



AGENTIC AI | LECTURE 14

OpenHands

Frameworks 1 · October 8, 2026

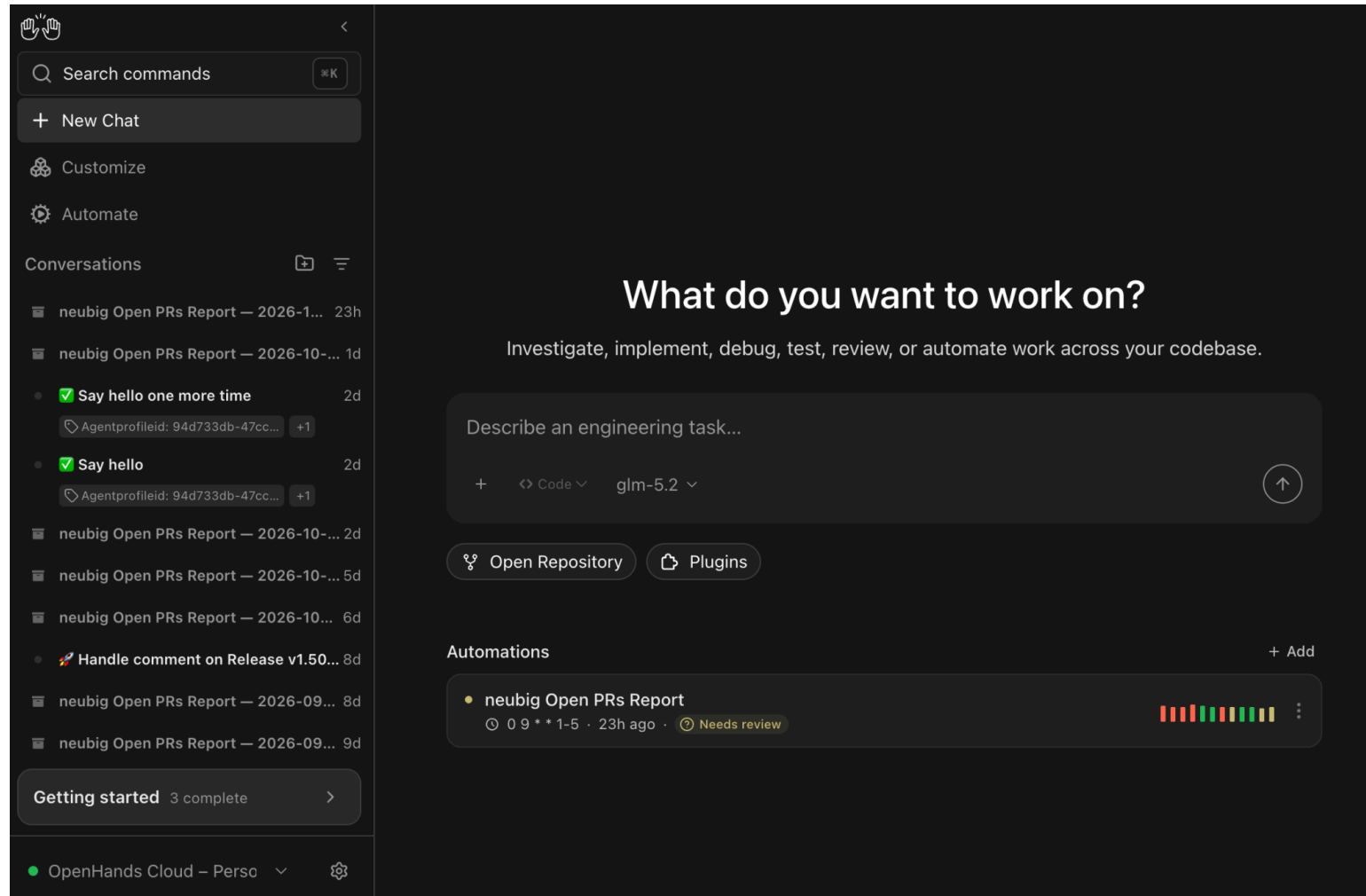
Carnegie Mellon University

Language Technologies Institute



Simple Frontend Demo

Agent Canvas





History

Code completion

Through 2023

- Copilot and early Cursor: code suggestions
- HumanEval: single functions, unit tests
- Human-led editing, execution, and checking

A HumanEval-style task

```
def has_close_elements(  
    numbers, threshold  
):  
    """Nearby number pairs"""  
    # Function completion
```

