# Navigating the challenges of AI in software development



Artificial Intelligence (AI) is largely recognised for its transformative potential across numerous sectors, particularly in the realm of software development. As organisations increasingly rely on AI to expedite coding processes, concerns have arisen regarding the long-term sustainability and maintenance of AI-generated code. In a recent episode of the podcast series, What the Dev?, Tanner Burson, the vice president of engineering at Prismatic, shared insights into the implications of AI on software development, outlining both the opportunities and challenges that lie ahead.

Burson highlighted that 2025 is poised to be a pivotal year for businesses grappling with the complexities of upkeeping and enhancing AI-assisted systems. Speaking to SD Times, he noted that the widespread adoption of generative AI and Large Language Models (LLMs) has indeed altered the landscape of software development, citing notable incidences like Google's assertion that 25% of their code is now being crafted or managed through in-house AI.

While these technologies promise increased efficacy, Burson cautioned that many organisations are integrating AI capabilities swiftly and without deep consideration of the long-term consequences on their codebases and development frameworks. Companies may soon face a reality where they encounter challenges with code they do not fully comprehend, raising potential issues of performance, security, and development productivity.

A significant concern involves the existing limitations of AI tooling. Burson explained that current AI systems typically lack a comprehensive understanding of an entire codebase, only analysing current files or a limited number of documents to generate code. This inadequate context can lead to misalignments with business systems and underlying databases, as well as potential neglect of crucial security requirements. Without careful reviews and refinements from knowledgeable engineers, code generated by AI could introduce unforeseen problems.

Addressing the paradox of speed versus long-term sustainability, Burson asserted that the traditional narrative of AI accelerating development may not hold for the maintenance phase. "Most engineers would agree that over the lifespan of a codebase, the time you spend writing code versus fixing bugs, fixing performance issues, altering the code for new requirements, is lower," he stated. He anticipates that the immediate effect of AI adoption will yield maintenance burdens as companies reconcile with codebases increasingly populated with AI-generated segments.

Looking toward the future, he believes there may be enhancements to AI systems, particularly in their ability to incorporate real-time performance feedback to better understand the operational context of the code. However, Burson pointed out that such advancements have yet to materialise into widely operable solutions.

As organisations proceed with AI technologies, Burson identified performance and security risks as the primary challenges. He cautioned against an overwhelming reliance on AI for rapid coding without establishing robust review processes. Traditional methods of code review—typically conducted after code commitment—may need reevaluation. Early reviews will be essential for AI-generated segments to ensure that they align with the overall systems and performance standards.

In a recent panel discussion on AI and testing, a notable observation was made regarding the pitfalls of using AI to generate both code and its subsequent tests. Burson concurred, indicating that without a thorough understanding of the system, assertions about the code meeting its requirements could be fundamentally flawed. The crux of software development remains in problem-solving, tailored to meet genuine customer needs, regardless of who or what writes the code.

In summary, while AI has the potential to revolutionise software development, its integration poses significant challenges that organisations must navigate thoughtfully. Burson's reflections suggest a pressing need for a balanced approach that prioritises code understanding and sustainability, ensuring that innovations in AI do not compromise the integrity of software systems. As the industry moves towards 2025, the conversation around AI's role in coding is likely to intensify, driving companies to reconsider their development strategies comprehensively.

Source: [Noah Wire Services](https://www.noahwire.com)

## Bibliography

1. <https://xbsoftware.com/blog/ai-in-software-development/> - This article discusses the wider adoption of generative AI in software development in 2025, including its benefits and challenges such as faster code generation, code reusability, security vulnerabilities, and technical debt.
2. <https://codeandpepper.com/creating-sustainable-code-with-ai-tools/> - This article highlights the role of AI tools in creating sustainable code through automated code review, test automation, and refactoring, which addresses concerns about code maintainability, adaptability, and scalability.
3. <https://www.dice.com/career-advice/how-ai-will-impact-software-development-in-2025-and-beyond> - This article outlines how AI will impact software development in 2025, including AI becoming the ultimate coding assistant, transforming code maintenance, and handling tasks such as generating code and updating libraries seamlessly.
4. <https://www.dice.com/career-advice/how-ai-will-impact-software-development-in-2025-and-beyond> - It also discusses the future role of developers as AI trainers and the integration of AI in every part of the software development process, anticipating bottlenecks and suggesting architectural improvements.
5. <https://arxiv.org/html/2403.03344v1> - This study evaluates the capability of generative AI models to generate sustainable code, considering metrics such as energy consumption and carbon emissions, and compares the performance of AI-generated code with human-generated code.
6. <https://codeandpepper.com/creating-sustainable-code-with-ai-tools/> - The article emphasizes the importance of AI in maintaining the integrity and sustainability of software through continuous validation of code changes and automated testing.
7. <https://xbsoftware.com/blog/ai-in-software-development/> - It highlights the need for careful reviews and refinements from knowledgeable engineers to ensure AI-generated code aligns with business systems and meets security requirements.
8. <https://www.dice.com/career-advice/how-ai-will-impact-software-development-in-2025-and-beyond> - The article notes that AI assistants will be able to autonomously detect the need for upgrades or security patches and make necessary changes, streamlining the maintenance workflow.
9. <https://codeandpepper.com/creating-sustainable-code-with-ai-tools/> - It discusses the importance of early reviews for AI-generated code to ensure it meets the overall system and performance standards, aligning with traditional code review methods.
10. <https://arxiv.org/html/2403.03344v1> - The study addresses the paradox of speed versus long-term sustainability by evaluating the sustainability awareness of AI models and their potential to contribute to environmental sustainability.
11. <https://www.dice.com/career-advice/how-ai-will-impact-software-development-in-2025-and-beyond> - The article suggests that future enhancements to AI systems could include incorporating real-time performance feedback to better understand the operational context of the code.
12. <https://sdtimes.com/ai/podcast-the-negative-long-term-impacts-of-ai-on-software-development-pipelines/> - Please view link - unable to able to access data