RAYGUN

Software intelligence

The secret to building flawless web and mobile apps

By Nick Harley and Freyja Spaven

Published by Raygun.com - www.raygun.com.

Table of Contents

SECTION 1

Introduction

Software bugs. Those pesky little critters.

Able to find their way into your production environment no matter how hard you try and keep them out. Lurking in the undergrowth of your codebase and ready to pounce upon you at 5 p.m. on a Friday afternoon, software bugs and performance issues are unwelcome in your software applications.

When your development team is held accountable to building software that is fast, intuitive, and doesn't crash, it's disappointing for everyone in the company, from the CEO right through to the front line developer and beyond, to hear that customers are experiencing issues.

Of course, no software development team codes bugs and problems into their software intentionally, but frankly they're a pain to deal with. After all, they stop teams from moving forward, i.e. shipping new features and improvements. You end up looking back on the past rather than what can be done in future. Creating frustrated and unhappy customers rather than advocates in the meantime.

It isn't just bugs that affect end users though, performance issues like slow loading pages or apps that completely crash for the end user are also the likely cause of them losing patience and heading elsewhere.

Bugs and performance issues lead developers around in circles, chewing up vast amounts of time that could be better spent writing new code.

Digging around in log files and support tickets is simply no fun, and many teams cite their inability to replicate the issue the customer reported as a reason for their own frustration in dealing with customer complaints.

Yet, many teams are discovering how they can build scalable, robust and higher quality applications without the frustrations that surround finding and fixing issues in their codebase.

Software intelligence can give development teams unrivaled insights into how their applications are performing for their customers, whilst creating workflows that make the processes around issue resolution, effortless.

Living in a world of poor quality software

As we browse around on the web each day or use our mobile apps, we inevitably encounter problems. Bugs, slow pages and app crashes are all part and parcel of our interactions with software applications each day.

When I was a teenager, I had a newspaper round as my first 'real job', and I would be paid peanuts to walk the streets and deliver the local news to households throughout my neighbourhood. I had some frustrations with the job, and didn't really mind the low wages when it came to pay day, which on reflection seemed ridiculous for the effort involved. The most frustrating thing about a newspaper round for me was having to deal with poorly maintained letterboxes for many of the houses on my round.

I grew up in England, so if you didn't know this, nearly all letterboxes there are located in the front door of the house, rather than a mailbox at the end of the driveway. Letterboxes that were completely broken were pretty common.

This left me wanting to avoid these houses on purpose and the homeowner remaining oblivious to my plight to deliver their weekly tabloid fix.

Why did these people never think to fix their letterboxes?

I didn't expect homeowners to lie awake at night, thinking up ways to perfect the proficiency of their letterboxes, but it occurred to me that I did not tell them anything was wrong, nor did any of them actually use them themselves. When was the last time you put something through your own letterbox?

Now that I work in web development, I've often used the analogy to explain the fact that we don't really use the websites and mobile apps we build in the same way our users do. We never get to experience things from their perspective, and users come and go without you really knowing if they had a good experience (or them telling you that they encountered issues). Instead, they just avoid using you in future. When was the last time you used your own payment system?

I'm sure you've been exposed to this yourself. How do you feel when the apps you use don't work properly or crash completely? It's likely you feel the developers that built them are not doing things properly, are of poorer quality than your own team or that they must be doing something wrong.

Unfortunately this might not be problem. You see, although you build software for a living, as you browse other websites and apps, you are the 'user' in this situation, not the 'developer'.

I'd also take a guess that when problems arose, you didn't open up Chrome dev tools or downloaded a sample crash report from your device, took the time to write up a detailed report and send it off to the company via their contact form. Without ever knowing if it reached the developer that could fix it or not.

Software issues are part and parcel of our daily lives, but the experience of how the developer gets to see what their users are experiencing (and by that I mean ***truly experiencing***, not synthetic testing), is fundamentally broken.

How do you get full visibility on the end user's experience when they use your application?

Breaking out of the development status quo

Waiting for frustrated customers to report problems before taking action, or developers spending countless hours digging through logs trying to find the root cause of a problem, is viewed as archaic to many fast-moving organizations.

That's with existing software developers individually spending around seven hours every week hunting down and fixing bugs.

Developers are often swayed when choosing the right framework or programming language to build their applications by what is 'hot right now' in the developer community, rather than if it is the most functional and appropriate choice. I'm sure I could bash in some nails using a wrench, yet a hammer might be the better tool for the job, even if my colleagues think the wrench is the cooler tool that everyone is using right now.

This way of thinking tends to make developers become huge advocates for a particular tool or methodology, even though it might not be serving their best interests.

The same goes for the way they handle problems in their applications. There's always the team that monitors errors using their own code, firing unhandled exceptions to a shared inbox. Or the team that built their own reporting solution. Or the team that don't have any monitoring at all and rely on customers to report issues that they experience.

Hello help desk?

I can't type my credit card number into that white box thingy in your mobile app.

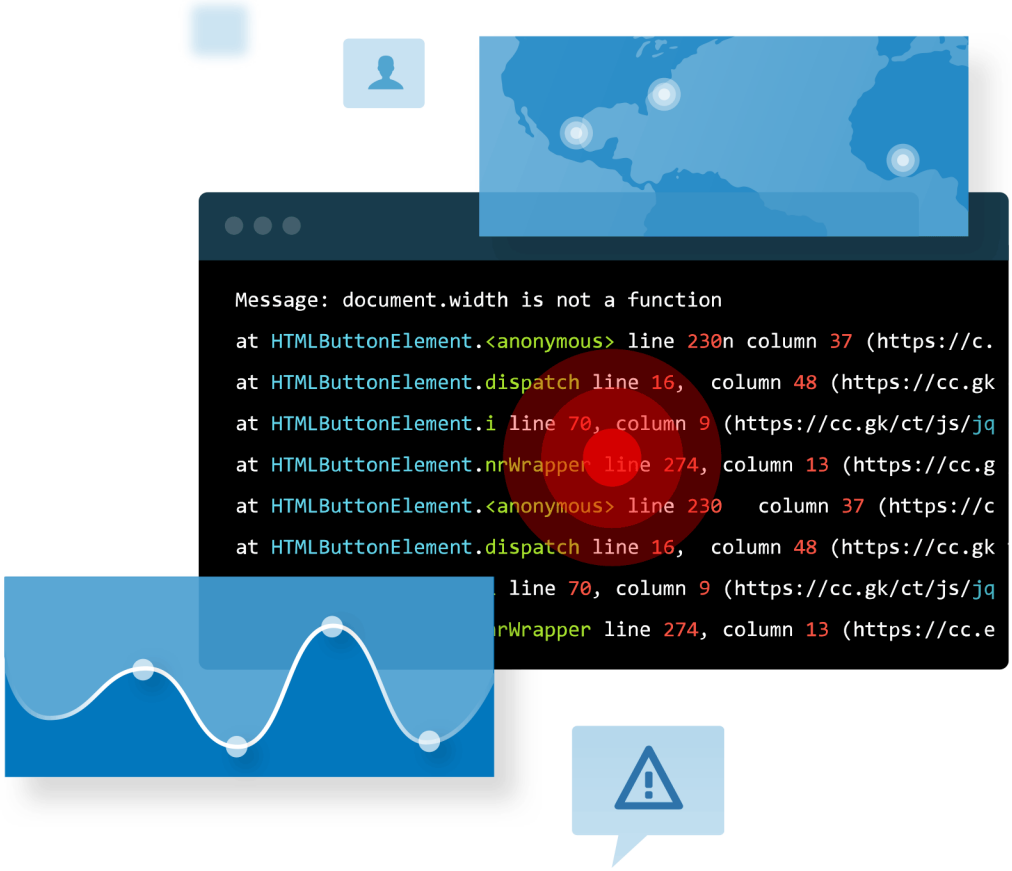
It keeps telling me that there is an error...

... I'm not happy!

Without extreme incidents or even knowledge that so many issues are happening in their applications and affecting users, teams carry on with the status quo, deeming their solution to errors, crashes and performance issues in the codebase acceptable, whilst frustrated users walk away without ever coming to the attention of the core development team. So many conversions, so much revenue and so much time is slipping away. Yet developers don't even try a new approach that could make their lives so much easier.

Software development as an industry has been around for many many years now, and as things have evolved there have been natural shifts in the way that software applications are put together. The tools available to developers to actually build applications has also greatly improved. Sophisticated monitoring is within reach for all software development teams, and monitoring the entire stack has never been more important.

With such a low barrier to entry (adding a few lines of code into production), visibility on every software issue that is affecting your application is just minutes away. You'll be surprised how many issues are present that you had no idea existed.



