Ensuring Trust and Integrity in Software Systems: Identifying, Integrating, and Measuring Veracity Requirements Technical Debt: A Systematic Literature Review.

Natalia Ortega, Judith Perera, Matthias Galster, Kelly Blincoe.

1. Introduction

Software requirements related to Veracity, or Veracity Requirements, pertain to the trustworthiness, truthfulness, authenticity, provenance, integrity, and demonstrability of data and human interactions (Perera, 2023). These requirements can be classified as either Functional, such as a software feature like a checklist, or Non-Functional, such as a quality attribute like 'the trustworthiness of the software system,' depending on the software system being developed and its context (Perera et al., 2023b) .

The increasing complexity of software systems, especially those leveraging artificial intelligence (AI), has introduced new challenges in ensuring these Veracity Requirements are met (Galster & Dietrich, 2023). The dense nature of AI components often leads to "black boxes" that even engineers struggle to understand, raising concerns about how to trust such systems (Galster & Dietrich, 2023). Moreover, hidden dependencies, such as those arising from code cloning, further complicate the landscape of Veracity Requirements by introducing security vulnerabilities (Dietrich et al., 2023). Furthermore, public trust in digital technologies is essential for the confident adoption of digital products (Galster & Dietrich, 2023). Thus, regulatory demands now necessitate methods to provide evidence for software compliance, regardless of whether AI is involved, alongside user concerns about the transparency of software-supported decisions (Galster & Dietrich, 2023). Investigating use cases such as monitoring Traditional Knowledge labels, enhancing transparency in organic product supply chains, and enabling contestability in AI decision-making can provide valuable insights into managing Veracity Requirements (Blincoe et al., 2023). Understanding who demands software provenance is also critical; regulators and third parties are the primary drivers, with end users and developers rarely requesting provenance information (Galster & Dietrich, 2023). This information is crucial for regulatory compliance and maintaining public trust in software-supported decisions (Galster & Dietrich, 2023).

Identifying Veracity Requirements across the software development lifecycle, from initial requirement gathering through to deployment and maintenance, would be critical. These requirements play a pivotal role in establishing and maintaining stakeholder trust, particularly in domains where data integrity is paramount, such as financial and healthcare systems (Perera et al., 2023a). Thus, consider Requirements Technical Debt (RTD), linked with Veracity, can incur additional costs, referred to as RTD Interest, which impact both upstream activities, such as Requirements Engineering (RE), and downstream activities, such as Implementation and Maintenance in the software development lifecycle (Perera et al., 2023a). RTD involves a feedback loop that includes the user (Perera et al., 2024). Therefore, it is crucial to precisely capture user needs and accurately formalize them as requirements to minimize the accumulation of RTD (Perera et al., 2023a). It is also essential to monitor these Veracity Requirements closely and avoid deviations during implementation (Perera et al., 2023a). These challenges highlight the need for rigorous Veracity Requirements throughout the software development process to ensure the reliability and trustworthiness of data.

Measuring RTD associated with Veracity Requirements would be critical for maintaining the integrity and trustworthiness of software systems. Perera et al. (2024) developed a conceptual model to quantify RTD. RTD encapsulates the outcomes of less-than-optimal decisions made during the identification, formalization, and implementation of requirements, which can be strategic or due to changing contexts (Perera et al., 2023a). Perera et al. (2024) identified Veracity Requirements Technical

Debt (VRTD), or Veracity Debt (a short version of VRTD), as a specific type of RTD. This concept is related to Veracity Requirements, as not considering the RTD of software systems can undermine data trustworthiness and reliability, impacting stakeholder confidence and overall system integrity (Perera et al., 2023a). Quantifying the impact of Veracity Debt associated with RTD could persuade stakeholders to prioritize its management, ensuring that software systems maintain the necessary standards of Veracity (Perera et al., 2023a). Perera et al. (2023b), through stakeholder interviews, revealed that Veracity Requirements can be categorized into data, process, financial, regulatory, and cultural dimensions, each requiring tailored approaches to manage and mitigate RTD effectively (Luczak-Roesch et al., 2023; Perera et al., 2023b).

Addressing Veracity Requirements throughout the software lifecycle could be critical for the development of trustworthy and reliable software systems. By identifying, categorizing, and measuring Veracity Requirements and their associated technical debt, stakeholders would ensure that their systems meet the highest standards of data integrity and public trust.

The aim of this paper is to understand how to manage Veracity Requirements Technical Debt (VRTD) throughout the software lifecycle to develop trustworthy and reliable software systems. By identifying, categorizing, and understanding how to measure Veracity Requirements and their associated technical debt to ensure stakeholders maintain data integrity and public trust in software systems.

A systematic literature review was conducted to comprehensively explore the identification, categorization, and measurement of Veracity Requirements and their associated technical debt, ensuring stakeholders maintain data integrity and public trust in software systems.

2. Systematic literature review (SLR)

The systematic literature review was conducted following the framework recommended by Khan et al. (2003). This review comprised five stages: formulating the research problems, identifying relevant work, assessing the quality of studies, summarizing the evidence, and interpreting the findings.

2.1 Formulating the research problems

Based on the introduction to Veracity Requirements, this paper aims to address their identification, categorization, and exploration of measurement methodologies, metrics, and techniques. Two main questions and seven sub-questions were formulated to gain a detailed understanding of these aspects (see Table 1).

Table 1: Systematic literature review research questions.

Main research questions	Sub-questions	Assessed aspects
MQ1: What are Veracity Requirements in software systems?	SQ1: Why are Veracity Requirements important for software systems? SQ2: What are examples of Veracity Requirements? SQ3: In which domains are Veracity Requirements most critical? SQ4: What are the key characteristics that define Veracity Requirements?	Definition and scope. Importance and impact. Attributes and features.
MQ2: How can we integrate the Veracity attributes in the RTD quantification model?	SQ5: How can the impact of RTD related to Veracity on software be assessed? SQ6: What approaches are employed to measure Veracity Debt or Veracity Requirements?	Measurement metrics. Methodologies measures.

	SQ7: What measurement metrics and methodologies can be integrated into the RTD Quantification Model?	

2.2 Identifying the relevant work

Eligible papers included in the systematic literature review were retrieved from the Scopus database, with searches conducted on titles, abstracts, and keywords considering these keywords 'Veracity', and 'requirement*', and 'software*'. Scopus is widely recognized as one of the leading and most popular research databases (Belter, 2015; Guz & Rushchitsky, 2009), covering a diverse range of scientific fields. It offers one of the broadest journal coverages across all disciplines (Mongeon & Paul-Hus, 2016) and boasts 20% greater accuracy in citations compared to other databases (Belter, 2015). Additionally, a complementary method called snowballing was employed, which involves retrieving relevant papers based on the reference lists of target papers to ensure comprehensive coverage and inclusion of any potentially missing papers (Wohlin, 2014). The searches were conducted on 20 May 2024 and yielded a total of 50 results from Scopus (www.scopus.com). A filtering process was carried out following a framework called Preferred Reporting Items for Systematic Literature Reviews and Meta-Analyses (Moher et al., 2010) (see Figure 1) PRISMA flow diagram of the literature review process (Moher et al., 2010). The papers were filtered in accordance with the following inclusion and exclusion criteria: firstly, before screening, papers were excluded if there was no English language. Veracity and requirements and software and research related to terms in the titles or abstracts. Subsequently, the rest of the papers' full texts were assessed for eligibility. The eligible papers were included if they met some of the following criteria during the screening stages:

(i) Veracity Requirements were presented.

