Automated Software Engineering (2024) 31:26
https://doi.org/10.1007/s10515-024-00426-z

# Future of software development with generative AI

**Jaakko Sauvola[1] · Sasu Tarkoma[2] · Mika Klemettinen[3] · Jukka Riekki[1] · David Doermann[4]**

Received: 5 December 2023 / Accepted: 15 February 2024 / Published online: 11 March 2024


**Abstract**
Generative AI is regarded as a major disruption to software development. Platforms, repositories, clouds, and the automation of tools and processes have been proven to improve productivity, cost, and quality. Generative AI, with its rapidly expanding capabilities, is a major step forward in this field. As a new key enabling technology, it can be used for many purposes, from creative dimensions to replacing repetitive and manual tasks. The number of opportunities increases with the capabilities of large-language models (LLMs). This has raised concerns about ethics, education, regulation, intellectual property, and even criminal activities. We analyzed the potential of generative AI and LLM technologies for future software development paths. We propose four primary scenarios, model trajectories for transitions between them, and reflect against relevant software development operations. The motivation for this research is clear: the software development industry needs new tools to understand the potential, limitations, and risks of generative AI, as well as guidelines for using it.

**Keywords** Software development · Generative AI · Real-time digital economy

## 1 Introduction

Software development is an essential enabler of future real-time digital economies. Technical advancements, economic growth, environmental concerns, and societal shifts drive these changes. Software is the engine of modern societies, industries, business sectors, and services in everyday life. The saying from 2011 by Marc

✉ Jaakko Sauvola
jaakko.sauvola@oulu.fi

[1] University of Oulu, Oulu, Finland

[2] University of Helsinki, Helsinki, Finland

[3] Business Finland, Helsinki, Finland

[4] University at Buffalo, Buffalo, USA

Andreessen, "Software is eating the world," turned during 2019 as "AI is eating the software" and now we may ask "Is generative AI redefining how software is being made?" (https://www.forbes.com/sites/cognitiveworld/2019/08/29/software-ate-the-world-now-ai-is-eating-software/). Almost every aspect of life envisions a shift towards real-time digital economies with strict service-level agreements (SLA). This affects the creation, maintenance, hosting, and updating of software, services, and systems. The need for more specialized and technically demanding development skills, tools, and processes is increasing, putting pressure on automation. This frontier requires talented engineers to develop and integrate new technologies, systems, and zero- and no-code platforms, to allow more people without software skills to be productive.

With generative AI, new software industry roles are emerging and some existing roles may be replaced. When assessing the impact, opportunities, and risks, we must consider the target domains, such as legacy, digitalizing the verticals, application economy, real-time environments, and new business models like "selling service over things." The launch of GPT3 in 2020, public availability in 2021, and GPT4 release in 2023 by OpenAI, Google Bard, and others took industry to a rapid transformation. The redefinition of roles, tools, processes, and operations simultaneously affect these changes and advancements. As people create GPT-powered proof-of-concepts (POCs) and services, new innovations appear almost every week. Game developers and graphics creation communities have adopted GPT with integrated multiplatform capabilities, such as AutoGPT models. Many GPT-based cloud services are available for vertical business. Generative AI is seen as a solution to resource, automate, and increase quality in software development. GPTs may free high-level skills for more important tasks, reduce costs, and optimize resources. Many corporations have welcomed this, allowing them to shift engineers to new business models and to improve productivity (Ebert and Louridas 2023). LLMs and their GPT-based services have created a key enabler technology area inspired by millions of people: In this movement, the OpenAI's ChatGPT has been the fastest-growing web application ever (https://www.forbes.com/sites/cindygordon/2023/02/02/chatgpt-is-the-fastest-growing-ap-in-the-history-of-web-applications/).

This paper addresses the potential effects of generative AI and LLM technologies on software development in the context of dense societal, academic, and industry discussions. We present four primary scenarios and assess them in the context of different SDOs. These opportunities and innovations have also raised concerns, challenges, and risks. AI will bring added automation and creativity, but also potential job losses, ethical questions, unclear ownership or IP, and illegal copying and stealing.

