DevOps in Embedded Software:

Uncovering Skepticism and Resistance

A Research Paper submitted to the Department of Engineering and Society

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science in Computer Science, School of Engineering

Peter Tessier

Spring 2025

On my honor as a student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Advisor

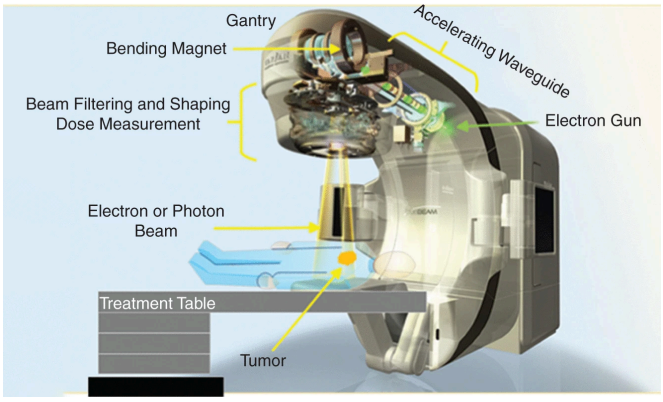
Sean Murray, Professor of STS, Department of Engineering and Society

I. Introduction

It was his ninth time receiving radiation treatment. As soon as the beam started, he felt an electric shock and heard a snap from the machine - this was not normal. As he tried to get up, he immediately felt another pulse from the machine shoot through his arm, a feeling as if his hand had been ripped off. The doctors were unaware that he had just been administered between 16,500 and 25,000 rad in less than a second. One thousand rad is enough to kill. After five excruciating months of hospitalization and most of his body developing paralysis, he ended up dying.

Figure 1

Therac-25: A view of the machine responsible for six radiation overdoses (Silvis-Cividjian, 2024)



This incident on March 21, 1986 was the fourth of six documented radiation overdoses caused by a software bug in the Therac-25 (displayed in Figure 1), now a canonical cautionary tale against the dangers of lackluster software development (Leveson & Turner, 1993). Indeed, across the years, faulty software has been responsible for the loss of lives, fortunes, and sanity, including the loss of a space probe, loss of aircraft, collapse of bridges, and false nuclear alerts (Unwin, C., & Ould, M. A., 1986). As the software industry has matured, a solution to software reliability struggles that also supercharges speed-to-market has been found in *DevOps*, a model

designed to seamlessly integrate the coding and non-coding aspects of producing software. This includes automating tests to gain confidence in future changes, writing tests before writing the actual source code (Test Driven Development, or TDD), and iterating fast with Continuous Integration/Continuous Deployment (CI/CD) (Hall, 2020). While such practices are relatively straightforward to apply to general-purpose software (like web and desktop applications), they face unique challenges when applied to embedded software, which runs on specialized hardware (like medical devices). For instance, it is much easier to make an automated test for an online palindrome detector than a physical carbon monoxide detector.

Though it is well understood how the nature of embedded software poses technical hurdles for adopting DevOps best practices, it is unclear how actors have responded to these challenges. In other words, there is a gap in understanding how embedded developers perceive DevOps compared to general-purpose developers. I have sought to fill that gap by analyzing the discourse from a sample of 12 Reddit posts and 9 statements from company representatives. Specifically, I compared the sentiment towards DevOps of each comment and post (62 in total) between subreddits for embedded developers and general-purpose developers. To complement the aggregated opinion of many developers, I also read and summarized the 9 sources from company representatives (mixed between embedded and general-purpose), identifying their perspective of DevOps.

From my findings, I conclude that the advantages of DevOps are widely recognized by the software industry, even the embedded sector, though embedded engineers tend to agree that their specialization lags behind their general-purpose peers. I argue that embedded teams must work to “practice what they preach” by actually implementing DevOps strategies, even if further research is needed to see exactly what a successful roadmap looks like. I also emphasize the

importance of building a DevOps culture, not simply enforcing the practices as a checkbox, but training employees and demonstrating their effectiveness. By shining the spotlight on DevOps for embedded software, this field can enjoy the same agility and reliability as general-purpose software, which is not only essential in safety-critical applications, but promotes quality of life in any product for both the developer and the client.

II. Problem Definition: Embedded Developers Have Reason to Question DevOps

I present a historical review of how the industry has achieved its modern DevOps standards, then discuss the technical and social challenges embedded developers face in practicing such standards.

From Ad Hoc Operations to Modern DevOps

In the 1970s, Dr. Winston Royce pioneered the "Waterfall Method" as an assembly-line-esque, requirements-driven way of developing software. Frustrated with its rigidity, a group of 17 engineers crafted the "Agile Manifesto," a flashy new method prioritizing frequent releases, dynamic requirements, and increased collaboration between sectors. As the popularity of Software as a Service (SaaS, meaning the product is updated online via subscription as opposed to physically) ballooned, the race for faster and better features and improvements only increased. Even though Agile promoted collaboration among developers, the demand for more frequent releases could not be met as the process was bottlenecked in the release stage, under the responsibility of the "operations" team, which could include anything from testing to deployment itself. Hence, DevOps was born to unite the two teams, making developers responsible for both the programming and release using the power of automation. At