(ii) A software example was evaluated or developed considering Veracity Requirements.

(iii) A measure or method to measure Veracity is evaluated or proposed.

The papers were excluded if:

(i) There was no identification, categorization, or exploration of measurement metrics and techniques about Veracity Requirements.

(ii) There is an indication that the document is about Big Data, which is frequently defined by the 5Vs: Volume, Velocity, Variety, Value, and Veracity.

After the filtering process, snowballing was adopted to identify additional papers. The snowballing was based on the previous filtered results after PRISMA (Wohlin, 2014). The inclusion and exclusion criteria were the same as the ones adopted in the PRISMA framework.

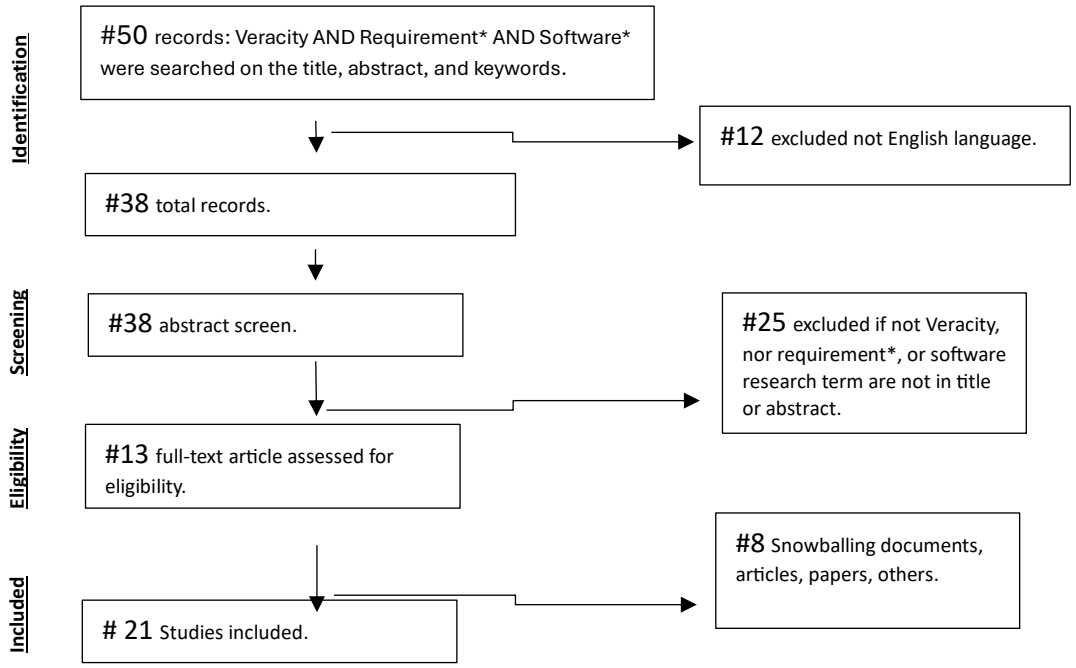

Figure 1: PRISMA flow diagram of the literature review process.

2.3 Assessing the quality of studies

The papers were classified based on the inclusion-exclusion criteria following PRISMA. There were 13 papers classified under the inclusion criteria, 8 articles were included per snowballing, and 25 documents were excluded based on the exclusion criteria.

2.4 Summarizing the evidence

The eligible papers were coded and analysed using a data extraction spreadsheet that involved the research questions assessed aspects. Due to the limited number of eligible papers, all 21 papers were considered.

2.4.1 Veracity Requirements in Software Systems

Assessing the importance and impact, as well as the attributes and features, of Veracity Requirements in software systems is essential for ensuring data accuracy, quality, and trustworthiness (Levi et al., 2016; Simpson & Foltz, 2017). These requirements are fundamental because they prevent errors and biases, leading to more accurate outcomes and enhanced user confidence (Simpson & Foltz, 2017). Examples include data validation checks, error detection mechanisms, and data cleansing and enrichment processes (Simpson & Foltz, 2017, 2018b). Veracity Requirements are particularly crucial in domains such as healthcare, finance, and government, where data accuracy is paramount (Simpson & Foltz, 2017). Key characteristics defining these requirements include accuracy, reliability, consistency, and inclusiveness, ensuring data correctly represents real-world scenarios, is trustworthy over time, is standardized across datasets, and includes comprehensive data points for analysis (Hovorushchenko et al., 2019; Simpson & Foltz, 2017). All software systems have Veracity Requirements, often classified as Non-Functional requirements comparable to 'security' or 'reliability' (Perera et al., 2024). However, some software systems may need specific Veracity Requirements to be fulfilled as Functional requirements (Perera et al., 2024). Perera et al. (2024) characterize and illustrate

the different types of Veracity Requirements pertaining to these dimensions: data, process, financial, regulatory, and cultural dimensions (Perera et al., 2024).

Dave et al. (2022) stated Implicit Requirements (IMR) and delineated a taxonomy for them. IMR identification is a critical part of the Requirements Engineering (RE) phase in Software Engineering, during which data is gathered to create SRS documents. IMRs represent subtle data that must be inferred rather than explicitly stated, and research has shown that they are crucial to the success of software development (Dave et al., 2022). Many software systems fail due to inadequate management of IMR data (Dave et al., 2022). Dave et al. (2022) delineate a taxonomy of IMRs into several categories—Security, Accessibility, Maintainability, Sustainability, and Usability—each accompanied by suitable examples to clarify these concepts. Additionally, a comprehensive overview of existing techniques, practices, and tools used in IMR identification is presented, offering a thorough understanding of how to manage and implement IMRs in software development projects effectively (Dave et al., 2022).

Requirements that are presented on a typically large Software Requirements Specifications (SRS) are documented, which can have hundreds of pages and contain thousands of requirements (Dave et al., 2022). Hovorushchenko (2018) also highlights the current and essential task of analysing SRS. This analysis is crucial for determining the feasibility of software projects that have incomplete or deficient requirements (Hovorushchenko, 2018). Ensuring the sufficiency of information within these requirements is necessary for the success of software projects (Hovorushchenko, 2018). Additionally, Hovorushchenko (2017b) confirms that the causes of many incidents and accidents related to software are often found in the SRS rather than in the coding itself. This highlights the critical importance of thorough and accurate SRS documentation in preventing software-related issues (Hovorushchenko, 2017b). Furthermore, Hovorushchenko (2018) notes that the insufficiency, inaccuracy, and distortion of the SRS information led to a decrease in the Veracity of software quality assessments.

