The Trailer of the ACM 2030 Roadmap for Software Engineering

Mauro Pezzè
USI Università della Svizzera Italiana, Lugano, Switzerland
Constructor Institute of Technology, Schaffhausen, Switzerland
Università degli studi di Milano Bicocca, Milano, Italy
mauro.pezze@usi.ch

Matteo Ciniselli
USI Università della Svizzera Italiana
Lugano, Switzerland
matteo.ciniselli@usi.ch

Luca Di Grazia
USI Università della Svizzera Italiana
Lugano, Switzerland
luca.di.grazia@usi.ch

Niccolò Puccinelli
USI Università della Svizzera Italiana
Lugano, Switzerland
niccolo.puccinelli@usi.ch

Ketai Qiu
USI Università della Svizzera Italiana
Lugano, Switzerland
ketai.qiu@usi.ch

DOI: 10.1145/ 3696117.3696126

ABSTRACT http://doi.org/10.1145/3696117.3696126
The landscape of software engineering has dramatically changed. The recent advances in AI, the new opportunities of quantum computing, and the new challenges of sustainability and cyber security upset the software engineering research prospective. The 2030 SE-Roadmap special issue of ACM TOSEM Transactions on Software Engineering and Methodology gives a 360° view of the research challenges of the 30ties with a thorough editorial, four roadmap papers from the ACM TOSEM editorial board, and over 30 peer-reviewed papers from the research community.

This paper previews the main content of the 2030 roadmap special issue with a report from the 2030 Software Engineering Roadmap workshop, co-located with ACM SIGSOFT FSE Foundations of Software engineering on July 15th and 16th, 2024 in Porto de Galinhas, Brazil, that spotlighted the software engineering research horizon to feed ideas into the ACM TOSEM special issue. The paper discusses the new challenges to software engineering that emerged in the SE2030 workshop: AI for software engineering, software engineering by and for humans, sustainable software engineering, automatic programming, cyber security, validation and verification, and quantum software engineering.

1. INTRODUCTION

The landscape of software engineering has dramatically changed, and we need a new roadmap for the research in software engineering. In this paper we overview the results of the 2030 Software Engineering Roadmap workshop that was co-located with ACM SIGSOFT FSE Foundations of Software engineering on July 15th and 16th, 2024 in Porto de Galinhas, Brazil. The liberating structure[1] format of the intensive two-full-days discussion among the over sixty participants offers an unbiased viewpoint of the main challenges to software engineering in this decade. We invited the authors of 59 of the 76 papers submitted to the workshop[2] to discuss the recent changes in software engineering, share a vision of the future evolution of the discipline, and shape a roadmap for the research community. We invited the authors of selected papers to extend their paper for the SE2030 roadmap ACM TOSEM special issue, by taking into account the comments of the early reviews, and the outcome of the workshop that we summarize in this paper. The discussion identified seven key areas that challenge the research in software engineering:

Artificial Intelligence for Software Engineering (AI4SE) The recent breakthrough in machine learning, generative AI and autonomous systems triggers the deepest change in the skyline of software engineering research and practice since the Internet revolution in the second half of the last century [34] [4]. The software engineering community has never seen a so fast and predominant growth of new research threads such as the study of machine learning in software engineering and the challenges of engineering machine-learning-driven systems, both of which have become the dominant themes of the main software engineering conferences and journals [19]. The discussion in the workshop highlighted the challenges of AI tools to engineer software systems, the core challenges of predicting defects and trusting AI, and the deep need of reliable and explainable AI in the development workflow.

Software Engineering by and for Humans Machine learning, Generative AI and autonomous systems shape a new landscape for software engineering by and for humans, and radically change even the basic concept of software artifact [9]. The new software systems challenge software engineers with new ethical, technical and fairness problems [12]. Humans become integral part of large software ecosystems, and the research in software engineering shall move beyond the shortsighted vision of users of software systems, toward a new vision of humans as integral part of cyber-physical ecosystems [35]. The discussion in the workshop has focused on how LLMs bridge technical and human aspects, the ethical implications of AI, and the need for continuous human oversight in software development.

Sustainable Software Engineering The concept of sustainable development extends beyond classic environmental concerns, and spreads over software systems in cyber-physical spaces. Sustainable software operations in cyber-physical spaces require new design, development, deployment, and maintenance approaches that minimize the ecological footprint,

[1] An excellent introduction to the liberating structure is available at `https://www.liberatingstructures.com`
[2] The list of workshop papers as well as the url of many of the papers is available at the workshop website `https://conf.researchr.org/home/2030-se`. Workshop pictures and latest updates are available at `https://www.inf.usi.ch/faculty/pezze/se2030.html`

enhance resource efficiency, and promote social responsibility [27]. The discussion in the workshop highlighted the complex intertwine of societal, environmental, and economic perspectives, the need for regulations and software sustainability measures, and the energy inefficiency of AI models.

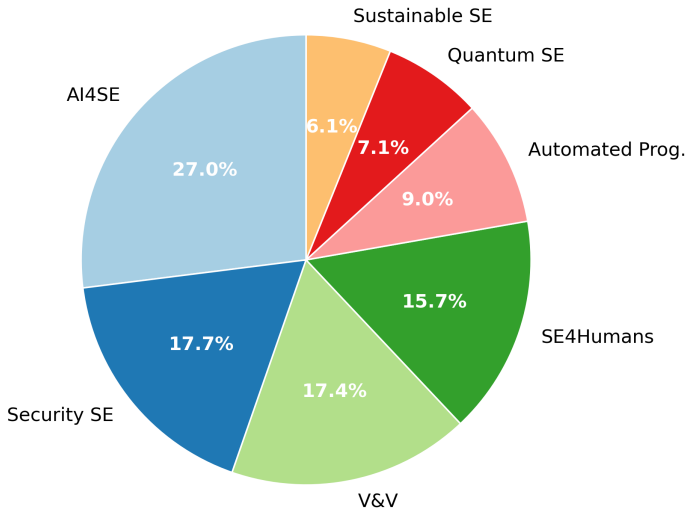
Automatic Programming Machine learning, and specifically deep neural networks and Large Language Models (LLMs), are the largest magnification factor of human productivity ever seen in software engineering since the pioneer times. They open new frontiers towards automatic programming, upset the quality and security scenario, and raise new societal and legal issues. The discussion in the workshop focused on the role of AI in generating code and upgrading legacy system [2], and in the challenges of context and prompt engineering.

