TOOLS

How Dev versus Ops became DevOps

Patrick Debois, founder of DevOpsDays, explains how the DevOps movement started and where it's headed.

by Java Magazine Staff

December 4, 2020

Download a PDF of this article

[*Java Magazine* initially published this interview in 2015. Patrick Debois was interviewed by Stephen Chin. —*Ed*.]

Widely known as the father of the developer operations (DevOps) movement for his work in bootstrapping the influential DevOpsDays conferences, Patrick Debois recounts the origins of the term, the concepts, and the subsequent growth of what is now standard practice at many large developer IT organizations.

Java Magazine: Tell us a little bit about how you came to found DevOpsDays.

Debois: Six or seven years ago, I was working on a project that involved agile development. Although I was a database admin, I really liked the flexibility of the agile mentality and tried to translate everything that agile development was doing into my systems work—such as faster feedback, testing at the

infrastructure level, and so forth. We called what we were doing at the time “agile system administration.”

Java Magazine: That’s not nearly as cool sounding as “DevOps.”

Debois: It was also narrowly focused, because it included only database and system administration. Then I went to the 2008 Agile Toronto conference and met Andrew Clay Shafer, a software developer who was already talking about agile infrastructure. We started the Agile System Administrator Google Group.

Later, I saw a talk by Jean-Paul Sergent about developer (Dev) teams and operations (Ops) teams working together. I didn’t really think about the term *DevOps* at that time, but I mentioned on Twitter that it would be cool to have a conference in Europe exploring some of these ideas. “Agile System Administrators” isn’t a very good name for a conference, so Sergent suggested we call it “DevOpsDays.” And that is how the DevOps movement started.

Java Magazine: The conference name came before the term *DevOps*?

Debois: Yes. We used Twitter to connect with about 60 people in Belgium. Two things were memorable about that conference. One was the energy of the conference people. Secondly, it was very international. Several attendees flew in from Australia and the United States. Others came from France, Germany, and the U.K. We started with just a couple of tweets about something that didn’t exist, and we had 60 people from all over the world come to that conference. After the conference, we continued our discussions using the hashtag #DevOps.

Java Magazine: Despite your initial Ops perspective, a lot of DevOps traction has come from developer interest. What attracts developers to DevOps?

Debois: I came from the admin side, but I always had an integration role. In the early days when Java didn’t have servlets, I created my own servlet loaders because I needed to have functioning websites. I had an affinity for developers because it always struck me that when you’re in the developer role, you often hate the operational constraints.

In discussions about infrastructure, admins would throw comments at you, such as, “No, it’s not the correct directory. You need to change the ports.” It’s really annoying that you can’t handle that yourself.

Java Magazine: Hence the need for Dev and Ops integration.

Debois: Before the DevOps movement, once your app was running, you didn’t have any insight into what else was going on. You couldn’t see the logins, you didn’t have access to the file system if something failed, and you couldn’t do debugging. You

couldn't see how things ran in production in order to avoid making the same mistakes. Those were the main drivers that led me to believe that all this needs to be more flexible.

Why can't Dev and Ops teams collaborate instead of hitting a brick wall? That was the incentive most developers I worked with had, and the one I think most people have.

If you're really passionate about what you're building and you want to improve things, you do want to go that extra mile, and you do want to have that information, and you do feel responsible for how your application is doing in production.

Java Magazine: Much of your focus has been on how people interact, as well as streamlining how developers and operations work together. In the past five years, there have been many advances in tooling associated with DevOps. Are these tools changing the way that people do development, or have they become too much of a crutch?

Debois: There is always pushback when you try to change culture, and a tool by itself can't change culture. What's good about a new tool or a new programming language is that it makes you think differently about your problem.

A good example of this is virtualization. In the beginning, people were using it in much the same way as when they didn't have virtualization. But once you start thinking deeply about virtualization as disposable and rebuildable environments, your concept of virtualization changes. One of the things that good tools can do is give you faster feedback on things that you or someone else is doing—feedback that can be captured via a simple chat system or as part of your testing system.

Tools that improve feedback also boost collaboration. You might want to collaborate, but if it takes a week to get a report about a problem, you feel less incentive to work on it. It's what's in your immediate attention span that keeps you going. The risk, of course, is if you're doing continuous integration or continuous delivery, you take it to a point where you're not getting feedback from your customers.

