Journal of Infrastructure, Policy and Development 2024, 8(8), 4457.
https://doi.org/10.24294/jipd.v8i8.4457

Article

Sustainable testing strategies for large-scale software applications

R. Kanesaraj Ramasamy*, Teddy Teh, Leong Chan Fook, Murali Dharan

Faculty of Computing and Informatics, Multimedia University, Cyberjaya 63100, Malaysia
*** Corresponding author:** R. Kanesaraj Ramasamy, r.kanesaraj@mmu.edu.my

CITATION

Ramasamy RK, Teh T, Fook LC, Dharan M. (2024). Sustainable testing strategies for large-scale software applications. Journal of Infrastructure, Policy and Development. 8(8): 4457. https://doi.org/10.24294/jipd.v8i8.4457

ARTICLE INFO

Received: 29 January 2024
Accepted: 22 April 2024
Available online: 23 August 2024

Abstract: As the complexity and scale of software applications increase, the challenges associated with testing these systems grow correspondingly, necessitating innovative and sustainable testing strategies. This paper explores a multifaceted approach aimed at addressing the intricate challenges inherent in testing large-scale software applications. Through a comprehensive examination of current industry practices and emerging trends, this study introduces a novel framework that integrates advanced testing techniques with state-of-the-art tools. This framework not only mitigates the challenges posed by the complexity and size of modern applications but also enhances the efficiency and effectiveness of the testing process. Key aspects of this research include a detailed exploration of test methodologies suited for large-scale applications, an evaluation of advanced tools designed for complex test scenarios, and an analysis of the impact of the test environment on sustainability. The findings offer valuable insights and actionable strategies for software development and testing professionals aiming to optimize testing processes and improve the quality and sustainability of their software in a rapidly evolving technological landscape.

Keywords: large-scale software testing; test automation; sustainable testing strategies; advanced testing techniques; software testing frameworks

1. Introduction

In the dynamic environment of software development, the proliferation of large-scale applications has ushered in a new era of innovation and complexity. As these applications grow and become complex, the testing process encounters a myriad of challenges that require a strategic and sustainable approach. This research seeks to explore and address the unique testing challenges associated with large-scale software applications, highlighting the need for comprehensive solutions in the areas of testing techniques, tools, methodologies, challenges, and environments, with a particular focus on test automation.

The advent of large-scale applications brings complexities beyond the traditional boundaries of testing methodologies. Ensuring the reliability, performance and security of such large-scale systems requires a thorough understanding of complex situations. This paper aims to contribute to the existing body of knowledge by examining the specific problems that arise in testing large-scale applications and proposing an integrated strategy for sustainable test practices.

We have undertaken an exhaustive exploration, delving into published papers over the past five years that dissect the multifaceted issues surrounding software testing. This extensive review has provided us with valuable insights into the nuances of testing challenges, offering a comprehensive understanding of how to sustain software testing strategies and make them inherently sustainable.

The research is motivated by the recognition that traditional testing approaches

can fall short in effectively addressing the scale and complexity of modern software applications. As these applications become an integral part of various industries, from finance to healthcare, the consequences of insufficient testing can be profound. Our study therefore aims to bridge the gap between the escalating demands of large-scale application development and the need for robust testing methodologies.

Key areas of focus include exploring advanced testing techniques tailored to large-scale applications, evaluating state-of-the-art tools capable of handling complex test scenarios, and in-depth analysis of the test environment's impact on sustainability. In addition, the research examines the role of test automation as a fundamental pillar for overcoming problems and increasing the efficiency of the testing process.

By addressing these critical aspects, this research aims to provide a comprehensive framework that not only meets the challenges posed by large-scale applications, but also contributes to the overall advancement of testing practices. The following sections dive into specific aspects of testing large-scale software applications and present a blueprint for achieving sustainability and efficiency in the testing process. Through this survey, we aim to offer actionable insights and recommendations for software development and testing professionals navigating the evolving landscape of large-scale application development.

2. Materials and methods

When searching IEEE Xplore for research on “sustainable testing strategies for large-scale software applications,” we adopted a focused and comprehensive keyword strategy. Our search queries (“experiment” and “large-scale software”) were specifically chosen to represent a broad database of publications related to experimentation in the field of large-scale software.

To ensure that the research is up-to-date and in line with the latest developments in the field, we applied a filter to include only publications from the past few years. This step was necessary to maintain the relevance and timeliness of the study.

The search yielded approximately 131 results. This number indicates that there is a significant amount of relevant research, but it is possible to review it thoroughly. We then began the full process of scanning and reviewing titles and summarizing these results. This careful review was important to identify articles that relate to the broad topic of sustainable testing strategies and provide specific insights, methodologies, or case studies relevant to our research focus.

This systematic approach to IEEE Xplore has proven to be efficient and effective, allowing us to assemble a collection of relevant and useful reference materials that provide a solid foundation for our research project.

The following are the research questions addressed:

RQ1: What are the challenges and environment for sustainable Testing Strategies for Large-Scale Software Applications

RQ2: What are the predominant testing methods used in large-scale software applications, and how do they contribute to the sustainability of the testing process?

RQ3: How do different software development methodologies and approaches to software development influence the choice and effectiveness of testing methods in large-scale software development?

RQ4: How are emerging technologies like AI and machine learning integrated into existing testing methods to enhance efficiency and sustainability in large-scale software applications?

3. Literature review

The testing of large-scale software applications presents a myriad of challenges and opportunities. Addressing these challenges requires a nuanced understanding of the testing environment and the adoption of sustainable testing strategies. This inquiry delves into four key research questions that collectively shed light on the intricacies of testing in the context of large-scale software applications.

Subsection

RQ1: What are the challenges and environment for sustainable Testing Strategies for Large-Scale Software Application?

In large-scale software applications, developing sustainable testing strategies is a multifaceted task. This paper endeavours to uncover the inherent challenges within these strategies and the environmental prerequisites necessary to ensure their enduring effectiveness in software testing sustainability. Here, we delineate key challenges and environmental factors crucial for the sustainability of software testing.

As software systems grow in complexity, and their interactions and underlying technologies become less deterministic, the issue of flakiness is poised to increase, as discussed in the study by Alshahwan et al. (2023). Furthermore, the build time of code poses a potential challenge, particularly in the context of large-scale systems comprising tens to hundreds of millions of lines of code, necessitating execution times extending into minutes or even hours (Ahlgren et al., 2021). Testing such extensive software systems requires techniques for accurately assessing functional logic, often involving the construction of realistic values for complex data types, without resorting to extensive boilerplate code (Alshahwan et al., 2023).

