U.S. Leadership in Software Engineering & AI Engineering: Critical Needs & Priorities Workshop Report

June 20-21, 2023

Software Productivity, Sustainability, and Quality (SPSQ)
Interagency Working Group (IWG)
and
Carnegie Mellon University's Software Engineering Institute (CMU SEI)

Anita Carleton and Forrest J. Shull, Organizers, CMU SEI
Ram D. Sriram and Sol Greenspan, Co-Chairs, NITRD

Table of Contents

Executive Summary

To inform a community strategy for building and maintaining U.S. leadership in software engineering and AI engineering, the Carnegie Mellon University's Software Engineering Institute (CMU SEI) and the Software Productivity, Sustainability, and Quality (SPSQ) Interagency Working Group co-hosted the workshop at the National Science Foundation from June 20th to the 21st, 2023.

The event gathered thought leaders from federal research funding agencies, research laboratories, mission agencies, and commercial organizations. Participants used the SEI's *Architecting the Future of Software Engineering: A National Agenda for Software Engineering Research and Development*[1] as a starting point because the areas of focus that the study identifies have been confirmed as increasingly critical, particularly due to the rapid advances of generative artificial intelligence (AI) in the two years since its release. Specifically, participants identified the following three research areas from the study as having direct relevance: AI-augmented Software Development, Assuring Continuously Evolving Software Systems, and Engineering AI-Enabled Software Systems.

Speakers and participants at the event worked to explore software-related challenges that are critical for multidisciplinary research across domains of importance to the nation, as well as the promising research that is needed to engineer the necessary systems reliably and well. In sessions organized around these research areas and in breakout discussions, participants almost unanimously remarked on the rapid acceleration of new technologies in the software development lifecycle and the role of AI in shaping the future of software systems. As attendees discussed the critical need for new approaches to navigate both the opportunities and the challenges of this changing landscape, five main themes emerged.

1. AI is transforming the software engineering process and how we engineer software systems. The increasing symbiosis of humans and machines is transforming every phase of the software development lifecycle.
2. Generative AI has reached a level of sophistication that may seem to resemble human intelligence, but it remains difficult to determine the level of trust that should be placed in its outputs.
3. It is imperative to redefine the discipline of software engineering to encompass the use of new technologies (including, but not limited to, generative AI), and to rethink the curricula, tools, and technologies associated with its practice. This effort is key to designing, building, evolving, and evaluating trustworthy software systems in a responsible, ethical way.
4. New technologies, including generative AI, seem to hold the promise of making almost everyone a programmer. As a result, AI literacy and the development of new skills are needed throughout the workforce.
5. The use of AI tools, such as large language models (LLMs), can mask the trade-offs being made between the functionality of software systems and their safety and security. Research is needed to identify and make explicit the key engineering trade-offs being made during the design, development, training, testing, and authorization of systems that include AI components.

[1] See https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=741193 to download a copy of the study.

Acknowledgements

The workshop organizers gratefully acknowledge the following individuals for their efforts in bringing the workshop to fruition:

Paul Nielsen, Anita Carleton, Forrest J. Shull, Kamie Roberts, Ram D. Sriram, Sol Greenspan, Erin Harper, John Robert, Ipek Ozkaya, Doug Schmidt, William Scherlis, Tom Longstaff, Eileen Wrubel, Michele Falce, Linda Canon, and Melissa Cornelius.

Introduction

Advancing the closely related disciplines of software engineering and artificial intelligence (AI) engineering is indispensable to our ability to develop and deploy intelligent software systems. The engineering of AI capabilities has its own set of unique and challenging requirements, but the two fields remain closely related because these capabilities are implemented in software. To date, there has been significant research within the software engineering discipline on technologies and practices needed to build AI-enabled systems. Most recently, research has focused significant attention on the fundamental theories, practices, and knowledge base for AI engineering to ensure that AI capabilities are incorporated into systems with expected trustworthiness and responsibility.

There has also been considerable excitement around the idea of using AI to help in the engineering of software systems at scale. Approaches that leverage large language models (LLMs) are already automating some tasks that were thought to require human creativity, including some aspects of software engineering. As the boundaries of software and AI engineering blend, the tools and techniques available to engineers for developing capabilities are also changing. This rapidly evolving technical environment creates further urgency to prioritize areas of critical need and allocate multidisciplinary resources to the most challenging and essential areas of concern.

Broad advances in the fields of software engineering and AI are providing critical and innovative capabilities across almost every domain, but the potential remains to do far more, particularly for applications that demand high levels of trustworthiness. The U.S. Leadership in Software Engineering & AI Engineering: Critical Needs & Priorities Workshop brought together approximately 70 participants to encourage new partnerships that will advance U.S. leadership and national interests through the disciplines of software and AI engineering, and that will positively impact progress across virtually all scientific domains.

Specific objectives for the workshop included the following:

- Characterize how software engineering capabilities are having a direct impact on the future of our nation.
- Inform a community strategy for building and maintaining U.S. leadership in software engineering and AI engineering. Produce a report that summarizes challenges, opportunities, and strategic priorities.
- Identify research questions that energize the computing community and spark new collaborations.
- Identify updates to roadmap outlined in the CMU SEI's Architecting the Future of Software Engineering: A National Agenda for Software Engineering Research and Development.

Keynote and Special Session Speakers

Kamie Roberts, Director of the NITRD National Coordination Office and co-chair of the NITRD Subcommittee of the National Science and Technology Council, welcomed participants and set the tone for the workshop by discussing the pivotal role of software capabilities and AI in shaping the nation's future. She noted that

NITRD, a multiagency program fostering innovations in IT, emphasizes the importance of software productivity, sustainability, and quality through coordinated federal research and development efforts. She went on to underscore that the critical juncture of software engineering and AI is integral to America's competitiveness, innovation, and national security. The National AI Initiative Act of 2020 and recent actions by the Biden-Harris Administration highlight the government's commitment to advancing responsible AI innovation. Software engineering and AI have driven economic growth, technological advancements, and global competitiveness, with major tech companies leveraging these technologies for profit and job creation. She praised the workshop organization, saying that bringing together academia, federal leaders, and industry stakeholders to explore future capabilities, research and development (R&D) needs, and trends is key to fostering innovation and progress. Her full remarks are available in Appendix B.

