Qt Group

The State of Software Quality in Safety-Critical Industries

Why Software Teams in Safety-Critical Industries Struggle to Prevent Technical Debt

2026

Qt Group

The High Cost of Technical Debt

Between June and October 2025, we partnered with Gatepoint Research to survey software engineering and product leaders across several safety-critical industries. Our goal was to better understand their strategies for mitigating rising software maintenance costs and preventing technical debt.

The results are based on a survey of 100 leaders across four safety-critical sectors: automotive, medtech, industrial automation, and aerospace.

Respondent Profile

54% Senior or Department Managers

27% VPs and Directors

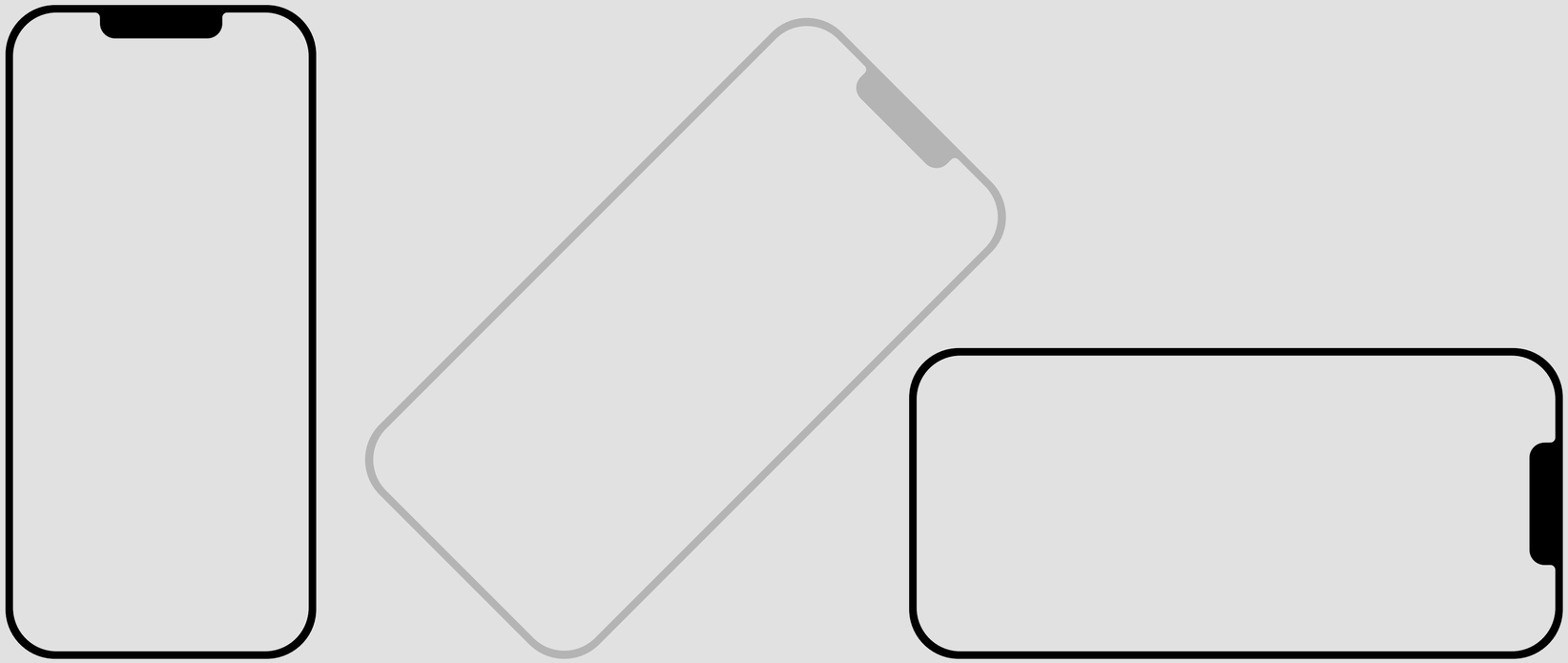
19% Software Engineers

These results offer insights and a benchmark for the shift in how high-stakes industries are prioritizing software quality to prevent technical debt from stalling future growth.

Survey conducted by:	Survey sponsored by:
GATEPOINT RESEARCH	Qt Group

Qt Group

Reading on a mobile device?
Turn it sideways for the best reading experience.



Qt Group

Contents

1

Your Software Architecture Is Drifting Faster Than You Think

Your Software Architecture Is Drifting Faster Than You Think

The gap between your design documents and your actual code widens with every sprint.

In our survey of 100 engineering and product leaders in safety-critical industries, only **5%** said their teams review architecture daily. Almost half wait until release phases to examine structural issues.

We surveyed leaders across automotive, medical devices, industrial automation, and aerospace and defense—industries where defects mean recalls, regulatory failures, or worse.

The pattern was clear: Teams building life-critical systems are struggling to match the pace that modern development demands . Many are still relying on processes built for a different era while being pushed to deliver at today's speed.

THE CONSEQUENCES?

Slower delivery and a growing disconnect between what organizations claim to prioritize and what their day-to-day practices support.

