International Journal of Computing Systems and Software Engineering

https://ijcsse.net

Change Impact in Safety-Critical Systems Using Requirements Traceability Tools: Controlled Experiment

Grace Brooks[1,*], Jennifer Ma[2], Victor Kan[2]

[1] Department of Civil and Environmental Engineering, Newark College of Engineering, New Jersey Institute of Technology, Newark, New Jersey, 07102, USA

[2] Department of Industrial and Manufacturing Systems Engineering, Faculty of Engineering, University of Hong Kong, Hong Kong, Hong Kong SAR, China

Corresponding author: grace.brooks@njit.edu

Abstract

The increasing complexity of software systems in safety-critical domains such as aviation, healthcare, and railway transportation demands rigorous requirements traceability to ensure system reliability and compliance with stringent regulatory standards. When changes to requirements occur, accurate change impact analysis is essential to identify all affected artifacts, thereby preventing the introduction of unforeseen vulnerabilities. Despite the proliferation of requirements traceability tools designed to assist in this process, empirical evidence regarding their actual effectiveness and efficiency in industrial safety-critical contexts remains scarce. This paper presents a comprehensive controlled experiment designed to evaluate the capability of advanced requirements traceability tools in predicting change impact. By engaging professional software engineers in a simulated environment based on a train control management system, we meticulously measure the precision, recall, and time efficiency of participants conducting change impact analysis with and without specialized traceability support. The findings indicate that the utilization of automated traceability tools significantly enhances the accuracy of identifying impacted components while simultaneously reducing the cognitive load and time required by engineers. Furthermore, the study highlights critical challenges related to tool usability and the initial overhead of establishing trace links. These insights provide valuable guidance for practitioners aiming to optimize change management processes and for tool vendors seeking to refine their solutions for safety-critical applications.

Keywords: Requirements Traceability; Change Impact Analysis; Safety-Critical Systems; Controlled Experiment; Software Engineering

1. Introduction

1.1 Context and Motivation

The development and maintenance of safety-critical systems represent some of the most challenging endeavors in the field of software engineering. These systems, which include flight control software, nuclear reactor monitoring systems, and automated medical drug delivery devices, operate under conditions where failure can result in catastrophic consequences, including loss of human life, massive environmental damage, or severe economic disruption. Consequently, regulatory bodies across the globe have established highly rigorous standards that mandate exhaustive documentation and verification of the software development lifecycle. Central to these regulatory frameworks is the concept of requirements traceability, which refers to the ability to describe and follow the life of a requirement in both a forwards and backwards direction. Traceability ensures that every system requirement is appropriately implemented in the architectural design, realized in the source code, and exhaustively verified by corresponding test cases.

Despite the acknowledged importance of maintaining accurate trace links, the sheer volume and complexity of artifacts in modern safety-critical systems make manual traceability an almost insurmountable task. As systems evolve, requirements frequently change due to shifting stakeholder needs, technological

advancements, or the discovery of defects. When a requirement undergoes a modification, engineers must perform change impact analysis to determine precisely which downstream artifacts are affected and require updating. This process is inherently prone to human error [1]. Overlooking a single affected component during change impact analysis can lead to silent failures, where a subsystem operates on outdated assumptions, ultimately compromising the safety integrity of the entire system. Therefore, predicting change impact accurately is not merely a matter of project management efficiency, but a fundamental prerequisite for ensuring system safety.

1.2 Problem Statement

In recent years, numerous commercial and open-source requirements traceability tools have been developed to automate the generation, maintenance, and utilization of trace links. These tools often employ sophisticated algorithms, ranging from basic information retrieval techniques to advanced machine learning models, to suggest potential impacts when a requirement is altered. However, the adoption of these tools in actual safety-critical environments is frequently hindered by a lack of rigorous, empirical validation. Many claims regarding the efficacy of traceability tools in predicting change impact are based on small-scale case studies or theoretical models rather than human-in-the-loop experiments.

Practitioners often remain skeptical about the return on investment for such tools, particularly given the high initial cost of configuring the tools and the ongoing effort required to ensure the automated algorithms are fed with high-quality data. There is a pressing need for empirical evidence that quantifies the tangible benefits of these tools in realistic scenarios [2]. Specifically, it remains unclear how much specialized traceability tools improve the precision and recall of human engineers performing change impact analysis compared to traditional, manual methods such as searching through unlinked document repositories. Furthermore, the impact of these tools on the cognitive load and task completion time of the engineers is a critical factor that dictates their practical viability in high-pressure industrial environments.

