A Systematic Mapping Study on Impact Analysis

Emanuele Gentili[1,2][a], Jonida Çarka[1][b] and Davide Falessi[1][c]
[1] *University of Rome Tor Vergata, Rome, Italy*
[2] *MBDA Italy S.p.a., Italy*

Keywords: Impact Analysis, Mapping Study, Software Changes.

Abstract: **Context:** Software change impact analysis (CIA) concerns managing the consequences of the changes in software artefacts. CIA solutions support software developers in making informed decisions regarding the impact of changes and ensure the overall stability and reliability of software systems. **Aim:** This paper presents a mapping study on CIA solutions. Our analysis focuses on two main dimensions: 1) the source of the change and 2) the target of the change. **Method:** We analyse 258 primary studies. **Results:** We show that in more than one-third of the cases, the Target and Source artefacts mentioned are in the general category. The second most analyzed artefact is Code. In contrast, the least mentioned source artefact is test, while the least mentioned target artefact is requirement. **Conclusions:** The identified research gaps offer opportunities to expand the knowledge and understanding of CIA techniques, ultimately benefiting practitioners and software development processes as a whole.

1 INTRODUCTION

Software development follows an iterative approach involving frequent changes to software artefacts (Turk et al., 2002). Managing and understanding the consequences of these changes is critical for ensuring the quality and reliability of software systems (Boehm and Sullivan, 2000). This process is known as software change impact analysis (CIA), which controls, understands, and manages the consequences of changes to software artefacts (Bohner, 1996).

Software undergoes changes for various reasons, such as addressing bugs, introducing new features, or enhancing performance. There are four main types of software maintenance: perfective, preventive, corrective, and adaptive. Perfective maintenance focuses on enhancing software functionality, while preventive maintenance aims to identify and address potential issues proactively. Corrective maintenance deals with bug fixing, while adaptive maintenance involves adapting software to operating environment changes or requirements (Lientz and Swanson, 1980).

Regardless of the reasons for software changes, it is essential to understand how we can limit and control those changes. CIA supports software developers in making informed decisions regarding the impact of changes and ensures the overall stability and reliability of software systems.

This paper presents a mapping study on solutions for CIA in software development. While several mapping studies have been conducted on the CIA (Kretsou et al., 2021; Li et al., 2013; Malhotra and Bansal, 2016; Malhotra and Khanna, 2019), our analysis focuses on two new dimensions: 1) the source of the change and 2) the target of the change. Thus, the contribution of this paper lies in the comprehensive characterisation of CIA solutions and their relationship to different types of artefacts, such as requirements, design, code, and tests.

Our study analyses 258 primary studies and provides a detailed characterisation of the solutions proposed for CIA. Our results show that few studies support changes when the source is a late artefact (e.g. test) and the target an early artefact (e.g. requirement).

The remainder of the paper is organised as follows. In Section 2 we present the related work on CIA. Sections 3 and 4 describe the methodology used for this mapping study, addressing the research questions and presenting a comprehensive analysis of the results. Section 5 discusses the implications of our results on researchers and practitioners. Finally, Sections 6 to 7 discuss the threats to validity and provide concluding remarks based on the study.

[a] https://orcid.org/0009–0002-4283-9114
[b] https://orcid.org/0000-0001-9315-3652
[c] https://orcid.org/0000-0002-6340-0058

Gentili, E., Çarka, J. and Falessi, D.
A Systematic Mapping Study on Impact Analysis.
DOI: 10.5220/0012758200003753
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 19th International Conference on Software Technologies (ICSOFT 2024)*, pages 375-382
ISBN: 978-989-758-706-1; ISSN: 2184-2833
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

2 RELATED WORK

This section presents the existing body of research on CIA in software engineering. This section comprises two subsections: Introduction to CIA, where we provide an overview of the studies conducted to investigate various aspects of impact analysis, underscoring its significance and relevance in software engineering. Additionally, we include Literature Reviews on CIA, which focuses on reviewing and discussing secondary studies that directly or indirectly pertain to our research. By examining the literature in these subsections, we aim to comprehensively understand the current research landscape in the realm of CIA.