## 2 AI is driving automation

The huge growth in the demand for software development has led to the automation of work and processes. The adoption of repositories, cloud services, and co-creation methods, such as Lean, Agile, DevOps, has been key to productivity. AI has further

enhanced the use of assets through automated developer environments, and has been established as a key factor in achieving productivity and growth.

The development and maintenance of software is experiencing accelerated transformation owing to the introduction of large language models (LLMs) and GPT-based services. Software developers adopt co-pilots, interpreters, and private or automated GPT models to optimize tools and workflows. The rapid introduction of these is pressuring existing co-creation processes such as agile or DevOps. Machine learning and NLP aid code discovery, production, review, testing, configuration, and optimization. In automation, simultaneous forces at play affect productivity (Elazhary 2021). Continuous process development has been the answer for decades, with many doctrines, e.g., "write once, copy endlessly," "customer in a loop," "code and test simultaneously," "test-driven development," and "create and share."

In addition to generative AI (creating new data from existing data), other types of AI affects software development, such as supervised learning (labeled correct answers), unsupervised learning (clustering without prior labels), and reinforcement learning (interacting to receive feedback). Although many previous solutions have used supervised learning principles, generative AI is viewed as a game changer in software development because of its cross-language and interpretation capabilities. Currently, tools are available to aid in code discovery, generation, optimization, documentation, testing, debugging, and simulation of various execution models. In the future, these different types of AI will be used in a mix of different roles. For example, by integrating supervised and unsupervised learning, development times can drop dramatically. An example is the automated configuration of systems that learns and adapts to preferences. The use of these AI types collaboratively drives productivity improvements.

For generative AI, recent studies have shown that developer productivity can grow quickly, but this goes far beyond mere tooling or processes (https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/unleashing-developer-productivity-with-generative-ai#/). The growth seems to be especially high (20–50%) in repetitive tasks, updates, development with templates, and ideating and creating "on-the-fly" the first prototypes. These include time-consuming phases, such as generation, UX design, documentation, refactoring, and the discovery of existing codes. However, tasks that require high creativity, complex skills, or orchestration have not produced satisfactory results. To manage generative AI, models and scenarios are required to address these challenges. The next section presents four new primary scenarios on these issues with the respective analyses.

## 3 Scenarios for future of software development

There are already tens of GPT-driven products available and hundreds of them are in development. Collectively, these tools aim to transform traditional roles and phases into dynamic, parallelized flows using variants of LLM models with applications, such as AutoGPT (https://medium.com/@fintankearney/exploring-the-promising-future-applications-of-autogpt-in-software-development-a5c2cde2d776), or to manage ecosystems, such as ChatDev (https://github.com/openbmb/chatdev). New

GPT-based tools have been published daily, such as FuturePedia (https://www.futurepedia.io). These target a vision in which the boundaries between design, coding, testing, and deployment overlap, or even disappear. For instance, as code is written, the testing, documentation, and deployment processes can be initiated almost simultaneously with continuous feedback.

Cloud capabilities enable the handling of parallel tasks while ensuring data integrity and security. Furthermore, GPU acceleration at the edge becomes important for real-time processing, particularly in machine-to-machine- and machine-to-user-centric tasks. This promises a landscape in which efficiency and user centricity redefine how we create and run software and services.

Advances in AI-driven automation have accelerated the evolution of SDOs. As AI becomes more sophisticated, it assumes more tasks, roles, and processes, which are the fundamental building blocks of SDOs. These building blocks have deep contextual interdependencies that continuously evolve in terms of performance, productivity, and automation. Humans must assume new roles, acquire relevant skills and expertise, and be able to design, understand, and control AI systems to create and test new AI models. Furthermore, humans must be able to critically evaluate the ethical, business, and social implications.