Software Security Analysis The extensive usage of machine learning in software quality and security introduces new societal and legal considerations [1] [25]. As software systems grow in complexity and scale, new security challenges emerge, emphasizing the need for innovative approaches to secure software engineering and cybersecurity [16] [28]. The discussion in the workshop highlighted the importance of security-aware tools for vulnerability detection and secure code generation, and the role of deep learning in anomaly detection and secure software supply chains.

Validation and Verification Machine learning and AI upset the validation and verification scenario. On one side, the adaptive and evolving nature of AI-driven systems changes the concept itself of test input and oracle, on the other side machine nearing and GAI open new frontiers to test automation. We need both a new conceptual approach to frame the problem of testing and analyzing AI-driven software systems and new studies on how exploit ML and GAI in software testing and analysis. The discussion in the workshop focused on the recent progress in knowledge compilation and meta-solvers that enhanced the scalability of various software engineering tools, and on the impact on test-case generation and configurable software analysis [21].

Quantum Software Engineering Quantum computing opens a new class of software systems that share very little if any at all with classic software systems. Quantum software engineering is still in its infancy and can greatly benefit from the open-minded studies of experienced software engineers towards new approaches to engineer quantum software systems. The discussion in the workshop focused on the challenges in quantum programming, including the lack of established knowledge, new design and testing methodologies, integration with classical systems, and the need for specialized education.

These research directions have significant implications for both academia and industry. Long term academic research shall develop new theories, methodologies, and educational curricula, to improve a deeper understanding of emerging technologies and their impacts. Industrial technology transfer shall use research results to design and implement innovative tools and practices that can improve software development, software quality and security, and address sustainability concerns. The following pie chart visualizes the trends of the discussion in the workshop, by showing the word count from the notes taken during the 2030 Software Engineering Roadmap workshop:

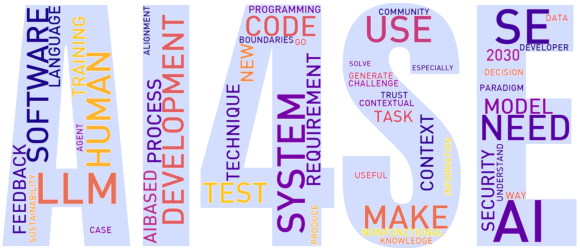


Sections 2–8 briefly overview the discussion in the seven areas. Each section starts with a word cloud of the most frequent terms from the notes of the 2030 Software Engineering Roadmap workshop, presents an overview of the main results, two lists of *open research questions* and *future research directions*, respectively, and conclude with an

Area summary.

Section 9 concludes with a summary of the main results of the 2030 Software Engineering Roadmap workshop.

2. AI4SE



AI tools, like Copilot for code generation, are commonly integrated in industrial software development workflows [20]. The key open issues are (i) incorporating bug and vulnerability detection in the workflow, to enhance software quality and security, and (ii) guaranteeing trust in AI and explainability, especially in dependable or safety-critical systems. There is a need for AI development processes that ensure the reliability and understandability of AI-generated artifacts. We shall bridge software engineering and AI to seamlessly integrate requirements in the development of AI driven systems.

Open Research Issues

Explainability and Trust Ensuring that AI-generated outputs are explainable and trustworthy, particularly in safety-critical systems.

Quality Assurance Establishing quality measures for AI auto-coding to ensure that AI-generated code meets high standards.

Defect and Vulnerability Prediction Incorporating AI-based bug detection into the development workflow to enhance software quality and security.

Feedback Loops and Knowledge Integration Implementing feedback loops, using knowledge graphs, and enhancing more determinism in AI systems.

Testing and Security Developing hybrid testing approaches using AI and fuzzy techniques, and enforcing security guidelines in LLMs [17].

Domain Adaptation Applying deep learning and LLMs to new domains like quantum software engineering where data or well-defined knowledge may be sparse.

Human-AI Collaboration Enhancing AI training processes with human feedback, labeling, and crowdsourcing to improve model performance.

Future Research Directions

Bridging AI and SE Developing methodologies that integrate AI techniques into SE practices, focusing on sustainability, transparency, security, trustworthiness, and compliance [11].

Contextual Learning Incorporating context into AI training, as human learning heavily depends on contextual information.

Evaluating Costs Assessing the cost of training, fine-tuning, and using LLMs from different perspectives to determine the advantages of AI over other alternatives.

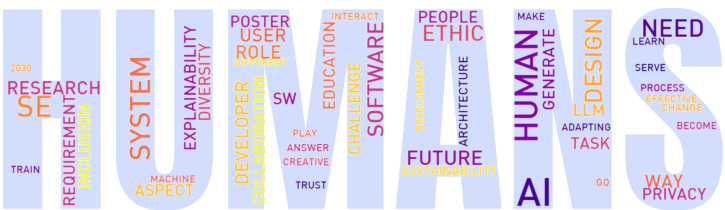
New Domains and Applications Exploring the use of AI in new domains such as quantum software engineering and low-resource environments.

Collaboration Between Academia and Industry Redefining the role of software engineers and adapting to the evolving landscape of AI and SE.

By addressing these open research issues and exploring future directions, the integration of AI in software engineering can be enhanced, leading to increasingly reliable, secure, and efficient software development processes.

AI4SE Future research should (i) develop robust quality measures for AI-generated code and integrate AI carefully into software development workflows and (ii) improve AI training with better feedback loops and multi-model approaches to address non-functional requirements.

