happiest minds
The Mindful IT Company
Born **Digital** . Born **Agile**

Productivity Improvement Tools in Software Development

Productivity Improvement Tools in Software Development

The rapid evolution of software development has significantly transformed the way teams collaborate and deliver projects. However, traditional software development methods often encounter challenges that affect productivity, including manual coding, inefficient debugging, and time-consuming documentation. These challenges not only slow down project delivery but also increase the likelihood of human errors, leading to rework and additional costs.

To overcome these challenges, productivity improvement tools have been developed to assist software engineers at various phases of the Software Development Life Cycle (SDLC). These tools aim to streamline the development process, enhance collaboration, and reduce the time spent on repetitive tasks.

This paper highlights the challenges faced in traditional software development, key productivity tools evaluated by our team at and discusses how these tools can enhance efficiency.

Challenges in Traditional Software Development

1. **Manual Code Writing:** Developers often spend significant time writing repetitive code and adhering to coding standards, leading to inefficiencies and potential errors.

2. **Design to Code Translation:** The manual translation of design mock-ups into code is a time-consuming process that can create inconsistencies between the design and the actual implementation.

3. **Code Documentation:** Generating accurate and detailed documentation, such as code comments and API documentation, is frequently neglected or rushed, leading to poor maintenance and future scalability issues.

4. **Complex Logic Understanding:** Understanding legacy code or complex logic in existing projects without clear visual representations can be challenging and time-consuming.

5. **Testing and Debugging:** Testing and debugging code to ensure quality and functionality often requires substantial manual intervention, extending the overall timeline for software delivery.

Productivity Improvement Tools

The Our team has evaluated several tools designed to address these productive challenges. Mentioned below are the tools, their capabilities, and how they fit into the SDLC:

1. Continue (Open-Source AI Code Assistant)

Continue AI offers real-time code improvement suggestions directly within the development environment. It focuses on enhancing code quality, providing recommendations on best practices, refactoring, and security improvements.

- **Supported Languages:** VS Code, JetBrains (C#, Java, Python).
- **SDLC Phase:** Software development and maintenance.
- **Use Cases:** Provides auto-complete, reference and chat functions, code optimization, and debuggiassistance.
- **Effort Savings:** 10-15% in daily development tasks.
- **Security:** Collects anonymous usage information for product improvement, with an opt-out option for telemetry.

Benefits: By automating code completion, Continue enhances developer productivity by reducing the time spent on repetitive coding and debugging tasks.
Best for: Continuous code optimization and maintenance.

Recommended Scenarios:
- **Routine Coding Tasks:** Continue AI helps streamline day-to-day coding tasks like writing functions, handling syntax, and refactoring code for improved efficiency.
- **Debugging Large Codebases:** In cases where debugging complex code is time-consuming, Continue AI can assist in finding potential errors and suggesting solutions more efficiently.
- **Enhancing Productivity in Maintenance:** For projects in the maintenance phase, this tool significantly reduces the time spent on code enhancement, debugging, and optimization.

Summary: Continue AI, evaluated as a VS Code extension, provides valuable assistance for explaining and debugging complex code. It also enables users to generate code for specific tasks interactively through its prompt box. As an open-source tool, Continue AI proves to be a practical asset for daily development activities, offering useful features for streamlining coding tasks.

2. GitHub Copilot (AI Code Completion)

GitHub Copilot is an AI-powered code completion tool that assists developers by suggesting real-time code snippets, functions, and even entire lines of code. It integrates seamlessly into popular IDEs, improving coding speed, reducing errors, and accelerating development workflows.

- **Supported Languages:** JavaScript, TypeScript, Python, Java, C#.
- **SDLC Phase:** Software development and testing.
- **Use Cases:** Provides real-time code suggestions, generates blocks of code, and helps with debugging.
- **Effort Savings:** : 20% in development tasks.
- **Security:** Does not store code for future training unless opted in.

Benefits: GitHub Copilot significantly accelerates development by providing context-aware code suggestions, allowing developers to focus on more critical aspects of the code.
Best for: Daily development tasks.

Recommended Scenarios:
- **Real-time Code Suggestions:** GitHub Copilot is ideal for developers working in languages such as JavaScript, Python, Java, C#, and others who need real-time code completion and suggestions.
- **Rapid Prototyping:** It can be particularly useful during the early stages of development allowing developers to quickly create prototypes or generate blocks of code to test functionality.
- **Debugging and Optimization:** Copilot helps with faster debugging by offering potential fixes and code optimizations.
- **Complex Code Generation:** When developers are dealing with frameworks or libraries, Copilot can suggest compatible code that fits the context, making it a great tool for both new and experienced

Summary: GitHub Copilot proves useful in accelerating development by auto-completing code snippets and functions, reducing coding time in common scenarios. Its intelligent suggestions facilitate faster iterations and help reduce errors, enabling developers to catch bugs earlier, potentially lowering defects. Additionally, Copilot's interactive chat window feature enhances coding by allowing users to input commands directly, providing more engaging and responsive assistance.

