RECAP

REPLAY & EXAMINE CAPTURED AI PROGRAMMING

An End-to-End Platform for Capturing, Replaying, and Analyzing AI-Assisted Programming Interactions

Keyu He★ · Qianou Ma★ · Valerie Chen · Wayne Chi · Tongshuang Wu

Carnegie Mellon University · ★ equal contribution

SCAN FOR EVERYTHING
DEMO VIDEOS · PAPER · CODE

01 · MOTIVATION

Chat logs and git histories alone lose the story

Studying how developers work with AI assistants requires the **full interaction context**, not isolated data sources.

> *Which prompt led to which edit? What was tried and discarded? How did strategy evolve over time?*

RECAP answers these: it passively records chats and fine-grained edits inside VS Code, merged into **one replayable timeline**.

02 · SYSTEM

Three layers, one unified timeline

1 Copilot Interaction Archiver
Passively captures Copilot chats, agent commands, and a **shadow git history** of every change, with unsaved snapshots every 5 seconds.

2 Session Replay Viewer
Merges chats and shadow commits into one chronological timeline: step through a session and see which prompt produced which change.

3 Analysis Layer
Extensible modules on the unified timeline: behavior classification, AI-reliance metrics, prompt clustering. Plug in your own.

03 · MECHANISM

Linking every prompt to its edits

FROM THE CHAT LOG
Text edit groups (TEGs)
Each AI response carries the exact paths and content it proposed.

FROM THE SHADOW GIT REPO
Fine-grained diffs
A commit on every save, plus every 5 seconds of in-editor change. Matched to TEGs by fuzzy line-level comparison within a 5-minute window.

AI-attributed edit | Human edit | External source

04 · DEPLOYMENT

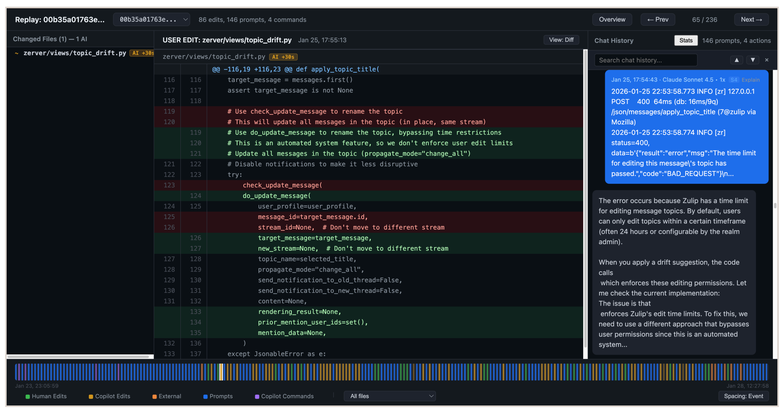
Deployed in a real classroom

A software engineering for ML course at **Carnegie Mellon University**, Spring 2026: students extended two LLM features in **Zulip**, Message Recap and Topic Title Improver, across a two-week project and 406 work sessions. Shadow git covered all 41 students, even when AI was used outside VS Code.

41 students using Copilot
2,034 prompts captured
8,239 code edits

05 · REPLAY

Step through one session, or scan all 41



The replay viewer. Chat, diffs, and one chronological timeline, with per-file AI attribution.

The cohort overview. Per-student timelines for all 41 students, sorted by AI-edit share; instructors spot outliers, then click into any session.

06 · PATTERNS

What neither chat nor code shows alone

Agentic generation: one strategy, two outcomes

One student pasted the full assignment spec for both features. The chat shows an identical strategy; only the linked diffs show one feature finishing and the other spiraling.

✓ Feature 1 · finished in 8 prompts
`#0 spec paste → #8 bug report → #10 success`

✗ Feature 2 · same approach, cascading errors
`#12 spec paste → #29 build error → #34 SyntaxError`

Error-pasting loop

The same `TypeError` pasted three times in 11 minutes. Chat alone looks like normal debugging; the linked diffs reveal no forward progress: each AI fix reintroduced the original error.

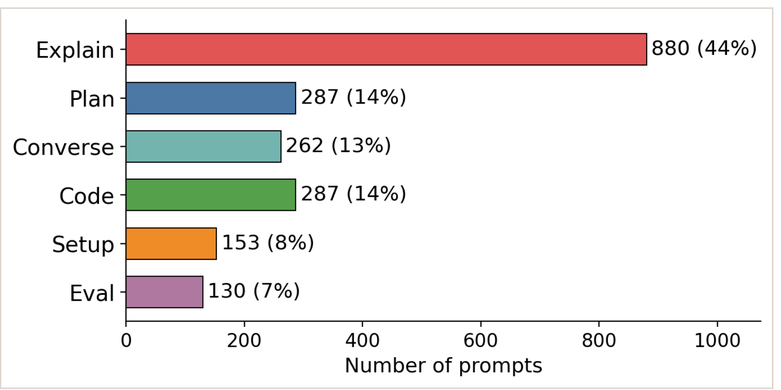
Cross-tool usage

Stuck for hours, a student pasted a ChatGPT architecture idea into Copilot Chat: *"This is what ChatGPT says and I think we should try and implement that."* Copilot generated multi-file edits from it, but the approach still failed.

07 · ANALYSES

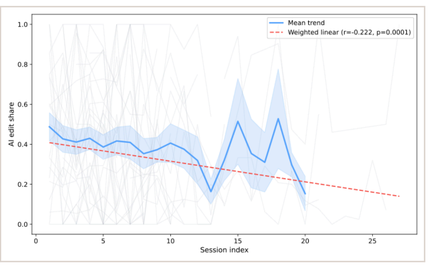
Three example modules, one linked dataset

A Behavior classification



Students used AI more for comprehension than generation. Explain covers 44% of prompts (explain-error alone 29%), three times the share of Code (14%).

B AI reliance over time



Reliance on AI for edits decreased over the project. AI edit share trends downward across successive work sessions ($r = -0.222$, $p < 0.001$).

C Prompt embedding & clustering

Prompts form coherent intent clusters: error debugging, endpoint testing, UI styling. Explorable by student, model, time, and category.

08 · Takeaways

Why RECAP

1. **Linked data tells a richer story.** Prompt-to-edit attribution enables analyses no single data source supports.
2. **Replay makes patterns visible.** Error-pasting loops and cross-tool usage surface within minutes of stepping through a session.
3. **Open and extensible.** MIT-licensed; add your own analysis modules without touching the capture layer.

09 · GET RECAP

Use it in your course or lab

Install — VS Code Marketplace
Source — GitHub
Paper — ACL Anthology

He, Ma, Chen, Chi & Wu. RECAP. In *Proc. ACL 2026, Vol. 3: System Demonstrations*, pp. 692–701. DOI 10.18653/v1/2026.acl-demo.68

Funding. Partially supported by NSF (CNS-2213791, 2414915), a Google Academic Research Award, and an Amazon AI Research Award.

Ethics. IRB-approved; informed, voluntary participation; identifiers SHA-256 hashed client-side; no individual student data released.

Future work. Capture for Cursor and Claude Code; linking behavior patterns to learning outcomes; longitudinal analysis.