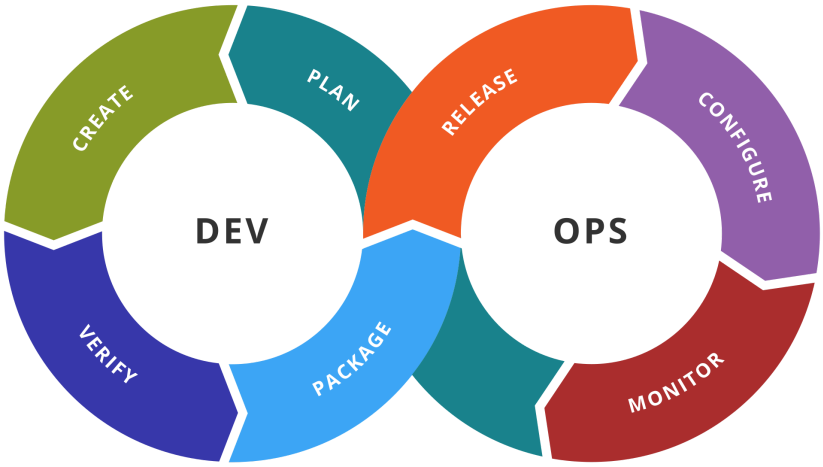
its core, DevOps is a cultural shift that unifies teams to get the software out the door quickly and reliably with the same high quality (Wang et al., 2022).

Now, in the contemporary tech industry, DevOps is the widely recognized standard. Indeed, a 2020 survey reported 99% of companies recognizing DevOps having a positive impact on them. Another study - DORA's State of DevOps 2019 - highlights the method's effect on both speed-to-market and quality, with top performers releasing 208 times more frequently and 106 times faster than low-performers, plus a failure rate 7 times lower (Hall, 2020). In a separate 2016 survey with 442 responses, 94% indicated interest in using CI in their next project, indicating overwhelming popularity (Hilton et. al.).

To understand what proper DevOps looks like, the industry-leading DevOps tool provider Atlassian has outlined several "best practices." First, it recommends following the Agile methodology, focusing on incremental improvements instead of a massive release date. Going hand-in-hand with Agile, it recommends flexing the code's muscles as early as possible using Continuous Integration/Continuous Deployment (CI/CD), where CI involves committing any changes to the "main" (stable, production-ready) code as frequently as possible, and CD ships those changes to real customers as frequently as possible. This means bugs are discovered much sooner, exponentially decreasing time and financial costs of fixing them. A few more suggestions include choosing the right tools, automating as much as possible, and prioritizing monitoring the system. It ends by emphasizing the need for a collaborative and transparent culture, saying that effort will be needed to establish such qualities if a company is siloed, as it claims most organizational structures tend to be. Such a streamlined production workflow enables a tight cycle of development, illustrated in Figure 2.

Figure 2

DevOps Stages: From planning to release, DevOps is all about unifying all aspects of the Software Development Life Cycle (SDLC) ("DevOps Toolchain," 2024)



Embedded DevOps Faces Unique Technical Challenges

Despite its benefits enjoyed by general-purpose software, DevOps remains tricky to implement in embedded spaces due to technical challenges that are widely recognized in the literature. In one 2016 study with four companies, Lwakatare et al. identified four such categories of key challenges: (a) Hardware dependency (as fast as software can be developed, it is bottlenecked by the relatively slow development of hardware), (b) imperfect test environment (in contrast to web applications that can be tested on servers that closely mirror production environments, embedded devices often have to test in their best guess of a production environment), (c) scarcity of tools (unlike in other areas of software like the web, there is a lack of software that enables automation of the release process), and (d) invisibility of usage data (software in places like websites can easily monitor metrics from users allowing continuous feedback, but such post-deployment data is not as accessible for embedded devices).

Wind River Principal Technologist Woolley (2021) agrees that challenges emerge from tight coupling to specialized hardware and peripherals (e.g., sensors) and development and

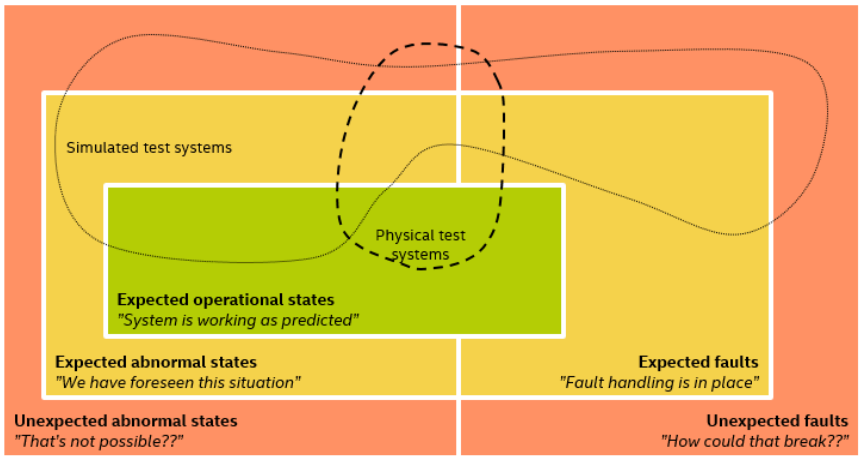
management tools failing to lend themselves to DevOps, also adding the difficulties of lower-level languages and increasing diversity of end hardware. Moreover, Bajer, Szlagor, & Wrzesniak (2015) further describe how limited hardware reduces the number of tests that can be executed in parallel and a greater number of deployment steps - often requiring physical cable connection - hinder release automation.

To mitigate such technical challenges, a few techniques have been introduced and accepted in embedded spaces. First, a Hardware Abstraction Layer (HAL) is a "layer" of software that interacts directly with the hardware in order to expose abstract functionality, which can then be used by the more specific embedded specifications. Then one can reuse the same HAL for a completely different project, improving reliability and minimizing the amount of new code to test (Yoo & Jerraya, 2003).