3. SOFTWARE ENGINEERING BY AND FOR HUMANS



A core challenge that emerged in the 2030 Software Engineering Roadmap workshop is the disconnection between software developers and users. Software engineers still build software for users according to some specifications. Popular software engineering approaches, like scrum, attempt to seamlessly include some users' needs in the development process with the role of product owner, however, there is still a large gap between users and software engineers. Large Language Models (LLMs) can bridge technical and human aspects [9]. Requirements engineering will play a significant role in instructing the machine to correctly capture the needs of the many individual users [30], with most activities being automated. AI can accelerate innovation, by radically changing the way humans interact with software systems. However, the role of humans in 2030 remains unclear. We need to understand which tasks will be performed by humans and which by AI. AI will dramatically change the way developers interact with systems. Keeping humans in the loop is essential to ensure meaningful engagement and development [34].

Open Research Issues

Ethical Rules and Regulations Defining and enforcing ethical rules and regulations, far beyond the AI Act and specific laws that have been recently approved.

Continuous Learning Mastering AI systems that continuously learn, thus impacting the way we generate code and potentially on human's creativity.

Explainability and User Adaptation LLM interfaces to let humans understand the decisions made in the background, and improve explainability. There is a concern that humans are adapting to systems rather than systems to humans, thus leading to a scenario where humans become passive entities.

Human-Centric Design Defining software development processes for humans, and balancing work-life and ethical issues [24].

Privacy and Regulations Rethinking both privacy by design and regulations on sustainability and AI in the software development process.

Diversity and Inclusion (DEI) Mastering the big AI systems' diversity and inclusion challenges [18], by both improving funding for DEI initiatives and addressing uncertainties about the impact on DEI research in academia.

Trust and Explainability Ensuring trust and understandability of AI systems.

Future Research Directions

AI and Human Collaboration Studying the potential of AI to improve both human-to-human and effective interactions [15].

Communication Channels Redefining communication channels between humans and AI systems as AI-powered tools, including AI-powered developers.

Privacy and Sustainability Defining software development paradigms and processes that implement privacy by design and adhere to stringent sustainability and AI regulations.

Automated Software Development Exploring design paradigms for fully AI-automated software development cycle from design to deployment and testing in fast sandbox environments.

User Involvement Ensuring that humans are first class actors in decision-making processes, and involving humans in system construction to build trustful systems [31].

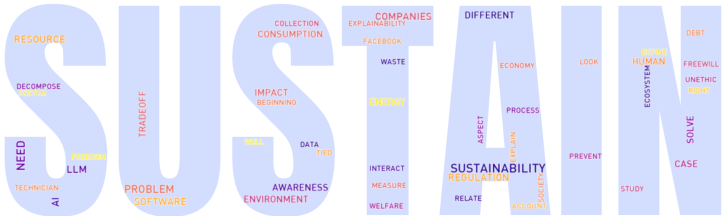
Transparency and Compliance Involving software engineering in implementing tools that provide insights on model training, enhancing transparency and compliance with upcoming regulations to democratize computer science and build trustful AI systems.

DEI Research Addressing the inconsistency in the quality of Diversity, Equity, and Inclusion (DEI) papers, and improving the reviewing processes to prevent the publication of misleading studies.

Education and Training Educating a new generation of software engineers, who can manage the new challenges of AI tools and software engineering [3].

Software Engineering by and for Humans: Research should focus on bridging the technical and human aspects of software engineering and defining ethical rules for AI systems. Improving human-to-human collaboration through AI and ensuring continuous human involvement are key to addressing ethical and interaction challenges.

4. SUSTAINABLE SOFTWARE ENGINEERING



Sustainable software engineering is becoming increasingly critical as we consider the welfare, freedom, and freewill within the human ecosystem. We need to approach sustainability from multiple perspectives: societal, environmental, and economic [6]. The importance of sustainability today is driven by the enormous demand for resources, evolving human values, and governmental regulations [13].

Open Research Issues

Corporate Responsibility Defining regulation to ensure that organizations adopt sustainable practices and prevent unethical data collection, as exemplified by cases like Meta[3].

Energy Consumption Preventing useless usage of expensive and unnecessary AI. AI processing requires substantial energy, and LLMs are particularly energy inefficient [33]. Periodic sustainability benchmarks could help monitor and improve their impact. However, the negative impacts of LLMs on companies are still largely unknown. This ties to explainability that is a first required step to properly assign tasks to different entities.

[3]The New York Times, "Meta Fined $1.3 Billion for Violating E.U. Data Privacy Rules, `https://www.nytimes.com/2023/05/22/business/meta-facebook-eu-privacy-fine.html`

Metrics for Sustainability Measuring the multiple facets of sustainability from the inception of the projects. Software waste and technical debt are pressing issues. We need metrics of software sustainability. However the concept of metrics itself is ill-defined [14].

Awareness and Education Raising awareness about sustainability within companies, often unaware of the core issues of software sustainability. Education plays a crucial role in fostering a sustainable approach within the software engineering community [26] [32].

Trade-offs in Sustainability Assessing whether people are ready to make these trade-offs [10]. Sustainability often involves trade-offs, such as balancing perfect services against resource consumption.

Future Research Directions

Developing a Sustainability Mindset Studying and understanding the different dimensions of sustainability and developing a "sustainability mindset" within organizations. This involves raising awareness among researchers about sustainability.

Modulating Resource Consumption Automatically distributing the resource consumption of LLMs according to the nature of the problem.

Regulatory Compliance Ensuring compliance with regulations and laws that are increasingly tailored to software.

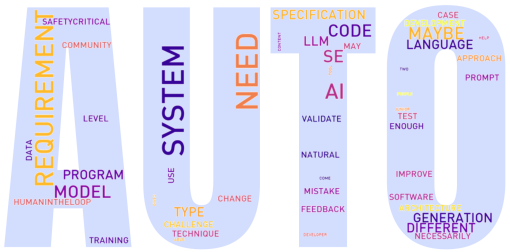
Developing Business Cases and Examples Developing business cases and case studies to witness the importance of sustainable practices.

