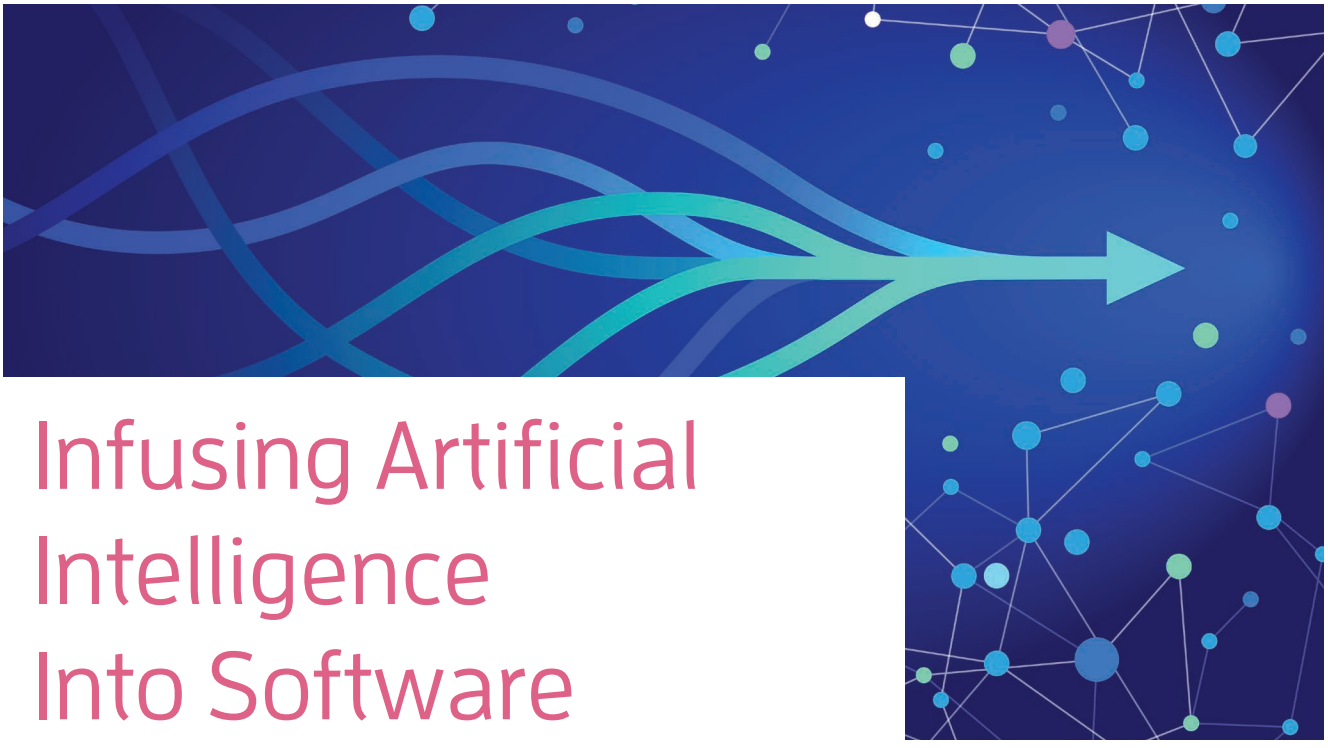

Infusing Artificial Intelligence Into Software Engineering and the DevSecOps Continuum

Tracy (Trac) Bannon, MITRE Corporation

The emergence of new artificial intelligence (AI) technologies, in particular, generative AI (GAI), shows groundbreaking potential, but there are challenges and limitations when evaluating GAI's applicability for software engineering.

Digital Object Identifier 10.1109/MC.2024.3423108
Date of current version: 27 August 2024

The software industry is inundated by industry reports, a surge in publishing, and breakneck growth in generative artificial intelligence (GAI) models, including the models' use in software engineering. Social media platforms contain additional evidence of rapid production of content through relentless opinions and experiences. Recently, Gartner stated that AI-assisted software engineering is at the apex of the hype cycle, the "peak of inflated expectations," estimating two to five years to reach productivity for production of viable software.[1] Clearly, the software engineering domain is in the early stages of experimentation with and adoption of AI-assisted software engineering tools.

