FIVE CRITICAL CHALLENGES FOR SOFTWARE AND AI ENGINEERING

SOFTWARE ENGINEERING INSTITUTE (SEI),
SOFTWARE SOLUTIONS DIVISION

Introduction

Advances in software engineering and artificial intelligence (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. To inform a community strategy for building and maintaining U.S. leadership in software engineering and AI engineering, the Software Engineering Institute (SEI) and the Networking and Information Technology Research and Development (NITRD) Program in the White House Office of Science and Technology Policy co-hosted a workshop at the National Science Foundation on June 20-21, 2023.

The event gathered thought leaders from federal research funding agencies, research laboratories, mission agencies, and commercial organizations to explore the fundamental research needed to support progress toward this goal. The workshop used the SEI's *Architecting the Future of Software Engineering: A National Agenda for Software Engineering Research and Development* (see further readings) as a starting point because the areas of focus identified in the study have been confirmed as even more critical and urgent, particularly due to the rapid advances of generative AI in the two years since its release. Specifically, three research areas from the study were identified by participants 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.

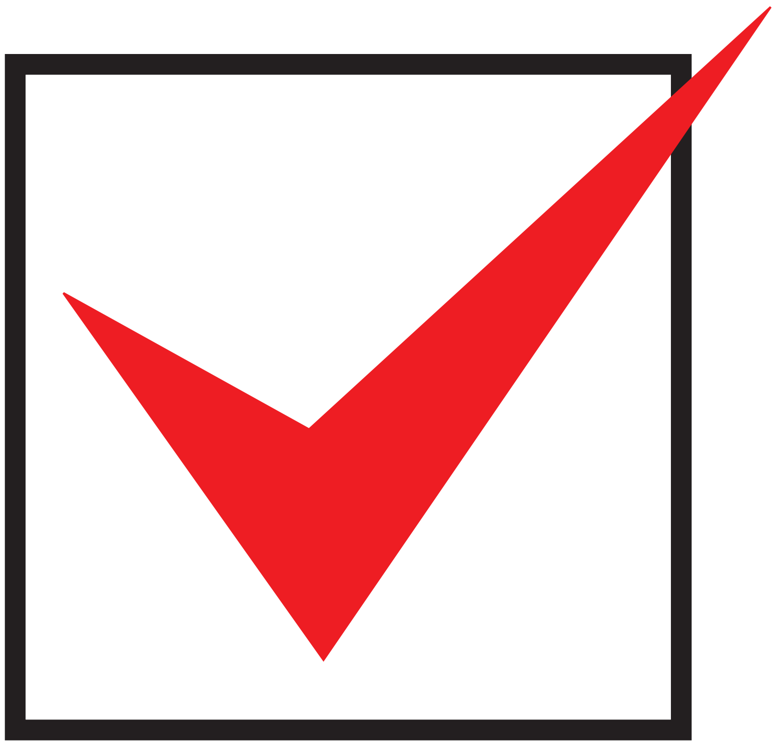
Workshop Goals and Motivation

The workshop organizers brought together participants to encourage new partnerships that will advance U.S. leadership and national interests through the disciplines of software and AI engineering, and positively impact progress across virtually all scientific domains. Specific objectives for the workshop included:

- 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 the CMU SEI National Agenda for Software Engineering National Study & Roadmap.

Executing and advancing the closely related disciplines of software engineering and AI engineering are indispensable to our ability to develop and deploy intelligent software systems effectively and rapidly. While the engineering of AI capabilities has unique and challenging requirements, these capabilities are implemented in software. To date, there has been significant research within software engineering on technologies and practices needed to build such AI-enabled systems with confidence. While comparatively more recent, the fundamental theories, practices, and knowledge base for AI engineering are receiving significant research attention 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 exploiting large language models 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 to develop top priority capabilities are also changing. The rapidly changing technical environment creates further urgency to prioritize areas of most critical need and allocate multidisciplinary resources to the most challenging and essential areas of concern.