2.4.2 Case Studies Related to Veracity Requirements

Medical monitoring software

For instance, in medical monitoring software, incorrect calculations can cause danger to a patient's life (Majma & Babamir, 2014). Therefore, the Veracity Requirements, like the data collected from patients, are of significant importance when incorporated into any medical software application (Majma & Babamir, 2014). Furthermore, Yang (2009) describes the software for the electrocardiogram bedside instrument as a multi-tasking real-time system designed for monitoring purposes. It ensures high Veracity in the process, thereby enhancing the instrument's utility as a reliable monitoring tool (Yang, 2009). When abnormalities are detected during processing, the software evaluates whether a warning is necessary, demonstrating an arrhythmia detection accuracy rate of up to 95.2% (Yang, 2009).

Quality assurance in manufacturing

In manufacturing, looking for software that incorporates models to make predictions about defect characterization, identification, and rectification is vital to ensuring the products meet the standards (Feng et al., 2023). Veracity Requirements related to the manufacturing process are crucial for ensuring that products meet quality standards. (Feng et al., 2023).

Accurate data in aerospace systems

Another example is aerospace, where to optimize the performance of the software system of an advanced telescope, it was necessary to develop analytical tools and processes that enable the testing of data Veracity in real-time by following existing measurements from different instruments (Levi et

al., 2016). The Veracity Requirements previously developed for other existing instruments were key to the correct technical development of the new telescope (Levi et al., 2016).

Ensuring integrity and security in cybersecurity systems

In cybersecurity, according to Simpson and Foltz (2017), all security software in the Veracity system is periodically refreshed from secure memory and reconfigured to avoid as-yet undetected threat activity. This regular refresh and reconfiguration process is crucial for maintaining the integrity and security of the system (Simpson & Foltz, 2017).

Nikhat et al. (2016) present a Model to Quantify Integrity (MQI) at the requirement phase of the software development process, which aims to address all significant constraints regarding secure requirements. Producing secure software requires requirement integrity, which encompasses trustworthiness, completeness, and Veracity (Nikhat et al., 2016).

Cultural data management in Māori software

In cultural software applications, like those developed for Māori communities system for managing tribal registrations, respecting the sovereignty of whānau (family) and hapū (tribe) over their data (Tairea et al., 2023). This system allows these groups to create and manage their own membership databases, which can be linked to a broader tribal database (Tairea et al., 2023). Although this software promotes collective contribution and knowledge sharing in a secure, private environment, enabling tribal communities to form private groups for administrators, historians, and members to view and contribute, some of them are reluctant to share that sensible information (Tairea et al., 2023).

2.4.3 Quantifying RTD for Software Requirements Concerning Veracity

VRTD, or Veracity Debt, captures the consequences of sub-optimal decisions made concerning Veracity Requirements, either deliberately (for strategic gains) or unconsciously (due to changes in context), during the identification, formalization, and implementation of requirements (Perera et al., 2023a). These deficiencies in requirement handling highlight the importance of thorough and accurate SRS, as the implications reach beyond development into the integrity of software quality assessments (Dave et al., 2022; Hovorushchenko, 2017b).

To understand the VRTD, first, it is necessary to comprehend that RTD participates in various stages of software development (Perera et al., 2023a). Software requirements concerning Veracity can be either Functional or Non-Functional, or both, depending on the software system. During development, most of the Non-Functional requirements are described. These requirements establish the criteria by which a system or its components are assessed for performance under specific conditions (Dave et al., 2022; Perera et al., 2023a). Functional requirements are defined prior to the start of development (Dave et al., 2022). Software requirements related to Veracity, or Veracity Requirements, relate to the trustworthiness, truthfulness, authenticity, and integrity of data and human interactions (Perera et al., 2023b).

Identifying Veracity Requirements across the software development, from initial requirement gathering through to deployment and maintenance, is critical (Perera et al., 2023b). These issues are paramount as the causes of many incidents and accidents related to software are often found in the Software Requirements Specifications (SRS) rather than in the coding itself (Hovorushchenko, 2017a). Such deficiencies in the SRS can lead to a decrease in the Veracity of software quality assessments (Hovorushchenko, 2017a) and an increase in VRTD (Perera et al., 2023b). VRTD relates specifically to Veracity Requirements, which are defined by trust, truth, authenticity, and demonstrability (Perera et al., 2023b).

According to Perera et al. (2024), by extracting concepts from the literature and categorizing and grouping them through a Systematic Mapping Study, 33 concepts were identified to develop conceptual models for RTD quantification. RTD Quantification Model (RTDQM) is combined with the Technical Debt Quantification Model (TDQM) to illustrate the cascading impact of RTD on the rest of the development phases, such as design and implementation. These models provide a framework for understanding and quantifying RTD across these stages—Requirements Engineering, Implementation, and Maintenance; also, the feedback loop involving the user is important for RTD (Perera et al., 2023a).

The 33 concepts of these models capture the importance of managing RTD from the early stages of the lifecycle, emphasizing the need to handle RTD from requirements engineering to implementation and maintenance (Perera et al., 2023a).

A mapping was done among the case studies to identify what concepts were considered (see Table 2). These concepts include a list of attributes relevant to Veracity (Perera et al., 2023a).

Majma and Babamir (2014) highlight several concepts, including patient behaviour monitoring and data collection from the patient's body using medical monitoring embedded software. Also, this software outlines three steps: defining variables and rules and designing a model to verify software behaviour. The concepts align with the conceptual model (Perera, 2024) relevant to Veracity, which includes User Need, (Documented) Requirement, Functional Requirement, Context, Risk, Scenario, Design Step, Design/Architectural Decision, Product Value, End User Satisfaction Level, and Priority of a Requirement. Thus, the importance of software behaviour Veracity in medicine.

Feng et al. (2023) emphasize the increasing demands for data analysis and software tools in additive manufacturing, reflecting the user needs. The process of deriving Functional requirements on additive manufacturing for software tools involves requirements engineering, which supports activities such as product design, design analysis, process planning, process monitoring, process modelling, process simulation, and production management. The context is provided by the rapid transition of AM for metals, which leads to new scenarios. The Veracity of data is a concern that introduces risks in managing and controlling processes, which implies that ensuring data Veracity is a prioritized requirement. The use of sensor measurement devices opens new scenarios, increasing the variety of data while decreasing its veracity. Data accuracy and Veracity in AM are needed to guarantee the quality that drives the necessity of designing trustworthy software that adds value to AM products by improving process efficiency and accuracy, ultimately leading to higher end-user satisfaction.

Levi et al. (2016) present a paper about the James Webb Space Telescope (JWST) Optical Telescope Element (OTE), which presents several clear concepts from Perera's (2024) conceptual model. The need for accurate alignment of the JWST reflects a critical user need to ensure the telescope's optimal performance. A comprehensive requirements engineering step is required to define strict accuracy demands for measurements, which are documented. The Functional requirements specify the use of various measurement instruments such as laser trackers, photogrammetry, and laser radar instruments. Thus, there is a high risk associated with software measurement developments. The prioritization of alignment accuracy is evident as it is crucial to optimize the telescope's performance. The design step includes measuring and aligning the telescope using different instruments and analytical tools, while design and architectural decisions determine the use of specific measurement instruments and software.

