Veracity Debt: Practitioners Voices on Managing Software Requirements concerning Veracity

Judith Perera[1][0000−0002−1092−4229], Ewan Tempero[1][0000−0002−3786−1707], Yu-Cheng Tu[1][0000−0001−7284−7081], and Kelly Blincoe[1][0000−0003−4092−9706]

[1] The University of Auckland, New Zealand
[2] jper120@aucklanduni.ac.nz
{e.tempero,yu-cheng.tu,k.blincoe}@auckland.ac.nz

Abstract. ***Context & Motivation:*** Veracity requirements address truth, trust, authenticity, and demonstrability across dimensions, data, financial, regulatory, process, and cultural in software and AI systems. ***Problem:*** Sub-optimal decisions made w.r.t. veracity requirements in the development of software could lead to adverse consequences such as huge cost overruns, legal concerns, bad reputation, and the loss of customers. This is captured by the phenomenon 'Veracity Debt'. This paper reports the results of an exploratory survey conducted with software practitioners to gather their perceptions on veracity requirements and quantifying veracity debt. ***Results:*** Results are promising since practitioners relate to the concepts discussed in the survey. Practitioners agreed that quantifying veracity debt aids informed decision-making and suggested considering more factors like legal liabilities, user feedback, team trust, reputation, and customer retention. ***Contribution:*** This study coins the term 'Veracity Debt' and presents practitioner insights on its impact and emphasizes the need for further research.

Keywords: Requirements Technical Debt · Veracity · Truth · Trust · Authenticity · Demonstrability · Regulatory · Data · Cultural · Process

1 Introduction

'Veracity' is a concept that becomes very relevant in the present-day. An everyday example is the veracity of online data, e.g., fake news, malicious rumors, fabricated reviews, fake images, and videos [12]. Applied to the context of software requirements, we define 'Veracity Requirements' as a specific category of software requirements *related to the notions of truth, trust, authenticity, and demonstrability* in software-intensive systems. We build this definition based on Luczak-Roesch et al.'s definition of veracity [13] that captures multiple dimensions of veracity without limiting it to data quality (their definition is described in Section 2). Veracity requirements can be functional or non-functional requirements, depending on the software system under development and the context (e.g., domain constraints, system environment, and business goals). Veracity requirements are becoming a concern, for example, in many software-intensive

systems and data infrastructures where attributes such as accuracy, precision, trust, and truthfulness are considered necessary qualities to have [3, 7]. We envision 'Veracity' as a software quality attribute to be met by software and AI products.

From a software quality perspective, we apply the concept of Requirements Technical Debt (RTD)—the consequences of sub-optimal decisions regarding requirements—to veracity requirements. These sub-optimal decisions, whether intentional or not, occur during requirement identification, documentation, implementation of features, or when making architectural choices. Examples include inadequate specification, ambiguities, defects (Requirement Smells), lack of end-user feedback, and incomplete implementation. We introduce the term 'Veracity Requirements TD' (or 'Veracity Debt') to describe the impact of such decisions on veracity requirements in software-intensive systems.

Previous stakeholder engagements revealed that veracity requirements span various dimensions, including *regulatory, data, financial data, cultural, and process.* Failing to meet these due to veracity debt can have adverse consequences. For instance, unmet regulatory requirements may result in legal issues, cost overruns, reputational damage, and customer loss. Similarly, inadequate implementation of data veracity requirements can lead to security and privacy risks. Moreover, neglecting cultural data sovereignty requirements may cause socio-cultural concerns, as seen in New Zealand. Effective veracity debt management can help software companies and stakeholders mitigate these risks.

This paper presents an exploratory survey of industry practitioners' perceptions of veracity requirements and veracity debt quantification for informed decision-making. Most practitioners agree that quantifying the consequences of unresolved veracity debt aids in addressing it. Key considerations include legal liabilities, user feedback, team trust, reduced trust, reputational impact, and customer loss. Additionally, practitioners perceive veracity debt as having a greater impact than general RTD.

We propose veracity as a multifaceted concept and a potential Software Quality Attribute for software and AI products, warranting further exploration. This study initiates a critical dialogue on its role in Requirements Engineering (RE) and lays the foundation for future research, including practitioner studies, to determine if veracity merits recognition as a distinct quality attribute in standards like ISO/IEC 25010.

The paper is structured as follows. Section 2 provides a background. Sections 3 and 4 report on the methodology and results, respectively. Section 5 discusses findings, implications to researchers and practitioners, and future work. Threats to validity are discussed in Section 6. The paper is concluded in Section 7.

2 Background and Related Work

To the best of our knowledge, this is the first study to apply veracity to software requirements, introduce the term 'Veracity Debt', and present practitioner insights. Below, we outline the background supporting this work.

