Evolution of Programmers' Trust in Generative AI Programming Assistants

Presentation by Connor Mackey and Sam Fruechtemeyer

Research Paper by Anshul Shah et al.

Outline

- Background
- Methodology
  - Study Design
  - Trust
  - MATCH Model
- Results
- Discussion

Background

- In the past 2 years, generative AI has made its way into programming
- Copilot is a popular tool for code autocompletions
- Students at the University of California conducted a study evaluating trust in Copilot
- (As a note, this thesis has not been peer-reviewed yet)

Methodology – Study Design

- Conducted in upper-level software engineering course in UC San Diego [1]
- 7tudents are given an 80-minute lecture on NLP and programming, then a 10-day project
- Students take the survey on the right before the lecture, after the lecture ("short-term"), and after the project ("long-term")
  - Some additional questions depending on when the student took the survey

Table 2: Survey on developer trust in AI systems from Amoozadeh et al. [3], rephrased to use "GitHub Copilot" in place of "AI system". Students rate their agreement on a scale of 1 (Strong Disagree) to 5 (Strong Agree) for each statement.

ID	Survey Question
S1	I trust GitHub Copilot's output.
S2	The output GitHub Copilot produces is as good as that which a highly competent person could produce.
S3	I know what will happen the next time I use GitHub Copilot because I understand how it behaves.
S4	I believe the output of GitHub Copilot even when I don't know for certain that it is correct.
S5	I have a personal preference for using GitHub Copilot for my tasks.
S6	Overall, I trust GitHub Copilot.

- Please explain your answer above. Why did your trust change (or why not)?

Methodology – Trust

- How do we measure trust?
- How Liao and Sundar define trustworthiness attributes [2]:
  - **Ability** - what can the tool do?
  - **Intention Benevolence** - why is this tool being developed?
  - **Process Integrity** – how do I know the tool is reliable?
- In addition, there are trust affordances:
  - **AI-generated content**
  - **Transparency**
  - **Interaction**
- We also need to consider the user:
  - Some users like to consider everything first
  - Others like to do now, think later
  - User experience also affects how they perceive output

Methodology – MATCH Model

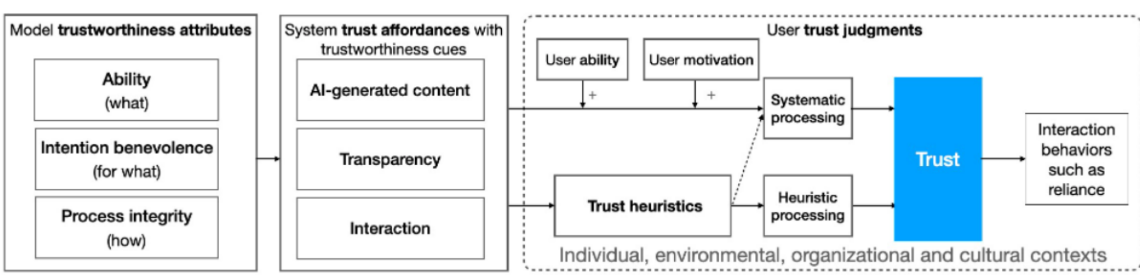
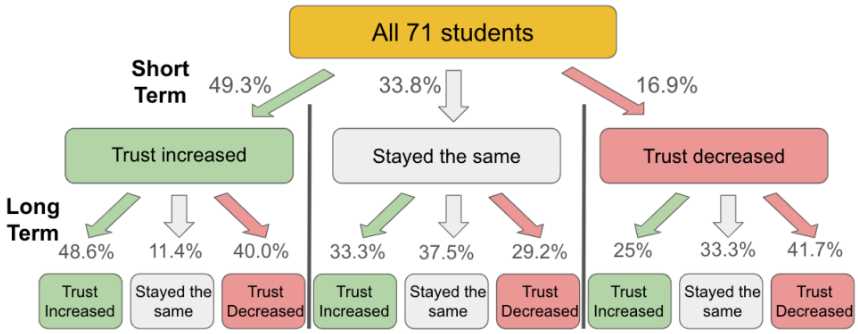


Figure 1: The MATCH model, reproduced exactly as presented by Liao and Sundar [19].

Results



	Increased	No Change	Decreased
Immediate Use	49.3%	33.8%	16.9%
Extended Use	39.4%	23.9%	36.6%

Students' Reasoning for Change in Trust

Table 6: Most common themes mentioned by students whose trust increased after immediate use (n = 35).

Code	Frequency
Learned about Copilot's features	29%
Copilot's context awareness	26%
Copilot's incorrect code output	26%
Copilot's general correctness	23%
Copilot supports code comprehension	20%
Understanding how Copilot works	17%

Table 7: Most common themes mentioned by students whose trust decreased after immediate use (n = 12).

Code	Frequency
Copilot's incorrect code output	75%
Copilot's general incorrectness	25%
Copilot's variability	17%
Copilot supports code comprehension	17%

Table 8: Most common themes mentioned by students whose trust increased after extended use (n = 28).

Code	Frequency
Copilot's correct code output	57%
Copilot requires a competent programmer	29%
Copilot's general correctness	18%
Copilot depends on the quality of prompts	18%
Copilot provides a scaffold	14%
Copilot supports code comprehension	14%

Table 9: Most common themes mentioned by students whose trust stayed the same after extended use (n = 17).

Code	Frequency
Copilot performed as expected	29%
Copilot requires a competent programmer	29%
Copilot depends on the quality of prompts	24%

Table 10: Most common themes mentioned by students whose trust decreased after extended use (n = 26).

Code	Frequency
Copilot requires a competent programmer	54%
Copilot's incorrect code output	54%
Copilot's inability to locate the correct code	31%
Copilot's general incorrectness	23%
Copilot's inability to debug its own code	12%

Observation 1

Observation: "Students had different reactions to the lecture and project components of the study"

Recommendation: "CS educators should provide opportunities for students to use AI programming assistants for tasks"

Observation 2

Observation: "Students valued Copilot’s correctness (or incorrectness) on various tasks."

Recommendation: "CS educators should ensure that students can still comprehend, modify, debug, and test code in large code bases *without* AI assistants"

Observation 3

Observation: "Students valued learning how Copilot works"

Recommendation: "CS educators should ensure that students are aware of how AI assistants generate output"

Observation 4

Observation: "Students valued learning about Copilot’s features"

Recommendation: ": CS educators should explicitly inform and demonstrate key features of AI assistants that are useful for contributing to a large code base"

Limitations

- Attitude but not behavior
- Only tested students with programming background
- Examines students' trust over only 10 days

Works Cited

[1] Anshul Shah, Thomas Rexin, Elena Tomson, Leo Porter, William G. Griswold, and Adalbert Gerald Soosai Raj. 2025. Evolution of Programmers' Trust in Generative AI Programming Assistants. In *25th Koli Calling International Conference on Computing Education Research*. ACM, New York, NY, USA, 12 pages. https://doi.org/XXXX

[2] Q. Vera Liao and S. Shyam Sundar. 2022. Designing for Responsible Trust in AI Systems: A Communication Perspective. In 2022 ACM Conference on Fairness, Accountability, and Transparency (FAccT '22), June 21–24, 2022, Seoul, Republic of Korea. ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3531146.3533182