



GENERATIVE AI – RECHARGING DEVELOPER PRODUCTIVITY

Abstract

Generative AI has the potential to revolutionize software development by automating tasks, improving code quality, and streamlining workflows across the software development lifecycle. More significantly, it promises to enhance developer productivity by addressing common challenges, eliminating inefficiencies, and assisting with the creation of high-quality and secure code.

This paper examines the typical challenges developers face and how generative AI approaches, including narrow transformers, can help achieve quantitative and qualitative benefits across the software development life cycle (SDLC). It further outlines key Infosys experiments and findings with generative AI tools like GitHub Copilot and GitHub Copilot Chat. It also discusses the importance of comprehensive training and upskilling programs to maximize the potential of any generative AI tool in driving innovation in software development.

Introduction

The growing intricacy of software development demands improved developer productivity. Software engineering and artificial intelligence (AI), two rapidly advancing fields, are now intersecting in transformative ways that will significantly reshape the future of software creation. Generative AI (GenAI), with its ability to produce new content, offers an innovative approach to software creation. Currently, GenAI models are capable of identifying patterns in data and generating new content that mirrors the learned structure. These models extend beyond natural language generation to include the

creation of music, art, design models, and even software code.

Merging generative AI with advanced software engineering can unlock immense potential. Generative AI can accelerate various software development lifecycle (SDLC) activities including code generation, test automation, and bug fixes, thereby significantly boosting productivity. When combined with low-code approaches, a reuse strategy, and a higher degree of engineering and automation maturity, it can create an environment for unprecedented improvements in both code quality and developer productivity.

What Influences Developer Productivity?

Developer productivity depends on multiple factors. Thus, a single metric may not accurately reflect the true productivity of development teams. The SPACE framework (which stands for Satisfaction and Well-being, Performance, Activity, Communication and Collaboration, and Efficiency and Flow), is an industry model developed to enhance the understanding and measurement of software engineering productivity. It can be a relevant lever to consider when evaluating developer productivity in the context of infusing GenAI in the SDLC.

Developers often encounter numerous challenges such as technical, teaming, collaboration, communication, flow, cultural, and workplace issues that may impact their productivity.



Figure 1 | The SPACE framework

Generative AI has the potential to address many of the challenges that developers encounter throughout the SDLC. It can improve productivity, streamline workflows, boost cooperation, deliver high-quality software solutions, and minimize incidents that affect satisfaction, performance, activity, communication, and efficiency.

Apart from increasing output and improving quality, GenAI can also analyze and understand complex patterns, enabling the creation

of intuitive and tailored user experiences. The infusion of GenAI will impact many facets of the value chain, leading to changes in working methods. For instance, AI-powered resource optimization can improve planning and resource allocation. Generative AI can also transform other areas such as enhancing developer skillsets, introducing new positions, and shifting the development focus from coding to strategic planning and design thinking.

Generative AI: Revolutionizing Software Development for a New Era

Generative AI can significantly boost developer productivity by reducing the time spent on tedious tasks, improving code quality, and enhancing end-user experiences. This technology aims to eliminate inefficiencies, reduce waste, and optimize resource utilization as well as overall development paradigms. As the entire value chain fundamentally changes, it will necessitate the creation of new skills and roles across the SDLC.

1. Planning



GenAI improves project management by analyzing historical data, optimizing resource allocation, and developing work breakdown structures. Key activities include data analysis, scenario planning, stakeholder engagement, user story generation, resource allocation, budgeting, timeline development, risk assessment, and documentation.

2. Design



It assists in creating user interface (UI) designs by identifying design patterns and offering prototypes, user experience (UX) design, design optimization, visualization, architecture, and recommendations.

3. Development



GenAI helps in writing and reviewing code, creating tests, and preparing documentation, thereby enhancing overall development efficiency.

4. Testing



It optimizes regression and visual testing by improving test case generation, optimizing tests, detecting bugs, and maintaining test suites.

5. Deployment



GenAI optimizes deployment pipelines, detects issues, and ensures smooth upgrades. Key activities include infrastructure as code (IaC) generation, configuration optimization, deployment strategies, and continuous deployment pipelines.

6. Maintenance



Generative AI can predict necessary maintenance, identify technical debt, and suggest optimizations. Key activities include automated bug detection, predictive maintenance, code refactoring, documentation updates, incident management, and training.

As software applications become increasingly complex and demanding, GenAI tools will become indispensable. For instance, as systems grow more intricate, developers will find it increasingly challenging to maintain code quality, avoid security flaws, ensure compatibility and scalability, and manage deadlines. By providing intelligent assistance and automation throughout the SDLC, generative AI tools can help overcome these challenges, streamline processes, and improve efficiency.



Experiments and Key Observations with GitHub Copilot and Copilot Chat

Real-time, context-aware code suggestions within integrated development environments (IDEs) from AI coding assistants are significantly boosting developer productivity. Key examples of AI coding assistants include GitHub Copilot/Copilot Chat, Amazon Q for Developer, Google Gemini Code Assist, Tabnine, and Replit. These tools not only decrease development lead time but also improve code quality, security, and compliance. They enhance productivity through smart prompts, contextual understanding, and real-time explanations. This allows developers to focus on more valuable tasks like innovation rather than spending effort on time-consuming and repetitive tasks.