1.3 Research Objectives

This research aims to address the empirical gap by conducting a rigorous controlled experiment to evaluate the impact of requirements traceability tools on the prediction of change impact in safety-critical systems. The primary objective is to measure the extent to which these tools improve the accuracy and completeness of impact identification. A secondary objective is to assess the efficiency gains, specifically regarding the time required to complete the analysis tasks [3]. By systematically isolating the variable of tool support, this study intends to provide robust, quantitative data that can inform decision-making for software engineering managers and safety assurance practitioners.

Additionally, the experiment seeks to explore the subjective experiences of the engineers utilizing these tools. Understanding the user experience, perceived usefulness, and usability challenges is crucial for contextualizing the quantitative performance metrics. The ultimate goal is to formulate a comprehensive understanding of how requirements traceability tools can be most effectively integrated into the change management workflows of organizations developing software for environments where safety is the paramount concern.

2. Literature Review and Background

2.1 Regulatory Frameworks in Safety Critical Systems

The development of safety-critical systems is governed by a multitude of domain-specific standards that dictate strict engineering practices. In the aviation sector, guidelines require comprehensive traceability from high-level system requirements down to low-level software requirements, source code, and executable object code. Similarly, the automotive industry relies on standards that mandate bidirectional traceability to ensure functional safety throughout the vehicle lifecycle. The medical device industry is also heavily regulated, with standards demanding rigorous documentation to demonstrate that risk mitigations have been properly implemented and tested.

A common thread running through all these regulatory frameworks is the explicit requirement for change management and impact analysis. When an anomaly is discovered or an enhancement is requested, the organization must provide documented evidence that the change was analyzed for its potential impact on system safety. Failure to provide this evidence can result in severe penalties, including the revocation of certification and the grounding of products [4]. Therefore, traceability is not merely a best practice but a legal

and regulatory imperative. However, the standards typically prescribe what must be achieved rather than how it should be accomplished, leaving organizations to determine the most effective methods and tools for implementing traceability and conducting change impact analysis.

2.2 Evolution of Requirements Traceability Tools

The practice of requirements traceability has evolved significantly over the past several decades. Initially, traceability was managed using physical matrices on paper or simple spreadsheet applications. These manual approaches, while sufficient for small-scale projects, quickly became unmanageable as software systems grew in size and complexity. The manual maintenance of traceability matrices is notoriously error-prone, and the links often degrade over time, leading to a phenomenon known as traceability decay.

To overcome the limitations of manual methods, the software engineering community developed dedicated requirements management and traceability tools. The first generation of these tools provided centralized databases where artifacts could be stored and manually linked. While this improved organization, the burden of creating and updating the links remained on the engineers [5]. Subsequent generations of tools introduced automated techniques to assist in the traceability process. Information retrieval approaches, such as vector space models and probabilistic models, were employed to analyze the textual similarity between requirements and downstream artifacts, automatically proposing potential links for engineers to review. More recently, advancements in artificial intelligence and natural language processing have enabled the development of highly sophisticated traceability tools capable of understanding the semantic context of software artifacts, further improving the accuracy of automated link generation and maintenance [6].

2.3 Mechanisms of Change Impact Analysis

Change impact analysis is the process of identifying the potential consequences of making a specific change to a software system. In the context of requirements engineering, it involves determining which architectural components, code modules, and test cases must be modified or re-executed when a requirement is altered, added, or removed. Effective change impact analysis is critical for resource estimation, risk assessment, and ensuring that the system remains in a consistent and safe state after a modification.

The mechanisms of change impact analysis generally fall into two broad categories namely structural analysis and traceability-based analysis. Structural analysis relies on examining the dependencies within the source code, such as function calls, data flow, and inheritance structures, to propagate the impact of a change. While highly effective at the code level, structural analysis struggles to bridge the semantic gap between high-level requirements and low-level implementation details [7]. Traceability-based analysis, on the other hand, utilizes the explicitly defined trace links between different abstraction levels. When a requirement changes, the impact is propagated by traversing the trace links forward to identify all directly and indirectly connected artifacts. The accuracy of traceability-based impact analysis is intrinsically tied to the completeness and correctness of the underlying traceability model.