This adoption marks a significant shift from traditional human-centric teams to integrated environments in which GAI does not merely support but actively participates in the development

0018-9162/24©2024IEEE

EDITOR **PHIL LAPLANTE**
IEEE Fellow;
plaplante@psu.edu

process. Today's modern software practices have been optimized for humans and are not yet fully equipped for the addition of GAI tooling.

I lead a collaborative research effort to address the emerging barriers to the adoption of GAI tools by development teams, understand the implications of team composition when GAI is introduced, and envision AI-driven software delivery platforms of the future (Figure 1).

Before the software industry can fully harness the potential of GAI in software engineering, we must first establish a baseline of current practices. This article explores the present state of the industry by examining scholarly publications and industry reports and preprints and summarizing the experiences of leading practitioners. By gaining a clear picture of today's practices, practitioners can identify the gaps and opportunities that pave the way for effective integration and utilization of GAI in software development. This foundational understanding is crucial to developing strategies that will drive the successful adoption and optimization of GAI tools, ultimately transforming software engineering practices.

DRIVING THE HYPE: ACCELERATED PUBLISHING

Stanford University reports that the number of AI-related publications[2] tripled from 2010 to 2022. In the late fall of 2022, OpenAI released ChatGPT to a broad user base, triggering a rapid

FIGURE 1. This image was generated by me, the author, using the DALL-E model.

increase in experimentation and publishing. My team wanted to investigate the rate of publication changes to understand how fast this discipline is accelerating.

I began by evaluating the counts of scholarly and preprint publishing of relevant articles. Since 2022, the publication of AI-related material, especially of preprint content, has accelerated constantly. To get an initial sense of the rate of publishing acceleration, I queried arXiv.org, an open access repository of preprint scholarly papers. In 2022, a total of 19 articles were submitted to arXiv.org in the computer science classification that included the term "software engineering" and a least one of the following "generative AI" or "GPT" or "LLM" with "LLM" representing the technical abbreviation of large language model. 2023 saw publication of 312 such articles. In the first four months of 2024, the number grew to 462 submitted (Table 1).

Data scientists are rapidly releasing GAI in addition to publications.[2] The release of 149 foundational models in 2023 marked an increase of over 100% from 2022. Regardless of a practitioner's career level or technical role, the pace of information and model publishing make it difficult to keep abreast of new research, techniques, discoveries, or models impacting organizational plans for experimentation.

THE EARLY DAYS OF GAI-ASSISTED SOFTWARE ENGINEERING

Software engineering teams have used traditional AI and machine learning (ML) in software engineering for several decades. For example, they have employed traditional AI techniques, such as expert systems and rules-based solutions, to improve code accuracy and analyze requirements since at least the 1980s. Applied usage of AI has been researched and studied for decades pre-GPT,[3] including the use of multiagent and agent-oriented methodologies. The practical application of GAI, however, is a more recent development, largely driven by the 2022 release of OpenAI's ChatGPT.

Models tailored to software engineering, such as Codex, PolyCoder, and CodeBERT, are relatively new, with many foundational models being designed

TABLE 1. AI-related scholarly papers posted to arXiv.org in the last three years.

Query	2022–present	Calendar year 2022	Calendar year 2023	January–April 2024	Calendar year 2024, estimated
order: -announced_date_first; size: 50; date_range: from 2022-01-01 to 2024-04-30; classification: Computer Science (cs); include_cross_list: True; terms: AND all="Software Engineering"; AND all="Generative AI" OR GPT OR LLM	793	19	312	492	1,384

and new techniques being explored to improve the usability, repeatability, and trustworthiness of GAI. Clearly, GAI-assisted software engineering is only in its infancy.

Preliminary findings from the team's systematic literature review of published and preprint articles from 2022 through May 2024, augmented by relevant industry studies, highlight that the current state of the practice focuses on evaluation of and experimentation with individual software engineering and software development lifecycle (SDLC) tasks, such as code completion or test case generation. Given the nature of GAI to lack predictability, usability, and trustworthiness, current industry and academic interest focuses heavily on addressing the accuracy, precision, predictability, and testability required for software engineering.

Current usage techniques of GAI to assist with software engineering fall into three categories: via direct prompting of a model, integration of GAI into a stand-alone software engineering tool, or, more recently, incorporation of GAI chatbots into the integrated development environment (IDE). The landscape becomes additionally complex given the two predominant and opposing techniques to access LLMs: externally hosted subscription-based tool-as-a-service solutions, such as Microsoft GitHub Copilot, or self-hosted solutions. Both access methods carry their own security and operational challenges.

