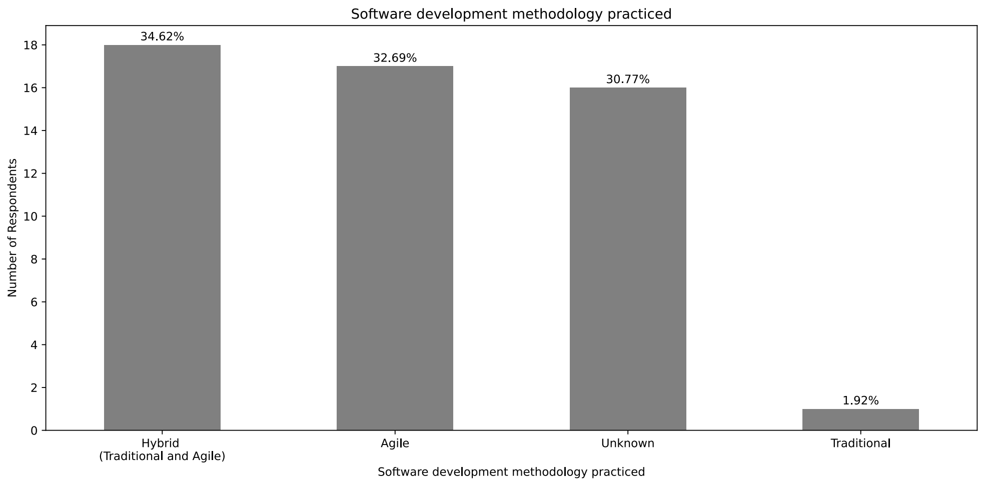


Figure B.18: Software Development Methodology

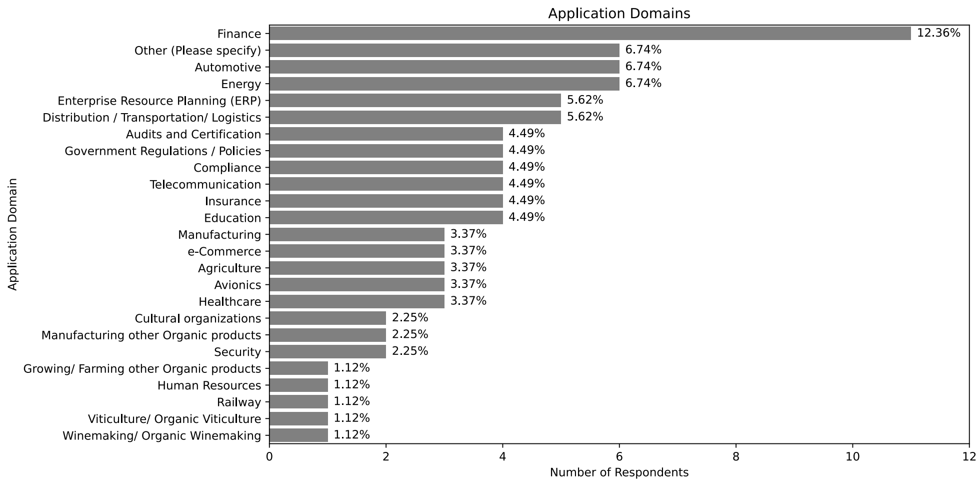


Figure B.19: Application Domains

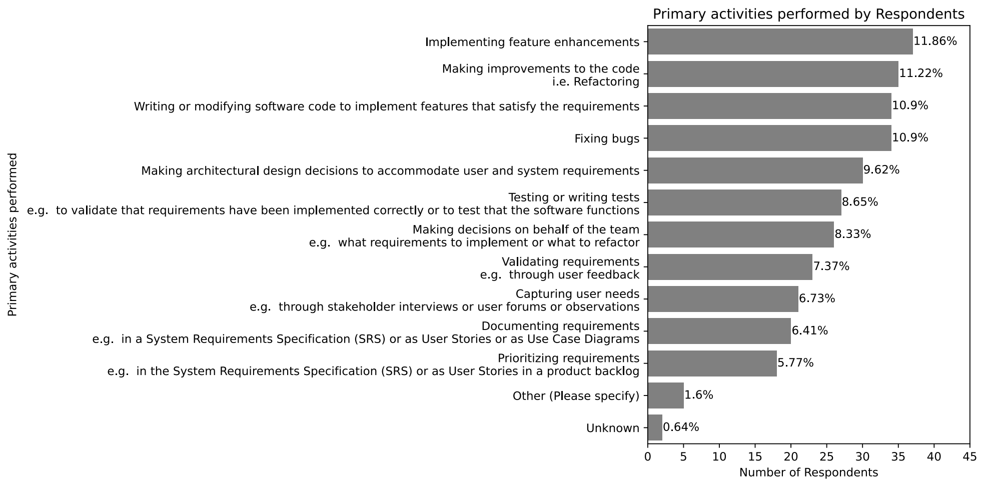


Figure B.20: Primary activities performed by the Respondents