Below, we conceptualize four scenarios for software development, each reflecting an interaction between human roles, tools, AI, and processes. These scenarios assume that more automation is desirable. For resource and productivity reasons, they assume that human and AI roles will co-exist, and many different combinations may be used for specific purposes. The scenarios can be formulated as presented in Table 1, using the parameters listed in Table 2.

These primary scenarios S1–S4 can be parameterized to model the scenarios for different types of SDOs. Many real-life operations are transitions between scenarios

**Table 1** Four scenarios for using AI in software development

*Scenario 1: Traditional Software Development Operations*. Humans own all roles, and tools and development environments provide automation. Humans are responsible for managing the process, designing, implementing, testing, and delivering and maintaining products. Tools are used to automate tasks, from code discovery to deployment

*Formulation*: S1: $P_{2-3}\{A(H_1) \rightarrow \{B(H_{2-3}) \rightarrow \{C(T_{1\text{-}3}), D(H_{2-3})\}\}\}$

*Scenario 2: AI in loop*. Humans dominate AI, but AI is beginning to manage larger and more complex work areas. AI is used to automate selected parts of manual and repetitive tasks such as code generation, documentation, testing, and deployment. AI is also used to assist humans in tasks such as design, troubleshooting, and decision-making

*Formulation*: S2: $P_2\{A(H_1) \rightarrow \{B(AI_{3-4} \text{ or } H_{2-3}) \rightarrow \{C(T_{1-2}), D(H_{2-3} \text{ or } AI_{3-4})\}\}\}$

*Scenario 3: AI assumes role(s)*. AI starts to assume selected roles. For example, AI is used to manage the process, design, implementation, testing, delivery, and maintenance. Humans focus on the most complex tasks and control the entire operation, responsible for overseeing that it is working correctly, and producing high-quality results

*Formulation*: S3: $P_2\{A(H_1 \text{ or } AI_2) \rightarrow \{B(H_{1-2} \text{ or } AI_{1-2}) \rightarrow \{C(T_{1-2}), D(AI_{2-3} \text{ or } H_{1-2})\}\}\}$

*Scenario 4: Human-in-the-loop*. AI manages development operations in various roles. Humans oversee and control the process, but their role is focused on overwatches such as operational control, problem solving, quality assurance, and security. AI roles are responsible for automating most or all the other tasks in the development lifecycle

*Formulation*: S4: $P_1\{A(AI_1 \uparrow H_1) \rightarrow \{B(AI_1 \uparrow H_{1-2}) \rightarrow \{C(T_1), D(AI_{1-2} \uparrow H_{2-3})\}\}\}$

**Table 2** Scenario parameters and definitions

| Parameter | Description |
|---|---|
| $H_n$ | Human in a role, where n = 1–4 is skill level, 1 = highest, 4 = lowest |
| $AI_n$ | AI in a role, where n = 1–4 is level of automation, 1 = highest, 5 = lowest |
| $T_n$ | Software development tools and environments, where n points to a level of automation of a tool, where n = 1–3, 1 = automated and 3 = manual |
| $P_n$ | Process and model to operate software development, where n = 1–3, 1 = automated, 2 = semi-automated, and 3 = manual |
| A | Management of software development |
| B | Worker in software development |
| C | Tools used in software development |
| D | Delivery/configuration/update/maintenance |
| ↑ | Supervision or oversight, where the entity within parentheses () is overseen by the entity to its left |
| → | Hierarchical relationship between the entities within braces {} |

and a subset or mix of primary scenarios. Furthermore, we can study these transitions and sub-models using relevant parameters such as efficiency, cost, and quality.

Further research on different trajectories might lead to sweet spots in vertical industries and those that can lead to unfavorable outcomes. To this end, we identified potential trajectories that were discussed between colleagues in the industry (Table 3).