Scalability poses a formidable challenge when attempting to align testing efforts with the size and expansion of the application. Conventional testing methods, such as manual testing, may struggle to scale effectively in several scenarios: Firstly, in large codebases, manually inspecting every line of code becomes impractical, resulting in insufficient coverage. Secondly, frequent releases, characteristic of continuous deployment practices, necessitate faster testing cycles, thus emphasizing the need for automated solutions (Zhi et al., 2019).

Resource constraints pose significant challenges in the realm of testing, as they necessitate allocation of both time and resources, thereby impacting development cycles and budgets. Testing delays have the potential to disrupt release schedules and impede project timelines. Moreover, manual testing incurs considerable expenses, while automation tools entail upfront investments. Furthermore, as the size of the test suite expands, executing the entire suite for large-scale systems becomes prohibitively costly (Dirim and Sozer, 2020).

Maintaining software quality necessitates the concurrent evolution of test cases alongside alterations in production code. Unfortunately, this co-evolution of production and test code frequently doesn't occur during software evolution, primarily

due to time constraints for test maintenance or insufficient knowledge to determine the necessity for test updates (Hurdugaci and Zaidman, 2012; Hu et al., 2023.)

Effective generation and management of extensive test data sets are paramount in testing endeavors. Large-scale applications necessitate the creation and organization of substantial volumes of test data to cover diverse scenarios. However, this process can be resource-intensive and time-consuming. Additionally, maintaining consistency and integrity of test data across diverse testing environments is imperative to ensure the reliability of test results (Liu et al., 2021).

The analysis and interpretation of substantial volumes of test data to discern trends and facilitate informed decision-making can be intricate and time-intensive. While leveraging data visualization tools and techniques can augment reporting and analysis capabilities, doing so effectively necessitates expertise and specialized knowledge (Uygun et al., 2020).

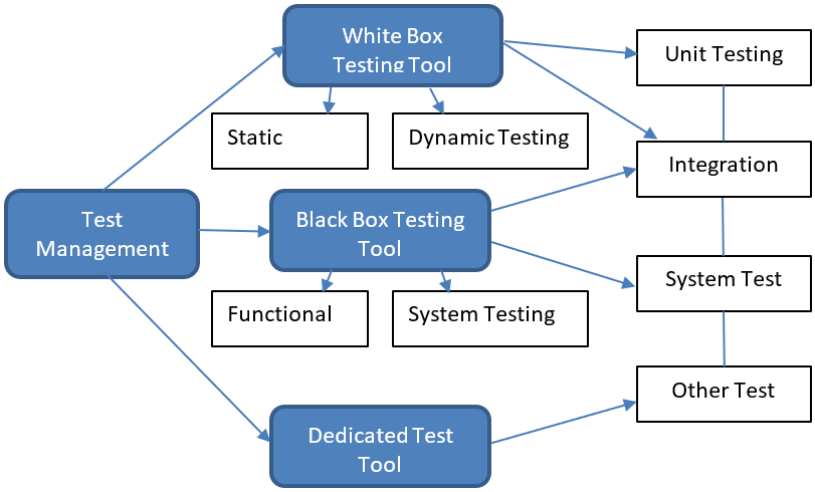

Figure 1. Classification of software testing tools.

As shown in **Figure 1**, integrating modern testing practices with existing legacy systems can be challenging because of compatibility and technical issues. Adapting methodologies or migrating to newer technologies may be necessary to overcome these hurdles (Ali et al., 2019). To ensure sustainable testing for large-scale software applications, several key components are crucial. This includes fostering a culture of testing throughout the software development lifecycle, emphasizing collaboration and continuous improvement (Agarwal et al., 2018). Automated testing tools enhance efficiency, particularly for regression testing and repetitive tasks, while test management tools streamline organization and execution (Alferidah and Ahmed, 2020; Wang and Ren, 2018) Investment in skill development and training programs for testers ensures they remain up-to-date with the latest tools and techniques (Hynninen et al., 2018). A robust performance testing infrastructure is essential to guarantee scalability and performance, alongside effective communication and collaboration practices among teams (Alshahwan et al., 2023; Barroca et al., 2015) Continuous monitoring and measurement of test effectiveness and efficiency through metrics are vital for identifying areas for improvement (Uygun et al., 2020). Finally, allocating sufficient budget and resources to testing activities is imperative for sustainability, with automation processes offering significant cost reductions for large-scale

applications (Valle-Gomez et al., 2019). This comprehensive approach to testing environment development and management ensures the success and longevity of testing strategies.

In the realm of large-scale software development, security testing is of paramount importance amidst escalating complexity and interdependencies within modern software ecosystems. Various software development methodologies adopt different approaches to integrate security testing seamlessly into the development lifecycle. In agile methodologies, security testing is intricately woven into each sprint through security-focused user stories and corresponding acceptance criteria, while traditional waterfall methodologies may relegate security testing to distinct phases towards the culmination of the development cycle. The efficacy of security testing methods varies depending on the development methodology employed; agile frameworks often favor automated security testing tools integrated with continuous integration/continuous deployment (CI/CD) pipelines, while methodologies with elongated development cycles may prefer comprehensive manual security testing.

Ultimately, irrespective of the chosen methodology, the incorporation of robust security testing practices is imperative for upholding the integrity and trustworthiness of large-scale software systems. In an era where cyber threats loom increasingly sophisticated and pervasive, the integration of comprehensive security measures serves as a bulwark against potential vulnerabilities and breaches.

RQ2: What are the predominant testing methods used in large-scale software applications, and how do they contribute to the sustainability of the testing process?

This section aims to identify and analyze various testing methods (unit testing, integration testing, system testing, etc.) used in large-scale software development. It also aims to study how these methods improve or affect the stability of the software testing process.

Table 1. Summary of the different testing methods.