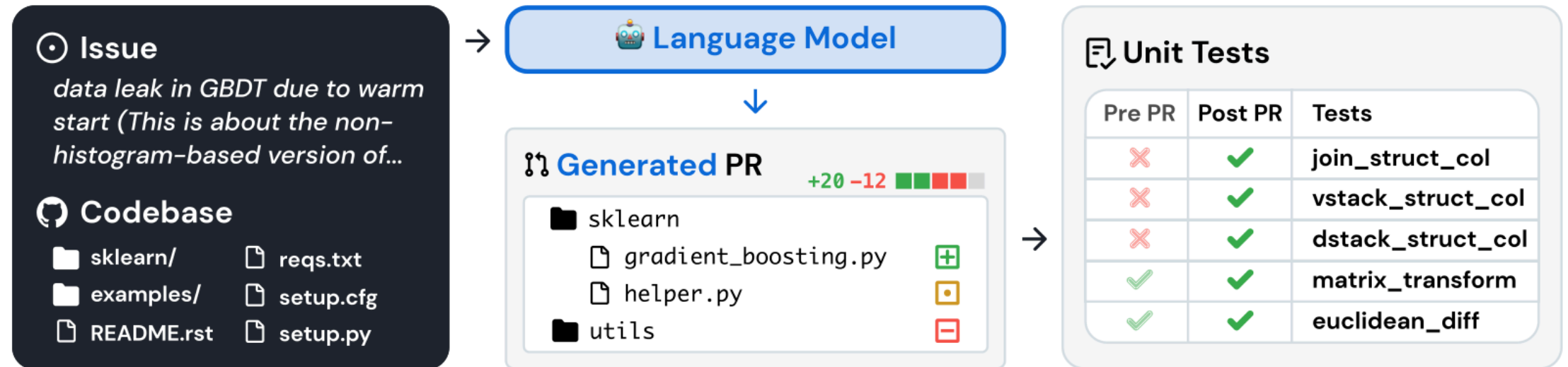
Input: a function specification

Output: a function implementation

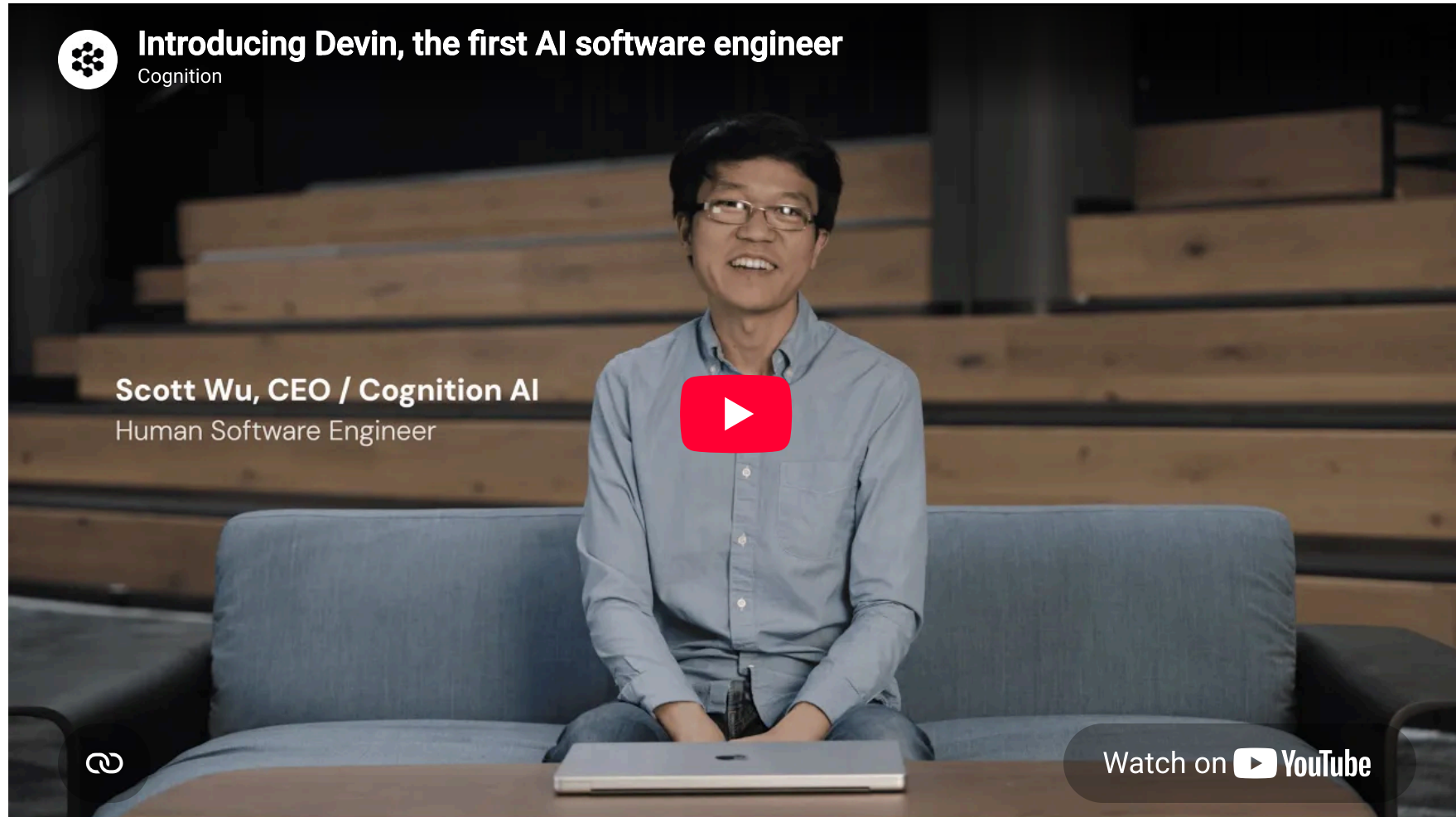
Repository-level evaluation

October 2023 · SWE-bench

- Real issue + repository snapshot
- Patch + repository tests; difficult early results