Definitions of Veracity Lozano et al. [8] highlight that 'Veracity' lacks a consensus definition in the literature. They describe it in the context of big data using terms such as truth, trust, uncertainty, credibility, reliability, noise, anomalies, imprecision, and quality. Luczak-Roesch et al. [13], in their work on the Veracity Project initiated by the New Zealand government in 2021, argue that veracity should not be limited to data quality. Instead, it must encompass multi-directional considerations, including artifact and data sovereignty, emphasizing that digital artifacts are inseparable from their real-world context. They define veracity broadly as *"truth, trust, authenticity, and demonstrability"*, the definition we adopted in this paper. The Veracity Project also identified various dimensions of veracity, such as regulatory, data, financial, cultural, and process, which we included as options in our survey.

Veracity in Software and AI Products Galster and Dietrich [7] explore retrofitting provenance (metadata about the origin, context, or history of data) into existing software systems to address questions about the collection and usage of data in software systems. Through industry interviews, they developed an influence map describing three dimensions of provenance: When, what, and how. Provenance aligns with *demonstrability and authenticity*, key aspects of the veracity definition used in this work.

Blincoe et al. [3] identify **trust** as a key aspect of veracity, emphasizing factors like transparency, explainability, trialability, interoperability, accuracy, ethical concerns, and fairness in AI systems [20]. They outline intrinsic properties (e.g., accuracy, security, reliability, transparency, interpretability, and fairness) and extrinsic properties (e.g., trust in tech creators, product reviews, user perceptions, and accountability) influencing trust in digital technologies. These properties complement the definition of veracity used in this work.

The European Commission's ethics guidelines for trustworthy AI [2] advocate for AI systems to be transparent and explainable, ensuring their functions and decisions are understandable to users. The guidelines discuss seven requirements for ensuring trust within AI systems, which include human agency and oversight, technical robustness and safety, transparency, privacy and data governance, diversity, non-discrimination and fairness, and societal and environmental well-being and accountability. These requirements align with the multiple facets that 'Veracity' encapsulates.

Veracity envisioned as a Software Quality Attribute The ISO/IEC 25010 Software Quality Model [1] evaluates software quality based on quality attributes such as functional suitability, performance efficiency, compatibility, interaction capability, reliability, security, maintainability, flexibility and safety. Currently, veracity is not included as one of the quality attributes in it. There may be overlaps (e.g., authenticity, integrity), but some of the facets of veracity (discussed above) may not be covered by the characteristics discussed under quality attributes of ISO/IEC 25010.

Montgomery et al. [16] identified 12 quality attributes in requirements engineering, with ambiguity, completeness, consistency, and correctness being the most studied. Existing quality attributes, like completeness and correctness, may

overlap with veracity in certain aspects. For instance, completeness can partially address the accuracy of data requirements, and correctness relates to maintaining data accuracy and reliability. However, veracity encompasses additional dimensions such as data provenance, contextual trust, and cultural considerations (e.g., indigenous data sovereignty) that are not explicitly covered by these attributes. For example, in AI-driven healthcare systems, trustworthiness extends beyond correctness and consistency to include the transparency of data sources and compliance with ethical norms.

This study takes an exploratory approach to the concept of veracity as a potential software quality attribute and its potential value within RE. Inspired by established quality attributes, veracity offers a framework for addressing challenges in trust-sensitive systems, such as ensuring the provenance and ethical use of data, which are inadequately covered by existing attributes. Through the lens of RTD, we examine the impact of poor quality, incomplete or incorrect veracity requirements on system trustworthiness, particularly in domains where trust and provenance are paramount.

Requirements Technical Debt and its Quantification Ernst et al. [5] first defined RTD as "the distance between the optimal requirements specification and the actual system implementation, under domain assumptions and constraints." Lenarduzzi and Fucci [10] later expanded this to include upstream activities such as requirements elicitation and specification translation, categorizing RTD into Incomplete Users' Needs, Requirement Smells, and Mismatch Implementation. Melo et al. [14] conducted a Systematic Literature Review (SLR), identifying RTD causes, detection strategies, and metrics to support its management.

In our previous work [17, 18], we conducted a Systematic Mapping Study (SMS), synthesized a definition of RTD from literature (including Ernst et al. [5] and Lenarduzzi and Fucci [10]), and developed a conceptual model for RTD quantification. We define RTD as: *"RTD captures the consequences of suboptimal 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."* Our model serves as a theoretical foundation for both functional and non-functional RTD quantification. In this paper, we apply a few concepts from our model, Cost of fixing, Benefit of fixing, and Consequences of not fixing RTD, to the phenomenon of veracity debt.

3 Methodology

Our goal was to explore the industry awareness of veracity requirements and practitioners' views on quantifying veracity debt. We used a survey methodology to gather insights into software engineering practices and challenges [15], following guidelines from Linaaker et al. [11] and Kitchenham and Pfleeger [9]. The survey had four sections: two addressing research questions on RTD (reported separately) and veracity debt (reported in this paper), and two on demographics

(individual, company), placed at the beginning and end to reduce respondent fatigue. Hosted anonymously on Qualtrics[3], the survey had an estimated completion time of 25 minutes.

