Whether you're a business leader, an engineering manager, a seasoned developer or in your first developer role, I hope that you'll find this report useful.

———

Brian Rue
CEO and Co-founder, Rollbar
When Cory Virok and I started Rollbar in 2012, we knew something was lacking in how software was being built. Developers continue to get better everyday at building applications — the widespread adoption of microservices architectures and open source are evidence of this. But, we realized something was still holding us back. And that was how we track and fix bugs.

We both personally felt the pain of spending hours, days, or even weeks investigating issues, combing through logs, to not just figure out how to fix a bug, but to figure out what the bug was. We had application performance monitoring (APM) tools but they only told us the health of our system and infrastructure. The rise of observability is helpful to gain that systematic insight as software becomes more complex. But we didn’t need to understand the health of our systems, we needed to know where our code was broken.

I knew we weren’t the only developers lacking that insight, and I know that pain is still felt in companies, large and small, today. That’s why I’m excited to share our first “Code Improvement Report” that provides insights into the current state of how developers are building software and dealing with the inevitability of bugs and errors.
Nearly every developer surveyed responded that traditional methods fall short. They’re spending too much time manually investigating and responding to errors. They’re not getting the information they need to remediate them quickly. And the tools available are too focused on system stability instead of code health. Plus, users are often the ones reporting issues. This is creating significant business problems, including the risk of losing users.

A serious pain point from those surveyed is that they’re spending too much time on errors. A majority said that, instead of fixing bugs and errors, they could be building new features and functionality. A significant amount also say that they spend up to half of their week fixing errors. It’s forcing developers to be less productive, while also affecting their happiness in their roles.

Nearly every engineer surveyed said they could be deploying updates more often, but they’re not able to. The biggest reason is because QA and testing takes too long. So, despite more investment in making sure everything works correctly, developers are not only reporting it’s slowing them down, they report that issues still happen and, as noted above, traditional methods for fixing those problems fall short.

We surveyed nearly 1,000 developers across the U.S. to find out, and uncovered key trends and insights, including:

1. **Traditional Error Monitoring Falls Short**

2. **Too Much Time is Spent Manually Fixing Errors**

3. **Fixing Errors & Bugs Are Slowing Deployments**
Part One:

**Traditional Error Monitoring Falls Short**

Developers don’t have the tools and information they need to fix code issues quickly. Not only are the standard methods of finding bugs lacking, but most issues are still being reported by users. This creates real risks for businesses by leaving a broken experience for users to find. Respondents say that users are reporting bugs via social media, compounding the potential fallout by posting those problems in a public forum. Developers are struggling to quickly and easily find and fix issues, not only before they impact users, but even after the problem has been reported.
Nearly every developer surveyed noted that traditional methods for monitoring errors in code aren’t good enough. The number one reason (39%) is because they need to manually respond to errors. Respondents said that it simply takes too long to find the information they need to quickly resolve errors. Plus, traditional tools to find errors and bugs are too focused on system stability and not on code health.

36% noted it takes too long to find contextual information to fix errors.
31% cited the need to manually respond to errors.
39% said traditional error monitoring tools are not focused on code health.

Most respondents say they’re already leveraging error monitoring tools. Yet, users are still catching bugs before they do and developers struggle to fix them even when they become aware.

How developers are detecting errors:

- Error monitoring tool: 46.3%
- Combination of various tools: 45.2%
- Logging tool: 39.7%
- Wait until somebody complains: 26.4%
- APM tool: 19.1%
- None: 8.8%
- Other: 2.8%

For many, losing users is a reality, not just a risk. And the issue is compounded when users are reporting errors before development teams are even aware of them.

34% say losing users is the biggest risk of errors in their software applications.
28% say it hurts their ability to attract new users who might read bad reviews.
26% actually reported losing a significant amount of users to errors.

