Rollbar

eBook

Why Developer Experience Matters and How Continuous Code Improvement Can Help

Table of Contents

Introduction

When's the last time you took a good look at your engineering team? Are they energetic and bright-eyed—or are they bent over their screens with a look of total stress and frustration? This question is more important than you may think.

Research is beginning to recognize that developer satisfaction is affected by deeper issues regarding coding culture and product quality. And these culture and quality issues can serve as an important wake-up call to uncover considerable flaws in your development lifecycle.

In this article, we examine the roots of software engineers' frustration and the insights this can shed on the development lifecycle. We then get down to business, showing you how to address these issues through continuous code improvement (CCI).

1 The Case for Developer Happiness

Believe it or not, your team's output isn't just determined by brain power or working harder, but by something much more basic: happiness.

Happiness has a direct impact on work quality and productivity. One study conducted by Oxford University's Saïd Business School found that happy employees are 13% more productive at work. The link between satisfaction and productivity is even more pronounced when it comes to software development.

According to research presented in the book Rethinking Productivity in Software Engineering, unhappy developers are alarmingly less productive at the workplace. These individuals are more prone to missing deadlines, producing poorly written code, discarding work, and deviating from work timelines. In addition, they exhibit low motivation, mental angst, and impaired cognitive performance. And these issues are exacerbated when you take peer effects into account.

Emotions are contagious. This is especially problematic in the workplace where one subpar employee can degrade the performance of other employees. Sigal Barsade, from the Wharton School of Business, found that the thoughts, emotions, and practices of just one employee can negatively affect entire teams in unexpected and damaging ways. Harvard Business Review backed this up, claiming that the "bad behaviors of one employee spill over into the behaviors of other employees through peer effects." In other words, dissatisfaction can spread like wildfire.

So what can companies do about this? It may be tempting to think of happiness as something beyond your control and simply blame things like disposition, innate laziness, or family troubles. But the truth is much more interesting: **Your developers' unhappiness is closely linked to the actual product they are working on**.

A recent survey of 1,300 developers found that software engineers' dissatisfaction is four times more likely to be due to what they are working on (both the artifact and process)—not personal reasons. This is a powerful discovery. It means that where there is an unhappy developer, there is most likely something wrong with the product and the way he or she is working on it.

Developer experience matters. Not just because you want to have happy and productive employees, but because dissatisfaction is indicative of deeper issues.

In order to get your team engaged and productive again, we need to take a closer look at the reasons behind this unhappiness.

2 What's at the Bottom of Developer Dissatisfaction?

Although helpful in weeding out bugs, the traditional software testing process is part of what is known as imperfect debugging. Imperfect debugging is not ideal, as it can:

- Getting stuck trying to solve a problem
- Working under time pressure
- Working with poorly written code or bad coding practices

These problems are all interconnected. Poor coding practices make it difficult to solve problems efficiently, which in turn increases time pressure, and bad code is often the result of management trying to save time and effort in the first place.

Another big cause of dissatisfaction is that developers find themselves spending way too much time on administration and unnecessary rote tasks. Instead of soaring ahead, getting creative, and allowing the juices to flow, developers are often bogged down with tedious tasks such as debugging and version control.

The bottom line is that developers are constantly being interrupted. And that elusive hyper-productive state of flow, getting into the zone, is constantly being broken. These problems highlight the importance of moving away from short-term thinking and quick fixes to strategically rethinking the entire software development lifecycle (SDLC) and developer culture.

The rest of this article is devoted to how to get developers back into the flow, making everyone happier and more productive, and as a result, shortening release times and improving overall product quality.

3 Elite Teams are Made Up of Happy Developers

There's a deep connection between cultivating an elite team and positive developer experience. In the following section, we'll discuss why.

Google's DevOps Research and Assessment (DORA) team has identified key markers of an elite team. These are:

- On-demand deployment multiple times a day
- Successful code changes in production in less than a day
- Service restored in less than an hour
- 0%–15% rate of failed deployments and code remediation

To pull these numbers off requires incredibly smooth operations. It calls for a strategy in which teams spend drastically less time on last-minute emergency hotfixes and patches and get to the important stuff as quickly as possible, naturally eliminating the things that make developers miserable. This isn't just a pipe dream—it can be possible through a radical change in approach.

