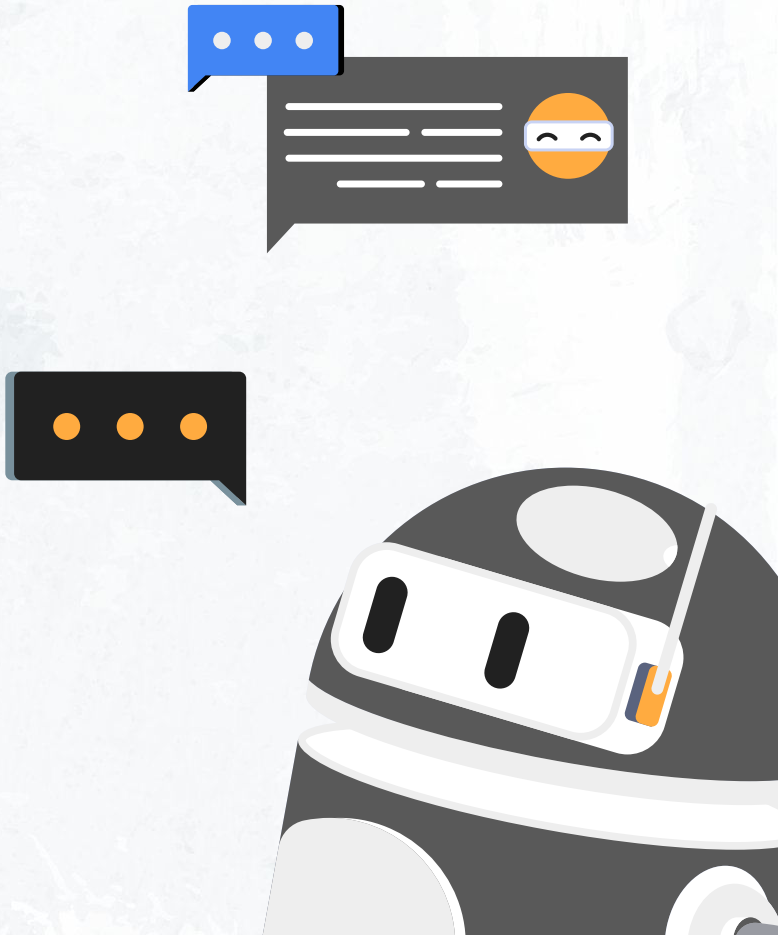
# A Large-Scale Survey on the Usability of AI Programming Assistants: Successes and Challenges

**Authors:** Jenny T. Liang, Chenyang Yang, Brad A. Myers

**Presenter: Shawal Khalid**



# Table of Contents

# Introduction

- The recent proliferation of AI programming assistants like GitHub Copilot and ChatGPT has revolutionized software development.
- AI programming assistants are recognized for their capability to provide quality code suggestions.
- **Paper released by Mar 2023 (version 1)**
  - **2nd version on Sept 2023**
- **OpenAI introduced ChatGPT in November 2022**
- **OpenAI introduced ChatGPT in February 2023**

# Motivation

**Reasons for low adoption:** Developers express concerns about generated code potentially containing defects, not adhering to project coding styles, or being difficult to understand.

**Lack of systematic investigation:** There hasn't been a systematic investigation into usability factors related to AI programming assistants, prompting our research to address this gap.

# Introduction & Motivation

## Investigate Framework

### Usage Characteristics

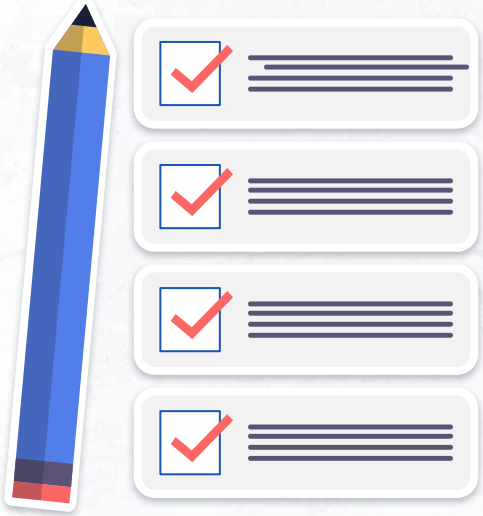
- Usage Pattern
- Motivation for using
- Motivation for not using
- Successful use case

