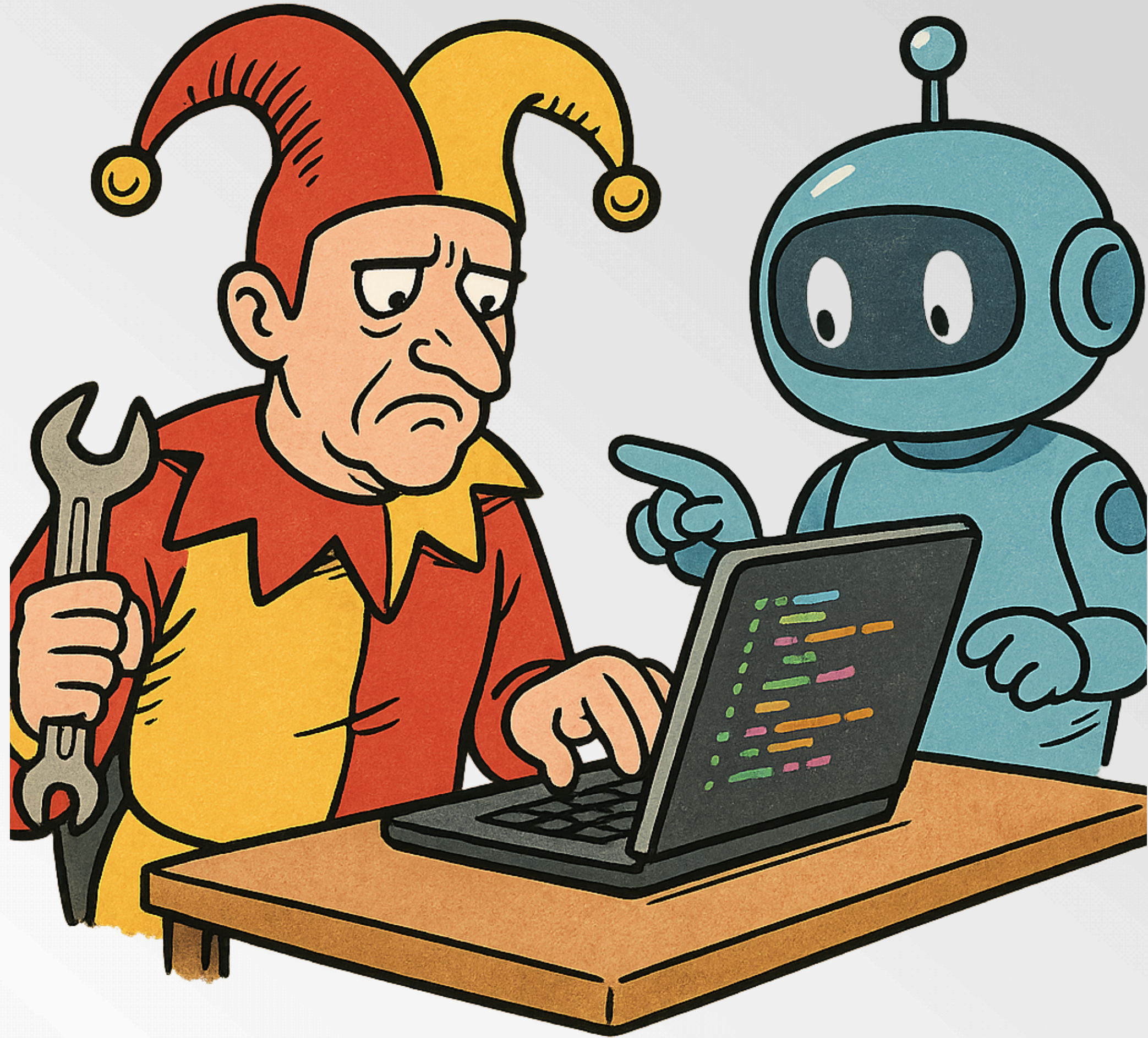




A fool with a tool is still a fool

Why we must AI wisely!