In May 2023, Stack Overflow,[4] the online collaborative question-and-answer platform, conducted an industry survey of 90,000 developers. The report highlighted the individual SDLC tasks in which today's software engineers and developers have begun to leverage GAI assistance. Of the survey subset of 37,700 software engineers and developers who reported they are currently using GAI, nearly 83% reported using it to assist with writing code, 49% reported getting help on debugging, and 34% reported using GAI to assist in documenting code.

In October 2023, the MITRE Corporation engaged via social media with a group of renowned subject matter experts (SMEs) in modern software engineering practices. MITRE collected engagement patterns, perspectives, and emerging techniques for applying AI (including traditional AI, ML, and GAI), which highlighted the many individual SDLC tasks that may benefit from AI.

> The current landscape of GAI-powered tools in software engineering emphasizes individual activities, diverting focus from teamwork.

From those expert opinions, MITRE constructed Figure 2, showing the integration of AI across the DevSecOps continuum and showcasing various AI techniques being applied at different stages of the SDLC. The central visual element is the infinity loop representing the continuous cycle of DevOps practices: plan, code, build, test, release, deploy, operate, and monitor. The DevOps symbol is annotated with potential AI-enabled software engineering activities.

The team refreshed this AI tool use study in March 2024, and the findings validated the potential for AI integration across the DevSecOps continuum. Of note, however, is that these different applications of AI are at the individual task level. The integration of more tools into the SDLC adds both complexity and promise. As of March 2024, researchers, such as Tufano et al.[5] and Nghiem et al.,[6] have contributed a set of preprint articles that begin to address more holistic concepts, such as orchestration of AI-agents to knit together SDLC workflows. Figure 2 also provides a discussion mechanism for identifying key areas in the SDLC in which AI can have the greatest impact, keeping in mind that adoption of AI-driven practices must lead to more secure and high-quality software. Individual activities appear self-explanatory, and some are ready for deployment, but challenges and emerging barriers demand further exploration.

CURRENT CHALLENGES AND EMERGING BARRIERS

The explosion of GPT/software engineering literature and models complicates the extraction of valuable insights. This abundance of research can obscure significant findings and create difficulties in identifying the most relevant and impactful materials. I have divided the myriad additional challenges and emerging barriers into four categories:

1. overfocus at the individual task level
2. ill-defined needs and poorly fitted measurements
3. quality and security of generated outputs
4. the absence of human/machine recalibration.

These are discussed in the following sections.

Overfocus at the individual task level

The current landscape of GAI-powered tools in software engineering emphasizes individual activities, diverting focus from teamwork. These tools are often designed for use by individual contributors, leading to siloed operations within teams. Challenges in this category include the following:

- *Task execution isolated to individual contributors*: GAI tools are often designed for use by individuals, which is known to lead to siloed operations within teams. This isolation hinders collaborative efforts and highlights the need for tools that