Another innovation mitigates the problems of hardware dependency and slow, complex deployment on hardware: Hardware in the Loop (HIL, or HWIL), which simulates hardware using software via a mathematical model (Koehler et al., 2008). Such a simulation has many advantages: it is extremely reproducible, can be run in parallel, can be automated much easier, and allows for inputs that real hardware typically doesn't allow, as illustrated in Figure 3 by Intel Director of Simulation Technology Ecosystem Jakob Engblom (2017). That being said, such a solution is imperfect, as simulations only mimic a true production environment, and hence risk inaccuracy, so it always must be balanced by some running on true hardware.

Figure 3

Simulated Versus Physical Test Systems: Many inputs not possible with pure hardware can be tested with simulation (SDLC) (Engblom, 2017)



Embedded DevOps Experiences Social Challenges

Even if the tools make DevOps as accessible as ever, the embedded field can't hope to reap the benefits of this practice if the developer culture resists it. Indeed, a 2007 paper from Atomic Object - a firm that helps software companies level up their practices - recounts how for the first time presenting to a room of embedded developers, they heard the objection to Test Driven Development (TDD) and CI: "Boy, guys, it sounds great, but you can't do this with firmware code because it's so close to the hardware," followed by "folded arms and furrowed brows" (Karlesky et al., 2007). They ended up taking it as a challenge, successfully automating much of the operations process, including providing an extensible testing framework complete with simulation. Their tight, modular testing setup proved especially useful in the embedded sphere because bugs could be rapidly pinpointed in either hardware or software.

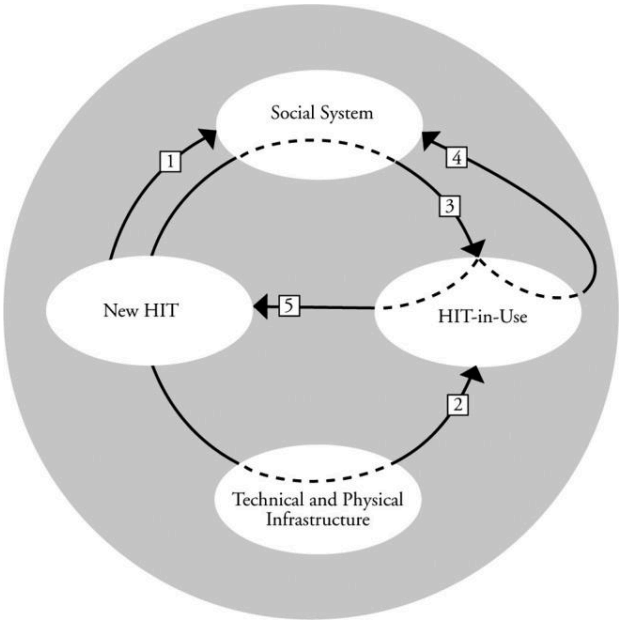
Sixteen years later (2023), a similar success story is recounted by the Master Thesis Student Fabian Segatz, whose degree project focused on providing a CI system to the laser-making startup Cobolt AB. In this case, the developers were much more eager to try out this new practice, and reported with optimism in the post-implementation survey. While the developers admitted to some challenges like initial setup and maintenance effort, the need to wait

for the CI server, and managing build tool dependencies, they were overall happy with the reduction of redundant manual tasks, faster bug feedback, and increased confidence in the main code. Of note, a bug in the system test server caused inaccurate reporting of test failures, which was a source of frustration and led to manually overriding the system. Segatz concluded that quality tools and time were two factors that would increase trust in the CI technology. Both this experience and the one with Atomic Object suggest the possibility of widespread adoption of DevOps practices in embedded, if only the developers accept it and use it properly.

Indeed, STS theory emphasizes the need for users to understand and trust technology for such technology to effectively accomplish its intended role. One framework that articulates this well is the Interactive Sociotechnical Analysis (ISTA) (Harrison et al., 2007). In its original application to Healthcare Information Technologies (HIT), it provided a lens through which to view the recursive interaction between doctors, nurses, etc. and this new technology, illustrated in Figure 4. For instance, new HIT reduced doctors' time interacting with patients, just as doctors influenced the next generation of HIT by demanding important features such as tiered alarms (as opposed to too many "high alert" alarms that doctors would end up ignoring).

Figure 4

Recursive Interaction Among HIT Actors: DevOps technology interacts with users in the same way as HIT does (Harrison et al., 2007)



In the same way that doctors responded with skepticism and frustration at times to this new technology that was supposed to be helpful, reports of aversion to DevOps practices that challenge the status quo have been observed in the tech industry. Such resistance occurred when author and tech CEO of Langr Software Solutions Jeff Langr tried enforcing TDD. Several developers complained and one even ended up leaving, refusing to test. As a consequence of this poor testing environment, a high-security chat application was shipped with an "embarrassing defect" (Tarlinder, 2016).

Left with questions from stories similar to Langr's, Laukkanen, Paasivaara, and Arvonen (2015) sought answers from 27 interviewees in the networking and telecommunications firm Ericsson, which was in the midst of a push for CI. They discovered that lack of time was the primary challenge in accepting CI, with developers even cutting corners and skipping testing in order to get the features out. Frustrations with learning the new technology, dropping everything to investigate build errors, and putting up with test automation failures illustrated more barriers to embracing the CI mindset. Interestingly enough, most stakeholders were in favor of CI as a

general practice, but were unhappy with how it was implemented in Ericsson, specifically with the lack of communication and clear direction.

