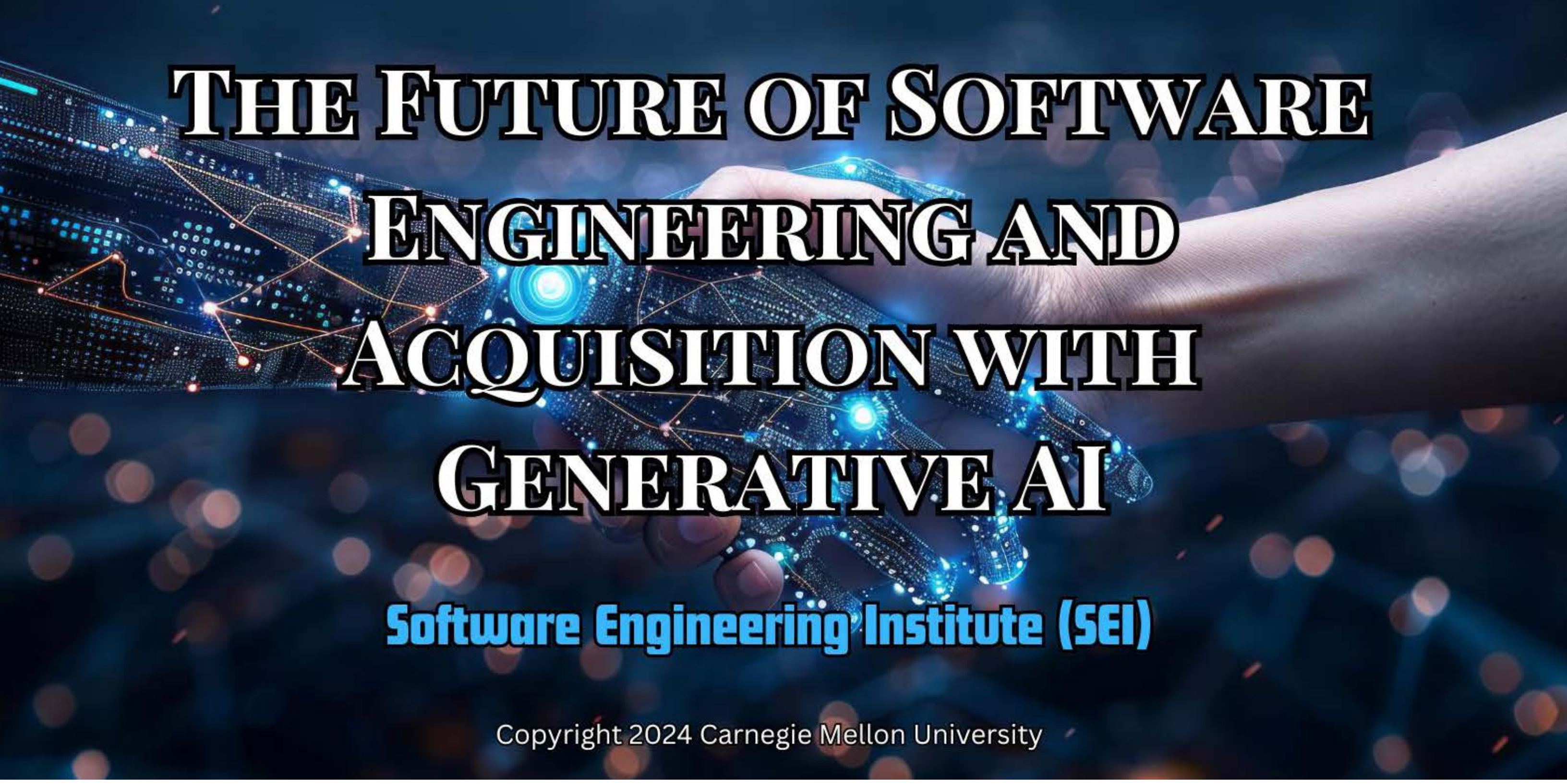

# The Future of Software Engineering and Acquisition with Generative AI

Software Engineering Institute (SEI)

## Introduction

This article examines the transformative potential of generative artificial intelligence (AI) in redefining software engineering and acquisition practices. Distinguished by the ability to create new content from vast datasets, generative AI promises increased productivity and innovation that is particularly relevant for the Department of Defense (DoD). However, adopting generative AI poses both opportunities and challenges. This article delves into the nature of generative AI, focusing on large language models (LLMs) that produce text-based content and highlighting their application for tasks like code generation, document summarization and discrepancy analysis, and decision support. However, the potential pitfalls of AI, such as overfitting and biased outputs, necessitate robust validation methods and human oversight. We advocate integrating generative AI with human expertise to navigate the challenges and fully leverage its potential in software engineering and acquisition.

## Demystifying Generative AI and LLMs

Generative AI is only one form of AI, which also includes machine learning, expert systems, neural networks, fuzzy logic, evolutionary algorithms, and reinforcement learning [1]. Understanding what makes generative AI different from other forms of AI is crucial for understanding how it can best be applied to solve software engineering and acquisition problems. For example, machine learning and generative AI both rely on training sophisticated models, but these models excel at different tasks. Machine learning typically focuses on classification problems (e.g., recognizing an object within an image), whereas generative AI differs in its ability to create new content (e.g., generating answers to user questions).

A large language model is a form of generative AI that creates text-based content and has many potential applications in software engineering and acquisition, both of which are domains with extensive text-based content. At its core, an LLM is a sophisticated neural network trained on enormous repositories of data encompassing books, code, articles, and websites. Through this training, an LLM grasps the intricate patterns and interconnections within the input it's trained upon. The probabilistic and randomized selection of the “next token” when generating outputs can provide users with an impression of correctness and style. Consequently, LLMs can produce coherent output, including grammatically accurate sentences and passages that closely resemble human-generated content, as well as syntactically and semantically precise software code segments.

## Challenges and Considerations

AI models are distinct from other types of models (e.g., simulation models) that encode precise mathematical or physics-based rules for a domain. AI models are statistically based, and they learn patterns from training on large data corpora. While this training allows AI models to discover relations that humans may not recognize, the statistical nature of AI models can also yield errors. Consequently, AI models face several pitfalls that include overfitting to specific datasets or failing to adapt to new and diverse data scenarios. These pitfalls underscore the need for robust validation methods to ensure these tools are enhancing, rather than compromising, the quality and reliability of software products.