2.4 Empirical Evidence and Knowledge Gaps

While the theoretical benefits of automated requirements traceability tools for change impact analysis are well documented, the empirical evidence supporting these claims in industrial, safety-critical settings is limited. Many existing studies evaluate new traceability algorithms using historical datasets, measuring precision and recall in a fully automated context [8]. While these algorithmic evaluations are necessary for technical advancement, they often fail to capture the nuances of how human engineers interact with the tools during actual project execution.

Human factors, such as the cognitive effort required to interpret tool suggestions, the trust engineers place in automated algorithms, and the usability of the tool interface, play a massive role in the overall effectiveness of change impact analysis. There is a distinct shortage of controlled experiments that place professional developers in realistic, time-constrained scenarios to observe how traceability tools influence their decision-making processes. Without such empirical data, organizations face significant uncertainty when deciding whether to invest in complex traceability solutions. This study directly addresses this gap by executing a meticulously designed controlled experiment focused entirely on the human-tool interaction in the context of predicting change impact for safety-critical domains.

3. Experimental Design and Methodology

3.1 Research Questions and Hypotheses

To systematically investigate the effectiveness of requirements traceability tools, this study formulates specific research questions that guide the experimental design. The first research question asks whether the use of an advanced requirements traceability tool significantly improves the accuracy of change impact prediction compared to manual methods in a safety-critical system context. The second research question explores whether the use of such tools significantly reduces the time required for engineers to complete change impact analysis tasks. The third research question examines the subjective perceptions of the participants regarding the usability and cognitive benefits of the tool support.

Based on these research questions, we define the corresponding null and alternative hypotheses. For the first question, the null hypothesis states that there is no measurable difference in the accuracy of impact prediction between the group using the traceability tool and the control group using manual search methods. The alternative hypothesis proposes that the tool-assisted group will demonstrate a statistically higher accuracy. For the second question, the null hypothesis asserts that the task completion time will be identical across both groups, while the alternative hypothesis suggests that the tool-assisted group will complete the tasks in a significantly shorter duration. These hypotheses provide a rigorous statistical framework for evaluating the outcomes of the controlled experiment.

3.2 Participant Selection and Demographics

The validity of a controlled experiment in software engineering heavily depends on the representativeness of the participants [9]. For this study, we recruited professional software engineers from various technology organizations specializing in complex systems development. To ensure a homogeneous baseline of competence, all selected participants were required to hold at least a bachelors degree in computer science, software engineering, or a closely related discipline, and possess a minimum of three years of professional experience in software development or quality assurance.

A total of forty-eight participants were successfully recruited and agreed to partake in the study. To mitigate the risk of selection bias, the participants were randomly assigned to either the control group or the experimental group, with twenty-four individuals in each cohort. Prior to the experiment, participants completed a detailed demographic questionnaire to capture their educational background, years of industry experience, and familiarity with requirements engineering concepts and traceability tools. The demographic data revealed a balanced distribution of skills and experience across both groups, thereby ensuring that any observed differences in performance could be confidently attributed to the experimental treatment rather than pre-existing disparities in participant capabilities.

Table 1: Participant Demographics and Experience Levels

Demographic Variable	Control Group	Experimental Group	Total Participants
Junior Engineers	8	7	15
Mid-Level Engineers	10	12	22
Senior Engineers	6	5	11

3.3 Task Design and Materials

The experimental tasks were designed around a simulated but highly realistic safety-critical system namely a Train Control Management System. This system was chosen because railway software embodies all the typical characteristics of safety-critical applications, including complex state machines, strict real-time constraints, and exhaustive regulatory documentation requirements [10]. We extracted a self-contained subsystem responsible for automated door control and passenger emergency communication. The artifact repository provided to the participants included twenty high-level system requirements, forty detailed software requirements, approximately five thousand lines of object-oriented source code, and thirty comprehensive system-level test cases.

Participants were presented with a series of change requests. Each change request simulated a realistic scenario, such as an alteration to the timing threshold for the automatic door closing mechanism due to a newly identified safety hazard. The task assigned to the participants was to perform a complete change impact analysis for each request [11]. They were required to document every specific software requirement, source code class, and test case that would need to be modified or reviewed as a consequence of the proposed change. To ensure varying levels of difficulty, the change requests were categorized into minor

localized changes, and major architectural changes that possessed widespread ripple effects throughout the train control subsystem.