Three Main Contradictions Driving the Problem

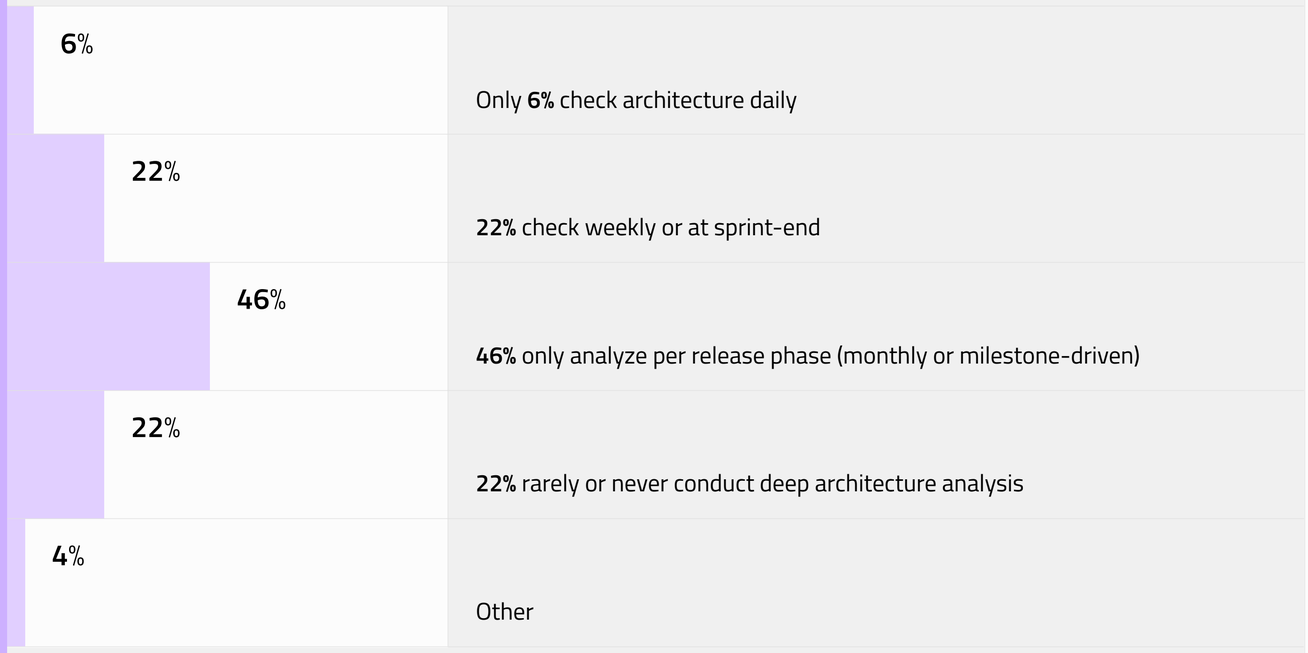
How did we get here? How did teams responsible for building systems where defects mean regulatory failures end up caught between impossible demands for speed and growing technical debt?

Our survey reveals three core contradictions that contribute to the increasing pressures in safety-critical software development.

Contradiction #1

Teams want speed. But most check architecture infrequently.

Half of the respondents (**50%**) report that the pressure to speed up time-to-market while maintaining safe and reliable software has reached its highest level in the past year. Yet when we asked about practices:



The industry wants Formula 1 performance from commuter-car processes. Half of the teams say the pressure for speed-with-quality has intensified most. Yet three-quarters are still only checking their architecture monthly, or even less frequently.

By the time architectural problems surface at release milestones, the damage is already baked into the codebase, the code has diverged from design, and dependencies have tangled.

What should take hours now takes weeks to unravel and fix.

Contradiction #2

Quality is "top priority." But tools and practices don't support it.

29% rank "balancing speed and quality" as their #1 priority in their software quality approach this year, followed by fostering a quality-first culture within teams. Yet:

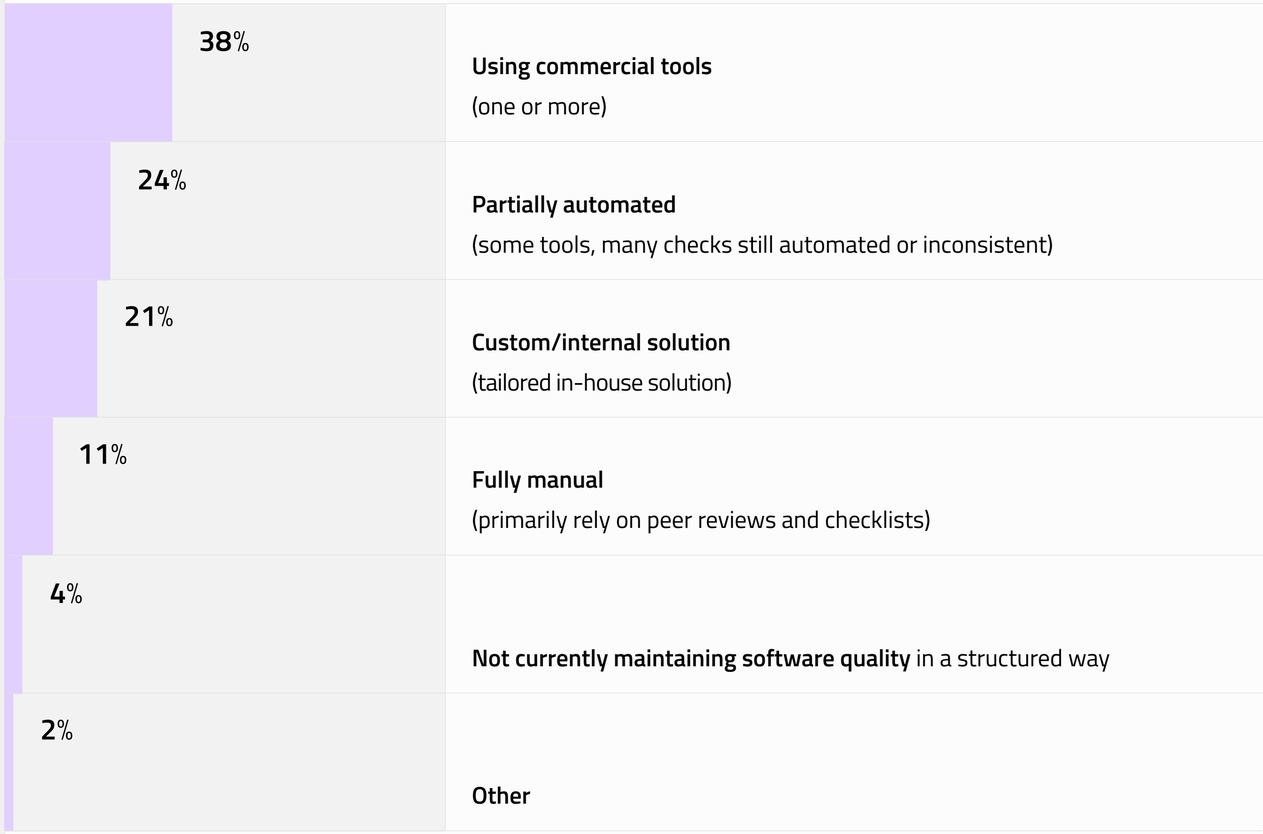
- **24%** say many checks are manual or inconsistent (partially automated)
- **11%** rely entirely on manual reviews and homegrown scripts
- **4%** don't maintain software quality in a structured way

Quality is a priority in principle — not in practice.

Organizations say quality matters, then rely on manual reviews and inconsistent processes that break under pressure.

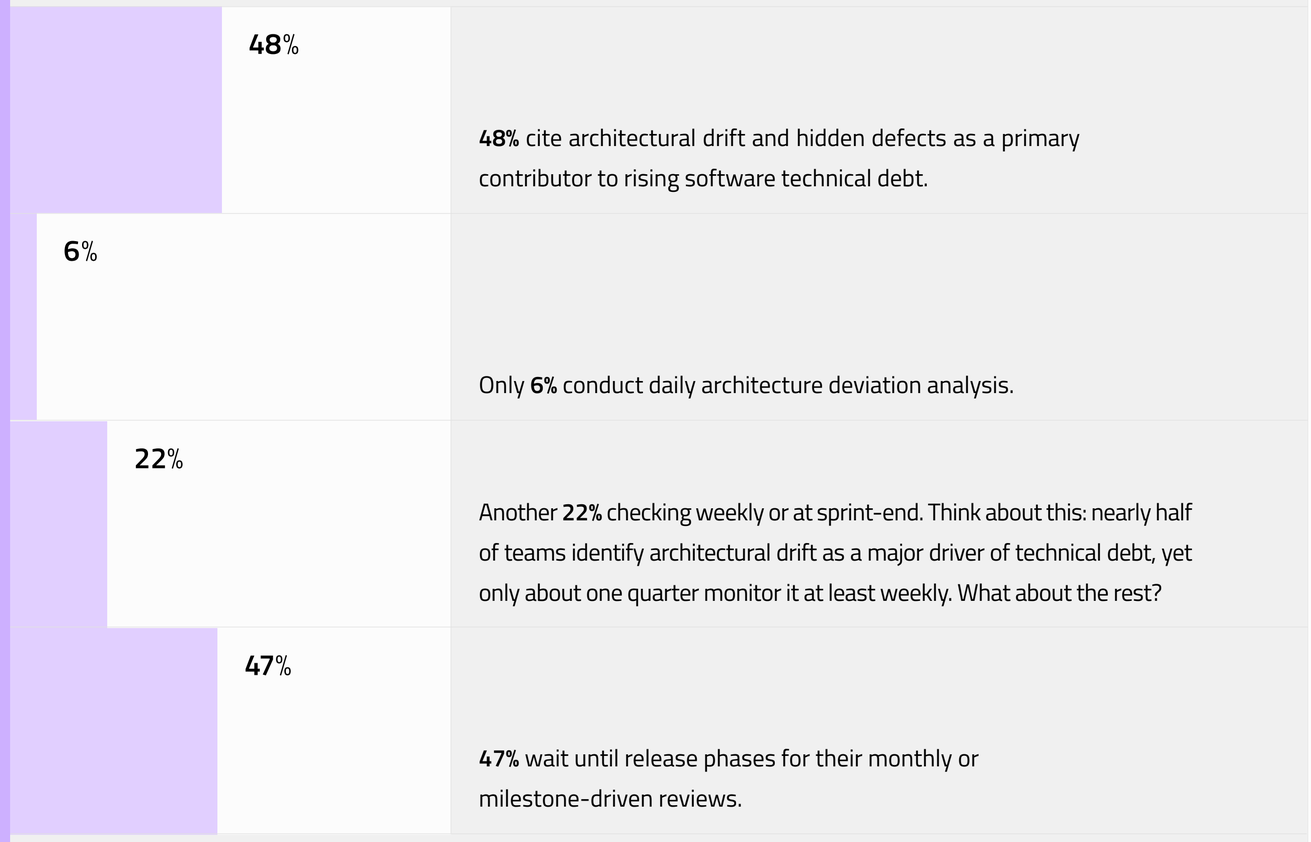
While **38%** use commercial tools, another **24%** supplement with manual processes and **11%** remain fully manual. This reveals significant gaps in achieving comprehensive automation for many organizations in safety-critical industries.

