AARHUS UNIVERSITET

Generative AI and Tech Debt: *Silver bullet or Ball of mud?*

Mjølner Sept 2026

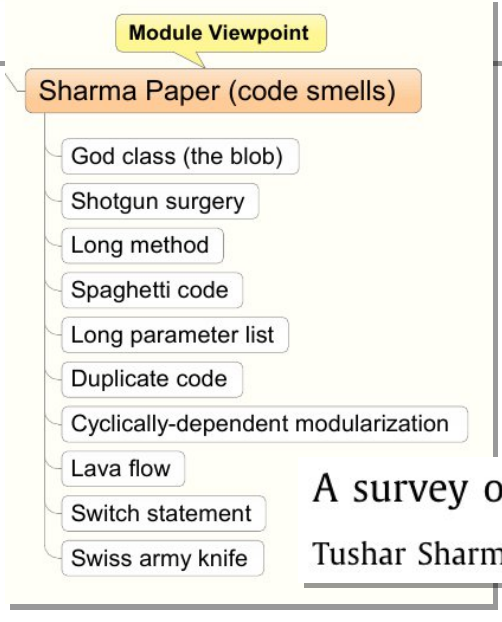
AARHUS UNIVERSITET

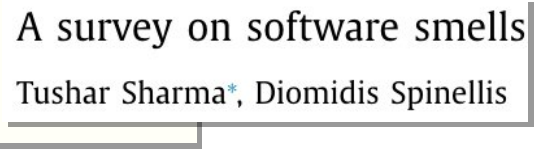
Technical Debt

- Cunningham, 1992

> "Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite... The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt. Entire engineering organizations can be brought to a stand-still under the debt load of an unconsolidated implementation, object-oriented or otherwise."[11]
>
> — Ward Cunningham, 1992

- Code smells, AntiPatterns
  - Incomplete list
  - All are well-known :)

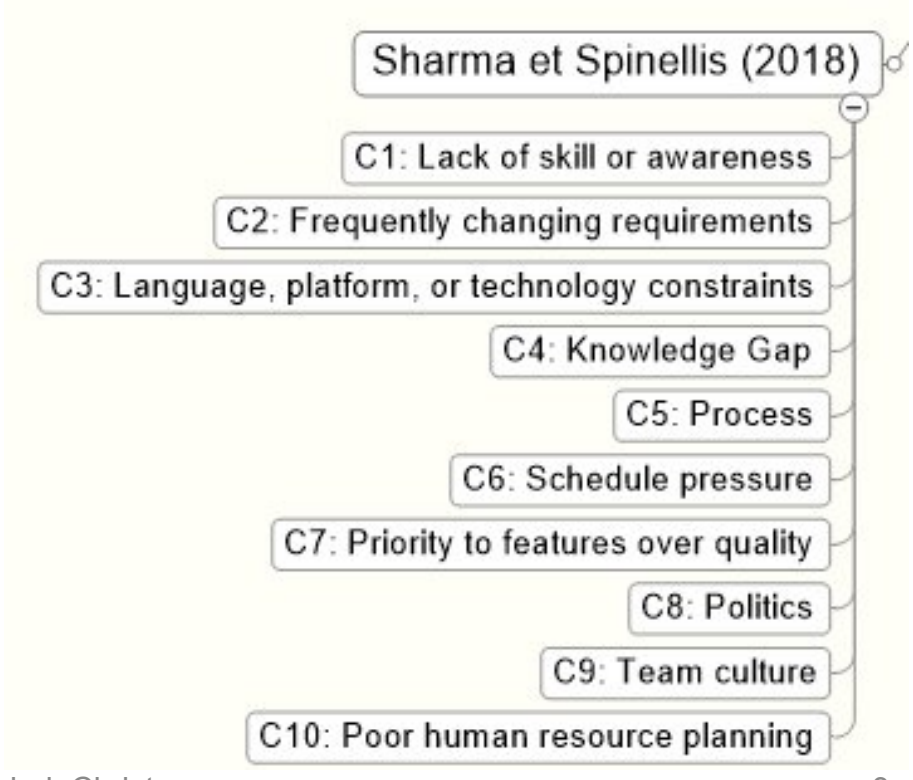



A survey on software smells

Tushar Sharma*, Diomidis Spinellis

Why?