Defining Sustainability Metrics and Benchmarks Defining sustainability metrics and benchmarks to reduce the environmental impact of software and AI systems. Establishing clear measures of software sustainability and integrating them from the beginning of a project is crucial for long-term success.

Promoting Educational Initiatives Developing educational initiatives to improve a sustainable approach within the software engineering community.

Sustainable Software Engineering: Research needs to develop specific sustainability metrics and foster a sustainability mindset within organizations. It is essential to balance resource consumption with service quality while minimizing software ecological footprint.

5. AUTOMATIC PROGRAMMING



Machine learning and generative AI already provide powerful tools for automatically generating code [23], and open important questions about the scalability of code generation as well as about quality and trustfulness of the generated code.

Open Research Issues

Program Synthesis Defining scalable ML-driven approaches to synthetize predictable, verifiable, and testable programs, by taking advantage of the results of template-based approaches.

Automation and Human-in-the-Loop Defining LLM-based agent systems to build teams and incorporate feedback. The degree of automation varies, and the human-in-the-loop issue may evolve over time. We need LLM-based agent systems to build teams and incorporate feedback.

Context and Adaptability Defining a comprehensive software engineering approach that encompasses many abstract levels such as requirements, architecture, specifications and validation, and is resistant to regulations, hardware, and architecture changes.

Validation and Formal Specifications Validating both system specifications and behavior through formal specifications, by transforming natural language into an intermediate level and then into code.

Domain-Specific Languages Defining domain-specific languages tailored to different types of systems, and integrated with LLMs for effective automatic programming.

Training Data and Model Performance Defining different models for various application domains, to ensure that training data do not lead AI to repeat human mistakes.

Human-in-the-Loop Interactions Keeping humans in the loop to refine and improve the outcomes of automatic programming that relies on the use of models[22].

Prototypes and Models Fast generating prototypes and models that encompasses code, as well as requirements, test cases, and design artifacts[5].

Conversational Approaches and Bug Handling Defining conversational approaches for improving requirements, and handling bugs, especially in safety-critical systems and non-functional requirements[36].

Training Data Quality Ensuring that training data do not lead AI to repeat human mistakes is vital. Different models may be required for various application domains.

Future Research Directions

Improving Program Synthesis Techniques Enhancing the longevity and effectiveness of program synthesis techniques.

Enhancing LLM Style Code Generation Developing robust prompt engineering methods, and integrating them into large systems.

Developing New Agent Systems Creating new agent systems based on LLMs that can build teams and incorporate feedback.

Context-Aware Systems Defining approaches to produce systems that adapt to changing contexts, such as regulations, hardware, and architecture.

Establishing Validation Methods Developing methods to validate system specifications and behavior through formal specifications.

Creating Domain-Specific Languages Defining domain-specific languages tailored to different systems, and integrating them with LLMs, to improve the effectiveness of automatic programming.

Ensuring High-Quality Training Data Ensuring that training data are diverse and accurate, and developing models for different application domains.

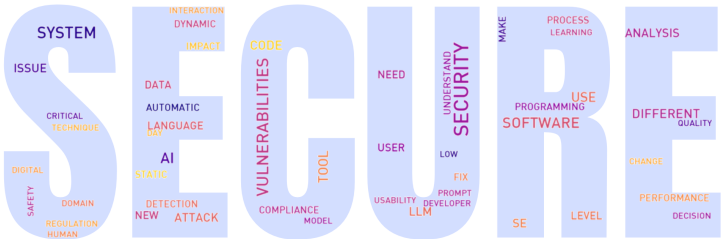
Enhancing Human-in-the-Loop Interactions Improving human-in-the-loop interactions to refine and improve AI outcomes.

Utilizing Prototypes and Models Emphasizing the use of prototypes and models in automatic programming.

Handling Bugs in Conversational Approaches Developing methods to handle bugs effectively in conversational approaches to improve the reliability of AI systems, especially in safety-critical applications.

Software Quality with Automatic Programming: Future work should (i) integrate code generation with Large Language Models (LLMs), (ii) improve prompt engineering techniques, (iii) develop domain-specific languages specific for different system types for improving software quality.

6. SOFTWARE SECURITY ANALYSIS



Software security analysis is crucial in today's landscape, where security threats and vulnerabilities are ever-present, and the domain language barrier complicates communication and tool integration, as terms have different meanings across various areas.

Open Research Issues

Domain Language Barrier Melting terms that are used with different meanings across areas by taking advantage of AI APIs to bridge the gaps.

Static and Dynamic Analysis Techniques Reducing the extensive validation and improving the usability of static and analysis techniques to find vulnerabilities.

Security-Aware Tools Defining security-aware tools that are pivotal in automatic vulnerability detection and compliance processes, such as risk modeling and minimum elements compliance.

Generating Secure Code Leveraging prompts and ensuring that LLMs contribute effectively, despite lower performance in dynamic analysis scenarios.

Detecting and Fixing Vulnerabilities Guaranteeing continuous commitment in vulnerability data and complexity in anomaly detection.

Deep Learning for Anomaly Detection Defining simple deep learning approaches to detect anomalies, and explain decisions.

Privacy Issues and Software Supply Chain Practices Enhancing API usability for developers with varying security expertise, and training developers with expertise in secure practices conforming to European Union data regulations.

AI Tools for Secure Algorithms Producing secure algorithms through a mix of experts, from natural language to machine instructions, requires tools for secure logic review [28].

Legal Accountability and Continuous Security Compliance Balancing legal and technical aspects for critical as well as non-critical systems, to both empower users and secure the interactions.

Edge Computation for Security Enhancing security by minimizing data exposure, vulnerabilities and securing system interactions, to prevent attacks to public institutes.

Learning Systems and AI in Security Defending against AI-based attacks and securing against vulnerabilities emphasizes the need for research in software security.

Future Research Directions

Addressing Domain Language Barriers Developing standardized terminology and AI APIs to improve communication and tool integration across different areas.

