Automated Software Testing Using Machine learning: *A Systematic Mapping Study*

Abdellatif AHAMMAD*, Manal EL BAJTA*, Maryam RADGUI*
**SI2M Laboratory, National Institute of Statistics and Applied Economics, Rabat, Morocco*
Emails: aahammad@insea.ac.ma, melbajta@insea.ac.ma, m.radgui@insea.ac.ma

2024 10th International Conference on Optimization and Applications (ICOA) | 979-8-3503-8735-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICOA62581.2024.10754031

***Abstract*—In today's digital world, software quality assurance is a crucial part of the Software Development Life Cycle (SDLC), and automated testing is essential to this effort. This study investigates how machine learning (ML) can augment automated testing by analyzing research from 2006 onward to determine its possible uses, prevalent techniques, and related advantages and disadvantages. Our findings reveal a growing interest in leveraging ML for testing, particularly in tasks like test case generation and user interface validation. Common ML techniques such as symbolic AI, evolutionary algorithms, and deep learning are emerging as the primary methods. ML holds promise for accelerating testing processes, enhancing accuracy, and improving adaptability. However, challenges such as sourcing high-quality training data, understanding complex ML models, and integrating ML with existing tools persist. This study illuminates the transformative potential of ML in testing and provides valuable insights for guiding future research in this dynamic field.**

***Index Terms*—Automated testing, Machine learning, Artificial Intelligence, Software Quality Assurance, Software Development**

I. Introduction

Software testing is one of the most important phases in the life cycle of software development to ensure the quality of it, and it can be found in all types of software, from scientific ones to game development and other fields. However, this phase can sometimes be time-consuming and resource-intensive and not always guarantee comprehensive coverage, especially if it's done through traditional ways. Nowadays, with the emergence of Machine Learning (ML) technologies, the integration of ML in software testing has attracted more interest, especially to enhance and optimize the software testing.

The integration of ML technologies with software testing can help reduce the general software development time and make the process fully automated. It can also help in different cases like test generation, defect prediction, and efficient test prioritization. From that, we can see that the integration of ML and software testing can be a major step to achieve an efficient software development cycle and reduce the amount of effort and cost for that.

This systematic mapping study (SMS) aims to provide a comprehensive overview of the current state of research on software testing using ML, by identifying the most prevalent types of software testing tasks where ML is applied, exploring which software testing activities are the most targeted, and also by addressing which ML algorithms are used and where they are most efficient and where they are not.

The structure of this paper is as follows: the Research Methodology section outlines the mapping questions, search strategy, paper selection criteria, quality assessment process, and data extraction strategy. The Results section presents the findings, organized around the answers to the mapping questions. Finally, the Discussion and Conclusion sections provide an analysis of the results and summarize the key insights from the study.

II. RESEARCH METHODOLOGY

A. Mapping Questions

This systematic mapping study aims to provide a comprehensive overview of the current research landscape in automated software testing using machine learning. The review is guided by the following mapping questions (MQs), which are presented in table I:

TABLE I: Mapping Questions

ID	Mapping Question
MQ1	Which publication sources and channels are the main targets for research on ML-based software testing?
MQ2	Has the frequency of publications on ML-based software testing increased over time?
MQ3	In which specific software testing tasks has machine learning been successfully applied?
MQ4	What ML algorithms and techniques are used in software testing?
MQ5	What are the reported benefits of using machine learning in automated software testing?
MQ6	What are the identified challenges in applying machine learning to automated software testing?

B. Search Strategy and Paper Selection Criteria

To ensure a structured approach, we utilized the PICO (Population, Intervention, Comparison, and Outcomes) [1] framework to identify the research questions, search strategy, and data analysis.

The PICO elements are defined as follows:

- **Population**: the population of interest encompasses general software systems or applications.
- **Intervention**: our focus for the intervention lies in the application of machine learning techniques to optimize or enhance the field of software testing.

- **Comparison**: the comparison is made between traditional automated testing methods and those that employ machine learning.
- **Outcome**: for the outcome focuses on assessing the effectiveness, efficiency, accuracy, and reliability of automated software testing utilizing machine learning.

