CONSTRUCTOR UNIVERSITY

AI-Enabled Coding

Instructor: Peter Baumann

email: pbaumann@constructor.university

tel: -3178

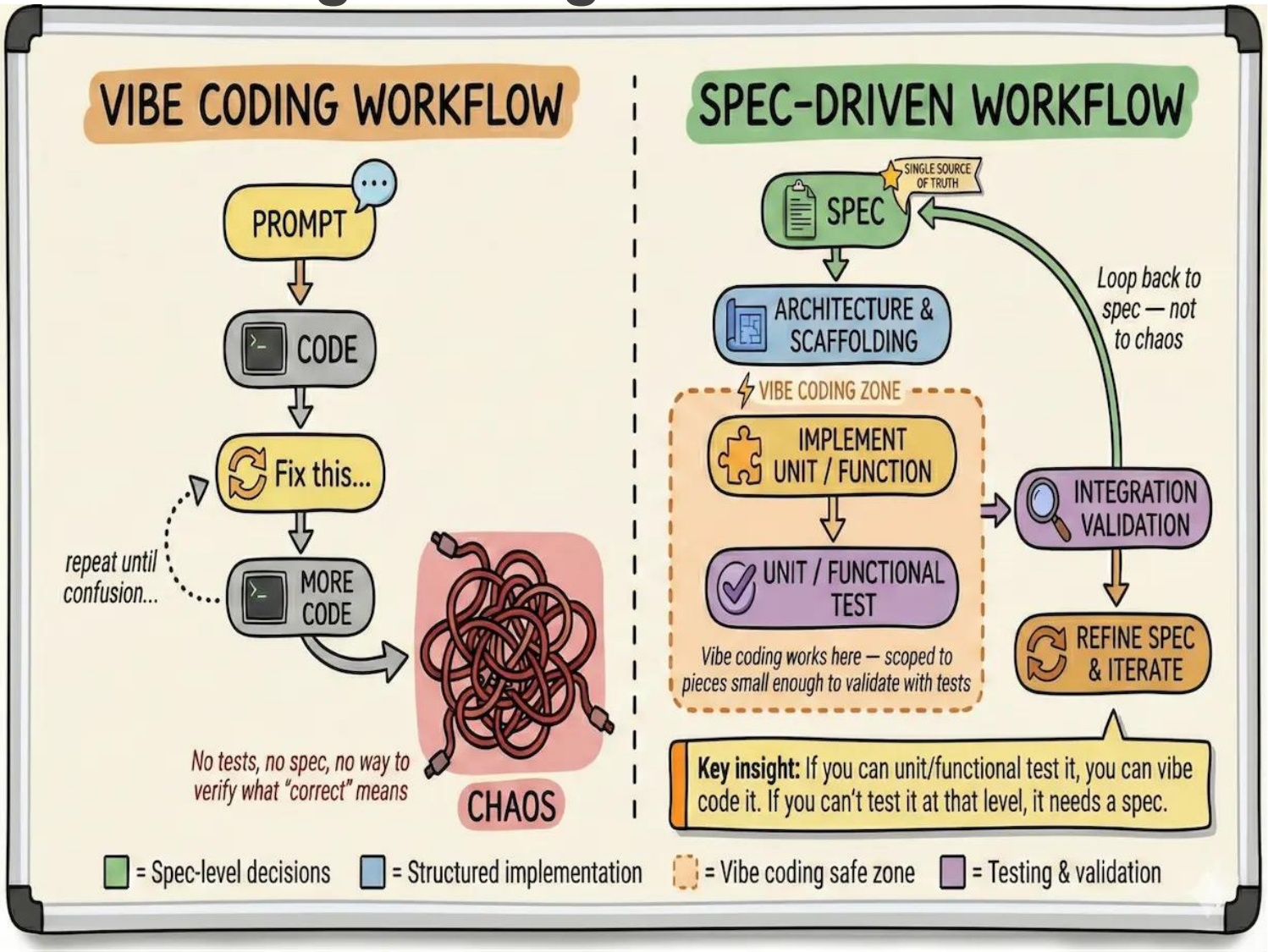
office: room 88, Research 1

Based on material by
Todd Wardzinski & others

AI Entering Coding

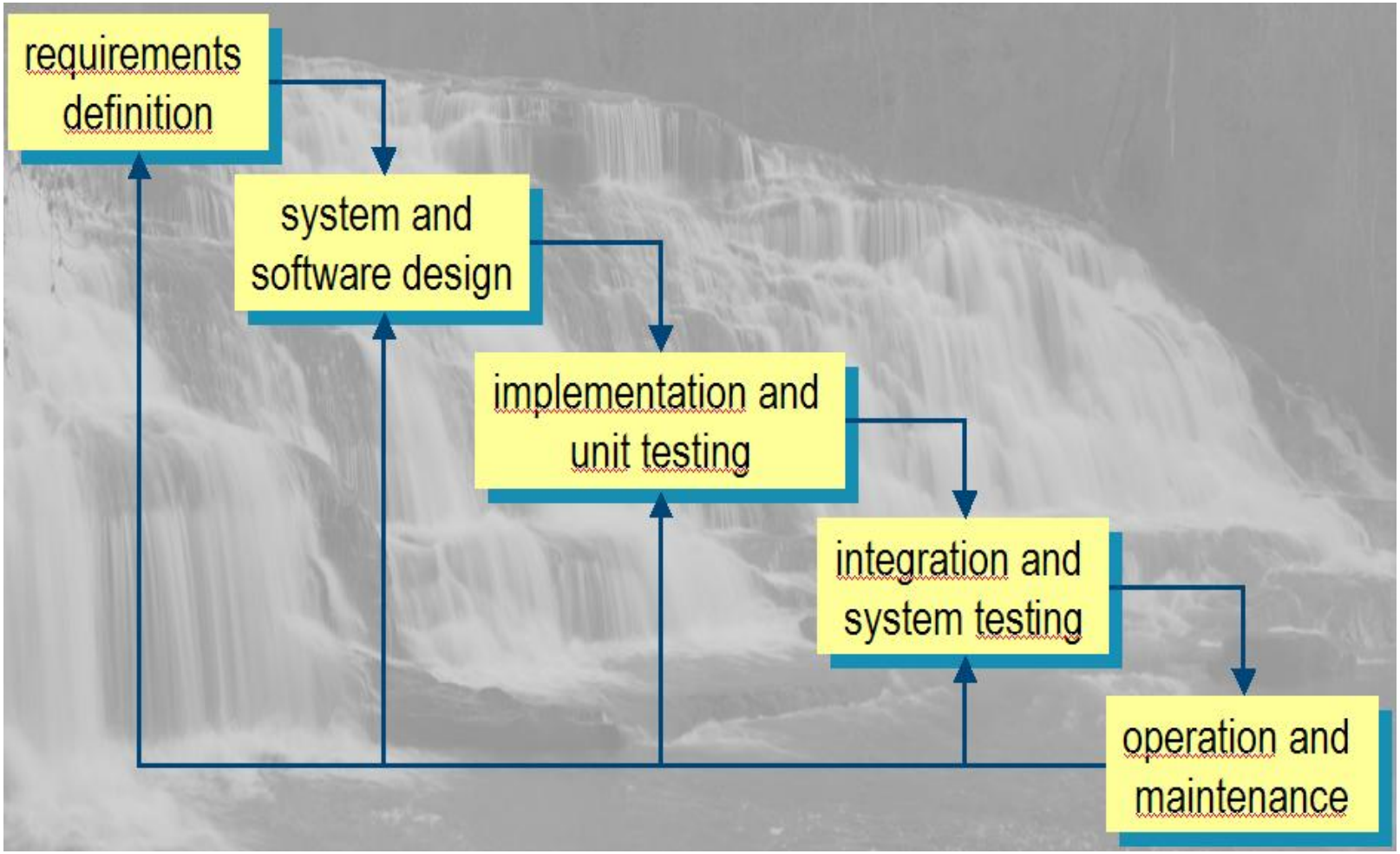
- Generative AI can generate code from written/spoken input
- „Vibe coding“
  - building software by conversing with AI rather than self-writing every line
- Increasing number of increasingly powerful tools
  - Cursor, GitHub Copilot, Windsurf, Claude, ChatGPT, …
- AI = perfect programmer?
- Programmers become jobless?

AI Entering Coding



[Todd Wardzinski / Nano Banana Pro]

AI in the Software Life Cycle?



Evolution of Coding

- 1950s: symbolic PLs - COBOL, FORTRAN, PL/1
- 1960s: structured programming
- 1980s: object orientation, UML
- 2000s: service-oriented architecture
- 2026+: AI-based („vibe“) coding support

- Goals: more safety, reliability, maintainability...and project efficiency

Evolution of Coding: Example

- “Natural-language IDEs”
- Developers will ask:
  - “Show me where the performance bottleneck is.”
  - “Refactor this service to be event-driven.”
  - “Generate a migration plan for the new infrastructure.”
- AI = the new command line
- Challenges:
  - validating logic; ensuring security; enforcing architecture; creating guardrails; etc.

Vibe Coding is...

= coding by ~~emotion~~ intent. “Vibe” = concept, idea, desired outcome

- AI translates into code

- ...good for
  - quick prototype; flashcard app; landing page; ...
- ...bad for
  - Modifying code
    - *“You change one small thing and four other features break. You ask the AI to fix those, and now something else is acting weird.”*
    - *"AI is still just soooooo stupid and it will fix one thing but destroy 10 other things in your code.“*