Infosys has conducted experiments with GitHub Copilot in several phases, each involving distinct teams, projects, and objectives. Our experiments include:

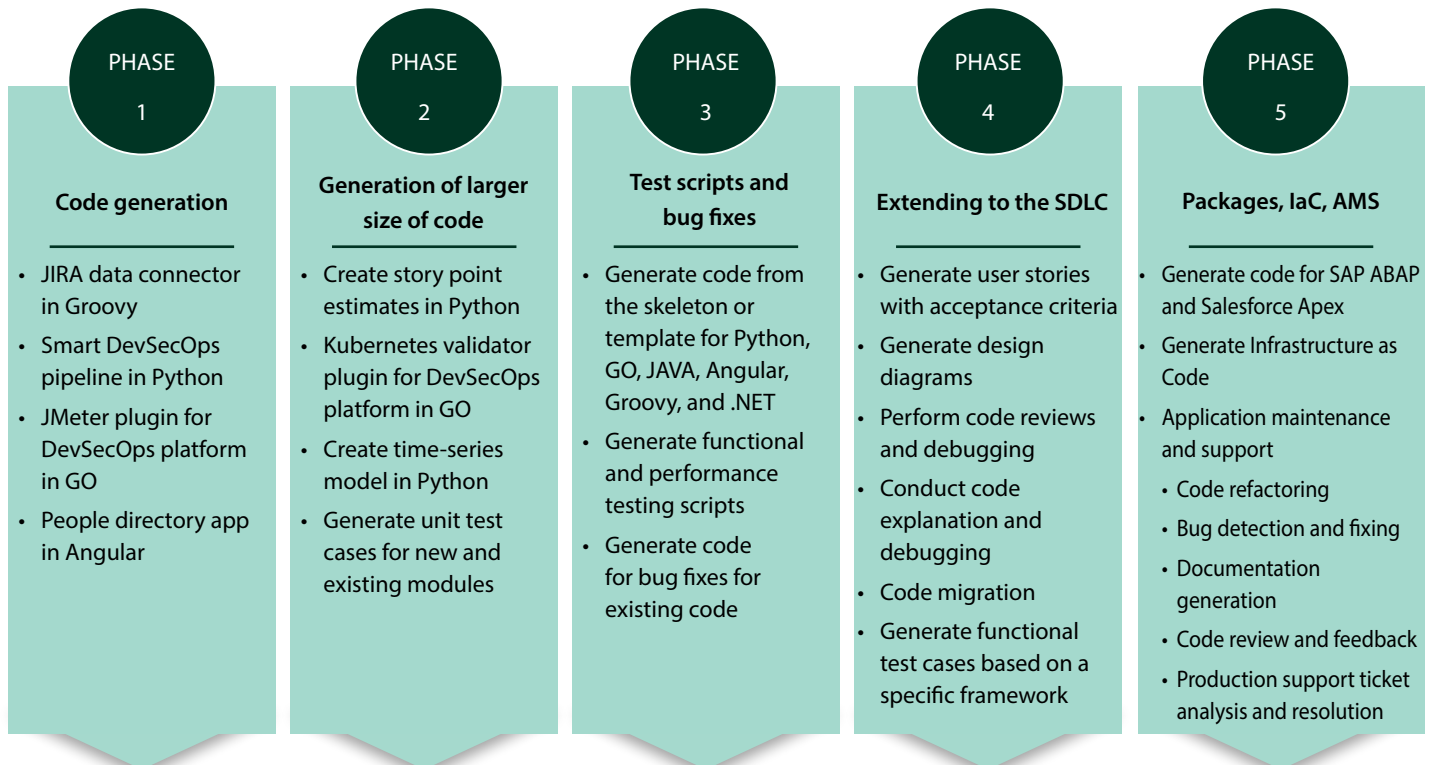
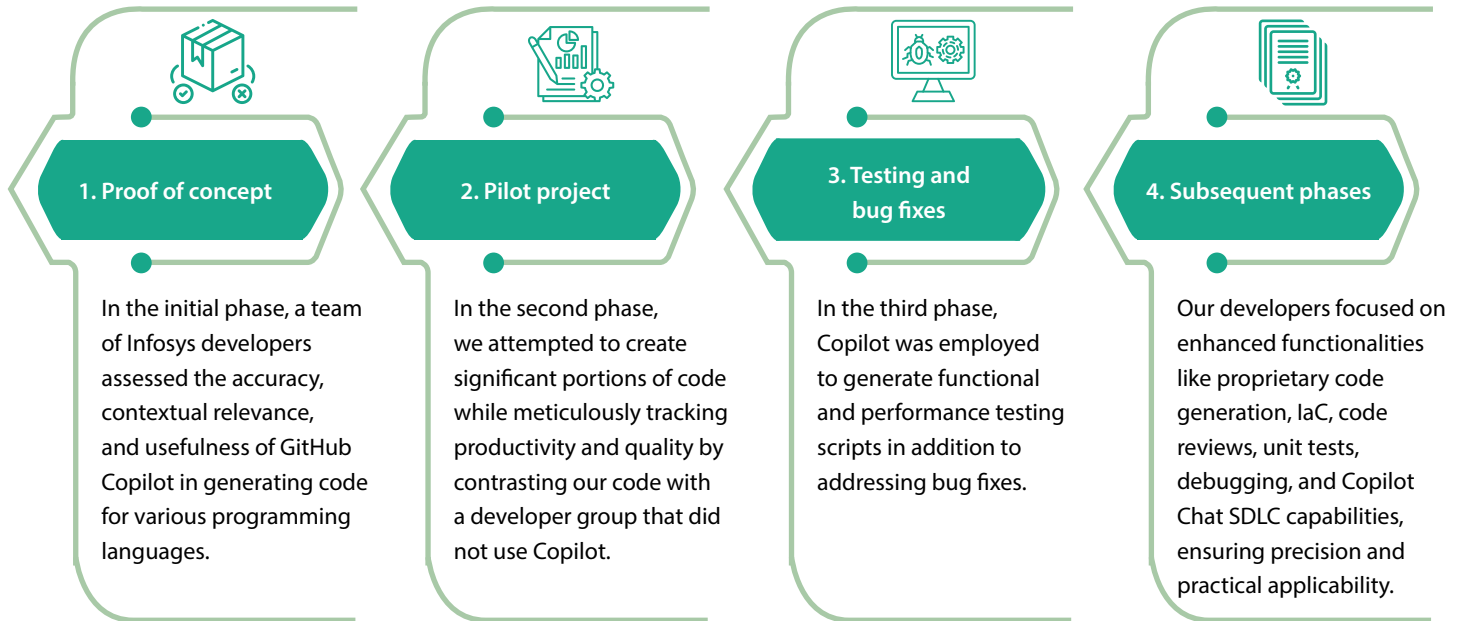