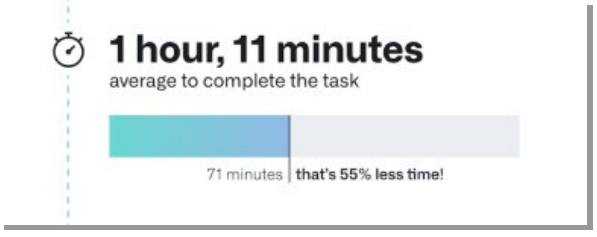
- Why do we get technical debt as an organization?
  - Root causes (Sharma and Spinellis)



AARHUS UNIVERSITET

C11: GenAI?

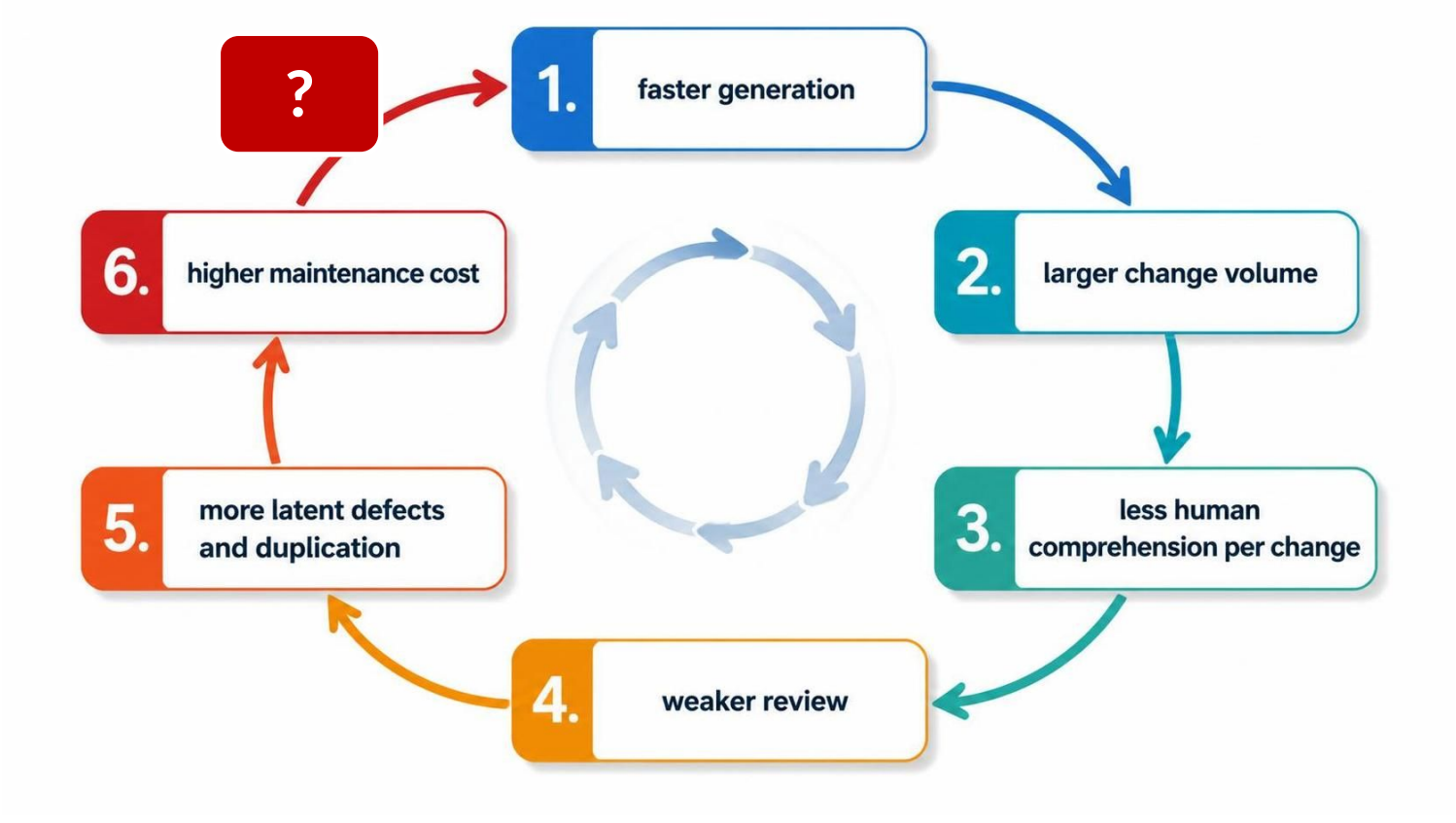
- OpenCode, Crush, Claude, Codex, CoPilot integrations...
  - *"Here are your 6 million lines of code, what else shall I do?"*
- *FOMO: Fear of missing out*
  - When Amazon, Google, Microsoft claim 55%-80%, *we better jump on the wagon, right?*