LLMs, for example, are generally adept at parsing and generating nuanced text, which is valuable for generating documentation, commenting on code, and facilitating conversational interfaces within development tools [2]. The application of LLMs is not without challenges, however, since they can misrepresent context or yield biased output based on the data they were trained on. Consequently, careful human review and oversight is needed to align the text output of LLMs with software development standards, governance policies, and ethical norms.

## Opportunities

Despite these issues, incorporating generative AI—particularly LLMs—into software engineering and acquisition processes can yield a number of benefits. For example, LLMs can enhance problem-solving capabilities, streamline the creation and management of technical documentation, and foster adaptive information-centric workflows. Naturally, generative AI must be applied judiciously, with careful attention to potential biases, error margins in novel situations, the clarity of user query interpretation, and the ethical implications of their deployment.

The synergy between human expertise and generative AI in software engineering and acquisition is essential to leverage the full potential of these technologies. As generative AI continues to progress, it should not supplant human involvement but rather complement it, ensuring that AI-augmented outputs are understandable and ethically sound. It is vital to maintain human oversight to validate the reliability and accuracy of outputs produced by generative AI.

While the generative AI discussion in this paper focuses on a single modality (text) in conjunction with LLMs, applications in other modalities are maturing quickly. Generative AI can already create images, audio, and video based on text input, each of which creates additional opportunities for the application of this technology. For example, generative AI can be used to:

1. Create prompt-based prototypes [3]
2. Simulate user interface designs
3. Create educational videos that demonstrate the use of new software tools or features
4. Automate the production of training materials based on acquisition documents
5. Generate realistic audio-visual scenarios for testing software interoperability

6. Craft visualizations that help stakeholders understand the implications of different software architectures.

## Generative AI in Software Engineering

The integration of generative AI presents various opportunities in software engineering tasks, such as code generation, configuration deployment, and testing support, as summarized below:

- It can generate boilerplate code, significantly enhancing software development workflows by reducing manual coding errors and increasing developer productivity.
- It can generate setup and provisioning files for software environments, ensuring consistency and accuracy across configurations of multiple deployments.
- It can generate tests for edge cases, increasing the coverage and reliability of software testing processes.

Used appropriately, tools that incorporate generative AI can help developers significantly accelerate the development of experimental defense capability by enabling rapid prototyping and simulation. An ongoing challenge, however, involves defining success criteria for the many emerging uses of generative AI in software engineering [4]. Our experience at the Carnegie Mellon University (CMU) Software Engineering Institute (SEI) —a DoD federally funded research and development center (FFRDC)—indicates that integrating generative AI into the software development lifecycle (SDLC) requires a measured approach, balancing concerns like disclosure, accuracy, and ethical use [5][6][7]. Success hinges on developing organizational policies for such concerns and adapting to evolving governance and regulations.

An empirical understanding of workflow alterations and data collection helps inform decisions about the success of new approaches. For instance, by tracking the time required for automated code generation versus manual coding practices, organizations can assess productivity gains and determine the optimal integration of AI-augmented methods within their development lifecycles. Moreover, traditional practices, such as code reviews with customized checklists, may even regain prominence, providing humans in the loop with the tools and methods to ensure the reliability and testability of code and systems developed with the assistance of generative AI.

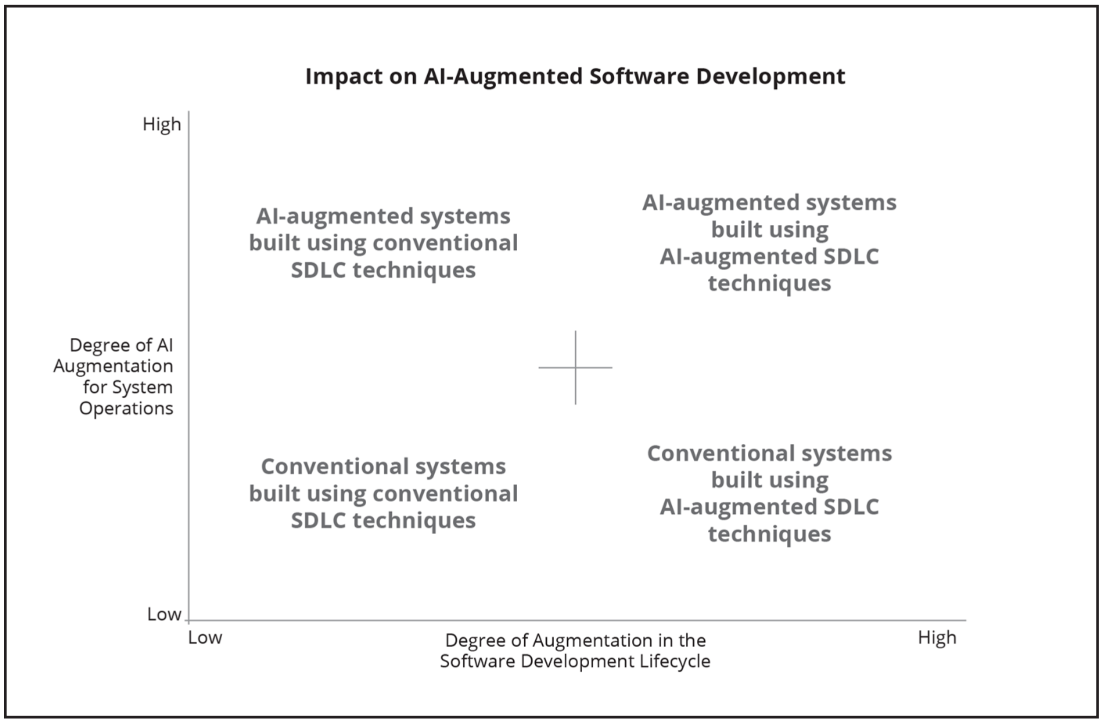
Impact on AI-Augmented Software Development
High
Low
Degree of AI Augmentation for System Operations
AI-augmented systems built using conventional SDLC techniques
AI-augmented systems built using AI-augmented SDLC techniques
Conventional systems built using conventional SDLC techniques
Conventional systems built using AI-augmented SDLC techniques
Low
Degree of Augmentation in the Software Development Lifecycle
High

**Figure 1.** *Taxonomy of AI Augmentation for System Operations and the Software Development Lifecycle.*

**Figure 1** expands upon the vision presented in our 2021 book, Architecting the Future of Software Engineering: A National Agenda for Software Engineering Research & Development, to codify opportunities for applying AI augmentation in both system operations and the SDLC, ranging from conventional methods to fully AI-augmented methods [8][9]. Use of generative AI is a driver of the degree of AI-augmentation in the SDLC axis in the scope of our discussion in this paper, but use of AI technologies in operations or the SDLC is not limited to generative AI.