Figure 2 | Phase-wise use cases of GitHub Copilot

Key Observations from Infosys Experiments with GitHub Copilot



Key Observations



Details

Positive outcomes

GitHub Copilot was able to do the following:

- Generate larger code segments
- Produce high-quality components for Python, Go, Java, Angular, Groovy, and .NET
- Generate meaningful unit and functional test cases
- Suggest appropriate code based on existing code
- Suggest improvements and identify potential issues in code, thereby spotting common mistakes and provide inline code suggestions and help explain code snippets when prompted, thereby helping understand code at a basic level
- Assist in code migration by suggesting equivalent code snippets in the target language or framework

GitHub Copilot Chat was able to achieve the following:

- Assist in drafting user stories and acceptance criteria by engaging in a conversational format
- Brainstorm and structure user stories and offer suggestions for acceptance criteria, based on inputs
- Generate design diagrams using Mermaid specifications. The Mermaid design-specific code can be visualized using GitHub or any available Mermaid editor or viewer
- Perform code reviews by highlighting potential issues and offering suggestions based on code snippets. It also helped debug by interpreting error messages and providing troubleshooting steps, offering real-time feedback and suggestions
- Provide in-depth explanations and debugging. It could interpret code, explain functionality, and help identify problems through conversation, offering explanations for code logic and errors
- Provide guidance on migration strategies, help identify potential challenges, and suggest solutions. It also assisted with code translation and adaptation
- Create test cases based on requirements or use cases and suggest appropriate testing frameworks and techniques
- Assist in refactoring by suggesting more efficient or modern code structures and practices. It helped in routine refactoring but may not always identify deeper architectural improvements or context-specific refactoring needs
- Help understand common issues and suggest resolutions when analyzing support tickets

GitHub Copilot and Copilot Chat assisted with the following tasks:

- Generating IaC scripts for tools like Ansible, Terraform, and AWS CloudFormation
- Creating documentation, including comments and explanations for code. It can streamline the documentation process but may require manual adjustments to ensure accuracy and completeness, especially for complex or highly specialized code
- **SAP ABAP** – GitHub Copilot and Copilot Chat were effective in generating boilerplate code and assisting with standard coding patterns
- **Salesforce Apex** – GitHub Copilot and Copilot Chat helped with code snippets, triggers, and general Apex code patterns

Significant effort savings were observed.

No issues were found regarding code quality, maintainability, security, performance, or open-source violations.