2.1 Introduction to CIA

Several studies have been conducted to address various aspects of CIA in the field of software engineering:

Lehnert (2011) addresses the need for a comprehensive framework to classify and compare approaches for CIA in software development. The paper introduces a new taxonomy for CIA derived from a literature review of 160 relevant studies. The taxonomy provides detailed and precise classification criteria, overcoming the limitations of existing frameworks. The applicability and usefulness of the taxonomy are demonstrated through the classification of multiple approaches. This research serves as a foundation for conducting a comprehensive software change impact analysis survey. The taxonomy addresses various levels of software, such as source code, architectural models, and miscellaneous artifacts, highlighting the need for a comprehensive classification system. Future work involves expanding the taxonomy to a more extensive set of studies and enhancing its criteria and granularity levels. For example, the study conducted by Lehnert (2011) does not explicitly mention software testing as an artifact in the taxonomy for CIA.

Bordin and Benitti (2018) focus on the need to prioritize research and practices in software maintenance in academia and industry. Their study analyzes how software maintenance is addressed in academic programs and industry practices, comparing them to identify alignment and potential areas of improvement. The study utilizes document review, survey, and comparative analysis methodologies. The findings reveal that while legacy systems and impact analysis are prevalent in industry practices, they are not extensively explored in academia. Conversely, topics related to the maintenance process and reengineering should be more utilized in industry practices.

Aung et al. (2020) conducted a systematic literature review to examine CIA in large-scale software development projects and the use of automated traceability-link recovery approaches. The review identifies 33 relevant studies and investigates various aspects of the CIA, including traceability approaches, CIA sets, evaluation degrees, trace direction, and methods for recovering traceability links between different artifact types. The study highlights the limited number of studies addressing the design and testing of impact analysis sets, likely due to the need for more available datasets. It suggests the need for further industrial case studies and the development of traceability tools to support fully automatic approaches utilizing machine learning and deep learning.

Anwer et al. (2019) focus on Requirement Change Management (RCM) in both in-house software development and Global Software Development (GSD) approaches, with a specific emphasis on impact analysis. Their study identifies challenges influencing RCM, particularly concerning CIA, through a systematic literature review of 43 papers and a questionnaire survey. While the study provides valuable insights for researchers and industry professionals to enhance their understanding and implementation of CIA in different development scenarios, it does not analyze other artifacts such as test design or source code.

Rinkevics and Torkar (2013) present a systematic literature review of Cumulative Voting (CV) and CV analysis methods in software engineering. Their study focuses on the practical use of CV and the detection of prioritization items with equal priority. The review highlights CV applications in various software engineering disciplines, including requirements prioritization and CIA. They introduce Equality of Cumulative Votes (ECVs) as a method to identify equal priority items in CV results. The evaluation of ECV on prioritization cases demonstrates its effectiveness in detecting equal items, providing insights for practitioners adopting CV, and supporting decision-making processes in software engineering.

These studies contribute to understanding CIA, software maintenance, and prioritization in software engineering. They offer valuable insights and highlight the need for further research and improvements in various areas, such as taxonomy development, traceability tools, best practices in requirement change management, and other artifacts such as software design, source code, and testing.

2.2 Literature Reviews on CIA

This subsection reviews and discusses related work, specifically secondary studies such as Systematic Literature Reviews (SLRs) or Systematic Mapping Studies (SMS), that are directly or indirectly related to our research. Table 1 reports some highlights of the four related secondary studies that we elaborate below.

Kretsou et al. (2021) conducted a systematic mapping study focusing on the usefulness of the CIA, the top researched subfields, and code-based approaches. The study highlighted the practical advantages of CIA, which encompasses all maintenance requests and can help reduce associated costs. It also identified four parameters for CIA quantification and found variations in the level of research across various aspects of the CIA.

Li et al. (2013) conducted a survey that explored code-based CIA techniques. Their study emphasized the key application areas of the CIA and presented a framework for comparing code-based techniques. It highlighted the importance of CIA in critical activities such as software comprehension, change propagation, regression test case selection, debugging, and attribute measurement.

Malhotra and Bansal (2016) conducted a literature review in 2016 specifically on software change prediction, focusing on change proneness. Their review highlighted the strong predictive power of object-oriented measures in change prediction.