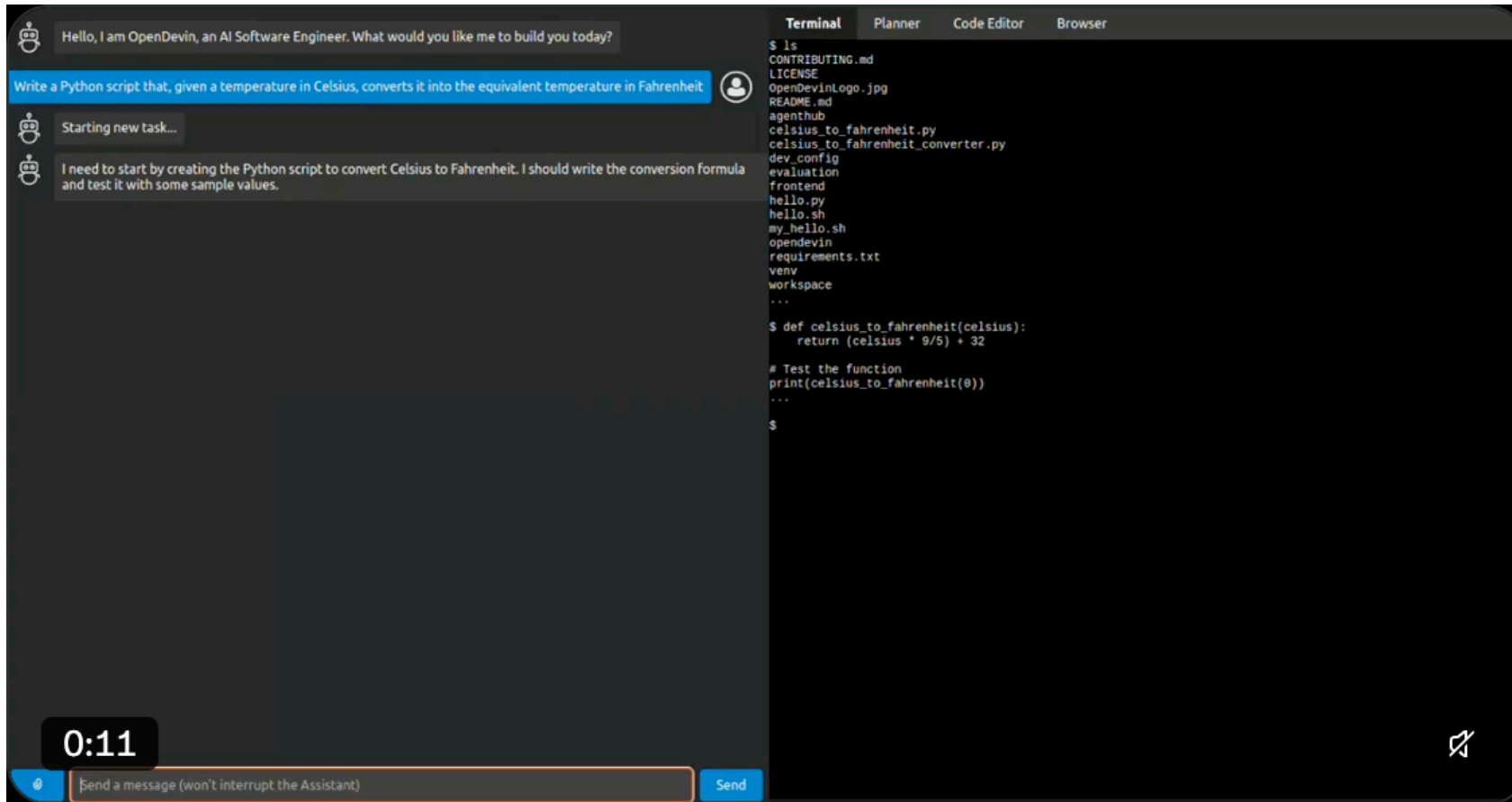


Devin



OpenDevin

Frontend + agent + Docker sandbox · March 26, 2024



Early coding agents

April 2024

SWE-agent

File search, code edits, fix verification

May 2024

OpenDevin CodeAct 1.0

Code execution, terminal commands, feedback

Agent–computer interfaces as a research topic

Benchmark progress

2025

Better models

Reasoning, code editing, tool use

Better agent systems

Tools, test environments, context, recovery

Benchmark performance: model + agent system

Deployment

2026

- Longer runs, interruptions, parallel tasks
- Progress visibility and result inspection
- Repeatable workflows, scoped access, verification



The OpenHands Software Agent SDK

Batteries included, but flexible

Ready-to-use agent

Models, tools, state, workspaces

Customizable components

Model choice, new tools,
instructions, deployment

Shared foundation for GUI, CLI, and integrations

Model and tools

```
llm = LLM(  
    model=os.getenv("LLM_MODEL", "gpt-5.5"),  
    api_key=os.getenv("LLM_API_KEY"),  
)  
  
agent = Agent(  
    llm=llm,  
    tools=[  
        Tool(name=TerminalTool.name),  
        Tool(name=FileEditorTool.name),  
        Tool(name=TaskTrackerTool.name),  
    ],  
)
```

- Model choice
- Terminal, editor, task tracker
- Reusable agent configuration

Conversation

```
cwd = os.getcwd()  
conversation = Conversation(agent=agent, workspace=cwd)  
  
conversation.send_message(  
    "3 project facts in FACTS.txt"  
)  
conversation.run()
```

- Workspace: files and execution
- Conversation: task, actions, observations
- Run: action–observation loop

Custom tools

```
class GrepAction(Action):
    pattern: str = Field(
        description="Regex to search for"
    )
    path: str = Field(default=".")

class GrepObservation(Observation):
    matches: list[str] = Field(
        default_factory=list
    )
```

- Action: typed input
- Executor: search implementation
- Observation: model-readable result

Conversation persistence

```
conversation = Conversation(  
    agent=agent,  
    workspace=cwd,  
    persistence_dir="./.conversations",  
    conversation_id=conversation_id,  
)
```

- Saved events and conversation state
- Restoration: same ID and storage
- Workspace persistence: a separate concern

Context management

```
condenser = LLMSummarizingCondenser(  
    llm=llm.model_copy(update={"usage_id": "condenser"}),  
    max_size=10,  
    keep_first=2,  
)  
agent = Agent(llm=llm, tools=tools, condenser=condenser)
```

- Older events → summary
- Original instructions + recent work
- Potential detail loss; files and tests as evidence

Remote execution

```
with DockerWorkspace(  
    server_image=server_image,  
    platform=detect_platform(),  
) as workspace:  
    agent = get_default_agent(llm=llm, cli_mode=True)  
    conversation = Conversation(agent=agent, workspace=workspace)  
    conversation.send_message(  
        "3 project facts in FACTS.txt"  
    )  
    conversation.run()
```

- Agent Server inside Docker
- Same conversation API; remote task and events

Action confirmation

```
conversation.set_confirmation_policy(AlwaysConfirm())  
conversation.send_message("Directory listing with ls -la")  
conversation.run()
```

- Pause before action execution
- Pending action → human decision
- Rejection → revised plan

Single Agent vs. Multi-agent

Specialized roles

Early multi-agent workflows: specialized roles

Manager	Issue plan
Reproducer	Bug reproduction
Fault localizer	Bug location
Editor	Code changes
Verifier	Fix verification

A single-agent baseline

- One agent: reproduce, locate, edit, test
- Shared context across activities
- Separate roles: handoffs and maintenance
- Additional agents for a clear purpose

General-purpose tools

A small tool set, many combinations

Edit files

Source, configuration,
documentation

Run commands

Search, dependencies,
tests

Browse

Documentation and
visual inspection

Skills

Additional instructions for the same agent

```
agent_context = AgentContext(  
    skills=[Skill(  
        name="repo.md",  
        content=repo_instructions,  
        trigger=None,  
    )],  
)  
agent = Agent(  
    llm=llm, tools=tools,  
    agent_context=agent_context,  
)
```

- Project conventions and recurring procedures
- Relevant instructions on demand
- Earlier name: microagents