The solution - software intelligence

Software performance is essential to scores of businesses. As we've covered, when customers experience issues with reliability and software quality, they'll simply go elsewhere.

But monitoring the entire tech stack as one has become complex, especially with so many languages, platforms, and operating systems that make up the development jigsaw.

By implementing a software intelligence platform into their application's technology stack, developers gain unrivalled insights into application performance. This allows them to pair the aggregation of problems with deep diagnostic details and actionable insights on how to fix the underlying cause.

> “Making sure developers have production insight is key to deploying more frequently, faster feedback loops, and smaller blast radiuses. Software Intelligence is a key part of the tooling which has allowed us to increase developer efficiency by 40 percent YoY.”

Pushpay’s VP Engineering Josh Robb knows the importance of full stack monitoring first hand.

Application monitoring is becoming more and more sophisticated. Applications are becoming more complex. It’s important to keep a view on your customers’ true experience whilst using your software applications.

Software intelligence can ensure you are delivering perfect software experiences for your customers each and every time - reaching the next level of insight you were previously missing or even unaware of, with all the tools you need to produce world class software with greater agility.

SECTION 2

Benefits over traditional Application Performance Monitoring (APM)

Even the greatest software teams and products experience errors, issues and outages at some point - it's an unfortunate but inevitable fact, no matter how hard we all try to avoid these situations.

I'm yet to meet a software development team who are actively looking for ways to take the product offline or cause a user unnecessary misery when their app won't work, but what separates the best from the rest when it comes to creating reliable and robust software is how teams deal with those problems when they arise and how they minimise the affect on end users.

When it comes to incident management, you should have your own processes in place (hopefully), but we all dread that notification ping at some unreasonable hour of the night. Every minute counts trying to rectify the issue.

The truth is, most developers find fixing crashes and performance issues to be an unenjoyable job. Software teams want to be building the future rather than fixing the past. Yet software bugs are often the cause of much frustration for end users and lost revenues for many businesses.

Application performance management (APM) tools are designed to monitor the performance and availability of software applications. Striving to detect and diagnose complex application performance problems to maintain an expected level of service.

APM has been commonplace in Enterprise software development for several years now. These tools are often good at identifying there is a problem, but not the underlying root cause for front line developers.

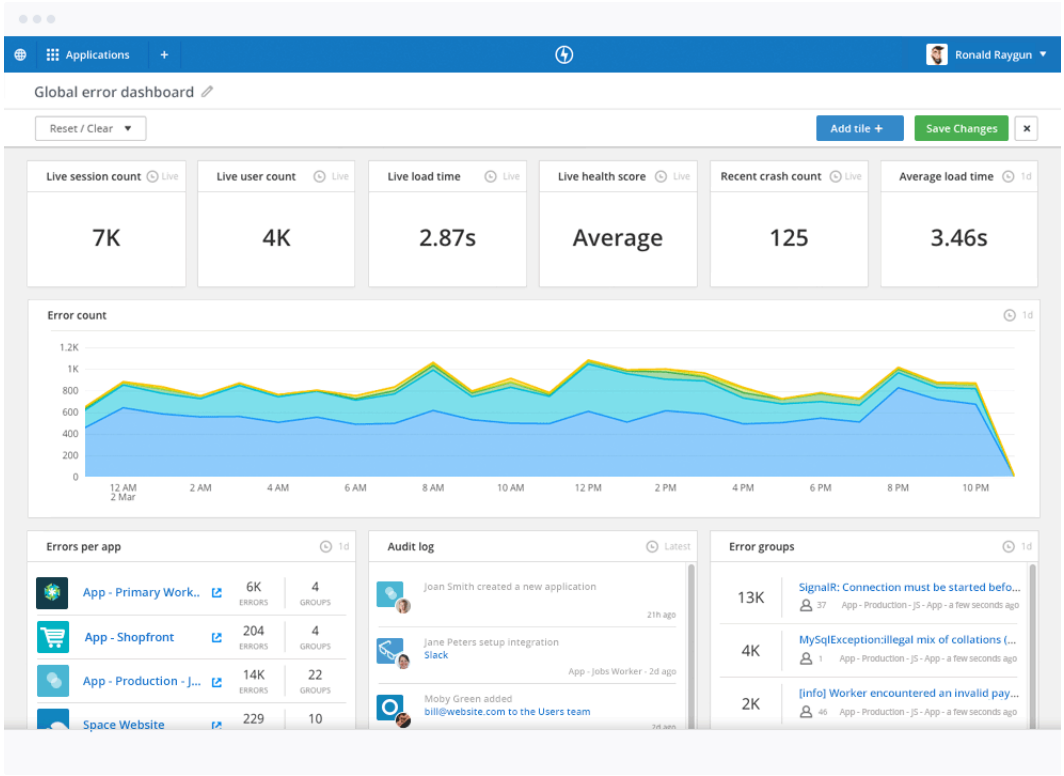
With these solutions, teams may be tied in to costly agreements that cover a specific number of servers with agents installed upon them. When they need more servers, languages and platforms to be monitored, this substantially increases the price. Fortunately, a software intelligence platform like Raygun does not operate on the same restrictions, allowing you to monitor all corners of your application's setup.

More than numbers on a dashboard

Sure, software teams are able to plug in various development tools to check the effectiveness of their software. It's not difficult to grab a couple of monitoring tools off the shelf, implement and view trend data over time for things such as server performance, request times, and error rates. Then stick the data on a pretty dashboard next to the office water cooler.

But we've forgotten something here, something important to all this - our customers.

Customers don't care about averages or medians that may be trending on your dashboards. They care about the experience they had and whether it was good or bad. Many teams just can't see the individual customers behind the numbers.



Happy that you reduced your Javascript error rate by 25 percent? Seeing the total number of events go down on your company dashboard? Good for you! But sorry to burst your bubble here, as there are thousands of users that are still experiencing a problem. That's nothing to celebrate.

Instead, software development teams are waking up to a new era of software monitoring tools that give them a much deeper look into how their applications are performing for users - with the ability to see exactly which specific users were affected by an issue and why.

Rather than influence trends, developers now have the ability to identify, diagnose, and fix every bug and performance issue in their software applications, with complete clarity.

This is the power that software intelligence brings to your organization, giving you a view of your entire application from 30,000 feet right down to a specific user and the specific line of code an error occurred on.

Finding bugs before your users do

Every software team is striving to deliver applications that perform for their customers and maintain high Net Promoter Scores (NPS). The combination of devices, browsers, operating systems and Internet speeds gives many users a unique experience when interacting with software applications.

Even the most rigorous testing and QA process can't stop all issues from falling through the gaps into production.

A dedicated software intelligence platform gives teams the ability to diagnose issues customers are experiencing with supreme accuracy, without the need for a single customer to even report that they were affected.

That's right, *before* they were affected.

In a typical scenario with these tools in place, when a user has taken actions that have resulted in an unhandled exception, the issue can be identified and brought to a developer's attention within seconds. Detailed diagnostic details are then made available to the developer in order to replicate the issue quickly, fix it, and deploy the update to production. The issue is then resolved before it affects any further users, who would have run into the same problem.

However it's not always a user that triggers the error or crash to become known to the developer. Database timeouts, or poorly executed code can trigger errors in backend systems, throwing an error that is picked up by the monitoring tool. When releasing new code, issues that have been introduced can be found using deployment tracking functionality. Developers can then fix up these issues before any users are affected.

In either case, issues were not reported by any of the users experiencing them. Rather the monitoring was sophisticated enough to show the exact cause of a problem and which users it affected.

Performance issues are also a hard thing to monitor. Although the page might have taken a long time to load, it did get there eventually. Not many customers would actively report a performance issue in this instance. It wouldn't throw an error that would be captured and a notification sent to the developer. But it would make the experience the user had with your application an unpleasant one. As much as we would all like flawless and blazingly fast applications, the reality means we don't always get what we want.

So how do we ensure that users are experiencing good load times and not closing their browsers if our page takes more than three seconds to load? Well, you can't maintain what you don't measure. So a dedicated software intelligence platform has a purpose to alert you when your site is experiencing slow load times, and what is causing them.

It could be a large image file that is not compressed, or a script that is taking a long time to load, but all of these issues will be raised into the line of sight of your development team, allowing them to fix software faults with the minimum amount of damage caused to end users as possible.

Why a full stack monitoring tool matters

Another huge benefit of a software intelligence platform is the ability to have all your data available within a single monitoring tool. That means you'll be able to spot problems that are linked between say, your database and your iOS app.