Key Observations



Details

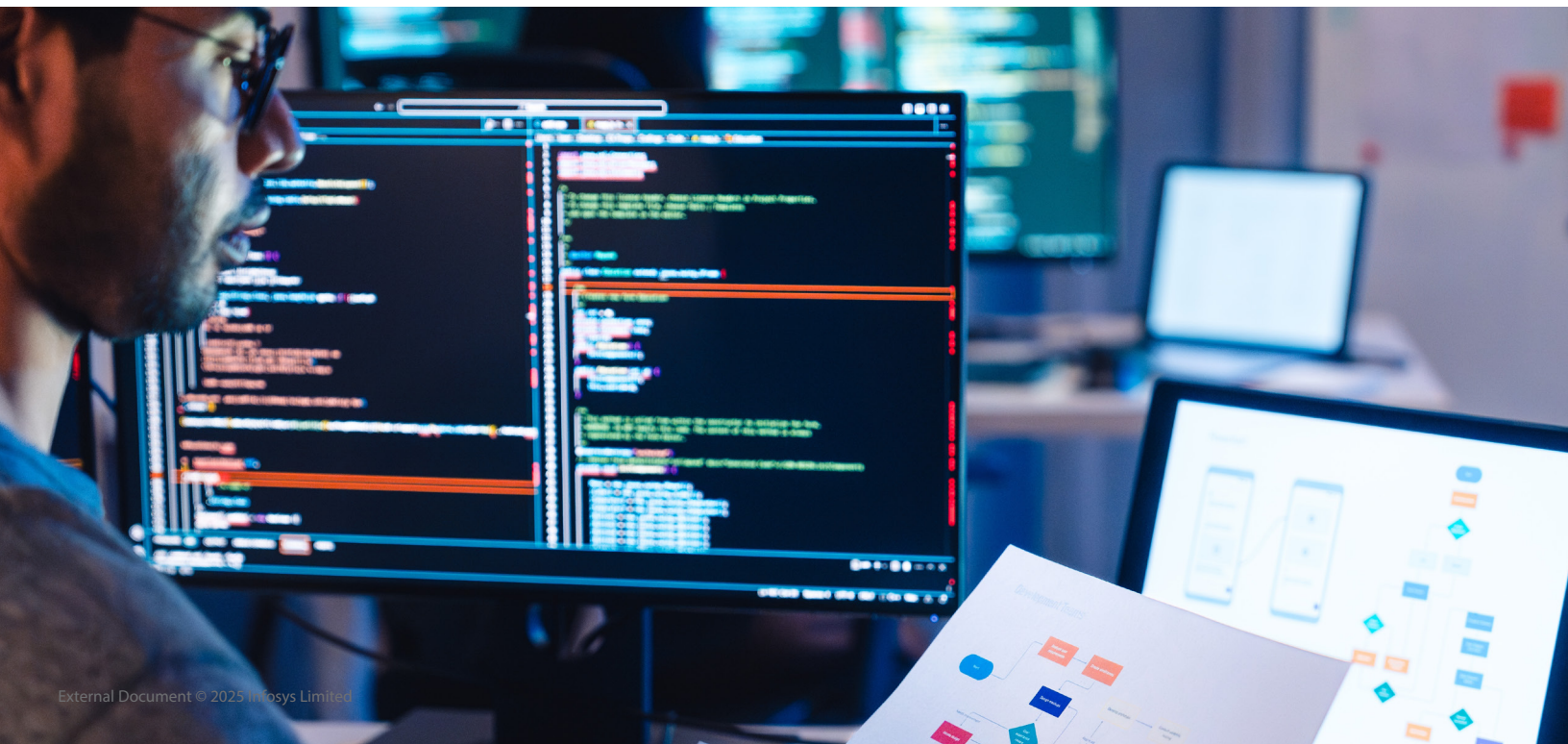
Need for specific intervention

- **GitHub Copilot** requires appropriate context to generate and recommend relevant and meaningful code
- **The user stories** generated may require manual refinement to ensure completeness and precision in acceptance criteria
- **Functional test cases** generated may need a thorough review and adjustment for completeness
- **The documentation** generated might require manual adjustments to ensure accuracy and completeness, especially for complex or highly specialized code

Limitations

- **Limited support** for IDEs
- **Performs best** with languages and frameworks that are well-represented in public code repositories
- **Does not produce visual diagrams** or integrate directly with diagramming tools
- **Code migration** is typically performed at the function, method, or file level. It is not designed to migrate entire codebases in a single operation
- **SAP ABAP specific syntax and business logic** might pose challenges and may require additional input or fine-tuning to manage complex scenarios
- **Handling production support tickets** often requires deep contextual knowledge of the system and its users, which Copilot Chat was not able to fully grasp in complex scenarios
- **Limited context window size**

To summarize, GitHub Copilot and Copilot Chat have proven to be highly effective tools for generating and improving code, providing insightful suggestions, and assisting with various development tasks. However, they do have limitations, such as the need for appropriate context and manual refinements, particularly for complex or specialized scenarios.








Scaling Up with GitHub Copilot: Enhancing Developer Productivity Through AI Pair Programming





To leverage GitHub Copilot effectively, Infosys positioned it as an essential AI pair programming tool and rolled it out across the entire organization. We encouraged our developers to integrate Copilot into their daily workflows and gathered extensive feedback and insights. This data was then used to refine our training programs, establish best practices, and share lessons learned. This ensured that every team member could maximize the benefits of this innovative tool.

Some of the key lessons from our GitHub Copilot implementation are:

1. Implement guardrails for secure usage of GitHub Copilot, covering the following aspects:

				
Technical	Infrastructure	Legal	Responsive AI	Client approvals
Implement policies to filter out public code and ensure no data retention for model improvements	Set up separate infrastructure for Infosys employees and those on client networks	Include clauses on ownership, warranty, and data non-retention for machine learning purposes	Ensure AI regulations, governance, bias fairness, explainable models, and data privacy	Obtain client approvals for the use of GitHub Copilot, ensuring compliance with data privacy, security policies, and contractual clauses

2. Enhance developer success through comprehensive GitHub Copilot training across the following areas:

				
Training program	Knowledge repository	Feedback	Coaching	Custom prompts
To ensure seamless adoption of GitHub Copilot, we implemented an immersive training program tailored to meet the diverse needs of our developers. This program encompassed a series of interactive workshops, hands-on coding sessions, and real-time code review exercises, designed to familiarize teams with the functionalities of Copilot.	Developers were given access to a comprehensive knowledge base, packed with tutorials, best practices, and troubleshooting guides. Mentorship opportunities were also provided, allowing less experienced developers to learn from seasoned professionals through AI pair programming and collaborative projects.	Regular feedback loops were established to continuously refine the training content and ensure that it remains relevant and aligned with the evolving needs of our organization. Through this holistic approach, we have empowered our developers to harness the full potential of GitHub Copilot, driving individual, organizational, and client success.	Under immersive coaching for GitHub Copilot, we introduced coaching programs where Copilot experts provided personalized guidance tailored to individual learning paces. These programs focused on practical problem-solving, enabling participants to apply Copilot capabilities to real-world scenarios, thereby enhancing their coding proficiency and confidence. Additionally, we implemented regular code audits to ensure best practices and continuous improvement.	The Prompt Store provides developers with a rich selection of customized prompts aimed at enhancing their productivity and success. These carefully selected prompts enable developers to address intricate coding problems with assurance and efficacy.