MCP tools

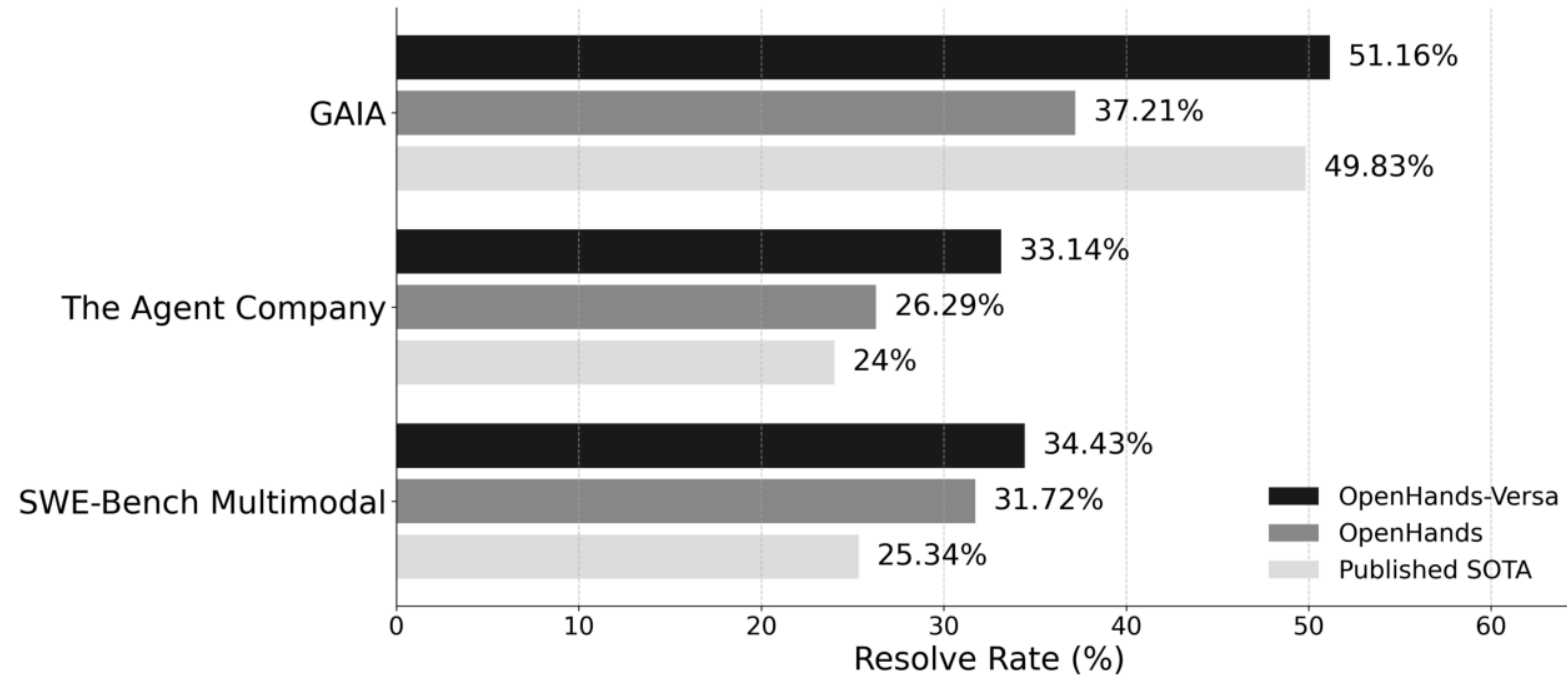
Model Context Protocol: external tool connections

```
mcp_config = {  
    "repomix": MCPServer(  
        command="npx", args=["-y", "repomix@1.4.2", "--mcp"]  
    ),  
}  
agent = Agent(llm=llm, tools=tools, mcp_config=mcp_config)
```

- Server-provided tool descriptions and execution
- Built-in and external tools together
- Server access and trust requirements

Generalist problem solving

OpenHands-Versa: coding + visual browsing + search + file access



Shared tools across coding, research, and office work

Division of labor

Parallel work

Independent issues or repositories

Independent review

Patch checking; separate task and permissions

Cost control

Cheaper models for suitable tasks

Task boundaries, shared context, coordination cost



Agent Hosting and Interface

Local and cloud execution

Local computer

Existing files, services, and context

Machine availability during execution

Cloud workspace

Continued work with the laptop off

Remote repository, dependencies, credentials

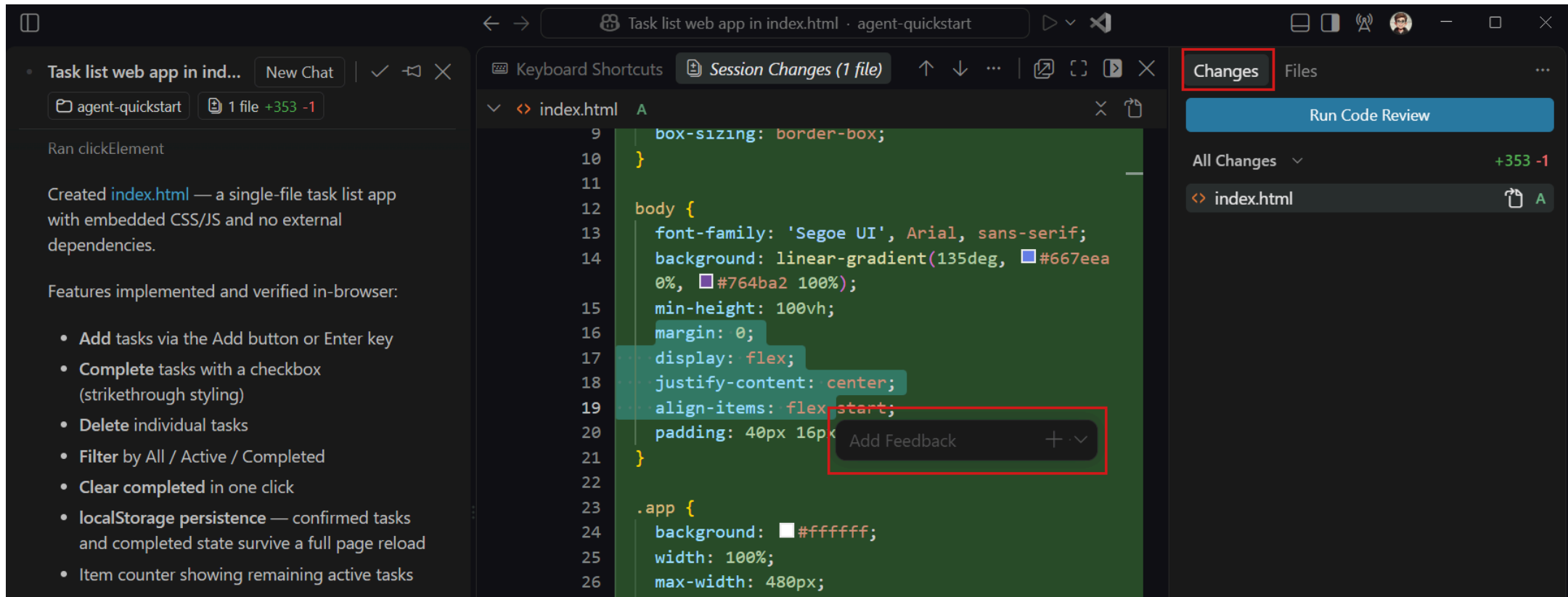
Interface choices

Execution location × interaction interface

	Local execution	Cloud execution
IDE	Editor workspace	Remote workspace
GUI	Desktop application	Browser application
CLI	Local terminal	Remote session
Integration	Local service	Hosted worker