3.1 Research Questions

- **RQ1: What are practitioners' perceptions about veracity requirements and veracity debt?**
  - RQ1.1: What are the most common types of veracity requirements supported by the software solutions developed in the industry?
  - RQ1.2: What are the most common types of veracity requirements that can cause veracity debt with a significant impact?
  - RQ1.3 What impact do practitioners perceive that veracity debt could have?
- **RQ2: How could the quantification of veracity debt support making informed decisions to manage it?**
  - RQ2.1: What concepts of RTD quantification support making informed decisions to fix veracity debt?
  - RQ2.2: What are the most common types of veracity requirements that could benefit from veracity debt quantification and why?
  - RQ2.3: What tools, techniques, and other factors may help quantify veracity debt?

3.2 Study Design

Question Design The survey included single-choice, multiple-choice, Likert scale, and open-ended questions, aimed at answering RQ1 and RQ2. The survey began with an introduction to familiarize respondents with key concepts, including veracity, veracity requirements, veracity debt, and preliminary definitions of types of veracity requirements (regulatory, data, financial data, process, and cultural). We provided an illustrative example using a case study from the organic products certification domain in New Zealand. Due to space constraints, the full questionnaire, including the introduction text, is available in our replication package[4]. Below is an example defining the regulatory type: *"regulatory veracity requirements refer to compliance with regulations, such as organic product regulations, standards, and other regional, local, or international laws."*

We first asked practitioners about the veracity requirements typically implemented in their applications (RQ1.1). Practitioners were asked to recall a recent veracity-related incident using the Critical Incident Technique (CIT) [6]. This method helped them describe a memorable incident and respond to related questions. We then inquired about the types of veracity requirements involved (RQ1.2) and the perceived impact of veracity debt (RQ1.3). The survey provided

[3] Qualtrics: `https://www.qualtrics.com/strategy/research/survey-software`
[4] `https://doi.org/10.5281/zenodo.14172400`

predefined veracity requirement types from the Veracity Project [13], as well as 'Other' as an option for additional responses.

We structured the remaining survey questions around the scenario of fixing veracity debt instances, focusing on the consequences of sub-optimal decisions concerning veracity requirements. This helped gather practitioner insights on whether the RTD quantification concepts, Cost of fixing, Benefit of fixing, and Consequences of not fixing aid in decision-making (RQ2.1). Practitioners were then asked which veracity requirement types would mostly benefit from quantifying the RTD quantification concepts (RQ2.2). Finally, we inquired about existing or new tools, techniques, and other factors that could support RTD quantification for veracity requirements (RQ2.3).

Participant selection The study targeted software practitioners involved in RE, software design, and implementation, including roles such as business analysts, requirements engineers, software developers, architects, and project managers. We recruited participants via the researchers' professional networks (e.g., LinkedIn, Twitter) and Veracity Lab's industrial connections[5], allowing them to share the invitation with colleagues who meet the inclusion criteria. Participation was voluntary, with the inclusion criteria requiring practitioners to be engaged in RE or software implementation in the tech sector (in New Zealand or overseas) and to be proficient in English.[6].

3.3 Data collection

The survey invitation was sent to participants on February 9, 2024, and the survey was closed on April 23, 2024, after responses significantly slowed and did not seem to increase. A total of 82 responses were collected, of which 52 were considered valid.

Demographics Among respondents, 25% worked for large international businesses, the most common company type. Of the 52 participants, 50% were software engineers, systems engineers, or developers, followed by other roles (17.31%) and systems/software architects or tech leads (9.62%). Non-technical roles included project/product managers and team leads (7.69%), business analysts and requirement engineers (5.77%), and QA managers/engineers (1.92%), with technical roles dominating. Most respondents had 5–10 years of experience (30.77%), followed by 1–2 years (23.08%) and over 10 years (17.31%). In methodologies, 34.62% used a hybrid agile-waterfall approach, 32.63% followed agile exclusively, and 1.92% adhered to waterfall. Finance was the top application domain (12.36%), with respondents primarily working on feature enhancements (11.86%), refactoring (11.22%), and implementing or modifying software features (10.9%).

[5] `https://veracity.wgtn.ac.nz/`

[6] Approved by the University of Auckland Human Participants Ethics Committee on 13/11/2023 for three years (Reference Number UAHPEC26580)

3.4 Data Analysis

Data Cleaning and Pre-processing The survey responses, downloaded as a CSV file from Qualtrics, were imported into a Python environment for analysis. Responses with empty data or a progress rate below 13% were excluded, as these respondents only completed the demographics section and the first survey question before exiting. Finally, 52 responses were considered valid. Since all questions in the survey were optional and the survey paths varied based on prerequisite questions, the analysis focused on the number of respondents who answered each question.

Coding Open-Ended Questions We conducted a qualitative content analysis on the responses to the open-ended questions. The methodology we followed for the coding process was Thematic Analysis by Braun and Clarke [4]. We utilized the NVivo[7] software to code the responses. The responses were first coded with initial codes and then grouped into common themes that emerged from the coding. We then extracted meaningful insights. The initial codes and their high-level themes were discussed and agreed upon among the authors during meetings. At least two other authors checked the codes prior to the discussions.