  - Complex projects; user driven projects, unclear requirements & specification
  - Security, high reliability
    - *Slobsquatting*
    - *Legal consequences? cf. EU Cyber Resilience Act*

“The S in ‘vibe coding’ stands for security.”

Vibe Coding Rules

- Be specific about what you want.
- Be specific about how you want it done.
- Define your constraints.
- Create acceptance tests that verify your specifications are actually implemented.
- Document your decisions.
- Be specific. Don‘t trust results blindly. Find ways to validate.
  - Know when to put AI aside and do it yourself.

The Reality...

- OSGeo:
  - “I've no idea about how much seriousness contributors using AI tools give in reviewing their output, but the reality is that even if they are well intended and super careful and read every line and (believe they) understand what it does, they just lack the experience to have a critical eye (which is the reason in the first place why they needed to use it).
  - The asymmetry of effort between the ease for them to push the PR and the ones bearing on the reviewer's shoulders.”
  - Darafei Praliaskouski, OSGeo Discuss list, 2026-04-11
- Kubernetes folks in similar position, ongoing discussion

Early-Stage ~~Insights~~ Thoughts

- AI can be tool for productivity enhancement
  - In some stages, in others not
  - Never trust results – but how to validate?
- Complexity limits
  - AI's context window can only see fragments
- AI cannot do the job for you
  - Your expertise (and experience!) remains essential
  - Future needs primarily ~~programmers~~ architects

„AI boosts your productivity 10x.“ But what if your coding skills are 0?

- Personal opinion: AI is Wild West now, near future will see development of methods & tools for structured, reproducible, quality-ensuring industrial use