Each quadrant in **Figure 1** is summarized below:

- **Conventional systems built using conventional SDLC techniques—**This quadrant represents a low degree of AI augmentation for both system operations and the SDLC, which is the baseline of most software-reliant projects today. An example is an avionics mission computing system that uses distributed object computing middleware and rate monotonic scheduling and is developed using conventional SDLC processes without any AI-augmented tools or methods.
- **Conventional systems built using AI-augmented techniques—**This quadrant represents an emerging area of research, development, and practice in the software engineering community, where system operations have a low degree of AI augmentation, but AI-augmented tools and methods are used in the SDLC. An example is a website hosting service where the content is not AI augmented, but the development process employs AI-based code generators (such as GitHub Copilot), AI-based code review tools (such as Codiga), and/or AI-based testing tools (such as DiffBlue Cover).
- **AI-augmented systems built using conventional SDLC techniques—**This quadrant represents a high degree of AI augmentation in systems, especially in their runtime operations, but uses conventional methods in the SDLC. An example is a recommendation engine in an e-commerce platform that employs machine learning algorithms to personalize recommendations, but the software itself is developed, tested, and deployed using conventional Agile methods and the React.js and Node.js frameworks.
- **AI-augmented systems built using AI-augmented techniques—**This quadrant represents the pinnacle of AI augmentation, with a high degree of AI-augmentation for both systems operations and the SDLC. An example is a self-driving car system that uses machine learning algorithms for navigation and decision making while also using AI-driven code generators, code review and repair tools, unit test generation, and DevOps tools for software development, testing, and deployment.

Applying generative AI for AI-augmented methods in software engineering is likely to change many processes across the SDLC. Further work is needed to address potential errors unique to generative AI (e.g., new tools for detecting and addressing these errors) and methods for measuring the impact of generative AI use (e.g., on feature delivery rates and data protection). Software engineering research to date has largely focused on demonstrating application of LLMs to improve routine software engineering tasks, demonstrating improvements along building conventional systems using AI-augmented SDLC techniques. Examples include:

- Using LLMs in test automation [10]
- Converting requirements to machine readable formats [11]
- Auto code completion [12]
- Code comment generation [13]
- Program repair [14]
- Code review [15]

Works such as these, focusing on improving tasks using LLM approaches, need to be complemented by approaches which look at end to end workflows and how to complement LLMs with other automation.

Moreover, research is needed to develop specialized AI models for domains and technologies that are uncommon outside of the DoD. For example, commercial generative AI will likely favor current technologies (such as service-oriented architectures and mobile cloud computing) and popular programming languages (such as Python and Rust). Consequently, it may be hard for DoD programs to leverage generative AI capabilities in less common settings, such as maintenance and modernization of systems that use older programming languages, such as Fortran, Jovial, or even Cobol.

## Generative AI in Acquisition

The application of generative AI to DoD acquisition is a potentially transformative shift, offering opportunities to streamline processes, enhance strategic decision making, and optimize use of limited expertise and resources in DoD acquisition [16]. In the highly complex, heavily regulated, and security-sensitive domain of DoD acquisition, generative AI can perform several pivotal tasks, including the following:

- It can summarize voluminous policy documents (e.g., DoD directives, instructions, memoranda, and guidance) and assist in updating the documentation to increase consistency.
- It can sift through extensive regulatory policies and standards to identify the most relevant areas for a specific acquisition or system context and assist in monitoring regulatory compliance throughout the system lifecycle.
- It can assist in identifying potential risks or threats within the acquisition process, for example from cyber threats or supply chain compromises. This proactive and ongoing identification allows for the implementation of robust security measures and risk mitigation strategies.

Success in applying generative AI within defense acquisition can be evaluated via several criteria, including the enhancement of national security, the efficiency of procurement processes, the regulatory compliance of acquired defense systems, and the effectiveness of risk management. Measurable outcomes include the reduction of development and procurement timelines, improvements in the quality and performance of defense capabilities, and higher mission resilience through regulatory compliance and risk mitigation.

The application of generative AI in defense acquisition workflows similarly includes multiple risks and considerations. The reliance on generative AI to inform decision-making processes necessitates clear process and scrutiny to reduce biases and ensure data integrity. Moreover, there's the challenge of ensuring that AI-generated solutions comply with international laws and ethical standards related

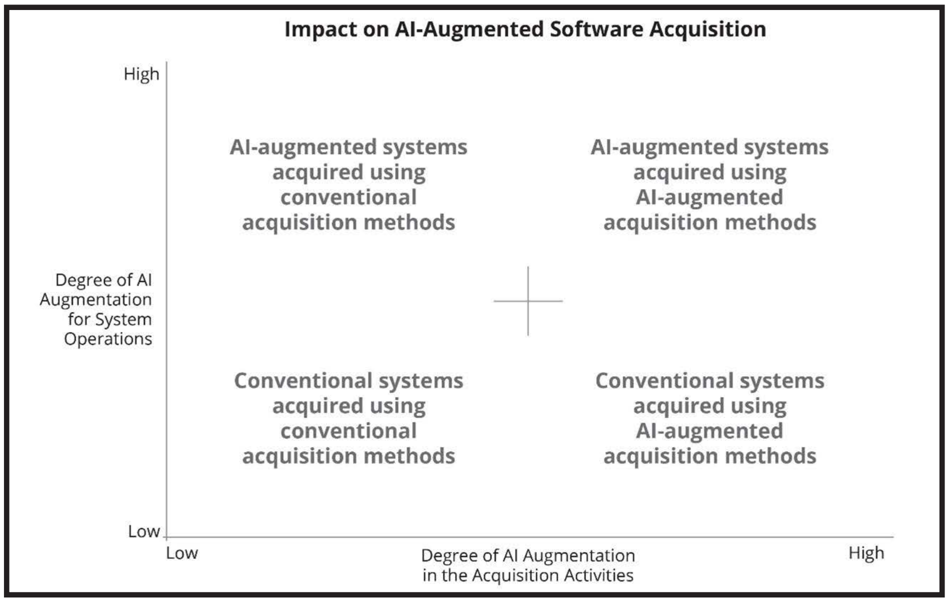
Impact on AI-Augmented Software Acquisition
High
AI-augmented systems acquired using conventional acquisition methods
AI-augmented systems acquired using AI-augmented acquisition methods
Degree of AI Augmentation for System Operations
Conventional systems acquired using conventional acquisition methods
Conventional systems acquired using AI-augmented acquisition methods
Low
Low
Degree of AI Augmentation in the Acquisition Activities
High

**Figure 2.** *Taxonomy of AI Augmentation for System Operations and Acquisition Activities.*

to defense acquisition. To mitigate these risks, a well-balanced approach that combines generative AI with human expertise and oversight is crucial to ensuring defense acquisition processes remain secure, efficient, and aligned with strategic objectives.