3. Codeium (AI-Powered Code Completion)

Codeium is an AI-based autocomplete tool that enhances developer productivity by suggesting code snippets, libraries, and best practices. It helps reduce coding errors and speeds up the development process by learning from a developer's context.

- **Supported Languages:** JavaScript, Python, Java, C#
- **SDLC Phase:** Software development and testing.
- **Use Cases:** Offers code suggestions, auto-completes lines, and helps with debugging.
- **Effort Savings:** 20% in development tasks.
- **Security:** Codeium ensures that code is not uploaded to external servers or used for training purposes.

Benefits: Codeium increases efficiency by providing developers with intelligent code completion and ensuring that sensitive data remains secure.
Best for: AI-assisted code generation and optimization.

Recommended Scenarios:
- **Speeding Up Code Writing:** Codeium excels at auto-completing lines of code, generating snippets, and optimizing code across various programming languages like JavaScript, TypeScript, Python, Java, and more.
- **Project Development:** For large development teams working on multiple features or modules, Codeium significantly reduces the time spent on boilerplate code writing, allowing developers to focus on business logic.

• **Refactoring Code:** It is effective in scenarios where you need to refactor code for better performance or readability, helping to keep the codebase clean and efficient.

Summary: Codeium provides AI-driven code auto-completion that helps reduce the time spent on routine coding tasks. It suggests optimized code patterns, highlights potential bugs early, and offers interactive coding support through a chat-based prompt feature, making development faster and more efficient.

4. Code Parrot (UI Code Generation)

Code Parrot is an open-source AI model that provides natural language to code translation. It helps in converting human-written descriptions into executable code, making it a valuable tool for rapid prototyping and automation.

- **Supported Languages:** Angular, React, Typescript, HTML, CSS
- **SDLC Phase:** Design to code.
- **Use Cases:** Generates frontend code from Figma or image files, facilitating faster UI development.
- **Effort Savings:** 15-20% in UI development.
- **Security:** Does not trade user data but stores it for product improvements.

Benefits: By automating the transition from design to code, Code Parrot reduces inconsistencies and accelerates the development of user interfaces.
Best for: Frontend UI development from design to code.

Recommended Scenarios:

• **Figma to Code:** When building a frontend from scratch, especially with existing UI/UX designs in Figma, Code Parrot can quickly generate HTML, CSS, and initial React or Angular components. This is helpful in scenarios where design mockups need to be translated into code rapidly.

• **Rapid Prototyping for Frontend Development:** If you have design files (either Figma or JPG images of mockups), this tool can save effort during the initial stages of UI development.

• **Component Reuse:** In situations where the development team aims to reuse existing components and adhere to coding standards, Code Parrot will fit perfectly by generating code that integrates seamlessly with the existing codebase.

Summary: Code Parrot, as an extension in VS Code, simplifies code generation from Figma files or screenshots (JPG or PNG). It provides step-by-step instructions along with the code required to create component files, HTML, and CSS. Outputs tend to be more precise when using image formats like JPG or PNG compared to Figma files. Overall, it saves significant time by reducing manual coding, and it supports interactive command input through a chat window, enhancing user control and efficiency in UI development.

5. Mintlify (Automated Code Documentation)

Mintlify automates the process of generating documentation directly from code. It helps maintain high-quality, up-to-date documentation by analyzing the code and producing clear, concise documentation aligned with industry standards.

Mintlify

- **Supported Languages:** Python, JavaScript, Java, TypeScript, C#
- **SDLC Phase:** Code documentation.
- **Use Cases:** Automatically generates code documentation such as docstrings for classes and methods.
- **Effort Savings:** 5% in documentation efforts.
- **Security:** Uses customer data for research but allows data deletion requests.

Benefits: Mintlify helps maintain high-quality documentation, saving developers time and ensuring better maintainability of the codebase.

Best for: Automating code documentation.

Recommended Scenarios:

- **Generating Docstrings:** Mintlify is particularly beneficial for developers needing to generate doc strings for functions, classes, or methods across multiple programming languages like Python, JavaScript, TypeScript, Java, and C#.

Summary: Mintlify's free version effectively generates accurate and detailed docstrings for code. However, other features, like API documentation and GitHub integration, can be somewhat challenging due to complex setup requirements, and these aspects lack advanced AI-driven support.

6. CodeToFlow (Code Visualization)

CodeToFlow visually transforms complex codebases into flowcharts, helping developers and teams understand code structures and logic. This tool aids in debugging and system design by providing a clear visual representation of the code.