Malhotra and Khanna (2019) systematically reviewed software change prediction. Their study examined experimental settings, characteristics and performances of prediction models, and the use of CK metrics (Chidamber and Kemerer, 1994), feature selection strategies, and evaluation metrics such as AUC. The review found that CK metrics were extensively used, feature selection strategies were commonly employed, and AUC was frequently used as a performance metric.

These mapping studies provide valuable insights into various aspects of CIA, including its usefulness, application areas, techniques, metrics, and prediction models.

3 METHODOLOGY

This section presents the approach used in this study. We begin by stating our objectives and research questions, formulated using the Goal-Question-Metrics (GQM) framework (Basili et al., 1994). These research questions focus on characterizing CIA solutions in terms of their input and output. Next, we describe the study selection process, where we conducted a comprehensive search in major digital libraries and applied filtering criteria based on domain, venue, and article type. The selected studies were then subjected to data collection, which involved classifying each paper based on two dimensions: the source of the change and the target of the change.

3.1 Objectives and Research Questions

The goal of this study, specified using the GQM approach, is to analyse existing CIA solutions for the purpose of characterisation with respect to the input and output from the point of view of researchers and practitioners in the context of software engineering.

Based on the GQM formulation and the aim detailed in Section 1, we investigate the following research questions:

- *RQ1: What is the source of the CIA?* RQ1 aims at characterising the primary studies in terms of the specific types of artefacts, e.g. requirements, that are considered by the solution provided in the study as the source of the CIA.
- *RQ2: What is the target of the CIA?* RQ2 aims at characterising the primary studies in terms of the specific types of artefacts, e.g. code, that are considered by the solution provided in the study as the target of the CIA.

3.2 Study Selection

Our goal is to identify any papers related to the topic of CIA, which we define as a technique used to limit or evaluate the possible outcomes of a change in a software system (Kretsou et al., 2021).

We conducted a search for academic literature in software engineering by using a specific search string and limited our search to three digital libraries: ACM Digital Library, IEEE Xplore, and SCOPUS. We chose these libraries because they are among the most frequently used digital libraries for software engineering (Croft et al., 2023). We decided to avoid other search engines, such as Google Scholar, to avoid including non-peer-reviewed articles. As a search query, to be comprehensive, we use the string “impact analysis” in the title, abstract or keywords.

Our initial search across three major digital libraries - SCOPUS, IEEE Xplore, and ACM Digital Library - yielded a total of 16,554 studies, including 14,573 from SCOPUS, 1,624 from IEEE Xplore, and 357 from ACM Digital Library. We applied various filtering criteria based on domain, venue, and article type to refine our results. We then downloaded all retrieved studies and removed duplicates using the bibtex-tidy tool[1]. Furthermore, in our categorization process, we classified the primary studies according

[1]https://github.com/FlamingTempura/bibtex-tidy.git

Table 1: Mapping studies in CIA.

	Kretsou et al. (2021)	**Li et al. (2013)**	**Malhotra and Khanna (2019)**	**Malhotra and Bansal (2016)**	**Our Work**
Title	Change impact analysis:A systematic mapping study	A survey of code-based change impact analysis techniques	Software Change Prediction: A Systematic Review and Future Guidelines	Software change prediction: a literature review	A Systematic Mapping Study of Impact Analysis
Year	2021	2012	2019	2016	2023
#Primary Studies	111	30	38	21	258
Main Focus	*Usefulness of CIA; *Top researched CIA subfields; *Only code;	*Key application areas of CIA; *Framework for comparing code-based CIA techniques;	*Change prediction; *Experimental settings; *Prediction models characteristics and performances;	*Change proneness;	*Artifacts supported as input and output of solutions;
Results	*practical advantages of CIA; * CIA quantification parameters; * some CIA aspects are more researched than others;	*CIA supports many critical activities such as software comprehension, change propagation, selection of regression, test cases, debugging and measuring various software attributes;	*The majority of studies have employed feature selection/dimensionality reduction strategies; *The majority of the datasets are open-source; *AUC metric is frequently used;	*Object-oriented measures have a strong predictive power;	*Under-studied and over-studied areas in CIA techniques, urging researchers to focus on late to early artifact solutions;

to the type of support they provided. Under the category of Identification, the papers proposed solutions to identify the impacted elements using terms like ”impact set” or ”identification.” On the other hand, under the category of Prevention, the papers presented solutions to mitigate or lessen the impact of system changes, employing terms such as ”maintainability” or ”stability.” As part of our exclusion criteria, we removed all studies that focused on solutions to limit or reduce the impact of changes to the system.