**Figure 2** depicts opportunities for applying AI augmentation in both system operations and acquisition activities, ranging from conventional to fully AI-augmented methods. Use of generative AI is a driver of the degree of AI-augmentation in the acquisition activities axis in the scope of our discussion in this paper, but use of AI technologies in operations and the SDLC is not limited to generative AI.

Each quadrant in **Figure 2** is summarized below.

- **Conventional systems acquired using conventional acquisition methods—** This quadrant represents a low degree of AI augmentation (if used at all) for both system operations and acquisition, which is the baseline of the vast majority of software-reliant acquisition programs today. An example is a military-grade GPS satellite system that uses traditional data transmission and encryption for operations and is developed using conventional acquisition processes without any AI-augmented tools or methods.
- **Conventional systems acquired using AI-augmented acquisition methods—** This quadrant represents an emerging area of research in the acquisition community in which system operations have a low degree of AI augmentation, but AI-augmented tools and methods are used in the acquisition activities. An example is a GPS-guided munition where the content is not AI-augmented, but the acquisition activities employ AI-assistance in identifying and analyzing relevant regulations, standards, and potential security risks.
- **AI-augmented systems acquired using conventional acquisition methods**— This quadrant represents a high degree of AI augmentation in systems, especially in operations, but uses conventional methods in the acquisition. An example is a radar system that employs machine learning to identify and prioritize possible targets, but the system is acquired using conventional methods.
- **AI-augmented systems acquired using AI-augmented acquisition methods**— This quadrant represents the pinnacle of AI augmentation, with a high degree of AI-augmentation for both systems operations and the acquisition. An example is an autonomous vehicle or platform that employs AI to navigate while also using AI-augmented acquisition processes, methods, and tools, such as text summarization and semi-automated regulatory compliance.

It is important to recognize that applying advanced tool support to acquisition tasks—especially generative AI-based techniques—is still in its infancy. Further work is needed, therefore, on developing more sophisticated generative AI models that can:

1. Understand and interpret large and complex acquisition documents
2. Enhance the data analytics capabilities to forecast project outcomes and risks more accurately
3. Create more intuitive interfaces for human-AI interaction to facilitate decision making
4. Conduct comprehensive studies on the long-term impacts and ethics of AI integration into the acquisition process

Moreover, acquisition professionals must be trained to manage and collaborate with AI-augmented processes and systems effectively to enable the seamless integration of AI tools within existing acquisition workflows, as discussed in the "Essential Generative AI Skills for the Workforce of Tomorrow" section on page 36.

## Example Software Engineering and Acquisition Use Cases

The ability of LLMs to generate plausible content for text and code applications in software engineering has motivated steadily increasing experimentation by researchers and practitioners. For example, a literature review of 229 research papers written between 2017-2023 on the application of LLMs to software engineering problems finds applications spanning requirements, design, development, testing, maintenance, and management activities, with development and testing being the most common [2].

Based on our work with many government organizations, the SEI has adopted a broader perspective and formulated several dozen ideas for using LLMs in common software engineering and acquisition activities (see Table 1 for examples) [5]. Two important observations emerged from this activity. First, most use cases represent human-AI partnerships in which an LLM or generative AI service could be used to help humans complete tasks more quickly (as opposed to replacing humans). Second, deciding which use cases would be most feasible, beneficial, or affordable is a non-trivial decision for those organizations just getting started with LLMs. A discussion on developing use cases and assessing the suitability of generative AI is available in the SEI report on Assessing Opportunities for LLMs in Software Engineering and Acquisition [5].

| Software Engineering Use Cases | Acquisition Use Cases |
|---|---|
| **SE1.** A developer uses an LLM to find vulnerabilities in existing code, hoping that the exercise will catch additional issues not already found by static analysis tools. | **A1.** A new acquisition specialist uses an LLM to generate an overview of relevant federal regulations for an upcoming RFP review, expecting the summary to save time in background reading. |
| **SE2.** A developer uses an LLM to generate code that parses structured input files and performs specified numerical analysis of its inputs, expecting it to generate code with the desired capabilities. | **A2.** A chief engineer uses an LLM to generate a comparison of alternatives from multiple proposals, expecting it to use the budget and schedule formulas from previous similar proposal reviews and generate accurate itemized comparisons. |
| **SE3.** A tester uses an LLM to create functional test cases, expecting it to produce a set of text test cases from a provided requirements document. | **A3.** A contract specialist uses an LLM to generate ideas for an RFI solicitation given a set of concerns and a vague problem description, expecting it to generate a draft RFI that is at least 75% aligned with their needs. |
| **SE4.** A developer uses an LLM to generate software documentation from code to be maintained, expecting it to summarize its functionality and interface. | **A4.** A CTO uses an LLM to create a report summarizing all uses of digital engineering technologies in the organization based on internal documents, expecting it can quickly produce a clear summary that is at least 90% correct. |

**Table 1.** *Sample Software Engineering and Acquisition Use Cases.*

| | |
|---|---|
| **SE5.** A software engineer who is unfamiliar with SQL uses an LLM to generate a SQL query from a natural language description, expecting it to generate a correct query that can be tested immediately. | **A5.** A program office lead uses an LLM to evaluate a contractor's code delivery for compliance with required design patterns, expecting that it will identify any instances in which the code fails to use required patterns. |
| **SE6.** A software architect uses an LLM to validate whether code that is ready for deployment is consistent with the system's architecture, expecting that it will reliably catch deviations from the intended architecture. | **A6.** A program manager uses an LLM to summarize a set of historical artifacts from the past six months in preparation for a high visibility program review and provides specific retrieval criteria (e.g., delivery tempo, status of open defects, and schedule), expecting it to generate an accurate summary of program status that complies with the retrieval criteria. |
| **SE7.** A developer uses an LLM to translate several classes from C++ to Rust, expecting that the translated code will pass the same tests and be more secure and memory safe. | **A7.** A program manager uses an LLM to generate a revised draft of a statement of work given a short starting description and a list of concerns (e.g., cybersecurity, software delivery tempo, and interoperability goals). The program manager expects it to generate a structure that can be quickly refined and that includes topics drawn from best practices that they may not think to request explicitly. |
| **SE8.** A developer uses an LLM to generate synthetic test data for a new feature being developed, expecting that it will quickly generate syntactically correct and representative data. | **A8.** A requirements engineer uses an LLM to generate draft requirements statements for a program upgrade based on past similar capabilities, expecting them to be a good starting point. |
| **SE9.** A developer provides an LLM with code that is failing in production and a description of the failures, expecting it to help the developer diagnose the root cause and propose a fix. | **A9.** A contract officer Is seeking funding to conduct research on a high priority topic they are not familiar with. The contract officer uses an LLM to create example project descriptions for their context, expecting it to produce reasonable descriptions. |

