Code Debt Detection in Hiring – How SCAV Saves Your Engineering Team 200 Hours/Year

The Meeting That Changed Everything

A VP of Engineering sat through yet another sprint review. The team had delivered 60% of their committed story points. The remaining 40%? Lost to fixing bugs, refactoring messy code, and untangling dependencies that should never have been tangled in the first place.

The same conversation happened every quarter.

"We need to move faster."

"We need to reduce technical debt."

"We need to hire better."

The VP realized something painful. The root cause wasn't the team's ability or work ethic. It was the hiring process itself. They were bringing in developers who wrote code that worked—but created cleanup projects for everyone else.

What Code Debt Actually Means

Technical debt is the hidden cost of taking shortcuts in software development. It shows up in two primary forms:

Code Debt refers to issues like flawed logic, inefficient algorithms, or poorly implemented patterns. Think of it as a broken stair that's easy to spot and repair. These are the coding-level shortcuts that create friction in day-to-day development.

Architectural Debt is more systemic, akin to a poorly planned city road network that causes traffic jams everywhere. These structural flaws ripple through the system, making them harder to pinpoint and fix.

The impact is staggering. Technical debt could account for **20–40% of an organization's technology assets**, and Gartner predicts that by 2026, **80% of this debt will stem from architectural issues** [1].

To make matters worse, hidden architectural issues can consume as much as **40% of digital transformation budgets**. During cloud migrations, approximately **30% of applications often remain unmoved** due to dependencies on outdated middleware, proprietary protocols, or undocumented components [1].

The Hidden Costs Your Team Pays Every Day

Every "good enough" commit from a junior developer adds to the technical debt pile. The evidence is clear: a **10% increase in technical debt can reduce gross profitability by 16%** [1].

Research published in *IEEE Transactions on Software Engineering* found that identified technical debt accounts for about **half of a project's total maintenance costs**, with the cost proportion of the most expensive Top-5 debt items being **1.8x more than the file proportion they contain** [2].

The study also found that **almost all covered maintenance costs persist as technical debt in the future, with an average increase of 6.8% between the last two releases.**

This isn't a problem that solves itself. It compounds.

The Cost Breakdown You Can't Ignore

Let's put this in numbers that matter to your business:

Cost Category	Annual Impact	How It Adds Up
Direct rework	13.5 hours/developer/week	$85B industry-wide
Delayed features	40% of transformation budgets lost	Hidden architecture issues
Developer burnout	Higher turnover + recruitment costs	The human toll
Team productivity drag	20% capacity lost to unplanned work	Innovation tax

According to 2025 data, **20% of engineering team capacity is consumed by unplanned work**, with vulnerability remediation being the number one cause [3]. That means one-fifth of your most valuable talent is perpetually distracted, pulled away from building new features to perform reactive labour.

The financial impact is measurable. **A company carrying a 40% technical debt burden could lose up to $600,000 annually**—an amount equivalent to the salaries of four senior engineers [1].

The ROI of Detecting Code Debt in Hiring

Now let's get to the ROI you've been waiting for.

The average organization wastes **23–42% of their development time due to technical debt**. For a team of 100 developers, that means you're effectively paying for 100 developers but getting the output equivalent of just 75. Twenty-five percent of your capacity is wasted. That's the equivalent of **25 fewer productive engineers** without any reduction in payroll [6].

When you hire a developer who creates code debt, you're not just hiring a person. You're hiring a future cleanup project that will consume thousands of engineering

hours. The average developer spends **13.5 hours every week** on technical debt. That's over **700 hours a year** spent on fixing past mistakes [7].

The ROI Calculation

Let's do the math for a typical mid-sized engineering team:

Metric	Value	Source
Developers on team	50	Assumption
Time wasted on tech debt per developer/week	13.5 hours	[7]
Total weekly hours lost	675 hours	Calculation
Annual hours lost	35,100 hours	Calculation
Cost per hour (avg. developer + overhead)	$100	Industry estimate
Annual cost of tech debt	**$3.51 million**	Calculation

Now, what if you could prevent just **20%** of that technical debt by screening for debt-prone developers at the hiring stage?

Prevention Scenario	Hours Saved/Year	Cost Savings
10% prevention	3,510 hours	$351,000
20% prevention	7,020 hours	$702,000
30% prevention	10,530 hours	$1,053,000

Even a modest 20% reduction in technical debt through better hiring saves over 7,000 engineering hours annually—the equivalent of freeing up 3-4 full-time engineers without adding headcount [8].