4 Results

4.1 RQ1: Practitioners' Perceptions about Veracity Requirements and Veracity Debt

RQ1.1: Most Common Types of Veracity Requirements Data veracity requirements (86%) was the most common type, followed by regulatory (70.3%) and process (59.5%). Data is expected to dominate across domains as it is used in all software applications. The high occurrence of regulatory requirements may stem from increasing compliance needs, such as the General Data Protection Regulation (GDPR).

RQ1.2: Most Common Types of Veracity Requirements Causing a Significant Impact The most common type of veracity requirements reported in the most recent critical incident that created a significant impact on the respondent's project (or company) was *financial data veracity* (56.8%). *Cultural* (38.7%) and *process* (35.5%) followed. One respondent selected 'Other' as the option but said *"Recent issues weren't veracity related"*. The percentages were calculated out of a total of 31 respondents. We derived common themes in the critical incidents described by practitioners using Thematic Analysis [4]. Common themes and some examples are as follows: **complicated regulatory requirements** [regulatory veracity]: *"The regulatory requirements were very complicated, and the end users did not have enough time to share them with the agile development team"*, **misinterpreted regulatory requirements** [regulatory veracity]: *"We launched a new process based on a misinterpretation of*

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

a regulation", **lack of trust of end user** [data veracity]: *"The project I am working on is a Digital License application for one of the state government. Recently, our application went live, but the application was not downloaded as expected (We expected above 1 million downloads within 3 months, but got only 400000 downloads). Major feedback we received from the general public was that they prefer not to use digital licenses over physical cards due to lack of trust towards digital wallets."* , **market restrictions** [cultural veracity]: *"Korean maps can only be allowed to be downloaded inside of Korea."*, **requirement changes** [data veracity]: *"Incorrect push notifications were sent to mobile clients because of changing requirements"* , **lack of communication with the end user** [regulatory veracity]: *"The regulatory requirements were very complicated, and the end users did not have enough time to share them with the agile development team."* , **user feedback** [cultural veracity]: *"Major feedback we received from the general public was that they prefer not to use digital license over physical card due to lack of trust towards digital wallets."*

RQ1.3: Perceived Impact of Veracity Debt Practitioners rated on a Likert scale whether the consequences of not fixing a problem pertaining to veracity requirements could have a much higher impact on the software architecture (e.g., rework on the software architecture or data infrastructure) or the end-user (e.g., failing to meet the desired level of veracity) or the software company (e.g., in terms of potential ROI, product success and company reputation), compared to general software requirements.

Thirty-eight respondents rated the Likert scale questions in total, and 18 out of the 38 strongly agreed that the impact on the software company would be higher, while 15 strongly agreed that the impact on the end-user was higher, and 12 strongly agreed that the impact on the software architecture would be higher, compared to general software requirements.

Those who agreed ('Strongly agreed' and 'Somewhat agreed') that the **impact on the software architecture** was higher for veracity requirements compared to general software requirements said, for example, that *"Data veracity requirements could lead to significant rework on data storage and security"*, while another mentioned that *"small changes may have disastrous consequences down the line"*. Those who strongly disagreed said *"it was independent of whether it is a veracity requirement that there would be an impact"*. It is more likely that the impact on the software architecture was higher for veracity requirements compared to general software requirements. For example, when decisions are driven by the need to comply with a regulation such as GDPR, this may mean that the software company has no other way around it but to cater to the requirements of the regulation by making the necessary changes to its data infrastructure. Furthermore, if regulations change and the data infrastructure no longer adheres to it, then this can cause rework.

Those who agreed ('Strongly agreed') that the **impact on the end-user** was high said, for example, that *"end users may not be able to proceed with the business flow"*. Those who disagreed did not give reasons. Some of those who

had a neutral stance said *"it depends on the veracity debt"*. Some of them said that capturing veracity needs is a partnership with the stakeholders and that business owners are responsible for clearly establishing regulatory and compliance requirements. However, this particular example does not explain why there cannot be an impact on the end user regardless of the responsibilities.

Those who strongly agreed that the **impact on the software company** would be higher did not provide reasons for their rating. However, it could be that the impact on the end-user and the impact on the software architecture could eventually lead to the impact on the software company. For example, the impact on the end-user could lead to the loss of customers, which will create an impact on the software company. Similarly, the impact on the software architecture may reduce the ability of the software company to cater to a large scale of customers, which would eventually impact the company. Those who somewhat agreed mentioned that *"the reputation may be damaged and that fines could be incurred"*. Interestingly, there were no respondents who disagreed with this statement. Therefore, this implies that the impact on the software company is always perceived to be higher for veracity debt, according to our data.