Category	Description	References
Automated and AI-Enhanced Testing	Encompasses methods that leverage automation and artificial intelligence to streamline and enhance the software testing process.	(Aggarwal et al., 2018; Guo et al., 2022; Lisitsyn et al., 2021; Mezhuyev et al., 2019; Zhong et al., 2019)
Performance, Load, and Scalability Testing	Focuses on testing methods that assess and ensure the performance, load handling, and scalability of software systems.	(Lei et al., 2019; Yurtseven and Bagriyanik, 2020)
Dynamic, Regression, and Security Testing	Includes testing methods that are dynamic and adaptive, focusing on regression testing and security aspects of software.	(Jarman et al., 2019; Zhao et al., 2023)
Hybrid and Specialized Testing Approaches	Comprises testing methods that blend different approaches or are specifically designed for certain types of systems or requirements.	(Ge et al., 2023; Köroğlu et al., 2020)
System-Specific and Optimization-Based Testing	Testing methods that are either tailored for specific systems, like microservices, or use optimization techniques to enhance testing efficiency.	(Gong and Cai, 2023; Jharko, 2021)

Referring to methods in **Table 1**, Automated and AI-Enhanced Testing streamline testing processes, reducing time and effort significantly. Tasks that previously consumed hours or days can now be completed swiftly, fostering quicker feedback loops in development. This efficiency aids in early issue identification and ensures timely delivery of high-quality software. Nevertheless, complexities in setting up and maintaining automated testing frameworks, especially for complex systems,

pose challenges. Designing robust test scripts, handling dynamic UI elements, and managing test data require ongoing investment in training, tooling, and infrastructure.

Performance, Load, and Scalability Testing are vital for identifying system bottlenecks, enabling optimization. While these tests provide valuable insights, designing realistic test scenarios is challenging. Mimicking real-world usage patterns demands careful planning and domain expertise. Failure to consider these factors can limit the effectiveness of these tests in improving system performance and reliability.

Dynamic, Regression, and Security Testing offer adaptability, crucial for agile environments, but complexity arises, particularly in security testing. Simulating realistic attack scenarios necessitates specialized skills and resources, posing challenges for organizations without dedicated cybersecurity teams.

Hybrid and Specialized Testing Approaches cater to specific needs, yielding accurate results. However, complexities in implementation and maintenance, along with potential increased overheads, need careful consideration.

System-Specific and Optimization-Based Testing align testing efforts with system characteristics, enhancing accuracy. Yet, implementing these methods, especially in complex projects, poses challenges. Designing and validating optimization algorithms require specialized expertise and significant upfront investment. Despite these hurdles, the benefits in improving software quality make them valuable additions to testing strategies.

The use of AI and machine learning technologies has attracted interest in software test automation, as discussed in papers (Aggarwal et al., 2018; Guo et al., 2022; Lisitsyn et al., 2021; Mezhuyev et al., 2019; Zhong et al., 2019). As mentioned by Zhong et al. (2019), which uses automation tools useful for Android applications, automatic function testing has been given special attention. The results show a significant reduction in manual testing problems and a significant improvement in the test environment. The groundbreaking approach using deep neuroevolution discussed by Aggarwal et al. (2018) has made considerable progress in GUI-based testing, effectively combining deep learning with evolutionary algorithms to streamline and improve the testing process. Guo et al. (2022), Mezhuyev et al. (2019), and Lisitsyn et al. (2021) also demonstrated the use of Generative Adversarial Networks (GAN) to generate high-level test data. By training AI models to generate high-quality test cases, a significant reduction in the traditional time and resources devoted to manual test case development has been observed.

Moreover, the importance of production testing in ensuring the efficient operation of large-scale software systems operate efficiently under varying loads and conditions is discussed by Yurtseven and Bagriyanik (2020). This paper discusses the issue of identifying performance issues in open-source projects, stating that it is important to identify and address performance bottlenecks. Similarly, Lei et al. (2019) introduce Kubemark, a tool for performance testing in microservices architecture using Kubernetes. This tool simulates large-scale Kubernetes clusters, allowing you to evaluate performance and scalability given the complexity of such architectures.

Futhermore, dynamic and regression testing, as discussed by Jarman et al. (2019), is essential for managing multiple test cases in large-scale applications. By focusing only on tests affected by new code changes, this technique increases the efficiency of the testing process. The security test shown by Zhao et al. (2023) is more important in

the cloud environment. This article highlights the need for vulnerability assessment and penetration testing to identify and mitigate potential threats and ensure the long-term security and integrity of cloud-based applications. Additionally, Ge et al. (2023) mention a hybrid testing approach that combines automated and clustered testing, leveraging the strengths of both to achieve widespread testing and a deeper understanding of potential problems. Köroğlu et al. (2020) discuss the integration of autonomous computing capabilities into existing systems. This approach improves system performance by reducing the need for human intervention and automating the response to system changes, which is particularly useful for large-scale complex systems.

Lastly, there are also papers that highlight the application of specific test methods and optimization techniques that are tailored to different system architectures. Microservice architecture testing by Jharko (2021) shows different levels of end-to-end testing. This is important to ensure the integrity of microservices, which are often complex and contain many interacting components. Gong and Cai (2023) discuss the use of exploratory-based software engineering techniques (SBST) that use metaheuristic optimization techniques to efficiently generate test results, especially useful in scenarios where traditional methods are not feasible due to the size of the software system.

RQ3: How do different software development methodologies and approaches to software development influence the choice and effectiveness of testing methods in large-scale software development?

This section is to understand the impact of different software development methodologies and approaches on the choice and effectiveness of test methods, particularly in the context of large-scale software development. This question is important because it examines how different development frameworks and strategies affect testing, which testing methods are preferred, and how effective these methods are in ensuring the quality and reliability of large-scale software systems. **Table 2** shows the different focuses of software development methodologies.

Table 2. Summary of the different focuses of different software development methodologies and approaches to software development.

Focus	References
Flexibility and Rapid Iteration	(Aggarwal et al., 2018; Guo et al., 2022; Lisitsyn et al., 2021; Mezhuyev et al., 2019; Zhong et al., 2019)
Sequential and Phase-Based Methodologies	(Gong and Cai, 2023; Jharko, 2021)
Security and Performance	(Lei et al., 2019; Yurtseven and Bagriyanik, 2020; Zhao et al., 2023)
Hybrid and Evolving Development Approaches	(Ge et al., 2023; Köroğlu et al., 2020)
Impact of Development Culture and Practices	(Aggarwal et al., 2018; Ge et al., 2023; Guo et al., 2022; Lisitsyn et al., 2021; Mezhuyev et al., 2019; Zhong et al., 2019)

Agile methodologies offer a distinct advantage in their capacity for flexibility and rapid iteration, allowing teams to promptly respond to evolving requirements or market conditions and deliver incremental value. However, these methodologies are susceptible to scope creep if requirements are not rigorously defined or managed,

potentially leading to project delays or overruns. On the other hand, sequential and phase-based methodologies provide a structured framework for software development, facilitating effective project planning and management. Nonetheless, they may suffer from limited flexibility, hindering their ability to accommodate changes once the project has commenced and potentially resulting in resistance to change or scope creep.