Critical Needs and Priorities: Five Primary Themes

In keynote speeches, breakout sessions, and lightning talks, 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 the critical need for new approaches to navigate both the opportunities and the challenges was discussed, five main themes emerged.

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 enhance productivity, quality, and efficiency. Software engineering should utilize AI tools and technology 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 machine learning (ML) tools, will fundamentally alter the ways in which applications are built – from design-to-code platforms and tools, to ML models that automatically generate code, to models that 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 large language models (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 test, 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/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 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 non-technical factors, including the need for a "smart assistant" to understand team dynamics and roles and respond appropriately to human interactions and needs.

While generative AI has reached a level of sophistication that may seem to resemble human intelligence, it is considerably harder to determine the level of trust that should be placed in the outputs.

The assurance of mission- and safety-critical cyber-physical systems (CPS) has become increasingly challenging due to the growing complexity of these systems. The introduction of AI elements further compounds these difficulties because they 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 the quality and sustainability of the code produced. 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 regurgitated old defects as often as they produced good fixes. Novice developers may lack the expertise to understand the limitations of the code being produced. AI-produced code will co-exist alongside human-built code for a long time. We have few options to help end users and developers decide whether or not 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 the 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., validation support and feedback loops provided); control (degree of ownership and autonomy); system features (e.g., ease of installation and performance measures); expectations (e.g., how good of a fit is the tool for style/goal of the developers). "Explainability" is not a proxy for "trustability." By their nature, many of our AI systems cannot explain cogently why they arrive at their conclusions.

One goal should be increasing our ability to build trustable systems out of untrusted components. A second goal to explore is adopting AI to generate evidence about a resulting system that can be independently verified (e.g., analogous to the development of proof-carrying code, or AI-generated code that comes with its own evidence). Another aspect of trust that requires research is whether AI tools leak intellectual property. It's possible a model might learn on a proprietary codebase and then recommend pieces of that codebase to inappropriate users. Today we don't trust AIs – 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.

Redefining the discipline of software engineering to encompass the use of new technologies (including but not limited to generative AI) is imperative, along with rethinking the associated curricula, tools, and technologies. This effort is key to designing and 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 effort brought about by AI, with new degrees and curricula incorporating 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 in supporting the development lifecycle.

Care must also be taken to make curricula equitable. Some initial observations as AI tools start to be used in software classes indicate that groups that are under-represented in technology disciplines are also less comfortable using these technologies. Factors such as this should be considered to avoid creating an environment where people with access to AI tools have clear advantages and other 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 in addressing the educational system's challenges can contribute to producing a workforce better equipped to address the nation's challenges effectively.

Enhancing 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 utilizing commercial solutions and services where they prove beneficial, identifying bottlenecks that hinder progress.

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. Instead of relying solely on those with traditional technical skills and expertise in software systems, and AI engineering, new tools promise to enable almost everyone to become a "programmer." For this approach to be successful, new skills and abilities must be cultivated across a much wider range of people. These new skills and abilities include problem solving and 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 given AI tool capabilities, the value of the remaining skills may increase dramatically. For example, research results from Microsoft about their Copilot tool that generates code via LLMs indicate 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 uncertainty and probabilistic reasoning, 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 be competent in asking the critical questions to enable their trustworthy development and responsible use. Initiatives can be launched 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 in our lives.

The use of AI tools such as LLMs can mask the tradeoffs being made between the functionality of software systems and their safety and security. Research is needed to identify and make explicit the key engineering tradeoffs 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 or are developed using AI components are top priority considerations. To achieve trustworthiness, engineers must navigate key tradeoffs in system development, ensuring the system performs as intended without overstepping its boundaries. This trust should extend 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 tradeoffs throughout the specification, design, training, testing, and authorization of critical systems.