IDE

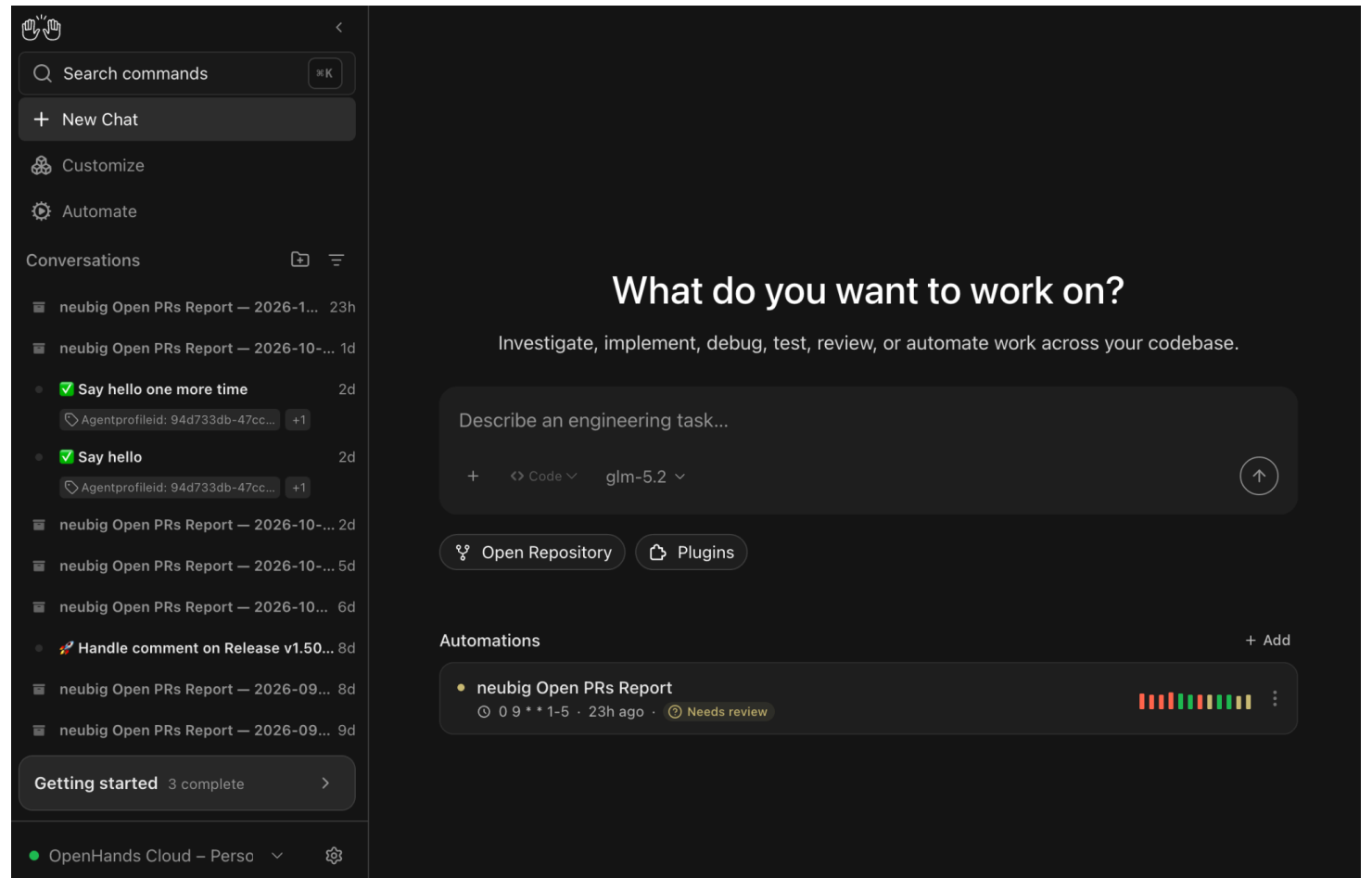
- Familiar coding workspace
- Files, diffs, and tests
- Task management and review



Custom GUI

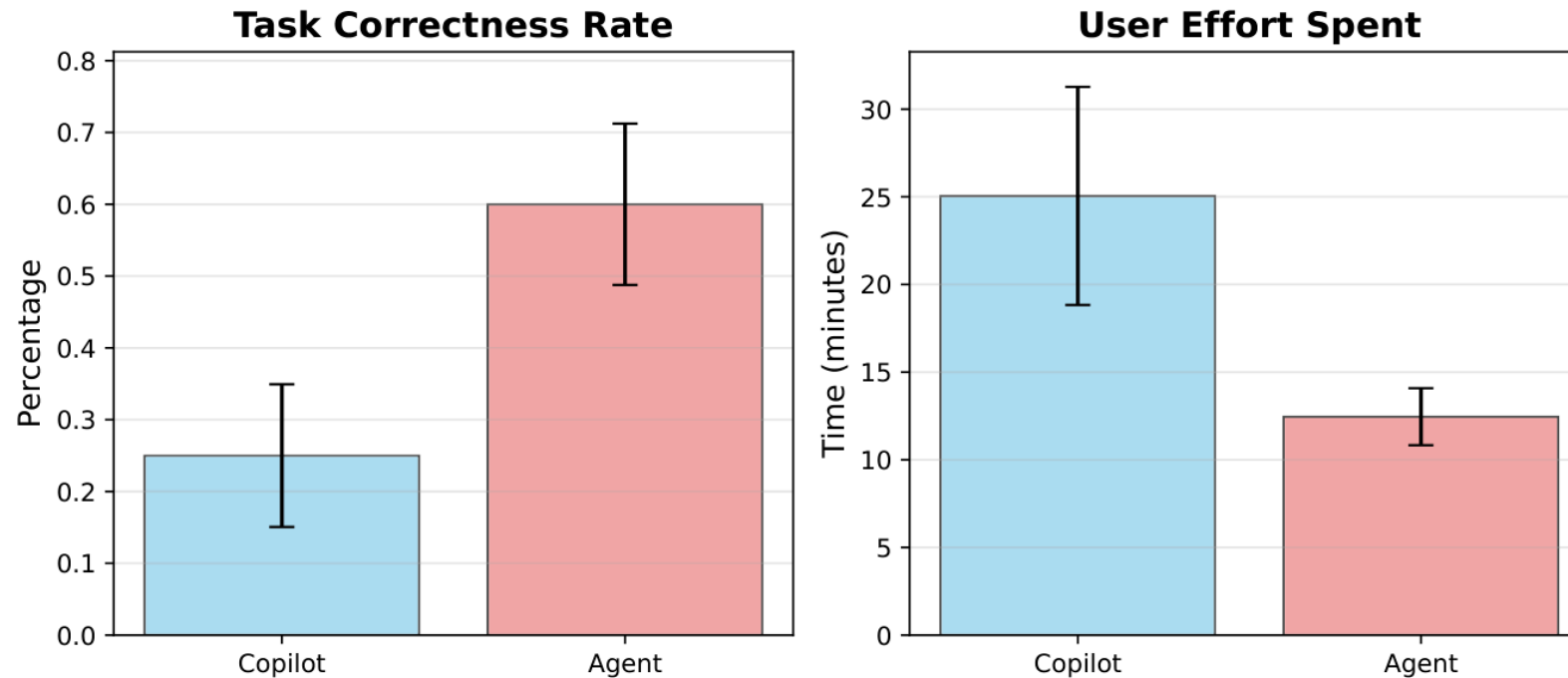
A workspace organized around agent tasks