When teams are set up badly in the first place, they may have no collaboration or visibility between Development and Ops. Though the rise of DevOps is an ever increasing trend in the industry, data is still often siloed and segregated from each other.

When an iOS application is monitored with a different tool than the company's Android app and again another tool for the production database, it becomes impossible to decipher why an issue is happening if it is connected to all three touch points. Even worse, if all the teams are also compartmentalised and not collaborative in their work.

Accounts departments might shudder at the thought of dealing with multiple vendors, but developers have been used to pulling together the best in breed tools to monitor each part of their technical stack. That makes a lot of sense, but in practice the limitations are obvious when a user session is buried within an anonymous Google Analytics session, the crash report in the iOS monitoring tool and error details are scarce.

Software intelligence tools bring all of this information together, so when an iOS app crashes, the entire user session is also made available. Errors and crashes that are linked to databases and other parts of the stack are also placed at developer's fingertips. With a single click this information can be accessed by anyone on the team.

Software intelligence breaks down the walls where valuable information is lost between teams or monitoring tools that only cover a small part of the full stack. Bring it all under one roof and the insights you gain will be more powerful than anything you've ever seen before.

The future of APM

Software intelligence is emerging as the next generation of application monitoring and it will change the way you build and maintain software. Giving teams insight into where customers are having poor user experiences across both web and mobile, with greater focus and accuracy.

With these new monitoring innovations, developers gain unrivaled insights into their applications with full visibility across their entire technology stack. Each and every bug or performance problem within an organization's software application can be brought into the view of the entire developer team with pin-point accuracy.

Giving actionable feedback in real-time to software development teams has now become the baseline, not the exception.

Every software team is striving to deliver applications that 'wow' their customers. Improved software quality is not something that happens organically, as teams must have the right tools and culture around fixing software problems.

Software intelligence paves the way for creating world beating software, delighting customers through flawless software experiences.

Surely every software team should be embracing that.

SECTION 3

So what exactly is software intelligence and why should I care?

Overview

Software development teams are often blown away by how much software intelligence benefits their business and how it transforms their software efforts.

> "I recommend Raygun because it creates a single point of transparency around faults in our systems, how we're managing those and working to reduce them."
>
> Karl Bunch, CTO, iSocket , Read Case Study

Software intelligence captures data about all these domains and how they overlap to create flawless customer experiences.

- Errors
- Performance
- Deployments
- Users

Generally speaking, only 1% of customers will ever report an issue.
This is why integrated software intelligence is more valuable than relying on feedback. You need to capture everything to understand software health.

Most businesses are spending significantly in marketing to attract customers. These users come to your site or app, and see if you deliver value to them. If your app is slow, or crashes, that user leaves. They're not sticking around to endure your lousy experience when competitors are just a single click away. In turn, software intelligence benefits your development team by reducing the cost to acquire customers and keep them happy.

Software engineers are expensive. When your customers have issues, are your engineers spending significant time trying to find log files? More time reproducing errors? Claiming it "Works on my machine" perhaps?

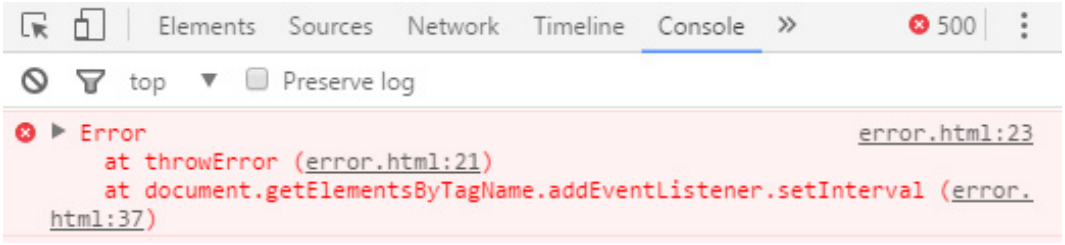
When you're working towards an ambitious roadmap, you need the team delivering features, not fixing old bugs. Wouldn't you rather have your team delivering features that your customers will love?

Using a software intelligence platform, stack trace information can be pulled in automatically to make debugging a breeze.

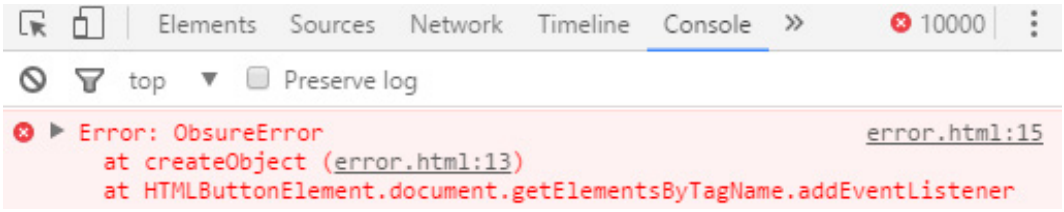
Software intelligence platforms provide deep insight into what has happened, with enough context to solve the problem. This makes your team far more productive.

Things do go wrong and you need to allocate developer effort to those issues. Let me pose a question:

What's a high priority? A bug that's occurred 500 times?



Or 10,000 times?



The correct answer is "It depends. Which one affected more users?"

If it was 10,000 errors impacting one customer, then it's not that big of a deal. If it's 500 errors affecting 250 customers, then it's a pretty big deal.

Amazon is famous for always measuring 'customer impact' of any outage or issue, because it's what matters.

Your software intelligence platform should make it immediately visible to management what issues to prioritize:

Message	Last seen	Count	Users
document.width is not a function - test	5 hours ago	15,348	2,155
Script error in 3rd party script	3 days ago	1,403	185

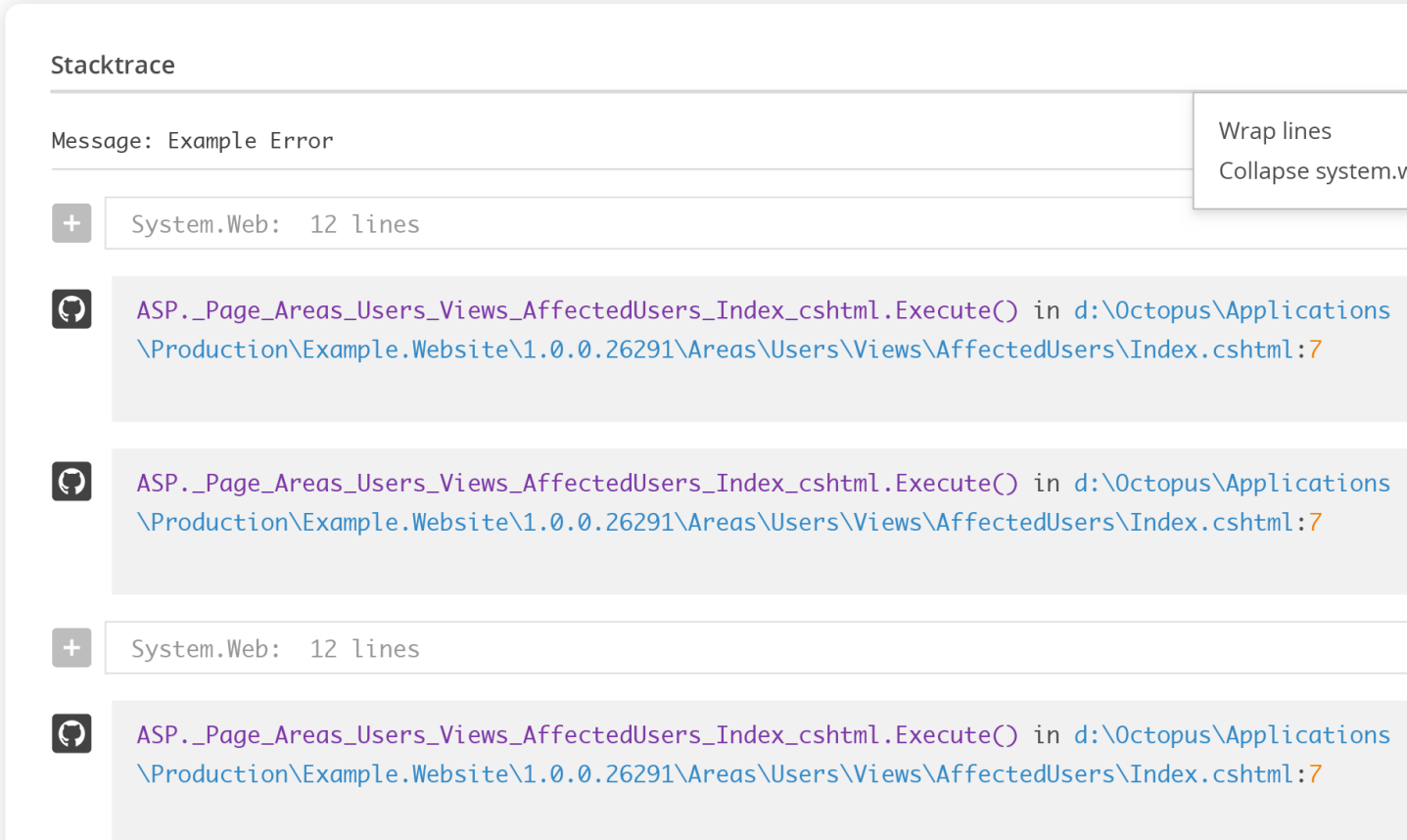
Raygun completes the CI/CD circuit by providing real-time feedback on software performance and errors to your team.