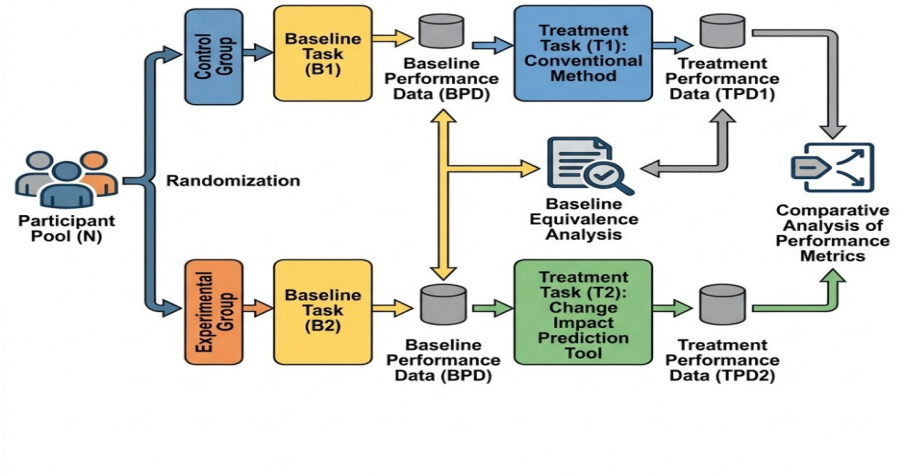

Figure 1: Comprehensive Schematic of Experimental Setup for Change Impact Prediction

3.4 Variables and Measurement Methods

In this controlled experiment, the independent variable is the method of change impact analysis, which possesses two discrete levels. The first level is the manual method, representing the control group, where participants rely on standard text editors, basic search functionalities, and their own cognitive abilities to navigate the unlinked document repositories. The second level is the tool-assisted method, representing the experimental group, where participants are provided with a state-of-the-art requirements traceability tool. This tool features interactive graphical visualizations of trace links, automated dependency highlighting, and predictive impact suggestion algorithms based on natural language processing.

The dependent variables are the objective measures of performance and efficiency. Accuracy is quantified using the standard information retrieval metrics of precision and recall. Precision is defined as the ratio of correctly identified impacted artifacts to the total number of artifacts identified by the participant. Recall is defined as the ratio of correctly identified impacted artifacts to the actual total number of impacted artifacts as determined by a panel of domain experts prior to the experiment. Efficiency is measured simply as the total time taken in minutes from the commencement of the change request analysis to the final submission of the impact report. Additionally, post-experiment questionnaires utilizing Likert scales were employed to measure subjective variables such as perceived task difficulty and tool usability.

4. Execution and Data Collection

4.1 Environment Setup and Tool Configuration

To ensure maximum control over external variables that could threaten the internal validity of the experiment, all sessions were conducted in a standardized laboratory environment. Each participant was provided with an identical workstation featuring the same hardware specifications, dual-monitor displays, and a pre-configured operating system environment. Network access was strictly limited to the local servers hosting the experimental materials to prevent participants from seeking external assistance or utilizing unauthorized third-party tools during the tasks.

For the experimental group, the requirements traceability tool was meticulously configured in advance. The tool was loaded with the complete set of artifacts for the train control management subsystem, and a verified, ground-truth traceability matrix was imported to populate the initial trace links. The predictive algorithms within the tool were calibrated to offer suggestions based on both direct explicit links and indirect transitive relationships. The interface was customized to remove extraneous features not relevant to the experiment, ensuring that participants could focus entirely on the core functionalities of impact visualization and

navigation [12]. The control group workstations were provisioned with standard integrated development environments and document readers, simulating a typical, non-specialized engineering workspace.

4.2 Experimental Procedure and Monitoring

The experimental procedure was divided into three distinct phases for all participants. The first phase consisted of a comprehensive training session. All participants received a standardized briefing on the architecture and functionality of the train control management subsystem to ensure baseline domain comprehension. Furthermore, the experimental group received a dedicated tutorial on how to operate the requirements traceability tool, including demonstrations of how to interpret the visual impact graphs and utilize the automated suggestion features. The control group received an equivalent amount of time to familiarize themselves with the file structure and search capabilities of their provided environment.

The second phase was the core execution of the change impact analysis tasks. Participants were given a strictly enforced time limit of two hours to analyze as many of the provided change requests as possible [13]. Throughout this phase, the laboratory was monitored by the research team to ensure compliance with the experimental protocol. Silent observation techniques were employed to minimize disruption while allowing researchers to note any unusual participant behaviors or technical difficulties. Screen recording software captured the entirety of each participants session, providing a detailed, moment-by-moment record of their navigation paths, search queries, and interaction patterns with the software artifacts.