Professor Doug Schmidt, Cornelius Vanderbilt Professor of Engineering, Associate Provost of Research, and Data Science Institute Co-Director at Vanderbilt University, provided the first keynote, which discussed the challenges of assuring the future of software engineering and AI engineering. He highlighted how generative AI is acting as a transformative force in software development, streamlining processes, and improving system quality. However, he also emphasized the broader scope of software engineering beyond programming, touching on challenges in various phases of the software development lifecycle. He emphasized prompt engineering and its implications for natural language programming, while acknowledging the evolving landscape of AI technologies. The talk went on to address concerns regarding AI-enabled software systems, including data dependency, safety-critical considerations, and the challenge of explainability. He shared experiences of integrating advanced technologies like ChatGPT into educational settings, showcasing its utility in code summarization, unit test generation, and problem-solving. He also stressed the importance of leveraging such tools to augment, rather than replace, learning processes, particularly emphasizing security and ethical considerations. The talk concluded with reflections on the evolving role of programmers and the need for a balanced approach to harnessing AI technologies for software engineering while ensuring accountability and reliability in real-world applications.[2]

The second keynote was provided by Dr. William Sanders, Dean of the College of Engineering, Carnegie Mellon University. He discussed the shift from AI in science fiction to its integration into physical systems, stressing the need for an engineering approach to AI. He further examined three perspectives on AI: theory, applications, and engineering, and he emphasized the latter as integral for effective implementation in engineered systems. He went on to outline four pillars of AI engineering: engineering AI mechanisms, embedding AI in systems, redefining the engineering process with AI, and adhering to ethical constraints. Dr. Sanders showcased examples of AI's impact on engineered systems, including improving quality of life for individuals with neurological damage through AI-driven limb movement restoration. He discussed CMU's initiatives in AI engineering education, including new degree programs and online certificate courses, aimed at equipping engineers with the skills to integrate AI into various domains. The talk concluded by stressing the importance of infusing AI engineering principles across engineering curricula. The slides used for this talk are available in Appendix C.

The final keynote was provided by Dr. Thomas Zimmermann of Microsoft Research, which focused on machine learning's (ML's) impact on future software engineers. He discussed his own AI journey since the fall of 2021—a journey marked by the popularity of GitHub Copilot. He further detailed Microsoft Research's focus on understanding Copilot's usage and its implications for software engineering, highlighting findings published in a recent article, “Taking Flight with Copilot.”[3] He discussed the rapid pace of innovation driven by AI advancements, including GPT-4's near-human performance and its integration into various Microsoft products. He explored ethical considerations in AI development, along with the need to incorporate user-

[2] Keynote speeches https://www.nitrd.gov/coordination-areas/spsq/usa-leadership-in-software-engineering-and-ai-engineering/
[3] ACM Queue, Volume 20, Issue 6, November/December 2022, pp 35-37, https://doi.org/10.2245/3582083.

centered design aspects into its processes. Dr. Zimmerman predicted a fundamental shift in the software engineering process due to AI's influence, which will require building trust in human-AI interaction. The talk reflected on the changing role of research, emphasizing the need for quick prototyping and agile communication methods. Other topics included trust within AI systems, the role of researchers in communicating findings, and the evolving software engineering curriculum in light of AI advancements. In closing, he emphasized the evolving nature of AI's relationship with software engineering and the importance of adapting to rapid changes in the field. His slides are available in Appendix C.

Session I: Government Agency Challenges

Over the years, software has evolved into a vital technology that enables critical national capabilities and has created enormous economic benefit for the country. Much of the attention surrounding new technologies centers on advancements from major commercial technology companies that have leveraged AI and software engineering to build highly profitable businesses. These advances, however, are changing the way software engineering is performed across the board, including at government agencies. Software already plays a pivotal role in virtually every aspect of modern government operations, and many government agencies are looking to software's new capabilities to enhance efficiency, support national security, enable policy implementation, foster innovation, ensure transparency, and much more.

To successfully support national interests and priorities through the use of new technology, researchers, commercial users, and government agencies must bridge communication gaps between each other. During Session I, led by Dr. Forrest Shull, speakers were asked to address the following questions:

- What is an example of the kinds of SW-enabled or AI-enabled systems and capabilities your agency will need in the future?
- How will these systems support critical national needs?
- What gaps or risk areas exist today that make building or acquiring those systems difficult?
- What R&D do you think we need to address those gaps or risks? What could the R&D community do to help?

Key Opportunities and Challenges for Government Agencies

The integration of AI and advanced software engineering techniques in government agencies presents both challenges and opportunities across many domains. Adapting to the changing nature of software development in government agencies requires the development of new workforce skills and a learning environment that allows employees to adapt to the changing nature of software development in government agencies.

The opportunities presented by AI and advanced software engineering can reshape how agencies address many of their challenges, fostering a future where technology plays a pivotal role in ensuring security, efficiency, and meaningful innovation. The information outlined below from Session I encompasses the wide-ranging missions of several federal agencies who were present at the workshop, including aviation security, cybersecurity, border security, emergency response, toxicology research, space exploration, and the establishment of trademarks and patents.

Security, Law Enforcement, and Disaster Response

There is much promise for the future use of AI and advanced software engineering techniques in security, law enforcement, and disaster response-related missions, such as preventing terrorism, securing borders, and responding to disasters. Some of the key challenges and opportunities are as follows:

- **Enabling Synthetic Data Generation**: Synthetic data generation is needed to support missions where there is limited data, such as non-intrusive inspection and scanning for the detection of chemical or biological agents.
- **Increasing Robustness for AI**: In the context of potential security threats and disadvantaged environments, tools and methodologies that ensure the robustness of AI are vital to operation in real-world contexts.
- **Recognizing and Preventing Adversarial AI**: Many concerns exist related to deep fakes and AI-based attacks, and a solid foundation for understanding and preventing these attacks is needed to move forward in this domain.
- **Developing New Frameworks for AI Security**: Existing cybersecurity frameworks are not adequate for implementing effective AI security. New, third-generation frameworks are needed that provide a better foundation for addressing challenges in the adversarial AI space.
- **Developing Common Lexicons and Assurance Frameworks:** There is a lack of clarity in AI-related terminology, definitions, and the frameworks that have been developed to ensure AI systems are trustworthy, secure, responsible, legal, and aligned with values.
- **Deploying Privacy Enhancing Technology:** Protecting citizens' privacy must be paramount when leveraging open source data and social media feeds for law enforcement, providing information without compromising individuals' identities.