Even with a dedicated QA team, issues will slip through. This is where you need a safety net to ensure that issues that slipped through are discovered quickly.

If you have invested in deployment tooling, then the effort for a team member to deploy is low. Once they can trust that issues are identified, and they do not need to go hunting for why the issues occurred, they can deploy fixes quickly. This makes your software team truly agile.

User tracking: Identifying affected users

I had an email from a company recently that apologized profusely for their recent period of downtime. They were so sorry that they had let me down. They were going to try harder in future and hoped that this kind of incident wouldn't happen again. The issue was however that I had no knowledge of this outage, I wasn't using the tool at the time it occurred, so if they had not told me of their outage, I wouldn't have known otherwise.



Although I can commend this company for being so transparent about the outage, all it did was damage my view of the reliability of their service and confidence in their brand. Suddenly my views of them had been lowered, and completely unnecessarily. In today's data driven world, there is honestly no need for this to happen.

Monitoring your applications with a software intelligence platform or having customer analytics services attached can have many associated benefits rather than just what they are fundamentally designed for.

Notably, by monitoring affected users for specific errors and crashes or poor user experiences, this can allow you to contact only users that have been affected by a bug or underlying issue. Tools that can show users that logged in between specific time periods can also be used to build segments and cohorts of users that have been affected.

By pulling out these segments of users and then those emails handed off to marketing or customer support, we need only communicate recent outages to those that were impacted, saving us having to notify our entire user base and risk a wider loss of confidence than is necessary.

Affected user tracking also allows you to view whether your software is getting better (or worse) with greater clarity. Should you release a new version and see thousands of errors hit you company's monitoring dashboard, you'd be forgiven for panicking and why total error counts have increased, leading to questions from your senior colleagues.

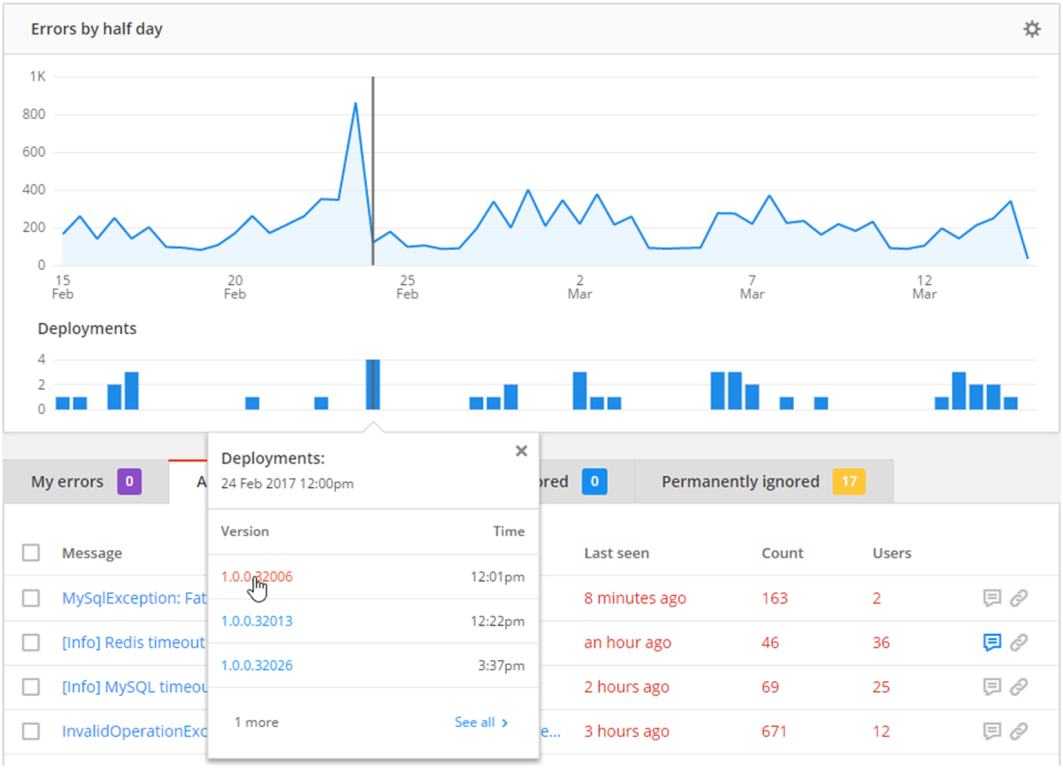
If however this release had a very specific bug that only affected people on a particular browser or operating system, your software intelligence platform would identify this is the case. The issue could be affecting only a small percentage of your overall user base, but you'd previously lacked the clarity to see this the moment your software went into the hands of users.

Software intelligence redefines the level of data insight you can obtain from your application monitoring tools.

Deployment tracking: Identifying problematic deployments

In a recent survey on DevOps and continuous delivery (which you can download here), DZone found that the deployment process was the biggest pain point for 29% of software professionals using the continuous delivery model. It doesn't have to be this way, if using deployment tracking to create a reliable workflow surrounding errors and deployments.

Deployment tracking gives you the peace of mind that problematic deployments are picked up immediately – including the commits that made them. Using this feature, you can track deployments and triage errors when you deploy any new code. See how we use the continuous deployment model and deploy safely and efficiently six time a day.



Deployment tracking takes the pain out of deployments and lets you identify:

- Which errors have been introduced with the new release
- Which errors are still occurring
- Which errors that have been fixed
- Which previously resolved errors have reoccurred

A software intelligence platform allows you to ship with the confidence that any errors will be found well before your end users experience them. View a practical example of this here.

SECTION 4

Presenting a business case for software intelligence

“We need to make performance improvements” is a generic statement and will not get you very far when requesting budget. Especially when hurdles like performance benchmarks and resource management present themselves.

However, if you can build an argument around how even minor performance improvements can drive growth, you may find yourself the hero of your department.

It’s no secret that addressing performance issues is one of the easiest ways of improving the stickiness of your app. The ‘ship fast and break things’ agile mentality is becoming old and tired - it’s simply unable keep up with the vast research that hail a user’s experience key to managing business metrics like churn rate.

For example, Pinterest is a Fortune 500 company that cites 2016 as their “biggest user acquisition win.” Interestingly, they directly attribute their results to discovering and fixing performance issues like user perceived wait time.

When Pinterest dedicated time and budget to rewriting pages with performance in mind first, they experienced a 40 percent decrease in Pinner wait time, a 15 percent increase in SEO traffic and a 15 percent increase in conversion rate to signup:

Because of the results of this project, Pinterest's engineering team now focus on software performance as one of the biggest opportunities for growth.

You could see the same improvements in your business.

But where can you start presenting a business case for improving performance?

It all starts with 'what gets measured, gets managed'.

Firstly, let's look at a practical application of how costly software errors can be to your business. Then we'll show you some metrics you can use to see how your business stacks up.

Labour costs

For presenting a business case for better software performance, it could be helpful to understand how much money fixing software errors and performance issues costs your company.

Developers and support teams spend a lot of their error resolution time digging through log files and attempting to replicate the issues. This is low value, reactive work.

On average, a developer may spend 32 hours per month discovering, triaging and resolving errors. On the average US software developer salary of $58,000 per year, this equates to costing you $889 per developer per month.

These hours are time away from proactively adding value to your applications. Reducing the time spent on error resolution by just 50% could save you thousands every month, while increasing the overall software quality of your application.

Note the above figures are based on the average salary for a software developer at $58,000 per year. In cities like Boston, Los Angeles, New York, San Francisco and Seattle, these salaries can triple. While reactive work is important, it rarely differentiates the business.

Having visibility and actionable data to reduce software errors can help shift the software development equation from 70% reactive, 30% proactive to 70% proactive, 30% reactive.

Proactive work – releasing new features and services – is where true value is added to the business and its users.