To identify relevant research on automated software testing using machine learning, the search spanned several major academic databases, including ACM Digital Library, IEEE Xplore, ScienceDirect, Scopus, and Springer Link.The search string used in this study is as follows: (”Automat* Software Test*” **OR** ”Test Automat*”) **AND** (”Machine Learn*” **OR** ”ML” **OR** ”Artificial Intelligence” **OR** ”AI” **OR** ”Deep Learn*” **OR** ”Neural Network*”) **AND** (”Enhance*” **OR** ”Improve*” **OR** ”Optimize*”).This search string combines terms related to automated software testing and various aspects of machine learning and artificial intelligence, along with terms indicating enhancement, improvement, or optimization. To ensure the search was effective and met the specific syntax requirements of different databases, we made minor adjustments to the string for each source.

After gathering the initial search results, each paper was retrieved by the first author, and specific information of each relevant paper was filled in an MS Excel file. We applied a selection process to identify the most relevant studies for this mapping study. As a first step, we eliminated duplicate titles and titles which were clearly not related to the review. To focus on recent advancements, the search results were limited to English-language publications from 2006 onwards, and we applied inclusion and exclusion criteria.

The inclusion criteria were as follows:

- Studies were required to be peer-reviewed research papers (journal articles, conference proceedings, and books) published in English from 2006 onwards.
- Selected papers needed to clearly demonstrate the application of machine learning techniques to enhance various aspects of software testing.
- Studies that offered insights into one or more of the defined research questions were prioritized.

The exclusion criteria were as follows:

- Non-English publications.
- Non-peer-reviewed sources (such as editorials, opinion pieces, and conference abstracts without full papers).
- Studies concentrating exclusively on traditional software testing without any machine learning component.
- Papers with restricted access (such as those requiring passwords).

By applying these criteria, we ensured that the most relevant and high-quality studies were included in this mapping study.

III. Quality Assessment (QA) Process

To maintain the integrity of our SMS, a thorough quality appraisal process is implemented. Each selected study is scrutinized based on the following critical factors:

1) Publication Ranking: Studies are given weighted scores based on conference or journal rankings to indicate established rigor within the field.
    - Conferences: CORE A*: +2 points, CORE A: +1.5 points, CORE B: +1 point, CORE C: +0.5 points, Unranked: +0 points
    - Journals: Q1: +2 points, Q2: +1.5 points, Q3: +1 point, Q4: +0.5 points, Unranked in JCR: +0 points
2) Relevance to Research Questions: Studies directly addressing the core research questions receive the highest scores.
    - Directly addresses core research questions: +1 point
    - Partially relevant: +0.5 points minimal relevance: +0 points
3) Study Completeness: The completeness and clarity of methodology, data analysis, and discussion are evaluated.
    - Thorough methodology, data analysis, and discussion: +1 point
    - Adequate process with room for improvement: +0.5 points
    - Significant shortcomings: +0 points
4) Empirical Value: Studies with empirical findings directly applicable to the systematic mapping study are prioritized.
    - Presents applicable empirical findings: +1 point
    - Limited or absent empirical findings: +0 points

IV. Data Extraction Strategy

Having identified a set of relevant studies through the search and selection process, a data extraction strategy is implemented to gather information required to answer the established mapping questions (MQs). This section details the specific data elements that are extracted from each study.

To extract data from the identified primary studies, we developed the following template shown in Table II.

TABLE II: Description of Data Items and Corresponding Mapping Questions

Data Item	**Value**	**MQ**
General Study ID	Integer	General
Article Title	Name of the article	General
Authors	Set of Names of the authors	General
Journal/Conference Name	Name of the journal/conference	MQ1
Source	which database	MQ1
Year of Publication	Calendar year	MQ2
Types of software testing tasks	List of Software Testing Tasks	MQ3
ML techniques	List of ML techniques	MQ4
Reported benefits	List of reported benefits	MQ5
Identified challenges	List of identified challenges	MQ6

For each question, we identified the necessary data items required to answer the question, which resulted in the table above. To better understand it, here is a set of mapping questions with explanations for the required data items:

MQ1: Which publication sources and channels are the main targets for research on ML-based software testing?