Moreover, security and performance-focused methodologies offer the advantage of prioritizing risk mitigation by identifying and addressing security vulnerabilities, performance bottlenecks, and scalability limitations. However, they may introduce increased overhead in terms of time, effort, and resources required for comprehensive testing, validation, and compliance activities. Meanwhile, hybrid and evolving development approaches afford organizations the flexibility to tailor their processes to suit specific project or team needs. Yet, these approaches can bring about complexity in managing diverse methodologies, practices, and tools across multiple projects or teams within an organization.

Furthermore, the impact of development culture and practices cannot be overstated. A culture aligned with effective development practices fosters teamwork, communication, and collaboration, contributing positively to project outcomes. Conversely, entrenched or resistant cultures may impede adaptation or innovation, hindering progress and stifling organizational growth.

The adoption of a development approach that prioritizes flexibility, rapid iteration, and continuous delivery often requires an adaptive and agile testing method, makes the need for an equally adaptive and agile testing methodology. Automated and AI-enhanced testing methods, as shown in papers (Aggarwal et al., 2018; Guo et al., 2022; Lisitsyn et al., 2021; Mezhuyev et al., 2019; Zhong et al., 2019), are very suitable for this method due to their ability to track changes and frequent updates. It enables continuous integration and testing, ensuring continuous and efficient validation of new developments.

Conversely, sequential, and phase-based methodologies are more structured and comprehensive than the more traditional, sequential method where development phases are clearly defined and divided. This approach is proven in a systematic and optimization-based test method, as shown in the papers (Gong and Cai, 2023; Jharko, 2021). Each development stage undergoes thorough testing to ensure the comprehensive evaluation of all components. This comprehensive approach to testing, while time-consuming, provides reliability, which is essential for large-scale systems.

Furthermore, for methodologies that emphasize security and performance, test methods that specifically target these areas are important, especially in cloud-based and high-load environments. As shown in papers (Lei et al., 2019; Yurtseven and Bagriyanik, 2020; Zhao et al., 2023), performance, load and security testing become an integral part of the development process. This method ensures that the software not only meets functional requirements, but also performs optimally in various environments and maintains strict security standards.

Moreover, hybrid or incremental development approaches, which can mix various aspects of traditional and modern methodologies, often use a combination of different testing strategies. This is reflected by Ge et al. (2023) and Köroğlu et al. (2020), which use a mixture of automatic, manual, and special testing strategies. This approach meets the unique needs of large-scale complex systems and provides greater

flexibility and adaptability.

Finally, these testing strategies play a key role in building a software development culture, as introduced in DevOps—a process that emphasizes continuous development, integration, and deployment requires automatic, continuous, and integrated testing methods throughout the development and business cycle. It is clear from the discussion (Aggarwal et al., 2018; Guo et al., 2022; Ge et al., 2023; Lisitsyn et al., 2021; Mezhuyev et al., 2019; Zhong et al., 2019) that testing methods that support fast and continuous delivery of software are crucial.

RQ4: How are emerging technologies like AI and machine learning integrated into existing testing methods to enhance efficiency and sustainability in large-scale software applications?

This section aims to integrate AI and machine learning in established testing methods for large-scale software applications. Investigating the transformative impact of these emerging technologies, we aim to understand their role in enhancing efficiency and sustainability. As organizations adopt AI and Machine Learning, synergy with traditional testing practices becomes crucial. This research question delves into the mechanisms through which these advancements optimize testing paradigms, offering insights into their influence on reliability and performance in the ever-evolving landscape of software development.

Table 3. Summary of the different focuses of different software development methodologies and approaches to software development.

Focus	Description	References
Integration of AI and Machine Learning in testing	Discuss the integration of AI and machine learning technologies into existing testing methods.	(Aggarwal et al., 2018; Guo et al., 2022; Lisitsyn et al., 2021; Mulla and Jayakumar, 2021; Mezhuyev et al., 2019; Zhong et al., 2019
Flexibility and Rapid Iteration in Testing with AI/ML	Examining the flexibility and rapid iteration aspects concerning AI and ML in testing, these references may shed light on how these technologies contribute to agile and iterative testing processes.	(Aggarwal et al., 2018; Guo et al., 2022; Mulla and Jayakumar, 2021)
Hybrid Development Approaches with AI/ML	The amalgamation of AI and ML in hybrid and evolving development approaches. It may highlight how these technologies contribute to the adaptability and scalability of testing practices in response to dynamic changes in development methodologies	(Aggarwal et al., 2018; Ge et al., 2023; Köroğlu et al., 2020; Yin, 2020; Zhong et al., 2019)

Referring to **Table 3**, integration of AI and machine learning in testing presents significant advantages and challenges. Firstly, AI and ML algorithms facilitate automated test case generation, leveraging historical data and code analysis to reduce manual effort and optimize test coverage. However, this advantage is countered by dependencies on large volumes of quality data and the complexity of implementation, demanding specialized skills and resources for development and deployment.

Furthermore, the flexibility and rapid iteration enabled by AI/ML in testing offer both promise and complexity. Automated test case generation adapts to evolving requirements, while dynamic prioritization techniques enhance efficiency. Yet, the complexity of implementation persists, alongside the challenge of data dependency, particularly in dynamic testing environments. Moreover, hybrid development approaches incorporating AI/ML provide adaptability and scalability in testing. These technologies enable quick adjustments to changing requirements and support efficient

resource allocation. Nonetheless, integrating AI/ML into existing workflows requires careful planning and may reveal skill and knowledge gaps within teams, necessitating training and cultural shifts to fully leverage their benefits.

Automated testing offers numerous benefits to software development, but it faces challenges in certain situations. In such cases, Artificial Intelligence (AI) emerges as a solution, addressing and overcoming these challenges. These papers (Aggarwal et al., 2018; Guo et al., 2022; Lisitsyn et al., 2021; Mulla and Jayakumar, 2021; Mezhuyev et al., 2019; Zhong et al., 2019) discuss on a comprehensive overview of various AI techniques applied in software testing, highlighting their role in automation testing, self-healing execution of Selenium tests, automating API test generation, and visual validation automation testing. The focus is on how AI contributes to increasing efficiency, improving the quality of test cases, and addressing challenges in different aspects of the software testing process. The practical applications of AI tools, such as Eggplant AI, App Vance, and Test.ai, are also discussed, providing insights into their use for automated testing. Overall, the paper demonstrates the integration of AI and machine learning to enhance testing practices and efficiency in the development lifecycle.