The conclusion from the ISTA framework and instances of DevOps resistance is that it is not enough to simply make a good enough technology. No matter how well-designed a technology is, it is important to intentionally integrate that technology into specific cultures, monitoring what adjustments must be made, how the technology is actually used, etc. in order to avoid undesired consequences. In the context of embedded DevOps, where technical challenges only exacerbate cultural tension, this means user perceptions must be understood and taken into account if there is any hope of widespread adoption. With such understanding, success stories such Karlesky et al.'s and Segatz's can be reproduced more effectively.

III. Research Approach: Discourse Analysis from Reddit and Professionals

To see how embedded developers perceive and use DevOps, I analyzed discourse from three primary sources, each providing a unique angle: (a) Reddit, (b) NDC Conference talks, and (c) Statements from tech companies. The goal was to leverage existing online data from a variety of perspectives to capture a holistic picture of embedded developers' perception of DevOps.

Quantitative Analysis Comparing Two Subreddits

Reddit is defined by TechTarget as "social media platform and forum-style website where content is socially curated and promoted by site members through voting" (Yasar & Stafford, 2025). Essentially, Reddit has numerous "subreddits" always beginning with "r/" which are digital communities devoted to a specific topic. Within subreddits, users can post and comment on posts. They can also add at most one "upvote" or "downvote" on each post and comment,

which reports the total number of upvotes subtracted by downvotes as a simple measure of popularity/agreement.

For my analysis, I chose to compare the two subreddits "r/embedded" and "r/ExperiencedDevs". According to their descriptions, the first is "dedicated to discussion and questions about embedded systems," and the second is "for experienced developers" (3+ years) (Embedded, n.d.; Experienced Devs, n.d.). Both have over 200K users and are in the top 1% of all subreddits, indicating they have a considerable amount of user traffic to analyze. It is due to all these data points that they were chosen as sources, so a quantitative understanding of how the first group (embedded developers) compare to the second group (experienced developers).

An assumption I make is that the majority of the "experienced devs" are general-purpose developers, which would indeed match the demographic of only 5% of developers being embedded (Noll, 2023). An additional consideration is that r/ExperiencedDevelopers was chosen as a control group over subreddits like r/softwaredevelopment and r/devops due to the ample amount of posts and comments focused on the discussion of DevOps practices found in r/ExperiencedDevelopers but not the others.

In both of these subreddits, the following process was followed:

1. Using each of the phrases "continuous integration" and "CI/CD", search the subreddit.
2. Select the top 3-5 posts, ignoring any irrelevant ones (some judgment was used as to exactly when to stop selecting the posts due to irrelevance). Stop once 6 posts are selected from the subreddit.
3. In each post, select the 4-6 comments with the most upvotes (again, some judgment was used here).

4. For each comment, give it a “DevOps Confidence” score from 1 to 5, with a 1 indicating the lowest amount of trust/acceptance of DevOps and a 5 indicating the highest. Table 1 characterizes the 5 groups.

Table 1

Example Sentiments of Each Level of DevOps Confidence (Created by Author)

DevOps Confidence	Example Sentiment
5	Automated tests aren't hard to do; it's obvious you need to do them
4	Automated tests can be hard to do but you should still do them
3	Sometimes you don't need to do automated tests if the effort outweighs the benefits
2	Automated tests are overrated
1	Automated tests cause more harm than good

It is worth mentioning that this method has several limitations:

- Due to the anonymous nature of responses, it is impossible to verify their integrity or weigh them based on demographics.
- The upvote system obscures a lot of information, like whether the same user upvoted several comments and how many downvotes there are.
- The DevOps Confidence score is subjective.
- Users of Reddit may not be representative of software developers in general.

Despite such limitations, the data collected still gives useful insight, especially due to the considerable sample size of 2,976 upvotes across 12 posts and 55 comments. Indeed, the quantity of data is the main reason Reddit was chosen to be included in the discourse analysis. By aggregating as much “average user” data as possible, I gained valuable statistics that represent the sentiments of embedded and general-purpose developers at large.

Qualitative Analysis Comparing Company Statements

Complimenting the quantitative study, I also performed an exploratory qualitative analysis focusing on professionals and companies as a whole. Due to the nuanced and unstructured nature of the location and content of such sources, both the method of finding sources and analyzing them were not as standardized as the step-by-step process with Reddit. Instead, an effort was made to understand how professionals and companies perceive DevOps from primary sources, with a goal to compare between embedded and general-purpose. The following sources emerged:

1. 3 Talks (roughly an hour each) from NDC Conferences, which are "high-end events for software developers" according to their website (*NDC Conferences*, 2025). These represent some of the most forward-thinking minds in the industry.
    a. *Continuous Delivery of Embedded Systems* by embedded developer Mike Long, from NDC Oslo 2015
    b. *Agile embedded development under regulatory constraints* by Espen Albrektsen, from NDC TechTown 2021, an annual conference with a focus on embedded development
    c. *Contrasting Embedded Software Development for the Space Shuttle and the Orion MPCV* by 30-year NASA engineer Darrel Raines, from NDC TechTown 2024 in Kongsberg, Norway
2. 3 Statements from general-purpose software company representatives
    a. Amazon AWS: *Automating safe, hands-off deployments* by Principal Software Engineer Clare Liguori in 2020

b. Google: Chapter 24 (Continuous Delivery) from the 2020 book *Software Engineering at Google: Lessons Learned from Programming Over Time* by release engineers Radha Narayan, Bobbi Jones, Sheri Shipe, and David Owens
c. Facebook: *Rapid release at massive scale* by release engineer Chuck Rossi in 2017

