Editor: **Robert Blumen**
SalesForce
robert@robertblumen.com

James Smith on Software Bugs and Quality

Priyanka Raghavan

From the Editor

James Smith of Bugsnag discusses software bugs and quality. Host Priyanka Raghavan spoke with Smith on topics including causes, types, and history of bugs; user experience and environments causing different bugs; and measuring, benchmarking, and fixing bugs based on data. We provide summary excerpts below; to hear the full interview, visit http://www.se-radio.net or access our archives via RSS at http://feeds.feedburner.com/se-radio.—*Robert Blumen*

Priyanka Raghavan: Why is it okay to ship software with bugs?

James Smith: Although you should reduce bugs as much as possible before you ship, it's a tradeoff. To be competitive, you might want to deliver features or products to customers more quickly. Most importantly, you can't fully prevent bugs: you can't test every single experience customers have.

Do you see more bugs in certain languages?

Yes. JavaScript 100%. It's easy to use and is a lot of people's first language. Many junior developers pick it up and introduce bugs. It's important to understand the fundamentals of typing even in a language like JavaScript, where it's magically typed behind the scenes.

Will a particular type of architecture or design pattern have more bugs?

The smaller the scope of your project and the better the contracts between your project and others, the less likely it is to have complicated, confusing bugs. This has been highlighted by the rise of microservices architectures. When an app does one thing and owns the data, you can anticipate problems that could arise more easily because you're not trying to map a complex state machine in your head. Microservices with contracts between services and applications force you to document and think about the relationship between these applications and about errors that could occur and cause the contract with other services to break down.

The contrary, interactive user interfaces, are most likely to have bugs. You're building something that people interact with in different, sometimes unanticipated, ways. Also, there's a ton of asynchronous code running. Most of your code in a UI, web, desktop, or mobile app is running in callbacks, waiting for someone to interact with your application. An exception in a callback doesn't kill execution for the rest of the application, it just causes that callback to fail. So for the customer, the whole application keeps working, but just your callback or your click handler might break.

How do you handle bugs coming from third-party libraries?

Digital Object Identifier 10.1109/MS.2021.3058704
Date of current version: 16 April 2021

Fortunately, most people are using open source third-party libraries these days. You shouldn't have bugs in third-party code, but you will. If it's open source, at least you can go and dig into it. If you're using an error-reporting, error-monitoring solution, it will show you the stack trace, the line of code that caused the crash, and all of the other code paths that the customer went through before the crash.

You shouldn't live on the bleeding edge. It's exciting to get hot new features, but you shouldn't immediately bump your dependencies as soon as something new comes out in beta. Selection of third-party libraries is an underrated part of software development. If you rely on something, you need to trust it, so researching third-party libraries and SDKs is critical and underrated.

SOFTWARE ENGINEERING RADIO

Visit www.se-radio.net to listen to these and other insightful hour-long podcasts.

RECENT EPISODES

- 446—Nigel Poulton, author of *The Kubernetes Book* and *Docker Deep Dive*, discusses Kubernetes fundamentals, why Kubernetes is gaining so much momentum, and why it's considered "the" Cloud OS with host Gavin Henry.
- 445—Host Justin Beyer spoke with Thomas Graf, cofounder of Cilium, to discuss eBPF and how it can be leveraged to improve kernel-level visibility.
- 444—Host Akshay and Tug Grall of Redis Labs discusses Redis, its evolution over the years, and the emerging use cases today.

UPCOMING EPISODES

- Host Adam Conrad talks with Dan Moore about build versus buy.
- Luke Kysow discusses service mesh with host Priyanka Raghavan.
- Host Jeremy Jung talks with Scott Hanselman about the Microsoft .NET framework.

Can some classes of bugs be found only by actual users in the field?

Huge teams used to work for months on QA going through QA scripts. The more we've gotten to lean, agile, rapid iteration, and being able to hot fix and patch things and componentize software, the faster we can ship. You can't now have a team of humans do two months of QA.

The left-hand side of software development has been replaced with what Capital One calls "team quality engineering"—trying to automate that as much as possible. From the right-hand side, you have data-driven instrumentation, with products that will tell you, "This is a problem, this is how many customers it affected, and this is how you fix it."

Bugs exist in the hands of customers, where data representing that user has gotten into a strange state. Preproduction and precustomer testing include unit and integration tests and cleanroom environments. But in reality, customers run software in a dirty environment because it needs to do things such as save state, cache, and authenticate the customer.

Most bugs are not due to code paths being missed, because there is usually good code coverage in testing. Most are about weird data structures and unclean data coming through. The problems that happen in the hands of your customers come from caching, authentication, cookies, local storage, and stuff that's stored on the device that is not in the format you expect.

Is it okay to delegate fighting bugs to our clients who paid money for software?

I think "test in production" is the wrong way of thinking about things. You have to be intentional about tradeoffs. We want to use tooling and technology to remediate bugs as quickly as possible. If resources are scarce, as they always are, fix the bugs that matter the most. That will vary by company and product.

You care most about bugs that affect key customers or that are happening in an important flow, such as login or payments. If it's a consumer mobile application that doesn't have people spending a lot of money, focus your time on bugs affecting the highest volume of customers. Whatever metric you use, be thoughtful about it, and use data to drive it. Then you can deliver software that's as stable as if you did a two-month QA process, and in fact improve it and get features to market more quickly.

There are great tools out there that support a data-driven approach to prioritizing and fixing bugs. But

ABOUT THE AUTHOR

PRIYANKA RAGHAVAN is a security architect at Maersk, Bangalore, 560077, India, where she makes day-to-day architectural decisions between security and usability or performance. She has more than 15 years in the software industry playing various roles from developer, team lead, software architect, and now security architect across three continents. She can be reached on Twitter @Priyankarags. Further information about her can be found at https://www.linkedin.com/in/priyanka-raghavan-6a1a405/. Contact her at priyankaraghavan@gmail.com.

even if you do this yourself, don't wait for customers to complain. By the time customers have complained, probably 50 other customers have already abandoned your product.

How do you use a stability score?

We want to understand what percentage of user interactions with your product are good or have failed. We will detect if there are unhandled exceptions, unhandled promise rejections, or exceptions in a callback. Or you may be using a framework that detects errors that cross an error boundary. So we build these hooks to failure states in your product.

These will cause your customers to have a bad experience. We don't magically stop bugs from happening, but we detect when they do happen. That all then feeds into the stability score. If one of those scenarios happens in your session, that counts as a failed session. The underlying concept is, I want to know which customers, what percentage of the customer base, had a positive experience.

IEEE Software (ISSN 0740-7459) is published bimonthly by the IEEE Computer Society. IEEE headquarters: Three Park Ave., 17th Floor, New York, NY 10016-5997. IEEE Computer Society Publications Office: 10662 Los Vaqueros Cir., Los Alamitos, CA 90720; +1 714 821 8380; fax +1 714 821 4010. IEEE Computer Society headquarters: 2001 L St., Ste. 700, Washington, DC 20036. Subscribe to *IEEE Software* by visiting www.computer.org/software.

Postmaster: Send undelivered copies and address changes to *IEEE Software*, Membership Processing Dept., IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854-4141. Periodicals Postage Paid at New York, NY, and at additional mailing offices. Canadian GST #125634188. Canada Post Publications Mail Agreement Number 40013885. Return undeliverable Canadian addresses to PO Box 122, Niagara Falls, ON L2E 6S8, Canada. Printed in the USA.