The Hidden Cost of DIY DevOps for Embedded Product Teams

Quantifying the Productivity Drain in Embedded Build, Configuration Management, Hardware Test Automation, and Release Tracking

Author: John N. Macdonald, 4TLAS, Inc

Date: June 2025

Executive Summary

Embedded-product engineering teams are quietly wasting 30 – 50 % of their annual capacity on the plumbing work required to build, fix, and maintain custom DevOps solutions for embedded development. For a 20-engineer embedded team with a loaded cost of $160 k per engineer, overlapping inefficiencies drain roughly $3.9 million of engineering budget every year. Even after adjusting for overlap, the net capacity lost totals up to $1.6 million—effectively the output of one to two full-feature teams. Deploying a purpose-built DevOps toolset that fills the gaps can realistically cut that waste in half, handing back close to $1 million of productive engineering time each year.

Table of Contents

1 Introduction

Over the last decade, DevOps turned web and mobile release cycles into a push-button sport—code merges, test farms that magically elasticize, and new features landing in production before lunch. Continuous Integration (CI) has transformed the software development time-to-market cycle.

Luckily, and finally, DevOps concepts and practices are making their way into embedded development, but they stall the instant the pipeline has to build for multiple targets, program a microcontroller, drive a hardware-in-loop test fixture, or serve up a re-creatable certification build from six months ago.

Generic CI platforms and the potpourri of DevOps tools that have spawned around them were never built for cross-compilers, flash programming devices, controlling oscilloscopes, or the exhaustive traceability demanded by safety-critical industries. So embedded engineers do what they've always done: they DIY it. They glue together shell scripts, Python, YAML, manifest files, spreadsheets, and a heroic dose of process to cover the gaps. It works—until it doesn't.

This whitepaper measures the hidden bill for that DIY culture and shows how there is a market gap for purpose-built, standardized tools that can free embedded teams to focus on the product, not the plumbing.

2 Embedded Specific Challenges

DevOps has proven to be the answer in the web and mobile industries, and these concepts can be the answer for embedded, but the embedded world collides with five hard realities that don't exist for web and mobile:

- **Discontinuous Delivery**: The firmware may be a line item on the Bill of Materials (BoM) like a resistor. Even in IoT, over-the-air updates are pulls, not pushes. The firmware that makes it out the door has to work. You might not get a second shot at it.
- **Hardware-in-the-Loop:** Hardware is finite, scarce, expensive, sometimes late, and messy, yet required for verification and validation.
- **Cross-Compile Toolchains:** The build environment drifts between laptops and runners, toolchain versions, and build script mayhem.
- **Complex Version Matrices:** A 3D matrix that maps software component versions and hardware versions over time. Your workflow demands flawless traceability to know who has what and who gets what.
- **Compliance and Traceability:** In safety-critical and other regulated industries, you must create and then deliver a mountain of documentation and other test-related collateral.

These specific challenges lead to embedded teams filling the gaps in the current DevOps platforms through customized DIY solutions without much thought. It's part of the job. The embedded development culture in engineering has always been "can-do," scratch, claw, and DIY. We'll make it happen.

However, the growth in complexity of embedded systems and the number of systems in the world is demanding a market solution. The good news is that you can feel the change in the air. Walk around the Embedded World Conference and you'll see the terminology posted everywhere (i.e., CI/CD, DevOps, automation, etc). Talk to team leadership, and they're now web and mobile workflow-curious. And where once proficiency in C was the only pillar on which an embedded developer needed to stand, developers are now pulling Python, YAML, and even JavaScript out of their toolboxes.

3 Methodology

- Literature scan of global surveys (2020–2025) targeting developer experience, DevOps, and embedded domains.
- Normalization of disparate metrics into hours-per-engineer-per-week and percentage-of-workweek values.
- Cost model based on a $160 k loaded annual salary, 48 working weeks, and a 20-engineer reference team.
- Embedded adjustment: where studies covered general software, we applied conservative uplifts (×1.2) to reflect longer compile cycles, configuration management burdens, and hardware-in-loop constraints typical in embedded environments.

4 Finding the Waste

4.1 Build Standardization and Tool-Chain Churn

- Developers lose 15 h/week to pipeline waits and fixes [1].
- 43 % of C++-heavy teams still call long builds a major pain [7].
- Atlassian's 2024 State of Developer Experience survey reports that 69 % of developers lose eight or more hours every week dealing with broken builds, flaky environments, and other pipeline issues [4].
- From more than a decade of project data, 4TLAS finds that up to 50 % of builds that succeed on a developer's workstation fail when the same code is built by a colleague or executed in the CI pipeline.
- Each build breakage or inconsistency absorbs multiple hours of combined developer and DevOps time to diagnose, reproduce, and roll forward or revert, compounding delivery delays.

