# An Analysis Method for the Impact of GenAI Code Suggestions on Software Engineers' Thought Processes

**Takahiro Yonekawa**[1], **Hiroko Yamano**[2], **Ichiro Sakata**[2, 3]

[1]Brain Signal, Inc., 27F, Shiroyama Trust Tower, 4-3-1 Toranomon, Minato-ku, Tokyo 105-6027, Japan
[2]Institute for Future Initiatives, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
[3]Graduate School of Engineering, The University of Tokyo, Tokyo, 7-3-1, Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan
yonekawa@bsgnl.com, yamano@ifi.u-tokyo.ac.jp, isakata@ipr-ctr.t.u-tokyo.ac.jp

### Abstract

Interactive generative AI can be used in software programming to generate sufficient quality of code. Software developers can utilize the output code of generative AI as well as website resources from search engine results. In this research, we present a framework for defining states of programming activity and for capturing the actions of developers in a time series. We also describe a scheme for analyzing the thought process of software developers by using a graph structure to describe state transitions. By applying these means, we showed that it is feasible to analyze the effects of changes in the development environment on programming activities.

## Introduction

Productivity improvement measures utilizing generative AI are being introduced in various operations. As an example, implementing an AI assistant to improve customer support productivity can lead to improved employee retention and worker learning (Brynjolfsson, Li, and Raymond 2023). Focusing on software development, a comparative experiment challenged developers to implement small-scale concrete applications as quickly as possible using GitHub Copilot. The results showed that the applications were completed more than 1.5 times faster. Experimental evidence supports the ability to generate code of sufficient quality quickly in software development (Peng et al. 2023).

When encountering a problem that cannot be resolved by a programming language textbook, one typically searches for relevant resources based on search engine rankings and reads their descriptions. Occasionally, example code is copied and modified as a prototype for one's own code. The availability of generative AI, particularly ChatGPT (OpenAI 2023) as an external resource, has altered the process of programming activities. Interactive generative AI tools respond to prompts with corresponding output. During programming activities, a capture program can record operations such as pasting generated code into a code editor, such as Visual Studio Code, by acquiring active application windows and keyboard/mouse actions. This research presents a framework for examining which parts of the programming process an interactive tool like ChatGPT can improve.



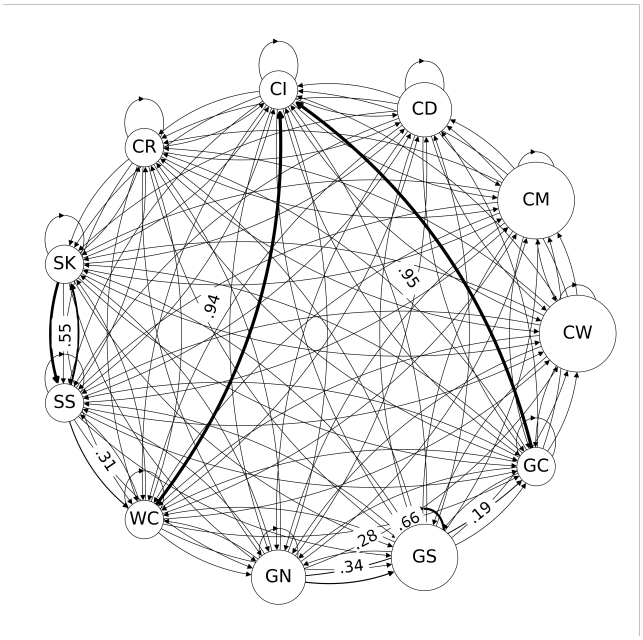


Figure 1: Graph structure of state transitions in programming activity

## Method

Table 1 shows the status of programming activities. The output of the capture program can be collected, including the instance ID of the target application window, activity status, and text exchanged via clipboard, with timestamp. By analyzing the pattern of state transitions, the developer's higher-level operational intentions during programming can be determined. The software developer's activities can be obtained chronologically, including the prompt content, use of obtained code, code functionality, and existence of subsequent prompts.