CURRENT APPROACH TO MAINTAINING AND VERIFYING SOFTWARE QUALITY



Contradiction #3

Architectural drift drives tech debt. But teams don't monitor it continuously.



48% cite architectural drift and hidden defects as a primary contributor to rising software technical debt.

Only **6%** conduct daily architecture deviation analysis.

Another **22%** checking weekly or at sprint-end. Think about this: nearly half of teams identify architectural drift as a major driver of technical debt, yet only about one quarter monitor it at least weekly. What about the rest?

47% wait until release phases for their monthly or milestone-driven reviews.

By the time these teams discover architectural problems, dozens or hundreds of commits have already introduced violations. Each shortcut, rushed deadline, and "we'll fix it later" decision has compounded.

Architecture doesn't drift all at once. It degrades gradually, silently, across weeks or months of unchecked development. By the time it shows up in release testing, the issue isn’t a small fix anymore, it’s a structural problem.

2

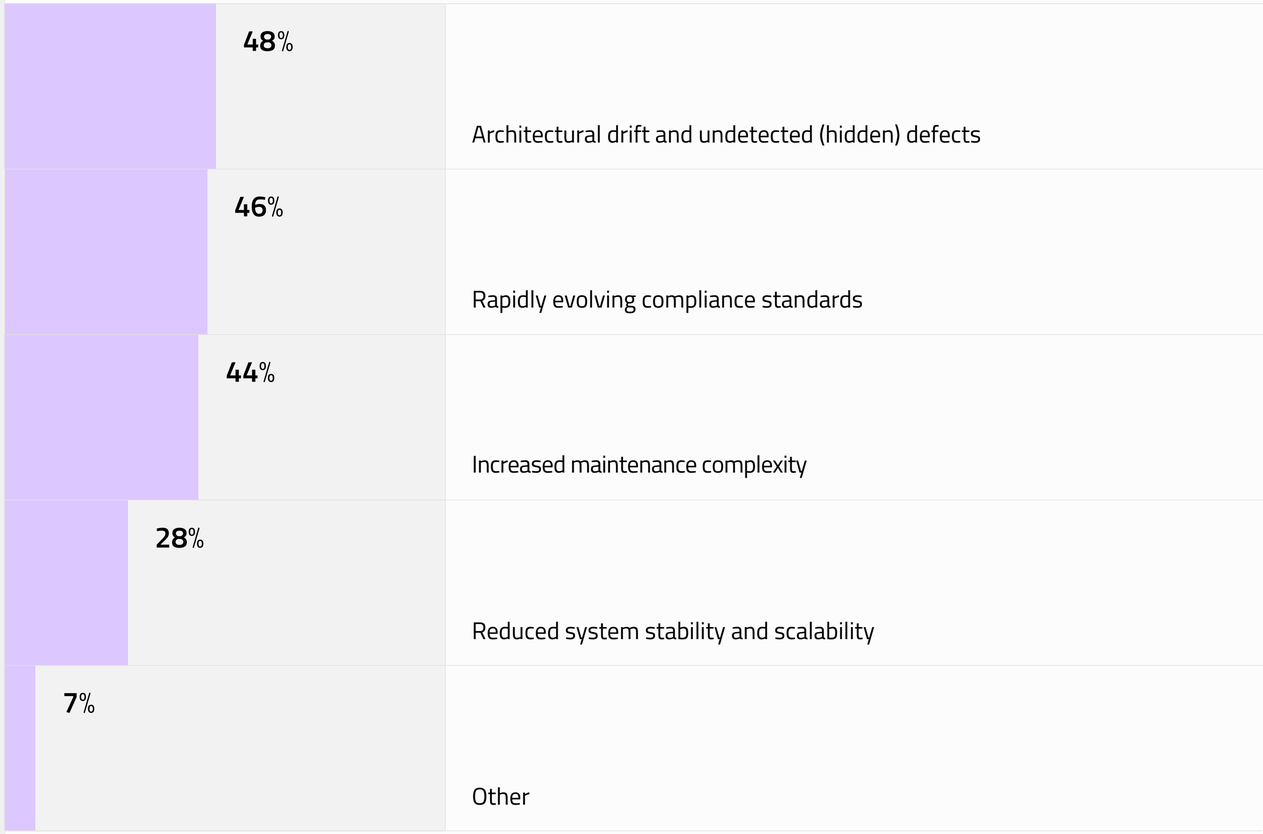
The Real Cost Is When Your Team Is Stuck Firefighting Instead of Innovating

The Real Cost Is When Your Team Is Stuck Firefighting Instead of Innovating

The pressure for speed with quality has become the most intense challenge.
But multiple factors contribute to mounting technical debt.