4.2 RQ2: Support of Quantification for Informed Decision-Making

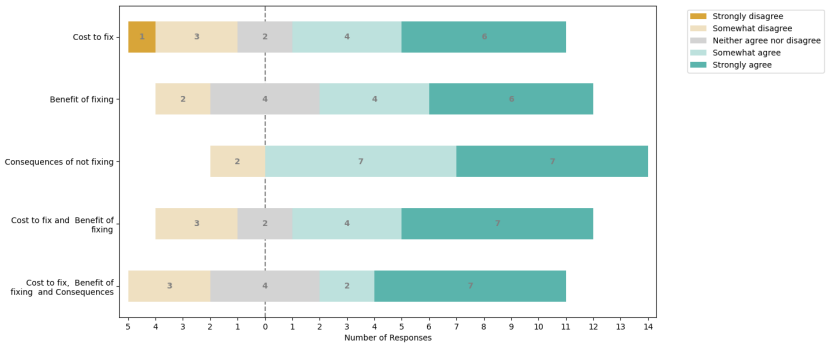

Fig. 1. Usefulness of quantifying RTD quantification concepts

RQ2.1: Concepts of RTD Quantification that support making Informed Decisions Respondents were asked to rate on a Likert scale whether they agreed with a given statement regarding the **usefulness of quantifying the Cost to fix, the Benefit of fixing, and the Consequences of not fixing** problems with veracity requirements, i.e., veracity debt instances. The statements also included the usefulness of quantifying the **combination of costs and benefits** as well as the **combination of costs, benefits, and consequences** for decision-making. See Figure 1.

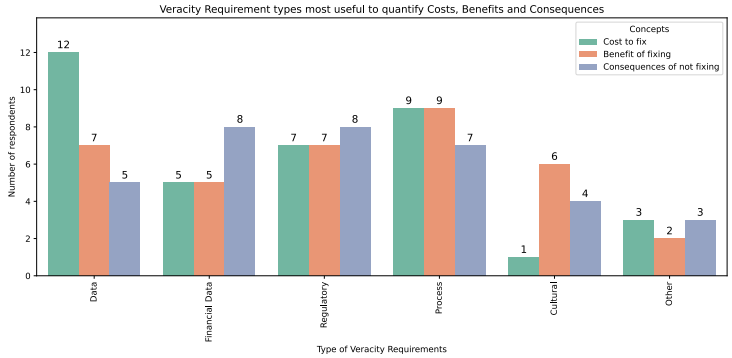

Fig. 2. Types of veracity requirements that would mostly benefit from quantifying the Cost to fix, Benefit of fixing, and the Consequences of not fixing

Out of 16 respondents who answered the Likert scale questions, six or more strongly agreed with all the statements. However, one respondent strongly disagreed that cost is a factor that would be useful to quantify to make an informed decision on whether and when to fix the veracity requirements problem. The reasoning was: *"In most of my situations fixes/changes need to be made as soon as possible", "Generally in domains in our operations veracity requirements are not optional. It's business critical. Analysis is just for budgets."*. This indicates that in this particular practitioner's domain, they may fix the problems with veracity requirements regardless of the Cost to fix as the veracity requirements are business critical. Among the others who gave reasoning for their ratings for the Likert scale questions, the following was observed. Strongly agreed: *"Not considering the cost can lead to damage in reputation"*, and Somewhat agreed: *"Similarly for regular requirements, depends on the size of the project"*.

RQ2.2: Most Common Types of Veracity Requirements that Could Benefit from Debt Quantification We gathered practitioner perceptions about what type of veracity requirements would mostly benefit from quantifying the cost of fixing veracity debt instances, the benefit of fixing, and the consequences of not fixing (i.e., veracity debt interest). We also provided participants with the option to provide their reasoning. Figure 2 illustrates the results.

According to practitioners' perceptions, the type of veracity requirements that would best benefit from quantifying **the Cost to fix** is *data veracity requirements*, 12 respondents chose this option. Some respondents explained the importance of data, mentioning that it is the base for any software solution. Respondents selected *process* (9 respondents) as the second option but gave no reasons. The third option was *regulatory*, chosen by seven respondents. Only one respondent gave reasons. The reason for selecting *regulatory* as the type of veracity requirement that would best benefit from quantifying the Cost to fix was that the respondent thought that regulatory would be the choice they would make

for the next question in the survey (type of veracity requirements that would best benefit from quantifying the Benefit of fixing). However, some respondents also commented that they do not see a point in gathering only the cost or the benefit for a specific type of requirement.

For the **Benefit of fixing**, the top three types of veracity requirements were *process (9), regulatory (7), and data (7)*. An interesting reasoning given by a respondent for selecting process was *"It is negotiable whether it should be fixed in contrast to, e.g., financial data or regulatory where it could be required to fix it."* Another respondent said that all tasks in the future will benefit from fixing process debts. — This response makes it unclear whether the respondent confused process veracity with process debt, which is a type of TD on its own. However, they could be overlaps which we have not explored yet. One of the respondents who chose regulatory said that the cost of non-compliance is usually easy to quantify and that the fee may be explicitly listed in the legislation. Another respondent said that regulatory is the choice because they want to reduce unnecessary fines. This is interesting since regulatory veracity debt could be potentially quantifiable in monetary terms, according to the respondents who mentioned that the costs or fines are already given in the legislation. A respondent who selected Data said that this type could encapsulate other types, so that was the reason for selecting Data. This is also interesting since we can already see that *regulatory* and *financial data* could already be seen as different types of data, apart from being different types of veracity. Hence, it may be worth investigating the relationships between these different types in future work.