We can also see many interesting sub-models using the S1–4, such as "collaborative partnership", "AI design with human validation", "human-AI role reversal", "decentralized model", "hybrid with external oversight", and "human learning from AI". These models bring interesting adaptations to organizations, proposing

**Table 3** Potential trajectories between the scenarios

1. *Gradual AI integration*. S1 → S2 → S3. This pathway respects the idea that, while AI can enhance many aspects of the software development process, high-skilled human input (such as intuition, creativity, and complex problem-solving) remains invaluable. S3, in particular, harnesses synergy between AI and human strengths
2. *Direct leap-to-balanced collaboration*. S1 → S3. This path utilizes the maturity of certain environments and their ecosystems both with high quality, processes, and roles, e.g., they have achieved high-level CMMI scores, or equivalent level or maturity
3. *Abrupt full automation*. S1 → S4 or S2 → S4. Moving from a human-centric model such as S1 or human-dominant S2 directly to a near-fully automated model such as S4 might lead to overlooked nuances and a lack of creativity. Although AI can automate many tasks, a sudden loss of human insight, oversight, and intuition can lead to unanticipated problems, loss of architectural control, and reduced quality
4. *Incomplete AI integration*. S1 → S2. Transitioning from S1 to S2 and halting the integration process may lead to a suboptimal setup where full potential of AI is used
5. *Overemphasis on automation*. S1 → S3 → S4. Transitioning from S1 to S3 and then regressing to S4 implies that an organization has moved to balanced collaboration but then shifted towards full automation, neglecting the value of human input

improved dynamics and profiles to achieve better productivity, quality, and cost management. However, we leave this to the next phases of our research. The next section analyses some of these scenarios for different SDO cases. For each category, we examined the potential of generative AI and LLMs in four cases analyzing the primary scenarios and their timely transition paths by assessing the benefits for each SDO.

## 4 Analysis of primary scenarios in SDO cases

We used the primary scenarios S1–4 to identify the potential AI adoption for the following SDO cases:

*Case 1: Legacy, maintenance, renewal of existing systems* This is S1 because many of these systems require manual maintenance, often with old APIs, architectures, and even languages. Critical roles for human experts will remain, but GPT and code-interpreter capabilities can help developers understand older code within such systems, debugging, testing, and analysis during maintenance operations, and even create new adapter modules for renewal. Thus, it is likely that S2 takes over in the mid-term.

*Case 2: Clean slate, products without legacy* This is well-suited to S2–S4 due to the absence of outdated architectures and languages. AI tools can be fully leveraged for a broad spectrum of tasks e.g., code generation, debugging, testing, deployment, and maintenance. For larger software operations, S2 is more likely to occur in the short-to mid-term, as AI is increasingly being used in automating repetitive and even customer (DevOps) facing tasks. For simpler applications, we see many offers with zero- and no-code environments, or code generation services. In the long term, S3 and later S4 are more likely as AI tools and their integration into processes matures.

*Case 3: Networked applications and services* These require low latency, high responsiveness, and reliability. S2 is the most likely fit, with AI as a design tool to plan and optimize the complexity, APIs, IOPs, and QoS requirements. Later, transitioning to S3 can bring many benefits for developers, e.g., dynamically adapting to varying configurations, creating customer and language variants, optimizing on-site performance, and performing updates. Ultimately, S4 can be achieved with emerging computing paradigms, such as the network-level edge computing and spatial computing environments. Human roles are primarily in design, architecture, and ensuring integrity, security, and performance. While many NFR or other problems can be pre-empted, human operators are required to intervene during unprecedented issues.

*Case 4: Special SDOs* Initially (S1), this area will demand highly skilled engineers and developers because of high-SLA and NFRs, interoperability challenges, and real-time system interdependencies. When software assets are moved to repositories, platforms, and development environments, a transition to S2 with features inherited from S3 can be expected. Later, AI is used in roles that burden humans and S4 could perform operational tasks, configurations, updates, and recovery, as we already see in some front-runner areas. AI can be used to analyze and correct the behavior of complex, interconnected systems with real-time performance.