Business costs

How costly can software errors be? Consider a national pizza delivery company.

Assume the company's e-commerce site generates five million errors per month as a result of its software developers continually releasing new features and services.

If only one percent of those errors (50,000) prohibit customers from ordering, at an average of $25 per pizza, the potential impact in lost revenue is $1.25 million per month. Eliminating 50% of that one percent would save the company $625,000 per month.

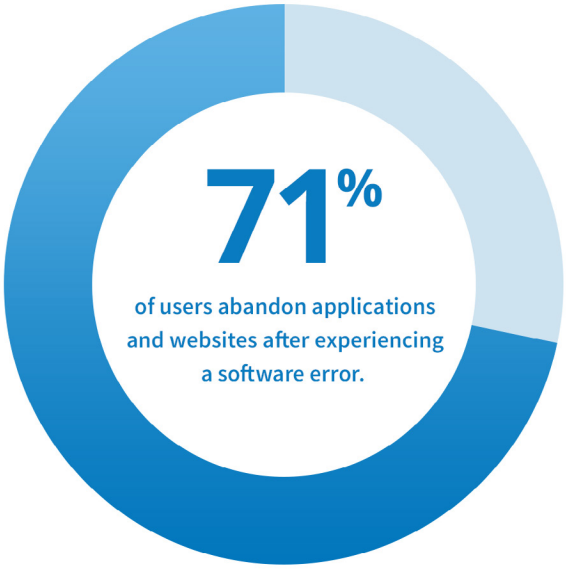
A common and immediate response from CTOs other technology professional is that five million errors per month is extremely high. Based on our data from 2,000 customers, five million errors per month is actually common, if not conservative.

Companies and their technology leaders are surprised how many errors their software has, even with the best quality and testing efforts in place. In fact, it is not uncommon to see applications and websites generate 25-75 million errors per month.

If you do not have visibility into these errors and their impact on users, you are at risk, never mind the potential upside you could receive from getting on top of errors and performance issues present in your application.

Giving a better customer experience

According to industry research, less than five percent of users let companies know when they experience software errors.



How can you fix problems when you do not even know about them?

You need visibility into your software. You need to understand how users are engaging with and experiencing your software after it is released, and you need actionable data to help you fix errors before they significantly impact users.

Measuring the impact of poor user experiences on your business

You can get an accurate picture of your application's health through these four metrics:

1. Median Application Response Time
2. P99 Application Response Time
3. Users affected by bugs
4. Resolved bugs vs. new bugs

Median Application Response Time

Forget averages, they lie. The Median response time is what 50% of your customers experience (or faster). Track that, and hold the team accountable to achieving a time or better. Performance makes money – 40% of users will leave a website that takes more than 3 seconds to load!

P99 Application Response Time

Medians are great, but you also need to appreciate the upper limit, which is catered for by tracking the P99 – the time taken for the 99th percentile of users. This will normally be slow, but make sure it's 5 seconds slow, not 25

seconds slow. Don't track P100 so often, as it's the domain of total timeouts, bad bots that hold connections open and is generally misleading about real users.

Users affected by bugs

Error counts are a misdirection. If you have 10,000 errors that affect one customer, it's not as bad as 500 errors affecting 250 customers. Measure the affected customers on a monthly basis, with a goal to reduce. With affected user tracking, you can make that missing connection between error and end user. Plus filter out any sensitive information for legal compliance.

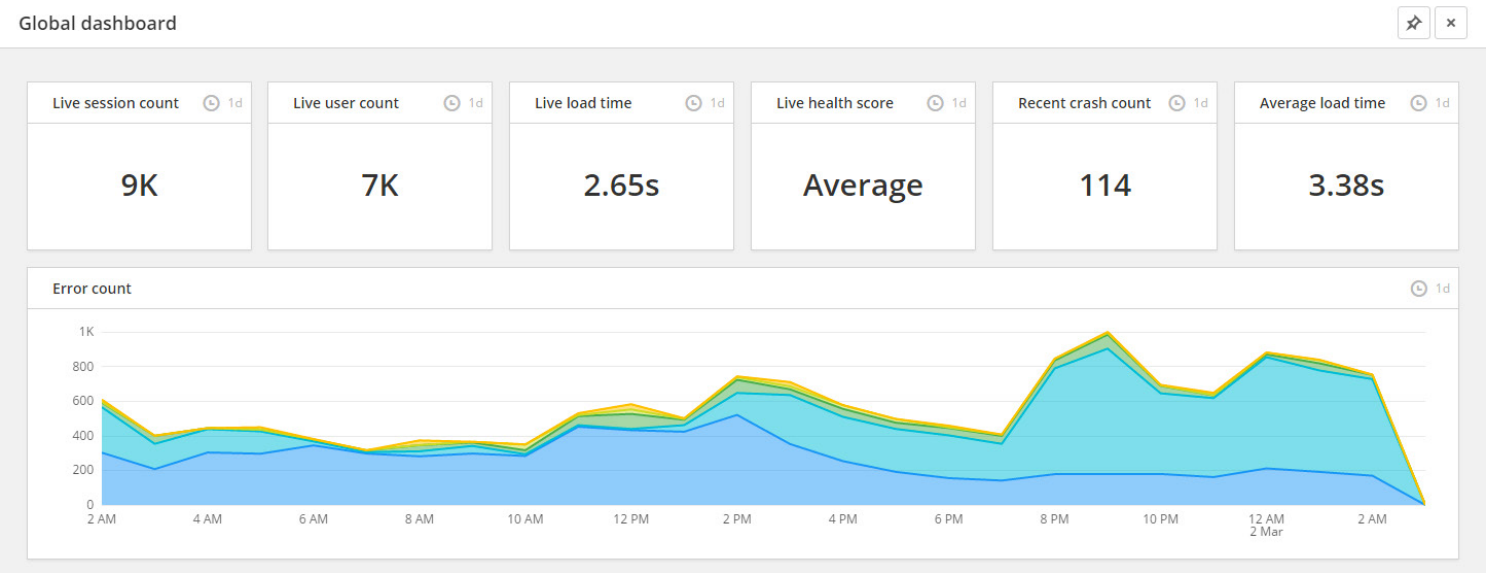
Resolved bugs >= New bugs

Software intelligence platforms will group bugs by their root cause. This makes it easy to manage the bug count, not the crash count (instances of a bug encountered). The team should be fixing errors at least as quickly as they are creating them. Ideally resolving more than are created.

These metrics provide benchmarks for measuring the impact performance has on your bottom line. Aim for incremental improvements over time, and you'll significantly improve user experience with less crashes, faster software and a reduction in technical debt.

Here at Raygun, we actually hold our senior team accountable to these metrics. We use the Raygun platform to track these metrics for us in real time so they stay front of mind. Raygun allows us to consolidate our metrics into one place and gives us a holistic view of our application performance.

We also made improvements to our Dashboard feature, which enables you to get your key software performance metrics up on a TV display for the whole team to see. The dashboard also gives tech leads the ability to take a quick glance at software performance without having to dive into the code base.



Now you know the impact poor performance can have on your business, and some benchmarks to measure where your software team can improve your end user experience.

Software intelligence helps you scale by automating your error resolution process and integrating with your current technology stack.