3. 3 Statements from embedded software company representatives
    a. Intel: *Continuous Delivery, Embedded Systems, and Simulation* by Director of Simulation Technology Ecosystem Jakob Engblom in 2017
    b. Cisco: *How Cisco IT Uses Agile Development with Distributed Teams and Complex Projects* in 2017
    c. eSOL: *Overcoming Major Challenges of Continuous Integration (CI) in Embedded Software Development* in 2025

The general method for analyzing these sources was reading/watching them and identifying where their sentiment stood towards DevOps. Contrasting the "average user" of Reddit, this is intended to present the "company standard," which could be different. Beyond simply judging how confident each source is of DevOps, additional insights were noted.

IV. Results: Embedded Developers Recognize the Need to Think About DevOps

On Reddit, Both Embedded and General-Purpose Developers Support DevOps

When adding up all the upvotes from both the r/embedded and r/ExperiencedDevs subreddits and sorting based on DevOps Confidence, the results plotted in Table 2 were found.

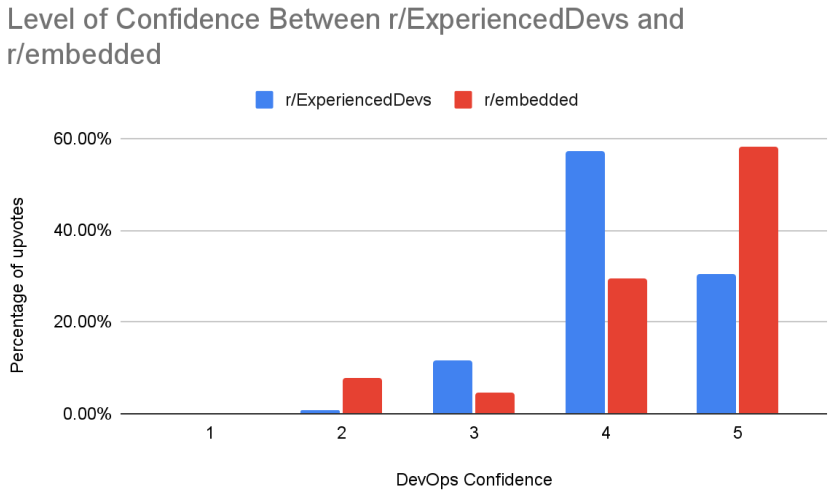
Table 2

Percentage of Upvotes from Each Level of DevOps Confidence (Created by Author)

	Percentage of Upvotes	
DevOps Confidence	r/ExperiencedDevs	r/embedded
1	0.00%	0.00%
2	0.85%	7.71%
3	11.63%	4.54%
4	57.10%	29.58%
5	30.42%	58.17%
Total	100.00%	100.00%

Figure 5

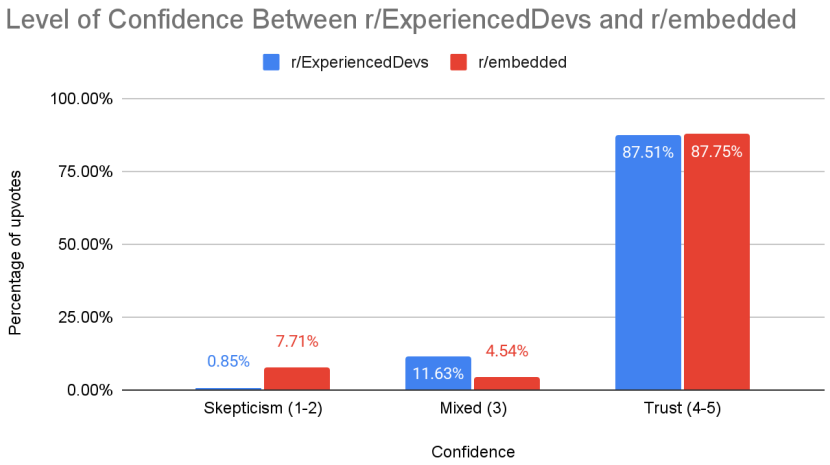
Bar Chart of Level of Confidence Between Subreddits (Created by Author)



If we partition the "Confidence Categories" into "Skepticism" (with a score of 1 or 2), "Mixed" (3), and "Trust" (4-5), we see that about 88% of upvotes from both the r/embedded and r/ExperiencedDevs subreddits are for posts and comments that fall under Trust.

Figure 6

Bar Chart of Level of Partitioned Confidence Between Subreddits: Both subreddits share about the same level of trust (Created by Author)



This gives evidence that embedded developers actually trust and accept DevOps roughly at the same rate as general-purpose developers, which is somewhat surprising.

