The Impact of Generative AI on Software Engineering Activities

Abstract

Generative AI is positioned to revolutionize software development, with potential far reaching implications for productivity. Generative AI applications leverage Large Language Models (LLMs) to understand language, imagery and code and then use what they learn to generate content, answering questions, organizing multimodal information, and writing text and code snippets.

A McKinsey report published in 2023 [1] reports that Generative AI applications such as ChatGPT and Github Copilot have the potential to enable software engineers to complete development tasks: achieving as much as double productivity over traditional methods. Activities such as inception and planning, system design, coding, testing, and maintenance can all be aided through thoughtful applications of Generative AI. This paper will include an introduction to Generative AI in the software engineering context along with samples of how it can be used. This will be followed by a discussion of productivity impacts and guidance for incorporating them into a software estimate.

Introduction

Artificial Intelligence (AI) is not about "lower cost of prediction" but rather about "enable vastly more productive products, services and organizational designs" [1]. AI is defined as the ability of software to perform tasks that traditionally require human intelligence [2]. AI systems are capable of tasks such as learning, reasoning, problem-solving, perception, speech recognition, and language.

In recent years, software engineering has witnessed a significant shift towards automation; DevOps platforms, Continuous Integration, Continuous Design, Low-Code/No-Code, etc. Generative Artificial Intelligence (GenAI) is the latest and appears to be the most groundbreaking evolution to date. GenAI is a type of AI where people prompt a computer to create new original content. It has recently garnered a huge degree of attention by the introduction of ChatGPT (Generative Pre-trained Transformer) which went from obscurity to over a million users in just five days [3] and over 100 million users in 60 days [4]. Michael Carbin, associate professor at MIT and founding advisor of MosaicLM reported "I feel like, for the first time, I can communicate with a computer, and it could interpret what I meant. We can now translate language into something a machine can understand. I can't think of anything that has been more powerful since the desktop computer."

GenAI leverages Large Language Models (LLM) and Generative Adversarial Networks (GANs) in order to understand language, imagery, and code. After significant training, GenAI applications can answer questions, reorganize information, provide essays and blogs, and write code snippets. GenAI applications are typically built using foundation models that contain expansive artificial neural networks inspired by the billions of neurons connected in the human brain.

GenAI has undergone significant evolution over the years, transforming from basic rule-based systems to sophisticated models capable of generating human-like text, images, and even more complex outputs. The integration of GenAI into software development has been a gradual process, driven by advancements in machine learning and neural network architectures. Examples of GenAI tools used for software development include ChatGPT, GitHub Copilot, Google Bard, Auto-GPT, Amazon Code Whisperer, etc. (Appendix A includes a list of popular GenAI tools (though not inclusive that support various aspects of Software Engineering). This paper discusses the origins of GenAI focusing specifically on software

engineering applications. Following this will be a discussion of the specific ways GenAI can help software engineers increase productivity along with some initial findings on the productivity potentials of using GenAI for software engineering tasks.

Application of Generative Artificial Intelligence on Software Engineering

"Generative AI tools such as ChatGPT have caused a stir among the developer community" says Marko Anastasov, cofounder of Semaphore CI/CD. "Some fear it will take their jobs, while others prefer to ignore it" [5]. At present, GenAI can fill gaps and accelerate implementation of solutions within the software lifecycle, but this does not alleviate the need for software developers to bring the human context of computing to the table. Most experts consider, at least for the short-term and midterm, that human software engineers will collaborate with GenAI tools, with the AI-based tools doing much of the repetitive and tedious software development tasks. The term Generative AI refers to a class of models and algorithms designed to generate new content based on extensive training with existing content on the same or related subjects. These models can create original content such as images, text, audio, and code. The core principles of Generative AI include learning via LLMs, neural networks, video, and the incorporation of probability and randomness. GenAI in the context of software engineering involves the use of AI techniques to automate or facilitate various activities of the software development process. These activities include code generation, testing, debugging and assistance in the design and architecture of a software system. Some of the software engineering activities where GenAI has potential to improve productivity include:

- Inception and planning – models to assist in analyzing, cleaning, and labeling large volumes of data, such as user feedback, market trends and existing systems logs.
- Code generation – models trained on large code repositories can be used to assist developers by writing code, completing code, or making optimization suggestions for existing code. These models build code based on natural language prompts as to what the requirements for the application are.
- Code translation and migration – models trained to translate from one programming language to another or to migrate code from one platform to another.
- Test case generation – Models that create test plans or test cases based on an understanding of the underlying code. These learn from feedback received during the testing process of applications from the training set.
- Documentation – models that review existing code and create documentation to facilitate ease of understanding and maintenance of the code.
- Code Standard enforcement – models that review the code and highlight instances where coding standards and best practices have been ignored or neglected.
- Architecture and design assistance – models that assist in the design and optimization of software architecture through analysis of dependencies between different components. These models may provide the opportunity for simulation to allow for iteration between a variety of design and/or architecture options.
- Maintenance - models trained on system logs, user feedback, and performance data to help diagnose issues, suggest fixes, and predict other priority areas of improvement.