- Journal/Conference Name: Identifies the primary journals and conferences where research is concentrated, indicating the most established and influential venues within the field.

MQ2: Has the frequency of publications on ML-based software testing increased over time?

- Year of Publication: Provides the essential data point to analyze trends, plot growth curves, and identify potential inflection points in the volume of research output.

MQ3: In which specific software testing tasks has machine learning been successfully applied?

- Types of software testing tasks: Explicitly captures whether ML is being used for : Test case generation, Test case prioritization, Defect prediction, Unit testing, Regression testing, Stress and load testing, GUI testing, End-to-end testing, Black-box testing, or in other specific tasks.

MQ4: What ML algorithms and techniques are used in software testing?

- ML techniques: Records the specific ML algorithms, techniques, and models employed, such as Symbolic AI, Evolutionary Algorithms, Deep Learning, Reinforcement Learning, NLP & LLM, Classification, Clustering, Optimization, and other methods and techniques within software testing use cases. Allowing us to identify the most prevalent approaches and patterns within software testing use cases.

MQ5: What are the reported benefits of using machine learning in automated software testing?

- Reported Benefits: Documents the advantages claimed by researchers, helping us understand the key ways ML may improve testing efficiency, accuracy, cost-effectiveness, etc.

MQ6: What are the identified challenges in applying machine learning to automated software testing?

- Identified Challenges: Records the limitations, drawbacks, or complexities highlighted in the research, providing a comprehensive view of the obstacles that need to be overcome.

V. RESULTS

A. *Quality Assessment*

In this systematic mapping study, rigorous quality assessment (QA) was conducted on selected literature to ensure reliability and relevance. Among the selected papers (105 papers), 44.76% (47 papers) scored above the medium level, contributing to our understanding of automated software testing with machine learning (ML). The table III outlines the quality levels of relevant studies in the systematic mapping study. It categorizes papers into five quality levels: Very low, Low, Medium, High, and Very high, based on their respective scores. The majority of papers fall into the Medium and Very low categories, comprising 25.47% and 42.45% of the total, respectively, while the least represented category is Very high, constituting only 5.66% of the total. No studies were excluded based on QA assessments.

TABLE III: Quality levels of relevant studies.

Quality level	Papers	Percent (%)
Very low ($0 < \text{score} \leq 1$)	45	42.85
Low ($1 < \text{score} \leq 2$)	13	12.38
Medium ($2 < \text{score} \leq 3$)	26	24.76
High ($3 < \text{score} \leq 4$)	15	14.28
Very high ($4 < \text{score} \leq 5$)	6	5.71

Table IV illustrates the distribution of papers published in journals (Q1, Q2, Q3, Q4) and conferences (CORE A*, CORE A, CORE B, CORE C), revealing a notable preference for CORE C conferences (16 papers) and Q2 journals (5 papers). Notably, no publications were found in Q1 or Q3 journals, or in CORE B conferences. This analysis underscores the specific venues where research on ML-based software testing is predominantly disseminated.

TABLE IV: Articles by their journal or conference rank

Journals	Number	Conferences	Number
Q1	0	CORE A*	3
Q2	5	CORE A	9
Q3	0	CORE B	0
Q4	6	CORE C	16

B. *MQ1:Publication Sources and Channels*

In our systematic mapping study, we discovered various types of scholarly works (see Table V, Figure 1) from a diverse sources (see Table VI). Most prevalent were conference papers (66), which quickly share research findings. Additionally, we found books (28), offering in-depth knowledge on the subject. We also included journal articles (11), known for their rigorous review process. This diversity showcases the range of valuable research in our field.

TABLE V: Articles by their publication channel.

Publication Type	Count
Conference Paper	66
Book	28
Journal Article	11

TABLE VI: Articles by their publication source.