What SCAV Evaluates That Legacy Tests Miss

Code Quality Implementation – Scalability, maintainability, and adherence to coding standards. Does the candidate write code that will remain clean as the system grows?

Code Evaluation and Improvement Reasoning – Can the candidate critique existing code and suggest meaningful improvements? This is the skill that prevents future debt.

System Thinking – Do they understand how their code fits into the larger architecture? Candidates who think in systems avoid architectural debt.

The SCAV Code Debt Assessment

SCAV's comprehensive coding proficiency component evaluates three critical dimensions of code quality:

SCAV Assessment Component	What It Measures	Why It Matters for Code Debt
Code Accuracy	Does the code work correctly?	Baseline quality that prevents immediate failures
Code Quality	Is the code maintainable, readable, and scalable?	Determines how much future rework will be needed
Code Debt	What shortcuts and flaws are embedded?	Identifies the hidden cost that will compound over time

The industry is moving in this direction. Codility recently partnered with CodeScene to integrate **CodeHealth™ metrics** into their assessments, recognizing that "the future of engineering evaluation isn't just about technical skills. It's about building teams that create sustainable, scalable solutions" [4][5].

The Research: Why Junior Developers Create More Debt

A study from Chalmers University found that **junior developers are more likely to introduce unintentional technical debt** due to their lack of experience. They "tend to focus on the here and now, and solve today's requirement" without considering how the code will evolve [9].

The researchers identified a specific pattern: **"Reckless-Inadvertent debt"** —messy code produced without mentorship, compounded by insufficient code reviews. The code works. But it's a ticking time bomb.

This is exactly why SCAV matters. It doesn't just assess whether a candidate can solve a problem. It assesses whether they can solve it in a way that reduces future work, not creates it.

The Fix: Detect Debt at the Source

You can't fix what you don't detect. And you can't detect what you don't measure.

SCAV's code debt detection gives you the visibility you need:

Capability	What It Identifies

Code Quality parameters	Maintainability issues before they become technical debt
System thinking evaluation	Architectural flaws before they compound
AI-Augmented assessment	Whether candidates treat AI as a crutch or a tool

By detecting debt-prone developers at the hiring stage, you prevent thousands of hours of future cleanup work. This is why organizations are moving beyond legacy assessments to frameworks like SCAV.

What This Means for Your Team

Every hour your engineers spend fixing avoidable debt is an hour they're not spending on new features, innovation, or growth. The costs are real:

- **The innovation tax**: 20% of capacity lost to unplanned work [3]
- **The security risk**: Debt correlates with vulnerabilities [8]
- **The team morale hit**: Developers burn out from constant firefighting
- **The competitive disadvantage**: Slower shipping means losing to faster competitors

Technical Debt Starts at Hiring

Technical debt is inevitable. But how much of it you accumulate depends on who you hire.

The next time you evaluate a developer, ask yourself: Does this candidate write code that will still be clean in 6 months? A year? Or are they creating a cleanup project that will consume thousands of engineering hours?

SCAV gives you the answer to that question.

Because what you can't assess, you can't prevent.

Ready to reduce your technical debt at the source?

The next article shows how SCAV accelerates your entire hiring process—from shortlist to offer in a few days.

References

1. https://recruiter.daily.dev/resources/technical-debt-considerations-hiring-software-engineers/#faqs
2. https://ieeexplore.ieee.org/document/10934744
3. https://devops.com/the-silent-technical-debt-why-manual-remediation-is-costing-you-more-than-you-think/
4. https://codescene.com/blog/codility-and-codescene-announce-strategic-partnership
5. https://www.codility.com/blog/beyond-technical-skills-code-health-assessment/?ft_utm_source=codescene&lt_utm_source=codescene&ft_utm_medium=partnerreferral&lt_utm_medium=partnerreferral&ft_utm_campaign=2025_CodeScene_Code_Health&lt_utm_campaign=2025_CodeScene_Code_Health
6. https://codescene.com/hubfs/calculate-business-costs-of-technical-debt.pdf#1#1
7. https://www.securecodewarrior.com/article/the-key-to-accelerating-productivity-and-cutting-costs
8. https://www.sonarsource.com/ko/blog/why-technical-debt-is-still-your-teams-biggest-productivity-drain/
9. https://business.udemy.com/blog/tech-debt-becomes-enterprise-skills-problem/?utm_source=direct&utm_medium=direct