Through thoughtful applications of GenAI, software engineers have the potential to boost productivity, reduce errors, and streamline various tasks throughout the software development process.

Generative AI Impacting Coding Activity Productivity

Generative AI has the potential to improve coding productivity through enhancing efficiency, reducing manual effort, and freeing up developers to do more creative (and fun) development activities. According to Surech Sambandam, CEO of Kissflow, "Just as low-code and no-code will not outright replace traditional developers and software engineers, OpenAI (ChatGPT provider) will provide useful tools that eliminate the repetitive tasks and accelerate time to market for application development. By entering queries in plain conversational language, ChatGPT can automatically generate boilerplate or suggested sample code for problems much faster than any developer can write and experiment with code from scratch" [5].

ChatGPT supports development and translation for many popular programming languages such as COBOL, Java, JavaScript, Rust, C, C++, Python, R, and SQL; it will also support more obscure languages though the developer is warned to accept results with caution and scrutiny. A McKinsey study [1] shows that developers can complete coding tasks up to twice as fast with generative AI. Their research indicates that activities such as documenting code and writing new code for simple functions can see productivity improvements of 50% and the optimization of existing code has shown improvements of nearly two thirds versus completely manual efforts.

The four areas where such productivity potential was observed were:

- Expediting manual and repetitive work by auto filling standard functions and documenting code functionality in a standard format.
- Jump starting the first draft of new code by allowing the developer to get into the programming mode more efficiently.
- Accelerating updates to existing code by having the GenAI tool review the code and make modifications based on prompts from the developer.
- Increasing developers' ability to tackle new challenges as GenAI can help developers brush up on unfamiliar code bases, languages or frameworks. It also provides the option for developers to 'consult' the GenAI tool for advice they might also seek from a colleague or mentor.

It should be noted that the surge in productivity is highly dependent on the complexity of the tasks and the experience of the developers. Several studies have noted that productivity gains in coding are mostly realized when the code is not highly complex. Figures 1, 2, and 3 provide examples of the capability of GenAI to write code, optimize code, and provide documentation for existing code.

ChatGPT

Certainly! You can calculate the average of an array of numbers in Python using the following code:

python — Copy code

```
def calculate_average(numbers):
    if not numbers:
        return 0  # Return 0 for an empty array to avoid division by ze
    else:
        return sum(numbers) / len(numbers)

# Example usage:
numbers_array = [1, 2, 3, 4, 5]
average_result = calculate_average(numbers_array)
print(f"The average of the array {numbers_array} is: {average_result}")
```

This code defines a function `calculate_average` that takes an array of numbers as input and returns the average. The `sum` function is used to add up all the elements in the array, and then the average is calculated by dividing the sum by the length of the array.

Figure 1- Example of code created by ChatGPT

You

Optimize the following code if (Requirements_Stability == "fullySpecified"): #FUlly Specified
if (Operating_Specification < 1.4):
reqStabilityREQ = 0.9
reqStabilityDES = 0.9
else:
reqStabilityREQ = 1.154-0.19*Operating_Specification + 0.012
reqStabilityDES = 1.154-0.19*Operating_Specification + 0.012
elif(Requirements_Stability == "fairlyWellSpecified"): #Fairly Well Specified
reqStabilityREQ = 1.0
reqStabilityDES = 1.0
elif(Requirements_Stability == "somewhatEvolutionary"): #Somewhat Evolutionary
reqStabilityREQ = 0.92+0.3*Operating_Specification
reqStabilityDES = 0.92+0.25*Operating_Specification
elif (Requirements_Stability == "fullyEvolutionary"): #Fully Evolutionary
if (Operating_Specification < 0.8):
reqStabilityREQ = 1.13
reqStabilityDES = 1.13
else:
reqStabilityREQ = 0.81+0.5*Operating_Specification
reqStabilityDES = 0.81+0.4*Operating_Specification