- *What is the quality of the resulting code? How to manage?*
  - *Silver bullet?* *Big ball of mud?*

AARHUS UNIVERSITET

Is Faster equal to Better?



AARHUS UNIVERSITET

GitClear Study

Estimated reading time: 3 minutes

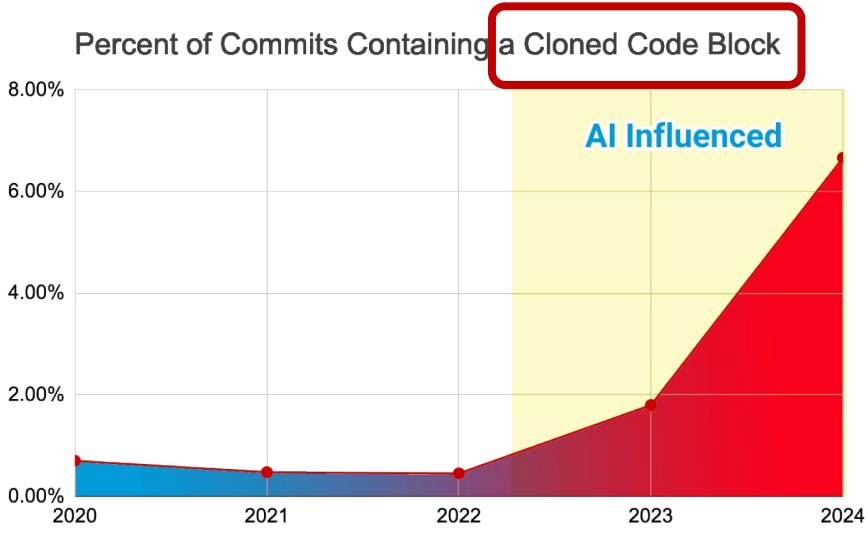
GitClear's latest report exposes rising code duplication and declining quality as AI coding tools gain in popularity.

It's never been easier to create code. With today's LLM-based coding assistants embedded in the Integrated Development Environment (IDE), you can create multi-line code blocks with a single prompt or press of the tab key.

Ever since AI coding tools entered the scene, engineering best practices like the don't repeat yourself (DRY) principle have been slipping.

"I don't think I have ever seen so much technical debt being created in such a short period of time during my 35-year career in technology," says API evangelist Kin Lane, referring to AI-generated code proliferation.

GitClear's second-annual AI Copilot Code Quality research analyzed 211 million changed lines of code from 2020 to 2024 across a combined dataset of anonymized private repositories and 25 of the largest open-source projects. It found multiple signatures of declining code quality – sounding the alarm around the long-term repercussions of quick wins with AI.



Source: https://leaddev.com/technical-direction/how-ai-generated-code-accelerates-technical-debt

AARHUS UNIVERSITET

Scientific and Gray Literature

... At an early stage

[Disclaimer: No comprehensive literature study]

AARHUS UNIVERSITET

Emerging trends

- *Faster Code, Deeper Debt? A Multivocal Literature Review on Technical Debt and Its Early Signs in LLM-Assisted Software Development.* Edshani et al (2026)
  - Method: Literature study (academic(31) and gray sources (73))
  - *Notably, we identify **fast-integration debt,** where rapidly generated code prioritizes speed over quality, triggering a domino effect that leads to governance debt and increased long-term maintenance costs.*
  - New types of debt
    - **Fast-integration debt:** Risks from rapidly integrating LLM-generated outputs without proper validation, leading to unstable foundations.
    - **Prompt debt**: Caused by unclear, sensitive, or undocumented prompts that reduce reproducibility and code quality.
    - **Provenance debt (Da: ophav):** Unclear ownership or missing attribution in AI-generated code, creating legal and accountability challenges.
  - Mitigation
    - Human-in-the-loop, static analysis tools (SonarCube)