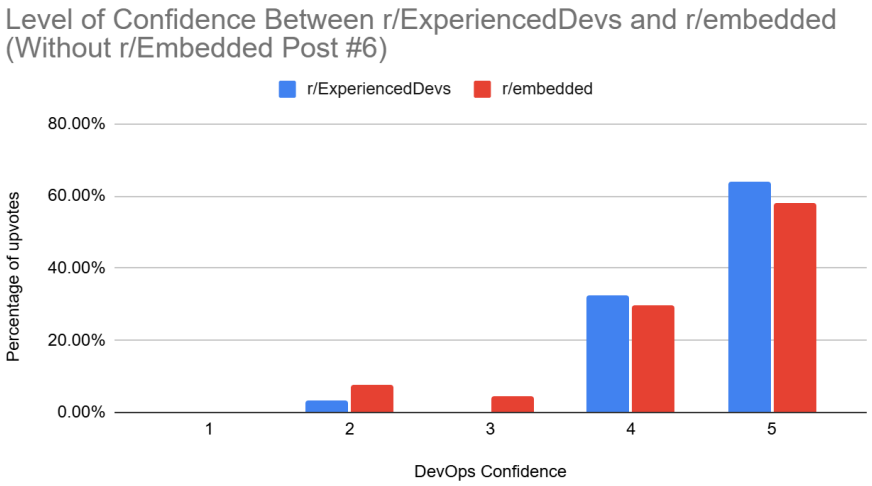
Besides these data-driven insights, several anecdotes are of additional interest. For instance, the two most-upvoted comments/posts in the selection from r/embedded are the posts *New YouTube course: Embedded CI/CD with HIL Testing using STM32CubeIDE, Git, and Jenkins* with 155 upvotes and *CI/CD for embedded software development* with 147 upvotes. In the first one, users praise the original poster (OP) for sharing his embedded CI/CD YouTube course, one commenting, “there is definitely a shortage on ci/cd for embedded.” In the second one, the OP laments the lack of CI/CD, saying that over his 7 year embedded career he has realized “the industry is (annoyingly) conservative and is struggling to catch up, compared with other forms of software development.” One of the commenters concurs, saying “Embedded developers, mostly the older ones, are very conservative indeed. I struggled to insert TDD into our projects, but now that they've seen the benefits of it, they're all the way in.” This implies that embedded developers widely recognize a need for DevOps practices to be revitalized in their field, with the suggestion that people need to see the benefits before trusting this process.

In r/ExperiencedDevs, more comments agree that culture is key to making DevOps work: "in my experience if the testing culture isn't set right at the start of the company it's very hard to correct it," and "CI/CD is just as much about culture as it is about automating stuff." Also in r/ExperiencedDevs, one post stands out, with the title *Anyone here with experience deploying CI/CD for embedded/HWIL testing?* Most responses agreed with the OP that automated tests for embedded are harder, but they all said it was worth it.

Another observation surrounding research method integrity is that one of the posts is dramatically larger than the others, with 1,387 upvotes across the post itself and top 4 comments (46.6% of all the upvotes in total!). If this post happened to be excluded, the results from Figure 7 would occur.

Figure 7

Bar Chart of Level of Modified Partitioned Confidence Between Subreddits: With one post removed, r/EmbeddedDevs shows higher trust (Created by Author)



In this scenario, r/ExperiencedDevs clearly has a slight edge against r/embedded in terms of DevOps confidence, instead of the two being tied, which means that if the data was chosen slightly differently, evidence could suggest that embedded developers are slightly more skeptical.

Overall, the data indicates support for DevOps practices from both embedded and general-purpose developers, though technical and social challenges for embedded DevOps are recognized, with many advocating for more attention towards this area.

Embedded Companies Strive to Strengthen Their DevOps to General-Purpose Levels

At large, the 9 sources selected from a mix of embedded and general-purpose developers share the same sentiment that DevOps is the way to go, again with embedded engineers treating "difficulty with embedded DevOps" as a well-established fact. First, I summarize the 3 NDC talks:

1. *Continuous Delivery of Embedded Systems* (2015): Embedded developer Mike Long advocates for DevOps practices like automated testing and CI/CD in embedded systems, illustrating the difference between extremely tedious "old school" development versus the streamlined, modern CI pipeline that non-embedded software has already embraced. An interesting case study at about 58 minutes is how Tesla was able to practice true CD via an "Over-the-Air" update of a new hill start feature incredibly quickly after receiving a request for the feature.
2. *Agile embedded development under regulatory constraints* (2021): Embedded developer Espen Albrektsen describes how his team has successfully implemented Agile methodologies in the development of their gas detector which must follow the strict legal regulation Functional Safety SIL2 (IEC61508). One might think that the speed of Agile might come in conflict with the rigid safety regulations defined by law, but in many ways the practices are well-suited for it, such as using automated tools to prove that 100% of the code has been tested and documenting required "equivalence class" and "boundary value" analysis.

3. *Contrasting Embedded Software Development for the Space Shuttle and the Orion MPCV* (2024): 30-year NASA engineer Darrel Raines relays the lessons he has learned from his time at NASA. He emphasizes the importance of quality assurance at every stage - documentation, peer reviews, unit tests, integration tests, system tests, and verification tests - especially in such critical software as spacecraft, as it has to be perfect the first time. This one is a bit different than the others since Raines does not mention automation, just the importance of taking tests seriously.

All three of these respected voices in the embedded field echo the same mantra: the non-software part of developing software needs special attention if the software is expected to be shipped with efficiency and quality, which is at the core of DevOps thinking. Though the third does not explicitly endorse automation, all three suggest that a DevOps mindset is accepted as the "right" one, even for embedded engineers.

Next, all 3 industry-leading companies (Amazon, Facebook, and Google) were found to practice DevOps to extreme levels, which is not surprising. Both Amazon AWS and Facebook deploy a new version to production (i.e., what real customers are seeing) every few hours. Meanwhile, the Google engineers didn't specify how many times they deployed, but still spoke highly of rapid deployment cycles: "A world of regular releases means that if a developer misses the release train, they'll be able to catch the next train in a matter of hours rather than days. This limits developer panic and greatly improves work–life balance for release engineers." All three companies release to a subset of customers before deploying to the entire client base as a built-in part of their pipeline, minimizing any potential consequences from hazardous releases.