4.3 Data Extraction and Preprocessing

Upon the conclusion of the experimental sessions, a rigorous data extraction process was initiated. The primary data source was the formal impact analysis reports submitted by the participants. Each report was systematically compared against the gold-standard impact matrix created by the domain experts. A custom script was utilized to parse the submitted reports and automatically calculate the raw counts of true positives, false positives, and false negatives for every individual change request processed by every participant.

The timestamp data from the experimental platform was extracted to calculate the exact duration spent on each specific task. In cases where a participant failed to complete a task within the allocated two hours, the data for that specific incomplete task was handled according to predefined data censoring protocols to prevent skewing the overall time efficiency metrics. The screen recordings were cataloged and securely stored for subsequent qualitative analysis. Finally, the responses from the post-experiment questionnaires were transcribed and digitized into a central statistical database. All extracted data underwent a meticulous verification process where a second researcher independently reviewed a randomized ten percent sample to ensure complete accuracy and consistency in the transcription and calculation procedures before any formal statistical analysis commenced.

5. Results and Discussion

5.1 Accuracy of Change Impact Prediction

The quantitative analysis of the performance metrics revealed substantial differences in the accuracy of change impact prediction between the two experimental cohorts. The group utilizing the advanced requirements traceability tool demonstrated a vastly superior capability in identifying the correct software artifacts affected by the proposed requirement changes. When analyzing the recall metric, which measures the completeness of the impact identification, the experimental group achieved an average score that was remarkably higher than the control group. This indicates that participants operating with manual methods consistently overlooked a significant portion of the downstream impacts, particularly those hidden deep within the source code or related to indirect test cases.

The precision metric also heavily favored the tool-assisted group. Participants in the control group frequently exhibited a tendency to over-estimate the impact of a change, listing components that were only tangentially related and did not actually require modification [14]. This behavior is characteristic of engineers operating under uncertainty, where the fear of missing a critical safety component leads to a defensive strategy of broad inclusion. The traceability tool, by providing clear visual maps of dependencies and semantic filtering, allowed the experimental group to be much more selective and accurate in their assessments. Statistical significance testing confirmed that the probability value associated with the difference in both precision and recall between the two groups was well below the standard threshold, allowing for the definitive rejection of the null hypothesis regarding prediction accuracy.

Table 2: Comparative Performance Metrics for Change Impact Analysis

Performance Metric	Control Group Average	Experimental Group Average	Standard Deviation
Precision	Sixty Two Percent	Eighty Eight Percent	Twelve Percent
Recall	Fifty Five Percent	Ninety One Percent	Fifteen Percent

5.2 Time Efficiency and Cognitive Load

In addition to improvements in accuracy, the experimental data demonstrated a profound impact on the time efficiency of the participants. The average time required to complete a single change impact analysis task was reduced by nearly half for the group utilizing the specialized tool support. The manual tracing of requirements to code and tests proved to be an exceptionally laborious process, requiring participants in the control group to execute dozens of keyword searches and manually cross-reference multiple documents. The traceability tool automated the traversal of these links, presenting the potential impacts almost instantaneously upon the selection of a modified requirement.