4 It's Time to Take a Fresh Approach

Developer dissatisfaction related to their work really boils down to the SDLC (Software Development Lifecycle), which is made up of the following phases:

1. Requirement analysis
2. Planning
3. Architectural design
4. Software development
5. Testing
6. Deployment

There are different methodologies for handling the SDLC, such as the waterfall model, spiral model, and Agile model.

When you stop to think about it, the development lifecycle is a never-ending loop. If there's one thing you can learn from developer dissatisfaction, it's that getting stuck on a certain phase of the lifecycle is incredibly frustrating and damaging—both to code quality as well as employee morale. Rethinking your SDLC methodology is therefore crucial.

The smarter your SDLC strategy, the faster things will go, the better the code will get, and the happier your team members will be.

Flow should be a key aspect of your strategy. This means moving away from a mindset of fixing to building; from rigid linear steps to continuous deployment with short iteration loops and a greater emphasis on agility, automation, and dynamic processes.

Let's start by introducing an important role to your company: the developer experience engineer.

5 Anyone Call for a Developer Experience Engineer?

With all due respect to Abraham Lincoln's opinion that "every man's happiness is his own responsibility," it's not necessarily the case when it comes to software developers in the workplace. Engineers should focus on what they do best, that is, designing and building things. They don't always know how to optimize processes, which is an essential component of work satisfaction. This is where the role of DXE (Developer Experience Engineer) comes in.

The DXE is in charge of maximizing developer productivity and boosting their work experience. It's his or her job to find the right tools, create the best environment, and instill best practices that will clear away the obstacles and allow developers to work on what matters at full force.

Some of the big players have already begun to pay close attention to the issue of effectiveness. In 2014, Twitter created an "engineering effectiveness group," and Google has a huge "engineering productivity" team devoted to optimizing the engineering process.

If you don't already have a DXE, it's definitely worth considering, especially as your teams grow in size and collaboration becomes messy and complex. The DXE can have a positive effect on developer satisfaction, productivity, and experience, as well as product quality. In addition, such a role can have an overarching impact on business value and revenue growth. A recent McKinsey report found that prioritizing developer velocity (not to be confused with subjecting them to more pressure) can increase revenue growth by four or five times.

Whether the role of DXE exists in your company yet or not, someone has to roll up their sleeves and start to put some conscious effort into these issues.

To understand what needs to be done, let's take a look at the connection between high-performing teams and developer satisfaction.

How can we achieve all this?
The answer lies in continuous code improvement.

6 Introducing Continuous Code Improvement

Innovation is a high-speed game, and it's only getting more intense. To keep up with the competition, companies must meet shorter and shorter release times, putting immense pressure on development teams. This pressure is counterproductive, resulting in less testing time, degraded code, and strung out employees. There is a way to turn this pressure cooker into gold and transform the sense of urgency into high quality. But to do so requires a shift in perspective, a shift made possible by CCI.

Development speed and code quality may seem like two diametrically opposed goals, but CCI thinks of these as two sides of the same coin. It is a practical solution that allows teams to release higher quality code at a faster pace.

