information

Review

The Use of AI in Software Engineering: A Synthetic Knowledge Synthesis of the Recent Research Literature

Peter Kokol

Faculty of Electrical Engineering and Computer Science, University of Maribor, Koroška Ulica 46, 2000 Maribor, Slovenia; peter.kokol@um.si

Abstract: Artificial intelligence (AI) has witnessed an exponential increase in use in various applications. Recently, the academic community started to research and inject new AI-based approaches to provide solutions to traditional software-engineering problems. However, a comprehensive and holistic understanding of the current status needs to be included. To close the above gap, synthetic knowledge synthesis was used to induce the research landscape of the contemporary research literature on the use of AI in software engineering. The synthesis resulted in 15 research categories and 5 themes—namely, natural language processing in software engineering, use of artificial intelligence in the management of the software development life cycle, use of machine learning in fault/defect prediction and effort estimation, employment of deep learning in intelligent software engineering and code management, and mining software repositories to improve software quality. The most productive country was China (n = 2042), followed by the United States (n = 1193), India (n = 934), Germany (n = 445), and Canada (n = 381). A high percentage (n = 47.4%) of papers were funded, showing the strong interest in this research topic. The convergence of AI and software engineering can significantly reduce the required resources, improve the quality, enhance the user experience, and improve the well-being of software developers.

Keywords: software engineering; artificial intelligence; machine learning; synthetic knowledge synthesis

Citation: Kokol, P. The Use of AI in Software Engineering: A Synthetic Knowledge Synthesis of the Recent Research Literature. *Information* **2024**, *15*, 354. https://doi.org/10.3390/info15060354

Academic Editor: Aneta Poniszewska-Maranda

Received: 6 May 2024
Revised: 5 June 2024
Accepted: 11 June 2024
Published: 14 June 2024

1. Introduction

In the last couple of years, artificial intelligence (AI) has witnessed exponential growth in development, the rise of AI use, and increased public interest at different levels, from individual to organizational [1–3]. Modern AI utilizes machine learning and other advanced techniques to generate new knowledge, content, hypotheses, and even innovative ideas by identifying patterns and information usually found in extensive data-sized databases. This has catalyzed the use of AI in a broad spectrum of applications, including software design and development [4]. Software companies shifted their focus to deploying AI paradigms to their existing development processes. The academic community started to research and inject new AI-based approaches to provide solutions to traditional software-engineering (SE) problems [5] and critical activities [6]. Examples include software testing [7], maintenance [8], requirements extraction [9], ambiguity resolution [10], software vulnerability detection [11], and software-engineering education [11]. Due to the increasing prevalence of AI use in software engineering, some reviews on the use of AI in software engineering have already been performed. Cao et al. [12] and Mohammadkhani et al. [13] synthetized the problem of explainability due to using black box AI methodologies in SE; Bano et al. and Ozkaya [14] explored the opportunities and challenges of large language models' employment in SE [15]; and Majumdar overviewed the research on generative AI use in analyzing software metadata [8]. The above reviews did not provide a complete and general landscape of AI use in SE. Thus, a general understanding of the current status, possible target applications, practical software engineering usage scenarios, unavoidable limitations, ethical concerns, and challenges is still required [6].

To partially close this gap, this paper presents a comprehensive research landscape of the current research literature on the use of AI in software engineering. The landscape serves as a framework for informing and solving theoretical and practical challenges in software development and design related to AI use. The research community and practicing software engineers can use it to improve their understanding of this fast-growing and highly innovative area. It can also inform novice researchers, grant administrators, software managers, and interested readers lacking specific domain knowledge to develop a perspective on the essential research dimensions. Finally, the landscape can guide and inform further research and serve as a starting point for more formal knowledge and evidence synthesis approaches. To achieve this aim, we set the following research questions:

- What is the volume of the research on using AI in software engineering?
- What is the maturity status of the research?
- What are the volume and dynamics of the production of the research literature on AI use in software?
- How is the research geographically distributed?
- Which information sources informing the scientific community are the most prolific?
- Which funding bodies sponsoring research on AI in software engineering are the most prolific?
- What are the most prolific research themes?

In the next section, the synthetic knowledge synthesis (SKS) is described in more detail, together with the search strategy, followed by a description of the quantitative and qualitative SKS outputs. First, we present and discuss the production dynamics of the research literature, the most productive countries, the institutions, and the prolific source titles. Next, we present the research landscapes and their qualitative interpretations, followed by the literature synthesis.

2. Materials and Methods

The research landscape representing AI use in software engineering was induced by synthetic knowledge synthesis [16]. SKS was developed to respond to the increased complexity of research evidence synthesis due to the accelerated rates of scientific knowledge doubling, exponential growth in scientific information production, and the availability of the research evidence in a digital format. SKS integrates quantitative and qualitative synthesis by triangulating descriptive bibliometrics, bibliometric mapping, and content analysis. In this way, SKS is reducing the weaknesses of traditional knowledge synthesis approaches by requiring fewer resources, being less time- and labor-consuming, and being conducted semi-automatically. As a result, it can be performed on big data-sized corpora, avoiding the sampling issue, which might reduce the analysis to only specific small research sub-areas, making the synthesis not reproducible and focused "on seeing the trees, without considering the forest" [17,18]. Furthermore, using triangulation as the SKS platform provides a more complete picture of the phenomena that helps increase the validity and encompasses the credibility, dependability, confirmability, and transferability (ecological validity) of the research findings [19].

A research landscape is a map/network of the relationships and associations between bibliometric units. In our present study, those units represent author keywords. The links on the map represent the relations between the keywords, proximity, similarity, and node-sized keywords' popularity (frequency of their occurrence in publications). The landscape areas (colored clusters) in our study represent strongly associated authors' keywords, either thematically or timewise. The third component of SKS is content analysis [7], a resourceful approach which, in our case, was used for a qualitative analysis/synthesis of the research publications' content in the form of categories and themes.

The SKS was completed using the following steps:

1. Research publications were harvested from the Scopus bibliographic database using the below search string.
2. Descriptive bibliometric analysis was performed using Scopus's built-in functionality.

3. Author keywords were used as meaningful units of information in the content analysis. First, bibliometric mapping was performed using VOSViewer [6]. Next, using inductive content analysis of the most popular authors' keywords, the node size, links, and proximity between the author keywords in individual clusters and their borders presented in the bibliometric map were analyzed to form categories and identify and name themes.
4. Finally, the themes and subcategory terms were applied to form search strings to locate relevant publications associated with the theme. The most interesting and influential were selected and analyzed to describe the categories' and themes' scope.

Scopus (Elsevier, Amsterdam, The Netherlands) was used as the source bibliographic database because it is deemed the largest abstract and citation database of the reviewed research literature. In addition to the advanced analytics services, it enables 20,000 records to be exported simultaneously.

The search query shown below was constructed using the recommendation provided by Farooq et al. [20]—namely, analyzing and synthesizing the search strategies used in previous review papers and the author keywords found in research papers related to AI and software development.