Implication: With cross-compiles and RTOS builds pushing jobs past the 10-minute "flow break" mark [2], embedded shops live on the right-hand tail of global workflow durations.

4.2 Configuration-Management Firefighting

- Typical mitigation & rollback work burns 6 h/engineer/week in mixed device fleets (industry average) — roughly 15 % of capacity.
- Building and maintaining the custom glue required to link builds, tests, and release artifacts on generic CI platforms (e.g., GitLab, GitHub) consumes an estimated 20–25 % of total engineering effort, according to Retool's 2020 survey of internal-tool builders [6].
- In 4TLAS client engagements, local demo and integration test builds are frequently unreproducible because configuration management stops at source code. Binary artifacts, toolchains, build environments, compiler switches, and hardware configuration settings drift out of sync, sabotaging repeatability.

Implication: When configuration data lives outside version control, every hand-off risks "works-on-my-machine" failures. Rebuilding previous release or integration builds for bug fixing and analysis steals sprint capacity and puts release timelines at risk.

4.3 Hardware Test Automation Upkeep

- Test-maintenance drag. Mabl's 2024 State of Testing in DevOps survey shows that 21 % of a QA team's testing time is now spent just keeping automated tests

alive, and 34 % of respondents call maintenance their single biggest pain-point—a 138 % jump versus 2022 [12].

- Budget sink. The World Quality Report 2022-23 finds that 30–50 % of the total test-automation budget is consumed by script maintenance—the largest single cost driver once frameworks are in place [13].
- Resource bottleneck. Perforce's 2023 State of Test Automation survey reports that 22 % of teams cite "lack of resources to automate and maintain tests" as their top challenge, outranking all other pain-points [14].
- 4TLAS has found that many organizations automating Hardware-in-the-Loop testing do so with PyTest, but the framework demands a fully custom layer to manage DUT farms, external equipment control, and orchestrate test content. QA engineers rarely have that software-development expertise, so the burden shifts to the product team. In fact, much testing stays in the manual domain due to the burden of custom requirements. Industry surveys show maintenance already consumes 21 % of QA time and is the #1 pain-point [12]. PyTest-based HIL only amplifies that drag and pulls it into the development team.

Implication: DIY HIL stacks divert scarce engineering hours into glue code instead of new tests, delaying defect discovery and stretching program schedules.

4.4 Delivery / Release Tracking Overhead

- McKinsey finds leading software organizations target 70 % inner-loop / 30 % outer-loop time split [5]. Many embedded organizations invert that ratio when release coordination is manual.
- Retool's survey shows bigger firms spend 20–40 % of engineering time building internal tools that often duplicate release-tracking functionality [6].
- Support and QA engineers lose ≈ 2.8 h/week searching for information and ≈ 2.0 h/week recreating existing assets, according to an APQC 2025 knowledge-worker survey [3].
- 4TLAS's field work shows that embedded software deliveries travel to a multitude of endpoints. From PLM systems for manufacturing, to FTP drops for system-integration and certification partners, to ad-hoc ZIP files emailed for pre-sales demos. This heterogeneous mix of delivery paths spawns confusion, duplicated effort, and schedule slip as product and engineering teams chase down the right artifact for each audience.

Implication: Fragmented delivery channels turn release engineering into a scavenger hunt, pushing work into the outer loop and slowing time-to-market.

4.5 Additional Industry Evidence

- IDC analyst survey [8] of over 800 software professionals found that only 16 % of the work-week is spent writing application code; the remaining 84 % lands in CI/CD, deployment, monitoring, and other plumbing tasks—placing the DevOps-related share well inside our 30–50 % waste band [8].
- The 2024 *State of Developer Experience* study by Harness & Wakefield Research reports 60–70 % of developer time consumed after coding—testing, deployment, security, governance, compliance—again corroborating a 30–50 % drag even when accounting for overlap with coding tasks [9].
- McKinsey's deep dive into more than 60 embedded-system programs notes R&D-budget overruns of 30–50 % stemming from unmanaged complexity, driving launch delays and ballooning DevOps overhead [10].
- 52% of CxOs said their teams use 2-5 tools for software development, while 54% of individual contributors report their teams use 6-14 tools, representing another disconnect within organizations [11].

5 Financial Impact Model

This section shows the mathematical models used herein.

5.1 Overlapping Hotspot Estimates

Hotspot	% Week Wasted	Hrs/Eng/Year	$/20-Eng Team
Build-toolchain churn	40 %	768	$1.28 M
Config-mgmt firefighting	15 %	288	$0.48 M
HW test-rig dev & fix	37 %	710	$1.18 M
Release-tracking overhead	30 %	576	$0.96 M
Total (non-additive)	—	—	≈ $3.9 M

Note: Percentages derive from independent studies and overlap; summation illustrates magnitude, not strict additivity.