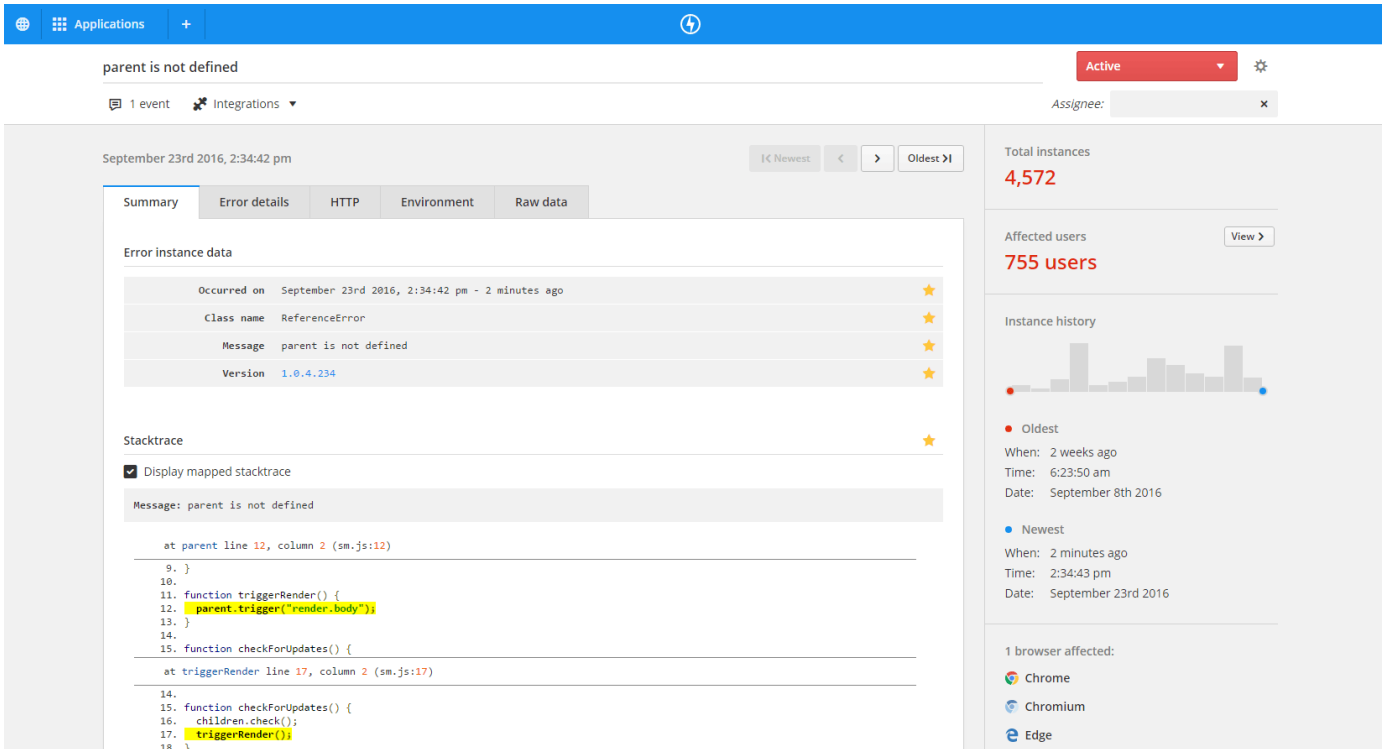
SECTION 5

Building a world class workflow

Traditional development workflows can fail your customers.

As your organization grows, processes and tools need to scale with you, not remain static and stagnant. Even workflows in Agile practices can be made more flexible. Automating your error resolution process is the least disruptive way of effectively managing outages, errors and bad deployments.

If you use a software intelligence platform like Raygun, you can use integrations and powerful features to help your software team reduce the human effort involved in your issue resolution process, saving time and money in the process.



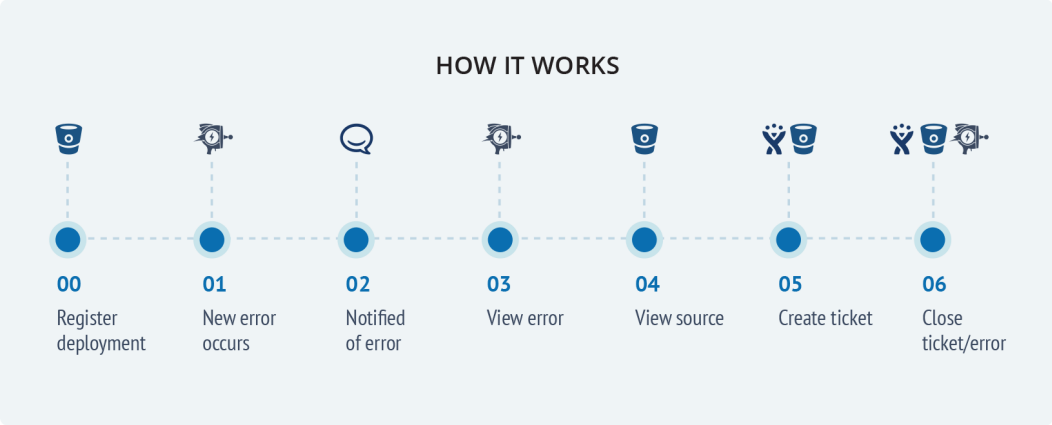
Too often, the diagnostic details your software team needs to find and fix errors are buried in log files. The stack trace gets swallowed by too much noise and it's almost impossible to isolate.

Raygun uses smart alerts via chatops and email to bring errors to your developer's attention, then points them straight to the stack trace.

As IT leaders spend less time in the code base, you'll want to make sure there is an error resolution process in place that doesn't require constant attention. This way, you can spend less time micro-managing and more time on growth projects.

Creating a highly tuned error fixing workflow isn't as complicated as you may think. It may even be a case of simplifying a few steps. Raygun fits in neatly with your technology stack and can help to automate the process with features like the 2-way sync with Slack and GitHub.

For example, here's how an error gets communicated:



Better visibility across your technology team

At the start of any new project, team members need to look at the 'whole pie' so to speak, then break things down into individual projects and tasks.

The overarching vision for the development of products is important, mainly to prioritize tasks, but also to keep the team on track and pointing in the right direction collaboratively. From initial concept right through to shipping day, each team member will be looking to contribute to the goal for the project. However, this isn't too bad when building software, but maintaining it is a different matter.

When teams share a more holistic view of their application, projects stay on track and run smoothly.

Integrating with your current workflow tools can give everyone a notification at the same time. So not only will the CTO be aware there is an issue in the front-end code, the front-end team can jump on it within a heartbeat. Automatic integrations with version control and issue tracking tools can also means manual replication of issues becomes a thing of the past.

Throwing code over the wall into QA

Some developers can spend months, or even years working on specific features or products and never see them ship. Or if they do ship, have no insights into how their work was received by customers.

Due to externalities, developers in large organisations can be asked to throw their hard work over the wall so to speak to the Quality Assurance (QA) department and software testing teams.

Robust software testing finds issues that could have affected customers should the code be placed into the hands of end users, never to be seen again unless it needs further tweaking.

QAs and testers act as the final guard between stakeholders and customers, and they can be petrified that users could be exposed to major bugs or outages should bad code be released into the wild, especially for mission critical software.

However once released, visibility to production issues rely on customers to report the issues they experience and for front line support staff to be the go-between of customer and developer.

The purpose of having a QA team with complex processes is to do as much as possible to prevent large bugs from hitting the user. Organizations are deathly afraid of those large, show stopping exceptions being present for users, and so they should be.

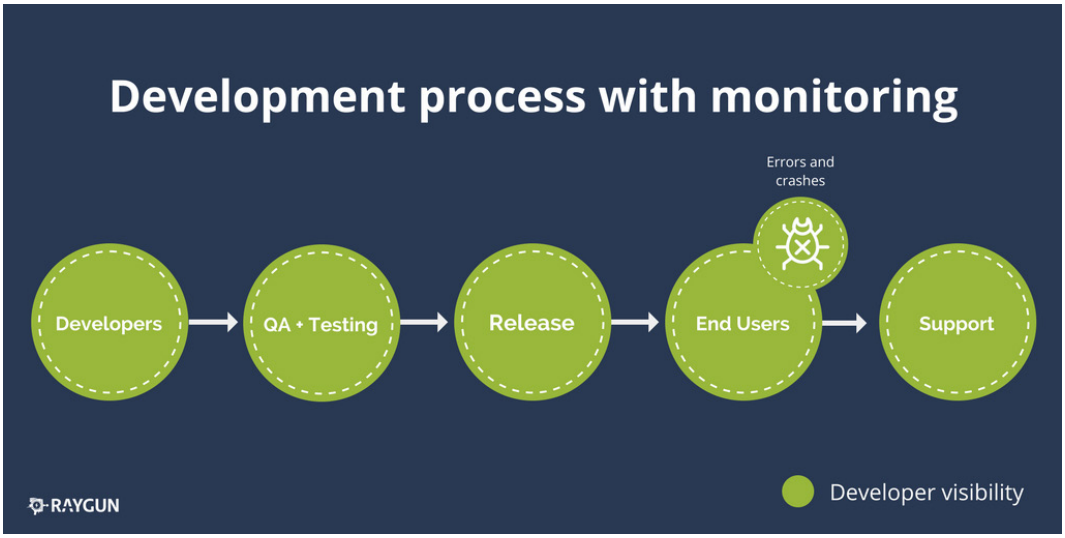
A bug identified and fixed in the conception or early design stages costs next to nothing, but bugs needing attention in production can cost considerably more. The challenge is to give the developer a view of errors and crashes caused by their own code. Though QA teams are effective, they're often not best placed to find hidden issues. Often the people who built it are the most qualified to debug it.