National Health and Toxicology

The application of AI and ML in areas such as computational toxicology could provide many benefits. For example, because of the increasing number of chemicals in the environment, traditional approaches for conducting research with animal models is becoming unsustainable. Some of the key challenges and opportunities for the use of AI and ML in this area of work are summarized below.

- **Enhancing Data Federation and Privacy Preservation:** The challenge of data federation, especially across clouds, is key in the context of computational toxicology. The importance of privacy-preserving approaches is paramount, as is multi-institutional data collaboration. Synthetic data could also be useful for protecting health data while allowing meaningful analysis.
- **Establishing the Practical Integration of New Technologies:** AI could be used as a copilot for toxicology researchers, allowing interactive querying and analysis of toxicology data. Generative models could also aid researchers in tasks such as extracting information from technical reports and publications. However, the potential for their widespread use will likely depend on practical considerations, such as the availability of cost-effective, hybrid use of on-premises and cloud computing to train models and perform computations.
- **Maintaining Sustainable Software Efforts:** Translating research software into sustainable forms that can be reused across various contexts is important. Such translation and reuse can be accomplished by engaging with software communities, providing independent support to research software engineers, and supporting practitioners as they work toward this goal.

Aerospace Advancement

AI and software engineering needs are growing in the context of various planned space missions, ranging from deep space exploration to air mobility, which all require capable and trustworthy AI. Some key challenges and opportunities include:

- **Developing AI for Complex Rover Operations:** Research is still needed to overcome many challenges in rover operations on the lunar surface, such as advanced AI for tasks like stereo imaging and path planning.

- **Handling Data for Interstellar Exploration:** Preparing for potential interstellar missions that generate and handle vast amounts of data is a critical challenge that could be partially met by the development of advanced AI systems.
- **Understanding Space Economies:** Exploring the concept of space economies and resource utilization on celestial bodies like the Moon, Mars, and the asteroid belt demands advanced technology support and new models.
- **Reducing Software Engineering Cost:** Applying AI-based tools in software engineering to substantially reduce verification costs is particularly promising in the aerospace industry.
- **Overcoming Regulatory Challenges and Integration Complexity:** The industry is facing challenges due to the lack of regulations, the absence of widely accepted assurance plans, and the need to identify effective AI tools with confidence. Building integrated solutions for verification and validation tools also requires consistent requirements and objectives to ensure seamless integration from top to bottom.

Power and Energy Research and Regulation

AI and software engineering have applications across several different energy-related domains, such as strategic computing for nuclear weapons, maintaining the power grid, and using computation for scientific discoveries. A common theme in this area is the need for correctness and high assurance in the software because of the potential catastrophic consequences of failures in these cyber-physical systems. Some key challenges and opportunities include the following:

- **Establishing the Quality of Training Data:** The quality of training data, particularly from the internet, poses problems in high assurance applications. Evaluating data for use in training language models and doing quality checks is critical.
- **Developing a Strategy for Cyber-Informed Engineering:** An ongoing effort aims to integrate a cyber perspective into the design and operation of complex systems. A national strategy for cyber-informed engineering, including the use of AI, is also needed.
- **Mitigating Malicious Injection Risks:** The injection of unverified or malicious content into platforms such as GitHub represents a serious risk. Adversaries injecting content that the chatbot might not recognize as problematic poses risks to the development and deployment of AI systems.
- **Understanding Evolving Adversarial Actions:** The potential for adversarial actions targeting AI applications is large and not fully understood. For example, understanding how AI "thinks" incorrectly provides a vector for adversaries targeting chatbots.

Information Protection

AI holds promise for addressing the complexities of providing legal protection for tools and ideas. In areas such as awarding trademarks and patents, the aim is to build a foundation for continuous innovation while minimizing redundancy in the examination process. For example, AI could help to identify and ensure unique trademarks for stakeholders and identify innovations that qualify for patents in a more timely manner. Effective AI tools could also help contribute to effective data management and reduce redundancy in the examination process. Some additional opportunities and challenges are identified below.

Ensuring Explainability and Trust: Using AI in patent examinations holds promise for making searches in increasingly large databases more efficient and effective, but explainability is a crucial challenge, especially in scenarios where examiners interact with inventors. Trust between examiners and AI tools hinges on the ability of examiners to comprehend how the tool conducted searches and the examiner's ability to interpret results.

Improving Data Categorization and Leveraging Expertise: With a vast amount of data available across many different fields of knowledge, categorizing and classifying diverse technological innovations is a significant challenge. Efforts are already underway to create a classification scheme to identify various types of technology and innovations. Because examiners cannot be experts in every field, this classification scheme, coupled with AI updates, could guide examiners in recognizing where new ideas fall and when external expertise might enhance the evaluation of applications.

Reducing Negative Impacts and Improving Reliability: Using AI in a trustworthy and valid way could reduce negative impacts downstream in the trademark and patent processes, such as reducing litigation. Building trust in AI tools and ensuring that examiners comprehend their results can lead to more reliable patents, fostering confidence that innovations are unique and minimizing legal disputes.

Facilitating Global Collaboration: Managing the ever-growing amount of data, especially data in multiple languages, requires collaboration not only within one agency but across many U.S. agencies and international offices. AI could help to create a unified approach to comprehending and managing data, avoiding redundancy, and promoting innovation on a global scale.

Session II: AI for Software Productivity, Sustainability, and Quality

Session II delved into the many opportunities and challenges of integrating AI into the software development lifecycle. In particular, it discussed the intersection of AI and software engineering and explored the achievability of promised advancements in productivity, sustainability, and quality. Key discussion revolved around identifying AI's contributions to different phases of software engineering, understanding changes in the overall life cycle that AI would provoke, and evaluating the promises made by the proponents of AI models. Emphasizing the need for ongoing research, this session underscored the swift advancements in AI tools while cautioning against underestimating the complexities of the technological understanding needed to apply them.

During Session II, led by Ipek Ozkaya, principal researcher and technical director of the Engineering Intelligent Software Systems group at the SEI, the speakers were asked to address the following questions:

- What are parts of the software engineering lifecycle where AI could make more of a contribution?
- How does the whole lifecycle look differently with AI in the mix?
- What would it mean for AI to be a “trusted partner” in software engineering? How would we know it when we have it?