Errors Create Real Business Risks

The State of Software Code Report // 2021
Part Two:

Too Much Time is Spent Manually Fixing Errors

Engineers are hired to build software. That means they deliver value to users by creating new features and improving functionality. But many of those we surveyed say they spend a good chunk of their time fixing bugs. That’s thousands of hours spent not working on features or innovations. It’s not only hurting their productivity but also their wellbeing, causing frustration or worse.
Fixing Bugs & Errors is Creating a Lot of Pain

44% of developers say fixing bugs & errors is their biggest pain point

Other pain points →

All of the top obstacles respondents noted relate to resolving issues. Testing is causing pain but should alleviate the fear of deploying by catching issues before a release (Part 3 dives into the consequences). Requirements gathering and estimating level of effort also impacts the productivity and happiness of developers.

Developers are Spending Too Much Time Fixing Bugs

8% say they spend up to 75% of their time fixing bugs

37% said they spend more than 25% of their time fixing bugs

That’s Thousands of Hours Spent Every Week Fixing Instead of Writing New Code

32% are spending up to 10 hours a week fixing bugs

16% are spending up to 15 hours a week fixing bugs

6% are spending up to 20 hours a week fixing bugs

That Time Could Be Spent Building New Features...

Developers surveyed said that fixing code is taking time away from more important projects at work.

52% would use that time to build new features and functionality

42% said they would simply be able “to do their job”

...Or Doing Literally Anything Else

It turns out that repairing broken code isn’t most developers’ favorite activity.

26% would rather spend time paying bills

21% would rather go to the dentist

20% would rather spend time with in-laws

Fixing Bugs is Taking a Toll on Developers Personally

All that time spent manually responding to bugs and errors is taking a huge toll on developer’s job performance, their morale, and even their quality of life.

Fixing bugs makes developers feel:

Frustrated 31%
Overwhelmed 22%
Burned out 17%
Resentful 12%
Want to quit their jobs 7%
Part Three:

Fixing Errors & Bugs Are Slowing Deployments

Velocity is increasingly becoming a primary goal for engineering teams. They’re evaluated on how quickly they can deliver new features and functionality — as well as how fast they can fix issues. But an inability to fix bugs issues can slow down testing, QA, and staging, compounding the issue. And nearly every respondent says they’re being held back from faster release cycles. Escaped bugs are the biggest culprit.
Development Teams Could Be Deploying More Frequently

84% of respondents say their team is being held back from deploying more often

Top 3 Reasons →

30% say their team is too small
36% say testing / QA takes too long because errors & bugs are hard to fix
16% say their budget is too small

43%
say errors delay their production schedules

Developers Struggle to Find the Source of Errors

Respondents say they could iterate faster but they have trouble seeing where errors originate. They also say they need capabilities to get error information in real-time, and they need access to rich contextual information for errors.

86% say new capabilities would allow them to iterate faster, including:

47%
Seeing where each error originates

34%
Real-time capabilities

33%
Access to rich contextual information about each error

33%
Aid in quality assurance

22%
Open API access

They Lack Resources to Have More Confidence in Deployments

86% say they need better tools to detect and fix code errors

36%
need a bigger team

38%
want improved project management

36%
need more financial resources

30%
need a bigger team

38%
want improved project management

36%
need more financial resources
Today’s businesses are software businesses. If there was any positive in 2020, it’s the power software has to allow us to continue in some “normal” sense. Yet, this survey illustrates that too many companies and their development teams still have a major blind spot when it comes to errors in their code.

Enterprises need to deploy releases frequently and continuously in order to deliver consistent customer value. Improving code by eliminating bugs is part of that. Even small releases can have big customer impacts. But developers are reporting that still remains an Achilles heel for many of them, and it’s something that should become a focus in 2021.

Methodology & Demographics:
This national survey of 950 developers and engineers was commissioned by Rollbar and conducted by Propeller Insights, an independent survey research firm between December 22 and 28, 2020. Propeller Insights uses quantitative and qualitative methodologies to measure and analyze marketplace and consumer opinions. Survey responses were nationally representative of the U.S. population for age, gender, region and ethnicity.
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. Learn more at Rollbar.com.