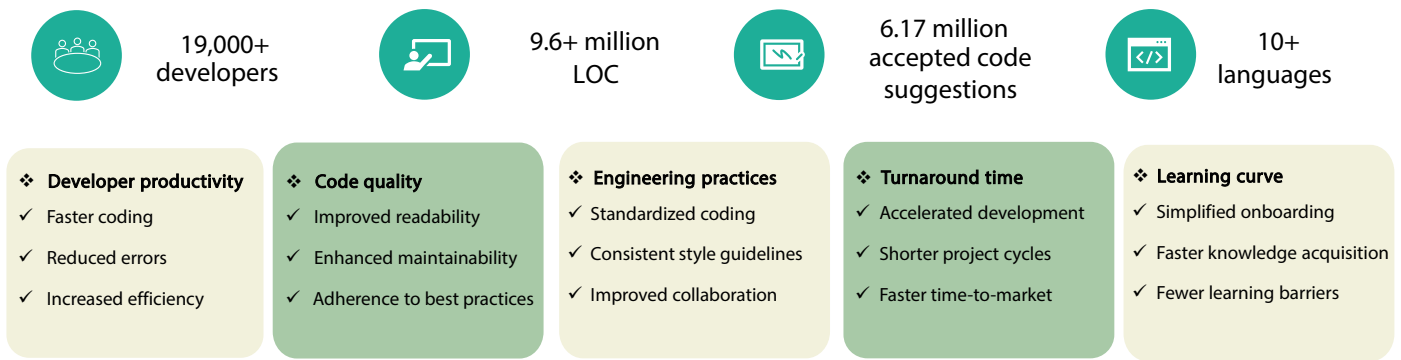


Figure 3 | GitHub Copilot adoption and benefits at Infosys

Harnessing Narrow Transformers: A Tailored Alternative to GitHub Copilot

Infosys has been monitoring the progress of narrow transformers, which demonstrate strong potential when using generative AI. Open-source models often struggle with generating domain-specific content due to their general training data. Conversely, narrow transformers excel at generating highly specialized content. By focusing on domain-specific knowledge, they outperform general models in accuracy and relevance, making them ideal for tailored applications.

Our strategy was to create narrow transformers fine-tuned with in-house proprietary knowledge, starting with an open-source model tailored for our industry.

The Narrow Transformer Solution Approach by Infosys

Curated models



By employing the narrow transformer approach, we have developed specialized AI models derived from open-source large language models (LLMs). These models are tailored to specific industry domains and excel at general coding tasks such as code completion, summarization, and documentation.

Architecture selection



Infosys' proven assessment and architectural selection process includes:

- Choosing the right model – Determine the appropriate model size, evaluate training data size, consider fine-tuning/pre-training options, quantization, and reinforcement learning from human feedback (RLHF)
- Designing the inference deployment architecture – Optimize request batching, enhance inference speed, and ensure observability

Infrastructure



We used the following infrastructure:

- GPU provisioned on-premises (For example, NVIDIA A100 – 40 GB GPU can support a 2 billion parameter model and can handle up to 100 concurrent users, depending on the specific workload and optimization)
- GPU compatible hardware
- Installation cost or cloud provisioning of hardware
- API development, inference tuning, and testing

Custom plugins



We leveraged custom plugins for VS Code, Visual Studio, IntelliJ IDEA, and Eclipse, connected to proprietary open-source models like Codegen and StarCoder. These plugins were fine-tuned for specific tasks such as code completion, generation, and documentation.

Narrow transformers that are developed and fine-tuned from open-source models, based on software programming languages, possess a variety of capabilities. Figure 4 lists out these capabilities.