Plone Conference 2025 Jyväskylä

Andreas Jung | info@zopyx.com | www.andreas-jung.com

Recent AI „indicents“ 😂

AI coding tool wipes production database, fabricates 4,000 users, and lies to cover its tracks

Car Dealership Disturbed When Its AI Is Caught Offering Chevys for \$1 Each

Lawyer caught using AI-generated false citations in court case penalised in Australian first

TRAVEL

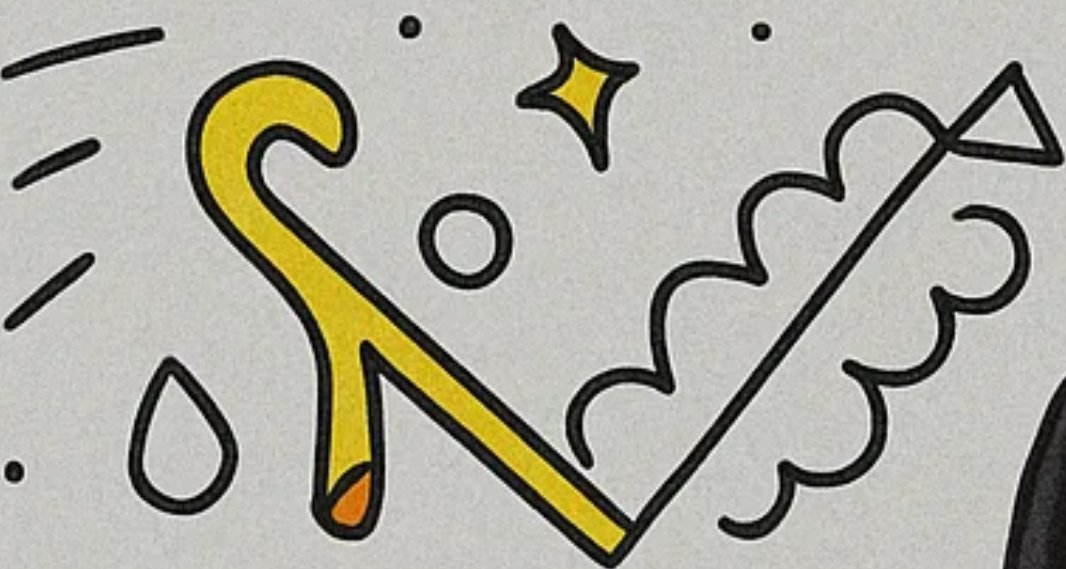
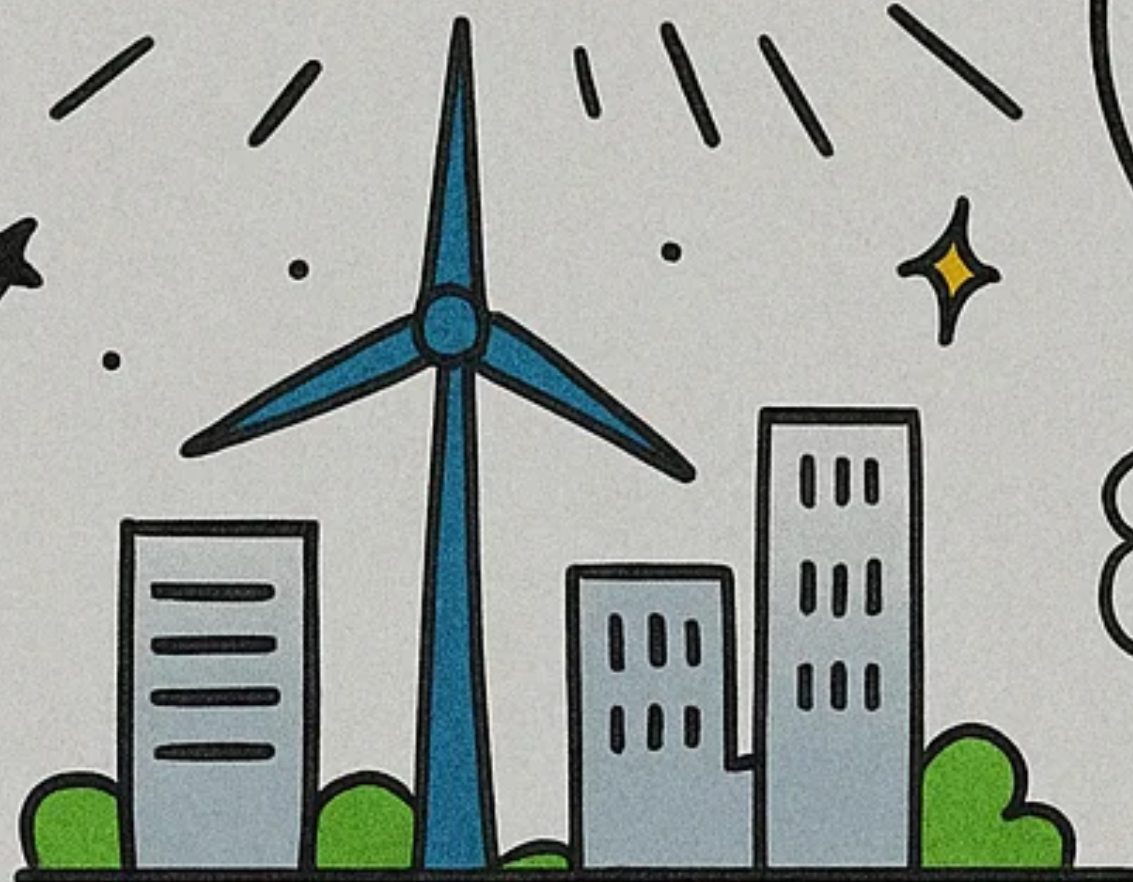
Air Canada's chatbot gave a passenger false information about claiming a discount, and the airline must now pay him compensation