Improving Static and Dynamic Analysis Techniques Improving the usability and validation of tools to effectively uncover vulnerabilities.

Advancing Security-Aware Tools Creating effective automatic vulnerability detection tools and compliance processes, to both improve risk modeling and ensure compliance with security standards.

Enhancing Secure Code Generation Improving prompts and LLM performance in dynamic analysis scenarios, to generate more secure code.

Balancing Vulnerability Data Addressing the imbalance in vulnerability data, and simplifying anomaly detection methods, to improve detection and fixing processes.

Simplifying Deep Learning Anomaly Detection Developing deep learning-based anomaly detection that is both explainable and user-friendly.

Streamlining Security Issue Reporting Developing a responsive and efficient vulnerability disclosure process.

Strengthening Privacy and Supply Chain Security Enhancing secure software supply chain practices and complying with EU data regulations.

Developing AI Tools for Secure Algorithms Creating tools that combine expert knowledge and AI to produce and review secure algorithms.

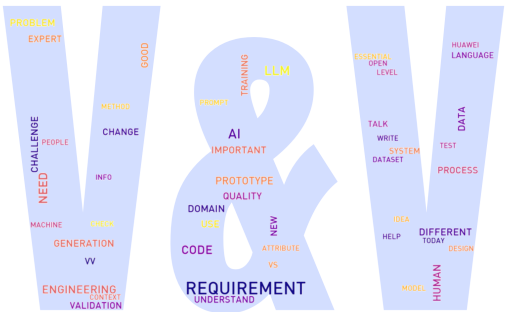
Ensuring Legal Accountability and Continuous Compliance Implementing continuous security compliance processes for both critical and non-critical systems.

Utilizing Edge Computation for Security Enhancing security by reducing data exposure and vulnerabilities, and addressing federation attacks.

Researching AI in Security Investigating the relationship between learning systems and AI to defend against AI-based attacks and secure systems.

Software Security Analysis: The research shall focus on (i) improving security-aware tools and anomaly detection with LLMs, (ii) improving the vulnerability disclosure process and (iii) securing software supply chains for robust software security.

7. VALIDATION AND VERIFICATION



Validating and verifying (V&V) code generated by machines poses unique challenges, and raises questions about how to assess quality and efficiency and ensure reliability of automatically generated software systems [8]. Requirements engineering in the context of AI systems demands transparency and new evaluation methods beyond traditional standards. Prompt engineering is crucial, as changing prompts can yield different outcomes from LLMs.

Open Research Issues

Quality and Efficiency Assessment Evaluating the quality, efficiency and reliability of machine-generated code.

Transparency in Requirements Engineering defining quality standards for machine-generated code, by incorporating transparency and trustfulness.

Impact of Prompt Engineering Defining consistent and effective prompt engineering strategies with predictable variability in the outcomes.

New Testing Methods Defining continuous validation approaches for evaluating machine-generated code [17].

Domain-Specific Needs Defining domain-specific methodologies to validate the requirements of different domains.

Robust Requirements Writing Writing robust quality requirements for automatically generate code, especially when automating code generation with LLMs.

Adapting Education in Requirement Engineering Training for requirement engineering by focusing on AI, automatic testing, and specialized processes.

Future Research Directions

Innovative Approaches to Prototyping Defining approaches for rapid prototyping from requirements, adapting to organizational contexts, and leveraging AI as a team member alongside domain experts to facilitate dialogue and ensure clarity [7].

Evolving Role of Requirement Engineers Improving the detection and refinement of requirements through collaborative tools and diverse data sets, with AI capabilities playing a significant role [30].

Human-Computer Interaction (HCI) in V&V Defining streamlining requirement processes with LLMs using HCI principles to ensure robust validation and verification.

Continuous Validation and Updates Establishing continuous validation processes to keep up with evolving requirements and AI capabilities.

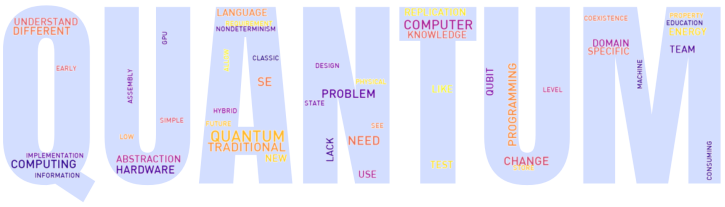
Improved Testing Methodologies Developing new testing methodologies that address the complexities of AI-generated code and the specific needs of various domains.

Integration of AI in Requirement Engineering Education Adapting educational programs to include AI-focused training for requirement engineering, emphasizing the importance of automatic testing and specialized processes.

AI-Assisted Requirement Refinement Utilizing AI to assist in the refinement and interpretation of requirements, ensuring accuracy and completeness.

Validation and Verification: Research should develop new evaluation methods and use AI to refine requirements and improve V&V transparency, by focusing on the improvement of human-computer interaction (HCI) in V&V processes for robust validation and verification of AI-generated code.

8. QUANTUM SOFTWARE ENGINEERING



Quantum Engineering raises unique challenges and requires deep changes in software engineering paradigms to cope with the distinctive characteristics of quantum computing. Requirements engineering remains fundamental, yet it may significantly differ from classic approaches. Programming in quantum environments resembles low-level assembly coding, and emphasizes the need for specialized education and training. Integration of quantum and classical computing systems may be possible with hybrid GPU setups.

Open Research Issues

Lack of Established Abstractions and Approaches Defining requirements engineering, adaptation, and design approaches that cope with the different abstractions in quantum programming.

Testing Methodologies Defining testing strategies for the non-replicable nature of qbit states of quantum software systems [29].

Non-Determinism and Compatibility Defining approaches that handle the non-determinism nature of quantum software systems and address the incompatibility with classic software architectures.

New Programming Paradigms Developing a mindset, programming paradigm, programming languages and methodologies for the distinctive nature of quantum engineering.

Energy-Intensive Hardware Balancing quantum and traditional computational resources to deal with the high energy costs of quantum computing.