In Simpson and Foltz's (2017) paper about Enterprise Level Security (ELS), concepts from Perera's (2024) conceptual model are identified. The need for identity assurance in ELS, especially without traditional accounts and passwords, highlights a user need for secure and trustworthy access control.

This involves rigorous requirements engineering ensuring security and trustworthiness. Documented requirements include registering all entities and applying multi-factor authentication. Functional requirements are supported by Non-Functional requirements like periodic vetting to establish trustworthiness baselines. Risks from untrustworthy scenarios include periodic vetting for increased surveillance and logging. Ensuring secure identity management adds product value and leads to high stakeholder satisfaction.

Tairea et al. (2023) present a context of managing tribal registries, and concepts from Perera's (2024) conceptual model are evident. The need for a reliable system to register and verify tribal members reflects a critical user need. Comprehensive requirements engineering is involved in defining specific requirements and ensuring the system meets the needs of the tribal community. Documented requirements include the steps and procedures for registering and verifying tribal members. Functional requirements encompass the ability to register members, verify their identities, and manage the registry information. Non-Functional requirements focus on maintaining cultural integrity and trust within digital spaces. The context involves varying practices between tribal entities, and different scenarios can differ in the registration and verification processes. The value added by ensuring the tribal registry system is reliable, culturally appropriate, and trustworthy enhances community engagement.

Table 2: Mapping 33 concepts from Perera et al. (2024) Models relevant to Veracity across the case studies.

	Concepts in the conceptual model (Perera, 2024) relevant to Veracity	Majma	Feng	Levi	Simpson	Tairea
1	User Need	x	x	x	x	x
2	Requirements Engineering Step		x	x	x	x
3	(Documented) Requirement	x	x	x	x	x
4	Functional Requirement	x	x	x	x	x
5	Non-Functional Requirement				x	x
6	RTD Item					
7	RTD Rectifying Step					
8	Context	x	x	x	x	x
9	Risk	x	x	x	x	x
10	Scenario	x	x		x	x
11	Quality Attribute		x		x	x
12	(Prioritized) Requirement		x	x	x	x
13	Design Step	x	x	x	x	x
14	Design/ Architectural Decision	x	x	x	x	x
15	Implementation cost of Architectural/ Design decision					
16	Total Cost of a RE Step					
17	Cost of Rectifying RTD					
18	RTD Interest					
19	New Code Cost associated with RTD					
20	Rework Code Cost associated with RTD					

21	New Design Cost associated with RTD					
22	Rework Design Cost associated with RTD					
23	New RE Cost associated with RTD					
24	Rework RE Cost associated with RTD					
25	Documentation cost of a requirement					
26	Benefit of Rectifying					
27	Benefit of taking RTD					
28	Benefit of not taking RTD					
29	Product Value	x	x	x	x	x
30	End user satisfaction level	x	x	x	x	x
31	RTD Interest Probability					
32	RTD Item Priority					
33	Priority of a requirement	x	x	x	x	x

2.4.4 Veracity measurements

The software quality model is presented on ISO 25010, which evaluates eight main characteristics: Reliability, Performance Efficiency, Functional Suitability, Compatibility, Maintainability, Portability, Usability, and Security (Hovorushchenko, 2017a). The definition of SRS is crucial to define these characteristics of software quality and impacting methods of quantitative evaluation of software quality (Hovorushchenko, 2017a). Today, quality software measures are conducted through source code, not considering the sufficiency of the SRS for software quality assessment (Hovorushchenko, 2017a).

Simpson and Foltz (2018a), Hovorushchenko (2018), and Feng et al. (2023) published works on models and methodologies to measure Veracity from different perspectives.

Simpson and Foltz (2017) introduced a model to identify the Veracity of the data based on an Enterprise Level Security (ELS) application security model that operates without accounts or passwords, where identity verification is crucial. The shift to claims-based access and privilege raises concerns about the trustworthiness of requesters (Simpson & Foltz, 2017). To address this, individuals and entities are periodically vetted, and their activities between vetting events are monitored to assess trustworthiness, referred to as Veracity (Simpson & Foltz, 2017). The strengths presented were to allow supervisors to mark pre-vetting data as resolved for persons, and the model appropriately addresses non-person entities generating automatic responses (Simpson & Foltz, 2017). Furthermore, Veracity measures can be utilized to meet the self-assessment requirement and offer management insight into insider threats (Simpson & Foltz, 2017). One of the weaknesses was that if there were unfavourable veracities for personal entities, manual resolutions were requested (Simpson & Foltz, 2017). Thus, system developers or decision-makers believe that using automated processes to respond to issues related to individuals' Veracity or trustworthiness is not appropriate or necessary (Simpson & Foltz, 2017).

Hovorushchenko (2018) introduced a novel methodology for evaluating the sufficiency of SRS information for assessing software complexity and quality based on metric analysis results and using ontologies. Developing these theoretical models helps identify deficiencies or sufficiencies of information about Non-Functional characteristics in the SRS (Hovorushchenko et al., 2019).

Ontological models of Non-Functional characteristics of software, developed based on ISO 25010:2011 and ISO 25023:2016 standards, are used to evaluate the extent to which the information in the requirements specifications is suitable for identifying the Non-Functional qualities (Hovorushchenko et al., 2019). These methods involve forming logical conclusions about the sufficiency of SRS information for software quality assessment by identifying rules for each element in the set (Hovorushchenko, 2018). However, this methodology needs to evolve so that software companies can streamline the process of developing SRS through automation, and the accuracy of software quality assessment improves with the addition of Veracity indicators (Hovorushchenko, 2018).

Feng et al. (2023) present a set of functions that model the laser-scanned track formation during cooling and solidification for the additive manufacturing (AM) sector. He outlines the Functional requirements for AM data analytics (DA) software tools, which are essential for defect detection, root cause analysis, and predicting the quality of fabricated parts (Feng et al., 2023). The context for these Functional requirements is based on the data types, which includes standardized definitions of all measured data (Feng et al., 2023). The value of the data consists of its Veracity, which is crucial because it serves as the input for software tools that implement various lifecycle functions in industrial applications (Feng et al., 2023). These functions include design, material selection, process-parameter setting, microstructural analysis, and part property prediction. Identifying these data types and Functional requirements helps AM technology users address challenges by ensuring data accuracy and reliability (Feng et al., 2023).

2.5 Interpreting the findings

Based on the data analysed above, the paper summarized the identification and categorization of Veracity Requirements and explored measurement metrics and techniques.

2.5.1 What are Veracity Requirements in software systems?

This paper addresses the research topic "What are Veracity Requirements in software systems?" by describing the essential qualities that comprise these requirements, offering examples from crucial fields, and examining the significance and influence of Veracity Requirements. The definition and scope are evaluated, the key characteristics and features are emphasized, and the importance of Veracity Requirements in assuring the reliability, completeness, and trustworthiness of software systems is emphasized.