Source	Count
Scopus	59
IEEE	24
Springer Link	12
Science Direct	6
ACM	4

C. *MQ2: frequency of publications over years*

The number of research papers on ML-based software testing has grown significantly since 2006, as illustrated in Figure 2. While there was a slow and steady increase initially, a clear acceleration is evident from 2019 onwards. This demonstrates

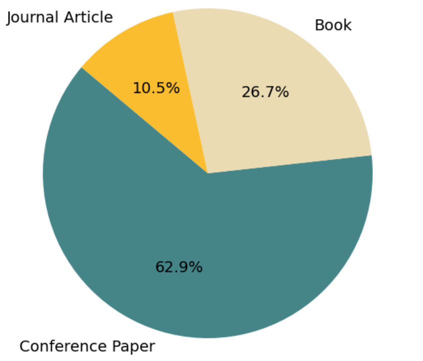

Fig. 1: Distribution of articles by their publication channel.

a growing interest and focus on using machine learning to improve software testing practices. Several factors may have contributed to this growth, including recent breakthroughs in ML techniques.

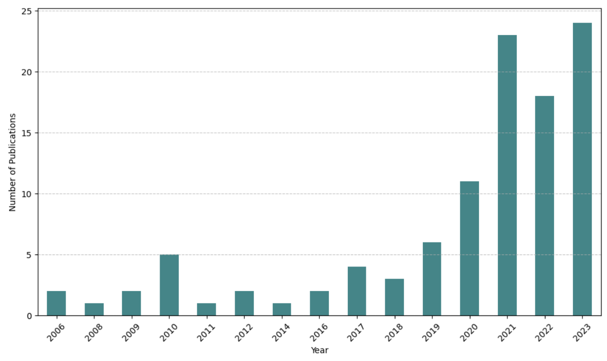

Fig. 2: Publication Distribution Per Year

D. MQ3: ML Applications in Specific Software Testing Tasks

Machine learning applications are concentrated in specific software testing tasks, with ”Test Case Generation” and ”UI Testing” leading the way, as illustrated in Figure 3 and detailed in Table VII. This signifies a growing trend towards leveraging ML to automate and streamline these critical processes, potentially improving both efficiency and effectiveness. While less frequent, applications in ”Test Case Prioritization” and ”Defect Prediction” are noteworthy, demonstrating ML's potential to optimize testing efforts and proactively address software quality concerns.

E. MQ4: Machine Learning Algorithms and Techniques in Software Testing

The landscape of machine learning (ML) techniques in automated software testing is diverse and dynamic, as illustrated in the figure 4 and table VIII. Symbolic AI (10) emerges as the most established technique, followed closely by Deep Learning (9) and Reinforcement Learning (8). This indicates a strong focus on reasoning and knowledge representation alongside the growing popularity of data-driven approaches. NLP & LLM (6) techniques show promise as an emerging trend,

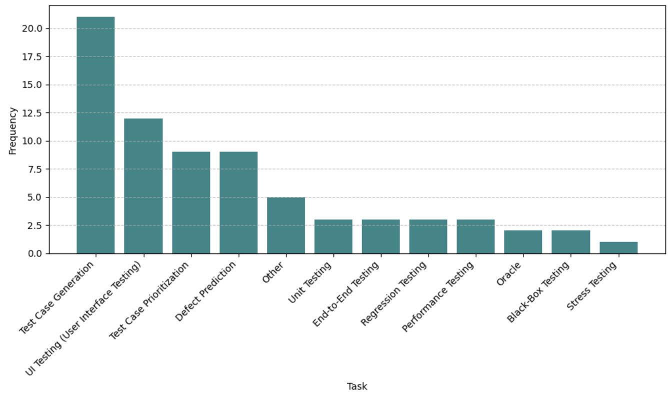

Fig. 3: Frequency of Machine Learning Applications Across Software Testing Tasks

TABLE VII: ML Applications in Specific Software Testing Tasks

Software Testing Task	**Number of Studies**
Test Case Generation	21
UI Testing	12
Test Case Prioritization	9
Defect Prediction	9
Other	5
Unit Testing	3
End-to-End Testing	3
Regression Testing	3
Performance Testing	3
Oracle	2
Black-Box Testing	2
Stress/Load Testing	1

while techniques like Evolutionary Algorithms (4), Clustering Algorithms (3), and Optimization Algorithms (2) find valuable niche applications. Notably, the ’Other’ category (3) signifies continued exploration beyond these established techniques. Overall, this analysis highlights the vibrant and evolving nature of ML applications in software testing, suggesting a future rich in innovation and improvement in software quality and reliability.