For the **Consequences of not fixing**, the popularity of the choices made by the respondents was in the order *financial data (8)*, *regulatory (8)*, and *process (7)*. Among the reasons given for their choices, a respondent who chose financial data mentioned that it could have drastic negative consequences, and they must be addressed as the most important ones. A respondent who chose regulatory mentioned that the consequences could be easily quantified for regulatory, giving the same example of fines being already included in the legislation. Respondents who selected process gave similar reasons for selecting the same option in the other scenarios as well.

RQ2.3: Tools, Techniques, and Other Factors to Consider The themes derived from the answers to the open-ended questions can be found in Figures 3 (Tools and techniques) and 4 (Other factors). Tools included only proprietary tools in comparison to the tools for general RTD quantification. *Excel, Jira, Confluence, Linters, and Github* were given as the proprietary tool options by the survey respondents. One participant said: *"Jira, Confluence, and GitHub (or other source code repositories) may have some relevant metrics, especially regarding transparency."* The following were among the techniques: *Change request handling, weighted shortest job first, value of information analysis, having legal experts look into compliance requirements early on, having meetings where veracity debt was on the agenda, and interviews with experts, refinements and story estimations.* However, some respondents also mentioned that they were

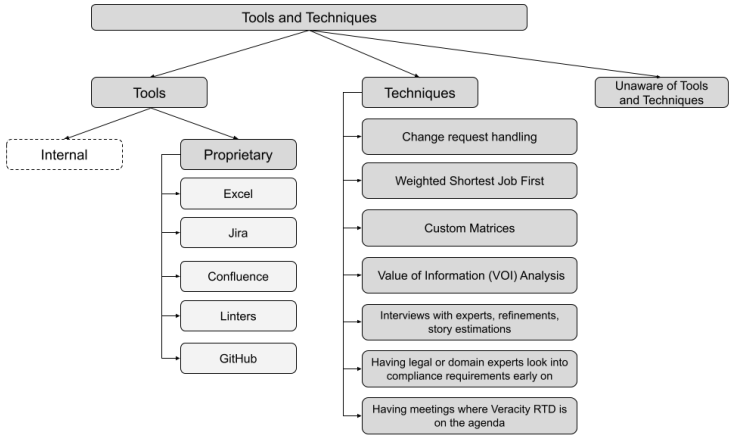

Fig. 3. Tools and techniques currently used in the industry

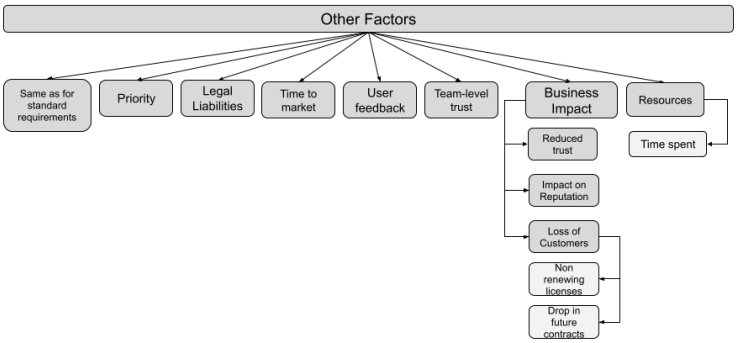

Fig. 4. Other factors to consider in quantification

unaware of tools and techniques that could be useful. For example, *" I'm not aware of any such tools, though that could be just my own ignorance. I would certainly welcome having them".*

For other factors (apart from costs, benefits, and consequences) useful to consider when deciding to fix a problem with veracity requirements compared to other software requirements, respondents mentioned *priority, legal liabilities, time to market, user feedback, team-level trust, business impact, and resources.* See Figure 4. In terms of the *business impact, reduced trust, impact on the company's reputation, and loss of customers* were identified as the recurrent themes. Loss of customers could be in terms of *non-renewing licenses or a drop in future contracts*, according to our respondents. *Time spent* was a resource to be considered as a factor to quantify, which can also be interpreted as development cost, which we discuss in our conceptual model in Perera et al. [18]. Some respondents also said that the same factors for RTD would also apply to veracity

debt. *Priority, timeline/ time to market, business impact, and resources* were common themes for both RTD and veracity debt. Some factors that seem specific to veracity debt were *legal liabilities, user feedback, team-level trust, reduced trust, impact on reputation, and loss of customers.*

5 Discussion

5.1 Impact of Veracity Debt

Respondents indicated that veracity debt could disrupt end-users' business processes, require major architectural rework (e.g., data storage and security), and lead to reputational damage and fines. Most practitioners agreed that its impact on a company is greater than that of general software requirements, with no opposing views. This emphasizes the need for stakeholders to prioritize its effective management.

