# Lecture IV — Planning with AI

## AI-Assisted Programming for PhD Researchers

Dr. Tobias Vlćek
Helmut Schmidt University — September 2026

# Why Plan First

## Vibe Coding

- Prompt, run, prompt again, accepting the output without reading it
- Fine for a throwaway script; ruinous for research code
- The first empirical study of the pattern found "impressionistic scanning", not review [1]
- For code you will publish or build on, this is the anti-pattern

## How Unplanned Agent Work Fails

- **Scope drift**: the change quietly grows past what you asked for
- **Architecture by accident**: structure no one chose, no one owns
- **Unreviewable diffs**: 800 lines you cannot meaningfully check
- **"It broke something elsewhere"**: no plan meant no boundaries

## The One-Sentence Rule

A quick test for whether you need a plan at all:

> Tip
>
> If you can describe the change in one clear sentence, just do it. Everything bigger gets a plan first.

# Specifications

## What a Spec Is

- A short document you write before any code
- It fixes four things: **goal**, **inputs/outputs**, **constraints**, **acceptance criteria**
- Not a design essay — half a page is plenty
- It is what you and the agent both agree to

## Acceptance Criteria Do the Work

- Each criterion must be checkable — pass or fail, no debate
- Vague in, vague out: "handles missing values" tells the agent nothing
- Sharpen it: rows with humidity `n/a`, empty, or `-99` become `NaN`
- Now both of you know when the code is done

### Example: Our Cleaning Module

The Lab 2 spec skeleton — you complete it in the lab:

```
# Spec: data loading and cleaning
Module: src/pipeline/io.py — load_raw(path), clean(df).
Acceptance criteria:
  [ ] three timestamp formats -> one datetime column
  [ ] temps unified to °C (temp_unit consulted, then dropped)
  [ ] humidity "", "n/a", -99 -> NaN; decimal commas parsed
  [ ] wind "calm" -> 0.0; column numeric
  [ ] station names stripped of whitespace
  [ ] exact duplicate rows dropped; report how many
Out of scope: statistics, plotting, the CLI.
```

### Specs Anchor the Conversation

- Agents drift — they forget, reinterpret, wander off task
- The spec is the fixed point you keep pointing back to
- Disagreements resolve against the spec, not against memory
- “That is not in the spec” ends most arguments fast

## The Interview Pattern

### Make the Agent Interview You

Paste this before you describe the task:

```
Before writing any code, ask me clarifying
questions about requirements and edge cases,
one at a time, until the task is unambiguous.
Only then propose a plan.
```

Surprisingly effective, and it costs you one extra prompt.

### Why It Works

- You know things you do not know you know — tacit constraints
- Questions surface hidden requirements early
- Early is when they are cheap; after 800 lines, they are not
- The agent’s questions are often sharper than your first brief

## Decomposition

### Verifiable Steps

- Break the work into steps that each end in something you can check
- A passing run, an inspectable output, a green test
- No step longer than ~30 minutes of agent work
- If you cannot name the check, the step is not ready

### Plan Mode Workflow

- In **plan mode** the agent explores and drafts a plan, editing nothing
- You review it, adjust, and only then switch to build mode
- The plan is a contract for the session
- This "generate, then evaluate" loop is what the GenAI education literature recommends [2]

### Reviewing a Plan

Before you approve, ask three questions:

> **i** Note
>
> Does it cover every acceptance criterion? Does it invent scope you never asked for? Does each step end in a check?

## Lab 2

### Your Mission

In Lab 2 you put planning to work on the cleaning module:

- Write the **cleaning spec** — goal, I/O, constraints, acceptance criteria
- Let the agent plan against it in plan mode, then build step by step
- Verify on real station data — not on the agent's say-so

Lab 2 — Spec to Code

## Continue Your Journey

### Next Up

- That wraps Day 1 — you can now plan work an agent can actually execute
- Tomorrow: making agents prove their work, not just claim it
- Lab 2 — Spec to Code starts now
- Lecture V — Making Agents Prove Their Work — tomorrow
- Course literature and references

## Bibliography


[1] A. Sarkar and I. Drosos, "Vibe coding: programming through conversation with artificial intelligence," in *Proceedings of the 36th Annual Conference of the Psychology of Programming Interest Group (PPIG 2025)*, Sept. 2025. doi: 10.48550/ARXIV.2506.23253.

[2] D. J. Bouvier *et al.*, "The Rest of the Robots: Generative AI in Post-introductory Computing Education," in *Proceedings of the 2025 Working Group Reports on Innovation and Technology in Computer Science Education*, Nijmegen Netherlands: ACM, June 2025, pp. 61–107. doi: 10.1145/3760545.3783970.