Common Abstractions Developing common abstractions across quantum and traditional computing domains.

Specialized Education and Training Developing training material for educating and training quantum engineers and programmers.

Future Research Directions

Innovative Design and Requirements Engineering Exploring new design approaches and requirements engineering that accommodates the unique characteristics of quantum systems.

Testing and Verification Strategies Developing novel testing and verification strategies that account for the physical properties and non-deterministic nature of quantum computing [29].

Interdisciplinary Collaboration Leveraging interdisciplinary collaboration to navigate hardware constraints and address the unique demands of quantum software engineering.

Integration with Classical Systems Investigating the integration of quantum and classical computing systems to create hybrid solutions.

Education and Training Programs Establishing specialized education and training programs to equip software engineers with the necessary skills for quantum programming.

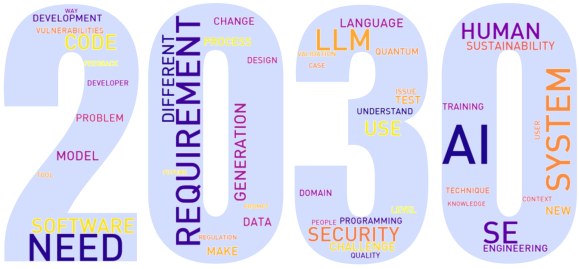
Energy Efficiency Researching ways to optimize energy consumption in quantum computing, balancing computation allocation between quantum and traditional hardware.

Development of New Languages Creating new programming languages and methodologies tailored to the unique requirements of quantum computing.

Common Abstractions and Frameworks Developing common abstractions and frameworks to bridge the gap between quantum and classical computing paradigms.

Quantum Software Engineering: Research should focus on (i) new design approaches and specialized training for quantum software to tackle unique quantum system challenges, and (ii) common abstractions for integrating quantum and classical computing systems.

9. FINAL CONSIDERATIONS



In the 2030 Software Engineering Roadmap workshop we focus on seven core areas that deeply challenges the software engineering research and that we summarize in Table 1:

Artificial Intelligence for Software Engineering (AI4SE) The main challenges in AI4SE involve ensuring trust in AI-generated outputs and integrating AI tools into the software development workflow. Future research should focus on enhancing AI training processes, improving model performance through multi-model approaches and feedback loops, and addressing non-functional requirements.

Software Engineering by and for Humans The key challenge of AI-powered systems is the disconnection between software developers and users. Ethical implications and continuous human involvement are crucial. Future research directions include bridging the technical and human aspects of software engineering, defining ethical rules and regulations, and enhancing human-to-human collaboration facilitated by AI.

Sustainable Software Engineering The environmental impact of software, including the energy inefficiency of LLMs, and software waste are significant concerns. Future research should aim to develop sustainability benchmarks, build a sustainability mindset within organizations, and balance resource consumption with software quality.

Automatic Programming The core challenges of automatic programming include understanding the capabilities of models and developing new techniques for context and prompt engineering. Future research should focus on integrating code generation with LLMs, improving prompt engineering, and developing domain-specific languages and methodologies focused on automated programming.

Software Security Analysis Domain language barriers and usability challenges hinder effective security analysis. Future research directions include improving security-aware tools for automatic vulnerability detection, improving anomaly detection with deep learning, and developing secure software supply chain practices that comply with regulatory standards.

Validation and Verification (V&V) Validating and verifying machine-generated code presents unique challenges. Future research should develop new evaluation methods, leverage AI for requirement refinement, and improve human-computer interaction in V&V processes to ensure robust validation and verification.

Quantum Software Engineering Quantum software engineering is an emerging field, with significant knowledge gaps and non-determinism challenges. Future research should focus on developing new design approaches, providing specialized education and training in quantum programming, and exploring common abstractions for integrating quantum and classical computing systems.

The *25/10 Crowd Sourcing*[4] panel highlighted five top research challenges:

Challenge	SUM	MEAN
Evaluating the impact of future systems on society and sustainability	37	4.75
Measuring sustainability	34	4.40
Requirements for ML-enabled socio-technical systems	25	4.25
Cross-cutting requirements (security performance, sustainability)	25	4.17
Empowering the interactions between humans and self-adaptive autonomous systems	25	4.17

Impact of future systems on society and sustainability emerges as the most critical challenge with the highest sum of votes (37) and the highest mean score (4.75). This indicates a strong consensus on the importance of understanding how future technologies will affect societal and environmental aspects, reflecting a growing concern for sustainability and human-centric engineering.

Measuring sustainability is also highly rated, with a sum of 34 and a mean score of 4.40. This challenge highlights the need for effective metrics to quantify sustainability, which is crucial for making informed decisions about technology deployment and its long-term effects.

Requirements for ML-enabled socio-technical systems and **Cross--cutting requirements (security performance, sustainability)** both received the same cumulative rate (25) and mean score (4.17). The focus on machine learning-enabled systems and cross-cutting requirements suggests a significant interest in addressing the integration of advanced technologies with comprehensive criteria that include security and sustainability.

Approaches to empower users interactions with self-adaptive autonomous systems also scored a mean of 4.17. This challenge emphasized the need to design systems that both prioritize user needs and improve user experience, by ensuring that technology effectively serves the human needs.

The results clearly reflect the emphasis on the disruptive effects of AI, user-centric approaches and sustainability in the development of future software systems. These challenges reveal key areas where future research and development efforts should be concentrated to align technological advancements with societal needs and expectations.