5.2 Quantifying Veracity Debt

Practitioners identified different veracity requirements as most benefiting from quantifying the RTD concepts: Cost of fixing, Benefit of fixing, and Consequences of not fixing. This suggests practitioners may prefer quantifying different RTD concepts based on the type of veracity requirements to make informed decisions about addressing veracity debt. However, the reasons given by the practitioners do not give a clear explanation as to why they had different choices for the types of veracity requirements that could benefit from quantifying a given RTD quantification concept.

For Cost to fix, the most popular choice was data veracity requirements, likely due to the high rework costs of data infrastructure in software systems. For Benefit of fixing, practitioners favored process veracity, reasoning that addressing these issues would yield future benefits, though their understanding of process veracity seemed unclear. For Consequences of not fixing, financial and regulatory were top choices, as these issues pose significant risks to a company's reputation and exposure to fines, quantifying the consequences will be more useful for making a decision regarding those types.

Some respondents believed that quantifying only one aspect, or just the cost and benefit, might not suffice for any type of veracity requirement. However, *practitioners generally agreed on the importance of quantifying the Consequences of not fixing veracity debt (i.e., veracity debt interest) for effective decision-making.* Techniques suggested for quantifying costs, benefits, and consequences included value of information analysis, early involvement of legal experts for compliance, and prioritizing veracity debt in meetings. While some practitioners lacked awareness of tools, key factors influencing the decision to fix veracity debt versus general RTD included legal liabilities, user feedback, team-level trust, reduced trust, impact on reputation, and loss of customers.

5.3 Implications to Researchers, Practitioners, and Future Work

Implications to Researchers and Practitioners Our findings suggest that software practitioners recognize the need to quantify and manage veracity debt. There may be relationships among the different types of veracity requirements — *regulatory, data, financial data, cultural, and process.* For instance, one type might introduce veracity requirements related to another or act as a constraint on another type. Additionally, there may be other types of veracity requirements that we have not yet encountered. Hence, it is worthwhile to investigate the different types of veracity requirements and their relationships further.

Software practitioners could benefit from our thinking to develop "veracity-aware software systems" to reduce and prevent costs associated with veracity debt and to provide customers with software and AI products that meet customer expectations in terms of veracity. However, our findings show that veracity's value depends on the understanding of its relationship with established quality attributes in RE as well as software quality models. Therefore, precisely defining veracity requirements and ensuring the measurability of veracity debt remains a key challenge, emphasizing the need for further research. We encourage follow-up studies to further investigate veracity conceptually as well as evaluate whether terming veracity as a distinct quality attribute improves outcomes or if it is better integrated into existing attributes. We argue for the former since it allows for distinguishing characteristics that are not covered by established concepts, such as trustworthiness and cultural sensitivity, and we see how this could be useful and timely, given the rise of AI systems.

Future Work An ongoing follow-up study involves semi-structured interviews with software practitioners, enabling deeper discussions and additional insights into managing veracity debt. The study aims to assess whether various types of veracity requirements are universally applicable across domains and to explore their interrelationships. It also seeks to determine whether the consequences of veracity debt for one type of requirement should be prioritized for quantification and management over another and whether distinct management strategies are necessary for different types.

6 Threats to validity

Internal Validity We acknowledge the possibility that some Likert scale questions may not have been well formulated, potentially introducing acquiescence bias. However, it is difficult to determine whether this bias significantly impacted the results. We hope that respondents provided ratings reflecting their personal opinions, as the data includes a range of responses, including strong or moderate disagreement, as well as neutral positions, suggesting varied perspectives.

Construct Validity Our survey instrument was designed based on the conceptual model reported in Perera et al. [18], but we avoided technical jargon, opting for more accessible language. Multiple-choice question options were clearly explained with examples to ensure respondents were well informed. To enhance

clarity, we used the terms 'measure' and 'estimate' alongside 'quantification', accommodating participants who might use different terms with similar meanings. Additionally, an introductory text was provided before the survey to familiarize respondents with the concept of 'Veracity Debt' and its different types or dimensions.

External Validity We employed *Convenience* and *Snowball* sampling methods to recruit participants, leveraging social media platforms, such as LinkedIn and Twitter, to reduce costs [11, 19]. Among the 52 valid survey responses, 61.52% of respondents disclosed their country of work, though answering this question was optional. Similarly, only 5.77% did not specify their role at work, leaving 94.23% of responses representative of our target population. Over half of the participants held technical roles. Therefore, while the participants came from diverse roles, company sizes, domains, and countries, our findings are likely more applicable to technical stakeholders than to the broader software practitioner population worldwide.