 PYTHON

DEVELOPER
TECHNOLOGY
EVANGELIST

 PLONE

ENERGY



AND MORE



SAVER
OF
PROJECTS



PHARMA

&
MANY
MORE

Agenda

- Current state of AI tools in software development
- What are the safe practices for using AI tools in today's world?
- „Vibe Coding“
 - What is „Vibe Coding“?
 - Experiences
- Conclusions

Recent AI models for software development

Model/Provider	Cost / Access	Context / Max Window
OpenAI – GPT-5	Input \$1.25M / Output \$10M tokens	≈400,000 tokens (≈272k input + 128k output)
Anthropic – Claude 4.5 Sonnet	Input \$3M / Output \$15M tokens	≈200,000 tokens (extended thinking up to 128k output)
Google – Gemini 2.5 Pro	Flash-Lite: Input \$0.10M / Output \$0.40M (higher for Pro/Ultra)	Up to 1,000,000 tokens (Pro/Ultra tiers)
Meta – Llama 4 (Scout / Maverick)	Open weights (infra cost depends on deployment)	Scout: up to 10M tokens; Maverick: ~1M tokens
Microsoft – Copilot (GitHub/M365)	~\$20/month Copilot Pro; API costs match chosen model	Inherits from underlying models (GPT-4.1, GPT-5, Gemini etc.)

- Token window = how much code AI can “see” at once.
- One line of code ≈ 5–10 tokens, so window size limits how many lines fit in context.
- Models, numbers and pricing changing almost every day

Where AI works reliably?

- Generate boilerplate/initial codebase code for new projects
- Create tests and test data
- Explain & document code
- Write project documentation, installation notes etc.
- Support for refactoring, migrations

✖ Where AI notoriously fails and fools us/me

- Lack of self-reflection regarding
 - completeness of work
 - consistency of work
 - correctness of work
-  AI constantly lies to me

**2½ successful AI stories
in the last 12 months**



Guarantees Of Origin - certificates in renewable energy

- FastAPI application deployed in Azure,
- Integrations with various other E.ON systems
- 🤯 over-designed architecture, under-engineered implementation, 50% coverage, hard to maintain and extend (blaming the junior dev)
- Claude Code:
 - 70 → 400 tests, coverage 50% → 95% 😎
 - refactoring and highly improved CI/CD in Azure DevOps
 - foundation for further refactoring



University of Saarbrücken

Central university shop and procurement system

- 2015: Plone 4.3 site (Dexterity), ArangoDB, on-premise hosting
- 2025: migration to Plone 6.1, Python 3.12, Azure hosting, updated functionality, decent UX, docs
- Claude Code for
 - complete source code documentation
 - deployment configurations for Azure (Terraform, Ansible)
 - massive migration from jQuery to native Javascript
 - consistent view and templates, highly improved UX
 - shiny, complex forms (originally ugly Dexterity forms)
- Claude Code enabled me to master deployment and frontend alone (not my primary skill or interest).