Giving developer teams better visibility of their code

Developers can create beautifully coded and complex features for users, but only ever get to see the results of their own colleagues internally, using what they built. There is a wall between them and the end user. This is how the process might look, with the Analysis, Design and Implementation phases of the Software Development Lifecycle compressed into 'Developers' below:

Utilizing a software intelligence platform, developers can instead have full visibility of their code whether it is in QA, software testing, production or end user's hands. How? Well, automated error monitoring and end user performance monitoring keeps a watchful eye on their software and when issues are found the entire team can be alerted immediately.

Rather than a wall in-front of them, developer teams now have a window into how users are interacting with their software and the features they build.

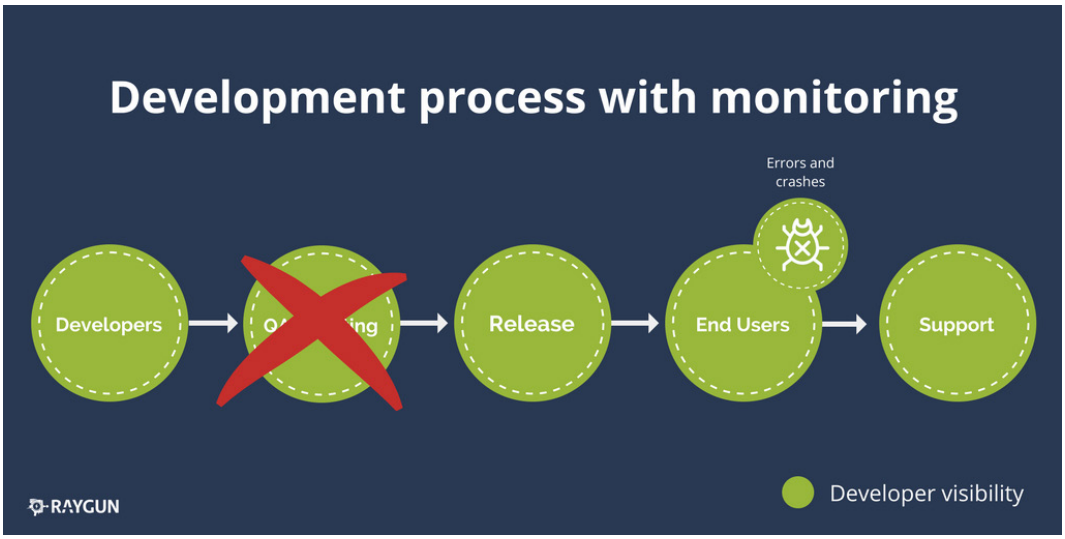


Deploying code safely without complex QA and software testing

For many teams that use automatic error monitoring tools, they've completely eliminated the need for thorough QA and software testing of the developer's code before it goes into production. Now this doesn't mean they do not continue to do thorough unit, integration and acceptance tests of their code at the development stage and have the wider developer team perform code and peer reviews, but they no longer need to have a strict, inflexible QA process in place, or if they do implement one, it is pushed back till the organization is much, much larger, and only for critical processes.

This is due to an automated software intelligence tool being the guardian from bad code causing issues for users, rather than relying on humans to run test after test to ensure code is safe to deploy and for customers to use.

You'll also see here that there is now a clear line of sight between developers and application, errors, crashes and performance issues. If we were to roll out a bug into production that caused an unhandled exception that affects



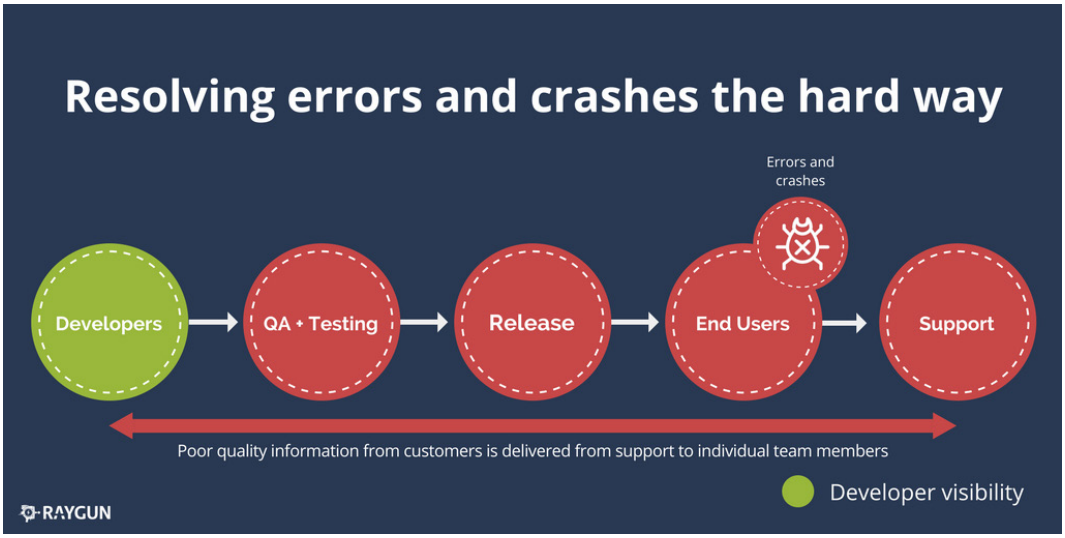
and is seen by many users, the entire team can be notified of the issue within a few seconds, with diagnostic details providing enough information to fix it quickly.

Again, without the need for long release cycles, software testing and QA processes, code can be shipped on smaller timescales (hours or days) and the issue fixed before any other users encounter the same problem. There is a huge safety net in place should we push bad code into production.

Resolving errors and crashes for users the easy way

Here's how this also breaks down for support issues specifically. In the traditional model when a customer reports an issue the support team will have to take this largely non-technical information to a specific individual on the developer team who is assigned to fix the problem.

Back and forth information ensues, which is often unhelpful for the developer, as they want information such as the user's browser, operating system, error messages seen, page it happened on, time of day etc so they can check the logs. Why are you putting the biggest effort upon your customers here? Taking time out of their day to explain problems you caused them?

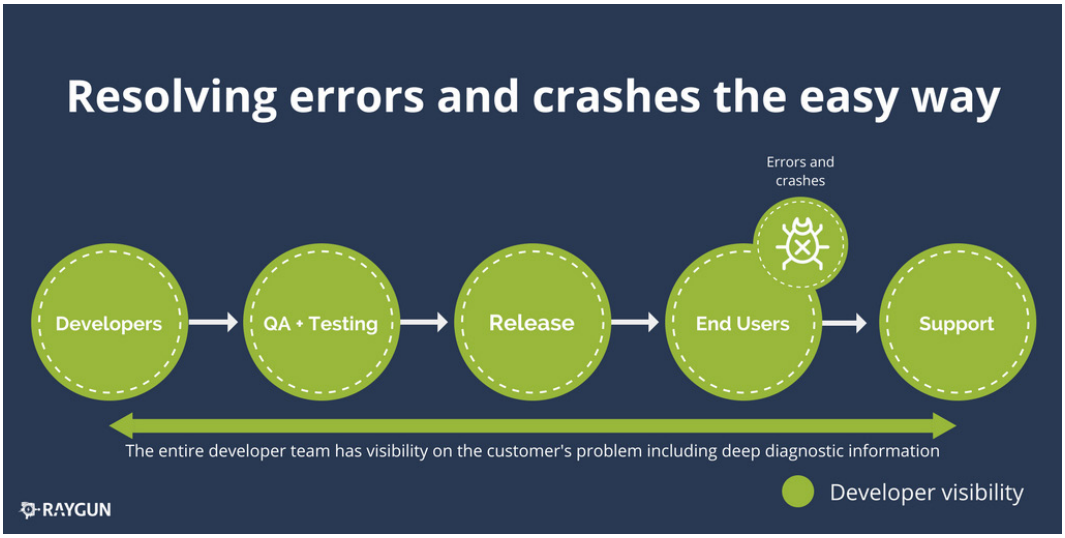


How often has your support team struggled to replicate and/or fix a software issue that a user encountered, but spent hours trying to debug it, digging through log files and going on pasted screenshots in Word documents from a now disgruntled and angry user?

Front line support staff are usually the ones dealing with the customer, and information is passed to the developer team inefficiently and often third hand, as the customer tries to explain which page, step or action they encountered the issue.

Our modern software development model with software intelligence in production environments have issues reported automatically, without the need for users to report them. The development team has full visibility on issues. Should support teams have a customer who has reported an issue, a simple search of their email address can bring up the diagnostic information about the exact error they encountered, including the stack trace and environment data that can be the key to a fast bug fix.

It's so, so easy.



Developers who implement software intelligence with their applications are also empowered to care more about how their code affects end users, because they get the insight into the issues they are encountering and faster feedback loops.