Code-Aware Models for Software

The development of code-aware models in the context of AI and ML is imperative. There has been significant increase in the number of possible applications for the emerging capabilities of AI and ML, particularly in program analysis and generative programming. Code copilot tools extend beyond code generation, showing promise for summarization, translation, and various other software engineering tasks, leading to increasing interest within the academic community. A multitude of papers have now been published exploring these facets.

One critical question that has emerged in this field of investigation is, “How much can we trust the code generated by these models?” Experiments conducted using foundational models have revealed discrepancies in code generation based on subtle variations in prompts. This underscores the models’ predominant reliance on code as text, neglecting the broader syntactic, semantic, and contextual aspects of code. Code-aware models need to incorporate diverse properties of syntax and semantics, while examining

factors such as the code's evolution and dynamic changes during its use. Textual properties like comments, bug reports, and configurations are also necessary.

Some code-aware models have already shown promising performance in vulnerability detection, clone detection, and search tasks, outperforming larger models while requiring substantially less data. Code property-aware models exhibit superior generalization, enhanced robustness, and improved efficiency, which highlights the need to conceptualize code not just as text but as a dynamic, evolving entity.

Vulnerabilities in AI Models

The field of program analysis has been a focal point of research for years, with applications ranging from analyzing defective software and surveying failures to conducting penetration tests on robotic vehicles. In recent years, AI has had a profound impact on this research and on the software development lifecycle in general. As a result, many critical questions surrounding the trustworthiness of AI models are now being explored.

In addition to code generation, AI presents many transformative opportunities for various aspects of the software development lifecycle. It could be leveraged to tackle challenging tasks such as unraveling complex software properties, extracting specifications from code using explainable graph models, and generating software specifications. However, backdoors, intentional or not, are pervasive in publicly available AI models. These could be injected from many sources, such as natural language processing or object detection, and raise concerns about the trustworthiness of AI models. In tests, injected triggers and phrases have been used to manipulate model outputs, underscoring the persistent and sometimes subtle nature of these vulnerabilities.

Rigorous reasoning systems are needed to ensure consistent and trustworthy results in the realm of AI, in addition to a proactive approach for identifying and repairing AI model vulnerabilities. Confining AI models within a controlled environment, akin to sandboxing in traditional software, is another potential strategy to mitigate risks. Building resilience against the inevitable existence of defects and vulnerabilities in AI will continue to be an important area of research.

Shaping the Department of Defense's (DoD's) AI Future

The intricate landscape of the Department of Defense (DoD) means AI development faces unique challenges. The integration of AI into the military ecosystem includes both business and warfighter applications, and efforts are already ongoing to determine how to provide a responsible and strategic AI infrastructure within the DoD.

For example, ongoing exercises and collaborations are focusing on the development of a data integration layer that could enable better real-time communication among global allies. Another area of work is the development of a forward-looking architecture for AI and ML infrastructure. Many challenges are shaping this effort, including addressing security challenges, developing acquisition strategies, and collaborating with industry partners to adapt their methods and results.

Improved data quality and performance metrics present another opportunity for the use of AI and ML. Shared ontologies and standardized vocabularies enhance data quality by creating a cohesive environment that transcends silos. The development of business performance metrics, including key performance indicators (KPIs), is needed to effectively channel information to decision-makers and determine alignment with DoD goals. A data mesh approach, emphasizing deep integration of data and technology, should be employed. Cultivating expertise within the DoD is also critical. Efforts to foster a collaborative environment that connects skilled individuals across organizations are increasing.

Challenges for Using Machine Learning in the Scientific Community

The scientific community has many opportunities for using AI to improve software productivity. It also faces some pivotal challenges that cast a shadow on the future of ML. One concern relates to the reliance on black-box models that are closely guarded by individual companies. The opacity of the models raises reproducibility issues, making research results unpredictable and potentially rendering them useless. Another challenge stems from the fact that the ML used for training and inference is produced by only a handful of large commercial companies that dominate the market. The high cost of specialized hardware, coupled with a lack of competition, poses a financial barrier for academics that could potentially hinder the democratization of ML research. Censorship in models and training data is also a concern. Entities beyond the influence of citizens make decisions about what answers are acceptable, introducing potential bias and curtailing open inquiry.

Another challenge involves the difficulty in debugging and profiling ML models, especially in the context of prompt injection. Unlike traditional code, debugging models and addressing prompt injection present significant challenges due to the inherent nature of these models as black boxes. Solving these fundamental problems is key to unlocking the true potential of large language models.

A collective effort from the scientific community and government is needed to make ML work for the scientific community. A call to action could include funding a foundational model, ensuring that hardware and software development align with broader goals, and investigating the use of open-source models and infrastructure in shaping the future of ML.

AI and Software Quality Enhancement

There are many ways that AI-enhanced tools can empower developers to confidently enhance or fix existing software, and the recent surge in data-driven ML is contributing to the development of tools that provide a practical means for software quality improvement. AI-enhanced tools could be particularly useful in the realm of automatic program repair, which seeks to seamlessly modify a program by fixing its defects while preserving its core functionality. Correctness is paramount, which means any AI interventions would have to align with the underlying semantics of the software.

Existing program-repair techniques can be classified as heuristic-based (i.e., based on cognitive science, or the study of how humans think) or semantics-based approaches (i.e., based on the use of different machine learning and logic-based approaches), and the two are not mutually exclusive. A third separate category could be developed for the use of AI-enhanced approaches: learning-based program repair. The integration of powerful heuristics opens avenues to address longstanding software quality problems. While ML provides many benefits for automatic code repair, it provides an augmentation of heuristics rather than a categorical shift, reinforcing the need for a symbiotic relationship between ML approaches and the techniques used to repair or improve the software. Despite the evolving landscape, AI's role in this area is not exactly revolutionary. It is, however, transformative, allowing the exploration of new problems with enhanced tools. The future holds further promise for novel solutions in software quality improvement as AI continues to augment, rather than replace, traditional techniques.

Trust in AI-Assisted Development

Many factors contribute to the trust developers place in AI-assisted tools. A comprehensive framework derived from interviews and surveys highlights key interplaying factors that developers consider when deciding whether to trust the tools they use. Most of these factors can be categorized as follows: personal factors, interaction factors, control factors, system properties, and expectations.