> *TITLE-ABS-KEY(("artificial intelligence" OR "machine learning" OR "deep learning" OR "intelligent system" OR "support vector machine" OR ("decision tree" AND (induction OR heuristic)) OR "random forest" OR "Markov decision process" OR "hidden Markov model" OR "fuzzy logic" OR "k-nearest neighbor" OR "naive Bayes" OR "Bayesian learning" OR "artificial neural network" OR "convolutional neural network" OR "recurrent neural network" OR "generative adversarial network" OR "deep belief network" OR "perceptron" OR {natural language processing} OR {natural language understanding} OR {general language model}) and ({software engineering} OR {software design} or {software development})) AND PUBYEAR > 2018 AND PUBYEAR < 2025.*

The search was performed on 14 February 2024. The resulting corpus was analyzed using SKS, focusing on bibliometric mapping and content analysis. Finally, using the identified themes and categories as a basis, we performed a literature synthesis and review.

3. Results and Discussion

3.1. Descriptive and Production Bibliometrics

3.1.1. Volume of Research

The search resulted in 9046 publications. There were 5187 conference papers, 3097 articles, 354 conference reviews, 185 book chapters, 179 review papers, 15 editorials, 10 retracted papers, 9 errata, 5 short surveys, 4 notes, and 1 data paper. The recall value of the search was 0.95 (i.e., the fraction of relevant instances among the relevant instances by forming 20 fundamental publications expected to be found in the corpus, meaning that 19 out of 20 expected publications were actually retrieved).

3.1.2. Maturity of Research and Most Prolific Information Sources

The paper type distribution shows that most publications were conference-related papers, revealing that the research is still in a maturation phase and that the core knowledge is still forming. This finding is also confirmed by the fact that the three most prolific titles are conference proceedings, namely Advances in Intelligent Systems and Computing (n = 456), ACM International Conference Proceeding Series (n = 341), and Lecture Notes in Computer Science Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics (n = 249). The first journal source title is Information Sciences (n = 245), followed by another conference proceedings, Ceur Workshop Proceedings (n = 190), indicating that the list of core journals still needs to be established. The H-index of the above source titles lies between 58 and 446, meaning that their quality is averagely high and that most publications still need to be published in top-tier journals. The H-index of the whole sub-field of AI use in software engineering has been 87 for the last 5 years.

The research productivity trend shown in Figure 1 is surprising since the productivity peak of the total number of publications was already reached in 2020. However, the number of publications stabilized in 2022. The decreasing number of publications follows the general trend in the productivity of software engineering publications indexed in Scopus, which also started to show negative trends in 2020 and is mainly due to the decreasing number of conference papers, while the number of articles started to increase in 2022. The above evidence might reveal the start of a positive trend toward reaching research maturity.

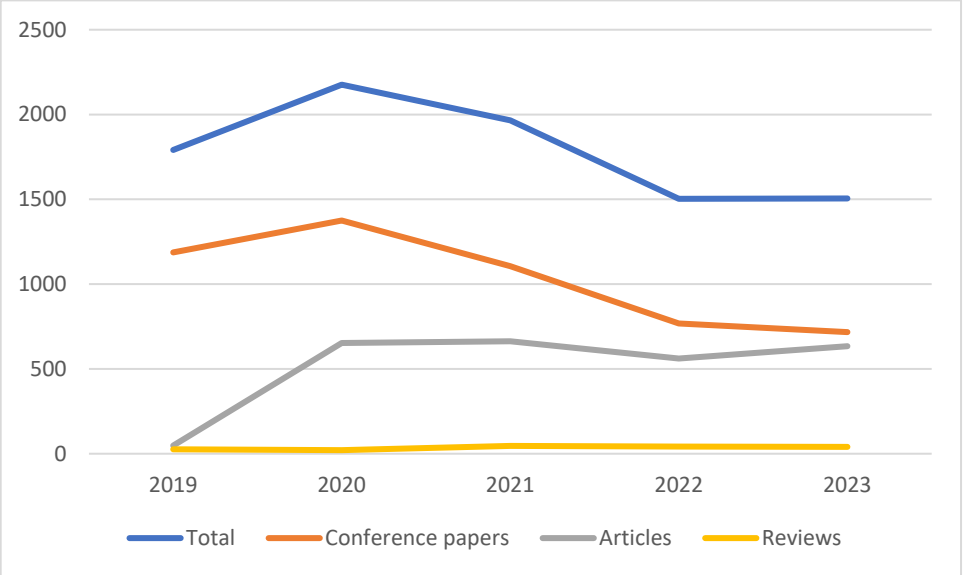

Figure 1. The dynamics of the research literature productivity.

3.1.3. Geographical Distribution of Research

By far the most productive countries were China (n = 2042), followed by the United States (n = 1193), India (n = 934), Germany (n = 445), and Canada (n = 381). This is in line with the Scimago Country Rankings (Elsevier, Amsterdam, The Netherlands), where the United States is ranked first in Software and second in Artificial Intelligence, China is first in Artificial Intelligence and second in Software, and the other top countries are among the 10 most productive in both categories. All the top productive countries also belong to the G20 [21]. China also prevails among the most productive institutions; among the first five, four are from China—namely, the Ministry of Education of the People's Republic of China (n = 90), Chinese Academy of Sciences (n = 89), Nanjing University (n = 72), and Peking University (n = 71). The only non-Chinese institution among the top five, in third place, is Monash University, Australia (n = 73). The most productive US institution is Carnegie Mellon University (n = 46), in 16th place, and the most productive European institution is the Chalmers University of Technology in Sweden (n = 54), in 9th place.

3.1.4. Most Prolific Funding Bodies

Another important indicator of the research state of a scientific field/sub-field is research funding [22]. Our analysis showed that 41.1% of papers are funded, notably more than in many other disciplines [18]; however, less than in a comparable sub-field—namely, the use of AI in pediatrics, where 47.4% of papers are funded [2]. The most prolific funding sponsor is the National Natural Science Foundation of China (n = 987), the National Science Foundation, USA (n = 284), the National Key Research and Development Program of China (n = 226), the Horizon 2020 Framework Programme (n = 154), and the European Regional Development Fund (n = 105).

3.2. Most Prolific Research Themes

Content analysis was performed using SKS Steps 3 and 4 and VOSViewer software, version 1.6.20 (Leiden University, Leiden, The Netherlands). The authors' keyword landscape is shown in Figure 2, and the synthesis of the results is shown in Table 1. The content analysis of the research landscape consists of 146 author keywords (Figure 2), resulting in 15 categories and 5 themes.