- **Supported Languages:** Python, C#, Java
- **SDLC Phase:** Low-level design.
- **Use Cases:** Visualizes code into flowcharts and diagrams, simplifying complex logic and aiding reverse engineering.
- **Effort Savings:** 5-10% in design documentation efforts.
- **Security:** No explicit information available.

Benefits: This tool assists teams in quickly understanding existing codebases, facilitating design documentation and troubleshooting complex logic.
Best for: Visualizing code through flowcharts.

Recommended Scenarios:
- **Reverse Engineering:** CodeToFlow is a great choice for teams looking to reverse engineer code to understand existing logic, especially when diving into a new project with complex architectures.
- **Complex Codebases:** It simplifies the process of breaking down large, complex codebases by turning them into visual flowcharts, making it easier to see how different classes or functions interact.
- **Generating Design Documentation:** This tool is helpful during the design phase for creating flow diagrams, sequence diagrams, and class diagrams based on existing code, saving manual design efforts.

Summary: CodeToFlow is useful for generating AI-powered class diagrams and other UML diagrams for individual class files, offering a visual approach to understanding complex code structures. A limitation is that it focuses solely on individual class files, without support for creating diagrams at the project level.

7. KushoAI (AI API Testing)

Kusho AI is an intelligent coding assistant that provides code suggestions, error identification, and refactoring advice. It enhances developer productivity by improving code quality and reducing time spent on repetitive tasks.

- **SDLC Phase:** Unit testing.
- **Use Cases:** Transforms API specifications into test suites, enabling quick unit testing for WebAPI endpoints.
- **Effort Savings:** 10-15% in unit testing efforts.
- **Security:** Collects data for improving its service, with privacy policies in place.

Benefits: KushoAI speeds up the API testing process, allowing developers to ensure quality without integrating test code into CI/CD pipelines.

Best for: API testing and validation.

Recommended Scenarios:

- **Testing APIs Without Writing Unit Tests:** Kusho AI is highly effective at quickly testing APIs without requiring developers to write unit test cases manually. It automates testing against API endpoints without needing to generate code.
- **Rapid API Validation:** For teams working on back-end services or microservices, Kusho AI can validate multiple API endpoints delivering results quickly and saving the effort needed during the testing phase.
- **Manual API Testing:** During the API development phase, before integration into a CI/CD pipeline, Kusho AI serves as a valuable tool for manual API testing, ensuring that endpoints function as expected.

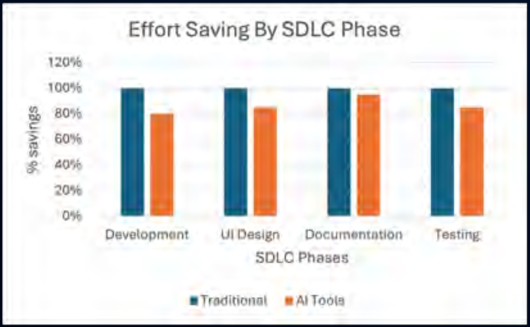
Summary: Kusho AI assists in creating unit test cases for given endpoints through its web-based platform. It generates a comprehensive set of test scenarios, although it does not display the underlying test code. While Kusho AI lacks CI/CD integration, it remains a valuable tool for quickly testing APIs that have been developed.

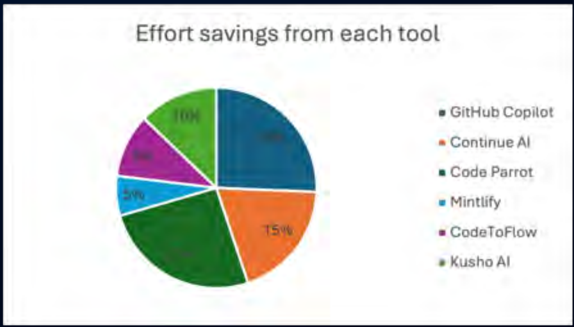
Summary of Recommendations:

- **Daily Development:**
  GitHub Copilot, Continue AI, Codeium
- **Frontend UI Development:**
  Code Parrot
- **Documentation:**
  Mintlify
- **Code Understanding & Reverse Engineering**
  CodeToFlow
- **API Testing:**

Overall, these AI tools enable development teams to focus on more complex, creative, and critical aspects of their projects, while routine and repetitive tasks are automated. This results in faster development cycles, reduced error rates, and improved productivity.

Here are some graphs demonstrating how AI tools help in saving effort during various stages of the SDLC:





Given below is a summary of these tools explaining the scenarios in which they can be used, their licensing costs and the potential effort savings they offer. Additionally, the summary addresses considerations around security and privacy.