- Progress and parallel runs
- Previews, diffs, and approvals
- Setup and workspace management



Human effort and oversight

20 regular copilot users · Python tasks · Copilot vs. OpenHands

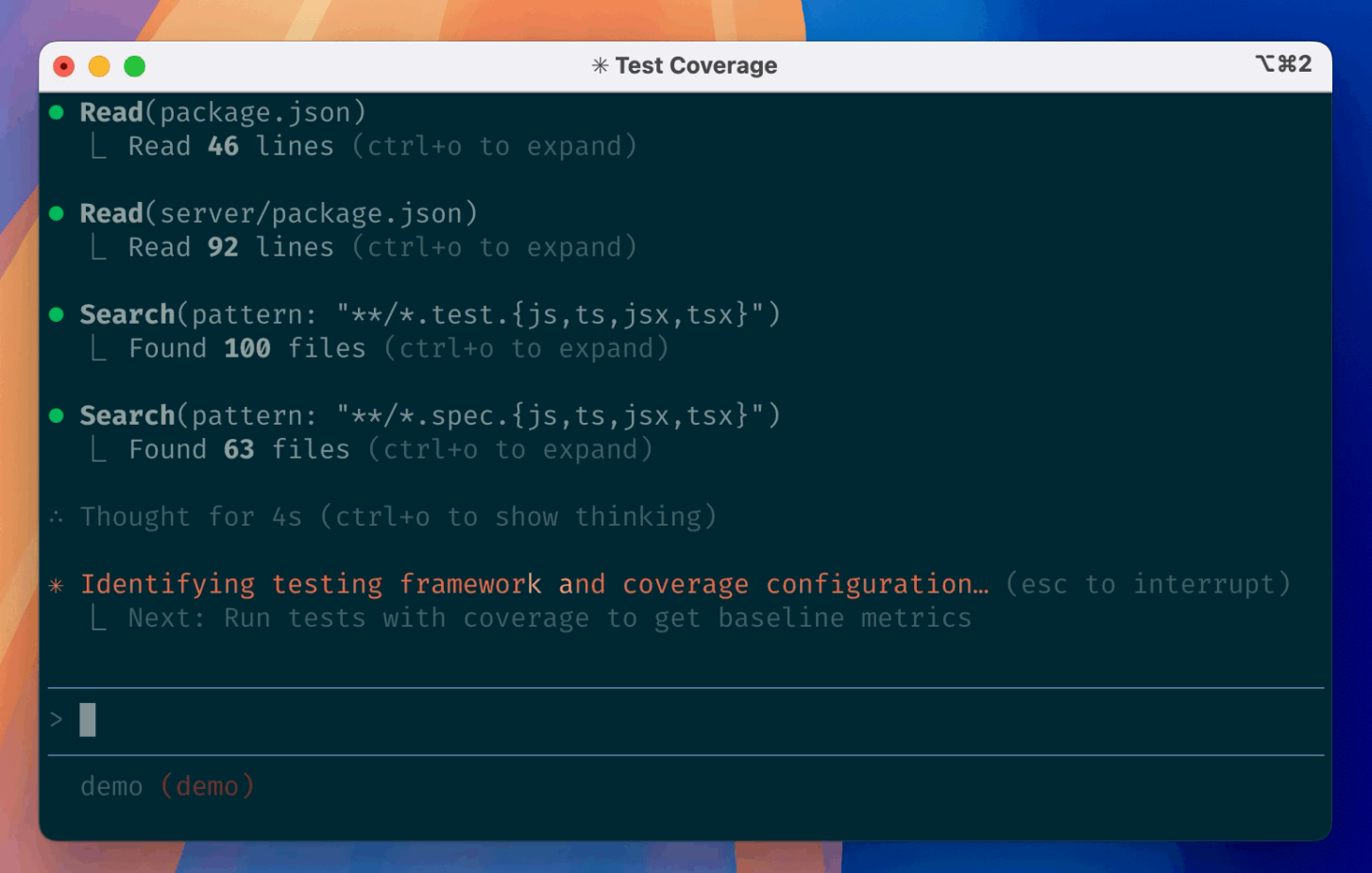


Lower active effort; continued need for understanding and oversight

CLI

A familiar interface
across environments

- Local and remote terminals
- Shell commands and scripts
- Separate visual previews