Survey respondents identified several primary contributors to rising technical debt (respondents could select multiple factors):

PRIMARY CONTRIBUTORS TO RISING SOFTWARE TECHNICAL DEBT



The result?

Engineers spend their days firefighting preventable problems instead of building new capabilities. And the consequences interlock in damaging ways.

When architectural problems surface at release, teams can't simply ship. They must stop and untangle dependencies that have devolved into spaghetti code, and rebuild the structure. Each violation that reaches production multiplies the eventual correction burden, driving rework costs significantly higher than early detection would have cost.

Meanwhile, as architecture drifts from documented design, teams accumulate regulatory risk and potential audit failures alongside their technical debt.

The ultimate consequence of this is system fragility, where changes create unexpected ripple effects, and hotfixes solve one problem only to spawn new ones.

START BY ASKING YOURSELF THREE QUESTIONS

1	**How often** do you check for architectural drift?			
	Daily (6% of teams)	**Weekly/sprint-end** (22%)	**Per release phase** (46%)	**Rarely/never** (22%)
2	What percentage of your quality checks are **manual**?			
	Fully automated (38%)	**Partially automated** (24%)	**Fully manual** (11%)	
3	**Can you visualize architectural drift in real-time**, or only discover it during reviews?			

Your answers reveal whether you're **ahead of the curve** or **caught in the speed trap**.

3

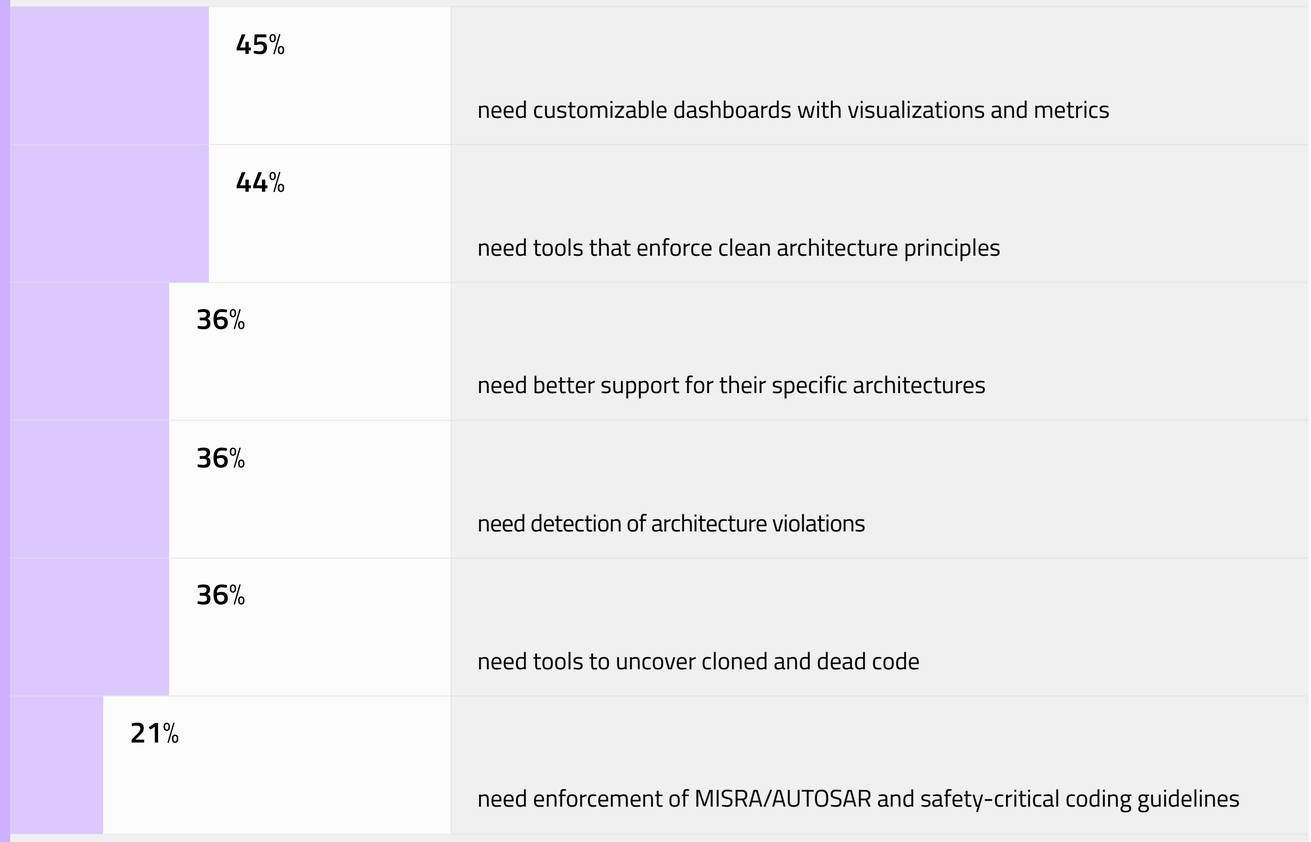
Your Tools Check Code Quality, But Do They Prevent Architectural Drift?

Your Tools Check Code Quality, But Do They Prevent Architectural Drift?