Overall, the 2030 Software Engineering Roadmap workshop has provided a comprehensive overview of the current and future trends in software engineering. The integration of AI, quantum computing, and autonomous systems into software engineering practices represents a paradigm shift, bringing significant advancements and posing new challenges. As the software engineering community addresses these changes, it is essential to focus

[4]An excellent introduction to the liberating structure is available at `https://www.liberatingstructures.com/12-2510-crowd-sourcing/`

Table 1: 2030 Open Issues and Future Research Directions in Software Engineering

Topic	Main Open Issues	Future Research Directions
AI for Software Engineering	Trust in machine-generated code, explainability, integration into workflows, quality measures for machine-generated code	Improving model performance with multi-models, defining feedback loops, addressing non-functional requirements, improving trainings
Software Engineering by and for Humans	Disconnection between developers and users, ethical implications, continuous human involvement	Bridging technical and human aspects, defining ethical rules and regulations, enhancing AI-human collaboration
Sustainable Software Engineering	Environmental impact, technical debt, energy inefficiency of LLMs	Developing sustainability metrics, building a sustainability mindset, balancing resource consumption with software quality
Automatic Programming	Capabilities of models, development of new techniques, context and prompt engineering	Integrating program synthesis and LLMs, improving prompt engineering, developing domain-specific languages
Software Security Analysis	Domain language barriers, usability challenges, continuous vulnerability detection	Improving security-aware tools, improving anomaly detection with deep learning, secure software supply chain practices
Validation and Verification (V&V)	Quality assessment of machine-generated code, transparency in requirements engineering, domain-specific needs	Developing new evaluation methods, leveraging AI for requirements refinement, enhancing HCI in V&V processes
Quantum Software Engineering	Lack of quantum physics knowledge, nondeterminism, compatibility with classical computing	Developing new design approaches, specialized education and training, exploring common abstractions for quantum and classical systems

on ethical considerations, sustainability, security, and the evolving role of humans within software ecosystems. The collaboration between academia and industry will be crucial in redefining the role of software engineers and adapting to the rapidly evolving landscape of software engineering. By embracing these challenges and opportunities, the software engineering community can build a resilient and innovative future, ensuring that software systems remain reliable, secure, and aligned with societal values.

Acknowledgment

We would like to thank Daniel Russo for the invaluable help in defining the ideal liberation structure for the program of the workshop.

10. REFERENCES

[1] Steven Arzt, Linda Schreiber, and Dominik Appelt. Position: How regulation will change software security research. *2030 Software Engineering Workshop, https://arxiv.org/abs/2406.04152*, 2024.

[2] Wesley K. G. Assunção, Luciano Marchezan, Alexander Egyed, and Rudolf Ramler. Contemporary software modernization: Perspectives and challenges to deal with legacy systems. *2030 Software Engineering Workshop, https://arxiv.org/abs/2407.04017*, 2024.

[3] Stefanie Betz and Birgit Penzenstadler. With great power comes great responsibility: The role of software engineers. *2030 Software Engineering Workshop, https://arxiv.org/abs/2407.08823*, 2024.

[4] Christian Birchler, Sajad Khatiri, Pooja Rani, Timo Kehrer, and Sebastiano Panichella. A roadmap for simulation-based testing of autonomous cyber-physical systems: Challenges and future direction. *2030 Software Engineering Workshop, https://arxiv.org/abs/2405.01064*, 2024.

[5] Lola Burgueño, Davide Di Ruscio, Houari Sahraoui, and Manuel Wimmer. The past, present, and future of automation in model-driven engineering. *2030 Software Engineering Workshop, https://arxiv.org/abs/2405.18539*, 2024.

[6] Coral Calero, Félix O. García, Gabriel Alberto García-Mireles, M. Ángeles Moraga, and Aurora Vizcaíno. Addressing sustainability-in software challenges. *2030 Software Engineering Workshop, https://arxiv.org/abs/2406.07380*, 2024.

[7] Roberto Casadei, Gianluca Aguzzi, Giorgio Audrito, Ferruccio Damiani, Danilo Pianini, Giordano Scarso, Gianluca Torta, and Mirko Viroli. Software engineering for collective cyber-physical ecosystems. *2030 Software Engineering Workshop, https://arxiv.org/abs/2406.04780*, 2024.

[8] Johan Cederbladh and Antonio Cicchetti. A road-map for transferring software engineering methods for model-based early v&v of behaviour to systems engineering. *2030 Software Engineering Workshop, https://arxiv.org/abs/2406.04037*, 2024.

[9] Matteo Ciniselli, Niccolò Puccinelli, Ketai Qiu, and Luca Di Grazia. From today's code to tomorrow's symphony: The ai transformation of developer's routine by 2030. *2030 Software Engineering Workshop, https://arxiv.org/abs/2405.12731*, 2024.

[10] Luís Cruz, Xavier Franch Gutierrez, and Silverio Martínez-Fernández. Innovating for tomorrow: The convergence of se and green ai. *2030 Software Engineering Workshop, https://arxiv.org/abs/2406.18142*, 2024.

[11] Aline de Campos, Jorge Melegati, Nicolas Nascimento, Rafael Chanin, Afonso Sales, and Igor Wiese. Some things never change: how far generative ai can really change software engineering practice. *2030 Software Engineering Workshop, https://arxiv.org/abs/2406.09725*, 2024.

[12] Ronnie de Souza Santos, Felipe Fronchetti, Savio Freire,

and Rodrigo Spinola. Software fairness debt. *2030 Software Engineering Workshop, https://arxiv.org/abs/2405.02490*, 2024.

[13] Lidia Fuentes. Engineering software for next-generation networks in a sustainable way. *2030 Software Engineering Workshop, https://doi.org/10.5281/zenodo.11579614*, June 2024.

[14] Jennifer Gross and Sofia Ouhbi. Clearing the path for software sustainability. *2030 Software Engineering Workshop, https://arxiv.org/abs/2405.15637*, 2024.

[15] Ahmed E Hassan, Gustavo A Oliva, Dayi Lin, Boyuan Chen, Zhen Ming, et al. Rethinking software engineering in the foundation model era: From task-driven ai copilots to goal-driven ai pair programmers. *2030 Software Engineering Workshop, arXiv preprint arXiv:2404.10225*, 2024.

[16] Junda He, Christoph Treude, and David Lo. LLM-based multi-agent systems for software engineering: Vision and the road ahead. *2030 Software Engineering Workshop, https://arxiv.org/abs/2404.04834*, 2024.