Survey findings also indicate that consistency and meeting expectations are paramount for trust in both AI and non-AI tools. However, developers using AI-assisted tools seem to prioritize validation support, autonomy, and source reputation more than their counterparts using traditional tools.

The landscape of AI-assisted development tools and the intricate web of trust woven between developers and their technological counterparts seem to be growing quickly. Survey respondents expressed a desire for increased AI support throughout the development life cycle, which seems to emphasize the growing trust developers place in AI technologies. This serves as a testament to a transformation in the developer community, where AI is not just a tool but a collaborative partner.

Session III: Software Engineering Research Areas and Gaps

The third session of the workshop focused on identifying software engineering research areas and gaps through the lens of both academia and industry. The overarching theme centered on revisiting the foundational software engineering principles that underly AI engineering. In that context, participants explored key industry trends, solutions, and emerging capabilities and identified transformative R&D in software engineering.

During this session, led by Anita Carleton, director of the Software Solutions Division at the SEI, speakers were asked to address the following questions:

- What are your proposals for the transformational R&D thrusts that will help engineer the software- and AI-intensive systems of the future?
- What are key industry trends, solutions, and emerging capabilities?

Navigating the Landscape of AI and LLMs for Research

Increasing the diversification and transparency of AI models presents an urgent challenge. A landscape where various models, data sources, and hardware configurations coexist is necessary for research institutions to properly use AI and LLMs. A proactive stance should be used to ensure access to diverse AI technologies, while also emphasizing the importance of transparency and data quality control.

One approach is exploring the concept of a "software genome," which refers to identifying fundamental building blocks within code, independent of programming languages. The Software Genome Project[4] seeks to redefine the unit of computation, providing a more abstract and universal approach to code generation. This move towards a new paradigm aims to bridge the gap between the low-level, compiler-centric view and the high-level abstraction, fostering a more intuitive understanding of AI-generated code.

Impact of AI and LLMs on Existing Software Ecosystems

Although the conventional view of software defines it as an asset, software could also be considered a liability. If software is primarily a tool for solving business problems, a rush within the industry to generate more software using LLMs and AI techniques raises concerns about the potential liabilities created in the process.

The coexistence of AI-generated and human-generated software systems in a hybrid world is almost inevitable, but there are challenges to integrating these types of systems. Like autonomous vehicles and airplanes, AI-generated systems need to adapt to diverse and complex environments in an intricate software ecosystem.

[4] Inspired by the Human Genome Project, the Software Genome Project treats the software source code as software DNA and is geared toward the secure monitoring and exploitation of open-source software. See https://arxiv.org/pdf/2311.09881.

AI and ML in software engineering might end up having the greatest impact on the evolution and improvement of existing systems rather than on the creation of entirely new ones. LLMs have significant potential for finding and fixing issues in large-scale, deployed code bases. There are many challenges to implementing automated code review processes for AI-generated changes, including policy implications related to unilateral access to source code repositories, potential IP leakage, and resource availability for small companies without as many financial options.

Potential challenges also include the equity concerns faced by underrepresented groups in technology who encounter limited opportunities to interact with AI systems. The adoption of AI tools in academia might inadvertently exacerbate existing disparities. Balancing innovation with ethical considerations, policy implications, and equity concerns becomes paramount in shaping a future where AI and LLMs coexist harmoniously with human-generated software systems.

Navigating Software and AI Realities in Critical Applications

The intersection between software engineering and AI presents many complexities for critical applications, particularly in the realms of defense and intelligence. Trustworthiness is vital to mission-critical software development, yet evolving hardware and firmware landscapes pose many new challenges.

Systems are becoming increasingly complex as they are driven by more resources, sensors, data, analytics, and AI. As the interconnectivity of systems grows, the premium on resiliency becomes paramount. Systems need to remain trustworthy in the face of internal attack surfaces, evolving IoT frameworks, and the demand for faster-than-human-thought operations. This exponential growth in system complexity must be matched with the continuous improvement of abstractions and tools. The evolving hardware landscape presents challenges as well, including the plateauing of Moore's Law and Dennard's Law.

Technical models and analyses must be incorporated into engineering data, including the use of technical models as engineering evidence. Evidence management and the explicit evaluation of architectures for resiliency are essential cultural shifts in software engineering. The convergence of mission, software, and hardware realities calls for a proactive and adaptive approach in AI research to ensure the development of trustworthy systems in critical applications.

Research Needs

Software and AI capabilities are advancing rapidly around the world, and not just in high-resource nation states. They will continue to advance in complexity and sophistication, without bound, for the foreseeable future. To bolster U.S. leadership in this incredibly competitive domain, a focus on research breakthroughs and development is needed in software engineering and AI engineering, system architectures, and defining trustworthy systems.

Academia, industry, and the federal space need collaboration mechanisms. One way to enhance collaboration is to invest in operationally relevant datasets and testbeds. Likewise, open access to resources in software engineering, such as models and data sets, is needed, along with the ability to break down large models into smaller pieces for better understanding and progress. Social factors, access, and soft skills in AI are also important. The following list provides an overview of important and necessary areas of research.

- **Software architectures for modern software needs.** Architectures for AI-based systems should be developed so that they are resilient to attack and support federated data sources. The development of modeling and analysis techniques is needed to guide early design decisions, facilitate downstream testing and evaluation (T&E), and enable evidence creation.

- **AI engineering practices for trustworthy use of ML and LLM capabilities.** Research is needed to enable the development of trustworthy systems to mitigate weaknesses in ML and LLMs and support ongoing updates to ML- and LLM-based capabilities as algorithms and training improve.
- **Data-intensive software engineering.** Software repositories contain a wealth of information regarding current and older projects. There is a need to support repository mining for defect repair, API compliance, refactoring, synthesis, transformation, and evidence-based T&E. Data federation, privacy protection, and multi-institutional data collaboration represent important challenges in integrating various types of data, such as health and environmental data.
- **Diverse, advanced technical models and analyses to support development and evolution.** The use of modeling and analysis is essential in modern practices of software engineering. Modeling and analysis must be integrated into practice in a way that allows for a diversity of tools. More robust code models must be built by considering different code properties such as syntax, semantics, and evolution, and incorporating them into the model's design and loss functions.
- **Cybersecurity considerations for AI-reliant and software-reliant systems.** Systems are growing in interconnection and complexity, with larger external and internal attack surfaces, including AI attack surfaces. A focus on cyber risk is needed, including how to measure and manage attack surfaces as threats continue to grow in sophistication and scale. Architectures devised for security and resiliency are needed, as well as models and tools to enhance cybersecurity.
- **Clear standards and guidance.** There is a need for clarity in the development of standards for AI systems, as they are often asked to meet a large and varied number of requirements related to trustworthiness, security, privacy, and ethical considerations.