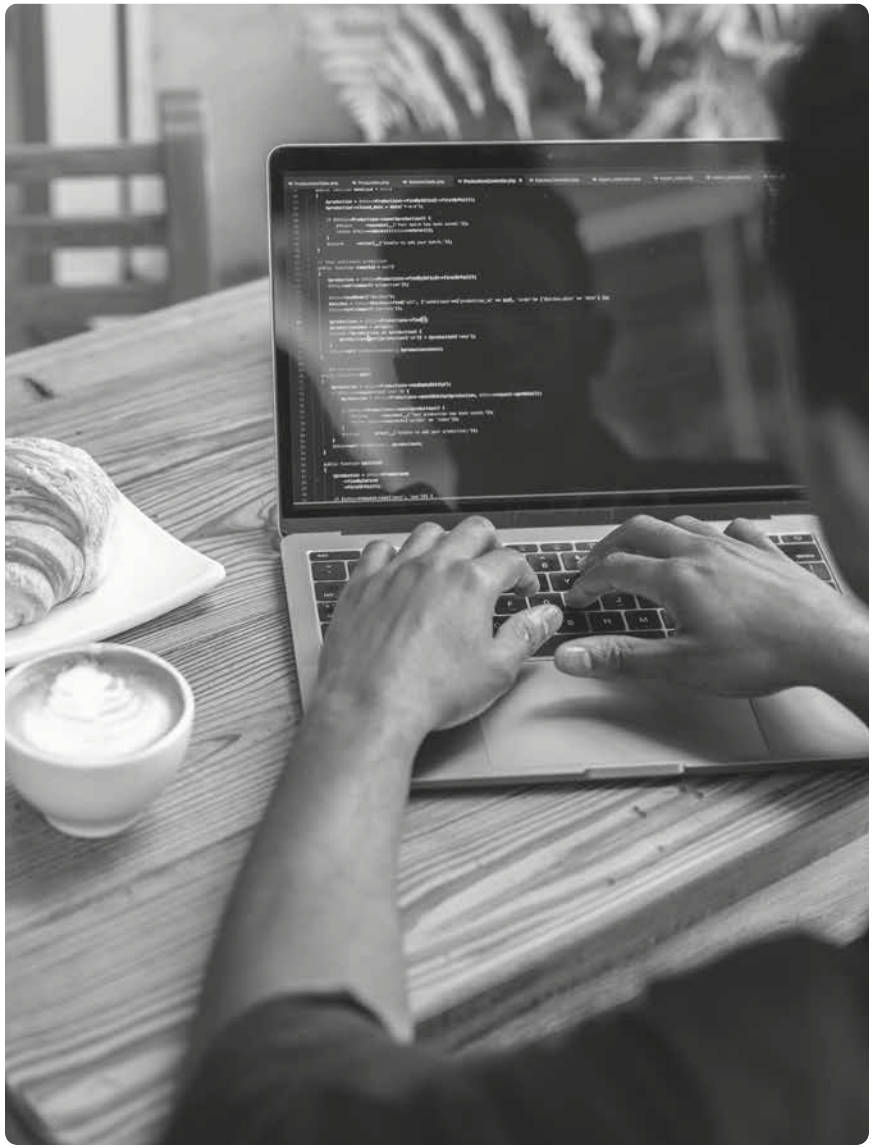
The CCI approach accepts that code is not a static work of art, but in fact, an ongoing process that is constantly changing, running, or breaking. Accepting that code can never be perfect is incredibly liberating because it frees up developers to start working dynamically.

The beauty of CCI is that it treats problems as solutions—errors, bugs, user feedback, and testing aren't obstacles you get stuck on, but important signposts that let you know what the problem is and how to solve it as effectively as possible.

CCI provides a quick feedback loop that closely monitors code, delivering information on code that must be improved, and providing tools to automatically remediate issues and release code as quickly as possible to continue the development lifecycle. So rather than getting stuck on a problem, it allows things to keep moving. It allows developers to cut to the chase and devote most of their time improving code rather than tediously monitoring, debugging, and hunting around for the cause of the problem.

Looking back at Google's DORA conclusions, continuous code improvement is at the heart of becoming an elite performer and creating happy, productive workers. It allows for speed, fewer errors, and dramatically faster fixes. But before we see how to implement this, let's consider some of the great benefits of adopting a CCI approach.

7 The Deep Impact of Continuous Code Improvement



If you implement CCI with the right tools, the benefits are immense and comprehensive:

- **Developers** release higher-quality code faster and with fewer bumps in the road.
- **Managers** enjoy more effective and satisfied team members, with fewer middle-of-the-night emergency incidents.
- **DevOps engineers** experience less friction in the SDLC and smoother operations overall.
- **DXEs** have a way to increase the satisfaction and productivity of their team members.
- **End-products** are of higher quality, with fewer bugs and new features.
- **End-customers** experience a better product with faster problem resolution.
- **Company revenues** increase through decreased time-to-market and increased customer satisfaction and loyalty.