Due to their wealth of experience and history that predates DevOps times, there is a theme of how a culture was an important aspect of DevOps adoption. In particular, the Google engineers devoted a whole section to "Changing Team Culture: Building Discipline into Deployment," describing the success of the Google Maps team developing the discipline to keep frequent release cycles on track. In contrast, they lament the still-laborious 50-hour release process of YouTube with many manual steps. Meanwhile, Amazon's author does not explicitly mention a cultural shift, but does report that the change towards a rapid, automated deployment pipeline was incremental instead of all at once. Finally, Facebook seems to be a bit different, since the culture prioritized rapid, continuous integration of new features since its earliest days, which the author claims helped Facebook rise above its competitors.

The final group of 3 consists of companies that specialize in embedded products: Intel, a leading manufacturer of semiconductor chips; Cisco, a digital communications technology conglomerate; and eSOL, a Japanese firm specializing in real-time OS. Like everyone before, all three spoke highly of DevOps practices.

Interestingly, Intel's Director of Simulation Technology Ecosystem wrote his 2016 blog post after being "inspired" by Mike Long's NDC talk about CD in embedded systems, with the takeaway that "being 'embedded' is no excuse not to work in a modern and efficient way." He recognizes the unique technical difficulties but maintains that the effort is worth it, additionally emphasizing the need for a combination of testing on simulation and real hardware. Similarly, eSOL reiterates the importance of CI in embedded development in its 2025 blog post, going into some detail about how they implemented hardware simulation to automate testing.

While the Cisco author shares the same positive sentiment as Intel and eSOL, they also give more valuable insight into the difficult cultural shift to put DevOps principle into practice in

their 2017 retrospective. They report some skepticism of fast release trains, trouble letting go of their old practices, and some tests still being done manually. Despite these struggles, senior Vice President Vjoy Pandey concluded that in the long run, “Continuous delivery improved quality, increased productivity, and improved the employee experience.”

IV. Conclusion: Embedded Developers Need to Eat Their Vegetables

Based on the data from both Reddit and company representatives, it seems safe to say that DevOps is accepted as “morally correct” in the software industry, even to embedded engineers. It also seems to be an accepted fact that embedded software is lagging behind the DevOps curve. Curiously, this seems to contradict the finding that embedded engineers tend to be as in favor of DevOps as their general-purpose peers, and that no embedded representatives condemn the practice. It is possible that the limited sample size of embedded representatives failed to capture any official DevOps dissent, but I posit the more likely reconciliation: Practicing DevOps is like eating your vegetables. No one will say that it is bad, but actually putting it into practice is another story, especially in embedded, where the vegetables are much harder to eat. This failure to “practice what you preach” seems to exist even in broader software, not just embedded, as suggested by the aforementioned 2016 study which found a whopping 94% of their 442 survey participants indicating interest in using CI in their next project, but only 40.27% of the 34,544 open-source GitHub repositories they parsed actually detected it (Hilton et. al.).

Whatever the case, it is clear that the embedded field needs to continue to pay attention to DevOps practices, since quantitative and anecdotal data shows it is helpful across the board, with no exception to embedded. Fortunately, many voices in the industry seem to recognize this problem, and such advocacy should be encouraged and continued.

Though this paper identifies embedded developers' concern about being behind in DevOps, there are several missing insights which future research should be devoted to. First, anecdotal evidence suggests that embedded DevOps is lacking, but no quantitative evidence of this was found. Second, the notion that embedded developers like to sing the praises of DevOps without actually putting it into practice could be explored via random interviews or surveys from a variety of embedded firms. Finally, several stories of embedded teams adopting DevOps practices were reviewed, but a data-driven, holistic idea of how best to shift embedded culture into the DevOps mindset is still missing. From the limited narratives seen here, it seems quality tools, open dialogue between management and workers, and time seem to be important factors for teams to accept DevOps.

In the end, by analyzing the discourse of a large quantity of Reddit users and high-quality company representatives, I was able to conclude that DevOps is accepted as the right thing to do, even in embedded, where most agree that it needs more work. I put forth the notion that DevOps is touted more than it is implemented, especially in the embedded sector. I also encourage future research to be done to understand the state of embedded DevOps, both culturally and in practice. Ultimately, by continuing to work to bring embedded DevOps up to speed with their general-purpose peers, their production will be faster, safer, and less worrisome.

References

Albrektsen, E. (Director). (2021, November 1). *Agile embedded development under regulatory constraints—Espen Albrektsen—NDC TechTown 2021* [Video recording]. https://www.youtube.com/watch?v=AzQQPyBwNyo

Bajer, M., Szlagor, M., & Wrzesniak, M. (2015). *Embedded software testing in research environment. A practical guide for non-experts.* 2015 4th Mediterranean Conference on Embedded Computing (MECO), 100–105. https://doi.org/10.1109/MECO.2015.7181877

DevOps toolchain. (2024). In Wikipedia. https://en.wikipedia.org/w/index.php?title=DevOps_toolchain&oldid=1230727551

Embedded. (n.d.). Reddit. Retrieved April 4, 2025, from https://www.reddit.com/r/embedded/

Engblom, J. (2017, February 7). *The Right Toolset for Testing (Testing Theory Part 2).* Intel. https://www.intel.com/content/www/us/en/developer/articles/technical/the-right-toolset-for-testing-part-2.html