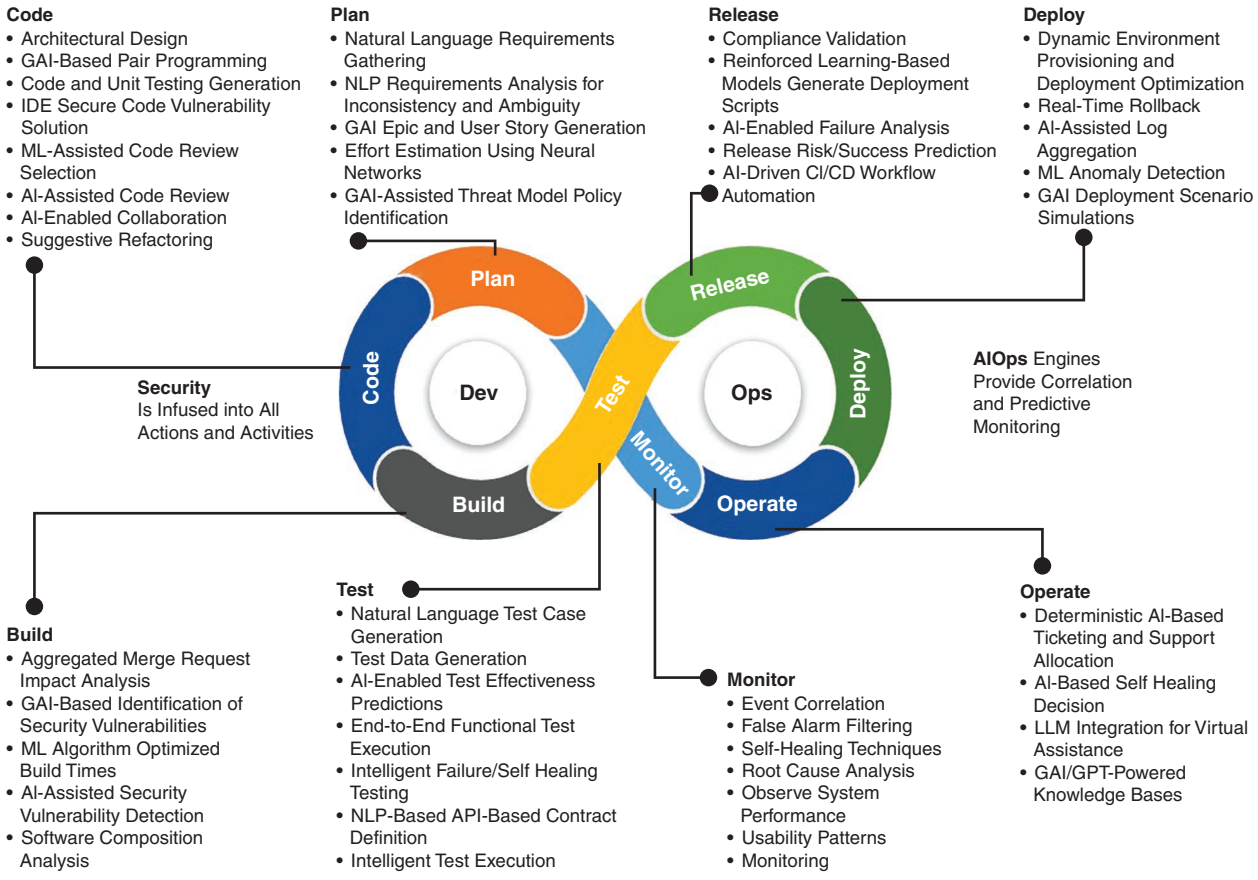

FIGURE 2. Infusing AI across the DevSecOps continuum.

support teamwork and collective problem solving. The Stack Overflow report[4] highlighted the current focus on individual versus organizational use and experimentation.

- *No cohesive GAI tool integration*: The current landscape of GAI tools in software engineering is highly fragmented, with each tool addressing specific tasks in a unique way, without a unified approach. This lack of standardization complicates the seamless integration of GAI tools into existing workflows, reducing overall efficiency.
- *Growing data and information silos*: The proliferation of AI tools has led to the creation of data silos, where information is stored in isolated pockets within an organization. These silos necessitate improved data management strategies to ensure effective sharing and utilization of data across different teams and tools.

Human–GAI sessions are not shared; they are individual conversations. The recent introduction of OpenAI ChatGPT 4o enterprise licensing has led to a reported new capability to define teams that may share a GPT session. This capability should be explored, tested, and evaluated.

- *GAI contradicts some DevSecOps principles*: Traceability, auditability, and explainability are core principles of modern software practices, but generated software assets lack lineage and provenance. DevSecOps is predicated on transparency and repeatability of tests and outcomes, whereas GAI often invokes randomness, providing users with multiple candidate solutions. The addition of external components, such as a rules engine, retrieval-augmented generation (RAG) components, or vector stores and external logic, increases the complexity of the software development toolchain.

Ill-defined needs and poorly fitted measurements

Organizations should start by clearly establishing the "why" and defining how to measure success when using GAI-empowered software engineering tools. It is essential to follow this

maxim to ensure meaningful and effective integration. Related concerns include the following:

- *Lack of defined mission/business need*: Many organizations struggle to clearly understand and define the mission or business need for integrating GAI into their processes. The release of ChatGPT to a broad consumer base has driven a manufactured focus on the immediate need to incorporate GAI. This lack of direction can result in missing or misaligned goals as well as over- or underutilization of the technology.
- *Inability to measure value and effectiveness*: Establishing consistent objectives and associated measures for evaluating the value and effectiveness of GAI-assisted software engineering tools poses significant difficulties. The lack of standardized approaches or frameworks makes it hard to accurately assess the performance and benefits of these technologies applied to the SDLC.
- *Ineffective productivity measurements*: Current productivity metrics do not adequately reflect the contributions of GAI, leading to skewed perceptions of its effectiveness. Adapting these metrics to better capture the unique impact of AI tools is essential to accurate performance evaluation. Industry reports have focused on individual productivity[5, 6] as opposed to team or organizational measures and frequently rely on subjective perceptions of individual productivity.