Several case examples demonstrate the significance of Veracity Requirements among domains, emphasizing the crucial part that these requirements play in guaranteeing data accuracy, reliability, and integrity in software systems. Perera et al. (2024) characterize and illustrate the different types of Veracity Requirements pertaining to the dimensions of Veracity using a case study example from the organic domain. The characterization of Veracity Requirements into data, process, financial, regulatory, and cultural dimensions. The case studies were grouped or identified according to each of these dimensions (see Table 3).

Table 3: Veracity Requirements among various domains: Importance, Examples, and Applications.

Perera' Dimensions	Importance of Veracity Requirements	Examples	Domains
Data	Veracity Requirements ensure the reliability and accuracy of data for advanced telescopes and help develop tools to test data Veracity in	Software to optimize the performance of the software system of an advanced telescope (Levi et al., 2016).	Aerospace

	real-time using existing measurements.		
	The data collected from patients must be highly accurate to ensure patient safety and effective treatment, with correct calculations to prevent endangering a patient's life	In medical monitoring software, incorrect calculations can cause danger to a patient's life (Majma & Babamir, 2014).	Healthcare
	The software's ability to detect abnormalities and determine if warnings are necessary underscores the importance of Veracity Requirements in medical devices to maintain high standards of patient care and safety	A reliable monitoring tool software for the electrocardiogram bedside instrument evaluates whether a warning is necessary, demonstrating an arrhythmia detection accuracy rate of up to 95.2% (Yang, 2009).	
Process	Developing Security software requires ensuring requirement integrity, which encompasses reliability, accuracy, and completeness. This requires regularly updating and readjusting security software to avoid hidden attacks.	Security software in the Veracity system is periodically refreshed from secure memory and reconfigured to avoid as-yet undetected threat activity (Simpson & Foltz, 2017)	Cybersecurity
	Throughout the software development lifecycle, Veracity Requirements play a crucial role in overcoming security-related restrictions. When creating safe software, it is imperative to ensure that requirements are verified, complete, and trustworthy.	Model to Quantify Integrity (MQI) at the requirement phase of the software development process aims to address all major constraints regarding secure requirements (Nikhat et al., 2016)	
Financial	No case studies on this SLR are related to this dimension.		
Regulatory	These requirements help ensure that products meet established standards by ensuring the software incorporates models that maintain product quality.	Software that incorporates models to make predictions about defect characterization, identification, and rectification is key to ensuring the products meet the standards (Feng et al., 2023)	Manufacturing
Cultural	Veracity standards play a crucial role in guaranteeing that the system fosters private and secure knowledge sharing while upholding user confidence. This demonstrates how, in culturally sensitive applications, data authenticity and user trust must	In cultural software applications, like those developed for Māori communities system for managing tribal registrations, respecting the sovereignty of whānau	Cultural preservation

	coexist in a balanced way, honouring the control of data by whānau (families) and hapū (tribes).	(family) and hapū (tribe) over their data (Tairea et al., 2023).	

While the scenarios included in this SLR may not cover every aspect described by Luczak-Roesch et al. (2023) and Perera et al. (2023b), as a whole, these case studies highlight the critical function of Veracity Requirements across a range of fields. For software systems to function successfully and be regarded as reliable, data correctness, integrity, and dependability should be guaranteed, regardless of the application—healthcare, manufacturing, aerospace, cybersecurity, or cultural. These kinds of Veracity Requirements draw attention to the various dimensions that are required to guarantee the correct operation and dependability of software systems in various industries.

2.5.2 How can RTD related to Veracity be measured?

In addressing this question, the SLR, RTDQM, and TDQM combined models were identified, as well as some methodologies to measure Veracity and SRS.

Perera et al. (2024) presented a combined model that quantifies RTD for software requirements mapping 33 concepts, where Veracity Requirements would be applicable to adopt (see Figure 2) the model to understand the quantification of Veracity Debt; the analysis revealed several findings:

- Commonly covered concepts include User Need, (Documented) Requirement, Functional Requirement, Context, Risk, and Design/Architectural Decision.
- Partially covered concepts include Requirements Engineering Step, Non-Functional Requirement, Scenario, Quality Attribute, (Prioritized) Requirement, Product Value, and End User Satisfaction Level.
- Not covered concepts include RTD Item, RTD Rectifying Step, Implementation Cost of Architectural/Design Decision, Total Cost of a RE Step, Cost of Rectifying RTD, RTD Interest, New Code Cost associated with RTD, Rework Code Cost associated with RTD, and New Design Cost associated.

A mapping of case studies was conducted to identify the relevant concepts considered and align them with attributes pertinent to Veracity as outlined by Perera et al. (2023a). This analysis revealed consistent themes across various domains, emphasizing the significance of Veracity in diverse contexts. In grey, Veracity Requirements were added to Perera et al.'s model to understand the quantification of Veracity Debt (see Figure 2).

