split

Featuring

FORRESTER®

Q&A: THE EVOLUTION OF FEATURE MANAGEMENT AND EXPERIMENTATION

ABOUT THE CONVERSATION

For many product and engineering teams, feature management tools have become an integral part of the daily routine. By decoupling release from deploy, advanced feature flagging platforms are introducing new levels of speed, safety, and confidence to the software development process. While the foundational benefits of turning ON and OFF features with toggle switches might seem like old news to some, the "Feature Management and Experimentation" category, as Forrester describes it, is evolving as quickly as the systems they empower.

The Forrester New Wave™: Feature Management And Experimentation, Q2 2021 report included a comprehensive evaluation of the emerging category, but a lot has changed since the time of its publishing. As engineers elevate their standards, they're craving additional help from new technologies to decrease the potential of human error.

How are the top software teams leveraging feature management tools today? Recently, Split caught up with guest speaker **Chris Condo, Principal Analyst at Forrester,** to talk about what's changed from the earlier days of feature management. It was interesting to get Condo's perspective on the latest industry movements, including trends in trunk-based development, testing in production, and the introduction of causal analysis. Most notably, we agreed that measurement and learning capabilities, like experimentation and impact monitoring, are heating up!

Q

As far as adoption, what kind of companies are you now seeing adopt feature management and experimentation solutions? How important are these to engineering teams?

A

Chris Condo: If you're a cloud native company or born in the cloud company, feature management is probably something that's old news to you. But, for many companies in industries such as insurance, healthcare management, oil and gas, financial services, or other regulated industries (especially government), the idea of developers owning more and more of the stack is relatively new and maybe even relatively scary.

In order to speed up innovation cycles and to enable "Agile" to really take hold at your organization and be part of your product delivery cycle, developers need to expand their ownership further into the operational zone of software. Releasing software automatically and then getting feedback back from production to help developers understand what's going on is very important.

Q

How do feature management, trunk-based development and testing in production impact efficiency and psychological safety?

A

CC: When I was a developer in 2010, we had long-lived branches. I'd have a feature branch. My colleague would have a feature branch, there might be five or more different feature branches. None of it was getting merged to the main branch. Everything was being tested independently of each other. By the time we tried to hit our milestone, we went to integrate it and it turns out that nothing matched anymore. Something drifted, maybe it was the specs have drifted, the APIs had drifted or the functionalities drifted. And so then you end up having to spend time fixing that all over again just to integrate your code back into the main branch.

When you have feature management, you have short-lived branches. The idea is to get your code working. It builds. You put a flag on it. You stick it back in the main branch. It's going through the same test cycle as all the rest of the code. It's on the same train, not running all in separate directions.

Then when the code actually gets into production, it's there. It's not necessarily released to anyone just yet. A product manager can test features behind the scenes to make sure they work properly. They can do testing to make sure it meets certain performance metrics, KPIs, almost anything. New code can be deployed to production and be released to users in two separate steps, which makes things less scary for everyone.

> *New code can be deployed to production and be released to users in two separate steps, which makes things less scary for everyone.*

What do you think has changed the most since Forrester's initial New Wave Report (Feature Management & Experimentation, Q2 2021)?

CC: When we did our last Wave, one of the interesting things that came up was that most developers understood the value of a feature toggle or a feature management system. In fact, many of them stated it was mission critical to their operations.

Then we asked the same developers: "How important is experimentation to you?" The idea of tying measurement and learning to feature flags was a relatively new concept two years ago. It was on their radar, but not mission critical to their daily operations. It was instead seen as very aspirational.

That's changing and fast.

So interest in measurement and learning is on the rise. Who's interested and why?

CC: Today I get inquiries about experimentation quite often. And these aren't just 'born in the cloud' companies. These are banks and financial institutions, people at healthcare companies. It's because they've all just been through a pandemic. They've all been through a crisis where digital experience was proven to be the number one differentiator between them and those that didn't value it.

The spectrum of those engrained digital experiences has broadly expanded, and it requires next-level analysis and a scientific approach to deliver products that can appeal to a wider range of end users. I think it's really exciting that we're hitting this moment at this time, and I'm really excited to be here. I can't say it enough.

Q

Speaking of taking a scientific-approach, let's talk about "causal analysis." For us, it's how we know if each feature you roll out is making things better or worse, before it gets to all users. How do you see this approach helping the development teams you speak with?

For readers not familiar, here's how causal analysis works: Your gradual rollout exposes some users to the existing experience and some to the new experience. Then, a system watches metrics separately for each of those. If there's a performance or behavior difference between the existing and new, it will be detected and alerts can be thrown. When you do this for every metric, for every flag, during every rollout by default, the system is able to catch unexpected consequences without additional effort.

A

CC: As a result, direct impact from released software is measured, not guessed. You can put telemetry behind things to better understand the full picture. You can answer important questions like: "When I turn this on, did something change? Did it make things better or worse?"

And, if it does make things worse, no problem. Go back to the old version. If it makes things better, gradually replace the old version with the new one. It's much simpler this way. There's much less guesswork.

Q

So this is about less guesswork and higher quality, more timely feedback. Aren't those addressed by adopting DevOps and Agile techniques?

A

CC: The idea of a Mobius loop with DevOps, where things virtuously go out the door and then somehow information comes back in, isn't fully realized; it's like a one-way street for most companies. The software engineers definitely have CI/CD automated. They definitely have things on their automated pipeline going out the door. But, once things pass their desk and go out the door, they're on to the next item. They're on to the next work thing. They have no idea if anyone's even using their software sometimes.

That's why causal analysis is so important. If you're not getting all of the feedback, you're not really doing "Agile."

Q

"You're not really doing Agile" is a pretty bold claim. How do teams turn that around?

A

CC: First off, "Agile" is all about constant iteration and getting information back about whether or not what you delivered is actually working. One of the cycles that we see taking hold is this practice where a business leader or a product manager says, "I've got an OKR. I've got an objective and key result. I want to move the needle on something." Maybe it's page load time. Maybe it's customer experience. Maybe it's stability, reliability, or scale. Or possibly, they just want to save money by seeing if things run on a smaller EC cluster. Whatever it is, there needs to be an objective that is measurable. This is step one. Step two is to get things into production, whether that's an A/B test or some other form of data collection.

What's the point of doing a change if you're not going to actually measure whether you move the needle or not? Measure the results and then decide what you want to do with it. If you're not doing this, you're really not doing DevOps and you're certainly not doing Agile. And that's what feature management and experimentation solutions help us do. It closes the loop and gives us this causal information that we're looking for to truly move forward. It's why I'm really excited about what it can do!

> *What's the point of doing a change if you're not going to actually measure whether you move the needle or not? Measure the results and then decide what you want to do with it.*

Q

What rewards await those who implement feature management and causal analysis?

A

CC: The result is a more harmonious DevOps process. How does this improve the developer experience? One of the common things we hear when we interview developers about using feature toggles or feature flags is that they sleep better at night. If something goes out the door and breaks, it can easily be pulled back and reverted without an infrastructure change. Just knowing this helps developers rest easy.

Feature Management and Experimentation brings an improved developer experience to companies, increased efficiency, and ultimately stronger outcomes. The idea that all of these things are stacked together by one technology is really quite remarkable, but it makes sense when you actually look at the science behind it and understand the technology that's operating it.