Java Magazine: That's pushing it to the extreme if you're not getting feedback from your customers.

Debois: That is an agile concept that was missing in practice. Ops teams were still finding anomalies and saying somebody else in the pipeline was responsible, which shouldn't be the case. Tools that get information to the developers, such as dashboards and monitoring systems, all help provide feedback and shorten the cycle. They support your process tremendously if you are willing to collaborate. Of course, if you're not willing to collaborate, it doesn't really matter what tool you use or don't use.

Java Magazine: How has DevOps changed the views of admin teams?

Debois: DevOps has changed the culture in the admin world. Admins used to be risk-averse. Their mantra used to be "We don't do this." Now, admins are saying "Let's do it more often if it's hard," "Let's see what happens when we try to automate," or "Let's feel the pain and then discuss things because now we're an equal stakeholder in the discussion."

For developers, who have always been at the center of feature discussions and product decisions, that culture was already there if they were doing agile development. But within an operations group or a systems group, this was not a common mentality.

Java Magazine: In the early days of extreme programming, we were pushing the development envelope and people were averse to making changes because they worried that more frequent releases would introduce more bugs. Now, it's completely shifted in the other direction to where most organizations are embracing agile development. It sounds as though DevOps is having the same kind of cultural effect on operations groups by making drastic changes in how applications are deployed.

Debois: This change comes partly from the fact that, in modern infrastructures, we're dealing less with hardware and more with an API or programmable infrastructure. So admins have evolved to thinking more programmatically about infrastructure, which is a natural fit with the current development process. Feedback cycles are often still too slow: Testing on Ubuntu and creating a VM [virtual machine] is much slower than doing something locally, although technologies such as Docker are improving the feedback cycle.

But it's complex. In general, developers assume the world doesn't change outside their apps, and admins assume the opposite: Admins own the apps and have to keep them running. Admins don't have the skills to change the apps, but they know the world is changing and so they deal with that part of the world that involves protecting the apps or making sure something is patched.

Java Magazine: Are there other disciplines where you can apply the best practices and experience you've gained from DevOps?

Debois: Interesting question. I deliberately made the decision of not working for another big technology company. I didn't want to be in the back room; I wanted to be in on product discussions. I realized that I could use all the technology knowledge I've garnered, and all the friends in the DevOps space, to make a difference in another domain.

I am currently in the broadcast video and live show space, which has interesting restrictions. One is the fact that you don't have

any users two minutes before the show, and then when the show is aired, you have tens or hundreds of thousands of users. Then after the show they're gone, so you don't have a test audience. You have only that one hour, so how do you do your testing?

It's like Formula One auto racing, where you go to a pit stop when you need a fix. I'm monitoring things in real time because, if it takes five minutes, your audience will miss five minutes of the show, which might be an issue.

Java Magazine: By that time, you've probably lost your audience.

Debois: Yes. There's technology complexity, but the real challenge is the content silo. The people who make the show's content are allowed to dream. I don't give them arguments on what's possible or not. I try to solve the issues. It's an interesting dialogue between content and producers, not unlike that between Dev and Ops. I see feedback during the show because we're on chat and I can monitor our business counters for how many hits or how many posts, so it is just another way of thinking about fast feedback and collaboration during a show.

Java Magazine: Could the feedback you're getting from video production and live audiences be brought into future DevOps discussions?

Debois: I used to do web development, where we had to scale up our back-end servers to monitor failures our end users were experiencing, which depended on all the different types of mobile phones they carried. The kind of testing I'm doing now is easy compared to that.

Mobile testing is hard, but testing is going to get much harder if the Internet of Things ever takes off. The sheer variety of devices, and the need to test all these networked things to keep them updated and humming, will make for a fascinating set of challenges and opportunities.

Dig deeper

- Video: Patrick Debois on DevOps for developers
- Book: *Continuous Delivery in Java*
- Tired of the "What is DevOps?" question?
- DevOps fundamentals for application developers
- Oracle DevOps services and tools

Java Magazine Staff

Java Magazine staff (javamag_us@oracle.com, @Oraclejavamag) deliver authoritative information about Java, the JVM, and JVM

languages to a community of more than one-quarter million developers.

Share this Page