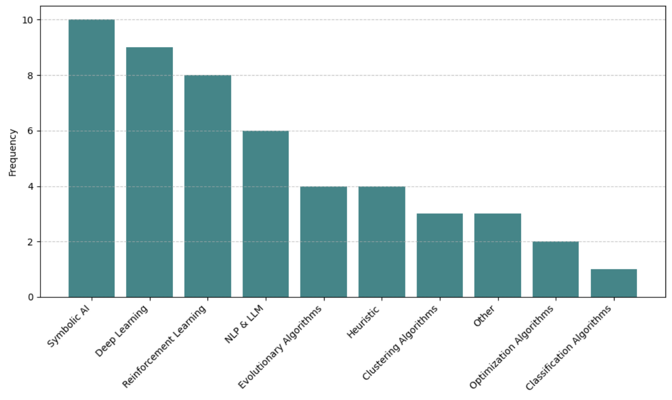

Fig. 4: Frequency of Machine Learning Algorithms and Techniques in Automated Software Testing

TABLE VIII: Frequency of Machine Learning Algorithms and Techniques in Automated Software Testing

Algorithm/Technique	Frequency
Symbolic AI	10
Deep Learning	9
Reinforcement Learning	8
NLP & LLM	6
Evolutionary Algorithms	4
Heuristic	4
Clustering Algorithms	3
Other	3
Optimization Algorithms	2
Classification Algorithms	1

F. MQ5: Reported benefits of using machine learning in automated software testing

Machine learning (ML) offers transformative benefits for automated software testing, enhancing efficiency, accuracy, and adaptability. ML helps to accelerate the execution and analysis of test cases [2], [3], improve code coverage prediction and assertion generation [4], [5], [6], and enhancing the ability to identify complex bugs in embedded systems [7], [8].

ML models can optimize the generation and use of test cases and other resources [9], [10], [11], [12], it can also help in reducing costs through automation [13], [14], and assisting in test suite maintenance [15]. ML-driven insights also empower better decision-making in testing [16], [17], The integration of ML revolutionizes software quality assurance, promising continuous improvement throughout the testing lifecycle.

G. MQ6: Identified challenges in applying machine learning to automated software testing

Several limitations are associated with machine learning (ML) in automated software testing. Primarily, acquiring and maintaining high-quality, extensive datasets to train ML models remains a significant hurdle [18], [19], [2]. The complexity and lack of interpretability of many ML models, especially deep learning models, also pose challenges, potentially leading to trust issues and hindering adoption [20], [6], [21]. Integrating these models into existing tools and processes is another major obstacle to incorporating ML into software testing [22], [23], [24], [17]. Additionally, ensuring the scalability and maintenance of ML models over time is difficult as software evolves and new features are added [9], [25], [3]. Finally, the resource intensiveness of ML models, requiring significant computational power and time, poses a barrier to widespread adoption and integration [14], [26].

VI. Discussion

This systematic mapping study reveals a promising landscape for machine learning (ML) in automated software testing. The rising interest in this area is evident from the increasing number of publications across diverse sources, including 66 conference papers, 28 books, and 11 journal articles. Our analysis illustrates a significant growth trend in ML-based software testing research since 2006, with a notable acceleration observed from 2019 onwards. This indicates a significant interest from the research community in combining ML and automated software testing, especially in conference papers and books published through various sources.

This study also highlights the major tasks targeted in software testing, such as test case generation, one of the earliest tasks where researchers began applying ML techniques, specifically evolutionary algorithms like genetic algorithms [27] and symbolic AI [28]. Other notable tasks include GUI testing, test case prioritization, and defect prediction, with contributions increasingly using deep learning and reinforcement learning. Recently, there has been interest in using large language models (LLMs) and natural language processing (NLP) [26] to enhance automated software testing, influenced by the impact of models like GPT.

ML offers substantial benefits such as enhanced efficiency, accuracy, adaptability, accelerated test case execution, improved code coverage prediction, cost reduction through automation, and better decision-making [16]. These advantages promise continuous improvement in software quality assurance.