A large-scale software application that utilizes artificial intelligence (AI) technology, specifically in the context of a virtual assistant that interacts with users through natural voice as discussed by Zhong et al. (2019) Understanding the characteristics and complexity of such systems, like Google Assistant, is crucial when investigating how emerging technologies like AI and machine learning are integrated into existing testing methods to enhance efficiency and sustainability in large-scale software applications. This system provides an example of a sophisticated AI-powered application, and studying its testing methods could offer insights into strategies for efficiently testing complex and widely used software systems. This involves continuous integration of emerging technologies like AI and machine learning into testing methods, ensuring that the testing processes evolve to maintain efficiency and effectiveness in the face of system updates and increasing user demands.

Moreover, Guo et al. (2022) introduce an innovative framework leveraging Generative Adversarial Networks (GANs) for the automatic generation of high-quality test cases, responding to the escalating challenges posed by the expanding scale and intricacy of contemporary software systems. The incorporation of GANs, a facet of artificial intelligence and machine learning, explicitly aligns with the aim of introducing flexibility and facilitating rapid iteration within the testing domain. Traditional software testing approaches face increasing costs in tandem with system complexity. By employing GANs, the proposed framework seeks to alleviate this challenge by automating test case generation while upholding software reliability. This automation not only enhances efficiency but also embodies flexibility, as indicated by the successful application of the GAN-based method across various testing scenarios, including unit and integration testing. The experimental results, particularly the comparison against random testing, empirically underscore the superior performance of the GAN-based approach, emphasizing its potential to streamline testing processes and accommodate rapid iterations in the dynamic landscape of software development.

Mulla and Jayakumar (2021), however, emphasize the transformative impact of artificial intelligence (AI) and machine learning (ML) on software testing practices.

While recognizing the nascent stage of their adoption, the text emphasizes the promising potential of AI and ML to revolutionize testing procedures. The versatility of AI is particularly highlighted through applications like automation testing, self-healing execution of Selenium tests, automating API test generation, and visual validation automation testing. These applications showcase AI's ability to address various aspects of testing, making it adaptable to diverse scenarios. Moreover, the integration of machine learning enhances the precision and efficiency of software testing by analyzing functions, ensuring accurate results, and reducing the time required for development. In the context of the research question, this integration is not just about automating existing processes but introduces flexibility and supports rapid iteration in testing methodologies. AI and ML technologies contribute to creating a more intelligent, adaptive, and efficient testing environment that is better equipped to handle the complexities of large-scale software applications. This evolution signifies a significant paradigm shift in how testing is approached in the dynamic landscape of software development.

Moving forward, Ge et al. (2023) present a comprehensive solution to the challenges associated with crowdsourced testing in the realm of mobile application development. The identified problem revolves around the diverse testing experience levels of crowd workers, posing a substantial threat to the quality of crowdsourced testing outcomes. To mitigate this issue, the authors propose a testing assistance approach that capitalizes on Android automated testing techniques, specifically employing dynamic and static analyses. These analyses culminate in the creation of an Annotated Window Transition Graph (AWTG) model for the App Under Test (AUT). The AWTG model serves as a pivotal component of a testing assistance pipeline, which encompasses test task extraction, recommendation, and guidance functionalities. Experimentation on real-world AUTs validates the efficacy of this approach, demonstrating its capacity to significantly enhance both the effectiveness and efficiency of crowdsourced testing. The relevance of this research to hybrid development approaches with AI/ML lies in its explicit integration of Android automated testing methodologies, showcasing the practical implementation of machine-assisted human intelligence. The AWTG model, constructed through a combination of dynamic and static analyses, aligns with the hybrid development paradigm by providing a flexible and adaptive framework. This approach embodies the rapid iteration and flexibility crucial in hybrid development scenarios, illustrating the symbiotic relationship between crowdsourced testing enhancements and advancements in AI/ML technologies discussed in the paper.

The empirical study by Köroğlu et al. (2020) conducted on over 12,000 open-source Android apps revealed intriguing insights into the test automation culture prevalent among mobile app developers. The findings indicate that only 8% of the mobile app development projects leverage automated testing practices, shedding light on the current state of test automation adoption in the Android app development ecosystem. This could involve investigating potential barriers to adoption, such as lack of awareness, resources, or expertise, and proposing strategies to overcome these challenges. Additionally, exploring successful case studies or best practices in automated testing implementation could provide valuable insights for increasing adoption rates in the Android app development ecosystem. Due to cost and time

constraints, developers may perceive automated testing as time-consuming and expensive to set up initially. Therefore, it is recommended to explore cost-effective solutions or tools tailored specifically for Android app testing. Secondly, recognizing the complexity of integration into existing development workflows, particularly in the fast-paced environment of Android app development, emphasizes the need for strategies to simplify the integration process for developers. Furthermore, the study uncovers that developers tend to follow similar test automation practices across projects, emphasizing a consistent approach within the developer community. Notably, popular projects, as measured by metrics such as the number of contributors, stars, and forks on GitHub, exhibit a higher likelihood of adopting test automation practices. The correlation between project popularity and test automation adoption underscores the significance of automated testing in contributing to the overall success and recognition of mobile apps. Leveraging AI and ML techniques, such as evolutionary algorithms or reinforcement learning approaches, can further enhance test case generation and selection processes, leading to more comprehensive test suites and improved defect detection. Thus, embracing automated testing practices and integrating AI and ML techniques can significantly benefit the Android app development ecosystem by ensuring higher quality and reliability in mobile applications.

Relating these findings to hybrid development approaches (Yin, 2020) with AI/ML, the study indirectly underscores the need for efficiency and effectiveness in the software development lifecycle, especially in the dynamic and rapidly evolving landscape of mobile app development. Hybrid development approaches, which often involve a combination of native and web technologies, can benefit from automated testing practices to ensure the quality, reliability, and performance of the developed applications. AI/ML technologies can play a pivotal role in enhancing automated testing by providing intelligent test case generation, identifying potential issues, and assisting in the optimization of testing strategies. The observed correlation between project popularity and test automation adoption aligns with the principles of efficiency and scalability inherent in hybrid development, where leveraging advanced technologies like AI/ML becomes essential for staying competitive and delivering high-quality applications.