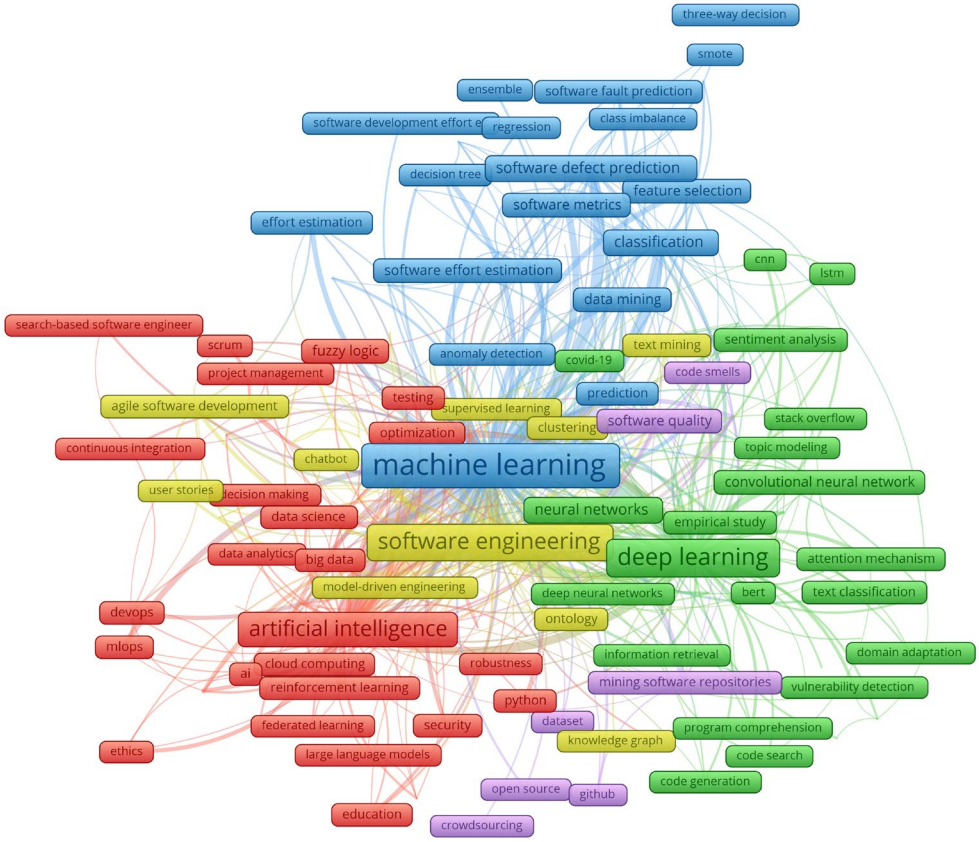

Figure 2. The research landscape of AI use in software engineering. Keywords appearing in 20 or more publications are shown. Colors represent clusters.

Table 1. Clusters and their associated keywords, categories and themes.

Cluster Color	Representative Keywords	Categories	Themes
Red (42 author keywords)	Artificial intelligence (560), Software development (173), Software testing (123), Fuzzy logic (98), Software (73), Big data (65), Reinforcement learning (64)	Ethical use of AI-based software engineering, Use of fuzzy logic in software development and testing, Automation of software testing in an agile environment, Project management of software life cycle using fuzzy logic, Data science and big data in software development	Use of artificial intelligence in management of software development life cycle
Yellow (25 author keywords)	Software engineering (673), Natural language processing (362), Requirement engineering (108), Agile software development (61)	Natural language processing in software development, Natural language processing in software requirements engineering, User stories understanding with natural language processing	Natural language processing (NLP) in software engineering

Table 1. *Cont.*

Cluster Color	Representative Keywords	Categories	Themes
Blue cluster (31 author keywords)	Machine learning (1504), Software development effort estimation (156), Classification (142), Software defect prediction (205), Data mining (102), Artificial neural network (184), Software metrics (84), Feature selection (82)	Software development effort estimation, Data mining in software fault/defect prediction, Machine learning and software metrics	Machine learning in fault/defect prediction and effort estimation
Green (39 author keywords)	Deep learning (770), Neural networks (123), Empirical software engineering (62), Attention mechanism (68), Code generation (34), Code search (33), COVID-19 (30), Technical depth (26), Program comprehension (31)	Deep learning in program comprehension and vulnerability detection, Technical depth and code smell detection, and classification, COVID-19 influence on software engineering	Deep learning in empirical software engineering focusing on code management
Viollet (9 author keywords)	Software quality (86), Software maintenance (62), Mining software repositories (43)	Mining software repositories to improve software quality and software maintenance, Crowdsourcing, GitHub, and open source software as sources for mining software development data	Mining software repositories to improve software quality

Literature Review of Research Categories and Themes

Use of artificial intelligence in management of software development life cycle
Ethical use of AI-based software engineering

Vakkuri et al. [23] noted that ethical considerations are mostly ignored while developing AI-based software systems. Consequently, general and high-level guidelines for managing ethics issues have been proposed [23–25].

Use of fuzzy logic in software development and testing

Fuzzy logic techniques have been used in selecting software requirements from the elicited software requirements or ordering them by preferences [26], cloud-based testing adaption [27], and software effort estimation [28]

Automation of software testing in an agile environment

Artificial intelligence has been used to generate test cases for automatic testing in agile environments [29–31] and to automate other phases of the software development lifecycle [32].

Project management of software life cycle using fuzzy logic

Fuzzy logic techniques have been used to support project management activities like cost and effort estimation [30,31], imputation of missing values in empirical software project management [32], management of outsourcing [33], risk assessment in the agile environment [34] and software product promotion [33].

Data science and big data in software development

Data science and the availability of extensive software development databases enabled the rise of computer- and AI-aided software engineering [35], estimation of story points in agile environments, and support of empirical software engineering in general [34].

Natural language processing (NLP) in software engineering
Natural language processing in software development

NLP technology can drastically improve software development tasks [36]. It can support bug categorization [35], development of more secure software, program decomposition [37], classifying commitments [38], programming and coding [39], writing coherent

and factually correct readmes [40], model-driven engineering [41], deployment of design patterns [42] and traceability management [43].

Natural language processing in software requirements engineering

NLP can support human-performed linguistic analysis in requirements engineering [44,45], such as identifying domain concepts [46–49], establishing traceability links [50], requirement classification [51,52], handling ambiguity [53,54], preference extraction form scenarios [55], classification of non-functional requirements [56], standardization of requirements in agile approaches [57] and requirement elicitation [58].

User stories understanding with natural language processing

NLP has also been used to extract feature, goal, and domain models from user stories [50,53,59], improve the completeness of acceptance criteria [60], or build software structures from user stories [61].

Machine learning in fault/defect prediction and effort estimation
Software development effort estimation

Software development effort estimation is one of the most popular AI techniques used by intelligent software engineering [62], and cost estimation is one of the most crucial software engineering tasks [63]. The different machine-learning algorithms include random forests [64], differential evolution [65], or extreme learning [59]. AI-based effort and cost estimation are used in traditional [66] and agile environments [60].

Data mining in software fault/defect prediction

Data mining and machine learning are used to classify software faults [67] for their detection [62] and prediction [68,69].

Machine learning and software metrics

Machine learning is used to detect code smells [70,71], support software size metric estimation [72], assess software component reusability [66], or predict test flakiness [73].

Deep learning in empirical software engineering focusing on code management
Deep learning in program comprehension and vulnerability detection

In various ways, deep learning is used in software and program code comprehension [74]. For example, the Hybrid-DeepCom and DeepComenter tools automatically generate code comments for Java functional units [75,76]. Similarly, deep learning is also used to generate pseudo-code from program code [77], classify code according to readability [78], or summarize code [79]. In the same context, deep learning is also used to detect code vulnerability [80,81].