Open Grid Europe (OGE)

Simulation of hydrogen distribution networks in Germany

- 12.000 km gas backbone distribution network in  aka gas pipelines
 -  government decided to enable distribution networks for hydrogen (Ukraine war, Germany's dependency on Russian gas)
 - completely new territory:
 - gas and hydrogen have very different physical parameters (density)
 - different behavior inside pipelines (pressure and flow)
 - new network components like electrolyzers (hydrogen producers)
- a simulation software is needed

AI detected

A

CHIEF PHYSICIST
BABYSITTER
(SOFTWARE DIVISION)

```
(cod::  
[--o;)  
[= (oo;):  
[=
```

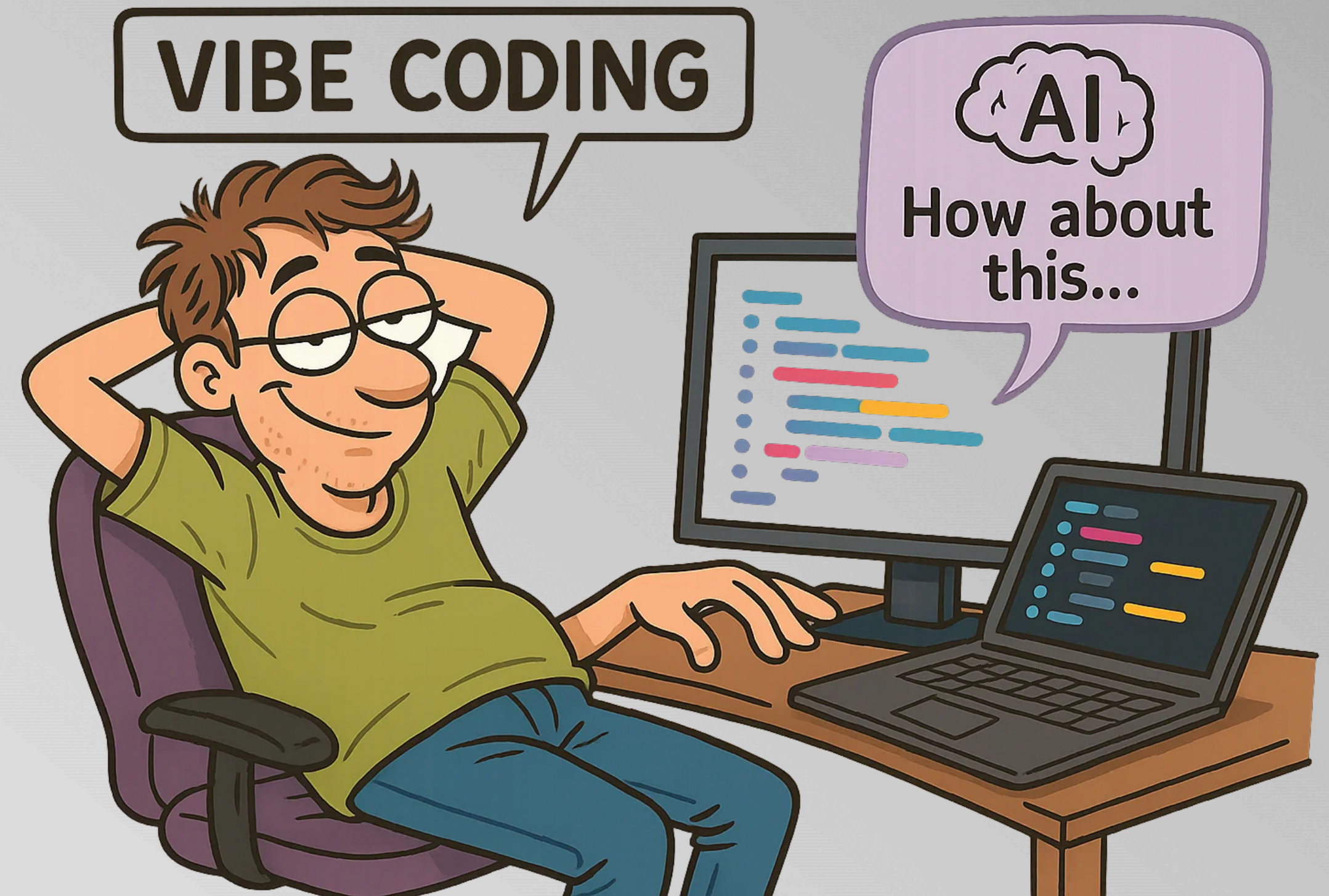

🤔 Open Grid Europe (OGE)

Simulation of hydrogen distribution networks in Germany

- AI development totally gone wrong
 - Physicists/non-programmers „trying“ to program using AI
 - AI-generated code doing **something**
 - AI-generated code thrown over the fence because „AI got stuck at some point“ or they got stuck with AI
 - no tests
 - no formal specifications