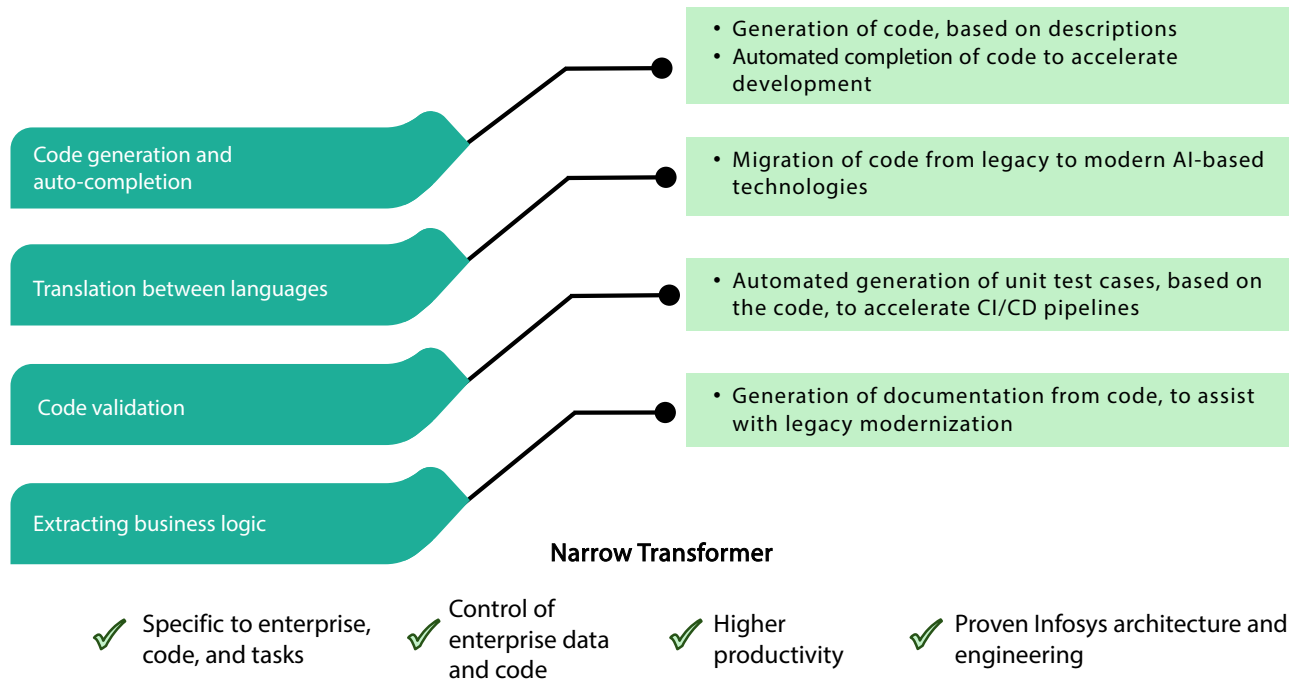


Figure 4 | Capabilities of narrow transformer-based models

Custom plugins for narrow transformers

Custom plugins for popular IDEs can connect to these narrow transformers to then generate high-quality code snippets in multiple programming languages (such as Java, Python, C/C++, and SQL) from natural language descriptions or partial code inputs. Additionally, they can generate unit tests, comments, and documentation for existing codebases, as well as refactor and optimize code for performance and readability.

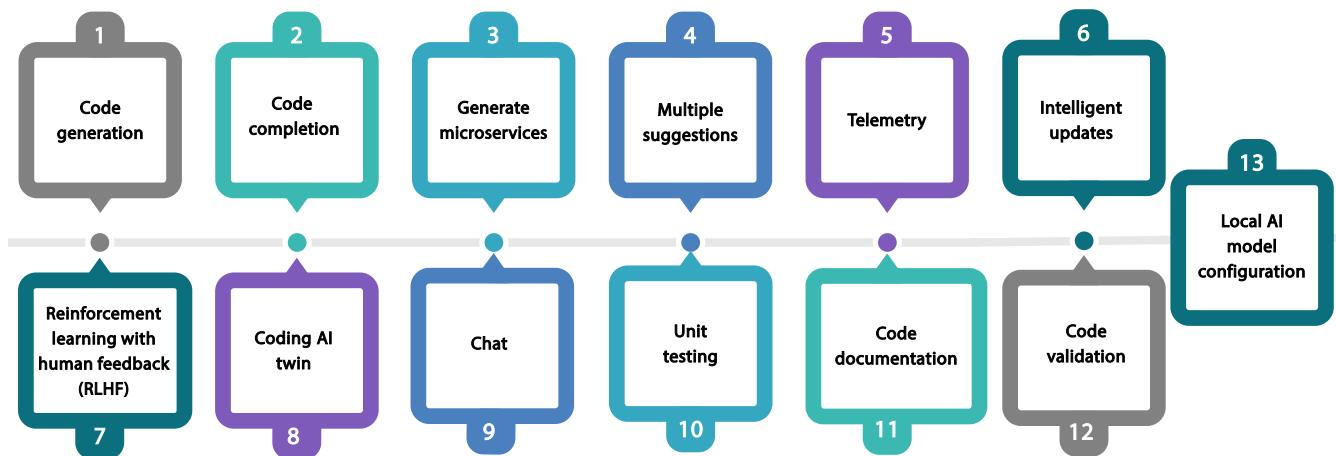


Figure 5 | Key features of IDE plugins

'NT-Java-1.1B' – A specialized model by Infosys

Infosys has created a groundbreaking NT-Java-1.1B Model, which is a tailored language model set to transform Java programming at Infosys. Meticulously researched, developed, and trained, this exceptional NT-Java-1.1B model operates without needing high GPU compute power and delivers impressive results.

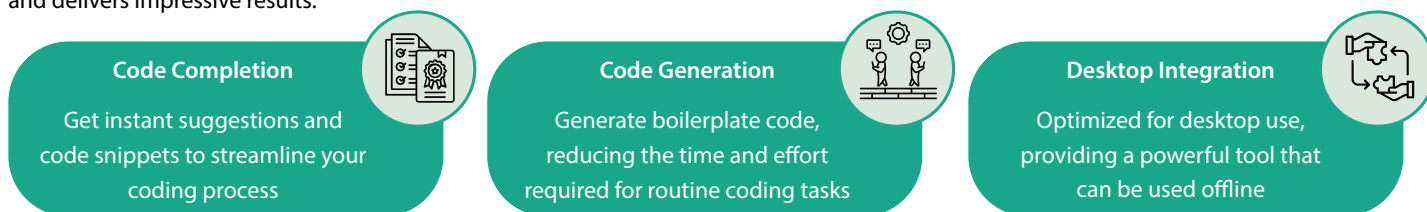


Figure 6 | Key features of NT-Java-1.1B

Infosys Topaz BankingSLM and Infosys Topaz ITOpsSLM

Infosys has introduced innovative small language models (SLMs) like the Infosys Topaz BankingSLM and Infosys Topaz ITOpsSLM. These models leverage NVIDIA AI and Infosys Topaz to create a strong foundation for scalable enterprise AI solutions.