**Table 1.** *Sample Software Engineering and Acquisition Use Cases.*

## Deciding When (and When Not) to use Generative AI

As generative AI continues to reshape day-to-day tasks in the software engineering and acquisition ecosystems, a key question to consider is when generative AI should and should not be used. Some transformative opportunities exist that boost productivity, such as coding, testing, simulation, document analysis, and data synthesis. However, challenges like "hallucinations" (which are incorrect information generated by an LLM) and data disclosure necessitate a measured approach.

Determining the suitability of generative AI for any given task depends on assessing the nature and complexity of the task against concerns like data disclosure, accuracy, and ethical use, especially in sensitive contexts like DoD acquisition programs. Recognizing such concerns and deciding how to address each helps decision makers make more informed choices. Multiple perspectives should therefore be considered before adopting generative AI since it sometimes produces incorrect results. **Figure 3** depicts this perspective based on the following two questions:

- How much time and effort are needed for users to recognize that results from generative AI are incorrect?
- What are the consequences of users acting on mistaken results?

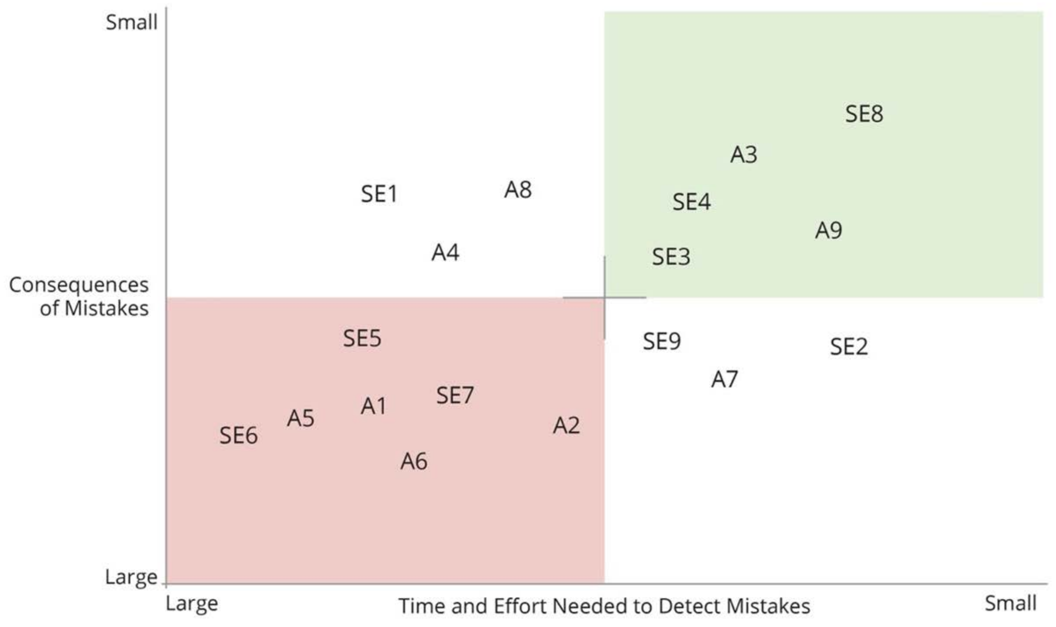

**Figure 3.** *Two Ways of Evaluating Concerns with the Generation of Incorrect Results.*

**Figure 3** shows a notional placement of the use cases from **Table 1**. The actual placement would require refinement of these use cases for specific application contexts, but the notional placement on these two questions provides insights into the opportunities for applying LLMs to a range of use cases. The upper-right (green) quadrant is ideal since mistakes have small consequences and users can detect them with minimal effort. Use cases in this quadrant are thus a good place for organizations to begin experimenting with generative AI adoption. In contrast, the lower-left quadrant represents the least favorable use cases for applying generative AI since mistakes have large consequences and require extensive time and effort for users to detect.

Software development organizations and acquisition programs can employ several strategies to manage concerns about the use of generative AI. The following sections describe local deployment, use of domain-specific models, and the establishment of ethical use guidelines as three candidate strategies relevant to government use cases, as well as use cases for other high-stakes domains, such as healthcare, finance, and law.

## Local Deployment to Mitigate Data Disclosure Challenges

Use of commercial generative AI services often requires users to share data they operate on (e.g., prompts, code being generated, and documents being summarized) with the service provider because the models are hosted remotely. While not all use cases require sharing sensitive data, many do, which is unacceptable for defense systems, defense software engineering, and defense acquisition. For example, uploading proprietary or controlled unclassified information (CUI) documents to a generative AI service violates data disclosure rules since those documents would be ingested into the generative AI service and therefore accessible to unauthorized individuals.

Strategies for handling sensitive data disclosure in the DoD and other high-stakes domains may involve the use of synthetic data to address disclosure concerns, although this approach has limitations [17]. New approaches for security classification adherence are also needed to ensure appropriate handling of classified and unclassified data. Human oversight remains vital for task-specific applications, with continuous human involvement ensuring that data is only disclosed as permitted by policy and regulations.

One way to significantly mitigate this risk is to rely on LLMs that are hosted on trusted networks and share no information with the model's owner. Local LLMs can include models that are trained locally, open-source models that are deployed locally, or commercial offerings that are deployed locally (e.g., complying with FedRAMP guidance). Although local LLMs may not be as powerful or up-to-date as their remote counterparts, they may be viable choices for many applications based on the success criteria, evaluation criteria, and risk concerns of each use case.