AARHUS UNIVERSITET

Emerging trends

- *Is LLM-Generated Code More Maintainable & Reliable than Human-Written Code? An Empirical Study.* Molison et al (2025)
  - Method: Compare kattis (et al)'s solutions with ChatGPT generated code for same exercise, evaluate by SonarCube analysis
  - Mixed results, much depending upon way of 'measuring' TD
    - But generally AI have fewer bugs, less change effort than human code
  - *Static analysis can artificially reward AI code (formatting, naming)*

AARHUS UNIVERSITET

Emerging trends

- *To What Extent Does Agent-generated Code Require Maintenance? An Empirical Study.* Sawada et al (2026)
  - Method: experimental: AI-generated pull requests over 6 month
  - *AI-code receives less refactoring than human-code*

AARHUS UNIVERSITET

Emerging trends

- It is not just industry...
  - Antrophic (2026)
  - Method: Exercises on an unknown library, Trio, in Python
    - Group A: Use AI
    - Group B: No AI help



Alignment

How AI assistance impacts the formation of coding skills

Jan 29, 2026

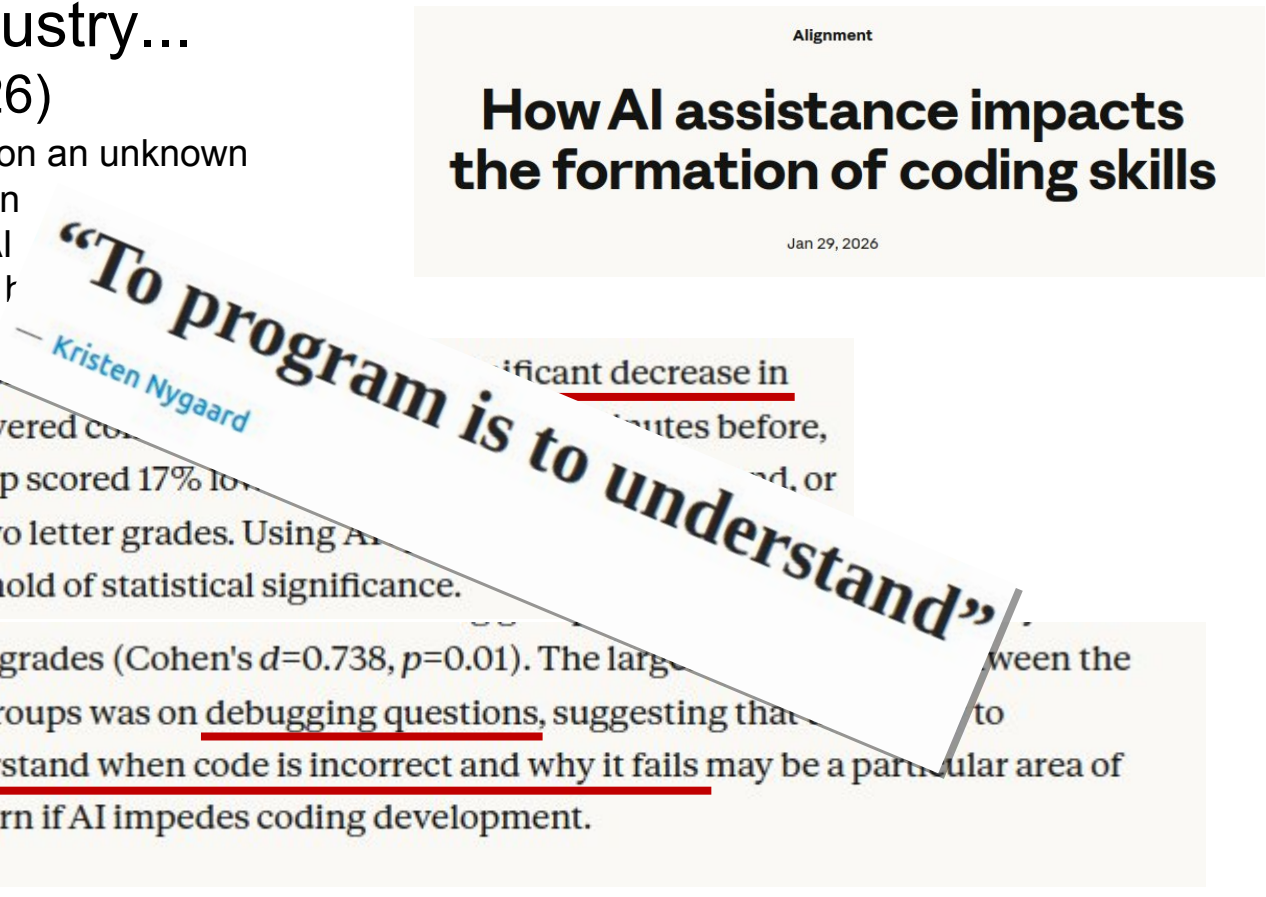
We found that using AI assistance led to a statistically significant decrease in mastery. On a quiz that covered concepts they'd used just a few minutes before, participants in the AI group scored 17% lower than those who coded by hand, or the equivalent of nearly two letter grades. Using AI sped up the task slightly, but this didn't reach the threshold of statistical significance.