Furthermore, AI-driven testing (Braiek and Khomh, 2020), a method leveraging AI and ML to automate testing activities, extends its applications from functional and visual testing to user interface testing and auto-correcting element selectors. For instance, in the field of AI Testing (Braiek and Khomh, 2020), researchers and practitioners are exploring how AI and ML technologies can construct the next generation of testing tools, capitalizing on advancements in cloud computing and big data. These tools aim to bridge the gap between human-present and machine-driven testing capabilities. Additionally, the adoption of AI and ML techniques in Testing AI has led to the development of AI-based testing tools addressing software quality and testing challenges. Despite challenges in standardization and best practices, the potential for AI to revolutionize automated testing is evident. Furthermore, in Self-Testing (Braiek and Khomh, 2020), the dynamic adaptation in AI-based systems prompts the development of self-testing mechanisms. Approaches such as Replication with Validation (RV) and Safe Adaptation with Validation (SAV) have been proposed, offering distributed testing and robust system designs. These examples underscore the

transformative potential of AI and ML integration in testing methodologies.

Furthermore, while exploring the integration of AI and machine learning into existing testing methods, it's imperative to consider potential challenges and criticisms associated with these technologies. Despite their promising potential, AI-driven testing solutions may encounter hurdles in real-world implementation. For instance, dependency on high-quality, representative data for training AI models could pose challenges in dynamic testing environments. Additionally, the complexity and overhead of implementing AI-driven testing solutions may present obstacles for organizations lacking the necessary expertise and resources. Moreover, concerns regarding the interpretability and transparency of AI models raise questions about the reliability and accountability of automated testing outcomes. Addressing these challenges requires careful consideration and mitigation strategies to ensure the effective integration of AI and machine learning into traditional testing methodologies.

In conclusion, the insights gleaned from the discussions (Aggarwal et al., 2018; Ge et al., 2023; Guo et al., 2022; Köroğlu et al., 2020; Lisitsyn et al., 2021; Mezhuyev et al., 2019; Yin, 2020; Zhong et al., 2019) underscore the pivotal role of testing methods that enable rapid and continuous software delivery. These findings align with the essence of RQ4, which delves into the integration of emerging technologies like AI and machine learning into conventional testing approaches. The emphasis on efficiency and sustainability in large-scale software applications resonates with the broader industry's recognition of the need to harness innovative technologies to enhance testing practices for contemporary software development challenges.

4. Discussion

The significance of automated and AI-enhanced testing in the context of large-scale software applications becomes increasingly evident when we delve deeper into the research findings presented in papers (Aggarwal et al., 2018; Guo et al., 2022; Lisitsyn et al., 2021; Mezhuyev et al., 2019; Zhong et al., 2019). This method stands out for its ability to address the unique challenges posed by the scale and complexity of modern software systems.

One of the key advantages of automated and AI-enhanced testing is its ability to streamline repetitive and time-consuming tasks. This not only leads to a more efficient use of resources but also ensures a higher degree of accuracy and consistency in testing. For instance, Zhong et al. (2019) highlight how automation tools have significantly reduced the issues associated with manual testing in Android applications. This reduction is not just in terms of time and resources but also in the minimization of human error, which is a critical factor in maintaining software quality.

Moreover, the adaptability of AI-enhanced testing is particularly relevant for large-scale applications that often undergo rapid changes. Traditional testing methods can struggle to keep pace with such rapid development cycles, leading to potential gaps in testing coverage. In contrast, AI-enhanced methods, as demonstrated in the research, can quickly adapt to changes in the software, ensuring that all aspects of the application are thoroughly tested. The use of deep neuroevolutionary in GUI-based testing, as explored by Aggarwal et al. (2018) exemplifies this adaptability. By combining deep learning with evolutionary algorithms, this approach has shown

significant improvements in both the breadth and depth of testing, far surpassing traditional methods.

The quality of testing is another area where automated and AI-enhanced methods excel. Guo et al. (2022), Lisitsyn et al. (2021) and Mezhuyev et al. (2019) discuss how AI models trained to generate high-quality test cases have led to a notable decrease in the time and resources dedicated to manual test case development. This shift is not merely about efficiency; it's about enhancing the quality of the testing process itself. High-quality, AI-generated test cases can cover a broader range of scenarios and conditions than manually created ones, leading to a more robust and reliable software product.

The potential for future research in this area is vast. Integrating automated and AI-enhanced testing more deeply with agile and DevOps practices could further enhance the responsiveness and effectiveness of the testing process. Additionally, customizing these methods to fit various types of large-scale applications, including those with unique or highly specialized requirements, presents a promising area of study. Furthermore, the exploration of emerging technologies, particularly advanced machine learning techniques, could lead to even more sophisticated and efficient testing methodologies.

In sum, automated and AI-enhanced testing represents a significant advancement in software testing for large-scale applications. Its efficiency, adaptability, and potential for enhancing the quality of testing make it a pivotal approach in the current technological landscape. As software systems continue to grow and become complex, the role of these advanced testing methods will become increasingly central, driving innovation, and ensuring the reliability of software products.

5. Conclusion

In conclusion, our research has delved into critical questions surrounding testing in large-scale software applications. We systematically investigated predominant testing methods and their contributions to sustainability, providing insights into the efficiency and effectiveness of these approaches. Furthermore, we explored the intricate relationship between software development methodologies and the choice of testing methods in large-scale development, uncovering the nuanced influences that shape testing practices. Addressing the challenges and environmental factors affecting sustainable testing strategies for large-scale software applications, we identified key issues and proposed viable solutions for consideration. Lastly, our inquiry extended to the integration of emerging technologies such as AI and machine learning into existing testing methods, revealing their potential to revolutionize efficiency and sustainability in the dynamic landscape of large-scale software applications. Overall, our comprehensive research illuminates crucial aspects of testing methodologies, development approaches, challenges, and cutting-edge technologies, contributing to a deeper understanding of the intricate dynamics inherent in ensuring the reliability and sustainability of large-scale software products. In addition to the insights provided in our research, it is crucial to consider avenues for future exploration in the realm of sustainable testing strategies for large-scale software applications. One potential direction for future research involves a comprehensive examination of existing testing

methodologies and their adaptability to evolving software development landscapes. By focusing on refining and optimizing established testing methods, researchers can uncover novel strategies for ensuring the reliability and sustainability of large-scale software products. Furthermore, investigating the integration of emerging technologies, such as AI and machine learning, into existing testing frameworks presents an exciting opportunity to revolutionize testing practices and enhance efficiency. Future studies could also explore innovative approaches for addressing the challenges and environmental factors identified in our research, thereby advancing the field of software testing and contributing to the development of more resilient and sustainable software systems

Author contributions: Conceptualization, RKR and TT; methodology, RKR; software, RKR; validation, RKR; formal analysis, TT, LCF and MD; investigation, RKR; resources, RKR; data curation, RKR; writing—original draft preparation, TT, LCF, and MD; writing—review and editing, RKR; visualization, RKR; supervision, RKR; project administration, RKR; funding acquisition, MD. All authors have read and agreed to the published version of the manuscript.