## Use of Domain-Specific Models to Enhance Generative AI Accuracy

Exploring the role of domain-specific models may aid the use of generative AI in specialized environments. Domain-specific LLMs are trained on data from a specific geographical or organizational context, which can capture nuances and patterns relevant to that particular environment [18]. These models contribute to improved accuracy and relevance in generating outputs tailored to local requirements, ensuring that the generated content aligns closely with the specific needs of the intended users or stakeholders. In the context of software engineering and acquisition, domain-specific models can be trained to understand and generate content that is deeply intertwined with the unique practices, terminology, and challenges of these fields.

For instance, within software engineering, domain-specific models could predict how changes to one part of a system might affect the rest or suggest software patterns that are most appropriate for a given requirement. In software acquisition, these models could simulate the project management lifecycle to forecast potential risks and outcomes, generate documentation that aligns with legal and industry standards, or optimize the allocation of resources. This tailored application of generative AI to the intricacies of software development and procurement processes can lead to more precise requirement analyses, better cost estimations, and improved strategic decision making, thereby enhancing the overall quality and reliability of software products and the efficiency of acquisition processes.

Domain-specific models for software engineering can also be trained on a vast repository of code unique to a specific programming language or framework. This specialization allows generative AI to offer more accurate and contextually relevant code suggestions, aiding developers in their coding tasks. Investing in domain-specific models for defense applications helps align their capabilities with the unique needs of hyperspectral imaging, radio frequency sensing, and other modalities.

Domain-specific models also come with challenges, however, including assembling enough quality training data and verifying output behaviors. These challenges should be evaluated carefully, so that the net benefit to the system is an improvement in productivity, capability, or some other relevant metric. Nevertheless, the incorporation of domain-specific models in generative AI has the potential to ensure a more tailored and context-aware application of AI technologies.

## Establishing Responsible and Ethical Use Guidelines

The responsible use of generative AI in tomorrow's workforce is critical to mitigate the potential risks and ethical concerns associated with this technology. Responsible and ethical development and use of generative AI is an area of concern that spans multiple activities, including (but not limited to) data collection and preparation, model development, and use of generative AI tools and services. Challenges include the perpetuation of biases inherent in training data, the necessity for consistent monitoring and updates to prevent misuse, and the complexities surrounding the explainability of sophisticated models. Addressing these challenges requires an ongoing commitment to research, collaboration, and transparency to foster an equilibrium between innovation, development, and responsible use. It is critical to establish the scope of activities for intended use and clarify the guidelines that are most applicable to stakeholders.

The capabilities of generative AI models are evolving rapidly, so it is essential to educate users on their responsible use. There is an emerging consensus among developers and users that the most effective generative AI tools are those that empower users with control over data privacy, model training parameters, and content generation constraints [19]. Usage patterns across software engineering and acquisition reveal a consistent interactive cycle that includes prompting the AI, executing an action based on its response, and then proceeding with further prompts.

AI-augmented methods should keep humans in the loop for multiple reasons, one of which is to be a safeguard and take responsibility for the outcome. Generative AI does make mistakes, so humans should operate with that assumption and compensate. This iterative, human-in-the-loop approach underscores the critical need for guidelines on the appropriate use of generative AI, accentuating the pivotal role of users. Some guidelines will be straightforward (such as simply reminding users of the organization's data privacy and information disclosure policies) whereas others may require strategies that include limiting use of generative AI services for particular use cases.

The landscape of responsible and ethical AI development and application is expanding, with numerous frameworks emerging to mitigate AI's unintended impacts, including those from generative AI. For instance, the Defense Innovation Unit (DIU) has formulated Responsible AI (RAI) guidelines to streamline the evaluation process for those involved in AI project development, such as program managers, commercial vendors, or government collaborators [20]. These guidelines cover a broad spectrum of considerations, including legal, procurement, technical, and operational aspects. They offer directives for both AI tool developers and users; for instance, outlining the extent of technical transparency required while safeguarding proprietary data. Organizations are encouraged to augment these guidelines with their specific procedures and data-sharing policies, ensuring alignment with their domain-specific requirements.

The broad adoption of responsible AI in software engineering and acquisition is also contingent on legal maturity. As service providers begin to indemnify outputs from generative AI tools against intellectual property infringements, we anticipate the formation of a trusted ecosystem. This ecosystem will be critical in fostering responsible use and ensuring that generative AI is leveraged to enhance, rather than compromise, the quality and reliability of software products and services.

## Essential Generative AI Skills for the Workforce of Tomorrow

Generative AI will augment, rather than replace, the capabilities of software engineers and acquisition professionals for the foreseeable future. Consequently, workers in both professions must maintain expertise in their respective domains. Software engineers will need proficiency with requirement analysis, software design, programming languages, testing practices, and deployment. Likewise, acquisition professionals will need proficiency with acquisition regulations, acquisition pathways, and their application to different system contexts [21].

To unlock the potential of generative AI, however, software engineers and acquisition professionals should cultivate new skills, such as those visualized in **Figure 4**. For example, both should learn prompt engineering and how to apply prompt patterns to elicit effective results from generative AI [22]. Likewise, both should master decomposing complex issues into manageable components and assessing which of these issues generative AI can help solve. Above all, users who responsibly exercise curiosity, experimentation, and a willingness to learn new skills will guide the DoD in successfully adapting to the dynamic landscape of generative AI.

When skillsets are visually summarized (as seen in **Figure 4**), it becomes clear that generative AI does not replace software and acquisition professionals, but rather augments their effectiveness through new skills, such as prompt engineering and problem decomposition. Individual effectiveness in utilizing generative AI may vary based on skills, experience, and adaptability. Some individuals may naturally excel in leveraging these tools for code generation, problem-solving, and documentation, whereas others may require more extensive training. Either way, continuous learning and adaptability are key.