Technical depth and code smell detection

Machine learning has recently been often used to detect self-admitted technical depth [72,82,83] or technical debt in general [74,84]. Technical debt is closely linked to code smells and anti-patterns, which are spotted [85,86], classified [87,88], or identified [89] with artificial intelligence techniques.

COVID-19 influence on software engineering

During the COVID-19 pandemic, highly collaborative software development teams were bound to work online in a distributed manner, and many software companies transferred to hybrid models after the pandemic. In such environments, AI can be used to enhance the well-being of developers [77,90].

Mining software repositories to improve software quality
Mining software repositories to enhance the quality of software and software maintenance

Software fault triaging has become a significant activity in terms of software maintenance. Convolutional neural networks were used to learn about developers' fault reports and automatically perform software fault triaging. Triaging has also been performed using

the K nearest neighbor approach on stack traces and categorical features [78]. A long short-term memory algorithm has been used to estimate the fault-fixing times and dependency graphs for the semantic versioning of third-party library components [91]. A fine-tuned transformer has been employed to predict both the objective behind opening an issue and its priority level in GitHub repositories [92].

GitHub and open source software as sources for mining software development data

Data mining in GitHub or open source repositories has been used to create evolving project datasets for intelligent/empirical software engineering, for example, for Eclipse Modelling Framework metamodels formation, anomaly detection or identification of migration reasons [93].

The common denominator in all of the above themes and categories is how to improve the software quality and user and developer satisfaction, reduce the time to market and lower the development costs. The current research on AI use in software engineering is focused on (1) software project management activities like the identification/detection of "dangerous" software components, prediction (estimation) of needed resources and improvement of software testing processes; (2) improving the code and software requirement quality; and (3) making the software use safer, more trustable and reliable.

3.3. Timeline of the Recent Research and Hot Topics

Figure 3 reveals that according to the average age of the author keywords indicated by color, the period 2020–2024 started with the research on the use of data and text mining in combination with deep learning in search-based software engineering, focusing on program comprehension, software metrics, and effort estimation in agile environments. In the middle of the period, the research was mainly linked to machine learning in software testing, mining software repositories for model-driven engineering, code smells, and software development effort estimation. The hot topics seem to be the research on using large language models and explainable machine learning (i.e., decision trees) for fault prediction, code understanding, ethics, and vulnerability detection.

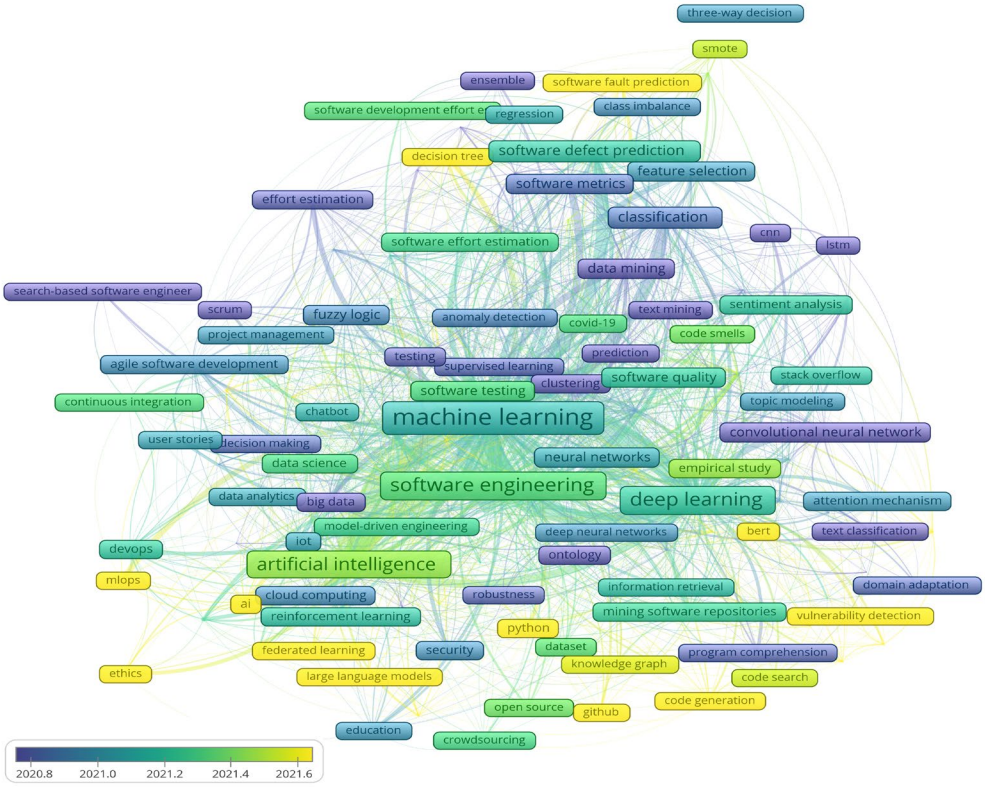

Figure 3. The research timeline landscape of AI use in software engineering.

3.4. Research Gaps and Challenges

Our analysis also revealed some research gaps and challenges that should be dealt with and require future research. At the present time, AI still needs to be more reliable, and software engineers and developers must thoroughly check its algorithms' output. Additionally, the increased use of AI in software engineering might result in more and more code, which will be less and less understood in terms of how it works, establishing a positive feedback loop, which will lead to more and more checking, which might result in more time being spent on checking then developing.

AI needs vast amounts of data to create its models. That is not a problem if software engineers can train AI algorithms using the data from public and open repositories—but if they work in unique domains, appropriate training sets might not be available.

As with AI use in general, much more research is needed to resolve the ethical concerns, even more so because software engineers are generally not trained and educated to deal with societal impacts and ethical issues.

The extensive use of AI in software engineering will require new specialist skills, which should be incorporated into software engineering curricula. Advanced tools, especially large language models, require sizeable computational power, storage space, and energy supply, which might increase costs. AI-based tools may require extra licensing fees.

3.5. Possible Future Research Trends

Possible research trends were identified by comparing the author keywords and the associated research literature from different epochs, as indicated by different colors in Figure 3 [94]. By comparison of the violet and blue epochs (the period before March 2021) with the green and yellow epochs (the period after March 2021), several feature directions were revealed:

- Development of transparent, fair, ethical, responsible, and sustainable intelligent software development processes [95] to eliminate management and for-profit bias, reduce human oversight, improve privacy, reduce cybercrime, make software more robust and accountable, and finally, reduce the digital divide.
- Self-adapting software that adapts to evolving user requirements [96], frequently changing due to fast ICT development, new user needs and requirements, and similar. In that manner, the software could become more robust, easier, and cheaper to maintain, and software development could be more sustainable.
- Self-healing and self-reflecting software that returns to a more functional condition after faults or performance and cybersecurity issues [97]. In that manner, the use of the software might become safer, more cost-effective, and more user-friendly.
- Collaborative software development eco-systems where AI partners with human developers take team dynamics and self-organization into account [98], enabling significant improvements in productivity, code quality, and developer empowerment.
- Adaptive continuous-learning platforms for software developers and engineers [85] to enable them to stay ahead of new technologies.
- New software engineering curricula [86] to teach future software engineers about the trends mentioned above.