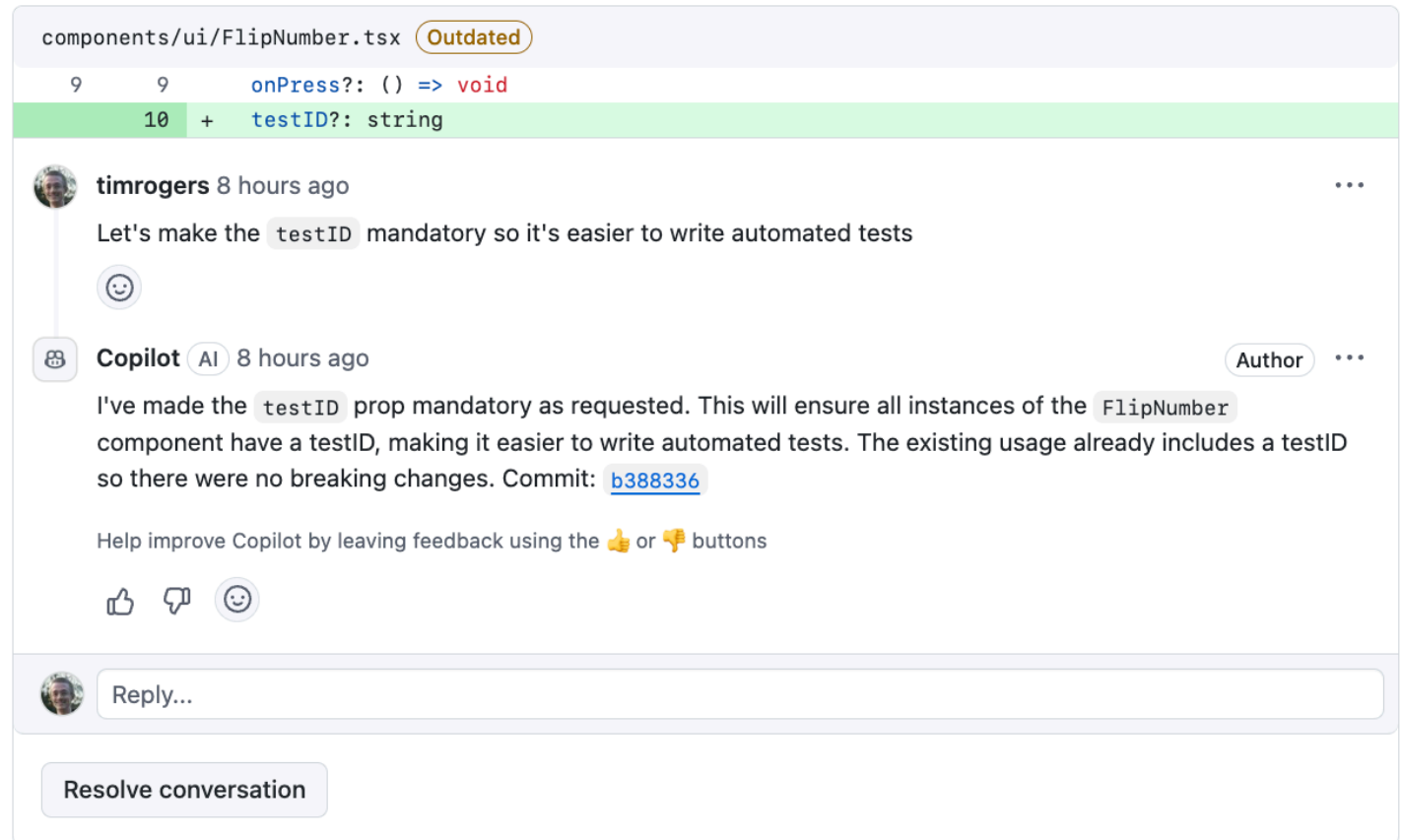


```
* Test Coverage
• Read(package.json)
  | Read 46 lines (ctrl+o to expand)
• Read(server/package.json)
  | Read 92 lines (ctrl+o to expand)
• Search(pattern: "**/*.test.{js,ts,jsx,tsx}")
  | Found 100 files (ctrl+o to expand)
• Search(pattern: "**/*.spec.{js,ts,jsx,tsx}")
  | Found 63 files (ctrl+o to expand)
∴ Thought for 4s (ctrl+o to show thinking)
* Identifying testing framework and coverage configuration... (esc to interrupt)
  | Next: Run tests with coverage to get baseline metrics
>
demo (demo)
```

Integrations

Agents in existing work channels

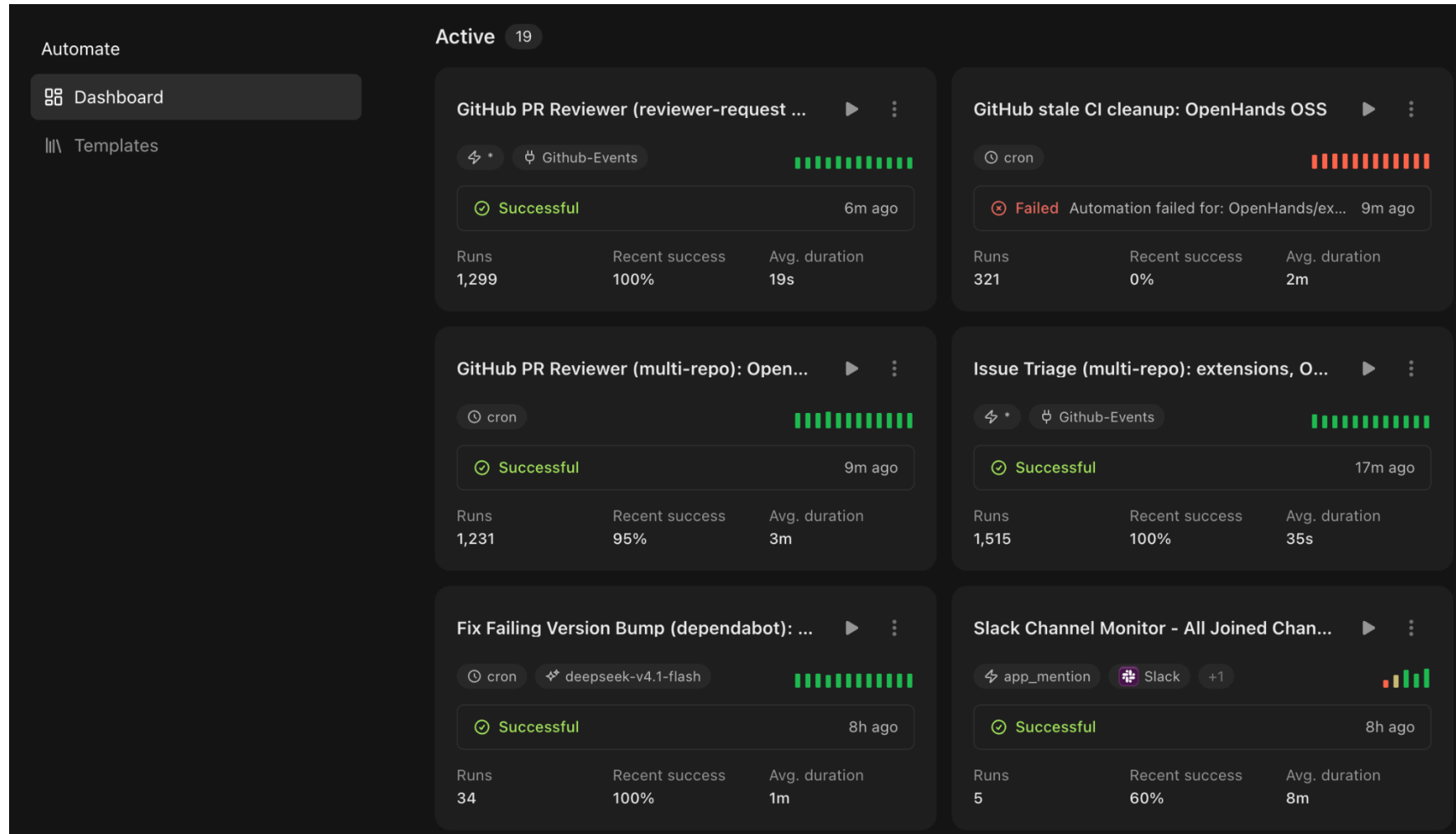
- Issues: task context
- PRs: diffs, tests, review
- Chat: requests and results