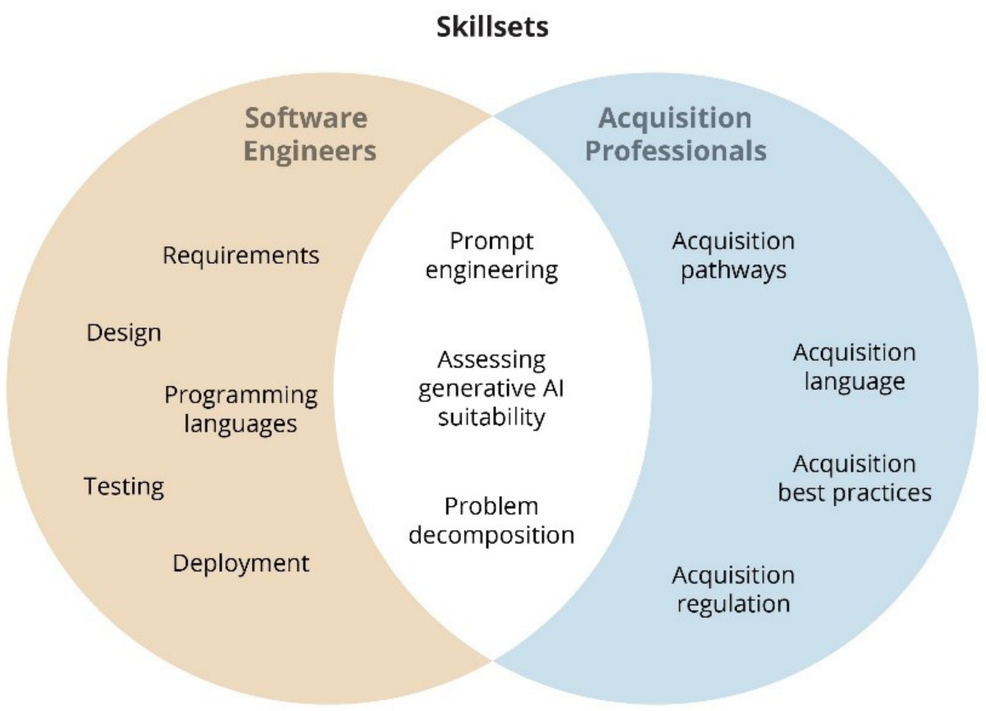

**Figure 4.** *Representative Skillsets for Software Engineers and Acquisition Professionals Using Generative AI.*

Given generative AI's tendency to make mistakes, validating the outputs of generative AI is an essential activity for both the software engineering and acquisition communities. The specific skills needed to validate output will vary with nature of the task and data, but two questions should always be considered: Is the information in the output correct, and is any information missing from the output? Here are some examples:

- **Generating source code for specific requirement**—Users can write unit tests to confirm that computations performed by generated code are correct. Likewise, users can inspect the generated code to confirm that it does not performing any unnecessary work.
- **Summarizing document contents from a specific stakeholder perspective**—Users can request a summary of main points from a document relevant for a specific stakeholder (e.g., a software safety engineer, reliability engineer, cybersecurity analyst, etc.) and fact check the summary by searching the source material for relevant facts. Users can review the source material to confirm that essential and relevant points are included in the summary.

Incorporating generative AI tools into the software engineering educational curricula will help the emerging workforce [23]. Of course, existing software engineers must stay abreast of advancements throughout their career because changes happen quickly. Generative AI tools can also assist the acquisition workforce in keeping current with updates to acquisition regulations. However, acquisition professionals are responsible for codifying significant shifts in best practices and new system types, pending the availability of adequate new data to train generative AI tools.

Organizations offering training for acquisition professionals, such as Defense Acquisition University, will also need to incorporate generative AI into their curricula. While generative AI itself may not replace jobs, those who excel in using generative AI tools might surpass others in the job market. As the landscape of system development evolves, adapting to change and acquiring skills in the use of generative AI will be crucial for staying competitive in the dynamic and exciting future envisioned by the software engineering community.

## Conclusion: We Must Learn to Navigate an AI-augmented Future for Software Engineering and Acquisition

The initial adoption phase of generative AI in software engineering and acquisition will likely be tumultuous as users navigate the applicability and utility of these tools to different tasks. Some ideas will be highly successful, whereas others will prove disappointing. This exploratory phase is crucial as we collectively learn the potential of generative AI to shape future research and application throughout the DoD.

Software engineers across the globe are already applying generative AI in software engineering today, demonstrating practical applications in code generation for routine tasks. This early adoption has the potential to grow into more impactful applications, including accelerating software modernization (e.g., by resolving technical debt and repairing critical errors) and quickly assembling prototypes (e.g., by crawling software repositories to identify compatible candidate software elements). Similarly, application of generative AI to acquisition activities has potential to improve the efficiency of summarization and document generation of acquisition artifacts.

Generative AI significantly lowers the barrier to entry for content generation, with potentially mixed implications. These technologies are empowering users without formal software engineering backgrounds to solve complex problems using natural language interfaces, which opens access to the ideas and imagination of a much larger population. This empowerment also brings challenges, however, especially in terms of ensuring the quality of generated content when users lack the deep technical knowledge traditionally associated with software engineering roles. The impact of incorporating generated content without the benefit of conventional engineering review on system stability, security, and operational accuracy are unknown. It is important to recognize that generative AI services are tools to assist users, they do not replace expertise in software engineering and acquisition.

The educational landscape for software engineering and acquisition must evolve to integrate generative AI, preparing students and professionals alike to harness these tools effectively while also understanding their limitations and inherent biases. This curriculum development will facilitate a new breed of software engineers and acquisition professionals skilled in generative AI use and critical evaluation, ensuring their ability to innovate responsibly while maintaining ethical standards. These future-focused educational strategies are essential as generative AI increasingly becomes integral to many disciplines and domains, emphasizing a blend of technical proficiency with a thorough grasp of AI's ethical and practical implications. This balanced approach will foster professionals who can navigate the complexities of using generative AI and contribute to its ethical advancement.

In conclusion, navigating the future AI-augmented software engineering and acquisition is a tapestry of opportunity and responsibility, weaving together advancements in AI with the need for ethical stewardship and thoughtful integration into high-stakes DoD socio-technical systems.

## Acknowledgements

We would like to thank Erin Harper and Ed Desautels for editorial and design support.

## Copyright

## References

[1] McKinsey & Company. “What is Generative AI?” McKinsey & Company Featured Insights. 19 Jan. 2023. https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-generative-ai#/. Accessed 19 Mar. 2024.

[2] Hou, X. et al. “Large Language Models for Software Engineering: A Systematic Literature Review.” arXiv, 12 Sept. 2023. https://arxiv.org/pdf/2308.10620.pdf.

[3] Jiang, Ellen, et al. “PromptMaker: Prompt-Based Prototyping with Large Language Models.” Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Sys-

tems, Association for Computing Machinery, 2022, pp. 1–8. ACM Digital Library, https://doi.org/10.1145/3491101.3503564.

[4] A. Fan et al., “Large Language Models for Software Engineering: Survey and Open Problems,” 2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE), Melbourne, Australia, 2023, pp. 31-53, doi: 10.1109/ICSE-FoSE59343.2023.00008.