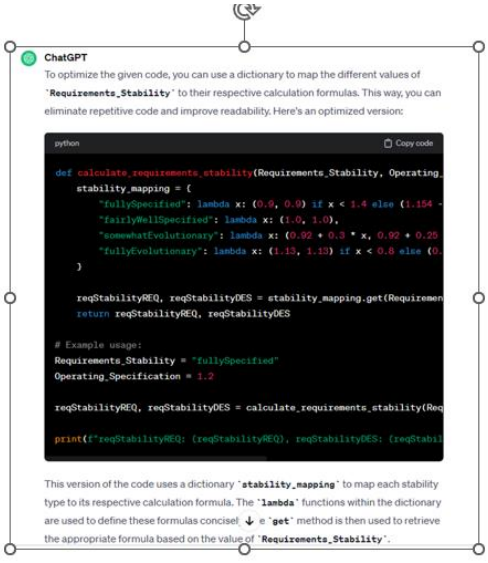

Figure 2- Example of code optimization through ChatGPT

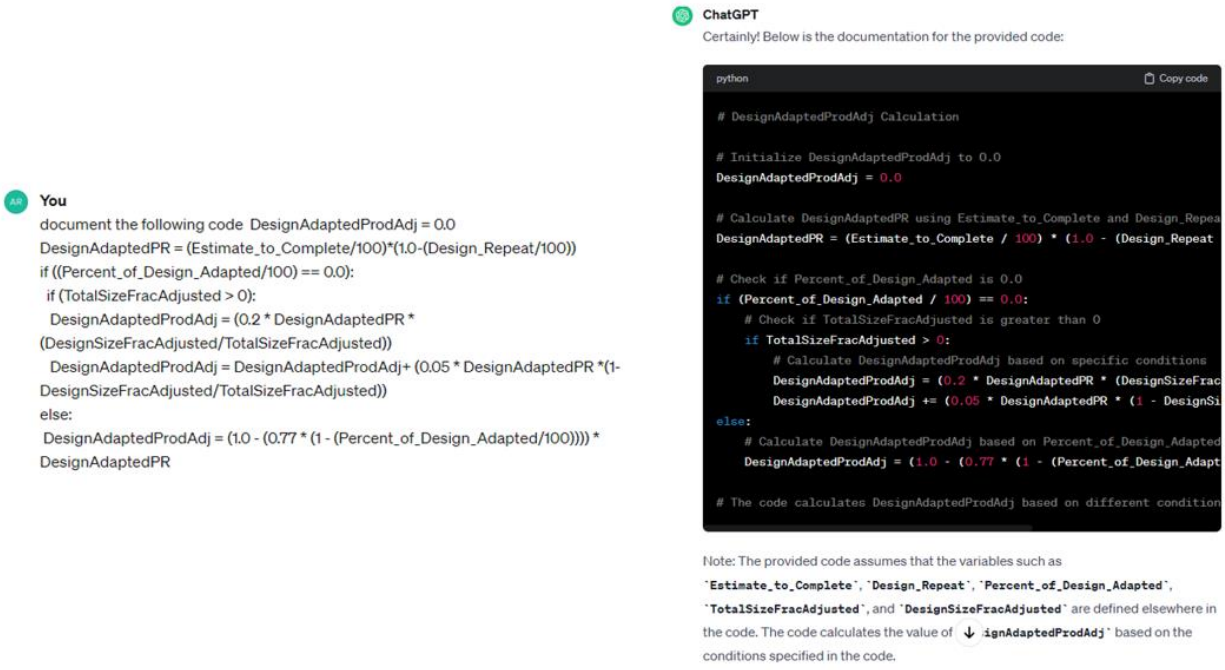

Figure 3- Code documentation using ChatGPT.

Generative AI Impacting Software Design Productivity

Engineers can use GenAI to create multiple architecture designs and then iterate on potential configurations to optimize solutions and accelerate system design. Some of the ways that GenAI facilitates the design process include:

- Early research analysis – researchers can use GenAI to assist in the process of brainstorming and ideation by generating diverse design concepts based on intelligent prompts.
- Rapid prototyping – GenAI tools can assist designers by turning textual descriptions into visual prototypes. This process allows designers to quickly translate their ideas into tangible representations.
- Enhanced collaboration – GenAI can power collaborative design tools that work from natural language inputs. This creates an environment where the design process is transparent and efficient, as it facilitates communication between the developers and the design team.
- Virtual design – researchers can use GenAI to create prompt-based drafts and designs, allowing them to iterate through options to quickly determine the optimal solution.
- Virtual simulations – researchers can utilize GenAI applications to optimize the virtual simulation phase of the design effort.
- Design Automation – GenAI tools can be used to automate the creation of design elements such as icons, buttons, or illustrations; freeing up designers to focus on higher level creative activities.
- Design Review Regulation – GenAI tools can assist in regulating design documents. They can act as a virtual Architectural Review Board, potentially cutting the review process time in half [6].

Figure 4 shows a design query for a user input form for a cost estimation model along with a portion of the code created.