### Usability of Application Assistants

- Usability issues
- Understanding outputting code
- Evaluating outputting code
- Modifying outputting code
- Giving up outputting code

### Additional Feedback

# Methodology

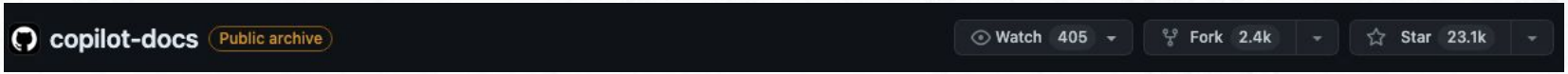


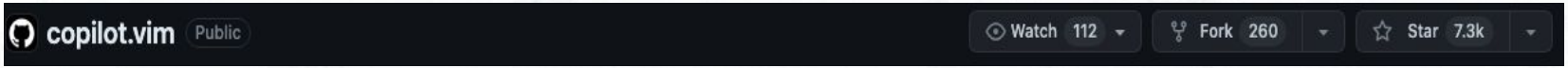
## ★ Sampling

The sampling strategy outlined above involves selecting participants from GitHub repositories.

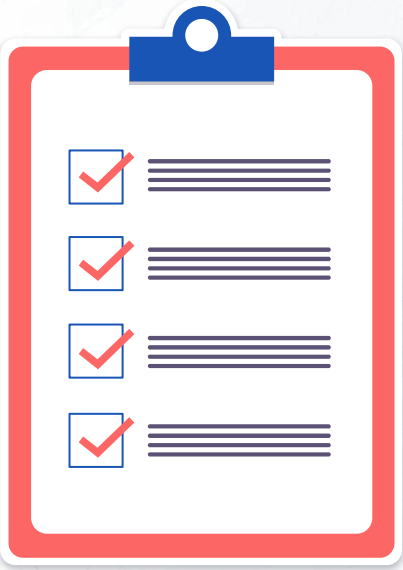
Here are the steps:

1. **Identification of Targets**
2. **Retrieval of Participants**
3. **Counting Participants**
4. **Merging Participants**
5. **Filtering Users**
6. **Sampling Verification**
7. **Sending Invitations**




# Methodology



## ★ Survey

1. **15 minute qualtrics survey.**
2. **Most questions in the survey were optional.**
3. **What they had any concerns**
   - **If yes, ask more specific experience**
4. **Survey also collect participants' background**

The survey was sent to all 10,530 GitHub users and received 410 responses, resulting in a response rate of around **4%**.

# Survey Questions

For this software project,estimate **what percent of your code** is written with the help of the following code generation tools.

For each of the following reasons **why you *use code generation tools*** in this software project, rank its importance.

For your software project, estimate how often the following reasons are **why you find yourself giving up on code created by code generation tools**.

For each of the following reasons **why *you do not use code generation tools***, rank its importance.

For your software project, **estimate how often you experience the following scenarios when using code generation tools**.

What types of **feedback** would you like to give to code generation tools to make its suggestions better? Why?

# Usage Characteristics

In this part, researchers analyzed by two category:

First part, using Quantitative method

- Usage Pattern
- Motivation for using
- Motivation for not using

Second part, using Qualitative method

- Successful use case

# Usage Characteristics - Usage Patterns

**Table 1: Participants' self-reported usage of popular AI programming assistants. An asterisk (*) denotes a write-in suggestion, which has limited information on its usage distribution. Percentages in italics on the chart (*N %*) represent the percent of the distribution that reported "Always"/"Often" (left) and "Rarely"/"Tried but gave up" (right).**

| Tool | # users | Med. % code written | Usage distribution |
|---|---|---|---|
| Amazon CodeWhisperer | 50 | 5% | *24%* *61%* |
| ChatGPT* | 25 | 20% | *59%* *14%* |
| GitHub Copilot | 306 | 30.5% | *46%* *30%* |
| TabNine | 118 | 20% | *27%* *66%* |
| Organization-specific code generation tool trained on proprietary code | 54 | 37% | *29%* *56%* |