However, challenges persist. Acquiring high-quality datasets, interpreting complex ML models, integrating ML with existing tools, and ensuring scalability and maintenance are significant challenges [18], [19], [2]. Additionally, ML's resource intensiveness remains a barrier to widespread adoption.

The results of this systematic mapping study indicate that while ML holds great potential for transforming automated software testing, addressing these challenges is essential to fully realize its benefits.

VII. Conclusion

This study demonstrates the significant promise of machine learning (ML) in automated software testing. The increase in publications, especially since 2019, reflects growing interest and the potential benefits of ML, including enhanced efficiency, accuracy, and adaptability. These benefits are particularly evident in tasks such as test case generation, GUI testing, test case prioritization, and defect prediction.

However, several challenges must be addressed to fully realize ML's potential in software testing. Key issues include acquiring high-quality datasets, interpreting complex models, integrating ML with existing tools, and ensuring scalability. Addressing these challenges will require continued research and collaboration between ML experts and software testing professionals.

Future research should focus on overcoming these obstacles and exploring new ML techniques to further advance the field. Despite the promise of machine learning for automated software testing, several challenges remain before we can fully realize its benefits. Continued investment in research and innovation will be crucial to overcome these

References

[1] Barbara Kitchenham, Stuart Charters, et al. Guidelines for performing systematic literature reviews in software engineering, 2007.

[2] Filippo Ricca, Alessandro Marchetto, and Andrea Stocco. Ai-based test automation: A grey literature analysis. In *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 263–270. IEEE, 2021.

[3] Jerry Gao, ShiTing Li, Chuanqi Tao, Yejun He, Amrutha Pavani Anumalasetty, Erica Wilson Joseph, Akshata Hatwar Kumbashi Sripathi, and Himabindu Nayani. An approach to gui test scenario generation using machine learning. In *2022 IEEE international conference on artificial intelligence testing (AITest)*, pages 79–86. IEEE, 2022.

[4] Giovanni Grano, Timofey V Titov, Sebastiano Panichella, and Harald C Gall. Branch coverage prediction in automated testing. *Journal of Software: Evolution and Process*, 31(9):e2158, 2019.

[5] Giovanni Grano, Timofey V Titov, Sebastiano Panichella, and Harald C Gall. How high will it be? using machine learning models to predict branch coverage in automated testing. In *2018 IEEE workshop on machine learning techniques for software quality evaluation (MaLTeSQuE)*, pages 19–24. IEEE, 2018.

[6] Michele Tufano, Dawn Drain, Alexey Svyatkovskiy, and Neel Sundaresan. Generating accurate assert statements for unit test cases using pretrained transformers. In *Proceedings of the 3rd ACM/IEEE International Conference on Automation of Software Test*, pages 54–64, 2022.

[7] Priyank Singhal, Shakti Kundu, Harshita Gupta, and Harsh Jain. Application of artificial intelligence in software testing. In *2021 10th International Conference on System Modeling & Advancement in Research Trends (SMART)*, pages 489–492. IEEE, 2021.

[8] Kristina Smilgyte and Jovita Nenortaite. Artificial neural networks application in software testing selection method. In *Hybrid Artificial Intelligent Systems: 6th International Conference, HAIS 2011, Wroclaw, Poland, May 23-25, 2011, Proceedings, Part I 6*, pages 247–254. Springer, 2011.

[9] Jayavel Kanniappan, Jithin Gangadharan, Rajesh Kumar Jayavel, and Aravind Nadanasabapathy. Ai-based automated approach for trend data generation and competitor benchmark to enhance voice ai services. In *Proceedings of International Conference on Data Science and Applications: ICDSA 2022, Volume 2*, pages 225–242. Springer, 2023.

[10] Salma Saber, Fatma Elbadry, Hagar Negm, Rana Abu El-Ershad, Omar Magdy, Mohamed Bahnassawi, Reem El Adawi, and AbdElMoniem Bayoumi. Autonomous gui testing using deep reinforcement learning. In *2021 17th International Computer Engineering Conference (ICENCO)*, pages 94–100. IEEE, 2021.