So how can we translate all this into action? Let's take a closer look.

8 Jumping Back into the Flow with CCI

Continuous code improvement is all about making the wheels of the SDLC turn faster with less friction so that developers can spend most of their time in a productive (and joyful) flow.

This section examines how to do this by integrating CCI into the SDLC.

Rethinking Developer Culture

Continuous Code Improvement calls for a big shift in developer culture and practices. Software has gone through profound changes in the last decade. Applications have become elusive creatures, comprising many different microservices and built on a decentralized cloud infrastructure. As such, they are updated and changed constantly and rapidly. However, the way developers work with code has not adapted to this new reality.

CCI allows developers to thrive in a world of modern complex architecture, with dynamic coding practices, shorter release cycles, and a new way to deal with errors.

Shorter Release Cycles

A major pillar of CCI is to shorten release cycles using compact feedback loops. It does so by adding a layer of information to each stage of the loop, from DevOps to staging to release. This extra information doesn't interfere with the lifecycle but actually speeds it up. It helps predict, detect, and remediate errors in real time.

In addition, developers work on smaller, more manageable pieces of code, allowing them to focus on the features that matter most and contribute to a smoother delivery pipeline. This means, for example, that developers can go ahead and improve their code without having to wait for long testing cycles to finish. The result is that deployment becomes less risky, new features are introduced more quickly, and the customer experience is vastly improved and minimally impacted by new releases.

Tackling Issues Proactively

Code changes constantly. So without a mechanism to cope with the unexpected, developers are constantly taken by surprise, forced to resolve issues reactively, and put out fires as they appear on the scene.

Standard automation solutions don't help matters since they rely on hard-coded algorithms, often leading to false or missed alarms and failing to identify unique bugs. Because of this, developers learn to mistrust these signals and wind up debugging manually, tediously digging through logs by hand.

With a continuous code improvement strategy, developers gain real-time visibility into errors and can proactively predict, detect, and fix issues faster through automated workflows. This can boost operations considerably, reducing mean time to acknowledge (MTTA) and mean time to resolution (MTTR).

Don't Do it Alone

There are many different ways to apply CCI to the development lifecycle, but tackling this huge task on your own can be quite daunting. This is where continuous code improvement platforms, such as Rollbar, can bring immense value.

According to McKinsey, companies that adopt strong tools such as continuous delivery platforms that drive developer velocity are 65 percent more innovative. Imagine a CCI platform that integrates seamlessly with all of your environments and tools. Examples include Kubernetes, Prometheus, and Weaveworks' Flagger, as well as issue-tracing tools such as GitHub and Jira.

Once integrated, a CCI platform can:

- Group error patterns and provide accurate and actionable alerts
- Provide top-notch error detection that continues to improve with automation-grade grouping, powered by machine learning algorithms that train on databases of billions of errors
- Deliver the contextual data and metadata needed to fix code as quickly as possible
- Automatically generate tickets to handle critical issues
- Proactively remediate issues with automated workflows
- Automatically halt or rollback a release or toggle a feature on or off that contains an error

The list of benefits goes on and on. To start benefiting from CCI immediately, you just need to find the platform that works best for you. So let the searching begin! It's worth the effort.

9 Conclusion

Developer experience matters. It's not just more enjoyable to work with happy people, it's also more productive. Luckily, the things that make developers happy are also the things that make your code great and your product amazing.

A happy software engineer is someone who feels his or her code is moving through the pipeline, impacting users. One of the best ways to do this is by bringing continuous code improvement into the center of your operations. With the right platform, this approach will not only make things move faster but also improve the quality of the work being done—and bring your team back into a healthy state of flow.

About Rollbar

Rollbar is the leading continuous code improvement platform that proactively discovers, predicts, and remediates errors with real-time AI-assisted workflows. With Rollbar, developers continually improve their code and constantly innovate rather than spending time monitoring, investigating, and debugging. More than 5,000 businesses, including Twilio, Salesforce, Twitch, and Affirm, use Rollbar to deploy better software, faster while quickly recovering from critical errors as they happen. Learn more at **rollbar.com**