Developed at the Infosys Center of Excellence for NVIDIA technologies, these SLMs use both general and industry-specific data. They are enhanced through NVIDIA AI Enterprise and NVIDIA

AI Foundry in collaboration with Sarvam AI. They are also fine-tuned with Infosys data for integration into solutions like Infosys Finacle and Infosys Topaz for business and IT operations, thereby providing robust foundational models for industry-specific applications.

We also offer these models as services such as pretraining-as-a-service and fine-tuning-as-a-service. These services help businesses develop custom AI models securely and in compliance with industry standards.

AI in Software Development Life Cycle (SDLC)

The integration of AI and generative AI technologies has significantly transformed the SDLC. Established practices now include AI-assisted requirements collection, user story creation, automated code production, unit test generation, code review processes, functional test case and script creation, test maintenance, and optimization. Emerging areas include AI-driven project management, predictive maintenance, and adaptive deployment strategies. Additionally, advancing fields cover AI-enhanced design and architecture, UI/UX creation, application maintenance and support, continuous learning systems for software development, and ethical AI governance within the SDLC.

As GenAI continues to evolve, it is anticipated to further transform traditional SDLC stages, blur phase boundaries, introduce new considerations such as AI reliability and bias mitigation, and redefine the role of human developers in an increasingly AI-supported environment.

Evangelization, Upskilling, and Coaching Programs for Generative AI Tools

To ensure broad acceptance of and expertise with GenAI tools, a multifaceted strategy is essential. This involves educating developers

on the advantages and use cases of these tools, providing resources such as tutorials and workshops, and fostering a collaborative environment through knowledge-sharing and hackathons. Some of the key steps include:

- **Creating and sharing success stories** and case studies that demonstrate how generative AI tools have enhanced developer productivity and software quality across various scenarios and domains
- **Organizing and participating in dedicated sessions**, webinars, workshops, hackathons, and communities of practice to showcase and demonstrate the features and capabilities of GenAI tools while also soliciting feedback and suggestions from users and stakeholders
- **Developing and offering online courses**, tutorials, documentation, and certification programs to help developers learn the fundamentals as well as advanced topics on GenAI along with the ethical and social implications of using these tools
- **Establishing and adhering to guidelines and standards** for developing, testing, deploying, and maintaining software products that utilize GenAI tools while ensuring data quality, security, privacy, and compliance

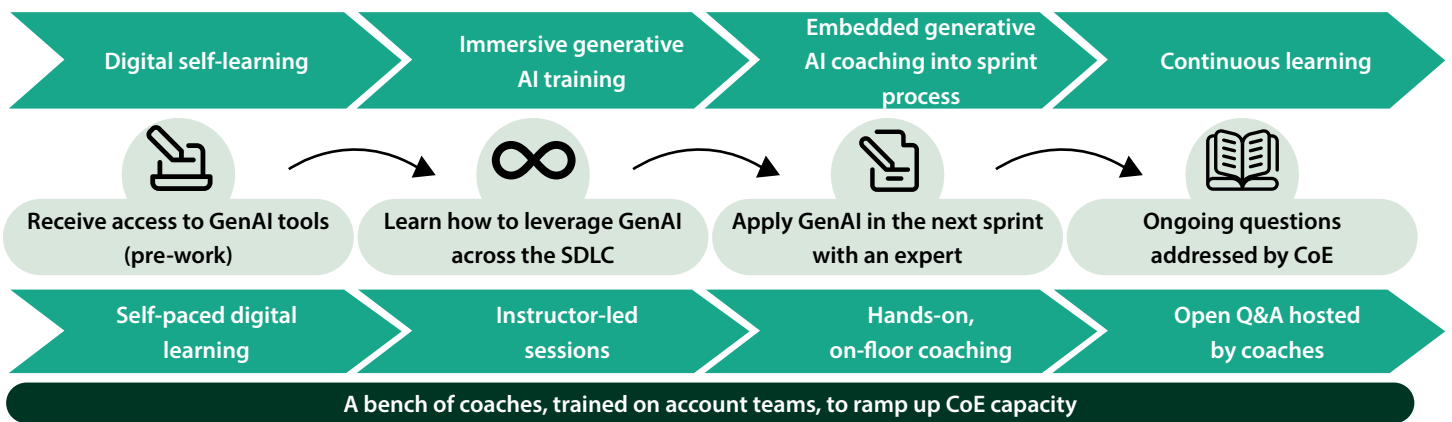


Figure 7 | Generative AI coaching – Upskilling developers for higher productivity levels

Such a strategy can yield higher productivity among trained users through:

<p>Better code quality and acceptance rates: Precise guidance leads to better quality code and higher acceptance rates.</p>	<p>Increased Copilot engagement: Focused training enhances the adoption and impact of GitHub Copilot.</p>	<p>Increased team productivity: Streamlined risk management and prompt strategies accelerate task completion.</p>	<p>Lower error rates: Review of suggestions from GitHub Copilot can reduce errors and debugging time.</p>	<p>Enhanced learning and skill development: Real-world exercises improve daily skills and knowledge.</p>
--	--	--	--	---

Qualitative and Quantitative Benefits of Generative AI in the SDLC

The benefits of GenAI tools are both qualitative and quantifiable. These tools have the potential to revolutionize the SDLC by making it faster, more efficient, and more creative. However, it is important to note that generative AI tools are still under development and should not be seen as a silver bullet. Developers must continue to use their judgment and expertise to ensure these tools are used effectively.