Always (1+ times daily) Often (once daily) Sometimes (weekly) Rarely (monthly) Tried but gave up

- GitHub **Copilot is the most popular AI programming assistant**, with 306 users, of whom 46% use it frequently. Users report writing 30.5% of their code with its assistance.
- Organization-specific AI programming assistants helped participants write the largest percentage of code, at 37%.
- Chatbot-based programming assistants, such as ChatGPT, were used by 25 participants. **Despite having the highest proportion of frequent users (59%), ChatGPT ranked second to last in terms of the amount of code it assisted with, accounting for only 20%**.

# Usage Characteristics - Motivation

★ For Using

| | Motivation | Distribution |
|---|---|---|
| | *A. For using* | |
| M1 | To have an autocomplete or reduce the amount of keystrokes I make. | 86% … 6.2% |
| M2 | To finish my programming tasks faster. | 76% … 12% |
| M3 | To skip needing to go online to find specific code snippets, programming syntax, or API calls I'm aware of, but can't remember. | 68% … 14% |
| M4 | To discover potential ways or starting points to write a solution to a problem I'm facing. | 50% … 24% |
| M5 | To find an edge case for my code I haven't considered. | 36% … 44% |

Very important | Important | Moderately important | Slightly important | Not important at all

1. Convenience in programming was a significant motivation
   - 86 % Reduce keystroke
   - 76 % Finish the programming task faster
   - 68 % Recall Solution
2. Finding potential code solutions (M4) and edge cases (M5)

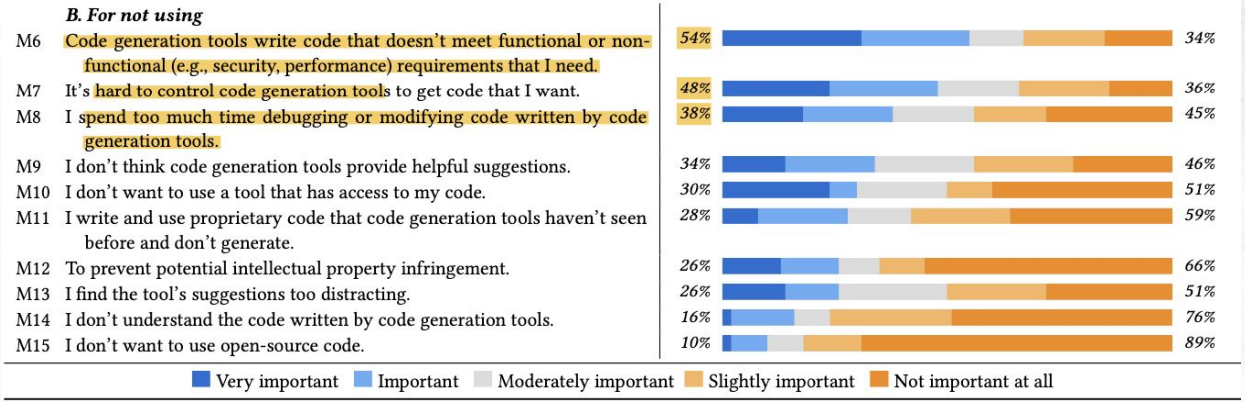
# Usage Characteristics - Motivation

★ **For Using**


https://www.youtube.com/watch?v=p7qMFhQh7rY

# Usage Characteristics - Motivation

★ For Not Using




- 54 % doesn't meet function or non-functional requirements
- 48 % hard to control code generation tools
- 38 % spend too much time for debugging or modifying code

# Usage Characteristics - Successful use cases

- **Repetitive code**
  - boilerplate code
  - Repetitive endpoints for CRUD
  - College assignments
- **Code with simple logic**
  - Consistent with previous work (only simple logic)
- **Autocomplete**
- **Quality Assurance**
- **Proof-of-concepts**
- **Learning**
- **Recalling**
- **Efficiency**
- **Documentation**

# Usage Characteristics - Key Findings

- GitHub Copilot users reported a median of 30.5% of their code being written with its help.