Conflict of interest: The authors declare no conflict of interest.

References

Agarwal, A., Gupta, S., & Choudhury, T. (2018). Continuous and Integrated Software Development using DevOps. In: Proceedings of the 2018 International Conference on Advances in Computing and Communication Engineering (ICACCE). https://doi.org/10.1109/icacce.2018.8458052

Aggarwal, P. K., Grover, P. S., & Ahuja, L. (2018). Incorporating Autonomic Capability as Quality Attribute for a Software System. In: Proceedings of the 2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO). https://doi.org/10.1109/icrito.2018.8748488

Ahlgren, J., Berezin, M., Bojarczuk, K., et al. (2021). Testing Web Enabled Simulation at Scale Using Metamorphic Testing. In: Proceedings of the 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). https://doi.org/10.1109/icse-seip52600.2021.00023

Al Alamin, M. A., & Uddin, G. (2021). Quality Assurance Challenges for Machine Learning Software Applications During Software Development Life Cycle Phases. In: Proceedings of the 2021 IEEE International Conference on Autonomous Systems (ICAS). https://doi.org/10.1109/icas49788.2021.9551151

Alferidah, S. K., & Ahmed, S. (2020). Automated Software Testing Tools. In: Proceedings of the 2020 International Conference on Computing and Information Technology (ICCIT-1441); 09–10 September 2020; Tabuk, Saudi Arabia. pp. 1–4. https://doi.org/10.1109/ICCIT-144147971.2020.9213735

Ali, S., & Li, H. (2019). Moving Software Testing to the Cloud: An Adoption Assessment Model Based on Fuzzy Multi-Attribute Decision Making Algorithm. In: Proceedings of the 2019 IEEE 6th International Conference on Industrial Engineering and Applications (ICIEA). https://doi.org/10.1109/iea.2019.8714986

Ali, S., Sun, H., Zhao, Y., et al. (2019). Testing-based Model Learning Approach for Legacy Components. In: Proceedings of the 2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST). https://doi.org/10.1109/ibcast.2019.8667149

Alshahwan, N., Harman, M., & Marginean, A. (2023). Software Testing Research Challenges: An Industrial Perspective. In: Proceedings of the 2023 IEEE Conference on Software Testing, Verification and Validation (ICST). https://doi.org/10.1109/icst57152.2023.00008

Barroca, L., Sharp, H., Salah, D., et al. (2015). Bridging the gap between research and agile practice: an evolutionary model. International Journal of System Assurance Engineering and Management, 9(2), 323–334. https://doi.org/10.1007/s13198-015-0355-5

Braiek, H. B., & Khomh, F. (2020). On testing machine learning programs. Journal of Systems and Software, 164, 110542.

https://doi.org/10.1016/j.jss.2020.110542

Dirim, S., & Sozer, H. (2020). Prioritization of Test Cases with Varying Test Costs and Fault Severities for Certification Testing. In: Proceedings of the 2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). https://doi.org/10.1109/icstw50294.2020.00069

Gao, C., Luo, W., & Xie, F. (2022). An Ontology-based Knowledge Base System for Military Software Testing. In: Proceedings of the 2022 9th International Conference on Dependable Systems and Their Applications (DSA). https://doi.org/10.1109/dsa56465.2022.00043

Ge, X., Yu, S., Fang, C., et al. (2023). Leveraging Android Automated Testing to Assist Crowdsourced Testing. IEEE Transactions on Software Engineering, 49(4), 2318–2336. https://doi.org/10.1109/tse.2022.3216879

Gong, J., & Cai, L. (2023). Analysis for Microservice Architecture Application Quality Model and Testing Method. In: Proceedings of the 2023 26th ACIS International Winter Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD-Winter). https://doi.org/10.1109/snpd-winter57765.2023.10223960

Guo, X., Okamura, H., & Dohi, T. (2022). Automated Software Test Data Generation with Generative Adversarial Networks. IEEE Access, 10, 20690–20700. https://doi.org/10.1109/access.2022.3153347

Hu, X., Liu, Z., Xia, X., et al. (2023). Identify and Update Test Cases When Production Code Changes: A Transformer-Based Approach. In: Proceedings of the 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE). https://doi.org/10.1109/ase56229.2023.00165

Hurdugaci, V., & Zaidman, A. (2012). Aiding Software Developers to Maintain Developer Tests. In: Proceedings of the 2012 16th European Conference on Software Maintenance and Reengineering. https://doi.org/10.1109/csmr.2012.12

Hynninen, T., Kasurinen, J., Knutas, A., et al. (2018). Software testing: Survey of the industry practices. In: Proceedings of the 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). https://doi.org/10.23919/mipro.2018.8400261

Jarman, D., Smith, R., Gosney, G., et al. (2019). Applying Combinatorial Testing to Large-Scale Data Processing at Adobe. In: Proceedings of the 2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). https://doi.org/10.1109/icstw.2019.00051

Jharko, E. (2021). Some Aspects of Quality Assurance in the Development of Digital Systems. In: Proceedings of the 2021 14th International Conference Management of Large-Scale System Development (MLSD). https://doi.org/10.1109/mlsd52249.2021.9600164

Köroğlu, Y., Sen, A., & Akin, A. (2020). Automated Functional Test Generation Practice for a Large-Scale Android Application. In: Proceedings of the 2020 Turkish National Software Engineering Symposium (UYMS). https://doi.org/10.1109/uyms50627.2020.9247060

Lei, Q., Liao, W., Jiang, Y., et al. (2019). Performance and Scalability Testing Strategy Based on Kubemark. In: Proceedings of the 2019 IEEE 4th International Conference on Cloud Computing and Big Data Analysis (ICCCBDA). https://doi.org/10.1109/icccbda.2019.8725658

Li, H., Chen T. H., Hassan A. E., et al. (2018). Flora. Adopting Autonomic Computing Capabilities in Existing Large-Scale Systems. Available online: https://petertsehsun.github.io/papers/peter_icse_seip2018.pdf (accessed on 15 February 2024).