3.6. Study Strengths and Limitations

Like similar studies, this study also has strengths and limitations. Its main strength is that it is the first quantitative and qualitative bibliometrics review of the use of AI in software engineering, encompassing the LLM and generative AI trends. As such, the visual and tabulated landscapes presented in the study show a multidimensional but holistic and most recent view of the use of AI in software development, which can support the software engineering community in addressing the theoretical and practical challenges of using and developing these new technologies. Additionally, the landscapes can serve as a guide for further research and a starting point for more formal knowledge synthesis endeavors such as systematic reviews and meta-analyses. However, this study also has

limitations. One of the limitations is that some of the most recent publications are published as preprints [87], are consequently not indexed in Scopus, and were thus not included in the analysis. Consequently, this study might not cover the latest research, especially regarding the use of LLM in software engineering [88,99]. The other limitation is that a part of the analysis is qualitative, which might introduce some bias; however, we believe that by strictly using triangulation, documented evidence collection, and the content analysis approach, we ensured that the results can be generalized to be used in real-world settings, and that they are credible and dependable.

4. Conclusions

In conclusion, our analysis revealed that research on AI has significantly impacted software development in recent years. From natural language processing in software engineering, the use of artificial intelligence in the management of the software development life cycle, the use of machine learning in fault/defect prediction and effort estimation, the employment of deep learning in intelligent software engineering and code management, to mining software repositories to improve software quality, AI has changed the way developers and engineers build software. The convergence of AI and software engineering has the potential to significantly reduce the required resources, improve the quality, and enhance the user experience with more intelligent user-centric applications. On the other hand, it may enhance the well-being of software developers and engineers with the automation of repetitive tasks, reduced workload, improved and more reliable/accurate predictive analysis, and speeded up development cycle. Finally, it can also help software managers monitor the overall team status, state, and performance, providing them with notifications if a team member needs to be more utilized, is over-extended, or is heading toward burnout.

Although we believe that we have analyzed the vast majority of peer-reviewed publications, due to the extremely rapid progress of the use of AI in software engineering, many of the latest findings are published in non-peer-reviewed media such as preprints. Therefore, in further work, we will develop tools so that we can automatically include such publications in the corpus and obtain an even more comprehensive synthesis of the state-of-the-art research. We also intend to periodically repeat the syntheses so that the most modern findings and trends will be readily available.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The author declare no conflicts of interest.

References

1. Ooi, K.-B.; Tan, G.W.-H.; Al-Emran, M.; Al-Sharafi, M.A.; Capatina, A.; Chakraborty, A.; Dwivedi, Y.K.; Huang, T.-L.; Kar, A.K.; Lee, V.-H.; et al. The Potential of Generative Artificial Intelligence Across Disciplines: Perspectives and Future Directions. *J. Comput. Inf. Syst.* **2023**, 1–32. [CrossRef]
2. Završnik, J.; Kokol, P.; Žlahtič, B.; Blažun Vošner, H. Artificial Intelligence and Pediatrics: Synthetic Knowledge Synthesis. *Electronics* **2024**, *13*, 512. [CrossRef]
3. Lo, D. Trustworthy and Synergistic Artificial Intelligence for Software Engineering: Vision and Roadmaps. *arXiv* **2023**, arXiv:2309.04142.
4. Belzner, L.; Gabor, T.; Wirsing, M. Large Language Model Assisted Software Engineering: Prospects, Challenges, and a Case Study. In *Proceedings of the Bridging the Gap Between AI and Reality*; Steffen, B., Ed.; Springer Nature: Cham, Switzerland, 2024; pp. 355–374.
5. Batarseh, F.A.; Mohod, R.; Kumar, A.; Bui, J. The Application of Artificial Intelligence in Software Engineering: A Review Challenging Conventional Wisdom. In *Data Democracy*; Batarseh, F.A., Yang, R., Eds.; Academic Press: Cambridge, MA, USA, 2020; pp. 179–232, ISBN 978-0-12-818366-3.