Quality and security of generated outputs

Garbage in, garbage out still applies to GAI models and GAI-empowered tools. Organizations must address concerns regarding transparency and assurance. This class of issues includes the following:

- *Security vulnerabilities in generated code*: GAI-generated code can introduce security vulnerabilities, posing risks to software integrity and safety.[9, 10] Upholding rigorous testing and human oversight is crucial to identify and mitigate these potential threats. An industry survey of 537 developers by Synk,[11] a software vulnerability detection security platform provider, found that 56% of respondents reported commonly encountering security issues in GAI code suggestions.
- *Accelerated pace of code development*: The volume of GAI-generated code is increasing, as are the associated risk and costs. Industry company GitClear[12] analyzed roughly 153 million changed lines of code in GitHub repositories. The code changes occurred between January 2020 and December 2023. GitClear's report found declines in the overall quality of generated code outputs, while the amount of code generated has increased, as has the frequency and number of changes made to a code base ("code churn"). Organizations often use code churn to assess the stability, quality, and maintainability of software, and measured code churn is on target to double in 2024.[12]

The release of ChatGPT to a broad consumer base has driven a manufactured focus on the immediate need to incorporate GAI.

Missing human/machine recalibration

Integrating GAI-assisted tools into software engineering necessitates a recalibration of expectations and processes to address the human impact. The challenges posed by GAI lead to the following concerns:

- *Adaptation of workflows and creation of new value streams*: Today's GAI-based solutions as well as research and experimentation are tailored for individual SDLC tasks and not the dynamic flow of software engineering. Integrating AI-assisted tools requires significant adaptation of existing workflows and the creation of new value streams. Organizations must be prepared to redesign their processes to fully leverage the capabilities of GAI, ensuring that their staff uses these tools effectively. However, these organizations often overlook workflow adaptations and are unprepared for the resulting fluctuations in metrics during adoption. Changes in tools, processes, and methods result in changes to metric baselines and, often, in the need for new and/or different measurements.
- *Human concerns on the trustworthiness of generated assets*: Software engineering, in its purest form, should be scientific and deterministic, and engineers must be able to trust the accuracy and security of the tools they use. LLMs, central to GAI, are subject to a phenomenon called *hallucinations*,[13] where outputs are fictitious or not trustworthy. These occurrences create a significant mental challenge for software engineers or developers attempting to leverage GAI-assisted tooling. Engineers must

be able to trust the accuracy and security of the tools they use. Stack Overflow[4] reported that 55% of survey respondents using AI are interested in using AI for software testing, but only 3% say they have high trust in AI tools. For seasoned software engineers, this means that use of AI tooling requires more oversight by the humans in the loop; for new-in-career professionals, it could mean decision fatigue caused by lack of experience and the need to check the tool outputs.

- *Altered team communication patterns*: Communicating with a GPT, chatbot, or GAI interface capable of conversation, such as via response, changes the communication patterns within a software engineering team. Preliminary observations and responses to in-person testimonials suggest a potential reduction in human-to-human communications, although the overall impact on software outcomes remains undetermined.
- *Intellectual property and data privacy challenges*: Using GAI in software development raises significant concerns regarding intellectual property (IP) and data protection. The ownership of GAI-generated outputs is often ambiguous due to the reliance on publicly sourced training data, potentially infringing on existing copyrights. U.S. copyright law mandates human intervention for protection, highlighting the necessity for human oversight in GAI-assisted creations.[14] Additionally, using third-party GAI services poses data privacy risks, especially when sensitive information is involved. Maintaining data integrity and transparency in GAI processes is crucial. Organizations must ask GAI service providers critical questions about their safeguards against security vulnerabilities and data breaches.