Figure 1 shows the workflow that depicts the entire search and study selection process, highlighting the number of papers retrieved at each stage. We conducted the process in April 2023 and obtained a total of 258 primary studies, and then we listed all of them in Appendix A.

Table 2 reports our 3 inclusion and 8 exclusion criteria as inspired by (Croft et al., 2023).

The first two authors applied the exclusion criteria. In the event of disagreements, we reached a consensus via a discussion with the third author; we had only three disagreements about exclusion criteria.

3.3 Data Collection

We classify each paper in the following two orthogonal dimensions:

1. *Source of the change*: The input of the solution is the element or component that undergoes the change and affects the system. For instance, suppose that a model is developed from requirements and it supports the understanding of how changes in requirements impact the design; then the source of the change is requirements. Based on the provided information, this dimension can have the following values:
   - Requirement: Specifications or requirements that define what the system should do. This category also includes standards that the system needs to adhere to (Pohl, 2010).
   - Design: UML diagrams or similar artifacts that represent the structure, architecture, or design of the system (Rumbaugh et al., 1996).
   - Code: The actual implementation of the system, including the source code. This category also includes the identification of bugs or defects in the code (Eckel, 2005).
   - Test: Artefacts and activities related to testing, such as test cases, test plans, or test scripts (Ammann and Offutt, 2008).
   - General: It’s important to note that these categories are not exhaustive and may vary depending on the specific context of the paper or study.

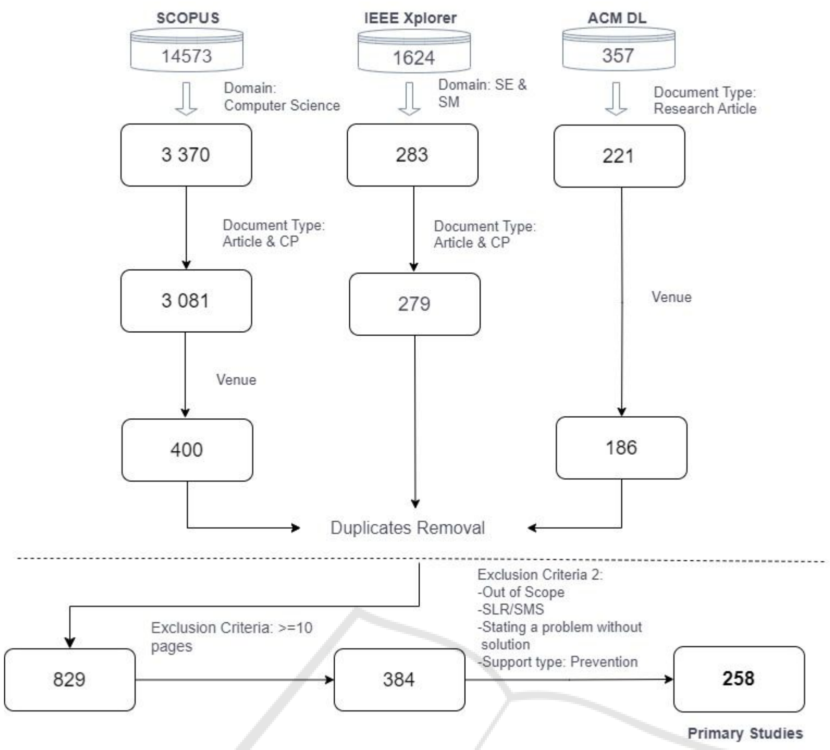

Figure 1: Study selection process.

Moreover, the primary study might be vague about the category of artefacts involved in the CIA. Therefore, we classify an artefact as general if the above-mentioned categories are inappropriate.

2. *Target of the change*: What is impacted by the change. For instance, suppose that a model is developed from requirements and it supports the understanding of how changes in requirements impact the design; then the target of the change is design. This dimension has the same values as the source of the solution.

The first two authors applied the exclusion criteria. In the event of disagreements, we reached a consensus via a discussion with the three authors; we had only three disagreements about exclusion criteria. We had 24 disagreements for the first dimension and 21 disagreements for the second dimension.