Critical Needs and Priorities: Five Primary Themes

Throughout the workshop, discussion and reflection on the rapid acceleration of new technologies in the software development lifecycle and the role of AI in shaping the future of software systems were paramount. The following five main themes emerged from discussions of the critical need for new approaches to navigate both the opportunities and challenges of these topics.

Theme 1: AI is transforming the software engineering process and how we engineer software systems. The increasing symbiosis of humans and machines is transforming every phase of the software development lifecycle.

In software engineering, we are witnessing the emergence of a symbiotic workforce, where autonomous, intelligent assistants will work with software engineers to develop systems. This revolution in the way we approach software development will reshape the entire lifecycle, giving rise to approaches that promise to in the lifecycle, and software engineering principles should serve as a foundation for the development, evolution, and evaluation of AI-enabled software. The use of AI will likely make it possible to automate much harder programming and software quality problems. While we recognize that tasks, skills, and tools will inevitably undergo transformation in this new paradigm, the specifics are not yet fully evident.

Current technological advances, especially those related to AI and ML tools, will fundamentally alter how applications are built, from design-to-code platforms and tools to ML models that automatically generate code and automate elements of application testing. ML-generated code is already in commercial codebases, and the overall percentage is already rapidly growing.

In fact, the experimental application of LLMs shows promise across the entire lifecycle. Effective application of LLMs may enable the ultimate "shift left" approach, where tasks that are traditionally done at a later stage of the process, such as testing or performance evaluation, can be done early, often before any code is

written, or incorporated effectively throughout software development. Design-to-code platforms and tools could make it easier for developers to bring their ideas to fruition as models automatically generate code and streamline repetitive coding tasks. Leveraging advanced automation techniques, including AI- and LLM-enabled capabilities for everything from coding and code review to deployment at scale, integration testing, and debugging could streamline workflows, improve code quality, and accelerate the development cycle. Research exploring how to apply LLMs is only in its early phases, however, and many potential issues must be addressed, including the following:

- A substantial number of solutions have been trained on a single proprietary data source or on proprietary algorithms, and as a result it is not clear how robust their inferences and conclusions are.
- Filtering issues can make conclusions hard to replicate, especially since it is not always clear what kind of filtering has been done. Some models are trained on data that specifically omits some knowledge, and, in other instances, the companies that own the models decide to censor some results.
- More diversity in models, systems, and applications is needed, and the research community should not put too much trust in a single model. Public funding might help address this issue by generating models and software and hardware infrastructures that remove the proprietary or black-box decision-making that influences results.
- Given the speed with which innovations can be developed in this space, the software research community has become increasingly focused on quick prototypes as opposed to long-term, systematic research.
- Most effective techniques will likely be based on hybrid solutions, that is, a combination of LLM, other AI, and data-driven automation approaches. Investigations of hybrid solutions should be accelerated.

While these new technologies promise to bring many benefits, they also have to the potential to quickly multiply negative effects, such as security problems and AI debt (i.e., the cost of the complex mix of processes and procedures needed to discover, train, and deploy predictive models that are accurate and dependable). We need to develop sound and empirically based methods now for determining what approaches to consider successful and how to guide future software development lifecycle optimizations. Moreover, successful integration of AI in software development also relies on many nontechnical factors, including the need for a "smart assistant" to understand team dynamics and roles and respond appropriately to human interactions and needs.

Theme 2: Generative AI has reached a level of sophistication that may seem to resemble human intelligence, but it remains difficult to determine the level of trust that should be placed in its outputs.

Assuring mission- and safety-critical cyber-physical systems (CPS) has become increasingly challenging due to their growing complexity. The introduction of AI elements further compounds these difficulties because such elements can create large bodies of new code quickly, complicate the understanding of system behavior, and introduce new attack vectors, including the poisoning of training data and prompt injection, in which AI prompts can include code to generate pernicious behaviors.

As a result, while it is already clear that generative AI can make software developers more productive in terms of producing code, there are well-founded worries about that code's quality and sustainability. These new AI tools may already be producing a huge wave of technical debt that could overwhelm downstream software engineering efforts. In some studies, generative AI tools such as LLMs regurgitated old defects as often as they produced good fixes. Novice developers may lack the expertise to understand the limitations of the code that LLMs and other AI tools produce. AI-produced code will coexist alongside human-built code for a long time. We have few options to help end users and developers decide whether to trust code generated by tools, and how this should compare to trust in human-written code. Do we trust an AI tool more or less than a human, even if humans may make more mistakes? Where do we address trust – in the

ML models themselves, in software engineering, in testing, in how users interact with the system, or all of the above?

Research has already begun on identifying the factors that can increase software developers' trust in AI tools. Key factors include source reputation; interaction (e.g., providing validation support and feedback loops); control (e.g., degree of ownership and autonomy); system features (e.g., ease of installation and performance measures); expectations (e.g., how well the tool fits the developers style or goal). "Explainability" is not a proxy for "trustworthiness." By their nature, many of our AI systems cannot cogently explain why they arrive at their conclusions.

One goal for improving trust should be to increase our ability to build trustworthy systems out of untrusted components. A second goal that researchers should explore involves how to adopt AI to generate evidence about a resulting system that can be independently verified (analogous to the development of proof-carrying code, or AI-generated code that comes with its own evidence). Yet another aspect of trust that requires research is whether AI tools leak intellectual property. For example, it's possible that a model that learns on a proprietary codebase could then recommend pieces of that codebase to inappropriate users.

Today, we don't trust AI tools, but we don't always trust humans either. Rather than focusing on making AI trustworthy, we could use it to help us increase trust, using techniques such as generating evidence and incorporating AI into software testing and reviews.

Data assurance is another new frontier in the assurance of AI. In fact, it is one of the key components that makes assurance hard for AI, given the difficulty of understanding how data affects the final behavior of the system. The scalability of assurance for large AI models also poses a significant hurdle. Although some verification techniques have improved, the rapid increase in model size outpaces these approaches, which can render current verification methods inadequate from the outset.