[5] Ozkaya, Ipek. “What Is Really Different in Engineering AI-Enabled Systems?” IEEE Software, vol. 37, no. 4, pp. 3–6, July-Aug. 2020, https://doi.org/10.1109/MS.2020.2993662.

[6] Bellomo, Stephany et al. “Assessing Opportunities for LLMs in Software Engineering and Acquisition.” Software Engineering Institute, 1 Nov. 2023, https://insights.sei.cmu.edu/library/assessing-opportunities-for-llms-in-software-engineering-and-acquisition/.

[7] Schmidt, Douglas et al. The Future of Software Engineering and Acquisition with Generative AI. Software Engineering Institute, 23 Jan. 2024, https://insights.sei.cmu.edu/library/the-future-of-software-engineering-and-acquisition-with-generative-ai/.

[8] Carleton, Anita et al. Architecting the Future of Software Engineering: A National Agenda for Software Engineering Research & Development. Software Engineering Institute, 2021, https://insights.sei.cmu.edu/library/architecting-the-future-of-software-engineering-a-national-agenda-for-software-engineering-research-development/.

[9] Ozkaya, Ipek et al. “Application of Large Language Models (LLMs) in Software Engineering: Overblown Hype or Disruptive Change?” SEI Blog, 2 Oct. 2023, https://insights.sei.cmu.edu/blog/application-of-large-language-models-llms-in-software-engineering-overblown-hype-or-disruptive-change/.

[10] Z. Liu, C. Chen, J. Wang, M. Chen, B. Wu, X. Che, D. Wang, Q. Wang. “Make LLM a Testing Expert: Bringing Human-like Interaction to Mobile GUI Testing via Functionality-aware Decisions.” 46th IEEE/ACM ICSE 2024.

[11] Tikayat Ray, Archana, et al. “Agile Methodology for the Standardization of Engineering Requirements Using Large Language Models.” Systems, vol. 11, July 2023, p. 352. https://doi.org/10.3390/systems11070352.

[12] Bird, Christian, et al. “Taking Flight with Copilot: Early Insights and Opportunities of AI-Powered Pair-Programming Tools.” Queue, vol. 20, no. 6, Jan. 2023, p. Pages 10:35-Pages 10:57. November/December, https://doi.org/10.1145/3582083.

[13] Geng, Mingyang, et al. “Large Language Models Are Few-Shot Summarizers: Multi-Intent Comment Generation via In-Context Learning.” IEEE Computer Society, 2024, pp. 453–65. www.computer.org, https://www.computer.org/csdl/proceedings-article/icse/2024/021700a453/1WDJaRUZRKg.

[14] Jin, Matthew, et al. “InferFix: End-to-End Program Repair with LLMs.” Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Association for Computing Machinery, 2023, pp. 1646–56. ACM Digital Library, https://doi.org/10.1145/3611643.3613892.

[15] Li, Lingwei, et al. “AUGER: Automatically Generating Review Comments with Pre-Training Models.” Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Association for Computing Machinery, 2022, pp. 1009–21. ACM Digital Library, https://doi.org/10.1145/3540250.3549099.

[16] Robert, John, and Douglas Schmidt. “10 Benefits and 10 Challenges of Applying Large Language Models to DoD Software Acquisition.” SEI Blog, 22 Jan. 2024, https://insights.sei.cmu.edu/blog/10-

benefits-and-10-challenges-of-applying-large-language-models-to-dod-software-acquisition/.

[17] Li, Zhuoyan, et al. “Synthetic Data Generation with Large Language Models for Text Classification: Potential and Limitations.” arXiv:2310.07849, 12 Oct. 2023. arXiv.org, https://doi.org/10.48550/arXiv.2310.07849.

[18] Nucci, Antonio. “What Is a Domain-Specific LLM? Examples and Benefits.” Aisera: Best Generative AI Platform For Enterprise, 11 Jan. 2024, https://aisera.com/blog/domain-specific-llm/..

[19] Zhang, Dawen et al. “Privacy and Copyright Protection in Generative AI: A Lifecycle Perspective.” 2024 IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI (CAIN), 2024.

[20] Defense Innovation Unit. “Responsible Artificial Intelligence: 2022 in Review.” Defense Innovation Unit Artificial Intelligence Portfolio, 1 May 2023, https://www.diu.mil/responsible-ai-guidelines. Accessed 19 Mar. 2024.

[21] Defense Acquisition University. “Adaptive Acquisition Framework.” DAU, https://aaf.dau.edu. Accessed 19 Mar. 2024.

[22] White, Jules et al. “A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT.” Proceedings of the 30th Pattern Languages of Programming (PLoP) Conference, Allerton Park, IL, October 23-25th, 2023.

[23] Kirova, Vassilka D. et al. “Software Engineering Education Must Adapt and Evolve for an LLM (Large Language Model) Environment.” Proceedings of the 55th ACM Technical Symposium on Computer Science Education, March 2024.

## About the Author

John E. Robert is a Principal Engineer at the SEI and the Deputy Director for the SEI’s Software Solutions Division. Robert provides leadership for software engineering research and the development of technologies in partnership with DoD programs and industry to enable the broad transition of new software engineering approaches. Robert has led multiple SEI technical partnerships with high-priority DoD programs, and was part of the lead author team on a national study for software engineering research and development in 2021 titled, *Architecting the Future of Software Engineering: A National Agenda for Software Engineering Research & Development*.

**John E. Robert**

**Deputy Director**

**Software Engineering Institute**

**jer@sei.cmu.edu**

James Ivers is a Principal Engineer and lead of the Architecture Design, Analysis, and Automation group at the SEI. His experience spans research and application of work in software architecture, code analysis, formal methods, and scaling our ability to evolve software. His most recent work focuses on using artificial intelligence (AI) for software engineering to support large-scale refactoring.

**James Ivers**

**Principal Engineer**

**Software Engineering Institute**

**jivers@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. 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**

Shen Zhang is a Senior Software Engineer within the Enabling Mission Capability at Scale directorate at the SEI. He provides expertise in software development processes and static code analysis for real-time mission critical software-intensive systems in the Air Force and Navy. More recently, he has been involved with using large language models to accelerate software engineering use cases. Prior to working at the SEI, he developed simulation products for industrial control systems in the commercial power industry.

**Shen Zhang**

**Software Engineer**

**Software Engineering Institute**

**szhang@sei.cmu.edu**