6. Sofian, H.; Yunus, N.A.M.; Ahmad, R. Systematic Mapping: Artificial Intelligence Techniques in Software Engineering. *IEEE Access* **2022**, *10*, 51021–51040. [CrossRef]
7. Amalfitano, D.; Faralli, S.; Hauck, J.C.R.; Matalonga, S.; Distante, D. Artificial Intelligence Applied to Software Testing: A Tertiary Study. *ACM Comput. Surv.* **2023**, *56*, 3616372. [CrossRef]
8. Majumdar, S.; Paul, S.; Paul, D.; Bandyopadhyay, A.; Chattopadhyay, S.; Das, P.P.; Clough, P.D.; Majumder, P. Generative AI for Software Metadata: Overview of the Information Retrieval in Software Engineering Track at FIRE 2023. Available online: https://api.semanticscholar.org/CorpusID:265043660 (accessed on 5 May 2024).
9. Kulkarni, V.; Kolhe, A.; Kulkarni, J. Intelligent Software Engineering: The Significance of Artificial Intelligence Techniques in Enhancing Software Development Lifecycle Processes. In *Proceedings of the Intelligent Systems Design and Applications*; Abraham, A., Gandhi, N., Hanne, T., Hong, T.-P., Nogueira Rios, T., Ding, W., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 67–82.
10. Satpute, R.S.; Agrawal, A. A Critical Study of Pragmatic Ambiguity Detection in Natural Language Requirements. *Int. J. Intell. Syst. Appl. Eng.* **2023**, *11*, 249–259.
11. Daun, M.; Brings, J. How ChatGPT Will Change Software Engineering Education. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education*; Association for Computing Machinery: New York, NY, USA, 2023; Volume 1, pp. 110–116.
12. Cao, S.; Sun, X.; Widyasari, R.; Lo, D.; Wu, X.; Bo, L.; Zhang, J.; Li, B.; Liu, W.; Wu, D.; et al. A Systematic Literature Review on Explainability for Machine/Deep Learning-Based Software Engineering Research. *arXiv* **2024**, arXiv:2401.14617.
13. Mohammadkhani, A.H.; Bommi, N.S.; Daboussi, M.; Sabnis, O.; Tantithamthavorn, C.; Hemmati, H. A Systematic Literature Review of Explainable AI for Software Engineering. *arXiv* **2023**, arXiv:2302.06065.
14. Ozkaya, I. The Next Frontier in Software Development: AI-Augmented Software Development Processes. *IEEE Softw.* **2023**, *40*, 4–9. [CrossRef]
15. Bano, M.; Hoda, R.; Zowghi, D.; Treude, C. Large Language Models for Qualitative Research in Software Engineering: Exploring Opportunities and Challenges. *Autom. Softw. Eng.* **2023**, *31*, 8. [CrossRef]
16. Kokol, P. Synthetic Knowledge Synthesis in Hospital Libraries. *J. Hosp. Librariansh.* **2023**, *24*, 1–8. [CrossRef]
17. Železnik, U.; Kokol, P.; Starc, J.; Železnik, D.; Završnik, J.; Vošner, H.B. Research Trends in Motivation and Weight Loss: A Bibliometric-Based Review. *Healthcare* **2023**, *11*, 3086. [CrossRef]
18. Markoulli, M.P.; Lee, C.I.S.G.; Byington, E.; Felps, W.A. Mapping Human Resource Management: Reviewing the Field and Charting Future Directions. *Hum. Resour. Manag. Rev.* **2017**, *27*, 367–396. [CrossRef]
19. Moon, M.D. Triangulation: A Method to Increase Validity, Reliability, and Legitimation in Clinical Research. *J. Emerg. Nurs.* **2019**, *45*, 103–105. [CrossRef]
20. Farooq, U.; Nasir, A.; Khan, K.I. An Assessment of the Quality of the Search Strategy: A Case of Bibliometric Studies Published in Business and Economics. *Scientometrics* **2023**, *128*, 4855–4874. [CrossRef]
21. G20. Wikipedia 2024. Available online: https://en.wikipedia.org/wiki/G20 (accessed on 5 May 2024).
22. Kokol, P. Discrepancies among Scopus and Web of Science, Coverage of Funding Information in Medical Journal Articles: A Follow-up Study. *J. Med. Libr. Assoc.* **2023**, *111*, 703–709. [CrossRef]
23. Vakkuri, V.; Kemell, K.-K.; Kultanen, J.; Abrahamsson, P. The Current State of Industrial Practice in Artificial Intelligence Ethics. *IEEE Softw.* **2020**, *37*, 50–57. [CrossRef]
24. Pasricha, S.; Wolf, M. Ethical Design of Computers: From Semiconductors to IoT and Artificial Intelligence. *IEEE Des. Test* **2024**, *41*, 7–16. [CrossRef]
25. Barletta, V.S.; Caivano, D.; Gigante, D.; Ragone, A. A Rapid Review of Responsible AI Frameworks: How to Guide the Development of Ethical AI. In Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering, Oulu, Finland, 14–16 June 2023; pp. 358–367.
26. Nazim, M.; Wali Mohammad, C.; Sadiq, M. A Comparison between Fuzzy AHP and Fuzzy TOPSIS Methods to Software Requirements Selection. *Alex. Eng. J.* **2022**, *61*, 10851–10870. [CrossRef]
27. Ali, S.; Ullah, N.; Abrar, M.F.; Yang, Z.; Huang, J.; Ali, R. Fuzzy Multicriteria Decision-Making Approach for Measuring the Possibility of Cloud Adoption for Software Testing. *Sci. Program.* **2020**, *2020*, 6597316. [CrossRef]
28. Le, T.-A.; Huynh, Q.-T.; Nguyen, T.-T.-N.; Thi, M.-H.T. A New Method for Enhancing Software Effort Estimation by Using ANFIS-Based Approach. In *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*; Springer: Cham, Switzerland, 2021; Volume 379, pp. 195–210. [CrossRef]
29. Rathi, N.; Srivathsav, R.; Chitlangia, R.; Pachghare, V.K. Automatic Selenium Code Generation for Testing. *Adv. Intell. Syst. Comput.* **2020**, *1039*, 194–200. [CrossRef]
30. Hooda, S.; Sood, V.M.; Singh, Y.; Dalal, S.; Sood, M. *Agile Software Development: Trends, Challenges and Applications*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2023; p. 365, ISBN 978-1-119-89683-8.
31. Fehlmann, T.; Kranich, E. ART for Agile: Autonomous Real-Time Testing in the Product Development Cycle. *Commun. Comput. Inf. Sci.* **2021**, *1442*, 377–390. [CrossRef]
32. Shankar, S.P.; Chaudhari, S.S. Framework for the Automation of SDLC Phases Using Artificial Intelligence and Machine Learning Techniques. *Int. J. Recent Innov. Trends Comput. Commun.* **2023**, *11*, 379–390. [CrossRef]