Increased visibility using plugins and third party integrations

DevOps is about knocking down barriers between developers and operations teams, and in doing so reducing organizational friction wherever possible. That's certainly the spirit behind ChatOps – the term for the hyper-collaborative way of running DevOps, including operating aspects of systems and infrastructure, through online chat.

If we are to understand the benefits of ChatOps, it's worth understanding a bit more about it's relationship with DevOps. Although still a relatively new concept, DevOps (Development and Operations) has become a hugely recognized term in the software delivery world. DevOps has the aim of creating greater efficiency in an organization through the four cornerstones of culture, automation, measurement and sharing.

By creating a culture where teams strive to automate tasks, measure and analyze everything they can with their fellow team members, putting this inside a chat program seems like a no-brainer, embodying all of the methodologies that DevOps aims to achieve. Teams get all the benefits of DevOps practices, packaged up into a collaborative tool. ChatOps therefore may have stemmed from the DevOps movement, but the practices go hand in hand for software development teams.

Born within GitHub, ChatOps evolved from its open source chat bot, Hubot, which has changed the way the web-based Git repository runs operations through the automation of deployment, graphing, monitoring, provisioning, and even mitigating security events.

This new way of working is empowering teams to become more productive, through conversation-driven collaboration.

Alerts are just the beginning

It's one thing to be alerted to an issue, that's just the notification part, but without the diagnostic details about the problem, we can spend a whole bunch of time just trying to work out the cause whilst our users continue to see problems, support queues start to fill up and the issue grows into a major company wide incident.

If we can get this information immediately at the time the error notification comes through, fixing and debugging can be done at record speed and your whole company can benefit, as well as your end users.

Many developers may automatically assume this only extends to servers falling over, database timeouts and more, but it can also include production errors your users are encountering everyday. If you knew that a recent update has broken your billing system and customers couldn't pay you, surely you'd be scrambling to fix that pretty quickly!

If you already use team chat applications in your business, you might already have several pieces needed to get an awesome issue management workflow operational, connecting these pieces together can create a world-class incident management workflow and ensure our entire team is following along from first alert to resolution.

Reaching issue management heaven

How do we use ChatOps to create a faster, easier and more transparent workflow around issue management and fixes? After all, most teams will have #war-room or #devops team room set up in their ChatOps solution. If you don't, maybe it's time to create one? But the conversation can often be one of confusion, blame or panic when incidents occur.

Here are the steps that will ensure issues are dealt with in an effective manner, no matter what time of the day or night your team is having to deal with them.

With a software intelligence platform keeping a watchful eye on your applications, contextual information about the problem is made available to your entire team instantly, including which specific users of your application have been affected.

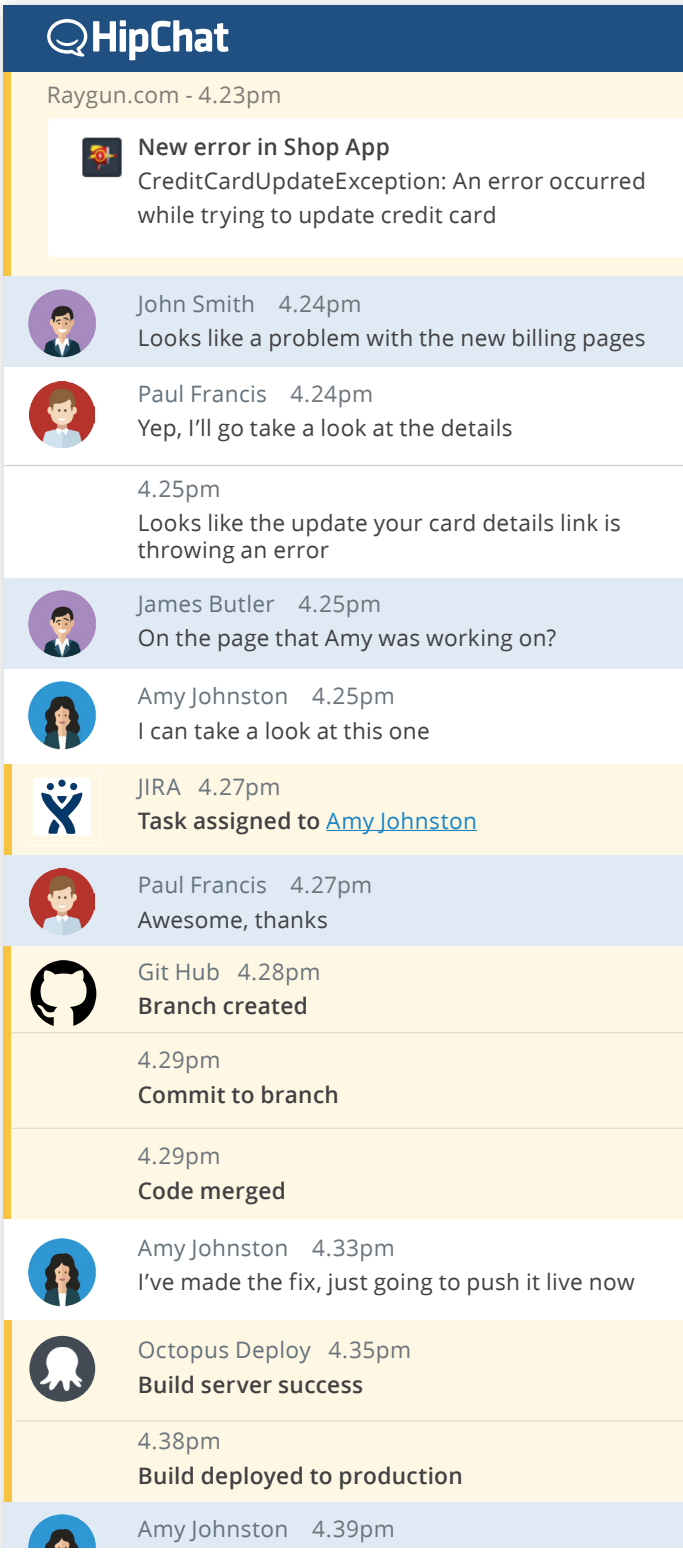
Should the inevitable happen and users are exposed to outages, bugs, errors, crashes or bad deploys, your entire team will know about it instantly via smart email alerts or team chat notifications, with all the diagnostic information you need to fix the problem quickly and efficiently.

When everyone has visibility of issues affecting your applications, communication becomes seamless.

Issues can be created in project management tools like JIRA and priority levels set with a click of a button. If we have a plugin enabled we can also send through JIRA activity into our team's HipChat room.

We can make then make the necessary changes in our codebase. As our software intelligence platform has given us the diagnostic information, right down to the line of code the issue stemmed from, finding out exactly what was causing the problem can take a few minutes. If you've ever had a major incident with time pressures to put out a fix, you're going to be thankful you had this kind of solution in place.

As we deploy the code to production, our automated bot can manage the build and deployment process. You'll notice here that there is no Quality Assurance (QA) step involved. Utilizing this workflow, if the issue re-occurs or causes further problems, we will still be listening out for errors in production, effectively removing the need for a QA step.



We've now gone from discovery of a software issue to a resolution, with our entire team being given full visibility of the situation. ChatOps made this whole process easy, transparent and stress-free.

Software Intelligence works alongside all of these services and more:

SECTION 6

Summary

The world of software development can be a messy one - If you don't have the right tools in place. Developers scrambling to fix issues in the company's codebase due to lack of information, confusion over technical support tickets and poor communication over who is watching the application for the inevitable problems that crop up, is a recipe for disaster.

All of this work adds a huge amount of human effort and wasted time to your organization.

Streamlining your process around issue discovery and subsequent resolution might seem like a huge task. Too big and scary to implement without a large budget and timeframe.

Getting started with software intelligence is easy.

All it takes is a few moments of your time to get started and the addition of a few lines of code to your application. Ship this into production and into the hands of users and watch, as you're automatically guided straight to the biggest problems affecting your users, right now.

There is little excuse. What could be more simple?

As we have covered in this ebook, many developers underestimate just how many problems exist in their applications that they don't even know about. Plugging in a software intelligence platform brings these issues into view for the entire software development team.

Software intelligence will improve your software. It'll make your customers happier, your NPS scores will increase. Your boss and peers will be happier knowing you are taking software quality seriously, resolving issues with greater speed and accuracy. There are no downsides.

If you still don't believe that software intelligence will radically improve your organization's software, give it a try or request a demo. You have nothing to lose and everything to gain.

Take a free trial

Try Raygun free for 14 days!

Book a demo

Take a few minutes to see the magic!