 - 🤯😭🤔
- Claude Code:
 - ~ 300 generated tests as foundation for refactoring
 - ..and then the project budget was exhausted 🤪💖



Vibe Coding



Vibe coding is a software development method that involves giving natural language prompts to a large language model (LLM) to generate code, focusing on iterative experimentation with the AI's suggestions rather than deep, traditional code review. It's a hands-off, AI-driven approach that makes creating small, low-stakes software projects like prototypes and simple apps faster and more accessible, though it introduces significant risks to security and maintenance in complex systems.

- Andrej Karpathy (former Sr. Director of AI at Tesla)

Vibe Coding: Market Promises vs. Reality

The promises

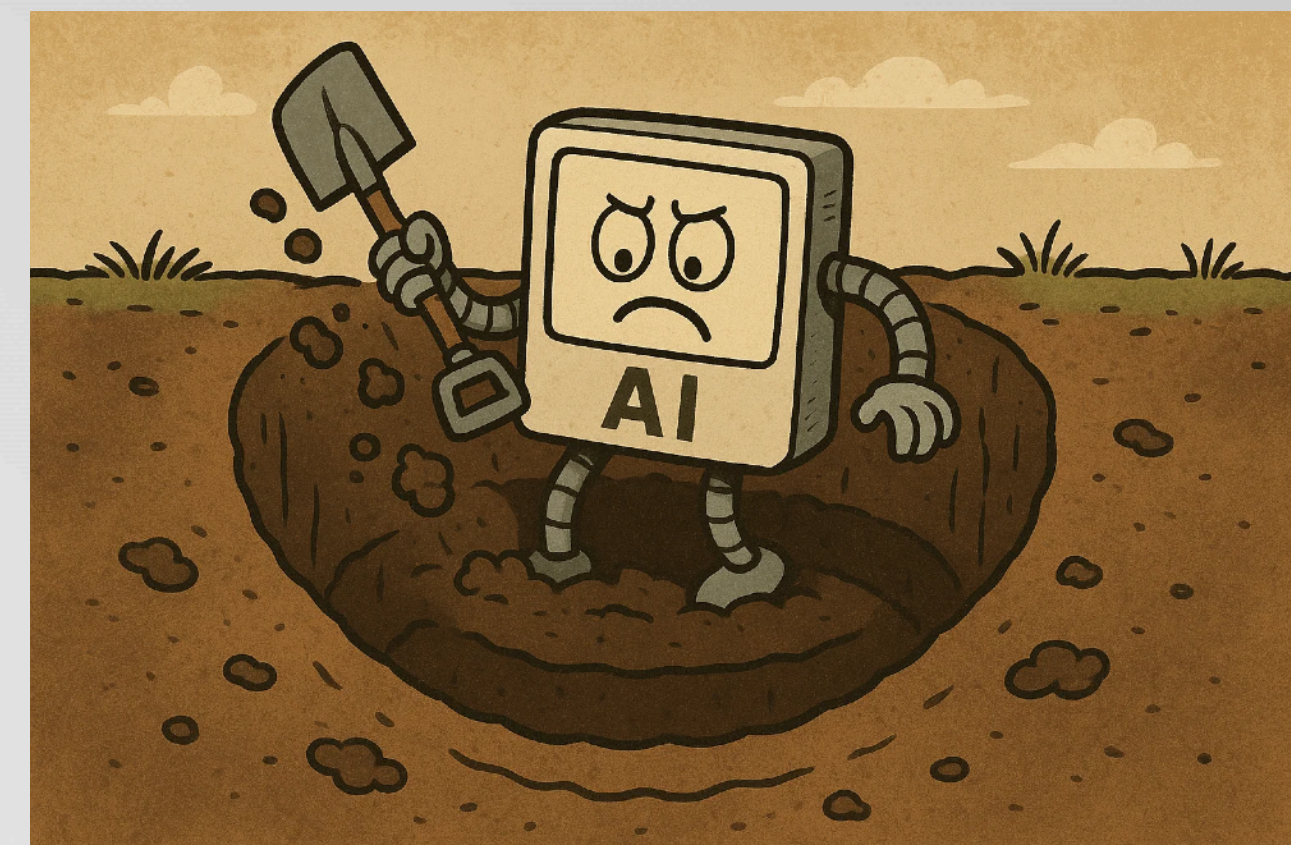
- Democratize coding → “anyone can build software”
- Faster time-to-market, rapid prototyping
- Lower cost & resource needs
- Direct idea → product flow
- New developer role: orchestrator/prompt engineer
- More innovation, more experiments