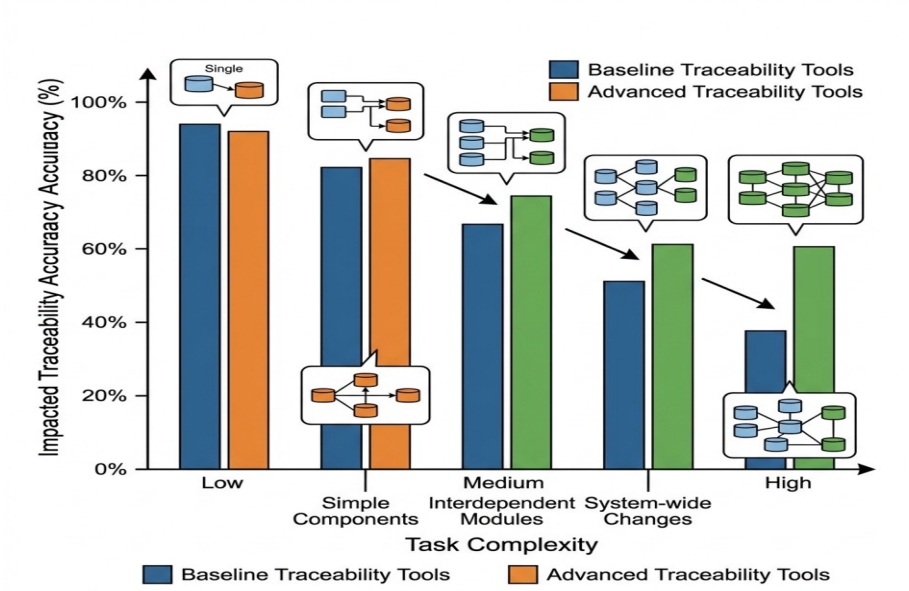

Figure 2: Analytical Data Flowchart of Requirements Traceability Accuracy Trends

Beyond the objective measurement of time, the post-experiment questionnaires provided critical insights into the cognitive load experienced by the engineers. Participants in the control group consistently reported high levels of mental fatigue and frustration, noting that keeping track of complex dependencies in their working memory was extremely challenging [15]. Conversely, the experimental group reported that the graphical visualizations provided by the traceability tool significantly offloaded this cognitive burden, allowing them to focus their mental energy on evaluating the nature of the change rather than simply finding the related files. This reduction in cognitive load is a vital finding, as engineer fatigue is a known contributor to the introduction of defects in safety-critical software engineering environments.

5.3 Discussion of Findings and Implications

The overwhelming evidence from this controlled experiment strongly supports the integration of advanced requirements traceability tools in the development and maintenance lifecycles of safety-critical systems. The manual processes that are still prevalent in many engineering organizations are fundamentally inadequate for managing the complexity of modern software, leading to unacceptable levels of inaccuracy in change impact analysis. The inability to reliably predict change impact directly undermines the safety assurance case required by regulatory bodies, exposing organizations to massive technical and legal risks.

However, the results also highlight certain nuances that practitioners must consider. While the tool provided massive benefits, it is entirely dependent on the existence of a high-quality, pre-established traceability matrix. The experiment simulated a scenario where the trace links were already established and accurate. In industrial reality, the creation and maintenance of these links require significant upfront investment and cultural discipline within the engineering team. Furthermore, some participants in the experimental group

exhibited signs of automation bias, occasionally trusting the tools suggestions without applying sufficient critical engineering judgment. Therefore, the successful deployment of these tools must be accompanied by comprehensive training programs that emphasize the tool as a decision-support mechanism rather than a fully autonomous oracle.

6. Conclusion and Threats to Validity

6.1 Summary of Contributions

This research provides a rigorous empirical foundation for evaluating the effectiveness of requirements traceability tools in predicting change impact within safety-critical systems. Through a meticulously designed controlled experiment involving professional software engineers, this study quantified the substantial benefits of specialized tool support. The findings conclusively demonstrate that automated traceability tools significantly elevate both the precision and recall of change impact analysis compared to traditional manual methodologies. Furthermore, the research illustrates that these tools drastically reduce the time required to perform complex analysis tasks while simultaneously lowering the cognitive burden on the engineering staff. These contributions offer actionable insights for software project managers and quality assurance professionals seeking to optimize their change management processes in highly regulated domains.

6.2 Internal, External, and Construct Validity

As with any empirical software engineering study, it is essential to acknowledge and evaluate the threats to validity [16]. Regarding internal validity, the primary threat was the potential difference in baseline skill levels between the control and experimental groups. This was mitigated through strict selection criteria and randomized assignment, validated by demographic analysis. Another internal threat was learning effects during the experiment; to counter this, the order of the change requests was randomized for each participant. Construct validity concerns whether the experiment accurately measured what it intended to measure. We utilized industry-standard metrics of precision, recall, and time, and based the tasks on a highly realistic train control system to ensure the construct accurately reflected real-world engineering challenges.

The primary threat to external validity lies in the generalization of the results. While the train control management system is representative of safety-critical applications, the scale of the experimental subsystem was necessarily constrained to fit within a two-hour laboratory session. Real-world systems possess millions of lines of code and tens of thousands of requirements, which could introduce scaling challenges not observed in this study. Additionally, the experiment assumed a perfect initial state of trace links, whereas industrial repositories often suffer from outdated or missing links, which would inevitably degrade the performance of the traceability tool.