The targeted application of AI/GenAI in software engineering yields significant effort savings. For instance, GitHub Copilot reduces effort by 25-30% on new projects (greenfield) and 15-20% on existing projects (brownfield).

Deliver faster, better, and more cost-effective solutions with the power of AI



Figure 8 | Qualitative benefits of generative AI tools

A Peek into the Future of Software Development

The use of generative AI tools within the SDLC has significantly enhanced automation, intelligence, and collaboration. These advancements empower developers to build superior software products more efficiently. By leveraging these tools, development teams can accelerate innovation, improve code quality, and deliver value to end-users swiftly and confidently.

The future of GenAI holds immense potential to further revolutionize developer productivity. Key opportunity areas include:

- Enhanced code generation
- Intelligent code review
- Natural language interfaces for code
- Ethical AI practices
- Autonomous testing and smart debugging
- Personalized learning and onboarding
- Democratization of software development
- Automated documentation
- Predictive project management
- Agentic AI

These advancements promise to make development cycles more efficient, responsive, and user-centric. The reveal of o1, an impressive reasoning model from OpenAI, signals significant progress in software engineering and developer experience, heralding a new era of human-AI collaboration. The synergy between human creativity and AI capabilities will lead to faster development cycles, more robust and secure software, and a democratized software development landscape.

Conclusion

Generative AI is a game-changing opportunity to empower developers and redefine software development processes. It automates development work, enhances code quality, and manages administrative tasks across the SDLC. It significantly boosts productivity by freeing developers from the repetitive tasks of code writing and routine testing.

Future innovation in GenAI will include tailored development environments and domain-specific AI assistants that can foster more efficient and effective development. This technology has the potential to re-architect the industry and shift the focus from coding to design, thereby enabling business users to become developers. However, skilled developers will continue to play a

vital role as they transition to training AI models and utilizing algorithms to fully unlock the capabilities of generative AI.

As developers enter the new playing field of AI-powered software development, the importance of security and ethical considerations will increase. Through the adoption of GenAI and an open human-AI working model, organizations can achieve accelerated development, enhanced software reliability, and the evolution of a new generation of software development. Generative AI will be a game changer for the software development industry by enabling continuous learning and adjustment, unlocking high potential and addressing daunting challenges.

References

- AI First Software Engineering – Intersection of AI & Software Engineering | LinkedIn
- <https://www.infosys.com/iki/techcompass/generative-ai.html>
- <https://queue.acm.org/detail.cfm?id=3454124>
- <https://github.com/features/copilot>
- <https://replit.com/>
- <https://github.com/features/copilot>
- <https://aws.amazon.com/q/>
- <https://www.googlecloudcommunity.com/gc/Gemini-Code-Assist/bd-p/cloud-duet-ai>
- <https://www.infosys.com/services/data-ai-topaz/offerings/responsible-ai.html>
- <https://www.infosys.com/newsroom/press-releases/2024/launch-small-language-models.html>
- Infosys Unveils Small Language Models – Infosys Topaz BankingSLM and Infosys Topaz ITOpsSLM – Built on NVIDIA AI Stack
- <https://www.tabnine.com/>
- <https://arxiv.org/abs/2310.18648>
- <https://www.forbesindia.com/article/leadership/infosys-is-building-core-ip-around-specialised-ai-models-cto-rafee-tarafdar/87473/1>
- <https://www.youtube.com/watch?v=-mEjti1eCA&t=70s>
- <https://openai.com/index/introducing-openai-o1-preview/>
- <https://github.blog/news-insights/product-news/openai-o1-in-github-copilot/>

About the Author

Srinivasa Sujit Rao

Principal Consultant

About Contributors

Naresh Choudhary, VP - Head - Enterprise Productivity and Engineering Excellence, Infosys

Pradeep Tharmarajan Sivakaminathan, AVP - Senior Industry Principal, Infosys

Amit Gulati, Delivery Manager, Infosys

Pradeep Chandra, Principal Consultant, Infosys

Vasudeva Niranjana Murthy, Principal Consultant, Infosys

Infosys Topaz is an AI-first set of services, solutions and platforms using generative AI technologies. It amplifies the potential of humans, enterprises and communities to create value. With 12,000+ AI use cases, 150+ pre-trained AI models, 10+ AI platforms steered by AI-first specialists and data strategists, and a 'responsible by design' approach, Infosys Topaz helps enterprises accelerate growth, unlock efficiencies at scale and build connected ecosystems. Connect with us at infosystopaz@infosys.com

For more information, contact askus@infosys.com

Infosys[®]
Navigate your next

© 2025 Infosys Limited, Bengaluru, India. All Rights Reserved. Infosys believes the information in this document is accurate as of its publication date; such information is subject to change without notice. Infosys acknowledges the proprietary rights of other companies to the trademarks, product names and such other intellectual property rights mentioned in this document. Except as expressly permitted, neither this documentation nor any part of it may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, printing, photocopying, recording or otherwise, without the prior permission of Infosys Limited and/ or any named intellectual property rights holders under this document.

[Infosys.com](https://www.infosys.com) | NYSE: INFY

Stay Connected   