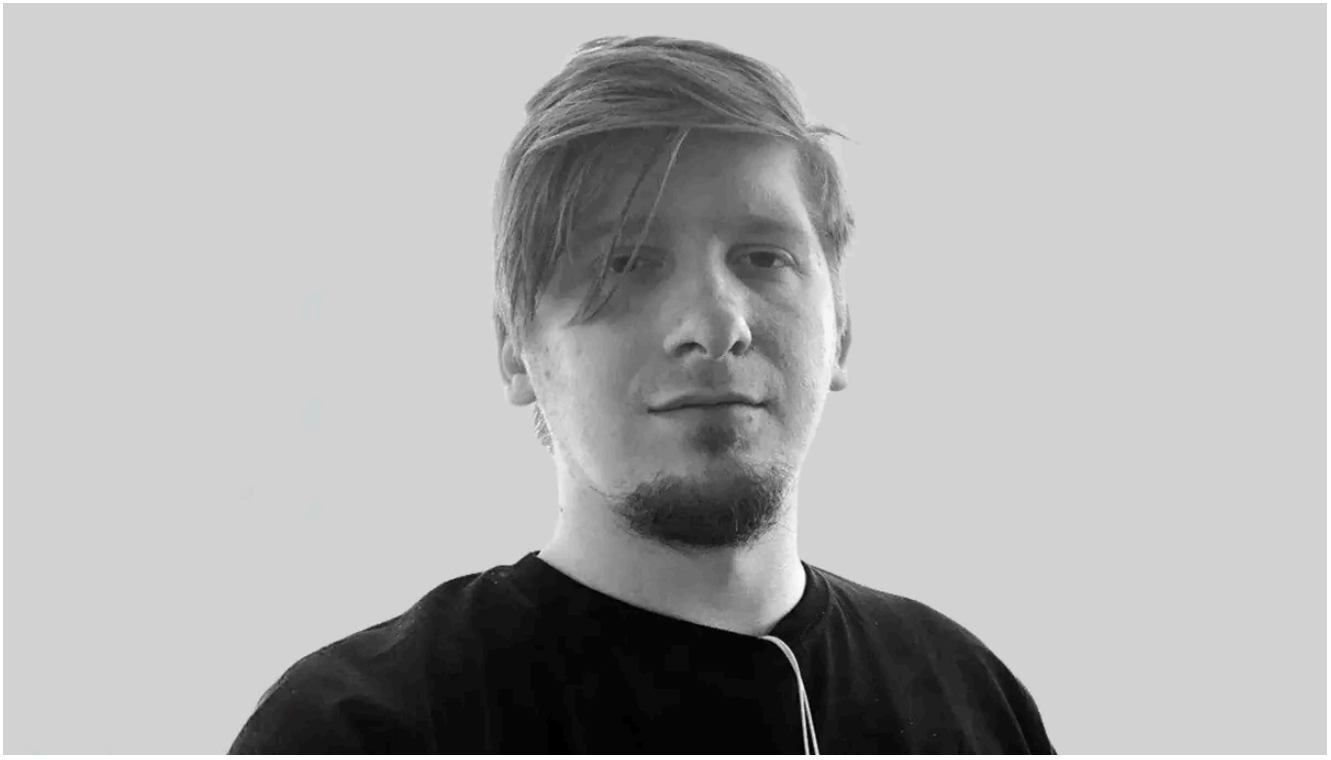
30.03.2026 7 MIN READ

“We're about a year away from a critical moment”

AI AND ML ARCHITECTURE PROGRAMMING CULTURE

Denis Tomilin, Solutions Architect, explains how technical debt is snowballing as AI generates more code, the difference between human slop and AI slop, when manual coding will be back in fashion, and the two types of developers that will survive in the future.

Summarize with AI →



ARTICLE AUTHORS

Asya Chachko
Editor-in-Chief

What is the “technical debt” problem that people have been discussing a lot lately in connection with AI?

— Technical debt occurs when an LLM or a person produces code without fully understanding the answers to important questions, so those questions are postponed for later.