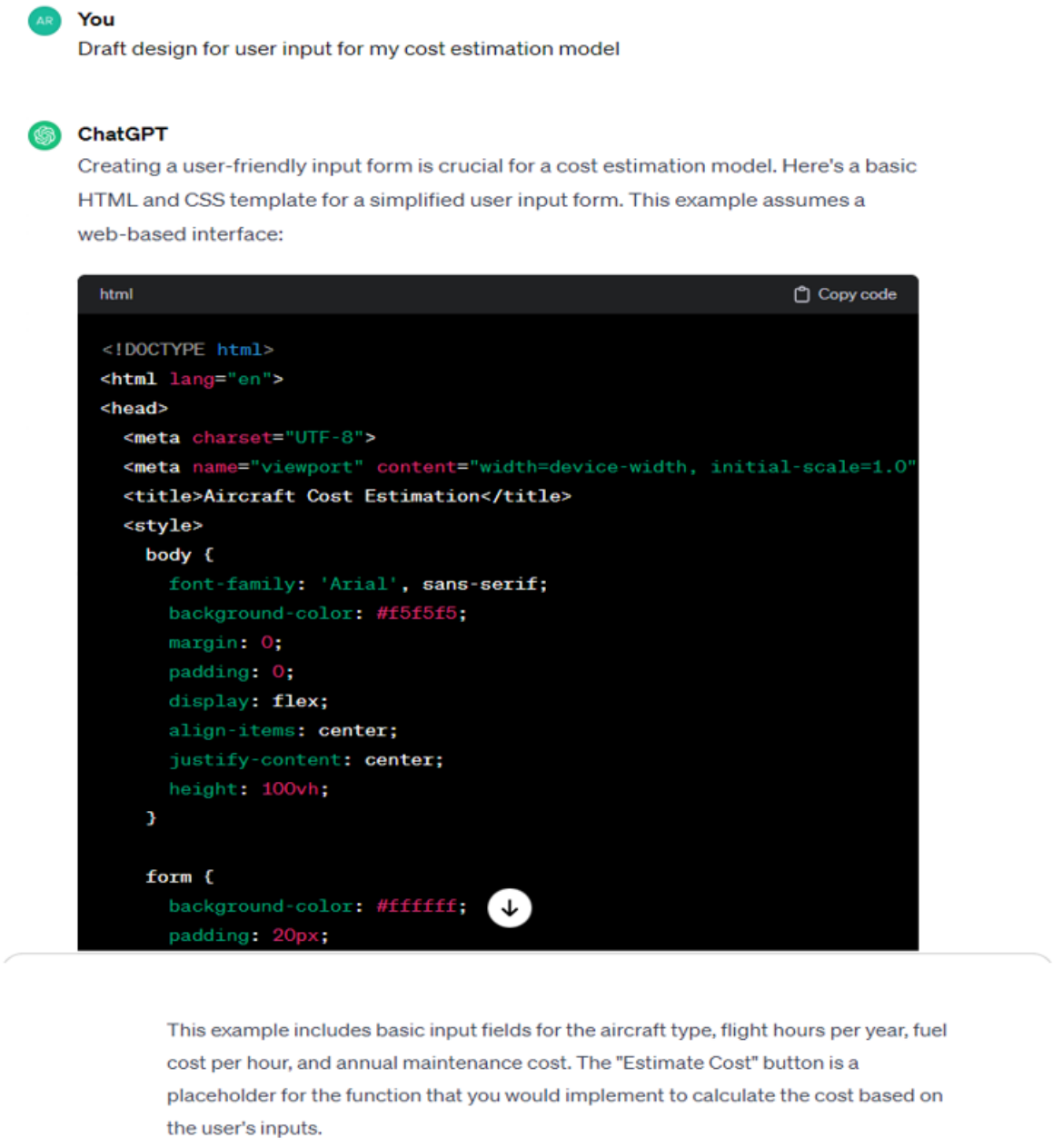

Figure 4 - Design support through ChatGPT for design of an input form for a cost estimation model for aircraft.

While there is some believable evidence that using GenAI for certain coding tasks has specific impact on team productivity, when the tasks are design related, there is less compelling evidence. Certainly, having the ability to automate some simulation and trade off studies will increase team productivity, the human aspect of design, including creativity, intuition, and user empathy remains a crucial element of the design process and cannot be reasonably replaced by AI.

Generative AI Impacting Software Testing and Quality Assurance (QA) Productivity

Generative AI can facilitate the automation of testing, bug detection, and quality assurance; creating opportunities for testers and business analysts to spend more time on the creative aspects of their jobs and less time on tedious tasks. GenAI can be used to generate test plans and test data sets based on prompts from the development team and the application code. Figure 5 shows an example for ChatGPT providing test data for a model that estimates the manufacturing costs of aircraft.