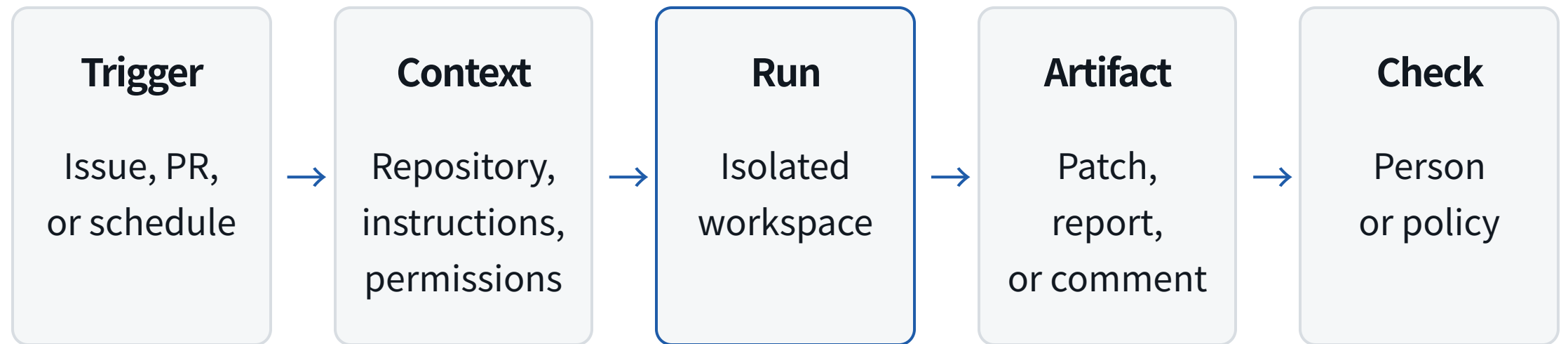
Automations for Software Development

Automation dashboard



An automation run

Event → bounded task → inspectable result



Automation demo

Existing issue-triage or PR-review run

- Trigger and instructions
- Repository and permissions
- Input event → run → output
- Output quality against the original task

Software factory

Recurring tasks, explicit handoffs, checks

Issue triage

Request and acceptance criteria

Implementation

Patch, tests, pull request

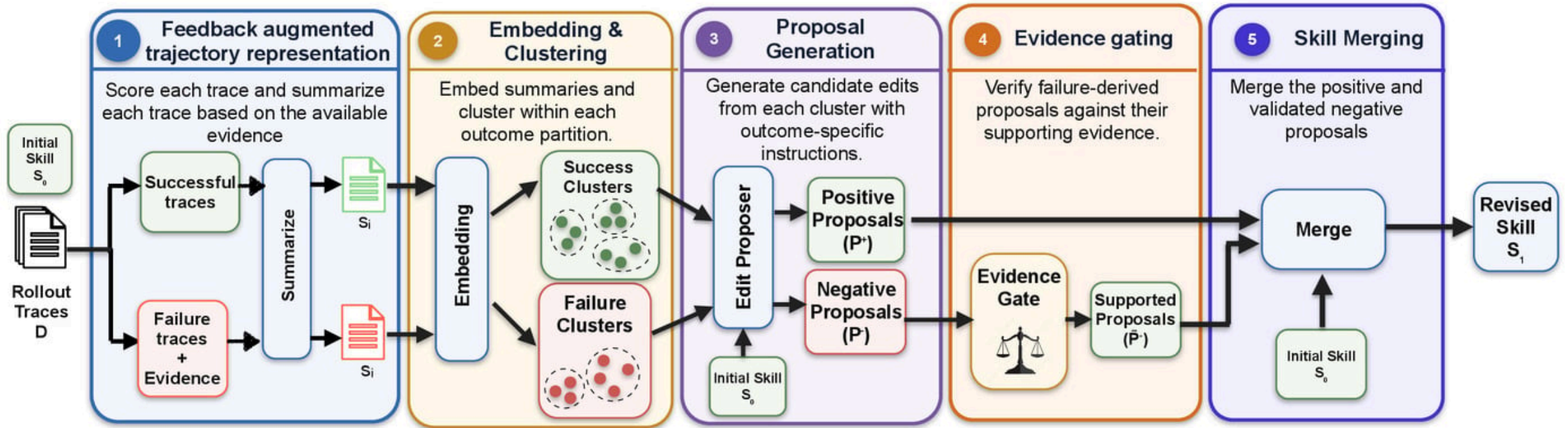
Code review

Independent verification

Merge check

Passing checks and approval on the current commit

Learning from past runs

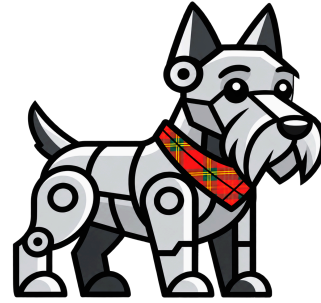


Skill refinement results

Held-out tasks · Accuracy (%) and PR-review F1

Task	Init.	Model	Initial	GEPA	T2S	SKILLREFINER
SpreadsheetBench	LLM	GPT-5.4-mini	77.5	78.5	79.0	84.0 [†] _*
	LLM	MiniMax-M2.7	68.0	75.0	69.5	76.0 _*
	Human	GPT-5.4-mini	86.0	86.0	86.5	86.5
	Human	MiniMax-M2.7	79.0	73.0	81.5	84.5 [†]
DAPO-Math	LLM	GPT-5.4-mini	90.0	90.0	91.0	91.0
	LLM	MiniMax-M2.7	53.5	53.5	61.0	64.0 [†]
PR Review	Human	GPT-5.4-mini	0.16	—	0.13	0.18 _*
	Human	MiniMax-M2.7	0.11	—	0.13	0.15

Higher scores than initial skills across both agent models



Questions?

Next Class: LangGraph