5.2 Effective Capacity Lost (30-50%)

Applying the aggregate range found in multiple studies gives a clearer picture of the dollars at stake.

Effective Capacity Lost	$/Engineer/Year	$/20-Eng Team/Year
30 %	$48 k	$0.96 M
40 %	$64 k	$1.28 M
50 %	$80 k	$1.60 M

6 Conclusion

DIY DevOps extracts a hidden tax of 30 – 50 % on embedded engineering capacity: $0.96 – 1.6 M per 20-engineer team. Cutting into that by just one-half saves almost 25% for an organization. If you can all-but-eliminate it, you've just put almost 50% back onto the bottom line.

The DevOps tool industry serves the web and mobile markets, but it's not yet complete for the embedded teams. The embedded market is ripe for a solution.

7 References

[1] Mike Vizard, “Survey Shows Mounting DevOps Frustration and Costs,” *DevOps.com*, 14 Apr 2021. https://devops.com/survey-shows-mounting-devops-frustration-and-costs/

[2] CircleCI, *The 2025 State of Software Delivery*, Mar 2025, pp. 2–6. https://circleci.com/landing-pages/assets/2025-state-of-software-delivery-report.pdf

[3] APQC, *Knowledge Worker Productivity Benchmarks Report*, 2025, Public source: https://www.apqc.org/blog/km-makes-knowledge-workers-more-productive-and-less-stressed-out

[4] Atlassian & Wakefield Research, *State of Developer Experience Report 2024*, Feb 2024. https://www.atlassian.com/software/compass/resources/state-of-developer-2024

[5] Chandra Gnanasambandam, “Can software developer productivity really be measured?,” *McKinsey Re:think*, 1 May 2024. https://www.mckinsey.com/~/media/mckinsey/email/rethink/2024/05/2024-05-01d.html

[6] Justin G., “The State of Internal Tools in 2020 – Survey Results,” *Retool Blog*, 5 Mar 2020. https://retool.com/blog/state-of-internal-tools-2020

[7] Christopher McArthur, “Breaking down the 2024 Survey Results,” *Modern C++ DevOps*, 22 Apr 2024, lines 33–37. https://moderncppdevops.com/2024-survey-results/

[8] Paul Krill, “Developers spend most of their time not coding – IDC report,” *InfoWorld*, 24 Feb 2025. https://www.infoworld.com/article/3831759/developers-spend-most-of-their-time-not-coding-idc-report.html

[9] Tim Anderson, “Developers spend too much time ‘not coding’ says Harness CEO,” *DevClass*, 20 May 2024. https://devclass.com/2024/05/20/interview-developers-spend-too-much-time-not-coding-says-harness-ceo/

[10] Johannes Deichmann et al., “Cracking the complexity code in embedded systems development,” McKinsey & Company, 25 Mar 2022. https://www.mckinsey.com/industries/industrials-and-electronics/our-insights/cracking-the-complexity-code-in-embedded-systems-development

[11] GitLab, “Global DevSecOps Survey 2024,” GitLab, May 2024. https://s204.q4cdn.com/984476563/files/doc_news/2024/06/2024secdev.pdf

[12] Mabl, *2024 State of Testing in DevOps*, Apr 2024, p. 7. https://www.mabl.com/state-of-testing-in-devops-2024

[13] Capgemini & OpenText, World Quality Report 2022-23: The Future Up Close, Nov 2023, p. 22. https://www.opentext.com/assets/documents/en-US/pdf/the-future-up-close-world-quality-report-2023-24-en.pdf

[14] Perfecto by Perforce, “The Latest Test Automation Trends: 4 Takeaways From the 2023 State of Test Automation Report,” blog post, 25 Jan 2023. https://www.perfecto.io/blog/test-automation-trends

About 4TLAS

4TLAS (pronounced “atlas”) helps embedded teams automate, integrate, and scale their development, test, compliance, and delivery by infusing modern development discipline into their workflow. Drawing on deep roots in embedded engineering, DevOps, and cloud automation, we offer purpose-built tools and expertise that strip away manual bottlenecks and create scalable, repeatable workflows. Our culture is anchored in integrity, curiosity-driven innovation, resilience, and a relentless focus on customer success—values that shape how we collaborate and the solutions we deliver. The result is a standardized approach that empowers engineering teams to release higher-quality firmware faster, with full traceability and lower cost, so they can spend more time innovating and less time wrestling with infrastructure.

Author: John Macdonald

Title: Co-founder | CEO

Contact: john.macdonald@4tlas.io

4TLAS: https://4tlas.io