- The most important reasons for using AI programming assistants were for autocomplete, completing programming tasks faster, or skipping going online to recall syntax.

- Participants successfully used these tools to generate code that was repetitive or had simple logic. Participants reported the most important reasons for not using AI programming assistants were because the code that the tools generated did not meet functional or non-functional requirements and because it was difficult to control the tool.

# USABILITY OF AI PROGRAMMING ASSISTANTS

| | Situation | Distribution | |
|---|---|---|---|
| | *A. Usability issues* | | |
| S1 | I don't know what part of my code or comments the code generation tool is using to make suggestions. | 30% | 48% |
| S2 | I give up on incorporating the code created by a code generation tool and write the code myself. | 28% | 35% |
| S3 | I have trouble controlling the tool to generate code that I find useful. | 26% | 48% |

★ Usability Issue

- The biggest challenge is that participants don't know which part of the input influenced the output.
- Having trouble with controlling the model.
- Giving up on incorporating the code created by generation tool

# USABILITY OF AI PROGRAMMING ASSISTANTS

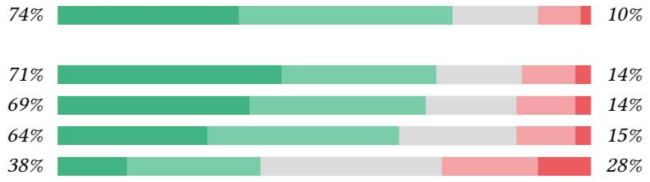
| | *B. Reasons for not understanding code output* | | |
|---|---|---|---|
| S10 | The generated code uses APIs or methods I don't know. | 25% | 43% |
| S11 | The generated code is too long to read quickly. | 23% | 45% |
| S12 | The generated code contains too many control structures (e.g., loops, if-else statements). | 19% | 44% |

★ Understanding outputted code

- AI Assistants generate unfamiliar APIs
- Generate too long to read quickly
- Too many control structure

# USABILITY OF AI PROGRAMMING ASSISTANTS

| | *C. Methods of evaluating code output* | |
|---|---|---|
| S13 | Quickly checking the generated code for specific keywords or logic structures | 74% |
| S14 | Compilers, type checkers, in-IDE syntax checkers, or linters | 71% |
| S15 | Executing the generated code | 69% |
| S16 | Examining details of the generated code's logic in depth | 64% |
| S17 | Consulting API documentation | 38% |



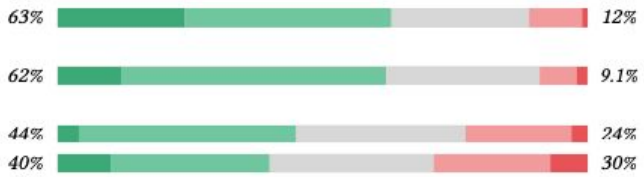

★ **Evaluating code output**

- **Quick Checks (S13, 74%)**
- **Compilers, IDES (S15, 71%)**
- **Code Execution (S16, 69%)**
- **In-depth Examination (S17, 38%)**
- **API Documentation (38%)**

# USABILITY OF AI PROGRAMMING ASSISTANTS

***D. Methods of modifying code output***

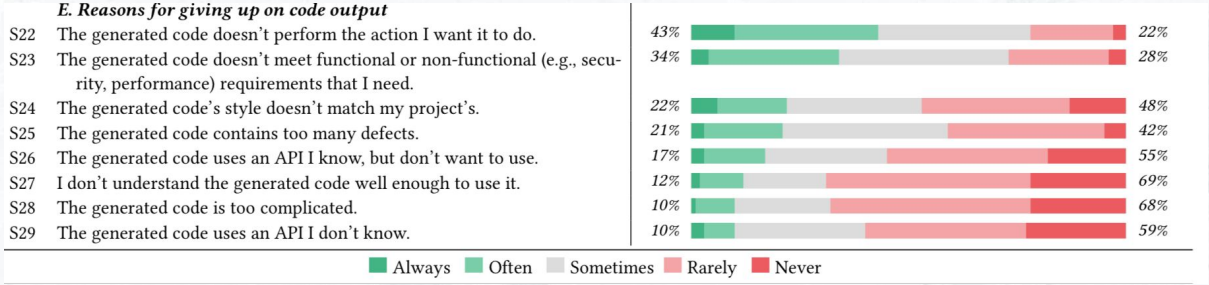
| | | | |
|---|---|---|---|
| S18 | When a code generation tool outputs something I don't want, I'm able to modify it to something I want. | 63% | 12% |
| S19 | I successfully incorporate the code created by a code generation tool by changing the generated code. | 62% | 9.1% |
| S20 | I use the code created by a code generation tool as-is. | 44% | 24% |
| S21 | I successfully incorporate the code created by a code generation tool by changing the code or comments around it and regenerating a new suggestion. | 40% | 30% |