A CALL TO ACTION

Based on the preliminary information collected and assessment of practitioner use, I recommend the following actions for organizations seeking to infuse GAI-enabled tools into the SDLC, whether or not modern practices like DevSecOps are used:

- *Focus on AI assurance*: MITRE defines AI assurance as a process for discovering, assessing, and managing risk throughout the lifecycle of an AI-enabled system so that it operates effectively for the benefit of its stakeholders.[15] Assuring the reliability of GAI-assisted tools in software engineering involves thoroughly understanding the mission problem and the GAI solution to identify specific assurance needs related to code quality, security, and compliance. This proactive approach helps predict and mitigate risks early. It is crucial to document and prioritize technical, operational, and compliance risks according to their impact and likelihood, focusing on the most critical ones first. Implementing rigorous testing protocols, such as automated testing and code reviews, ensures that GAI outputs meet quality and performance standards. Effective risk management, which includes refining AI models and improving user training, facilitates the safe integration of GAI tools into the SDLC, enhancing productivity while maintaining security and quality. Comprehensive documentation of the risk management process helps sustain trustworthiness and reliability of the SDLC and resulting software.
- *Include AI/GAI in enterprise strategy*: Be prepared to meet the goals of the enterprise strategy by conducting a needs assessment. This can be an informal discussion of types of software engineering capabilities that may benefit from GAI. Subsequently, identify a pilot program with limited focus and measurable outcomes. A consistent monitoring and feedback loop is crucial to initial experimentation. This approach will enable an organization to observe and plan for requisite skills development while establishing initial AI governance inclusive of data protection and usage.
- *Deliberately plan for humans in the loop and human ownership of all generated assets*: Regardless of the decisions necessary when designing and delivering software, humans have ultimate accountability. That means humans should immediately verify outputs of any content generated, whether programming code, documentation, or automation scripts. The software engineering workforce must be well prepared to use GAI-assisted tools and yet recognize their limitations. Leading practices are to view the GAI outputs as unverified and potentially flawed. Humans can adopt the mindset of treating GAI

> Integrating GAI-assisted tools into software engineering necessitates a recalibration of expectations and processes to address the human impact.

tools and GAI agents as novices or young apprentices. The GAI-assisted tools will provide some energy and inspiration but must be under constant close supervision.

- *Adopt an assistive mentality*: When integrating GAI into the SDLC, software engineers must adopt an assistive mentality rather than viewing GAI as a replacement. As mentioned, they should view GAI as an assistant or a creative muse that enhances human capabilities. This perspective emphasizes collaboration between AI tools and software engineers. GAI tools can provide support by automating repetitive tasks, offering insights, and suggesting solutions; humans focus on strategic decision making and creative problem solving.
- *Enable safe insulated experimentation*: Enabling safe insulated experimentation is crucial when evaluating GAI-assisted tools for the SDLC. This approach involves creating controlled environments in which potential impacts can be thoroughly assessed without risking the integrity of production systems. Leveraging sandboxed environments, AI red teaming, and rigorous human-in-the-loop testing can aid in identifying vulnerabilities. An insulated experimentation approach enables thorough evaluation and iterative improvement while fostering trust and safety in GAI-assisted software development.
- *Provide dedicated tracking of thought leadership*: Tracking thought leadership on GAI in software engineering is crucial for maintaining competitive advantage and fostering innovation. However, organizations must reduce overall churn and cognitive overload of the employee population. To improve the consumption of high-quality, relevant, and viable information, organizations should allocate resources for continuous learning, curate a central repository of GAI knowledge, engage with industry experts, conduct regular internal workshops, and disseminate information through summaries and knowledge-sharing sessions. They must implement feedback loops with practitioners to gauge learning and challenges. Finally, tracking bleeding-edge technology will benefit from engaging with academic and industry partners, for instance, through an information sharing and analysis center. These steps will assist organizations to stay informed, be nimble, and refine their adoption strategies in this rapidly evolving field.
- *Invest in prompt engineering and prompt libraries*: Today's use of GAI tooling for many software engineering tasks requires prompt engineering, but organizations often overlook the critical role this plays in the effective use of GAI. Organizations must develop specialized prompt repositories tailored to software engineering personas in order to maximize the utility and accuracy of AI-generated outputs. They must also be aware of emerging techniques and tools to reduce the need for excessive prompt engineering using RAGs, knowledge graphs, and vector components.
- *Make sure the organization and SDLC are ready for GAI*: Before integrating GAI into the SDLC, organizations must assess the current state of their methods and processes. If the existing SDLC often fails to meet quality, security, and value objectives, introducing GAI could exacerbate these issues. Organizations should first stabilize and optimize their SDLC to handle the added complexity and challenges that GAI presents. This involves addressing any current inefficiencies and establishing robust practices to support assistive tool integration.
- *Focus on code completion instead of code generation*: Software engineering must focus on code completion rather than full code generation. Current GAI tools are better at translating intentions into precise code snippets and explaining existing code than generating complex code from scratch. To leverage the translation capability, software engineers should integrate GAI-powered code completion tools, prioritize code quality improvements, use GAI for generating code explanations and documentation, provide clear task context, and regularly review AI-suggested code to maintain standards and avoid errors. This approach enhances productivity and code maintainability while utilizing GAI's strengths effectively. It also addresses the near-term challenge of GAI models lacking full context for an entire code base, which can pose IP challenges.