Vibe Coding: Market Promises vs. Reality

Own experiment: relaunch of my website zopyx.com

- old: Wordpress, new stack: AstroJS, Tailwind CSS, 3 languages, Cursor IDE.
- ✅ Marketing texts and wording based on existing website, CV, and hints.
- ✅ Rapid initial iterations with promising outcomes (shiny, nice Tailwind usage)
- 🤔 Subpages generated with random AstroJS component structure.
- 🤔 Ongoing refactoring problematic (destruction of its own code).
- 🤔 Language trees: incomplete work, inconsistent work, and painful iterations.
- 🤔 AI **constantly** lying at you, doing incomplete work, lying about progress.
- 😭 AI digging its own hole with little chance to escape from by itself.



Vibe Coding: Market Promises vs. Reality

The reality

- Works best for MVPs, scaffolding, internal tools
- Human oversight essential (debugging, testing, validation)
- Fragile code, tech debt, security risks
- Limited by context windows & model accuracy
- Poor fit for complex, mission-critical systems
- Adoption is incremental, not disruptive overnight



Vibe Coding: Market Promises vs. Reality

Other issues and consequences (excerpt)

- **Unknown provenance of AI code** → treat outputs like unverified paste.
- **Copyleft spillover risk (GPL/AGPL)** → accidental inclusion can force disclosure.
- **No copyright in purely AI-generated code** → only your human edits are protectable.
- **EU AI Act = transparency, not liability shield** → you still own compliance.
- **Governance controls required** → run SCA/clone scans, keep logs, add notices, check license compatibility, maintain indemnity settings.
- **Patent exposure (FTO)** → generated code may implement patented methods; verify before shipping.

Responsibilities for (us) senior developers?

Vibe coding doesn't reduce a senior dev's responsibility

- **Accountability doesn't move:** We own correctness, security, and delivery.
- **AI governance:** We define where AI is allowed, with guardrails and red-zones.
- **Quality gates:** "No unreviewed AI code." (Tests, benchmarks, threat modeling)
- **Licenses & provenance:** Run SCA/clone checks; block unclear/GPL/AGPL drift.
- **Architecture first:** Use AI for scaffolds; humans own contracts & non-functionals.
- **Security posture:** Treat AI output as untrusted until scans/review pass.
- **Auditability:** Tag AI-assisted PRs; keep prompt/output logs; maintain SBOM/"AI-BOM".
- **Metrics:** Track defect density, rework, coverage/perf deltas, and token spend.

**AI is a tool.
But wisdom is the difference.**

THANK YOU!

Questions?