If teams have quality tools, why aren't they working?

Teams aren't suffering from lack of tools. They're suffering from tools that don't surface the structural problems that slow them down.

When we asked what capabilities are missing from static analysis and quality tools today, respondents (respondents could select multiple factors) said:

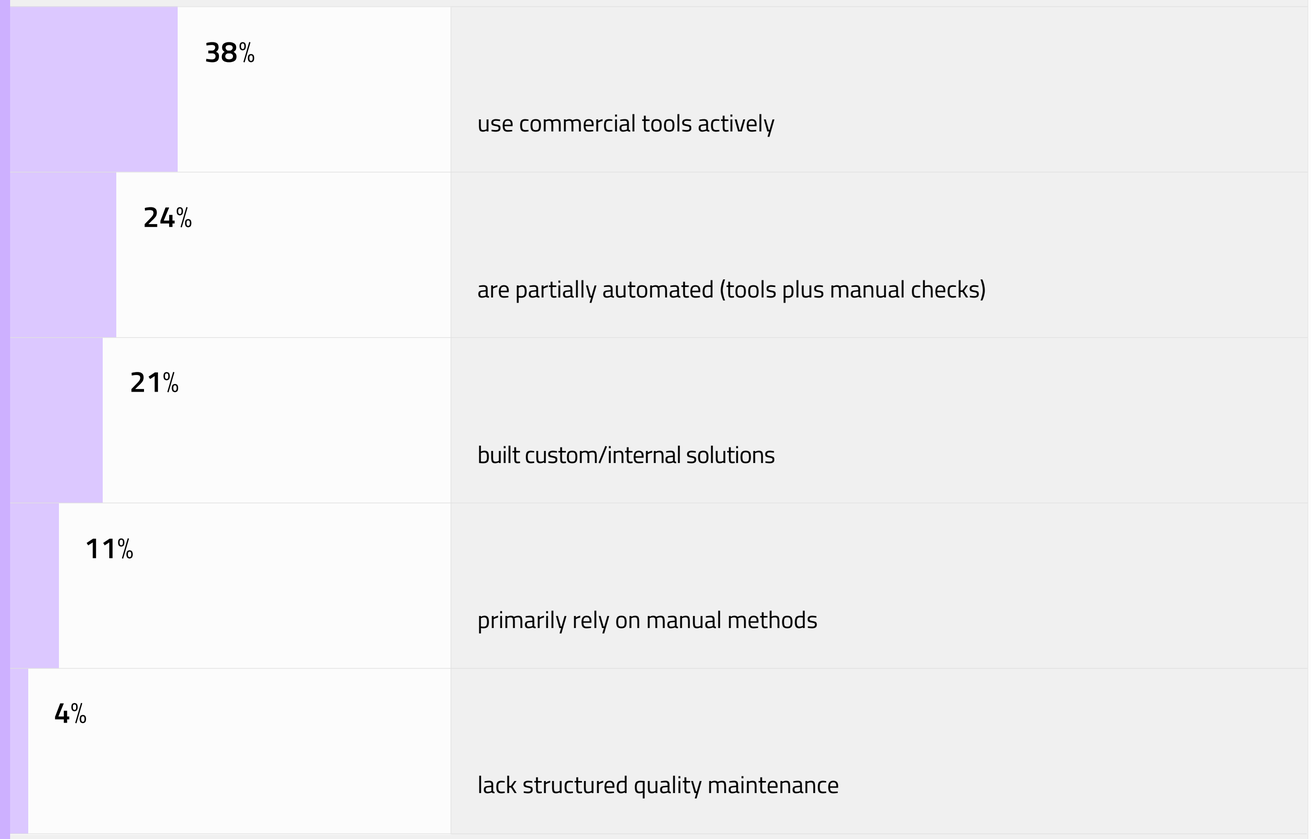


Notice the pattern?

The top gaps are all about visibility and control. Teams have tools that check code quality, but they don't provide :

- Clear visibility into what's happening with architecture.
- Proactive enforcement of architectural principles.
- Flexibility to handle specific architectures and use cases.
- Actionable insights that prevent problems, not just detect them.

The data reveals an automation gap that forces teams to compromise on manual solutions that don’t scale. Current quality maintenance approaches break down as follows:



The fact that **21%** built custom internal solutions is telling. When commercial tools don't meet needs, teams invest in significant effort building their own and often end up creating new maintenance burdens in the process.

When we asked about selection criteria for new tools, the message couldn’t be clearer. Teams don't want more features. They want features that actually work for their specific situations.