THE ROAD AHEAD

GAI has groundbreaking potential but also limitations and challenges. Today, GAI is a tool designed for use by humans who make up the software value delivery stream. GAI inclusion in the SDLC focuses on use of GAI-assisted software engineering tools. This requires full understanding of the impacts of GAI tooling on human trust; comprehensive, validated, and accounted-for observation; and experimentation. Further, the entire software practice must recalibrate holistic software team performance throughout the evolution of GAI–human teaming. The following sections summarize a set of projected imperatives based on initial findings and research.

Humans first

Positioning humans first means considering the well-being of humans as an

overall imperative. That said, it does not preclude the evolution of roles, responsibilities, and skills. The software industry must begin to focus on team and organizational experimentation rather than individual experimentation.

Trust

Human/machine trust extends beyond human use of GAI-assisted tooling to a concept called *calibrated trust*.[16] Calibrated trust enables organizations to measure appropriate trust of emerging technologies, prevents under- or over-trusting, and enables software-centric organizations to make informed decisions. It will be as important to software engineering and delivery teams as to the ultimate end users and recipients of software value.

From copilots to GAI agents as team members

The evolution from GAI-assistive tooling to GAI agents is currently still at the experimental stage. As the efficacy becomes understood, the ability to infuse the GAI agent autonomously will create unique challenges. In essence, the GAI agent may act as a team member of sorts. Organizations must define specific team responsibilities for the GAI agent; they must understand the human–machine teaming dynamics. Integrating GAI as a team member in the SDLC involves giving the GAI agent autonomy and defining its role within the team. GAI may be able to autonomously handle tasks, which would allow it to operate independently while collaborating with human team members. This setup may maximize efficiency by freeing human team members to focus on complex problem solving and strategic planning.

AI as the team

The next step in AI integration may be the ultimate transition from GAI as a team member to AI as the team, possibly achieved through agentic platforms. These platforms employ autonomous AI agents capable of executing end-to-end software development tasks independently, from requirements gathering to deployment. The transition to agentic platforms will require addressing the challenges and incorporation of many of the recommendations identified in this article. Users of tomorrow's platforms must trust the platforms, and the resulting software must be secure and reliable and meet the mission needs of the end users.

Recent articles and industry announcements for agentic platforms that outline new architectural patterns include AutoDev,[5] OpenDevin, and Nvidia's inference microservices. Emerging research by industry and academia focuses on agentic orchestration platforms. Thus, the software industry is likely near the intersection of no code, low code, and "pro code" (also known as *agentic delivery platforms*.)

This article presented observations from top experts, showcasing the rapid evolution and immense potential of GAI in software engineering. As both research and excitement accelerate, these insights will evolve, and differing opinions will emerge due to the fast-paced changes and the challenges in fully assessing the state of the practice. Researchers, software professionals, and organizations may have contrary opinions given the pace of change and inability to comprehensively assess the state of the practice. Organizations must individually assess and engage with their own SMEs to navigate these developments.

CALL FOR PARTICIPATION

MITRE is leading a collaborative research effort with industry and academic experts. Together, we are conducting further experiments to refine our understanding, implications, and research road map for broader GAI adoption. The study will investigate how GAI affects team dynamics, integrates with workflows, influences long-term sustainability, and affects human roles in software delivery.