33. Kataev, M.; Bulysheva, L.; Xu, L.; Ekhlakov, Y.; Permyakova, N.; Jovanovic, V. Fuzzy Model Estimation of the Risk Factors Impact on the Target of Promotion of the Software Product. *Enterp. Inf. Syst.* **2020**, *14*, 797–811. [CrossRef]
34. Klus, H.; Knieke, C.; Rausch, A.; Wittek, S. Software Engineering Meets Artificial Intelligence. *Electron. Commun. EASST* **2023**, *82*, 1–16. [CrossRef]
35. Ahmed, H.A.; Bawany, N.Z.; Shamsi, J.A. Capbug-a Framework for Automatic Bug Categorization and Prioritization Using Nlp and Machine Learning Algorithms. *IEEE Access* **2021**, *9*, 50496–50512. [CrossRef]
36. Sawant, A.A.; Devanbu, P. Naturally!: How Breakthroughs in Natural Language Processing Can Dramatically Help Developers. *IEEE Softw.* **2021**, *38*, 118–123. [CrossRef]
37. Charitsis, C.; Piech, C.; Mitchell, J.C. Using NLP to Quantify Program Decomposition in CS1. In Proceedings of the Proceedings of the Ninth ACM Conference on Learning, New York, NY, USA, 1–3 June 2022; pp. 113–120.
38. dos Santos, G.E.; Figueiredo, E. Commit Classification Using Natural Language Processing: Experiments over Labeled Datasets. 2020. Available online: https://cibse2020.ppgia.pucpr.br/images/artigos/4/S04_P1.pdf (accessed on 5 May 2024).
39. Wong, M.-F.; Guo, S.; Hang, C.-N.; Ho, S.-W.; Tan, C.-W. Natural Language Generation and Understanding of Big Code for AI-Assisted Programming: A Review. *Entropy* **2023**, *25*, 888. [CrossRef]
40. Koreeda, Y.; Morishita, T.; Imaichi, O.; Sogawa, Y. LARCH: Large Language Model-Based Automatic Readme Creation with Heuristics. In Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, Birmingham, UK, 21–25 October 2023; pp. 5066–5070.
41. Di Sipio, C.; Di Rocco, J.; Di Ruscio, D.; Nguyen, P.T. MORGAN: A Modeling Recommender System Based on Graph Kernel. *Softw. Syst. Model.* **2023**, *22*, 1427–1449. [CrossRef]
42. Jánki, Z.R.; Bilicki, V. The Impact of the Web Data Access Object (WebDAO) Design Pattern on Productivity. *Computers* **2023**, *12*, 149. [CrossRef]
43. Pauzi, Z.; Capiluppi, A. Applications of Natural Language Processing in Software Traceability: A Systematic Mapping Study. *J. Syst. Softw.* **2023**, *198*, 111616. [CrossRef]
44. Zhao, L.; Alhoshan, W.; Ferrari, A.; Letsholo, K.J.; Ajagbe, M.A.; Chioasca, E.-V.; Batista-Navarro, R.T. Natural Language Processing for Requirements Engineering. *ACM Comput. Surv.* **2021**, *54*, 1–41. [CrossRef]
45. Umar, M.A.; Lano, K. Advances in Automated Support for Requirements Engineering: A Systematic Literature Review. *Requir. Eng.* **2024**, *29*, 177–207. [CrossRef]
46. Arulmohan, S.; Meurs, M.-J.; Mosser, S. Extracting Domain Models from Textual Requirements in the Era of Large Language Models. In Proceedings of the 2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), Västerås, Sweden, 1–6 October 2023; pp. 580–587.
47. Vahabi, S.; Hozhabri, A. Automatic Use Case Classification Based on Topic Grouping for Requirements Engineering. *Innov. Syst. Softw. Eng.* **2024**, *20*, 85–96. [CrossRef]
48. Kochbati, T.; Li, S.; Gérard, S.; Mraidha, C. From User Stories to Models: A Machine Learning Empowered Automation. In Proceedings of the MODELSWARD 2022-9th International Conference on Model-Driven Engineering and Software Development, Virtual Event, 8–10 February 2021; pp. 28–40.
49. Gunes, T.; Aydemir, F.B. Automated Goal Model Extraction from User Stories Using NLP. In Proceedings of the 2020 IEEE 28th International Requirements Engineering Conference, Zurich, Switzerland, 31 August–4 September 2020; pp. 382–387.
50. Wang, Y.; Hu, K.; Jiang, B.; Xia, X.; Tang, X.-S. A Systematic Literature Review of Software Traceability Links Automation Techniques. *Jisuanji Xuebao/Chin. J. Comput.* **2023**, *46*, 1919–1946.
51. Hey, T.; Keim, J.; Koziolek, A.; Tichy, W.F. NoRBERT: Transfer Learning for Requirements Classification. In Proceedings of the 2020 IEEE 28th International Requirements Engineering Conference, Zurich, Switzerland, 31 August–4 September 2020; pp. 169–179.
52. Alhoshan, W.; Ferrari, A.; Zhao, L. Zero-Shot Learning for Requirements Classification: An Exploratory Study. *Inf. Softw. Technol.* **2023**, *159*, 107202. [CrossRef]
53. Ezzini, S.; Abualhaija, S.; Arora, C.; Sabetzadeh, M.; Briand, L.C. Using Domain-Specific Corpora for Improved Handling of Ambiguity in Requirements. In Proceedings of the 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), Madrid, Spain, 22–30 May 2021; pp. 1485–1497.
54. Sarmiento-Calisaya, E.; do Prado Leite, J.C.S. Early Analysis of Requirements Using NLP and Petri-Nets. *J. Syst. Softw.* **2024**, *208*, 111901. [CrossRef]
55. Shen, Y.; Breaux, T. Stakeholder Preference Extraction from Scenarios. *IEEE Trans. Softw. Eng.* **2024**, *50*, 69–84. [CrossRef]
56. García, S.E.M.; Fernández-y-Fernández, C.A.; Pérez, E.G.R. Classification of Non-Functional Requirements Using Convolutional Neural Networks. *Program. Comput. Softw.* **2023**, *49*, 705–711. [CrossRef]
57. Tikayat Ray, A.; Cole, B.F.; Pinon Fischer, O.J.; Bhat, A.P.; White, R.T.; Mavris, D.N. Agile Methodology for the Standardization of Engineering Requirements Using Large Language Models. *Systems* **2023**, *11*, 352. [CrossRef]
58. Calle Gallego, J.M.; Zapata Jaramillo, C.M. QUARE: Towards a Question-Answering Model for Requirements Elicitation. *Autom. Softw. Eng.* **2023**, *30*, 25. [CrossRef]
59. De Carvalho, H.D.P.; Fagundes, R.; Santos, W. Extreme Learning Machine Applied to Software Development Effort Estimation. *IEEE Access* **2021**, *9*, 92676–92687. [CrossRef]
60. Rodríguez Sánchez, E.; Vázquez Santacruz, E.F.; Cervantes Maceda, H. Effort and Cost Estimation Using Decision Tree Techniques and Story Points in Agile Software Development. *Mathematics* **2023**, *11*, 1477. [CrossRef]