6.3 Future Research Directions

The findings of this controlled experiment open several promising avenues for future research. Subsequent studies should investigate the performance of requirements traceability tools in environments where the underlying trace links are incomplete or partially degraded, simulating a more realistic industrial starting point. Research is also needed to evaluate the long-term adoption and maintenance challenges of these tools across multi-year software lifecycles. Finally, as artificial intelligence continues to advance, future experiments should explore the efficacy of predictive impact models that dynamically learn from historical change data to automatically suggest missing trace links and predict unforeseen architectural impacts without relying entirely on explicitly defined static traceability matrices.

References

1. Montero Guerra, J. M., Danvila-del-Valle, I., & Méndez-Suárez, M. (2023). The impact of digital transformation on talent management. Technological Forecasting and Social Change, 188, 122291.

2. Al-Banna, A.; Yaqot, M.; Menezes, B.C. Investment strategies in Industry 4.0 for enhanced supply chain resilience: An empirical analysis. Cogent Bus. Manag. 2024, 11, 2298187.

3. Okoli, C., & Schabram, K. (2010). A guide to conducting a systematic literature review of information systems. Sprouts.

4. Heavin, C., & Power, D. J. (2018). Challenges for digital transformation—Towards a conceptual decision support guide for managers. Journal of Decision System, 27(Suppl. S1), 38–45.

5. Lee, W.; Sunwoo, D.; Gerstlauer, A.; John, L.K. Cloud-Guided QoS and Energy Management for Mobile Interactive Web Applications. In Proceedings of the 2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and

Systems (MOBILESoft), Buenos Aires, Argentina, 22–23 May 2017; pp. 25–29.

6. Di Nardo, V.; Fino, R.; Fiore, M.; Mignogna, G.; Mongiello, M.; Simeone, G. Usage of Gamification Techniques in Software Engineering Education and Training: A Systematic Review. Computers 2024, 13, 196.

7. Kuhlmann, S., & Heuberger, M. (2023). Digital transformation going local: Implementation, impacts and constraints from a German perspective. Public Money & Management, 43(2), 147–155.

8. Fuchs, J.; Schneider, R.; Oks, S.; Franke, J. Service-based integration of modular control components in digital manufacturing platforms. In Proceedings of the IEEE 19th International Conference on Industrial Informatics (INDIN), Palma de Mallorca, Spain, 21–23 July 2021.

9. Ferreira, J.; Santos, B.; Oliveira, W.; Antunes, N.; Cabral, B.; Fernandes, J.P. On Security and Energy Efficiency in Android Smartphones. In Proceedings of the 2023 IEEE/ACM 10th International Conference on Mobile Software Engineering and Systems (MOBILESoft), Melbourne, Australia, 14–15 May 2023; pp. 87–95.

10. Kessel, L., & Graf-Vlachy, L. (2022). Chief digital officers: The state of the art and the road ahead. Management Review Quarterly, 72(4), 1249–1286.

11. Moi, L., & Cabiddu, F. (2021). Leading digital transformation through an Agile marketing capability: The case of Spotahome. Journal of Management & Governance, 25(4), 1145–1177.

12. Enyejo, J.O.; Fajana, O.P.; Jok, I.S.; Ihejirika, C.J.; Awotiwon, B.O.; Olola, T.M. Digital twin technology, predictive analytics, and sustainable project management in global supply chains for risk mitigation, optimization, and carbon footprint reduction through green initiatives. Int. J. Innov. Sci. Res. Technol. 2024, 9, 1344.

13. Jin, M.; Chen, Y. Has green innovation been improved by intelligent manufacturing?—Evidence from listed Chinese manufacturing enterprises. Technol. Forecast. Soc. Change 2024, 205, 123487.

14. Collart, A.J.; Canales, E. How might broad adoption of blockchain-based traceability impact the US fresh produce supply chain? Appl. Econ. Perspect. Policy 2022, 44, 219–236.

15. Tsekouropoulos, G., Vasileiou, A., Hoxha, G., Theocharis, D., Theodoridou, E., & Grigoriadis, T. (2025). Leadership 4.0: Navigating the challenges of the digital transformation in healthcare and beyond. Administrative Sciences, 15(6), 194.

16. Shadish, W.R.; Cook, T.D.; Campbell, D.T. Experimental and Quasi-Experimental Designs for Generalized Causal Inference; Houghton, Mifflin and Company: Boston, MA, USA, 2002.