Recent studies suggest an interest in using GPT to automate tasks at least partially (Elazhary 2021; https://www.gartner.com/en/documents/4348899). Certain tasks will be automated, and some roles will be replaced by AI or change their focus, and new roles will be needed to orchestrate the development.

## 5 Considerations beyond software development

Generative AI offers tools to renew software development, business models, UX, and engineering. We've seen new business models built with autonomously generated expert content, logic, and NLP narratives. These examples create videos, music, images, avatars, AR, and e.g., limitless game scenarios (and any mix of these) on the fly. Risks must be addressed in business dynamics, liabilities, workforce, security, IP ownership, and ethics as generative AI will integrate deeply into any businesses and their monetization. New roles, such as AI system supervision and optimization, need to be introduced. When demand for special skills is continually evolving, continuous education must provide applicable skills and new ways to adapt (Bull and Kharrufa 2023). Cybersecurity will become a design issue: The AI's capability to produce malware and virus strains requires a new pre-emptive and preventive approaches that must be embedded in the software design phase and as in-built property of any SDO product.

Generative AI brings new ethical dilemmas and intellectual property (IP) challenges. The ownership of AI-generated code remains ambiguous: Who is responsible for the generated end-product? AI-assisted creation demands legally sound guidelines to ensure accountability. From a regulatory perspective, the responsibilities of automatically generated codes and content that might bypass ethical considerations, need to be addressed. Finally, integrating generative AI into software development, business models, operational setups in organizations' dynamics and ethical norms needs new rules.

## 6 Summary

This study examines the role of generative AI in the future of software development and proposes primary future scenarios S1–4, as frameworks for understanding its role and potential. Using these scenarios, we analyzed potential advancements, shift(s), trajectories and also risks for different SDO cases from the industry: legacy, clean slate, networked, and special operations. We also addressed critical challenges and ethical considerations. Generative AI presents promising prospects in multiple levels of SDOs and offers a strategic tool to drive productivity, resource optimization, and cost/time savings. Since the introduction of GPT3 and 4 the adoption speed of the software development community has been phenomenal. Further research is required to examine the implementation of the proposed scenarios in practical SDOs in the industry.

**Author contributions** All authors contributed to the writing and reviewing the paper.

**Funding** Open Access funding provided by University of Oulu (including Oulu University Hospital).

**Data availability** No datasets were generated or analysed during the current study.

## Declarations

**Competing interests** The authors declare no competing interests.



## References

Forbes Homepage. https://www.forbes.com/sites/cognitiveworld/2019/08/29/software-ate-the-world-now-ai-is-eating-software/

Ebert, C., Louridas, P.: Generative AI for software practitioners. IEEE Softw. **40**(4), 30–38 (2023). https://doi.org/10.1109/MS.2023.3265877

Forbes Homepage. https://www.forbes.com/sites/cindygordon/2023/02/02/chatgpt-is-the-fastest-growing-ap-in-the-history-of-web-applications/

Elazhary, O.: Investigation of the interplay between developers and automation. In: Proceedings of the 43rd International Conference on Software Engineering: Companion Proceedings (ICSE '21), pp. 153–155. IEEE Press (2021)

https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/unleashing-developer-productivity-with-generative-ai#/

Medium Homepage. https://medium.com/@fintankearney/exploring-the-promising-future-applications-of-autogpt-in-software-development-a5c2cde2d776

ChatDev Homepage. https://github.com/openbmb/chatdev

FuturePedia Homepage. https://www.futurepedia.io

Large Language Models as Tool Makers. https://arxiv.org/pdf/2305.17126.pdf

Homepage Gartner. https://www.gartner.com/en/documents/4348899

Bull, C., Kharrufa, A.: Generative AI assistants in software development education: a vision for integrating generative AI into educational practice, not instinctively defending against it. In: IEEE Software (2023). https://doi.org/10.1109/MS.2023.3300574