★ **Modifying outputted Code**

- **Participants overall reported regularly having success with modifying the outputted code (S18, 63%), most often by changing the generated code itself (S19, 62%) rather than by changing the input context (S20, 40%).**
- **A smaller proportion of participants (S21, 44%) often used the generated code as-is.**

# USABILITY OF AI PROGRAMMING ASSISTANTS




★ **Giving up on code output**

- **43% Generate code didn't perform the intended action**
- **34% Generate code did not meet functional or non-functional requirements**

# USABILITY - Key Findings

- **The most frequent usability challenges reported by participants include:**

- Understanding what part of the input caused the outputted code.
- Giving up on using the outputted code.
- Controlling the tool's generations.

- **Participants most often gave up on using the outputted code because:**

- The code did not perform the intended action.
- The code did not account for certain functional and non-functional requirements.

# Additional Feedback

| | Concern | Distribution | |
|---|---|---|---|
| C1 | Code generation tools produce code that infringe on intellectual property. | 46% | 32% |
| C2 | Code generation tools have access to my code. | 41% | 38% |
| C3 | Code generation tools do not generate proprietary APIs or code. | 29% | 46% |
| C4 | Code generation tools may produce open-source code. | 29% | 53% |

Very concerned | Concerned | Moderately concerned | Slightly concerned | Not concerned at all

★ **Level of concern on issues related to AI programming assistants**

- **Intellectual Property (C1, 46%)**
- **Code Access (C2, 41%)**
- **Proprietary APIs (C3, 29%)**
- **Open Source Code (C4, 29%)**

# Additional Feedback

## ★ User Feedback (52X)

- Provide **feedback directly** to the AI programming assistant
- Feedback ranged from correcting outputted code to **teaching the model their personal coding style**.
- Preferences varied between providing **feedback in natural language or through code**.
- Some suggested using **rating systems like "like/dislike" buttons** to streamline the feedback process.

# Additional Feedback

## ★ Better understanding of code context (20X)

- Participants expressed the need for AI programming assistants to grasp code context better
- This includes understanding context from other files in the same workspace and nuances behind APIs and programming languages.
- Examples include recognizing when code is using deprecated APIs.
- *“To be able to better describe the contexts of our projects during creation. For a better understanding of our code generator." (P208)”*

# Additional Feedback

## ★ Tool Configuration (17X)

- Some participants wanted the ability to customize the tool's settings.
- Requests included distinguishing between long and short code generation, adjustable parameters, controlling the frequency of suggestions.
- Customization could help the model adapt to different developer modes.

# Implications

- Learning APIs and Programming Languages
- Aligning AI Programming Assistants to Developers
- Control Over Outputs

# Reflections

## Strengths

- 410 survey responses
- Relevance
- Practical Implications

## Weakness

- Lack of Diversity- Only GitHub
- Subjectivity
- Suggestions

## Interesting Findings

- Distinction between acceleration and exploration
- Desire for more natural language interactions

## Relation to Humans in SE

- Align with human needs and ways of working
- Importance of user control and the ability to understand tool's behavior

## Thoughts?

## Survey Feedback time!

# Class Activity (Introducing DEVIN!)

- **World 1st AI Software Engineer**
- **Capabilities**: Equipped with a shell, code editor, and web browser.
- Utilizes the **web browser to access API** documentation for learning and problem-solving.
- Automatically **adds debugging print statements** to code when encountering errors.

# Class Activity