4 STUDY RESULTS

Figure 2 reports the number of studies published across periods of over 20 years. According to Figure 2, there has been a significant increase in the number of studies over time, even if the trend is not perfectly monotone, with a spike in 2018.

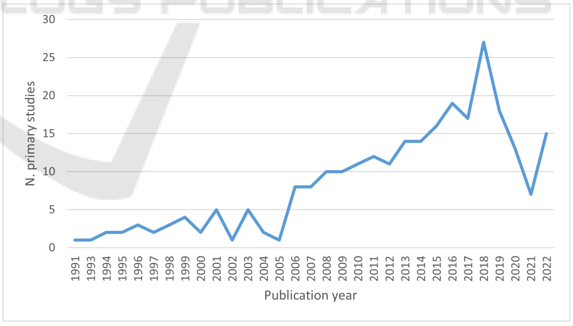

Figure 2: Distribution of primary studies across years.

Figure 3 shows the proportion of primary studies across target artefacts and source artefacts. According to Figure 2, in more than one-third of the cases, the Target and Source artefacts mentioned are in the general category. The second most analyzed artefact is Code. In contrast, the least mentioned source artefact is test, while the least mentioned target artefact is requirement.

Figure 4 reports the frequency of CIA solutions for specific source and target combinations.

Table 2: Inclusion/Exclusion Criteria.

Inclusion Criteria
I1. The study relates to the field of Computer Science, and informs the practice of Software Engineering and Maintenance
I2. The study proposes a solution and/or its evaluation to support CIA.
I3. The study is a full paper longer than nine pages.
Exclusion Criteria
E1. Solely a literature review or survey article.
E2. Non peer-reviewed academic literature.
E3. Academic articles other than conference or journal papers, such as book chapters or dissertations.
E4. Studies not written in English.
E5. Studies whose full-text is unavailable.
E6. Studies published to a venue unrelated to the discipline of Computer Science.
E7. Studies that are less than ten pages long.
E8. Studies that state a problem without any solution
E9. Studies propose a solution to limit or reduce the impact of changes to the system.

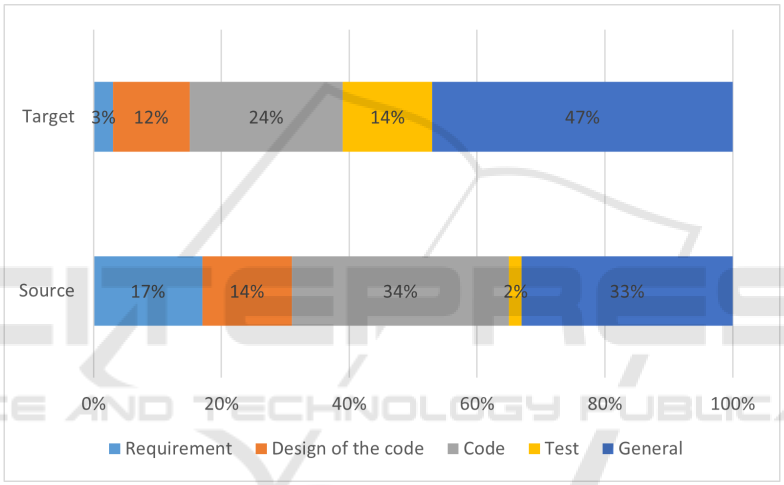

Figure 3: Proportion of primary studies across target and source artefacts.

According to Figure4 two combinations of source-target, i.e., General-General and Code-Code, cover 41% of the primary studies. No primary studies provide support for four combinations (Design-Requirement, Test-Requirement, Test-Design, and General-Requirement).

5 DISCUSSION

In this section, we discuss the implications of our study for researchers. As in (Kretsou et al., 2021) we organize the discussion section according to over-studied areas and under-studied areas. Regarding the under-studied areas, we address the missing source and target combinations, such as the Requirement to Test or Requirement to Design and provide possible explanations for this observation. This could be attributed to several factors. One possible explanation is that the research community has predominantly focused on studying specific combinations considered more critical or commonly encountered in practice such as Code to Code. As a result, other combinations may have received less attention, leading to limited empirical evidence and understanding in those areas. Another reason is that the software development process follows a specific direction, from Requirement to Design to Code to Test. Therefore, a change in Test might be perceived as unlikely to impact requirements. However, since the software development lifecycle is mostly iterative, changes in Test might impact Requirements or other artefacts. Identifying and examining the reasons behind these missing combinations can guide future research efforts to bridge these gaps and provide a more comprehensive understanding of CIA.