With humans, the reason is usually laziness. AI, on the other hand, makes assumptions about the task and selects the most common solutions.

For example, you say: “AI, build a website.” Without clarifying the requirements, the model will pick the most popular framework, like Next.js, the most popular backend, and the most popular database. It tries to solve the task as quickly and simply as possible.

Then you look at the result and say: “Great, now let’s see how this works on a mobile device.”

At that point, the LLM goes back into the code and realizes the system doesn’t support mobile apps and would need to be rewritten.

That’s technical debt. The simplest decisions were made at the beginning, and later proved to be wrong.

Is there a difference between manually written code and AI-generated code in this case?

— There's human slop and AI slop — people write bad code, too. There’s a prejudice that code written by AI is messy and unverified. But the quality of code from modern models like Opus is already quite high — that's not where the problem lies.

The real issue is that AI agents produce a huge amount of code. There’s a lot of code on GitHub that was written by AI agents, but users sometimes mark it as manually written. As a result, in the overall dataset, we sometimes don’t even know which code was written by AI.

So, the popularity of AI-assisted programming contributes to the growth of this debt?

— There is one thing that cannot be replaced: ownership. Only a human really knows where the system is going and what decisions need to be made for the future.

With AI, we can write code very quickly. But the time we save should be spent on making architectural and product decisions. Unfortunately, developers often don’t do that and postpone decisions until later.

Eventually, those gaps show up as errors caused by decisions that were never made or were made incorrectly.

These errors accumulate, and we’re already starting to see a snowball effect.

Can you predict when this debt might reach a critical mass?

— This is very speculative. But there is a theory that the cadence of software in the enterprise sector is about two years.

For the first two years, we build, support, and adapt some software. Then the next two years are spent on refactoring. We realize what needs to be rewritten, fixed, or which providers need to be replaced. After that, the system becomes stable again for another two years.

Following that logic, roughly speaking, we’re about a year away from a critical moment, likely occurring in Spring next year. It won't be one dramatic event — instead, there will be a growing realization across teams that nobody fully understands the systems they've built. Around that time in Spring, news stories are likely to emerge, claiming everything needs to be rebuilt.

That’s when many people will start saying that we need to go back to manual coding. The real shift won't be back to manual coding entirely, but toward a 70/30 split: AI for boilerplate, humans for architecture and novel problems.

Why do you say this is speculative?

— Because by the time we reach the refactoring phase of poor-quality code, two things might happen. Either the models will evolve, or people will simply learn how to use the tools correctly.

Problems will still appear, someone will get blamed — but it will probably just become another reason to switch vendors.

You said the amount of AI-generated code is growing very quickly. If AI starts learning from code written by AI, won’t the quality decline?

— Right now, many companies are hiring developers for very good money with a simple task: you write code for specific tickets. You write one piece of code and submit it, then write another and submit it.

These companies use that code as training data to fine-tune their models — essentially learning from expert examples. Once they have enough, they'll say, 'Thank you, goodbye.' So in the near future, AI will be able to write very clean code.

Will programmers still be needed then?

— Models will keep getting better at writing code. Fewer and fewer decisions will need to be made by humans — and fewer developers will be needed. That’s clearly the trend.

For routine tasks, code will be generated automatically without human involvement.

But for innovation — for example, a brain chip and its interface — the code hasn’t been written yet. AI doesn’t know how to work with those things, and the risks are too high.

So wherever there are many unknowns and unexplored paths, we will still need senior developers. There will be fewer of them, but they will still be necessary.

I think in the end, two types of engineers will remain. The first type will be developers who deeply understand the business domain and engineering and can use LLMs to design solutions. They may not write code in traditional programming languages — they might work in higher-level specifications or domain-specific languages — but they still need to understand what the system does and why.

The second type will be developers working on innovation. These will be real technical specialists who can write and read code — in a sense, classic programmers.

Right now, hiring for junior developers has dropped sharply. People aren't being trained anymore — and in my opinion, that's a strategic mistake.

We recently had a hackathon focused on vibe coding. The organizers concluded that the scope of vibe coding is very limited and will likely remain so.