To help us meet this goal, please consider participating in our industry-wide survey (https://mitrefedramp.gov1.qualtrics.com/jfe/form/SV_d6m42nHCafJ9UGO), workshops, and/or hands-on team experiment events. Your contributions will be invaluable in shaping the future of AI-driven software engineering. Please contact Tracy Bannon at tbannon@mitre.org for more information and to join our collaborative efforts.

ACKNOWLEDGMENT

This work was supported by the MITRE Independent Research and Development Program.

REFERENCES

1. "Gartner hype cycles, explained." Gartner. Accessed: Mar. 9, 2024. [Online.] Available: https://www.gartner.com/en/articles/what-s-new-in-the-2023-gartner-hype-cycle-for-emerging-technologies
2. AI Index Steering Committee, "AI index report," Stanford Institute for Human-Centered AI, Stanford Univ., Stanford, CA, USA, 2024. [Online.] Available: https://aiindex.stanford.edu/report/
3. J. Rech and K. D. Althoff, "Artificial intelligence and software engineering: Status and future trends," *KI*, vol. 18, no. 3, pp. 5–11, 2004.
4. "Stack overflow developer survey 2023." Stack Overflow. Accessed: 12 Mar. 2024. [Online.] Available: https://survey.stackoverflow.co/2023/#ai
5. M. Tufano, A. Agarwal, J. Jang, R. Z. Moghaddam, and N. Sundaresan, "AutoDev: Automated AI-driven development," 2024, *arXiv:2403.08299*.
6. N. Khanh, A. M. Nguyen, and N. D. Q. Bui, "Envisioning the next-generation AI coding assistants:

Insights & proposals," 2024, *arXiv:2403.14592*.
7. S. Peng et al., "The impact of AI on developer productivity: Evidence from GitHub Copilot," 2023, *arXiv:2302.06590*.
8. A. Ziegler, "Measuring GitHub Copilot's impact on productivity," Communications of the ACM, New York, NY, USA, Feb. 15, 2024. [Online.] Available: https://cacm.acm.org/research/measuring-github-copilots-impact-on-productivity
9. N. Perry, M. Srivastava, D. Kumar, and D. Boneh, "Do users write more insecure code with AI assistants?" in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2023, pp. 2785–2799, doi: 10.1145/3576915.3623157.
10. H. Pearce, B. Ahmad, B. Tan, B. Dolan-Gavitt, and R. Karri, "Asleep at the keyboard? Assessing the security of GitHub Copilot's code contributions," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Francisco, CA, USA, 2022, pp. 754–768, doi: 10.1109/SP46214.2022.9833571.
11. "AI code, security, and trust in modern development." Snyk. Accessed: Mar. 1, 2024. [Online.] Available: https://go.snyk.io/2023-ai-code-security-report-dwn-typ.html
12. W. Harding and M. Kloster. "Coding on Copilot: 2023 data shows downward pressure on code quality." GitClear. Accessed: Mar. 1, 2024. [Online.] Available: https://www.gitclear.com/coding_on_copilot_data_shows_ais_downward_pressure_on_code_quality
13. M. Lee, "A mathematical investigation of hallucination and creativity in GPT models," *Mathematics*, vol. 11, no. 10, May 2023, Art. no. 2320, doi: 10.3390/math11102320.
14. "Federal court rules work generated by artificial intelligence alone is not eligible for copyright protection." K&L Gates. Accessed: Jun. 14, 2024. [Online.] Available: https://www.klgates.com/Federal-Court-Rules-Work-Generated-by-Artificial-Intelligence-Alone-Is-Not-Eligible-for-Copyright-Protection-8-30-2023
15. D. Robbins et al., "AI assurance," The MITRE Corporation, McLean, Virginia, USA, 2024. [Online.] Available: https://www.mitre.org/sites/default/files/2024-06/PR-24-1768-AI-Assurance-A-Repeatable-Process-Assuring-AI-Enabled-Systems.pdf
16. "Calibrated trust | Calibrated trust toolkit." Available: https://www.mitre.org/sites/default/files/2021-11/prs-17-4208-human-machine-teaming-systems-engineering-guide.pdf

TRACY (TRAC) BANNON is Sr. Principal - Software Architecture Leader & Researcher at the MITRE Corporation, Bedford, MA 01730 USA. Contact her at tbannon@mitre.org.

Digital Object Identifier 10.1109/MC.2024.3437463