Figure 1 shows a schematic image representing the state transitions of the programming process. Each node in this graph structure is labeled by the state code in Table 1. The attribute variables of each node include the instance ID of the application window, the elapsed time spent at the node, and the text body exchanged. The weight of the directed edge

| State code | Action state | State of thought | Active window |
|---|---|---|---|
| CW | Manually typing code | Generates logic in code form directly from thoughts. | Code Editor |
| CM | Manually editing code | Modifies existing code based on new insights. | |
| CD | Manually deleting code blocks | Deletes previously written code after review and consideration. | |
| CI | Inserting code blocks from clipboard | Appends previously stored code in external memory to the current codebase. | |
| CR | Replacing code blocks with those from clipboard | Replaces existing code with previously stored code in external memory. | |
| SK | Conducting a keyword search | Thinks of keywords to generate a list of search results. | Search Engine |
| SS | Selecting a search result | Reviews search results and selects relevant information. | |
| WC | Copying code blocks from a website | Identifies useful code snippets for external memory and subsequent use. | Website |
| GN | Submitting a new prompt to GenAI | Crafts complex prompts combining natural language and code for GenAI interaction. | GenAI Dialog |
| GS | Submitting the subsequent prompt to GenAI | Creates prompts based on GenAI interaction history for further knowledge and code. | |
| GC | Copying code blocks from GenAI output | Selects relevant GenAI responses for code writing and preparation. | |

Table 1: State of developer behavior in programming activities

represents the probability of the state transition frequency. This state transition pattern reflects the characteristics of the software developer.

The node sizes and edge weights in Figure 1 show examples of screen transitions when the author coded a small-scale web application with references to both ChatGPT and the Google search engine. The edge weights indicate transition probabilities, and the node radii represent dwell times.

An example of the typical programming process starts from entering a prompt using GenAI (GN), copying the resulting code to the clipboard (GC), and pasting it into the application code (CI). In this case, based on author's trial, the transition probability is estimated as .28 if the desired code is found immediately, .34 if the prompt is rewritten several times. If the prompt was rewritten, the transition probability of interacting with GenAI by adding a subsequent prompt (GS) is .66, and the transition probability of finding the desired code and copying it to the clipboard (GC) is .19.

The framework we propose captures the thought process of software developers for each purpose and situation of software development behavior, such as per developer, per target software category, per software development phase, etc. By analyzing changes in the network structure of thought, it will also be possible to investigate differences between cases in which GenAI is introduced and cases in which it is not.

## Discussion

This research introduces a data collection framework designed to allow GenAI to quantitatively assess shifts in software development activities, with the goal of improving working conditions for developers by addressing productivity and stress challenges. It discusses how GenAI can explore the relationship between coding and cognitive processes, using both GenAI and traditional web resources.

At present, the network graph values are based on the programming trial of the author using the developed framework. The state of thought associated with each action state in Table 1 reflects the developer's intention at that moment based on the observable operations. The validity of these thoughts should be examined in detail from the collected subject data.

The research suggests that despite these challenges, the framework could offer targeted recommendations for enhancing programming practices by analyzing the state transition network model during code development. This approach aims to continuously improve and provide programming support in areas that require focused efforts. For instance, it encourages more efficient and accurate programming behavior. By tailoring action states to specific business scenarios, GenAI's application could have broad utility and demonstrate its impact on software development. The monitoring of changes in the state of thought during the programming process, as observed by the proposed framework, may contribute to the psychological stability or well-being of programmers.

## References

Brynjolfsson, E.; Li, D.; and Raymond, L. R. 2023. Generative AI at work. Technical report, National Bureau of Economic Research.

OpenAI. 2023. GPT-4.

Peng, S.; Kalliamvakou, E.; Cihon, P.; and Demirer, M. 2023. The impact of ai on developer productivity: Evidence from github copilot. *arXiv preprint arXiv:2302.06590*.