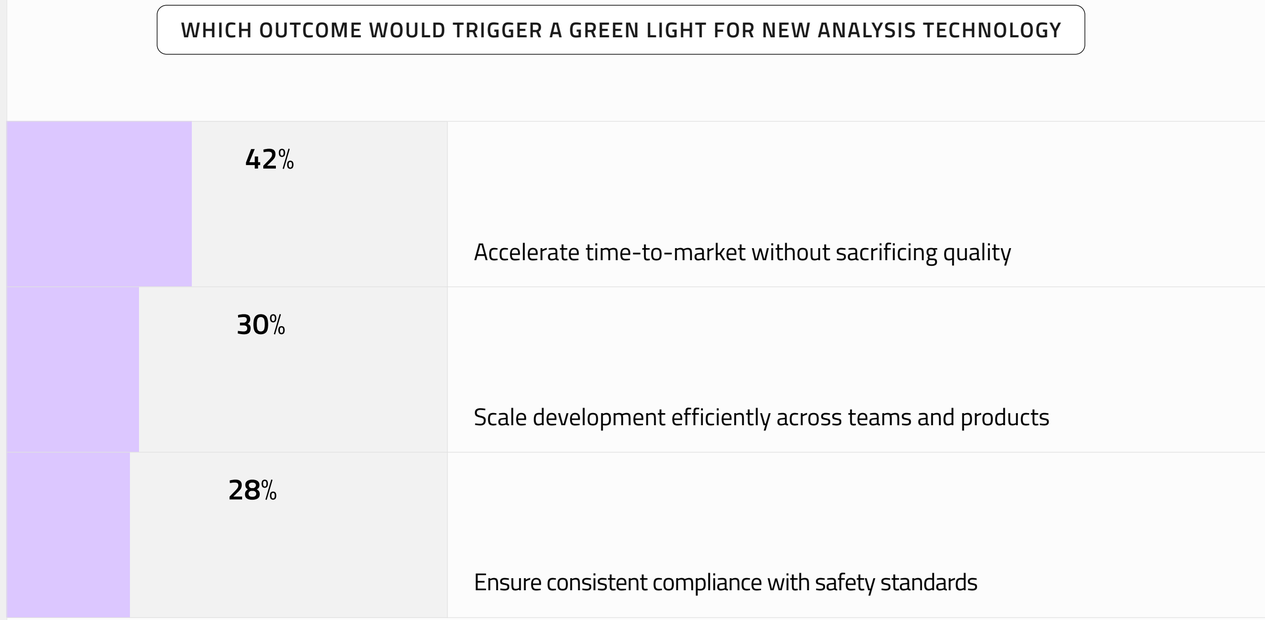
4

What Would Make Teams Finally Change?

What Would Make Teams Finally Change?

Organizations know they have problems. But knowing isn't enough. Action requires a trigger, and the survey reveals exactly what that trigger is.

Teams will tolerate technical debt as long as they can maintain velocity. But the moment debt starts slowing them down, urgency kicks in. We asked "which outcome would trigger a green light for new analysis technology".



The top answer wasn't "reduce technical debt" or "improve compliance." It was achieving what currently seems impossible: going faster without breaking things.

Nearly half of teams identify architectural drift as a major problem, yet only about one quarter monitor it at least weekly, and fewer still have automated enforcement.

That's the crisis—teams lack the visibility and control to address them before they become critical.

The pattern of delayed action

THE SURVEY FINDINGS REVEAL WHEN ORGANIZATIONS TYPICALLY REACH THAT BREAKING POINT:

Velocity drops

The **50%** who say speed-with-quality pressure has intensified most are experiencing this now — features that once took days now take weeks.

Complexity increases

The **44%** struggling with maintenance complexity find that simple changes require understanding increasingly tangled dependencies.

Compliance becomes urgent

The **46%** citing rapidly evolving standards face the reality that their architecture has drifted from documented design just as audits approach.

Stability degrades

Stability degrades. The **28%** experiencing reduced system stability watch as changes create unexpected ripple effects and hotfixes spawn new problems.

→

By the time these triggers intensify enough to force action, teams are deep in crisis mode.

The architectural drift that accumulates gradually over months requires significant effort to correct — if (a big IF) it can be corrected without fundamental refactoring.

5

Teams Need Architectural Control That Matches the Pace They're Being Asked to Maintain

Teams Need Architectural Control That Matches the Pace They're Being Asked to Maintain

Organizations succeeding in safety-critical software aren't running more manual reviews or buying more tools. They've made four fundamental shifts which are presented on the following pages.

SHIFT 1 | **BUILT CONTINUOUS ARCHITECTURAL VISIBILITY**

Instead of checking architecture at release milestones, leading teams monitor it continuously. Drift becomes visible immediately, not weeks later.

For example, a commit triggers an automated architecture check within seconds. The dashboard flags deviations before code review begins. Teams see trends over time rather than point-in-time snapshots. Discussion happens during development, not at release.

This doesn't mean daily manual reviews. It means automated analysis integrated into existing workflows.

SHIFT 2 | **AUTOMATED ENFORCEMENT, NOT JUST DETECTION**

Detection tells you what broke. Enforcement prevents it from breaking in the first place.

The **44%** who identified the need for clean architecture enforcement understand this crucial distinction. Modern approaches flag violations AND block commits that introduce architectural drift, with clear explanations of why the violation matters and how to fix it.

The difference is proactive prevention versus reactive problem-solving.

SHIFT 3 | ADAPTED TOOLS TO THEIR CONTEXT

Generic tools treat all code the same. But safety-critical systems have specific patterns, risks, and compliance needs.

The **36%** asking for better support of specific architectures, combined with the **36%** needing architecture violation detection, recognize that one size doesn't fit all.

Effective tools let teams define their own architectural rules, compliance requirements, and quality gates, then enforce them automatically based on their unique context.

SHIFT 4 | CONNECTED QUALITY TO BUSINESS OUTCOMES

Teams who successfully invest in quality tooling don't talk about "technical excellence" in isolation. They connect architectural control directly to business outcomes (% investment trigger):

This framing transforms quality from a cost center to a velocity enabler.