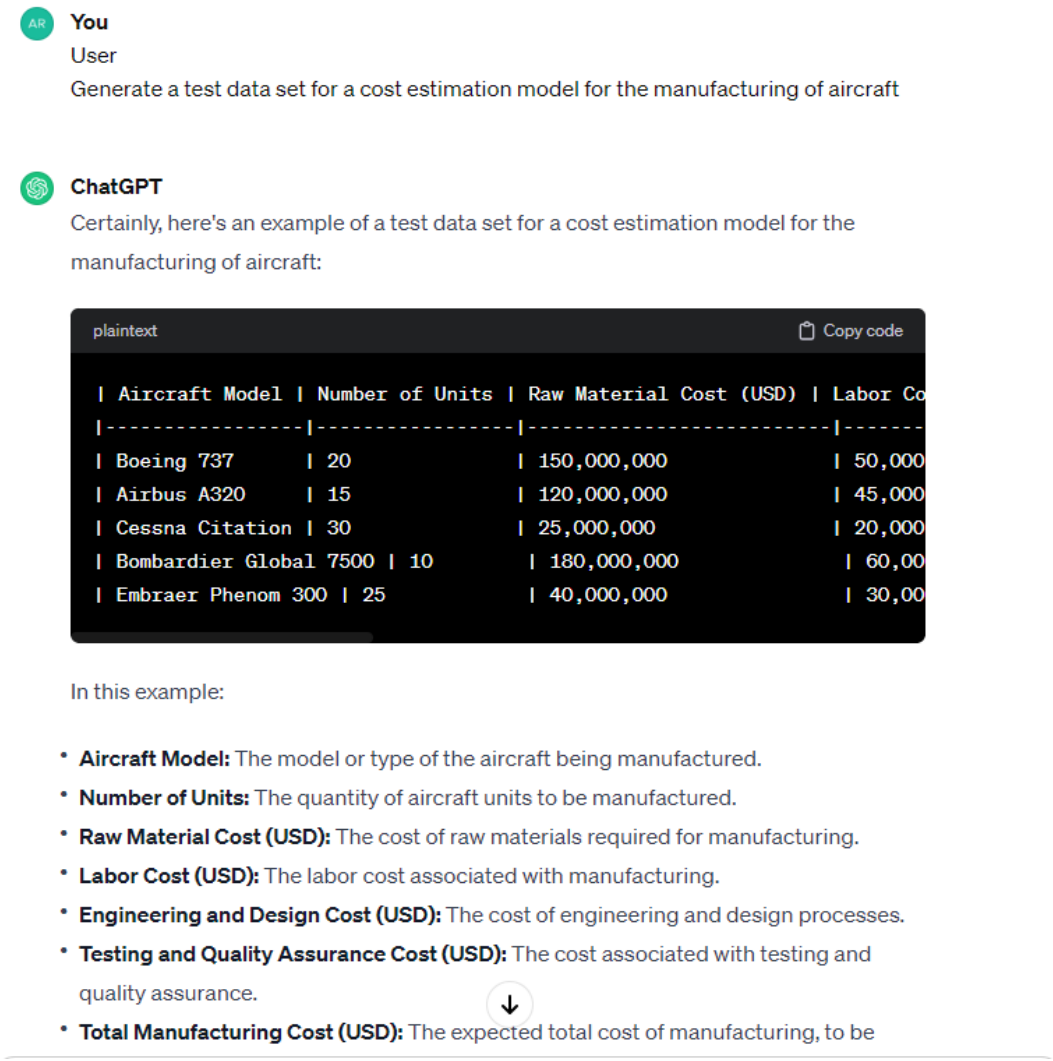

Figure 5- ChatGPT Query for data set for a cost estimation model for aircraft manufacturing costs.

The areas where test team productivity benefits from GenAI applications include:

- Data sets – GenAI applications, when prompted appropriately, can generate test sets for various types of applications being developed increasing test cycles by 25% [6].
- Generation of automated tests – GenAI Applications, when provided with code snippets can develop tests for a test environment such as Fitnesse.

- Test Plans - GenAI applications can develop detailed test plans when provided with the code of the application or with a natural language description of the application.
- Unit test generation – GenAI applications can develop unit tests based on code snippets.
- GenAI augmented test case creation can optimize the number of test cases needed to support development efforts while increasing overall code coverage for better quality assurance.
- For Greenfield development, there can be a 20-40% increase in overall code quality through automatic bug-fixing creating an environment where developers can focus on bug avoidance rather than bug fixing [6].
- GenAI can assist in the creation of test scripts based on the application's code. This helps in automatically generating scripts that are aligned with the codebase, reducing the manual effort. Through Natural Language Processing (NLP), GenAI tools can understand intended behaviors of an application and can use this to generate relevant test cases. GenAI can also assist in keeping automated test scripts up to date based on changes to the codebase.
- Through analysis of code, GenAI applications can review code patterns and historical data to predict potential areas where bugs might occur. This proactive approach helps in identifying and addressing potential issues before they become critical bugs.

Generative AI Impacting Collaboration and Team Dynamics

Generative AI tools and technologies have been increasingly integrated into software development workflows to enhance collaboration within teams. GenAI enhances collaboration within software development teams by automating tasks, providing intelligent assistance, and improving communication. Some of the tasks that facilitate this collaboration include:

- Code generation and auto-completion – Tools like GitHub Copilot or Tabnine provide intelligent code suggestions and auto completions as developers write code. This reduces manual errors while accelerating development time and helping to maintain coding standards.
- Pair Programming Assistance – GenAI tools such as AIXcoder, Bing Chat, and GitHub Copilot support collaborative coding by providing Virtual Pair Programming suggesting code changes, discussing design patterns and offering insights.
- Code review and quality Assurance – GenAI tools such as DeepCode assist in automated code analysis, identifying potential issues and suggesting improvements.
- Natural Language Interfaces – GenAI tools such as ChatGPT use natural language processing (NLP) and TuringBots integrated into collaboration platforms to allow team members to interact and communicate. This can streamline communications and facilitate quick access to information.
- Knowledge sharing and Training – GenAI tools can analyze code bases and generate learning resources or tutorials for team members, supporting knowledge sharing and on-boarding of new team members.
- Continuous Integration and Delivery (CI/CD) – GenAI supports CI/CD pipelines by automating testing processes, identifying issues early in the development cycle and ensuring smooth deployment workflows. This sets the stage for collaborative efforts in maintaining a reliable and efficient development pipeline.