Theme 3: It is imperative to redefine the discipline of software engineering to encompass the use of new technologies (including, but not limited, to generative AI), and to rethink the curricula, tools, and technologies associated with its practice. This effort is key to designing, building, evolving, and evaluating trustworthy software systems in a responsible, ethical way.

Redefining the software engineering discipline with AI is leading toward a revolution that changes how engineering solutions are explored, systems are built, and AI aids in the operation of systems. Education is a crucial aspect of any transformation brought about by AI, and there is a need for new degrees and curricula that incorporate AI into various engineering disciplines.

To keep up with the rapid advancement of AI technologies, software engineering curricula must include instruction on both the application of AI in the software engineering lifecycle and on how tools can facilitate the design, development, training, testing, and authorization of AI-enabled software. This evolution of software engineering curricula, both at the undergraduate and graduate levels, requires a dynamic component to ensure that the workforce is well equipped to effectively use these tools to support the development lifecycle.

Educators must take care to make curricula equitable. As AI tools start to be used in software classes, initial observations indicate that groups that are underrepresented in technology disciplines are also less comfortable using these technologies. Educators need to consider this factor to avoid creating an environment where people with access to AI tools have clear advantages while groups without equitable access get left behind.

Retaining talent in academia is also a concern. PhD students and faculty often face financial challenges due to the demanding nature of research and the need to secure funding. Efforts to make PhD programs more attractive, reduce funding restrictions, and provide sustained funding can help address these issues. The cost

of undergraduate education is also a significant concern. Government involvement to address the educational system's challenges can contribute to producing a workforce that is better equipped to address the nation's challenges effectively.

Enhancing the fluidity between academia and other sectors can promote knowledge exchange. Incentivizing collaboration between universities and industry is crucial to address important research needs effectively. Key elements in fostering such collaboration include establishing public-modeled problems, data repositories, and testbeds to facilitate joint research efforts. Government agencies can also play a role by effectively using commercial solutions and services where they prove beneficial and identifying bottlenecks that hinder progress.

Theme 4: New technologies, including generative AI, seem to hold the promise of making almost everyone a programmer. As a result, AI literacy and the development of new skills are needed throughout the workforce.

The landscape of programming is evolving dramatically. Today, everyone can become a "programmer," not just those with traditional technical skills and expertise in software, systems, and AI engineering. To be successful in this emerging arena, new skills and abilities must be cultivated across a much wider range of people. These new skills and abilities include problem solving, critical thinking, and a general understanding of AI and ML.

The skills needed by professionally trained software programmers and engineers will also shift. While many traditional software engineering skills will likely become less valuable in the face of AI tool capabilities, the value of the remaining skills may increase dramatically. For example, research by Microsoft on their Copilot tool that generates code via LLMs indicates that users need to spend less time writing code, but more time understanding and reasoning about code.

Software engineers will need a firm grasp of probabilistic reasoning to deal with uncertainty; an increased capacity to detect problems and make informed design decisions; strong systems thinking skills; and a keen awareness of the ethics of AI. The discipline of prompt engineering is beginning to gain traction, which involves programming in natural language and has potential applications in various stages of software development. Different prompts given to code models result in the generation of different code, highlighting the challenge of obtaining trustworthy output from these models.

Moreover, the potential impact on society and the economy of using AI in software systems necessitates that decision-makers and leaders in all domains comprehend the fundamental principles of AI and have the competence to ask the critical questions that enable the safe development and responsible use of these tools. New initiatives are needed to provide training, workshops, and resources to ensure that individuals in positions of influence and authority are equipped to make informed decisions regarding AI technologies and their applications. By empowering leaders with AI literacy, we can foster the responsible and beneficial integration of AI into our lives.

Theme 5: The use of AI tools, such as LLMs, can mask the trade-offs being made between the functionality of software systems and their safety and security. Research is needed to identify and make explicit the key engineering trade-offs being made during the design, development, training, testing, and authorization of systems that include AI components.

Trust, trustworthiness, and confidence in software systems that include AI or are developed with AI components are top priority considerations. To achieve trustworthiness, engineers must navigate key trade-offs in system development to ensure the system performs as intended without overstepping its boundaries. This trust should endure as the system inevitably changes over time, providing measurable confidence in the system's evolving performance. Research is essential to enable this outcome by providing mechanisms

for identifying engineering trade-offs throughout the specification, design, training, testing, and authorization processes of critical systems.

Explicit trade-offs that set limits on AI systems are also needed to address concerns for both direct users and others potentially impacted by the system's actions or data. Although technologies like ChatGPT currently implement some features that prevent harm at the expense of performance, explicit engineering trade-offs are needed during system development to clarify the relationship between functionality and safety and security. Research in AI-enabled systems must identify and analyze these trade-offs explicitly to maintain safety and security throughout the software engineering lifecycle.

Additionally, AI-enabled tools should be designed to explicitly show the trade-offs involved in developing a system instead of obfuscating or concealing them from key decision-makers. Making engineering trade-offs transparent to decision-makers is especially critical to ensure the development of robust and trustworthy systems when incorporating technologies like smart coding assistants.

About the Authors

The Software Engineering Institute (SEI) at Carnegie Mellon University is a federally funded research and development center (FFRDC)—a nonprofit, public–private partnership that conducts research for the United States government. One of only 10 FFRDCs sponsored by the U.S. Department of Defense (DoD), the SEI conducts R&D in software engineering, systems engineering, cybersecurity, and many other areas of computing, working to introduce private-sector innovations into government.

The SEI works with partners throughout the U.S. government, the private sector, and academia. These partnerships enable it to take innovations from concept to practice, closing the gap between research and use.

The NITRD Program is the nation's primary source of federally funded work to pioneer information technologies in computing, networking, and software. The NITRD Subcommittee of the National Science and Technology Council's Committee on Science and Technology Enterprise guides the multiagency NITRD Program in its efforts to provide the R&D foundations for ensuring continued U.S. technological leadership that meets the nation's advanced IT needs. The National Coordination Office (NCO) supports the NITRD Subcommittee and the Interagency Working Groups (IWGs) and teams that report to it. The NITRD Subcommittee's Co-Chairs are Kamie Roberts, NCO Director, and Margaret Martonosi, Assistant Director of the NSF Directorate for Computer and Information Science and Engineering. More information about NITRD is available online at https://www.nitrd.gov/.