CONCLUSION

While our survey reveals the gaps, solutions that address these specific needs are already being implemented by forward-thinking organizations, i.e. the **6%** checking daily and the **22%** checking weekly.

They didn't slow down to achieve that visibility. They invested in automation that provides architectural control at the pace modern development demands.

These four shifts aren’t theoretical luxuries. They’re practical necessities to maintain software quality, and they require you to make a choice.

6

Choosing Your Path: Architectural Decisions That Shape What Comes Next

Choosing Your Path: Architectural Decisions That Shape What Comes Next

Most teams default to one of two paths, and they both lead to the same place.

A third path exists, but it requires a different kind of investment .

Path 1
Continue current practices

Check architecture per release phase or less frequently. Accept that drift will accumulate between checks. React when problems become critical.

This is the path of **46%** who only analyze at release phases and **22%** who rarely check. And it means slower velocity over time as technical debt compounds.

Path 2
Add manual processes

Increase review frequency, add more checks, and build custom scripts.

The **21%** who built custom solutions and **11%** relying on manual methods have learned what this path delivers: bottlenecks that don't scale as teams grow. You trade one problem for another.

Path 3
Invest in modern architectural control

Automate monitoring, enforce principles proactively, and integrate quality into workflow.

This is the path of the **6%** checking daily and **22%** checking weekly. They invest in automation that provides visibility without bottlenecks.

The teams choosing Path 3 are accelerating without accumulating debt. The ones on Paths 1 or 2 will face the same challenges next year or find themselves deeper in crisis.

IF YOU'RE EXPERIENCING THESE SYMPTOMS

- Architecture drift discovered late in development cycles.
- Manual quality processes that can't keep pace with delivery demands.
- Tools that don't support your specific architectural patterns.
- Growing technical debt despite quality efforts.
- Difficulty scaling development across teams.
- The intensifying pressure to go faster while maintaining safety.

You're not alone. You're in the majority.

But that doesn't mean you have to stay there.

Your Roadmap to Path 3

Step 1
Start with an honest assessment

Use the diagnostic questions earlier in this report. Where do you actually fall in the data?

If you're checking architecture less frequently than weekly, you're in the **72%** at risk of accumulating the drift and hidden defects that **48%** identify as primary debt drivers

Step 2
Build your case with business outcomes

The strongest investment trigger is accelerating time-to-market without sacrificing quality (**42%**). Don't pitch "better architecture."

Instead, pitch sustainable velocity. Show how continuous enforcement prevents the six-week delays and last-minute rebuilds that kill launches.

Step 3
Evaluate tools for what's missing

Look for capabilities the survey revealed as critical gaps: customizable dashboards (**45%** need this), clean architecture enforcement (**44%**), and support for your specific architectural patterns (**36%**). Generic code quality isn't enough for safety-critical systems.

Step 4
Act before the problem forces your hand

The **50%** experiencing intensified speed-quality pressure and **44%** struggling with maintenance complexity are already in the danger zone. It’s a choice between reactive intervention and proactive prevention.

What sets the 28% apart

Earlier, we established two truths: You can't go faster by looking less frequently. And you can't go faster by looking manually.

What the data shows instead is that 28% of teams have found a more sustainable way forward — one that aligns architectural oversight with the pace of modern software development.

These teams aren't moving faster because they work harder or take on more risks. They've made a different kind of investment: automated, continuous architectural control tailored to their environment.

That shift gives them consistent visibility without slowing teams down.

The 6% checking daily and 22% checking weekly aren't exceptional outliers. They've simply removed friction.

By automating insight and enforcement, they've reduced bottlenecks, caught drifts earlier, and kept complexity from compounding. As a result, they're the group seeing acceleration rather than slowing down.

Teams checking less frequently aren't doing anything wrong. Many are operating with tools that weren't designed for today's pace or architectural complexity. But gaps in visibility allow drift and hidden defects to accumulate.

The 48% who cite these as primary debt drivers are experiencing the result: certification delays, rework, and reactive cycles that are hard to escape.

Ultimately, this isn't a question of effort or intent.

It's a question of when and how architectural visibility is built into the workflow.

Some teams address architectural visibility proactively, while velocity is high and complexity is manageable. Others are forced to address it later, when delays and risk make the decision unavoidable.

The difference isn't capability, it's timing.

Qt Group

Winning Teams Build This Infrastructure. You Can Too.

Axivion Architecture Verification is how teams close the gap between intention and practice. Customers report 50% less manual rework time and cut audit preparation time by 70-80%.

We help leading organizations in turning compliance from a bottleneck to competitive advantage.

Explore Axivion Architecture Verification

qt.io/quality-assurance/axivion-architecture-verification

Qt Group

About Qt Group

Qt Group (Nasdaq Helsinki: QTCOM) is a global software company, trusted by industry leaders and over **1.5 million** developers worldwide to create applications and smart devices that users love.

We help our customers to increase productivity through the entire product development lifecycle: from UI design and software development to quality management and deployment. Our customers are in more than **70 different industries in over 180 countries.** Qt Group employs some **900 people,** and its net sales in 2024 were **209.1 MEUR.**

To learn more, visit **www.qt.io**