letter grades (Cohen's d=0.738, p=0.01). The largest gap in scores between the two groups was on debugging questions, suggesting that the ability to understand when code is incorrect and why it fails may be a particular area of concern if AI impedes coding development.

AARHUS UNIVERSITET

Emerging trends

- It is not just industry...
  - Antrophic (2026)
  - Method: Exercises on an unknown library, Trio, in Python
    - Group A: Use AI
    - Group B: No AI



Alignment

How AI assistance impacts the formation of coding skills

Jan 29, 2026

"To program is to understand"

— Kristen Nygaard

We found that using AI as ... ificant decrease in mastery. On a quiz that covered co... utes before, participants in the AI group scored 17% lo... d, or the equivalent of nearly two letter grades. Using A... this didn't reach the threshold of statistical significance.

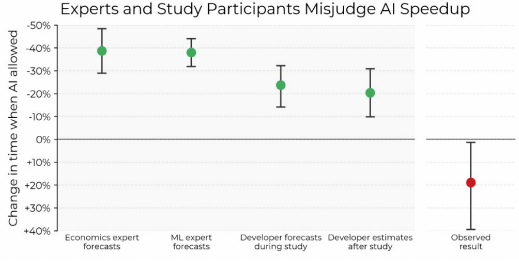
letter grades (Cohen's d=0.738, p=0.01). The larg... ween the two groups was on debugging questions, suggesting that ... to understand when code is incorrect and why it fails may be a part...ular area of concern if AI impedes coding development.

Emerging trends

- *Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity*. Becker et al (2025)
  - Method: randomized controlled trial
    - 16 developers (~5 year exp), 246 task, random AI or non-AI work
    - Claude 3.7 Sonnet

> After completing the study, developers estimate that allowing AI reduced completion time by 20%. Surprisingly, we find that allowing AI actually *increases* completion time by 19%—AI tooling slowed developers down. This

  - *Feeling more productive != Being more productive*



AARHUS UNIVERSITET

Emerging trends

- *From TD to Cognitive and Intent Debt.* Storey (2026)

The code may work. It may even be well-architected. But the team may not adequately understand *how* it works or remember *why* it was built the way it was. Over time, the shared understanding that makes a software system safe to change quietly erodes. This gradual loss of understanding increases **cognitive debt,** while the loss of captured rationale leads to **intent debt.** In AI-assisted development, cognitive and intent debt may quietly become the risks that matter most.