[17] Sinclair Hudson, Sophia Jit, Boyue Caroline Hu, and Marsha Chechik. A software engineering perspective on testing large language models: Research, practice, tools and benchmarks. *2030 Software Engineering Workshop, https://arxiv.org/abs/2406.08216*, 2024.

[18] Sonja M. Hyrynsalmi, Sebastian Baltes, Chris Brown, Rafael Prikladnicki, Gema Rodriguez-Perez, Alexander Serebrenik, Jocelyn Simmonds, Bianca Trinkenreich, Yi Wang, and Grischa Liebel. Bridging gaps, building futures: Advancing software developer diversity and inclusion through future-oriented research. *2030 Software Engineering Workshop, https://arxiv.org/abs/2404.07142*, 2024.

[19] Victoria Jackson, Bogdan Vasilescu, Daniel Russo, Paul Ralph, Maliheh Izadi, Rafael Prikladnicki, Sarah D'Angelo, Sarah Inman, Anielle Lisboa, and Andre van der Hoek. Creativity, generative ai, and software development: A research agenda. *2030 Software Engineering Workshop, https://arxiv.org/abs/2406.01966*, 2024.

[20] Marcus Kessel and Colin Atkinson. Morescient gai for software engineering. *2030 Software Engineering Workshop, https://arxiv.org/abs/2406.04710*, 2024.

[21] Rui Li, Huai Liu, Pak-Lok Poon, Dave Towey, Chang-Ai Sun, Zheng Zheng, Zhi Quan Zhou, and Tsong Yueh Chen. Metamorphic relation generation: State of the art and visions for future research. *2030 Software Engineering Workshop, https://arxiv.org/abs/2406.05397*, 2024.

[22] Zhihao Lin, Wei Ma, Tao Lin, Yaowen Zheng, Jingquan Ge, Jun Wang, Jacques Klein, Tegawende Bissyande, Yang Liu, and Li Li. Open-source ai-based se tools: Opportunities and challenges of collaborative software learning. *2030 Software Engineering Workshop, https://arxiv.org/abs/2404.06201*, 2024.

[23] Michael R. Lyu, Baishakhi Ray, Abhik Roychoudhury, Shin Hwei Tan, and Patanamon Thongtanunam. Automatic programming: Large language models and beyond. *2030 Software Engineering Workshop, https://arxiv.org/abs/2405.02213*, 2024.

[24] Antonio Mastropaolo, Camilo Escobar-Velásquez, and Mario Linares-Vásquez. The rise and fall(?) of software engineering. *2030 Software Engineering Workshop, https://arxiv.org/abs/2406.10141*, 2024.

[25] Facundo Molina and Alessandra Gorla. Test oracle automation in the era of LLMs. *2030 Software Engineering Workshop, https://arxiv.org/abs/2405.12766*, 2024.

[26] Ana Moreira, Ola Leifler, Stefanie Betz, Ian Brooks, Rafael Capilla, Vlad Constantin Coroama, Leticia Duboc, Joao Paulo Fernandes, Rogardt Heldal, Patricia Lago, Ngoc-Thanh Nguyen, Shola Oyedeji, Birgit Penzenstadler, Anne Kathrin Peters, Jari Porras, and Colin C. Venters. A road less travelled and beyond: Towards a roadmap for integrating sustainability into computing education. *2030 Software Engineering Workshop, https://arxiv.org/abs/2406.18945*, 2024.

[27] Shola Oyedeji, Ruzanna Chitchyan, Mikhail Ola Adisa, and Hatef Shamshiri. Integrating sustainability concerns into agile software development process. *2030 Software Engineering Workshop, https://arxiv.org/abs/2407.17426*, 2024.

[28] Nikhil Patnaik, Joseph Hallett, and Awais Rashid. Saltzer & schroeder for 2030: Security engineering principles in a world of ai. *2030 Software Engineering Workshop, https://arxiv.org/html/2407.05710v1*, 07 2024.

[29] Neilson Carlos Leite Ramalho, Higor Amario de Souza, and Marcos Lordello Chaim. Testing and debugging quantum programs: The road to 2030. *2030 Software Engineering Workshop, https://arxiv.org/abs/2405.09178*, 2024.

[30] Diana Robinson, Christian Cabrera, Andrew D. Gordon, Neil D. Lawrence, and Lars Mennen. Requirements are all you need: The final frontier for end-user software engineering. *2030 Software Engineering Workshop, https://arxiv.org/abs/2405.13708*, 2024.

[31] Martina De Sanctis, Paola Inverardi, and Patrizio Pelliccione. Engineering digital systems for humanity: Challenges and opportunities. *2030 Software Engineering Workshop, https://arxiv.org/abs/2406.09065*, 2024.

[32] Hatef Shamshiri, Ashok Tripathi, Shola Oyedeji, and Jari Porras. Exploring the experiences of experts: Sustainability in agile software development – insights from the finnish software industry. *2030 Software Engineering Workshop, https://arxiv.org/abs/2407.06978*, 2024.

[33] Jieke Shi, Zhou Yang, and David Lo. Efficient and green large language models for software engineering: Vision and the road ahead. *2030 Software Engineering Workshop, https://arxiv.org/abs/2404.04566*, 2024.

[34] Valerio Terragni, Partha Roop, and Kelly Blincoe. The future of software engineering in an ai-driven world. *2030 Software Engineering Workshop, https://arxiv.org/abs/2406.07737*, 2024.

[35] Qing Wang, Junjie Wang, Mingyang Li, Yawen Wang, and Zhe Liu. A roadmap for software testing in open collaborative development environments. *2030 Software Engineering Workshop, https://arxiv.org/abs/2406.05438*, 2024.

[36] Qi Xin, Haojun Wu, Steven P. Reiss, and Jifeng Xuan. Towards practical and useful automated program repair for debugging. *2030 Software Engineering Workshop, https://arxiv.org/abs/2407.08958*, 2024.