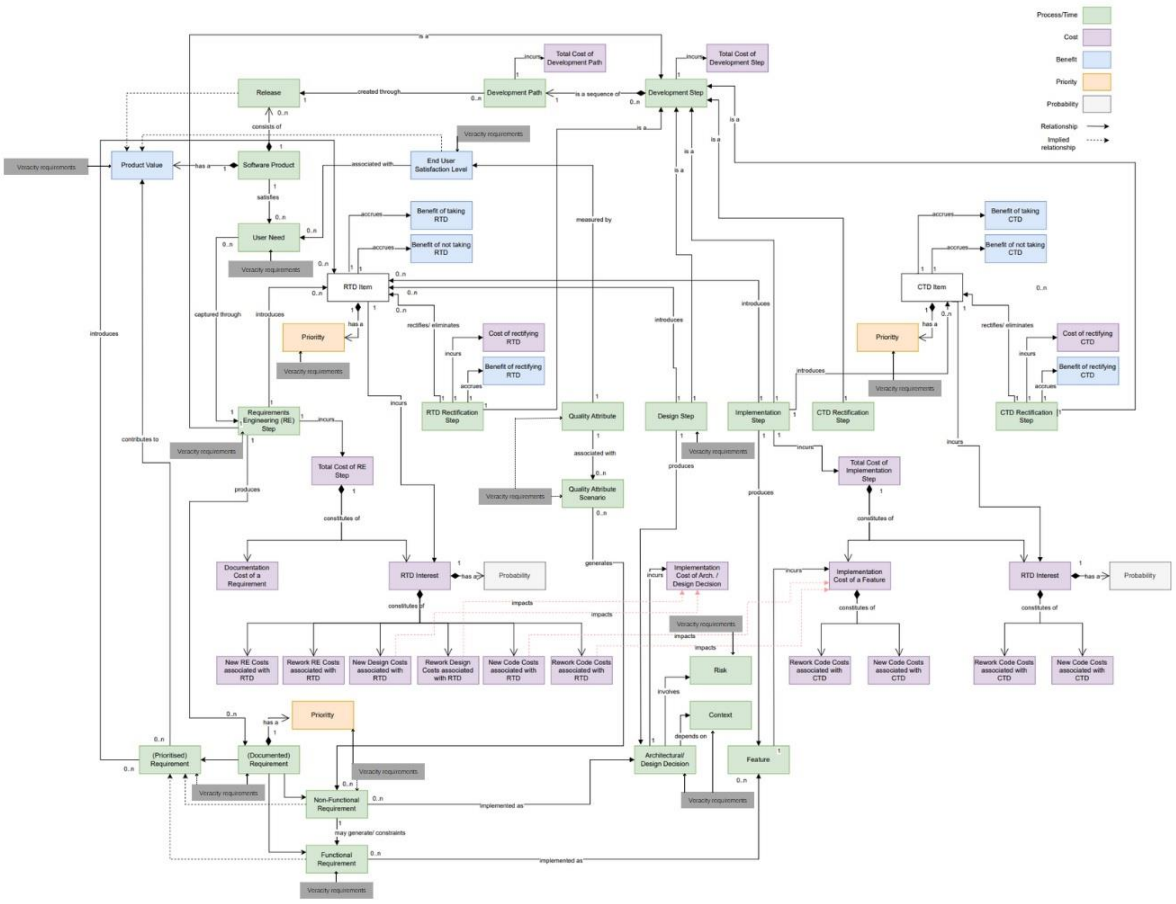

Figure 2: Adopting Perera et al.'s model to include Veracity Requirements to understand the quantification of Veracity Debt.

All case studies show the notion of Priority of a Requirement and emphasize how important it is to prioritize meeting the most important demands first. Setting requirements in order of importance is critical for the proper development and deployment of software systems since it leads to reduced incidents and accidents related to software and increases the Veracity of software quality assessments (Hovorushchenko et al., 2023).

The table lists show methods and methodologies identified on Veracity and SLR measurements, presented by Feng et al. (2023), Hovorushchenko (2018), and Simpson and Foltz (2017), stressing the advantages and disadvantages of each (see Table 4):

Table 4: Overview of techniques and tools for Veracity evaluation.

Techniques/Tool	Author	Metrics/ measurement	Strengths	Weaknesses
Model to identify the Veracity of the Veracity data.	Simpson and Foltz (2017)	A set of entity attributes is formulated linguistically, and the results are counterclaim for a non-person and	Automation about the Veracity of the Veracity data responses related to ELS, especially for non-person entities.	If there are unfavourable veracities for personal entities, manual resolutions are requested.

		counterclaim for a person.		
Theoretical methodology for evaluating the sufficiency of SRS information for assessing software complexity and quality	Hovorushchenko (2018)	A set of software quality measures has the weights to evaluate the sufficiency of the SRS for software quality assessment based on the weighted ontology.	Identify deficiencies or sufficiencies of information about Non-Functional characteristics in the specification of requirements for specific software. Software companies can automate the procedure of SRS development	The methodology needs to evolve and mature through various stages of enhancement and validation.
A set of functions that model the laser-scanned track formation during cooling and solidification for the additive manufacturing (AM) sector	Feng et al. (2023)	A set of Functional requirements formulated linguistically, where each function relates its input data to output data.	Standardization of the definitions of all measured data.	Data's value is derived from its Veracity, which still cannot be measured precisely.

Veracity and SRS measures are addressed in the table through a comparative analysis of different approaches. Simpson's data Veracity identification technique has advantages in automation, but it needs human interaction for personal entities (Simpson & Foltz, 2017). Feng's functions for the additive manufacturing industry encounter difficulties in accurately measuring data Veracity (Feng et al., 2023). Hovorushchenko's theoretical methodology needs more refining and validation for full automation. However, it helps identify deficiencies or sufficiencies of information about Non-Functional characteristics in the SRS while standardizing data measures (Hovorushchenko, 2018).

3. Discussion

The SLR addressed the first main question of this paper, which was to assess aspects, including definition and scope, importance, and impact, as well as attributes and features for Veracity Requirements.

The same as Dave et al. (2022) outlined a definition and a taxonomy for Implicit Requirements (IMR), and ISO 29148 provides guidelines for the comprehensive development of requirements, including their characteristics and attributes; the same concept can be applied to define Veracity Requirements as another type of requirement. According to the International Organization for Standardization (ISO) 24765:2017, where the standard vocabulary for systems and software engineering was developed, 23 requirement definitions are registered (IEEE, 2017). These definitions cover different aspects and types of requirements such as: 'requirements allocation', 'requirements attributes', 'requirements documentation', and 'requirements management' from standard descriptions to management in engineering contexts (IEEE, 2017). This ISO defines a requirement as a statement that conveys a need

and its related constraints and conditions (IEEE, 2017). Furthermore, ISO 29148 provides a united treatment of the processes and products involved in engineering requirements throughout the life cycle of systems and software (IEEE., 2018). This ISO outlines the guidelines for the comprehensive development of requirements that contain the characteristics and attributes based on Requirements information items, Business requirements specification, Stakeholder requirements specification, System requirements specification, and Software requirements specification (SRS) (IEEE., 2018). A requirement can be defined by identifying characteristics and attributes that allow it to be formulated and detailed in a way that can be validated and verified. By undertaking this, different types of requirements can be created, such as Quality requirements, Usability requirements, and Interface requirements (IEEE., 2018). Although there is a list of 23 definitions related to 'requirements' based on ISO 24765, there is no definition specific to Veracity Requirements. However, ISO 29148 provides how to define a requirement by identifying a list of attributes and characteristics that allow the creation of different types of requirements. An example of this is what Dave et al. (2022) did by creating one type of requirement and its taxonomy, IMR. The same concept can be used to define Veracity Requirements as distinct types of requirements. Veracity Requirements can be characterized and be related to truth, trust, authenticity, provenance, and integrity of data and human interactions (Perera et al., 2024). These attributes ensure that data accurately represents real-world scenarios and remains trustworthy over time. This is the first attempt definition that can contribute to a deeper understanding of Veracity Requirements in software systems.

Perera et al. (2023a) introduced the dimensions of the Veracity Requirements: data, process, regulatory, financial, and cultural. However, the financial dimension identified by Perera et al. (2023a) needs further exploration because it was not identified in the case studies presented. Thus, more case studies on this dimension are required to finally be considered a dimension.

As Simpson and Foltz (2018a) mentioned, Veracity Requirements are particularly necessary in domains like healthcare, finance, and government. The case studies presented across various domains, including medical monitoring software, quality assurance in manufacturing, aerospace systems, cybersecurity, and cultural data management in Māori software, further exemplify the varied applications and importance of Veracity Requirements. These examples emphasize the importance of applying Veracity Requirements, which contribute to more reliable software systems.

The second main question tried to understand how Veracity Requirements can be integrated into the RTD quantification model and address measurement metrics and methodological approaches to VRTD. Based on the case studies identified in this SLR and mapped to the 33 concepts by Perera J (2023) on RTDQM and TDQM, to understand the quantification of Veracity Debt, the analysis revealed the importance of considering Veracity Requirements mainly on the concept 'Priority of a Requirement'. Implementing Veracity Requirements is essential, except in cases where RTD is already being managed effectively. Practitioners can discover the costs and benefits of RTD, acting as a reference for getting new quantification approaches that count how Veracity Requirements could impact RTD.