- Team Coordination and Project Management - the integration of GenAI tools into project management platforms assists in task prioritization, resource allocation, and predicting project platforms, enhancing team coordination and decision-making.

Productivity and Cost Implications with Generative AI

A McKinsey study [1] found that GenAI can significantly improve developer speed for some tasks such as code documentation (45-50% faster), code generations (35-45% faster) and code refactoring (20-30% faster). They also found that these productivity gains were limited in cases where the developer experience was less than a year or where the complexity of the tasks were high rather than nominal. The study indicates that in some cases, the tasks took 7-10% longer with junior level resources. The same study indicated that organizations that use more than one GenAI tool realize additional time improvements of 1.5 to 2.5.

Another McKinsey report [7] estimates that GenAI could directly impact current annual spending on the software engineering function by 20-45%. To realize productivity improvements, organizations will need to invest in training for skills necessary to be productive with GenAI tools and possibly transition to new roles as their repetitive tasks are assumed by GenAI. An internal McKinsey empirical study of their software engineering teams found that training on GenAI tools rapidly reduced time needed to generate and refactor code. The engineers also reported better work experiences, increased happiness, and fulfillment.

According to Joe Walsh, principal and technology leader of Launch Consulting "By incorporating GitHub Copilot into Visual Studio code for a recent project – we saw programmers reduce ten-minute tasks, such as writing a small function, down to the 30 seconds it took to simply write out a comment that explains the function. Often these Copilot functions will work out of the box without any need for changes".

There appears to be a lot of evidence that applying GenAI to software engineering tasks has great potential to increase productivity on software projects by automating many of the repetitive tasks developers perform to free them up to do more development – adding features and expanding portfolios. But we need to recognize that the applications of GenAI in software engineering are in their infancy, and while the promise of productivity improvements is enticing, there is not enough industry data to support specific gains or losses at this point in time.

We should be cautious with our understanding as to how project costs are impacted. The high percentages of improvements cited in the sources referenced here apply to the specific software engineering tasks noted. They do not apply to entire development or enhancement projects that an organization may undergo. It is important that the software cost estimator collaborates closely with the development team to understand how and where they will be using GenAI in their projects. As with the introduction of agile development to the software engineer's toolbox, the introduction of GenAI to software engineering tasks needs to be studied and metrics need to be collected so that the models we use for estimation properly incorporate productivity impacts.

Future Trends and Challenges with Generative AI

"ChatGPT and AlphaCode are sure to have an immense impact on how organizations develop applications – from enabling faster and more efficient development cycles to optimize customer experiences over the

next three years" says David Ben Shabat, Vice President of research and development at Quali. "As AI continues to develop, businesses will be able to use these models to optimize customer experiences, increase customer engagement, reduce customer service costs, as well as overall cost reduction" [6].

GenAI will not replace the need for software developers – it will free developers Some predictions about the future of GenAI from this article on BuiltIn [8]:

- to do more of the fun software engineering activities.
- GenAI will democratize software – making it possible to develop more software using natural language.
- GenAI will speed up digital transformation in traditional sectors by increasing access to organizations lagging behind in digitization.

GenAI tools can be a great asset to the software engineering team but there are areas where these teams should be wary. It is important to remember that the commercial GenAI tools available have been trained mostly on large Open-Source codebases, therefore the solutions they generate can be generic rather than meet particular needs. The better the software engineering team gets at prompting, the better code they will generate. Larger organizations are increasingly considering using their own codebase to train GenAI tools to get results that align with their existing portfolios.

McKinsey's study [1] also indicates that organizations are changing the roles to support their AI ambitions in recent years. They are hiring more data engineers, machine learning engineers and AI data scientists while hiring fewer AI related software engineering roles. Prompt engineering seems to be increasing as a role for organizations using GenAI tools on software engineering projects.

Another McKinsey study [7] reports that GenAI has the potential to change the workforce significantly, augmented by the capabilities of individual workers by enabling automation of many of their individual activities. They have estimated that half of today's work activities could be automated between 2030 and 2060, with a midpoint in 2045. GenAI has shifted the balance for AI inspired automation closer to knowledge workers. The percentage of automation for expertise has gone from 16% in 2017 to 49% in 2023. This study also reports that funding for GenAI is significant and growing rapidly, reaching a total of $12 billion in the first five months of 2023.