The NITRD Software Productivity, Sustainability, and Quality (SPSQ) IWG coordinates federal R&D to achieve orders-of-magnitude reduction in software defects and the time and cost of developing and sustaining software. The U.S. government and the national economy depend on increasingly complex software; improved software development technology is essential to U.S. innovation, to leadership in emerging technologies,

Appendix A: Remarks from Ms. Kamie Roberts

Good morning, I am Kamie Roberts, the Director of the NITRD National Coordination Office, and co-chair of the NITRD Subcommittee of the National Science and Technology Council. I want to welcome everyone to this event and extend a big thank you to our outstanding speakers who will provide the compelling background for a rich discussion on how software capabilities and artificial intelligence have a direct impact on the future of our nation.

First, a little background on NITRD. We are a multiagency program that promotes innovations in pivotal emerging technologies to ensure the U.S. maintains its leadership in IT. For the past three decades, NITRD has been at the frontier of computing, networking, data, and software that has led to many breakthroughs, from weather modeling to agriculture, from clean energy to advancing the understanding of human diseases like COVID-19. IT is everything, everywhere all at once.

The Software Productivity, Sustainability, and Quality (SPSQ) Interagency Working Group (IWG) was formed in 1991 to coordinate Federal SPSQ R&D across 19 participating agencies to achieve orders-of-magnitude reduction in software defects and the time and cost of developing and sustaining software. The U.S. government and the national economy depend on increasingly complex software; improved software development technology is essential to U.S. innovation, to leadership in emerging technologies, and to security and prosperity.

SPSQ R&D advances timely and affordable development and sustainability of low-defect, low-vulnerability software; this includes R&D to improve software development productivity, quality, measurement, assurance, and adaptability, while also providing essential characteristics such as security, privacy, usability, and reliability.

The NITRD SPSQ co-chairs are Sol Greenspan, Program Director Computing & Information Science and Engineering (CISE) at the National Science Foundation, and Ram D. Sriram, Chief of the Software and Systems Division, Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST).

NITRD's IWGs are committed to public-private partnerships through various forms of community engagement and public outreach. This commitment to coordination, collaboration, and transparency has contributed to growing participation and increasing R&D funding in the program's pursuit of NITRD innovation and public access to it. In real time, this commitment is currently carried out through the fundamentals of this workshop.

This workshop comes at a pivotal time in our nation where software is vital to America's global competitiveness, innovation, and national security. It is undoubtedly true that the economy, the nation's infrastructure, education, and healthcare rely heavily on software capabilities, while artificial intelligence is of particular importance. As a result of this reality, software and artificial intelligence engineering continue to receive national attention.

National Priorities on the Radar

The National AI Initiative Act of 2020 provides for a coordinated program across the entire federal government to accelerate AI research and application for the nation's economic prosperity and national security. The mission of the National AI Initiative is to ensure continued U.S. leadership in AI research and development, lead the world in the development and use of trustworthy AI in the public and private sectors, and prepare the present and future U.S. workforce for the integration of AI systems across all sectors of the economy and society. On May 4, this year, the Biden-Harris administration announced new actions that will further promote responsible American innovation in AI and protect people's rights and safety. These steps build on the administration's strong record of leadership to ensure technology improves the lives of the American people and break new ground in the federal government's ongoing effort to advance a cohesive and comprehensive approach to AI-related risks and opportunities.

Through AI, the U.S. has made significant strides in healthcare, energy, transportation, and space exploration. In addition, software engineering and AI have created huge economic benefits for the country. Countless tech startups and companies have emerged as a result of software and AI innovations, creating jobs and contributing to economic growth. Major tech companies, such as the few that are here to name, IBM, Microsoft, Google, Apple, and Facebook, have all leveraged AI and software engineering to build highly profitable businesses that have helped drive the U.S. economy forward. Overall, software engineering and AI have been crucial to the U.S.'s economic growth, technological advancements, and global competitiveness.

This workshop is a collaborative and dynamic meeting of thought leaders. Our academic researchers will explore the groundwork needed to engineer the software and AI intensive systems of the future, the step-by -step progress being made across the critical domains, while identifying key industry trends and proposing solutions that will shed light on emerging capabilities.

Our federal leaders will dive into the future capabilities of AI and software enabled systems, discuss the R&D needed to address the gaps and risk areas, and how the capabilities support topics of national interest.

Last, but certainly not least, our industry stakeholders will address the software engineering lifecycle and evaluate where AI is mature enough for use while also being reliable and trustworthy.

This exciting workshop will create a truly unique environment to foster innovation and progress in software and AI engineering. I am so excited about this gathering of experts where a profound impact can be made on the objectives of this workshop that lays the foundation for research and development in the near- and long-term R&D in this focus area.

Appendix B: Attendee List

Fraol Batole
Nadim Bodair
Yuanfang Cai
Linda Canon
Anita Carleton
Lynn Robert Carter
Jeffrey Carver
Ruth Cheng
Mike Conway
Robert Crumbley
Dionisio de Niz
Prem Devanbu
Allan Dianic
Erica Dretzka
Sebastian Elbaum
Michele Falce
Kyle Fox
Sol Greenspan
Sandeep Gupta
Erin Harper
Anthony Harris
Amy Henninger
John Hurley
Sayem Mohammad Imtiaz
Clemente Izurieta
Brittany Johnson-Matthews
Daniel Justice
Ron Koontz
Wing Lam
Claire Le Goues
Rick Linger
Thomas Longstaff
Trini Lopez
Michael Lowry
Andrian Marcus
Collin McMillan
Erik Meijer
Tim Menzies
Kevin Moran
Paul Nielsen
Olusola Odeyomi
Brigid OHearn
Vadim Okun
Ipek Ozkaya
Denys Poshyvanyk
David Rabson
Mona Rahimi
Hridesh Rajan
Baishakhi Ray
John Robert
David Rosenblum
Gardy Rosius
William Sanders
Bill Scherlis
Douglas Schmidt
Arun Seraphin
Greg Shannon
Robert Shemeley
Forrest Shull
Ram D. Sriram
Kathryn Stolee
Mohammad Wardat
Shiyi Wei
Bill Wilson
Hyrum Wright
Eileen Wrubel
Nelson Yang
Ziyu Yao
Xiangyu Zhang