— I disagree. Having only one role is no longer enough, as many things are being optimized.

We already need PMs who can vibe-code. We need QAs who can also be business analysts or release managers — and they should be able to vibe-code too.

The scope of vibe coding as a skill for non-developers – PMs, QAs, analysts – is enormous and undervalued.

How do you see the role of a modern architect?

— Architects definitely need to know how to vibe-code. Modern architects should function more as consultants. Last year, every project sale I worked on already included a ready mini-application — a proof of concept.

Architects must be able to validate their assumptions very quickly. Ideally, a client explains on a call what they want to build, and at the same time, AI agents transcribe the conversation in real time, conduct research, and offer suggestions: ask about this, propose that feature.

Where should people look to stay up to date on innovation?

— You should look at China. Not just at the models – DeepSeek and Qwen are already open-source and production-viable – but at the engineering decisions behind them. Alibaba hosts thousands of MCP services and Qwen natively supports it.

But what's really striking is the breadth of adoption in China. AI agents are embedded into super-apps – you order food, book flights, pay bills without leaving the chat. They're in smart glasses with voice-activated payments. They're in schools. It's not "here's a chatbot for developers" — it's AI embedded into how people already live, learn, and pay for things.

Many of these ideas started as copies, but in the last two or three years, the quality has improved enormously. They are increasingly competitive and may become major providers of powerful open-source models.

Frequently Asked Questions →

AI AND ML ARCHITECTURE PROGRAMMING CULTURE SHARE

You may also find this interesting

BLOG POST

BLOG POST

"I find pleasure in coding. So why should I give the most fun part to AI?"

AI Coding Assistants: Helpful or Harmful?

MOST WANTED 1 of 3

Senior Power BI Engineer

ARMENIA, BRAZIL, BULGARIA, COLOMBIA, CYPRUS, GEORGIA, INDIA, KAZAKHSTAN, LATVIA, MEXICO, POLAND, REMOTE.LATAM, ROMANIA, SERBIA, UKRAINE, URUGUAY

We are looking for a Senior Business Intelligence Specialist to design impactful data visualizations and dashboards that support senior leadersh across Fixed Income departments, playing a key role in our global technology transformation

Senior AI Consultant

USA

DataArt AI Lab seeks a Senior AI Consultant to collaborate with leadership and cross-functional teams in designing and implementing AI solutio boost productivity, streamline workflows, and drive innovation across diverse industries

Senior Java Developer

INDIA

We're seeking an experienced Senior Java Developer to build scalable, real-time microservices using the latest Java technologies in a collabor cutting-edge environment

Senior Business Analyst with NetSuite Experience

ARMENIA, BRAZIL, BULGARIA, COLOMBIA, CYPRUS, GEORGIA, INDIA, LATVIA, MEXICO, POLAND, REMOTE.LATAM, ROMANIA, SERBIA, URUGUAY

Advanced technological solutions, skilled technical specialists, and the importance of human relations are what make the work on projects of thi company comfortable and interesting

All vacancies →

FAQ: AI-Generated Code, Technical Debt, and the Future of Software Engineering

What causes technical debt in AI-generated code?

How is AI-generated code different from manually written code?

Does AI-assisted programming accelerate technical debt accumulation?

When might AI-driven technical debt reach a breaking point?

Will AI models degrade if they train on AI-generated code?

Will programmers still be needed as AI improves?

What types of developers will remain essential in the future?

How important is vibe coding for non-developers?

What skills should modern software architects develop?

Where should engineers look to stay ahead of AI innovation trends?

DataArt supports tech talent at every stage with continuous learning, challenging projects, and career paths tailored to individual goals. Flexible remote work and a global community create a culture where people thrive — and future-proof their careers.

Vacancies

About Us

What we Do

For Beginners

Internships

For Pros

Work at DataArt

How to join us?

Workplace options

English at DataArt

Centers of Development

Stories

IT Museum →
IT engineering heritage

Proggy-Buggy →

Try me! →

Programming Olympiad

IT Camp →
Conference for IT beginners