Engblom, J. (2017, March 13). *Continuous Delivery, Embedded Systems, and Simulation.* Intel. https://www.intel.com/content/www/us/en/developer/articles/technical/continuous-delivery-embedded-systems-and-simulation.html

Experienced Devs. (n.d.). Retrieved April 4, 2025, from https://www.reddit.com/r/ExperiencedDevs/

Hall, T. (2020). *DevOps Best Practices.* Atlassian. https://www.atlassian.com/devops/what-is-devops/devops-best-practices

Harrison, M. I., Koppel, R., & Bar-Lev, S. (2007). *Unintended consequences of information technologies in health care--an interactive sociotechnical analysis.* Journal of the

American Medical Informatics Association : JAMIA, 14(5), 542–549. https://doi.org/10.1197/jamia.M2384

Hilton, M., Tunnell, T., Huang, K., Marinov, D., & Dig, D. (2016). *Usage, costs, and benefits of continuous integration in open-source projects.* Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, 426–437. https://doi.org/10.1145/2970276.2970358

How Cisco IT Uses Agile Development with Distributed Teams and Complex Projects. (2017, November 14). Cisco. https://www.cisco.com/c/en/us/solutions/collateral/enterprise/cisco-on-cisco/cs-boit-0617 2016-agile-development.html

Karlesky, M., Williams, G., Bereza, W., & Fletcher, M. (2007). *Mocking the Embedded World: Test-Driven Development, Continuous Integration, and Design Patterns.*

Koehler, C., Mayer, A., & Herkersdorf, A. (2008). *Determining the Fidelity of Hardware-In-the-Loop Simulation Coupling Systems.* 2008 IEEE International Behavioral Modeling and Simulation Workshop, 13–18. https://doi.org/10.1109/BMAS.2008.4751232

Laukkanen, E., Paasivaara, M., & Arvonen, T. (2015). *Stakeholder Perceptions of the Adoption of Continuous Integration – A Case Study*. 2015 Agile Conference, 11–20. https://doi.org/10.1109/Agile.2015.15

Leveson, N. G., & Turner, C. S. (1993). *An investigation of the Therac-25 accidents*. IEEE, 26(7), 18–41. https://doi.org/10.1109/MC.1993.274940

Liguori, C. (2020). *Automating safe, hands-off deployments*. Amazon Web Services, Inc. https://aws.amazon.com/builders-library/automating-safe-hands-off-deployments/

Long, M. (Director). (2015, June 24). *Continuous Delivery of Embedded Systems—Mike Long* [Video recording]. https://vimeo.com/131632946

Lwakatare, L. E., Karvonen, T., Sauvola, T., Kuvaja, P., Olsson, H. H., Bosch, J., & Oivo, M. (2016). *Towards DevOps in the Embedded Systems Domain: Why is It So Hard?* 2016 49th Hawaii International Conference on System Sciences (HICSS), 5437–5446. https://doi.org/10.1109/HICSS.2016.671

Narayan, R., Jones, B., Shipe, S., & Owens, D. (2020). *Chapter 24: Continuous Delivery.* In L. Carey (Ed.), Software Engineering at Google: Lessons Learned from Programming Over Time (1st edition). O'Reilly Media.

NDC Conferences. (2025). NDC Conferences. Retrieved April 4, 2025, from https://ndcconferences.com/

Noll, B. (2023, June 12). *Embedded Software Development*. Developer Nation. https://www.developernation.net/blog/embedded-software-development

Overcoming Major Challenges of Continuous Integration (CI) in Embedded Software Development. (2025, March 6). eSOL. https://blog.esol.com/embedded/continuous-integration_20230512

Raines, D. (Director). (2025, January 9). *Contrasting Embedded Software Development for the Space Shuttle and the Orion MPCV - Darrel Raines* [Video recording]. https://www.youtube.com/watch?v=yR6Jkkch4fo

Rossi, C. (2017, August 31). *Rapid release at massive scale*. Engineering at Meta. https://engineering.fb.com/2017/08/31/web/rapid-release-at-massive-scale/

Segatz, F. (2023). *Continuous Integration for Embedded Software with Modular Firmware Architecture.* https://kth.diva-portal.org/smash/get/diva2:1835965/FULLTEXT01.pdf

Silvis-Cividjian, N. (2024). *Therac-25 Accidents: We Keep on Learning From Them.* Computer, 57(12), 69–78. https://doi.org/10.1109/MC.2024.3450197

Tarlinder, A. (2016). *Developer Testing: Building Quality into Software.* Pearson Education. https://books.google.com/books?id=Csz3DAAAQBAJ

Wang, C., Liu, A., & Carter, D. (2022, June 2). *Evolution of the Software Development Life Cycle (SDLC) & the Future of DevOps*. Sapphire Ventures. https://sapphireventures.com/blog/the-future-of-devops/

Woolley, R. (2021). *Deploying Embedded Applications Faster with Containers.* Wind River. https://www.windriver.com/resource/deploying-embedded-applications-faster-with-containers

Yasar, K., & Stafford, C. (2025, January). *What is Reddit? How it Works, History and Pros and Cons.* TechTarget. Search CIO. https://www.techtarget.com/searchcio/definition/Reddit

Yoo, S., & Jerraya, A. A. (2003). *Introduction to Hardware Abstraction Layers for SoC.* In A. A. Jerraya, S. Yoo, D. Verkest, & N. Wehn (Eds.), Embedded Software for SoC (pp. 179–186). Springer US. https://doi.org/10.1007/0-306-48709-8_14