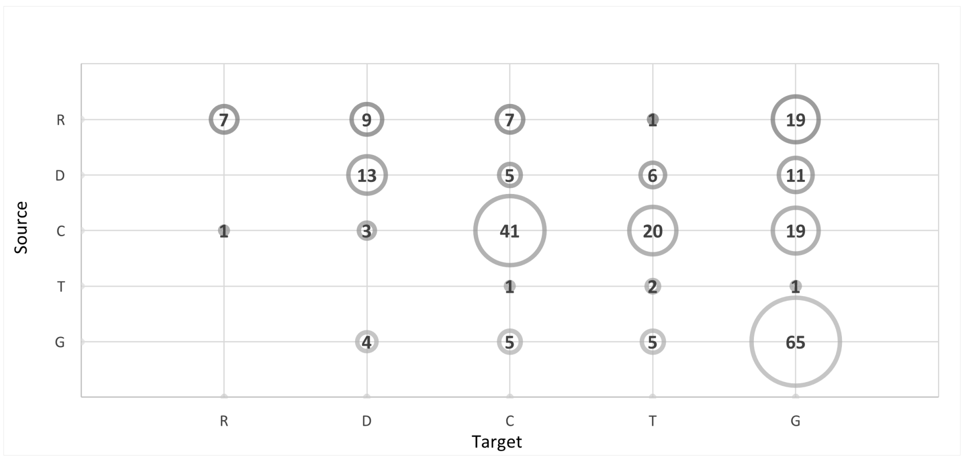

Figure 4: Frequency of CIA solutions for specific source and target combinations.

In conclusion, our study offers valuable insights for researchers in the field of CIA. By addressing the missing source and target combination, we provide a basis for future research endeavours. The identified research gaps and areas of further investigation present opportunities to enhance the knowledge and understanding of CIA solutions, ultimately benefiting practitioners and software development processes as a whole.

6 THREATS TO VALIDITY

Threats to validity can impact the reliability and externalizability of the study's findings. To address these threats, we took specific measures, as described below.

One possible threat to validity is the possibility of missing relevant papers. To mitigate this threat, we searched three main databases (ACM Digital Library, IEEE Xplore, and SCOPUS) to ensure a broader scope of coverage (Croft et al., 2023). Additionally, we utilised multiple search criteria, including title, keywords, and abstract, and employed two specific search terms (”impact analysis”) to capture relevant literature related to CIA.

Another potential threat is the inclusion of papers that are of low quality. To address this concern, we applied a rigorous selection process during the analysis phase, specifically considering only peer-reviewed venues of Conference and Journal papers and papers with more than ten pages (See Inclusion Criteria I3 and Exclusion Criteria E6 in Table 2). This approach helped ensure that the included papers met a certain quality standard.

There is also a threat related to the misclassification of papers during the analysis phase. To minimise this threat, we employed a meticulous procedure. All the researchers performed each analysis individually, and the results were subsequently merged. Inconsistencies were carefully identified and resolved, ensuring the accuracy of the classification process.

Furthermore, there is a possibility of wrong understanding or misinterpretation of the artefact types. We adopted standard definitions of artefacts described in section 3 to mitigate this threat. By adhering to established definitions, we aimed to maintain consistency and accuracy in identifying and categorising artefacts throughout the study.

By addressing these potential threats to validity and implementing appropriate measures, we aimed to enhance the reliability and validity of our study's findings.

7 CONCLUSIONS

This paper provides a comprehensive analysis of solutions for CIA (Change Impact Analysis) in software development. The study specifically examines two key dimensions: the origin of the change and the destination of the change offered by each solution. The main contribution of this paper lies in thoroughly characterizing CIA solutions and their connections to various artefacts, including requirements, design, code, and tests. By analyzing 258 primary studies, we offer a detailed overview of the proposed solutions for CIA.