7 Conclusion

The work reported in this paper introduces a new research direction, recognizing the growing importance of veracity requirements in contemporary and future software systems. The software industry can benefit from our research by putting forward the voice of practitioners reported in our survey to be aware of veracity debt and device plans to manage it. By considering veracity debt and developing 'veracity-aware software systems', software practitioners can minimize costs and deliver software products that align with customer expectations related to veracity. However, the concept of veracity debt is still immature and would benefit from further investigations.

Acknowledgments. This research is funded by the New Zealand Ministry of Business, Innovation, and Employment via The Science for Technological Innovation (SfTI) National Science Challenge Veracity Technology Spearhead.

Disclosure of Interests. The authors have no competing interests to declare.

References

1. ISO/IEC 25010:2011 - systems and software engineering — systems and software quality requirements and evaluation (square) — system and software quality models (2011), `https://iso25000.com/index.php/en/iso-25000-standards/iso-25010`, accessed: 2025-01-25
2. Ala-Pietilä, P., Bonnet, Y., Bergmann, U., Bielikova, M., Bonefeld-Dahl, C., Bauer, W., Bouarfa, L., Chatila, R., Coeckelbergh, M., Dignum, V., et al.: The assessment list for trustworthy artificial intelligence (ALTAI). European Commission (2020)
3. Blincoe, K., Luczak-Roesch, M., Miller, T., Galster, M.: Human-centric literature on trust for sfti veracity spearhead. arXiv preprint arXiv:2306.00226 (2023)

4. Braun, V., Clarke, V.: Using thematic analysis in psychology. Qualitative research in psychology **3**(2), 77–101 (2006)
5. Ernst, N.A.: On the role of requirements in understanding and managing technical debt. In: 2012 Third International Workshop on Managing Technical Debt. pp. 61–64. IEEE (2012). https://doi.org/10.1109/MTD.2012.6226002
6. Flanagan, J.C.: The critical incident technique. Psychological bulletin **51**(4), 327 (1954)
7. Galster, M., Dietrich, J.: Towards understanding provenance in industry (2023)
8. García Lozano, M., Brynielsson, J., Franke, U., Rosell, M., Tjörnhammar, E., Varga, S., Vlassov, V.: Veracity assessment of online data. Decision Support Systems **129**, 113132 (2020). https://doi.org/https://doi.org/10.1016/j.dss.2019.113132
9. Kitchenham, B.A., Pfleeger, S.L.: Personal opinion surveys. In: Guide to advanced empirical software engineering, pp. 63–92. Springer (2008)
10. Lenarduzzi, V., Fucci, D.: Towards a holistic definition of requirements debt. In: 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. pp. 1–5. IEEE (2019). https://doi.org/10.1109/ESEM.2019.8870159
11. Linåker, J., Sulaman, S.M., de Mello, R.M., Höst, M.: Guidelines for conducting surveys in software engineering. Department of Computer Science, Lund University (2015)
12. Lozano, M.G., Brynielsson, J., Franke, U., Rosell, M., Tjörnhammar, E., Varga, S., Vlassov, V.: Veracity assessment of online data. Decision Support Systems **129**, 113132 (2020)
13. Luczak-Roesch, M., Galster, M., Shedlock, K.: The veracity grand challenge in computing: A perspective from aotearoa new zealand. Commun. ACM **66**(7), 67–69 (2023). https://doi.org/10.1145/3589154, `https://doi.org/10.1145/3589154`
14. Melo, A., Fagundes, R., Lenarduzzi, V., Santos, W.B.: Identification and measurement of requirements technical debt in software development: A systematic literature review. Journal of Systems and Software p. 111483 (2022)
15. Mendez, D., Avgeriou, P., Kalinowski, M., Ali, N.B.: Handbook on teaching empirical software engineering
16. Montgomery, L., Fucci, D., Bouraffa, A., Scholz, L., Maalej, W.: Empirical research on requirements quality: a systematic mapping study. Requirements Engineering **27**(2), 183–209 (2022)
17. Perera, J., Tempero, E., Tu, Y.C., Blincoe, K.: Quantifying requirements technical debt: A systematic mapping study and a conceptual model. In: 2023 IEEE 31st International Requirements Engineering Conference (RE). pp. 123–133 (2023). https://doi.org/10.1109/RE57278.2023.00021
18. Perera, J., Tempero, E., Tu, Y.C., Blincoe, K.: Modelling the quantification of requirements technical debt. Requirements Engineering pp. 1–38 (2024). https://doi.org/10.1007/s00766-024-00424-3
19. Ralph, P., Ali, N.b., Baltes, S., Bianculli, D., Diaz, J., Dittrich, Y., Ernst, N., Felderer, M., Feldt, R., Filieri, A., et al.: Empirical standards for software engineering research. arXiv preprint arXiv:2010.03525 (2020)
20. Yang, X., Jabeen, G., Luo, P., Zhu, X.L., Liu, M.H.: A unified measurement solution of software trustworthiness based on social-to-software framework. Journal of Computer Science and Technology **33**, 603–620 (2018)