Forrester has coined the phrase TuringBots which are defined as "AI-Powered software that augments developers' and application development teams' automation and semi-autonomous capabilities to plan, analyze, design, code, test, deliver, and provide assistive intelligence on code, applications, and development processes" [9]. Forrester analysts believe that these TuringBots will become a ubiquitous part of the lives of software engineers and that they have the potential to revolutionize the way we all interact with technology. Examples of TuringBots that are commonly used today include ChatGPT, GitHub Copilot, Tabnine, and others. Forrester's grand vision for TuringBots would include the following capabilities:

- Analyze TuringBots
- Dev-Insights TuringBot
- Design TuringBot
- Coder TuringBot
- Tester TuringBot

- Delivery TuringBot

Some of the challenges that organizations adopting GenAI face include:

- Intellectual property (IP) concerns – Since the training data of many tools comes from all over the Internet, this could create an infringement on copyright, trademark, patents, etc. According to an article in the MIT article on GenAI [10], some organizations are leveraging open-source technology to build their own LLMs, capitalizing on and protecting their own data and IP, and ensuring they are not violating others IP.
- Privacy – GenAI tools can gain and share access to users' input information and output it in a form that makes individuals identifiable.
- Explainability – GenAI tools do not identify all sources of data, making it hard for end users to verify that outputs are correct.
- Security – Bad actors could manipulate GenAI tools to provide malicious outputs.
- Concerns associated with replacing knowledge workers in the software engineering disciplines – while it is clear that GenAI will be replacing many repetitive and tedious software engineering tasks, software engineers should view GenAI as a collaborative partner that assists inwriting better code and reducing errors. The industry will always require smart architects, engineers and testers who get the big picture and have domain specific knowledge.

Case Studies

In a paper published by Cornell University [11], a case study is presented from a controlled trial of GitHub Copilot a GenAI tool that acts in a pair programming capacity to suggest code and entire functions in real time based on context. Programmers were tasked to implement an HTTP server in Javascript as quickly as possible. One group had access to GitHub Copilot and watched a brief training video. The control group did not have access to Copilot but had access to the internet to search sites like Stack Overflow. The First group completed the task 55.8% faster than the control group. There were 95 participants in this study; 50 were placed in the control group and the other 45 were in the Copilot group. Most were between 25-34 with a high education level (4-year degree or more).

McKinsey did an internal study [1] that focused on three areas of software engineering showing that GenAI is likely to improve productivity:

- Code: e.g., build new functionality to elevate the customer experience.
- Refactor: e.g., refactor a piece of code into microservices to improve maintainability and reusability.
- Documentation: e.g., document code capabilities so changes are easier.

The study was conducted over a period of several weeks. Each task was performed by a test group that had access to two GenAI tools and a control group that used no AI assistance. Each participant was in the test group for half of the tasks and in the control group for the next. The areas where GenAI shined in this study were:

- Expediting manual and repetitive tasks such as auto-filling standard functions, completing coding statements as developer type and documenting code functionality in standard format.

- Jump starting the first draft of new code thus eliminating the brain fog that sometimes occurs when faced with a blank screen.
- Accelerating updates to existing code, spending less time adapting code from online libraries and improving prewritten code.
- Increasing developers' ability to tackle new challenges by helping developers rapidly brush up on unfamiliar code, language, or frameworks.

Conclusion

GenAI stands as a transformative force in the realm of software engineering, promising to redefine the landscape of productivity and innovation. The ability of GenAI models, to understand and generate human-like text has led to the democratization of software development, making it something even the 'common man or woman' could dabble with.

One of the significant impacts of GenAI on software engineering productivity lies in the acceleration of code generation and documentation. By providing developers with the capability to effortlessly draft code snippets, generate comprehensive documentation and assist in debugging, GenAI is streamlining and expediting the development process. This not only reduces time spent on routine tasks but also allows developers to focus more on creative problem solving and higher-level architectural decisions.

The democratization of technical knowledge empowers stakeholders with varying levels of technical expertise to engage more meaningfully in the software development process. It promotes effective communication between developers and non-technical stakeholders, breaking down barriers and fostering a more inclusive and efficient development environment.

Many studies and reports tell compelling stories as to how the introduction of GenAI promises to improve productivity to various software engineering tasks. Intuition tells us that they are probably not wrong. It also indicates that productivity will continue to improve if a collection of GenAI tools, with different foci, are used in concert on development projects. Project planners and software estimators should be cautious with how they apply these results to a specific estimate. At present, many tasks that are being automated through GenAI are repetitive and tedious tasks, not more complex tasks. In addition, several studies also indicate that developers with little experience may be less productive when using GenAI tools for software engineering tasks. It is important that estimators collaborate with the development team to understand how and where GenAI tools are being applied.

The advent of GenAI has ushered in a new era of possibilities for software engineering, driving increased efficiency, collaboration, and innovation. As technology continues to evolve, striking a balance between harnessing its potential and addressing associated challenges will be crucial in realizing its full impact on shaping the future of software development.

References

[1] Agrawal, Ajay; Gans, Joshua; Goldfarb, Avi, "Power and Prediction: The Disruptive Economics of Artificial Intelligence", Harvard Business Review Press, Massachusetts, 2022

[2] Deniz, Begum, et. al., "Unleashing developer productivity generative AI", Mckinsey Digital, June 27, 2023, retrieved from https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/unleashing-developer-productivity-with-generative-ai October, 2023

[3] Kalyanpur, Trushant, 2023 Kaggle AI report – Generative AI, august 2023, Retrieved from https://www.kaggle.com/code/trushk/2023-kaggle-ai-report-generative-ai November 2023

[4] McAfee, Andrew, et.al., "How to Capitalize on Generative AI", Harvard Business Review School Publishing, 2023

[5] Sacolick, Isaac, "ChatGPT and software development", Infoworld, Feb 27, 2023, Retrieved from https://www.infoworld.com/article/3689172/chatgpt-and-software-development.html October 2023.

