Lecture VII — Extending the Agent

AI-Assisted Programming for PhD Researchers

Dr. Tobias Vlćek
Helmut Schmidt University — September 2026

Why Extend

Out of the Box

Your agent already has three things:

- The **files** in your project
- A **shell** to run commands
- Its **training data** — frozen, and often out of date

. . .

It lacks today's docs, your library's real API, your reference manager, and your recurring workflows.

Four Extension Mechanisms

This session tours all four, simplest to most powerful:

- **Skills** — canned instructions it loads on demand
- **MCP servers** — give the agent new tools
- **Subagents** — delegate a subtask to a fresh context
- **Worktrees** — run work in parallel without collisions

We start with the simplest, a plain markdown file, and build up to tools and parallel agents.

Skills

Recurring Instructions, Packaged

- The lowest-effort extension: just a versioned markdown file
- No install, no server, no keys — you write instructions and save them
- The agent reads a skill only when the task matches its description
- AGENTS.md is always-on; a skill is on-demand expertise
- Keeps your context lean until the moment the knowledge is needed

Skill or AGENTS.md?

- **AGENTS.md** — always-on rules applied to every interaction
- **A skill** — on-demand expertise the agent loads only when the task matches its `description`

- Reach for a skill when the instructions are for a *specific recurring task*, not every session
- You already maintain an AGENTS.md from Lectures 02 and 04 — a skill sits beside it
- Both are just versioned markdown you own

The Course Writing Skills

We ship adapted skills you will run on real text this afternoon:

- **academic-grammar**, **conciseness**, **hedging-language** — grammar, filler, over-claiming
- **acronym-check**, **term-consistency**, **tense-consistency** — the consistency trio
- **humanizer** — strip AI-tell phrasing
- **literature-grounder** — add grounded citations
- Eight editors, each doing one job

Skills Compose

- Small, single-purpose skills chain together over one piece of text
- Run **academic-grammar** → **conciseness** → **hedging-language** on a paragraph
- Each does one job, so each diff is easy to review
- A skill can even reach an external API with no server — **reference-lookup** queries OpenAlex for a DOI
- The rule of thumb: skills before servers

Anatomy of a Skill

Frontmatter says *when*; the body says *how*:

```
---
name: figure-caption-checker
description: Check figure captions for completeness. Use when reviewing
figures in a manuscript.
---

Check each figure caption against these rules:
- State what the figure shows, how it was measured, and n.
- Report units on every axis and every value.
- No interpretation in the caption — that belongs in the text.
- Flag any caption that breaks a rule and suggest a fix.
```

Write Your Own

- Any workflow you explain to the agent twice deserves a skill
- Keep them short — a page of instructions, not an essay
- Version them with your project, like any other file
- A stale skill misleads; prune it when the workflow changes
- Vet a downloaded skill by reading it, not its star count — flashy repos are often fake-starred, and a skill carries supply-chain risk like any dependency

MCP — Model Context Protocol

What MCP Is

- An open standard for connecting agents to external systems
- A **server** exposes tools — functions the agent can call
- Servers exist for GitHub, docs lookup, web search, databases, and more
- Any MCP-aware host can use them: OpenCode, Claude Code, Zed
- "USB for agents" — write the server once, plug it in anywhere

More setup than a skill (a server to install and run), but it reaches real external systems.

Wiring One Up

GitHub's is a **hosted** server — a `remote` URL, no command to install. Add it to `opencode.json`, then restart:

```
{
  "mcp": {
    "github": {
      "type": "remote",
      "url": "https://api.githubcopilot.com/mcp/",
      "enabled": true
    }
  }
}
```

Authenticate with `opencode mcp auth github`; the agent then sees the tools and decides when to call them.

Live: Ask GitHub

You type a plain request:

```
On this repo, list the open issues and
summarize the newest one in two sentences.
```

...

- The agent recognizes the GitHub server can answer this
- It calls the issue-listing tool, then reads the newest issue
- It summarizes the result back to you — read-only, nothing changed

MCP Is Also an Attack Surface

- A local server runs **code on your machine** with your permissions
- Its tool descriptions enter your context, a channel for prompt injection

Same lethal trifecta as this morning — install only servers you trust [1].

Subagents

Delegation with a Clean Slate

- Spawn a fresh-context agent for a self-contained subtask
- Good for research or a review that would flood your main session
- It does the messy work and returns only a **summary**
- Your own context stays lean and focused

The Fresh-Eyes Reviewer

- Yesterday's Lab 4 second reviewer was this pattern, done by hand
- A subagent automates it: spawn one to review the code just written
- Fresh context means no attachment to the code it reviews
- It never saw the reasoning that produced the bug, so it questions it

When (Not) to Delegate

- **Good:** broad exploration, independent verification, bulk research
- **Bad:** tiny tasks — the spin-up overhead outweighs the work
- **Bad:** anything needing your judgment partway through

A subagent returns a summary, not a transcript — delegate only when the summary is what you actually need.

Git Worktrees

Two Agents, One Repo?

- Two agents in one working tree is chaos — both edit the same files
- Branches alone do not help: only one branch is checked out at a time
- You need a second checkout, not just a second branch

Worktrees: Extra Checkouts

A worktree is a second folder backed by the **same** repository:

```
# create a sibling folder on another branch
git worktree add ../proj-docs docs-branch
# see every checkout attached to this repo
git worktree list
# clean up when done (the branch survives)
git worktree remove ../proj-docs
```

Run a second agent in `../proj-docs` safely — separate files.

Caveats

- Worktrees isolate **files**, not databases, servers, ports, or caches
- Two agents sharing one dev database still collide
- Clean up with `git worktree remove` so stale folders do not pile up

Today: one parallel docs-improvement worktree while your main session keeps working — one stunt, not a habit.

Automation Outlook

Agents Without a Chair

Beyond interactive sessions, agents can run unattended:

- **Headless runs** — `opencode run "..."` with no human in the loop
- **Scheduled jobs** — nightly cleanups, dependency bumps
- **CI bots** — reviewing pull requests or fixing issues automatically

. . .

Powerful, and it inherits every risk from this morning. Guardrails and checks come first. This is a pointer to the literature, not course material.

Lab 5

Your Mission

Five short exercises to put extension into practice:

- Run a **writing skill** on real prose and review the diff
- **Write your own skill** for a workflow you repeat
- Build a verified reading list with the **reference-lookup** skill — open every DOI yourself
- Wire **GitHub's MCP server** into `opencode.json` and use it read-only
- Pull off one **worktree** stunt in parallel

Continue Your Journey

Next Up

- **Lab 5 — MCP and Skills** starts now
- Lab 5 — MCP and skills
- Lecture VIII — Your Research Project — 15:00, bring your problem
- Course literature and references

Bibliography

[1] S. Willison, "The lethal trifecta for AI agents: private data, untrusted content, and external communication." Accessed: July 09, 2026. [Online]. Available: https://simonwillison.net/2025/Jun/16/the-lethal-trifecta/