61. Heng, S.; Snoeck, M.; Tsilionis, K. Building a Software Architecture out of User Stories and BDD Scenarios: Research Agenda. In Proceedings of the CEUR Workshop Proceedings, Ljubljana, Slovenia, 29 November 2022; Volume 3134, pp. 40–46.
62. Mcmurray, S.; Sodhro, A.H. A Study on ML-Based Software Defect Detection for Security Traceability in Smart Healthcare Applications. *Sensors* **2023**, *23*, 3470. [CrossRef] [PubMed]
63. Jadhav, A.; Kaur, M.; Akter, F. Evolution of Software Development Effort and Cost Estimation Techniques: Five Decades Study Using Automated Text Mining Approach. *Math. Probl. Eng.* **2022**, *2022*, 5782587. [CrossRef]
64. Priya Varshini, A.G.; Anitha Kumari, K.; Varadarajan, V. Estimating Software Development Efforts Using a Random Forest-Based Stacked Ensemble Approach. *Electronics* **2021**, *10*, 1195. [CrossRef]
65. Singal, P.; Kumari, A.C.; Sharma, P. Estimation of Software Development Effort: A Differential Evolution Approach. *Procedia Comput. Sci.* **2020**, *167*, 2643–2652.
66. Khan, F.; Lingala, G. Machine Learning Techniques For Software Component Reusability. In Proceedings of the 2022 3rd International Conference for Emerging Technology (INCET), Belgaum, India, 27–29 May 2022.
67. Gupta, N.; Sinha, R.R.; Goyal, A.; Sunda, N.; Sharma, D. Analyze the Performance of Software by Machine Learning Methods for Fault Prediction Techniques. *Int. J. Recent Innov. Trends Comput. Commun.* **2023**, *11*, 178–187. [CrossRef]
68. Nasser, A.B.; Ghanem, W.; Abdul-Qawy, A.S.H.; Ali, M.A.H.; Saad, A.-M.; Ghaleb, S.A.A.; Alduais, N. A Robust Tuned K-Nearest Neighbours Classifier for Software Defect Prediction. *Lect. Notes Netw. Syst.* **2023**, *573*, 181–193. [CrossRef]
69. Khan, M.A.; Elmitwally, N.S.; Abbas, S.; Aftab, S.; Ahmad, M.; Fayaz, M.; Khan, F. Software Defect Prediction Using Artificial Neural Networks: A Systematic Literature Review. *Sci. Program.* **2022**, *2022*, 2117339. [CrossRef]
70. Hilmi, M.A.A.; Puspaningrum, A.; Darsih; Siahaan, D.O.; Samosir, H.S.; Rahma, A.S. Research Trends, Detection Methods, Practices, and Challenges in Code Smell: SLR. *IEEE Access* **2023**, *11*, 129536–129551. [CrossRef]
71. Soomlek, C.; van Rijn, J.N.; Bonsangue, M.M. *Automatic Human-Like Detection of Code Smells*; Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer International Publishing: Cham, Switzerland, 2021; Volume 12986. [CrossRef]
72. Mock, M. *Utilization of Machine Learning for the Detection of Self-Admitted Vulnerabilities. Lecture Notes in Computer Science*; Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics; Springer International Publishing: Cham, Switzerland, 2024; Volume 14484. [CrossRef]
73. Pontillo, V.; Palomba, F.; Ferrucci, F. Toward Static Test Flakiness Prediction: A Feasibility Study. In Proceedings of the 5th International Workshop on Machine Learning Techniques for Software Quality Evolution, Athens, Greece, 23 August 2021; pp. 19–24.
74. Albuquerque, D.; Guimaraes, E.; Tonin, G.; Rodriguez, P.; Perkusich, M.; Almeida, H.; Perkusich, A.; Chagas, F. Managing Technical Debt Using Intelligent Techniques—A Systematic Mapping Study. *IEEE Trans. Softw. Eng.* **2023**, *49*, 2202–2220. [CrossRef]
75. Hu, X.; Li, G.; Xia, X.; Lo, D.; Jin, Z. Deep Code Comment Generation with Hybrid Lexical and Syntactical Information. *Empir. Softw. Eng.* **2020**, *25*, 2179–2217. [CrossRef]
76. Li, B.; Yan, M.; Xia, X.; Hu, X.; Li, G.; Lo, D. Deep Commenter: A Deep Code Comment Generation Tool with Hybrid Lexical and Syntactical Information. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual, 8–13 November 2020; pp. 1571–1575.
77. Georgiou, K.; Charmanas, K.; Papageorgiadis, K.; Mittas, N.; Christidis, G.; Angelis, L. A Data-Driven Framework for Knowledge Exchange Analysis of Development Issues in Medical Applications: A Case Study of COVID-19. In Proceedings of the 2023 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Durres, Albania, 6–8 September 2023; pp. 386–393.
78. Sabor, K.K.; Hamdaqa, M.; Hamou-Lhadj, A. Automatic Prediction of the Severity of Bugs Using Stack Traces and Categorical Features. *Inf. Softw. Technol.* **2020**, *123*, 106205. [CrossRef]
79. Li, M.; Yu, H.; Fan, G.; Zhou, Z.; Huang, J. ClassSum: A Deep Learning Model for Class-Level Code Summarization. *Neural Comput. Appl.* **2023**, *35*, 3373–3393. [CrossRef]
80. Jain, R.; Gervasoni, N.; Ndhlovu, M.; Rawat, S. A Code Centric Evaluation of C/C++ Vulnerability Datasets for Deep Learning Based Vulnerability Detection Techniques. In Proceedings of the 16th Innovations in Software Engineering Conference, Allahabad, India, 23–25 February 2023.
81. Iannone, E.; Guadagni, R.; Ferrucci, F.; De Lucia, A.; Palomba, F. The Secret Life of Software Vulnerabilities: A Large-Scale Empirical Study. *IEEE Trans. Softw. Eng.* **2023**, *49*, 44–63. [CrossRef]
82. Zhu, K.; Yin, M.; Li, Y. Detecting and Classifying Self-Admitted of Technical Debt with CNN-BiLSTM. *J. Phys. Conf. Ser.* **2021**, *1955*, 012102. [CrossRef]
83. Wang, X.; Liu, J.; Li, L.; Chen, X.; Liu, X.; Wu, H. Detecting and Explaining Self-Admitted Technical Debts with Attention-Based Neural Networks. In Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering, Virtual, 21–25 December 2020; pp. 871–882.
84. Tsoukalas, D.; Kehagias, D.; Siavvas, M.; Chatzigeorgiou, A. Technical Debt Forecasting: An Empirical Study on Open-Source Repositories. *J. Syst. Softw.* **2020**, *170*, 110777. [CrossRef]
85. Scott, E.; Campo, M. An Adaptive 3D Virtual Learning Environment for Training Software Developers in Scrum. *Interact. Learn. Environ.* **2023**, *31*, 5200–5218. [CrossRef]

86. Kirova, V.D.; Ku, C.S.; Laracy, J.R.; Marlowe, T.J. Software Engineering Education Must Adapt and Evolve for an LLM Environment. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education*; Association for Computing Machinery: New York, NY, USA, 2024; Volume 1, pp. 666–672.
87. Nguyen-Duc, A.; Cabrero-Daniel, B.; Przybylek, A.; Arora, C.; Khanna, D.; Herda, T.; Rafiq, U.; Melegati, J.; Guerra, E.; Kemell, K.-K.; et al. Generative Artificial Intelligence for Software Engineering—A Research Agenda. *arXiv* **2023**, arXiv:2310.18648.
88. Alshahwan, N.; Harman, M.; Harper, I.; Marginean, A.; Sengupta, S.; Wang, E. Assured LLM-Based Software Engineering. *arXiv* **2024**, arXiv:2402.04380.
89. Shcherban, S.; Liang, P.; Tahir, A.; Li, X. Automatic Identification of Code Smell Discussions on Stack Overflow: A Preliminary Investigation. In Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Bari, Italy, 5–9 October 2020.
90. Awan, W.N.; Paasivaara, M.; Gloor, P.; Salman, I. Creating Happier and More Productive Software Engineering Teams through AI and Machine Learning. 2024, Volume 3621. Available online: https://ceur-ws.org/Vol-3621/phd-paper1.pdf (accessed on 5 May 2024).
91. Li, W.; Wu, F.; Fu, C.; Zhou, F. A Large-Scale Empirical Study on Semantic Versioning in Golang Ecosystem. In Proceedings of the 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE), Luxembourg, 11–15 September 2023; pp. 1604–1614.
92. Izadi, M.; Akbari, K.; Heydarnoori, A. Predicting the Objective and Priority of Issue Reports in Software Repositories. *Empir. Softw. Eng.* **2022**, *27*, 50. [CrossRef]
93. He, H.; He, R.; Gu, H.; Zhou, M. A Large-Scale Empirical Study on Java Library Migrations: Prevalence, Trends, and Rationales. In Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, 23–28 August 2021; pp. 478–490.
94. Kokol, P.; Završnik, J.; Vošner, H.B. Bibliographic-Based Identification of Hot Future Research Topics: An Opportunity for Hospital Librarianship. *J. Hosp. Librariansh.* **2018**, *18*, 315–322. [CrossRef]
95. Akbar, M.A.; Khan, A.A.; Liang, P. Ethical Aspects of ChatGPT in Software Engineering Research. *IEEE Trans. Artif. Intell.* **2023**, 1–14. [CrossRef]
96. Menezes, T. A Review to Find Elicitation Methods for Business Process Automation Software. *Software* **2023**, *2*, 177–196. [CrossRef]
97. Self-Healing Code Is the Future of Software Development—Stack Overflow. Available online: https://stackoverflow.blog/2023/12/28/self-healing-code-is-the-future-of-software-development/ (accessed on 3 May 2024).
98. Ahmad, A.; Waseem, M.; Liang, P.; Fahmideh, M.; Aktar, M.S.; Mikkonen, T. Towards Human-Bot Collaborative Software Architecting with ChatGPT. In *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*; Association for Computing Machinery: New York, NY, USA, 2023; pp. 279–285.
99. Khojah, R.; Mohamad, M.; Leitner, P.; Neto, F.G.dO. Beyond Code Generation: An Observational Study of ChatGPT Usage in Software Engineering Practice. *arXiv* **2024**, arXiv:2404.14901.