[4] Gnanasambandam, Chandra, et. al., "Yes, you can measure software developer productivity", McKinsey & Company, August 17, 2023, Retrieved from https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/yes-you-can-measure-software-developer-productivity?utm_source=https://www.iceaaonline.com/publications/#newsbrief November 2023

[6] Joshi, Sunil, "Generative AI's Impact on Developers", DevOps.com (powered by Techstrong), September 27,2023, retrieved from https://devops.com/generative-ais-impact-on-developers/#:~:text=Most%20organizations%20believe%20that%20there,development%20and%20higher%2Dquality%20code. November 2023

[7] McKinsey & Company, "The economic potential of generative AI", June 2023, Retrieved from https://www.mckinsey.com/~/media/mckinsey/business%20functions/mckinsey%20digital/our%20insights/the%20economic%20potential%20of%20generative%20ai%20the%20next%20productivity%20frontier/the-economic-potential-of-generative-ai-the-next-productivity-frontier.pdf?shouldIndex=false, January 2024

[8] Maes, Oliver, "What No-Code Can Tell us about the Future of AI in the Workplace", BuiltIn, May 30, 2023, retrieved from https://builtin.com/software-engineering-perspectives/what-no-code-tells-us-about-future-ai-workplaces on January 2024

[9] Vellanki, Shanti, "Generative AI, Revolutionizing Software Development and the Role of Engineers and Leaders", LinkedIn Article, June 16, 2023, retrieved from https://www.linkedin.com/pulse/generative-ai-revolutionizing-software-development-role-vellanki/ February 2024

[10] MIT Technology review Insights, "The great acceleration: CIO perspectives on generative AI", July 18, 2023, retrieved from The great acceleration: CIO perspectives on generative AI | MIT Technology Review January 2024

[11] Peng, Sida, et. al., "The Impact of AI on Developer Productivity: Evidence from GitHub Copilot", Cornell University, arXiv:2303.06590v1, Retrieve from https://arxiv.org/abs/2302.06590v1 October 2023

[12] McKendrick, Joe, "Generative AI means more productivity, but will that help software developers succeed or displace their roles?", ZDNet, November 8, 2023, retrieved at https://www.zdnet.com/article/generative-ais-revolution-in-productivity-are-retrenching-software-developer-roles/ February 2024

Appendix A - Popular GenAI Tools for Software Engineering Tasks

Most studies believe that success with GenAI tools for software engineering tasks is more likely if more than one tool is used. Care should be taken to identify multiple tools that complement one another rather than excel at the same types of capabilities:

- **OpenAI's ChatGPT** is a free app that was instrumental in the recent surge of GenAI awareness. It is able to generate code snippets, translate code to new language, refactor code for optimization, and automate rote tasks.
- **Google Bard** is a free coding assistant that can code in 20 programming languages including Python, Java, C++ and Javascript. It can generate code from prompts, document code that is pasted into its UI, and refactor code.
- **GitHub Copilot** is designed to be an AI pair programmer. It uses OpenAI Codex which was trained on billions of lines of code and is designed to write functional code in languages such as Python, GO, PHP, Ruby, and Swift. It can be integrated into a development environment and thus has access to users' architecture, style, and conventions.
- **Amazon Code Whisperer** is also trained on billions of lines of publicly available code, as well as Amazon code. It is quite similar to GitHub Copilot in capabilities but is optimized for AWS API's such as Ec2, Lambda, and S3 infrastructure types.
- **Tabnine** is an AI coding assistant that uses OpenAI Codex to generate code suggestions. It autocompletes code or functions, matching the style and conventions of existing code based. It is popular because it integrates into a wide variety of development environments.
- **CodeWP** has been designed and trained explicitly to generate PHP, Jave and jQuery compatible with WordPress, its plugins, and its database.
- **CodeSquire** is a tool that brings AI coding assistance to different applications. It uses Chrome Extension to bring its AI features to Google CoLab, Big Query, and Jyupter Lab. Designed for data scientists, it turns prompts into code, writes Structured Query Language (SQL), completes lines of code, and generates new functions.
- **What the Diff** is designed to help teams review code and create better documentation. It can be integrated with GitHub or GitLab repositories. When code is committed, it creates a natural language summary of the differences between the new and old code. This includes an accurate summary of what code has changed and what it has done.
- **AI Query** turns plain English into SQL queries, as well as translate complex SQL queries back into English. It currently supports PostgreSQL, MySQL, MariaDB. And MS SQL Server.
- **WatsonX** is a code assistant powered by state-of-the-art code, only based on Large Language Model (LLM), trained in 115 different programming languages, and 1.5 trillion tokens of code used to recommend code.
- **Microsoft Sketch2Code** generates HTML5 from a hand-drawn user interface sketch.
- **Carnegie Mellon's PolyCoder** performs code autocompletion, generates code from comments, performs pair programming, and unit tests.