Explicit tradeoffs 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 tradeoffs are needed during system development to clarify the relationship between functionality and safety/security. Research in AI-enabled systems must identify and analyze these tradeoffs explicitly to maintain safety and security throughout the software engineering lifecycle.

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

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, participants at the workshop identified a need to focus on research breakthroughs and development in software engineering and AI engineering, system architectures, and defining trustable systems. Presentations and discussion from multiple federal agencies showed the extent to which their plans for executing their missions rely on advanced software and AI capabilities.

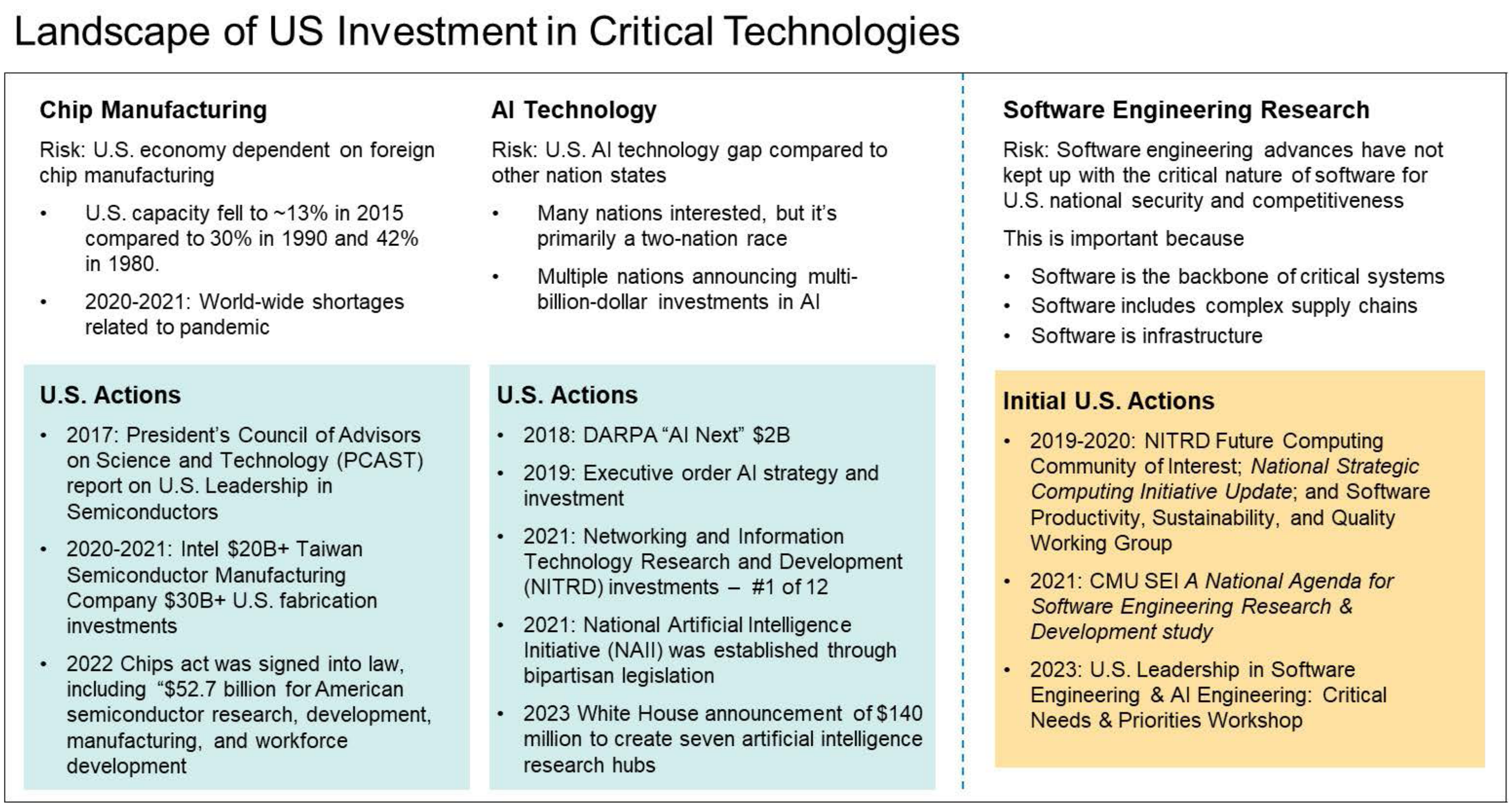
Workshop participants also discussed the importance of improving collaboration mechanisms among academia, industry, and the federal space, including suggestions to invest in operationally relevant datasets and testbeds to enhance collaboration. Likewise, participants highlighted the need for open access to resources, such as models and data sets, in software engineering and the importance of breaking down large models into smaller pieces for better understanding and progress. The significance of social factors, access, and soft skills in AI and the importance of taking a multi-disciplinary approach were also acknowledged. The high priority themes identified also revealed a significant need for intentional crosscutting progress in data, standards, and all tradeoffs and aspects of trust. Specific areas of needed research discussed included:

- 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 test 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 have 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 are important challenges in integrating various types of data, such as health and environmental data.
- Diverse, advanced technical models and analyses to support development, evolution, and T&S. The use of modeling and analysis is essential in modern practice. 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 since threats are growing 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.

Conclusion and Next Steps

This workshop delved into various aspects of software and AI engineering, addressing challenges, opportunities, and ethical considerations. It highlighted the paradigm shift brought about by AI and large language models, requiring alignment between models, researchers, and diverse user groups. Participants emphasized the need for transparency, trustworthiness, and collaboration across different sectors to effectively navigate the evolving landscape of AI technology.

The workshop also highlighted the impact of AI on various domains, including workforce, cybersecurity, and autonomous systems, and the importance of collaboration and engagement with stakeholders was emphasized. The growing influence of AI in society, along with the acceleration of technology in general, demands interdisciplinary collaboration, technical advocacy for broader use cases, and


Landscape of US Investment in Critical Technologies

Chip Manufacturing

Risk: U.S. economy dependent on foreign chip manufacturing

- U.S. capacity fell to ~13% in 2015 compared to 30% in 1990 and 42% in 1980.
- 2020-2021: World-wide shortages related to pandemic

U.S. Actions

- 2017: President's Council of Advisors on Science and Technology (PCAST) report on U.S. Leadership in Semiconductors
- 2020-2021: Intel $20B+ Taiwan Semiconductor Manufacturing Company $30B+ U.S. fabrication investments
- 2022 Chips act was signed into law, including "$52.7 billion for American semiconductor research, development, manufacturing, and workforce development

AI Technology

Risk: U.S. AI technology gap compared to other nation states

- Many nations interested, but it's primarily a two-nation race
- Multiple nations announcing multi-billion-dollar investments in AI

U.S. Actions

- 2018: DARPA "AI Next" $2B
- 2019: Executive order AI strategy and investment
- 2021: Networking and Information Technology Research and Development (NITRD) investments – #1 of 12
- 2021: National Artificial Intelligence Initiative (NAII) was established through bipartisan legislation
- 2023 White House announcement of $140 million to create seven artificial intelligence research hubs

Software Engineering Research

Risk: Software engineering advances have not kept up with the critical nature of software for U.S. national security and competitiveness

This is important because

- Software is the backbone of critical systems
- Software includes complex supply chains
- Software is infrastructure

Initial U.S. Actions

- 2019-2020: NITRD Future Computing Community of Interest; *National Strategic Computing Initiative Update*; and Software Productivity, Sustainability, and Quality Working Group
- 2021: CMU SEI *A National Agenda for Software Engineering Research & Development study*
- 2023: U.S. Leadership in Software Engineering & AI Engineering: Critical Needs & Priorities Workshop

Figure 1. *Investment in U.S. Software Technology.*

policy development informed by the research community.