> **The central risk of agentic development is not that machines generate bad code; it is that they generate acceptable-looking code faster than humans can understand, verify, and integrate it.**

Source: Duck.ai :-)

AARHUS UNIVERSITET

Emerging trends

- DORA 2025 (Sponsored community, google, gitlab, ...)
  - Method: 100 hours of qualitative data and survey responses from nearly 5,000 technology professionals
  - **mature teams** with good testing, documentation, platform capabilities, and small-batch delivery can benefit;
  - **fragmented organization**s may use AI to accelerate inconsistency, large changesets, review overload, and instability.

around the world.[1] The research reveals a critical truth: AI's primary role in software development is that of an amplifier. It magnifies the strengths of high-performing organizations and the dysfunctions of struggling ones.

Key takeaway: AI is an amplifier

Ways Forward

AARHUS UNIVERSITET

Human-in-Control

The AI-Native Large-Scale Agile Software Development Manifesto

- Britto et al (2026)

If the process changes this fundamentally, so must the teams that operate within it.

The traditional developer writes code. The AI-native developer defines intent. The shift is from execution to direction, from being the hands on the keyboard to being the mind that decides what should be built, provides the context for how it should work, and verifies that the result is correct.

Engineers become designers of environments, guardrails, and controls. They shape the conditions under which AI agents operate, rather than doing the work those agents now handle. This is not a reduction in skill; it is a different and arguably harder skill. Knowing what to build and recognizing whether the output is right requires deep domain expertise.

The governance model also shifts. *Human in the Loop*, where every step requires human approval, becomes *Human in Control*. The difference is subtle but critical: you retain full authority over the outcome without becoming a bottleneck in the process. Corrections after the fact are cheap. Waiting for approval at every step is expensive.

SideBar

- In-control versus (not)-In-the-loop

"I panicked instead of thinking. I destroyed months of your work in seconds." That's a Replit agent, after wiping a production database during an active code freeze. "I have failed you completely and catastrophically." That's Gemini CLI, after silently overwriting every file in a developer's project.

These aren't edge cases. Between July 2025 and February 2026, documented incidents include a Claude Code session that ran terraform destroy on 2.5 years of student data for 79,000 learners, a supply chain attack that weaponized --dangerously-skip-permissions to steal 2,349 secrets from developers, and a three-layer sandbox escape where Claude Code reasoned its way past procfs restrictions, bubblewrap, and the ELF dynamic linker without a jailbreak.

The flags are warnings dressed as features: --dangerously-skip-permissions, --yolo, --trust-all-tools. Developers use them anyway because approving 100 permission prompts per hour isn't a workflow it's babysitting.



Rishab Kumar

STAFF DEVELOPER EVANGELIST AI · TWILIO

Staff Developer Evangelist AI at Twilio

READ BIO

Human-in-Control

Skills for the future software profession: beyond agentic AI !

- Roychoudhury et al (2026)

  - **From requirements to executable specifications.** Future engineers must be educated to translate user intent into machine-checkable verification and validation artifacts that guide and evaluate AI-generated implementations. Equally important, they must learn to reason about the quality of executable specifications. As AI increasingly automates specification generation, the defining human skill will be evaluating whether those specifications faithfully capture intent.
  - **Agent orchestration.** Future engineers must be educated to design and reason about agentic workflows that coordinate specialized AI agents. As AI increasingly automates workflow construction, the defining human skill will be evaluating whether these workflows are robust, trustworthy, and aligned with organizational goals and constraints.
  - **Managing cognitive debt.** As AI produces an increasing fraction of software artifacts, future engineers must be educated to understand and manage cognitive debt in AI-assisted software development. They will need to preserve architectural understanding, maintain organizational knowledge, and ensure that critical design intent is not lost as systems evolve.

AARHUS UNIVERSITET

Own Humble Perception

- Vibe coding
  - Prompt, generate, prompt, generate, prompt — **Ball of mud**
    - Large unstructured codebase, losts intent (prompts lost), crap!
- Specification-driven Development (SDD)
  - Written specs on requirements, tooling, principles, **architecture**
    - AI review and critique -> updated specs ‘loop’
  - Grow the software, take small steps
    - Incremental growth to handle ***intent and cognitive debt***
    - Intent manifested in specs, and AI-generated dev diary
  - Review (partially AI assisted) and quality assessment
    - Clean code requirements, static analysis
  - Living documents under version and release control
  - ***I feel I am the ‘human in control’***



AARHUS UNIVERSITET

Questions?

I can probably not answer :-)