Our results show the presence of under-studied and over-studied areas. The identified research gaps offer opportunities to expand the knowledge and understanding of CIA techniques, ultimately benefiting practitioners and software development processes as a whole. We note that there are few studies supporting changes where the source is a late artefact (e.g. test) and the target an early artefact (e.g. requirement). Since software development is iterative, researchers are encouraged to focus future efforts on providing CIA solutions for late to early artefacts.

REFERENCES

Ammann, P. and Offutt, J. (2008). *Introduction to Software Testing*. Cambridge University Press.

Anwer, S., Wen, L., Wang, Z., and Mahmood, S. (2019). Comparative analysis of requirement change management challenges between in-house and global software development: Findings of literature and industry survey. *IEEE Access*, 7:116585–116611.

Aung, T. W. W., Huo, H., and Sui, Y. (2020). A literature review of automatic traceability links recovery for software change impact analysis. In *ICPC '20: 28th International Conference on Program Comprehension, Seoul, Republic of Korea, July 13-15, 2020*, pages 14–24. ACM.

Basili, V. R., Caldiera, G., and Rombach, D. H. (1994). *The Goal Question Metric Approach*, volume I. John Wiley & Sons.

Boehm, B. W. and Sullivan, K. J. (2000). Software economics: a roadmap. In Finkelstein, A., editor, *22nd International Conference on on Software Engineering, Future of Software Engineering Track, ICSE 2000, Limerick Ireland, June 4-11, 2000*, pages 319–343. ACM.

Bohner, S. A. (1996). Impact analysis in the software change process: a year 2000 perspective. In *1996 International Conference on Software Maintenance (ICSM '96), 4-8 November 1996, Monterey, CA, USA, Proceedings*, pages 42–51. IEEE Computer Society.

Bordin, A. S. and Benitti, F. B. V. (2018). Software maintenance: what do we teach and what does the industry practice? In Kulesza, U., editor, *Proceedings of the XXXII Brazilian Symposium on Software Engineering, SBES 2018, Sao Carlos, Brazil, September 17-21, 2018*, pages 270–279. ACM.

Chidamber, S. R. and Kemerer, C. F. (1994). A metrics suite for object oriented design. *IEEE Transactions on software engineering*, 20(6):476–493.

Croft, R., Xie, Y., and Babar, M. A. (2023). Data preparation for software vulnerability prediction: A systematic literature review. *IEEE Trans. Software Eng.*, 49(3):1044–1063.

Eckel, B. (2005). *Thinking in Java (4th Edition)*. Prentice Hall PTR, USA.

Kretsou, M., Arvanitou, E., Ampatzoglou, A., Deligiannis, I. S., and Gerogiannis, V. C. (2021). Change impact analysis: A systematic mapping study. *J. Syst. Softw.*, 174:110892.

Lehnert, S. (2011). A taxonomy for software change impact analysis. In Cleve, A. and Robbes, R., editors, *Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th annual ERCIM Workshop on Software Evolution, EVOL/IWPSE 2011, Szeged, Hungary, September 5-6, 2011*, pages 41–50. ACM.

Li, B., Sun, X., Leung, H., and Zhang, S. (2013). A survey of code-based change impact analysis techniques. *Softw. Test. Verification Reliab.*, 23(8):613–646.

Lientz, B. and Swanson, E. (1980). Software maintenance management. *Iee Proceedings E Computers and Digital Techniques*, 127.

Malhotra, R. and Bansal, A. J. (2016). Software change prediction: a literature review. *Int. J. Comput. Appl. Technol.*, 54(4):240–256.

Malhotra, R. and Khanna, M. (2019). Software change prediction: A systematic review and future guidelines. *e Informatica Softw. Eng. J.*, 13(1):227–259.

Pohl, K. (2010). *Requirements Engineering - Fundamentals, Principles, and Techniques*. Springer.

Rinkevics, K. and Torkar, R. (2013). Equality in cumulative voting: A systematic review with an improvement proposal. *Inf. Softw. Technol.*, 55(2):267–287.

Rumbaugh, J., Jacobson, I., and Booch, G. (1996). The unified modeling language. *University Video Communications*.

Turk, D., France, R., and Rumpe, B. (2002). Limitations of agile software processes. In *Third international conference on eXtreme programming and agile processes in software engineering (XP 2002)*, pages 43–46. Citeseer.

APPENDIX

All the primary studies used in this paper are accessible via the following link: https://doi.org/10.5281/zenodo.8235889.