Tool Name	SDLC Phase where it can be used	Scenarios/Use Case where it can be used	Open Source or Licensed	License Cost	Effort Savings (%)	Security and Privacy
Continue	Software development and maintenance	Continue AI can be used as an extension in day-to-day development tasks (tab to auto complete, reference and chat, highlight and instruct,), code optimisation, debugging.	Open Source (Apache 2.0 license)	NA	10-15% of effort savings**	Continue collects and reports anonymous usage information to help us improve our product. This data enables us to understand user interactions and optimize the user experience effectively. You can opt out of telemetry collection at any time if you prefer not to share your usage information. https://docs.continue.dev/telemetry https://www.continue.dev/privacy
Code Parrot	Design to Code	It can be used in cases where we want to create UI from scratch using either Figma files or screenshot (jpg files) of mock-ups. It helps in generating HTML, CSS, along with initial React/Angular components. Its accuracy of code generation using Figma is 50-60% however the accuracy grows to 80-85% using jpg files. This has been found while carrying out the POCs.	Licensed	$19 per developer per month	15-20% of effort saving in UI development (HTML, CSS, Components)**	CodeParrot's privacy policy states that it does not sell, rent, or trade user data, including code snippets and project information, to third parties. The platform uses collected data primarily to improve and develop its services. However, the policy does not explicitly state whether the code is used for training purposes. It mentions that data is stored to enable product features and provide access to past work, but users can request the deletion of their data. https://codeparrot.ai/docs/privacy-policy
Mintlify	Code Documentation (specifically)	Rather than going for a full-blown license of Mintlify, we recommend Mintlify doc writer extension for VS code would help developers in generating AI based code documentation i.e. generating docstring for classes.	Licensed	$400/month/project	5% effort saving in generating docstring.**	Mintlify's privacy policy does not explicitly state that customer data is used for training purposes. However, it mentions that personal information may be used for internal business purposes, such as research and development, which could potentially involve data usage for improving their services. Mintlify also emphasizes that they do not sell personal information to third parties, and they offer users the ability to request the deletion of their data. https://writer.mintlify.com/policies/privacy

CodeToFlow	Low level designing	Code to Flow can be used in cases where we need to do a reverse engineering to understand the low-level design of projects/classes. It can use used to generate flow diagram, sequence dig. and class dig.	Licensed	$67 lifetime access	5-10% effort saving in design documentation. **	No information available around its privacy policy.
GitHub Copilot	Development and Testing	GitHub Copilot can be used in day to day development tasks like providing real-time code suggestions, completing lines, generating entire blocks of code, code optimisation, debugging etc.	Licensed	$19 per user per month	20% of effort savings**	GitHub Copilot does not upload your code to a public repository or store it for future training by default. However, it may collect data about your usage, including the code you write to improve the service and the underlying models. GitHub provides an option to opt-out of this data collection, allowing users to control what is shared. https://resources.github.com/learn/pathways/copilot/essentials/how-github-copilot-handles-data/
Codeium	Development and Testing	Codeium AI can be used in day-to-day development tasks like code suggestions, auto-completing lines, and generating code snippets, code optimisation, debugging etc.	Licensed	$19 per user per month	20% of effort savings**	Codeium does not upload your code to its servers or repository for future training. According to Codeium, the code you write while using the tool stays local to your machine, and it does not get stored or sent back for training purposes.
KushoAI	Unit testing	Kusho AI can be used to quickly test various unit test scenarios against the Web API endpoint(s). This tool does not generate the code for the unit tests however it directly gives results. Due to this limitation, it cannot be used to integrate with CI/CD pipelines. Its use is merely limited to unit testing the developed API (without the code base).	Licensed	Pricing details not available on website. Need to contact owner for an enterprise license.	10-15% of effort savings**	Kusho AI collects data, user information https://kusho.ai/privacy-policy

**The effort savings are based on the proof of concept (POC) carried out while evaluating the tool.

Conclusion

Incorporating AI-powered productivity tools into the SDLC can significantly reduce manual effort, mitigate human errors, and shorten project timelines. From code generation to documentation and testing, these tools can address inefficiencies in traditional software development by automating repetitive tasks and enhancing collaboration. Our team's evaluation demonstrates that integrating these tools can lead to significant effort savings, improved code quality, and faster project delivery.

The continuous evolution of these tools, along with their focus on security and privacy, positions them as valuable assets in driving productivity within software development teams. By adopting the right combination of tools for different SDLC phases, software development organizations can significantly enhance their productivity and deliver high-quality software efficiently.

About the Author

Nitesh Chheda brings 16 years of IT experience with a strong technical foundation in Microsoft technologies and Cloud. He has extensive expertise in managing projects across various domains using Agile and Waterfall methodologies. At Happiest Minds, Nitesh serves as a Senior Project Manager in Product & Digital Engineering Services and contributes to the Microsoft Practice vertical.