Through this SLR, three types of attempts to measure Veracity and SRS were summarized by identifying strengths and weaknesses in various domains. These approaches to Veracity Requirements and SRS provide the following benefits: automation, though it still needs human intervention for personal entities (Simpson & Foltz, 2017); theoretical methodology, though it still needs to be refined for complete automation, successfully identifies if the information is insufficient in Non-Functional aspects of SRS (Hovorushchenko, 2018), and functions that standardize data measurements in the additive manufacturing sector (Feng et al., 2023). However, the methodology introduced by Hovorushchenko is the most accurate for transferring to Veracity Requirements and VRTD by evaluating the sufficiency of information for software quality assessment for Non-Functional requirements. The methodology

presented by Hovorushchenko (2018) could be beneficial when implemented during or before the formalization of requirements identified in Perera's Concept Map for RTDQM. By evaluating the sufficiency of the information for software quality assessment, this approach can help identify and reduce ambiguous requirements, particularly for Non-Functional ones (see Figure 3). This paper encourages researchers to investigate this methodology to assess whether there are sufficient or insufficient Veracity Requirements.

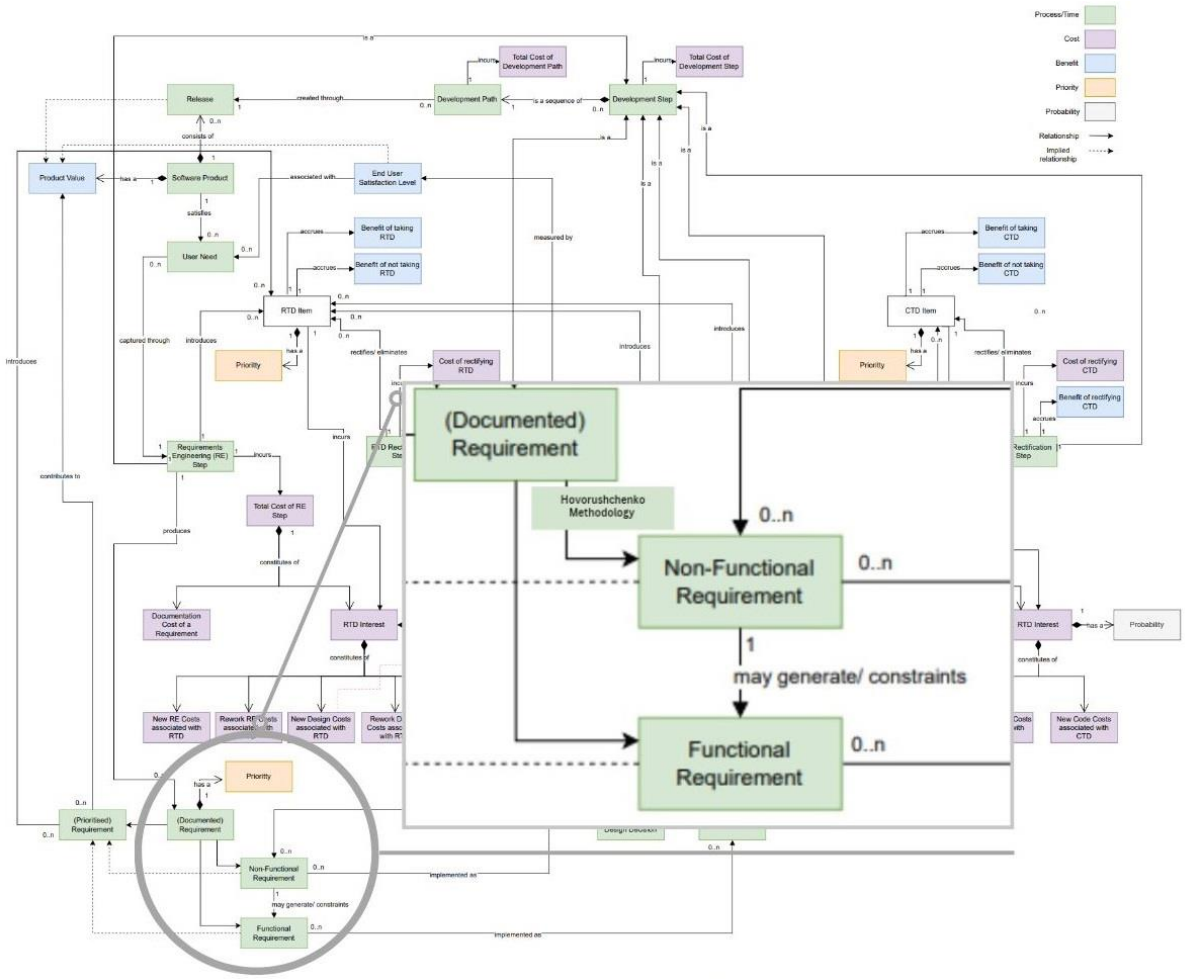

Figure 3: Enhancing Non-Functional requirements formalization with Hovorushchenko's Methodology in Perera's RTDQM.

Some limitations are valid to mention that sourced publications were only from the Scopus database; however, in most cases, papers found in other databases such as IEEE, ACM, and Web Science are available on SCOPUS (Guz & Rushchitsky, 2009), backward and forward snowballing was conducted to complement the search (Wohlin, 2014). Furthermore, due to the limited number of eligible papers, the analysis was based on all 21 available papers, which may affect the generalizability and robustness of the findings.

4. Conclusion

In conclusion, this SLR addresses the definition, importance, attributes, and main domains of Veracity Requirements. By applying the guidelines from ISO 29148 for definitions of requirements based on characteristics and attributes, Veracity Requirements are considered a new type of requirement based on attributes. This SLR identified Veracity Requirements as another type of requirement to consider

when developing software. Veracity Requirements are characterized by accuracy, reliability, consistency, and inclusiveness. These attributes guarantee that data accurately corresponds to real-world scenarios and remains trustworthy over time, especially in domains like manufacturing, healthcare, cybersecurity, and aerospace, which impact the accuracy of software outcomes.

By analysing the combination of the RTDQM and TDQM models to illustrate the cascading impact of RTD on development phases such as design and implementation, it becomes clear that measuring and managing Veracity Requirements is crucial. The identification of various methodologies, techniques, and tools for measuring Veracity Debt in SRSs highlights the challenges in achieving accurate Veracity Requirements. Hovorushchenko's (2018) theoretical methodology, which evaluates Non-Functional requirements based on the sufficiency of information in SRS for software quality assessment, offers a potential approach that could be integrated with Perera's models to measure RTD and VRTD. Implementing Hovorushchenko's methodology during or before the formalization of Non-Functional requirements, as identified in Perera's Concept Map for RTDQM, can help in evaluating the sufficiency of information for those requirements, thereby reducing ambiguous requirements.

Despite software systems' increasing complexity, it is vital to continue investigating approaches to developing tools to measure Veracity Requirements. This would contribute to more accurate and trustworthy software systems and meet user concerns about the transparency of software-supported decisions in different domains.