Making investment decisions in the right technical domains and fostering powerful partnerships is key to meeting the critical needs and priorities of the U.S. for software and AI engineering. For example, the figure above shows the actions taken to avoid the risks of a U.S. economy dependent on foreign chip manufacturing, which industry investments of around $50 billion and a proposed government investment of another $50 billion. AI technology investment followed a similar path, where a possible U.S. technology gap motivated major government and industry investment. The increasing awareness of the risks to national security and the U.S. economy motivated action in those cases, and such concerns also underscore the importance of making a similar strategic investment in software engineering research.

Copyright

Further Readings

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

https://www.sei.cmu.edu/publications/annual-reviews/2022-year-in-review/year_in_review_article.cfm?customel_data-pageid_315013=493993

https://insights.sei.cmu.edu/library/envisioning-the-future-of-software-engineering/

https://insights.sei.cmu.edu/news/to-lead-software-and-ai-engineering-us-faces-five-critical-needs-says-workshop-summary/

About the Authors

Anita Carleton is an Executive Leadership Team Member and Division Director of the Software Solutions Division at the Carnegie Mellon University Software Engineering Institute with more than 35 years of senior leadership experience. She has most recently led a national study titled "Architecting the Future of Software Engineering: A National Agenda for Software Engineering Research & Development." Carleton serves as the chair of the IEEE Software Advisory Board. Carleton received her MBA from the MIT Sloan School of Management where she was the recipient of the MIT Sloan Leadership Fellowship.

Anita Carleton

Division Director

Software Engineering Institute

adc@sei.cmu.edu

Dr. Doug Schmidt is the Cornelius Vanderbilt Professor of Engineering, Associate Chair of Computer Science, and a Senior Researcher at the Institute for Software Integrated Systems, all at Vanderbilt University. He is also a Visiting Scientist at the Software Engineering Institute at Carnegie Mellon University. Dr. Schmidt is an internationally renowned and widely cited researcher whose work focuses on pattern-oriented middleware, Java concurrency and parallelism, and generative AI.

Dr. Doug Schmidt

Professor of Engineering

Vanderbilt University

d.schmidt@vanderbilt.edu

Dr. Forrest Shull is the Principal Director for Advanced Computing and Software at the Office of the Under Secretary of Defense for Research and Engineering (OUSD (R&E)). Dr. Shull provides strategic direction for implementing advanced computing and software solutions across the Department of Defense (DoD), while coordinating scientific and technical development activities. Prior to his current role, Dr. Shull served as the Lead for Defense Software Acquisition Policy Research at the Software Engineering Institute at Carnegie Mellon.

Dr. Forrest Shull

Principal Director

Office of the Under Secretary of Defense (R&E)

forrest.j.shull.civ@mail.mil

John Robert is a Principal Engineer at the Software Engineering Institute and the Deputy Director for the Software Solutions Division. Mr. Robert provides leadership for software engineering research and the development of technologies in partnership with Department of Defense (DoD) programs and industry to enable the broad transition of new software engineering approaches. Mr. Robert has led multiple SEI technical partnerships with high priority DoD programs, resulting in high customer value and beneficial connections to SEI research.

John Robert

Principal Engineer

Software Engineering Institute

jer@sei.cmu.edu

Dr. Ipek Ozkaya is a principal researcher and the technical director of the Engineering Intelligent Software Systems group at the Software Engineering Institute. Her areas of work include software architecture, software design automation, and managing technical debt in software-reliant and AI-enabled systems. At the SEI, she has worked with several government and industry organizations in domains including avionics, power and automation, IoT, healthcare, and IT. Ozkaya is the co-author of a practitioner book titled *Managing Technical Debt: Reducing Friction in Software Development.*

Dr. Ipek Ozkaya

Technical Director

Software Engineering Institute

ozkaya@sei.cmu.edu