Li, H., Chen, P., Liu, Y., et al. (2022). Design and application of an automated test platform based on image recognition. In: Proceedings of the 2022 IEEE International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC); 5–7 August 2022; Chongqing, China. pp. 363–366. https://doi.org/10.1109/SDPC55702.2022.9915914

Lisitsyn, A. B., Nikitina, M. A., & Chernukha, I. M. (2021). Product Quality and Safety Management in Large-Scale Systems. In: Proceedings of the 2021 14th International Conference Management of Large-Scale System Development (MLSD). https://doi.org/10.1109/mlsd52249.2021.9600217

Liu, P., Li, Y., Zeng, L., et al. (2021). Big Data-based Testing: Characteristics, Challenges, and Future Directions. In: Proceedings of the 2021 7th International Symposium on System and Software Reliability (ISSSR). https://doi.org/10.1109/isssr53171.2021.00012

Meng, S., Dai, Y., Luo, L., et al. (2019). Reliability Modeling and Optimization for Large-Scale Network Systems. In: Proceedings of the 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C). https://doi.org/10.1109/qrs-c.2019.00108

Mezhuyev, V., Al-Emran, M., Ismail, M. A., et al. (2019). The Acceptance of Search-Based Software Engineering Techniques: An Empirical Evaluation Using the Technology Acceptance Model. IEEE Access, 7, 101073–101085.

https://doi.org/10.1109/access.2019.2917913

Mulla, N., & Jayakumar, N. (2021). Role of Machine Learning & Artificial Intelligence Techniques in Software Testing. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(6), 2913–2921. https://doi.org/10.17762/turcomat.v12i6.5800

Tsai, C. H., Tsai, S. C., & Huang, S. K. (2021). REST API Fuzzing by Coverage Level Guided Blackbox Testing. In: Proceedings of the 2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS). pp. 291–300. https://doi.org/10.48550/arXiv.2112.15485

Uygun, Y., Oguz, R. F., Olmezogullari, E., et al. (2020). On the Large-scale Graph Data Processing for User Interface Testing in Big Data Science Projects. In: Proceedings of the 2020 IEEE International Conference on Big Data (Big Data). https://doi.org/10.1109/bigdata50022.2020.9378153

Valle-Gomez, K. J., Delgado-Perez, P., Medina-Bulo, I., et al. (2019). Software Testing: Cost Reduction in Industry 4.0. In: Proceedings of the 2019 IEEE/ACM 14th International Workshop on Automation of Software Test (AST). https://doi.org/10.1109/ast.2019.00018

Wang, J., & Ren, D. (2018). Research on Software Testing Technology Under the Background of Big Data. In: Proceedings of the 2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC). https://doi.org/10.1109/imcec.2018.8469275

Wang, J., & Wu, J. (2019). Research on Performance Automation Testing Technology Based on JMeter. In: Proceedings of the 2019 International Conference on Robots & Intelligent System (ICRIS). https://doi.org/10.1109/icris.2019.00023

Wu, H., Yu, S., Niu, X., et al. (2023). Enhancing Fault Injection Testing of Service Systems via Fault-Tolerance Bottleneck. IEEE Transactions on Software Engineering, 1–17. https://doi.org/10.1109/tse.2023.3285357

Xu, X., He, H., Song, W., et al. (2021). Analysis on the Quality Model of Big Data Software. In: Proceedings of the 2021 IEEE/ACIS 19th International Conference on Computer and Information Science (ICIS); 23 August 2021; Shanghai, China. pp. 78–81.

Yin, L. (2020). Test Suite Generation for Software Reliability Testing Based on Hybrid Musa and Markov Method. In: Proceedings of the 2020 7th International Conference on Dependable Systems and Their Applications (DSA). https://doi.org/10.1109/dsa51864.2020.00087

Yurtseven, I., & Bagriyanik, S. (2020). A Review of Penetration Testing and Vulnerability Assessment in Cloud Environment. In: Proceedings of the 2020 Turkish National Software Engineering Symposium (UYMS). https://doi.org/10.1109/uyms50627.2020.9247071

Zatsarinnyy, A., & Ionenkov, Y. (2021). The Efficiency and Quality of Information Systems. In: Proceedings of the 2021 14th International Conference Management of Large-Scale System Development (MLSD). https://doi.org/10.1109/mlsd52249.2021.9600143

Zhang, H., Tuo, A., & Li, G. (2019). Model Checking is Possible to Verify Large-scale Vehicle Distributed Application Systems. In: Proceedings of the 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE). https://doi.org/10.23919/date.2019.8714795

Zhang, Z., Wang, Y., Wang, Z., et al. (2019). How to Effectively Reduce Tens of Millions of Tests: An Industrial Case Study on Adaptive Random Testing. IEEE Transactions on Reliability, 68(4), 1429–1443. https://doi.org/10.1109/tr.2019.2927643

Zhao, Y., Xiao, L., Bondi, A. B., et al. (2023). A Large-Scale Empirical Study of Real-Life Performance Issues in Open Source Projects. IEEE Transactions on Software Engineering, 49(2), 924–946. https://doi.org/10.1109/tse.2022.3167628

Zhi, C., Deng, S., Yin, J., et al. (2019). Quality Assessment for Large-Scale Industrial Software Systems: Experience Report at Alibaba. In: Proceedings of the 2019 26th Asia-Pacific Software Engineering Conference (APSEC). https://doi.org/10.1109/apsec48747.2019.00028

Zhong, H., Zhang, L., & Khurshid, S. (2019). TestSage: Regression Test Selection for Large-Scale Web Service Testing. In: Proceedings of the 2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST). https://doi.org/10.1109/icst.2019.00052

Zimmermann, D. (2022). Automated GUI-based Software-Testing Using Deep Neuroevolution. In: Proceedings of the 2022 IEEE Conference on Software Testing, Verification and Validation (ICST). https://doi.org/10.1109/icst53961.2022.00060

Zimmermann, D., & Koziolek, A. (2023). GUI-Based Software Testing: An Automated Approach Using GPT-4 and Selenium WebDriver. In: Proceedings of the 2023 38th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW). https://doi.org/10.1109/asew60602.2023.00028