Acknowledgment

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

References

Belter, C. W. (2015). *Bibliometric indicators: opportunities and limits*. *J Med Libr Assoc*, *103*(4), 219-221. https://doi.org/10.3163/1536-5050.103.4.014

Blincoe, K., Luczak-Roesch, M., Miller, T., & Galster, M. (2023). *Human-centric literature review on trust*. https://veracity.wgtn.ac.nz/wp-content/uploads/2023/09/4_Human-centric-literature-review-on-trust.pdf

Dave, D., Celestino, A., Varde, A. S., & Anu, V. (2022). *Management of Implicit Requirements Data in Large SRS Documents: Taxonomy and Techniques* [Article]. *SIGMOD Record*, *51*(2), 18-29. https://doi.org/10.1145/3552490.3552494

Dietrich, J., Rasheed, S., & Jordan, A. (2023). *Discovering security blind spots in software supply chains*. Veracity Lab. https://veracity.wgtn.ac.nz/wp-content/uploads/2023/09/7_Discovering-security-blindspots-in-software-supply-chains.pdf

Feng, S. C., Moges, T., Park, H., Yakout, M., Jones, A. T., Ko, H., & Witherell, P. (2023). *Functional Requirements of Software Tools for Laser-Based Powder Bed Fusion Additive Manufacturing for Metals* [Article]. *Journal of Computing and Information Science in Engineering*, *23*(3), Article 031005. https://doi.org/10.1115/1.4054933

Galster, M., & Dietrich, J. (2023). *Understanding software provenance mechanisms in the industry*. https://veracity.wgtn.ac.nz/wp-content/uploads/2023/09/3_Understanding-software-provenance-mechanisms-in-industry.pdf

Guz, A. N., & Rushchitsky, J. J. (2009). *Scopus: A system for the evaluation of scientific journals*. *International Applied Mechanics*, *45*(4), 351-362. https://doi.org/10.1007/s10778-009-0189-4

Hovorushchenko, T. (2017a). *Forming the logical conclusion about sufficiency of information of software requirements specification for software quality assessment by ISO 25010:2011*. 2017 IEEE 1st Ukraine Conference on Electrical and Computer Engineering, UKRCON 2017 - Proceedings,

Hovorushchenko, T. (2017b). *The rules and method of forming the logical conclusion about sufficiency of information for software metric analysis*. Proceedings of the 12th International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT 2017,

Hovorushchenko, T. (2018). *Methodology of evaluating the sufficiency of information for software quality assessment according to ISO 25010* [Article]. *Journal of Information and Organizational Sciences*, *42*(1), 63-85. https://doi.org/10.31341/jios.42.1.4

Hovorushchenko, T., Medzatyi, D., Voichur, Y., & Lebiga, M. (2023). *Method for forecasting the level of software quality based on quality attributes* [Article]. *Journal of Intelligent & Fuzzy Systems*, *44*(3), 3891-3905. https://doi.org/10.3233/JIFS-222394

Hovorushchenko, T., Pavlova, O., & Boyarchuk, A. (2019). *Modeling of nonfunctional characteristics of the software for selection of accurate scope of information for their evaluation*. CEUR Workshop Proceedings,

IEEE. (2017). *ISO/IEC/IEEE International Standard - Systems and software engineering--Vocabulary*. *IEEE*. https://doi.org/10.1109/IEEESTD.2017.8016712

IEEE. (2018). *ISO/IEC/IEEE International Standard - Systems and software engineering - Life cycle processes - Requirements engineering. IEEE*. https://doi.org/10.1109/IEEESTD.2018.8559686

Khan, K. S., Kunz, R., Kleijnen, J., & Antes, G. (2003). *Five steps to conducting a systematic review*. *Journal of the Royal Society of Medicine*, *96*(3), 118-121. https://doi.org/10.1258/jrsm.96.3.118

Levi, J., Glassman, T., Farey, M., & Liepmann, T. (2016). *Data veracity checks for the alignment of the JWST optical telescope element*. Proceedings of SPIE - The International Society for Optical Engineering,

Luczak-Roesch, M., Galster, M., & Shedlock, K. (2023). *The Veracity Grand Challenge in Computing: A Perspective from Aotearoa New Zealand*. https://veracity.wgtn.ac.nz/wp-content/uploads/2023/06/The-Veracity-Grand-Challenge-in-Computing_-A-Perspective-from-Aotearoa-New-Zealand.pdf

Majma, N., & Babamir, S. M. (2014). *Specification and Verification of Medical Monitoring System Using Petri-nets*. *Journal of Medical Signals and Sensors*, *4*(3). https://doi.org/10.4103/2228-7477.137769

Moher, D., Liberati, A., Tetzlaff, J., & Altman, D. G. (2010). *Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement*. *International Journal of Surgery*, *8*(5), 336-341. https://doi.org/https://doi.org/10.1016/j.ijsu.2010.02.007

Mongeon, P., & Paul-Hus, A. (2016). *The journal coverage of Web of Science and Scopus: a comparative analysis*. *Scientometrics*, *106*(1), 213-228. https://doi.org/10.1007/s11192-015-1765-5

Nikhat, P., Kumar, N. S., & Khan, M. H. (2016). *Model to quantify integrity at requirement phase*. *Indian Journal of Science and Technology*, *9*(29), Article 89280. https://doi.org/10.17485/ijst/2016/v9i29/89280

Perera, J., Tempero, E., Blincoe, K., & Tu, Y.-C. (2024). *Modelling the Quantification of Requirements Technical Debt. Requirements Engineering* (Veracity Lab, Issue.

Perera, J., Tempero, E., Tu, Y.-C., & Blincoe, K. (2023a). *Quantifying Requirements Technical Debt: A Systematic Mapping Study and a Conceptual Model*.

Perera, J., Tempero, E., Tu, Y.-C., & Blincoe, K. (2023b). *Towards quantifying technical debt for veracity requirements*. https://veracity.wgtn.ac.nz/wp-

content/uploads/2024/03/8_Towards-quantifying-technical-debt-for-veracity-requirements.pdf
Simpson, W. R., & Foltz, K. E. (2017). *Enterprise level security: Insider threat counter-claims*. Lecture Notes in Engineering and Computer Science,
Simpson, W. R., & Foltz, K. E. (2018a). *Insider threat metrics in Enterprise Level security* [Article]. *IAENG International Journal of Computer Science*, *45*(4), 610-622. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85057422451&partnerID=40&md5=c0975b9c8f663e1aa4c6dea35f831dc2
Simpson, W. R., & Foltz, K. E. (2018b). *Insider Threat Veracity Issues*. In *Transactions on Engineering Technologies: World Congress on Engineering and Computer Science 2017* (pp. 303-315). Springer Singapore. https://doi.org/10.1007/978-981-13-2191-7_21
Tairea, B., Pilbrow, C., & Reeves, S. (2023). *Secure digital technologies for tribal registries*. https://veracity.wgtn.ac.nz/wp-content/uploads/2023/09/6_Secure-digital-technologies-for-tribal-registries.pdf
Wohlin, C. (2014). *Guidelines for snowballing in systematic literature studies and a replication in software engineering* Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, London, England, United Kingdom. https://doi.org/10.1145/2601248.2601268
Yang, L. (2009). *Development and application of ECG bedside monitor*. ICEMI 2009 - Proceedings of 9th International Conference on Electronic Measurement and Instruments.,