[11] Kamna Solanki, Yudhvir Singh, and Sandeep Dalal. Test case prioritization: an approach based on modified ant colony optimization (m-aco). In *2015 International Conference on Computer, Communication and Control (IC4)*, pages 1–6. IEEE, 2015.

[12] Lukas Rosenbauer, David Pätzel, Anthony Stein, and Jörg Hähner. Transfer learning for automated test case prioritization using xcsf. In *Applications of Evolutionary Computation: 24th International Conference, EvoApplications 2021, Held as Part of EvoStar 2021, Virtual Event, April 7–9, 2021, Proceedings 24*, pages 681–696. Springer, 2021.

[13] Per Erik Strandberg, Mirgita Frasheri, and Eduard Paul Enoiu. Ethical ai-powered regression test selection. In *2021 IEEE International Conference on Artificial Intelligence Testing (AITest)*, pages 83–84. IEEE, 2021.

[14] Mustafa Abdul Salam, Sanaa Taha, and Mohamed Gamal Hamed. Advanced framework for automated testing of mobile applications. In *2022 4th Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, pages 233–238. IEEE, 2022.

[15] Myeongsoo Kim, Qi Xin, Saurabh Sinha, and Alessandro Orso. Automated test generation for rest apis: No time to rest yet. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 289–301, 2022.

[16] Sahar Tahvili and Leo Hatvani. *Artificial Intelligence Methods for Optimization of the Software Testing Process: With Practical Examples and Exercises*. Academic Press, 2022.

[17] Anushka Lal and Girish Kumar. Intelligent testing in software industry. In *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 01–06. IEEE, 2021.

[18] Andrea Stocco. How artificial intelligence can improve web development and testing. In *Companion Proceedings of the 3rd International Conference on the Art, Science, and Engineering of Programming*, pages 1–4, 2019.

[19] Phuoc Pham, Vu Nguyen, and Tien Nguyen. A review of ai-augmented end-to-end test automation tools. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–4, 2022.

[20] Mahmudul Islam, Farhan Khan, Sabrina Alam, and Mahady Hasan. Artificial intelligence in software testing: A systematic review. In *TENCON 2023-2023 IEEE Region 10 Conference (TENCON)*, pages 524–529. IEEE, 2023.

[21] Susmita Haldar and Luiz Fernando Capretz. Explainable software defect prediction from cross company project metrics using machine learning. In *2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 150–157. IEEE, 2023.

[22] Pierre Ortegat, Benoît Vanderose, and Xavier Devroey. Towards automated testing for simple programming exercises. In *Proceedings of the 4th International Workshop on Education through Advanced Software Engineering and Artificial Intelligence*, pages 33–36, 2022.

[23] Ali Sedaghatbaf, Mahshid Helali Moghadam, and Mehrdad Saadatmand. Automated performance testing based on active deep learning. In *2021 IEEE/ACM International Conference on Automation of Software Test (AST)*, pages 11–19. IEEE, 2021.

[24] Naresh Chauhan et al. Role of machine learning in software testing. In *2021 5th International Conference on Information Systems and Computer Networks (ISCON)*, pages 1–5. IEEE, 2021.

[25] Daniel Gerber, Urwashi Kapasiya, Lukas Rosenbauer, and Jörg Hähner. Automation of user interface testing by reinforcement learning-based monkey agents. In *International Conference on Complex Computational Ecosystems*, pages 3–15. Springer, 2023.

[26] Daniel Zimmermann and Anne Koziolek. Automating gui-based software testing with gpt-3. In *2023 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 62–65. IEEE, 2023.

[27] Baowen Xu, Xiaoyuan Xie, Liang Shi, and Changhai Nie. Application of genetic algorithms in software testing. In *Advances in Machine Learning Applications in Software Engineering*, pages 287–317. IGI Global, 2007.

[28] Chongchong Zhao, Guoxin Ai, Xiao Yu, and Xiaofeng Wang. Research on automated testing framework based on ontology and multi-agent. In *2010 Third International Symposium on Knowledge Acquisition and Modeling*, pages 206–209. IEEE, 2010.