


Mapping DevOps learnings to management

Posted on May 17, 2013 by fimblo (<https://labs.spotify.com/author/fimblo/>)

 One person likes this. Be the first of your friends.

There are many blog posts, articles and tweets about DevOps out there on the internet. Some of them discuss the pros/cons, some the consequences of its introduction, while others discuss how it was implemented.

Although this article refers to some DevOps adoption aspects, its main focus is on applying DevOps principles to a different area: engineering leadership.

In this article, I'll refer to "us" every now and then, and to avoid confusion, here's a quick intro:

- Ingrid Franck is the engineering team's agile coach.
- Ramon van Alteren is the engineering team's product owner.
- Mattias Jansson (yours truly) is the engineering team's chapter lead, sometimes lazily called a team lead, like I do in this article.

The engineering team itself consists at the time of writing of nine people.

DevOps culture at Spotify

Many parts of DevOps culture have pervaded Spotify from its early beginnings.

The first six people employed by Spotify were engineers, one of whom had an Operational role. This was back when Spotify was just another startup in an apartment. This background and admission of the importance of operational thinking from the very beginnings of Spotify history has heavily influenced the relationship between Dev and Ops.

We've come a long way since then- there are hundreds of engineers at Spotify now, spread out in four cities and three time zones. Although the DevOps mentality does not permeate the hearts and souls of every individual in the engineering team, and though it is not actually mentioned by name anywhere, one can see it show up everywhere in the day-to-day workflow as well as in conversations by the coffee machine.

Many startups are staffed by developers and the odd business guy. In those firms the operations engineer is hired once the code has been written and there is a need for someone to deploy and maintain the system. At Spotify, the two camps are overlapping in responsibilities, skill sets and interests. We have some uncommonly opsish developers here, and likewise many of our ops engineers have a strong developer background. The advantage of having this overlap is immense in the day-to-day, solving potential blockers long before they arise.

Backend developers deploy their code in production by themselves, with or without an ops engineer to hold their hand. This in turn, more often than not, encourages the dev in question to think seriously about traditionally operations-focussed problem areas such as monitoring, logging, packaging, and availability.

Since we have thousands of servers in production, our ops engineers have moved from thinking in terms of individual servers to clusters of servers. One-off manual fixes on individual servers is avoided when possible- instead, the ops engineer, through code, modifies the state of the authoritative data model of the backend, which in turn reflects onto reality via our configuration management system, backed by puppet (<https://puppetlabs.com/>).

Backend services typically have two so-called System Owners – one from Dev and one from Ops. Their core responsibilities reside in their respective dominions – the dev system owner owns the code, design and architecture, while the ops system owner owns the service once life is blown into it when it is deployed and is serving traffic. However, these two areas have great overlap, and thus the two system owners have regular checkups to discuss scalability, changes in neighbouring backend topology, coming new products which will affect service behaviour, etc.

A final example is how we have a dedicated team working on automation tools and services. Developer and Operations staff work side-by-side for weeks at a time solving specific problems, raised and prioritised by Ops themselves.

All this being said, our organisation still has a long way to go. The numbers of customers, servers, data centers, services, offices, and staff are growing all the time, and thus yesterday's solutions have a marked tendency of scaling badly into the present. On top of this, the ratio of Dev to Ops staff has changed in a way that has diluted the DevOps mentality in some ways.

Lessons learned of DevOps

The lessons the DevOps movement has taught us are many, but one of the most important is the value of aligning the goals of Dev and Ops. Get them to work side-by-side, give them space to learn from each other. By getting the two groups to communicate regularly, the developer will have a chance at understanding the reasons why Ops need to act a blocker at times- and will learn how to plan ahead and produce changes in alignment with the requirements of the Operational environment. Also, once Ops start to see their hardware and the services running on them as malleable datastructures which one can apply code upon, the developer suddenly has a different reach- he/she will be able to affect not just the code as it exists in packages, but will have much more flexibility on how the packages are applied in production.

Likewise, by injecting operational thinking into the development process, the frequency with which Operations engineers need to spend time on interruptions and clean-up is lowered, and their time can be spent on longer-term projects.

Generally, the DevOps work methods have helped both sides of the organisation think of the entire system – the value-stream (http://en.wikipedia.org/wiki/Value_stream_mapping) – not just the components they are traditionally responsible for.

Problems in Management-land

In many modern tech companies, one can find three distinct responsibilities which tangent an engineering team. In some smaller firms, and indeed even in larger ones, these responsibilities are typically gathered in one or two roles. At Spotify, we try to separate them so that three separate people own these distinct responsibilities. So what responsibilities do these roles entail?

- The Product owner is accountable for delivering products to one or more stakeholders in a timely fashion. (PO)
- The Team lead's mission is to maintain the team so its members are fit for the challenges expected of them, and is responsible for the architectural soundness of the internals of the product. (TL)
- The Agile coach is dedicated to nurture an environment of engaged and healthy team members, that continuously improve themselves as team members, their product deliveries and their team collaboration. (AC)

These three roles are at times at odds with each other. In most organisations, the people who have these roles have differing missions and can pull the team in different directions.

Have you ever experienced a conflict between the Product owner, pushing for an essential new feature, and the Team lead, who is concerned with the team's frustrations over the mountain of technical debt in the existing codebase? Or the Agile coach who feels that the team needs to stop and reflect more often in order to figure out how to improve themselves, but the Product owner hesitates and seems worried that the rhythm of the team will be disrupted? Or when the Team lead feels that the team is agile enough and actively blocks the increasingly agitated attempts by the Agile coach to help the team help themselves?

The problems we mention above are in many ways similar to the conflicting goals between Dev and Ops groupings in archetypical firms.

Companies new to DevOps often discover blockers (structural, social, cultural, etc) inside their firm, making adoption difficult. Even firms where DevOps prevails will find plenty to disagree upon. It is often the case that the entire problem set resembles two people in a single bed with a blanket that is too small to cover both. The result is a lot of pulling/shuffling of this blanket to try and cover all the exposed parts.

It is not unusual to see old-school Operations engineers who refuse to see infrastructure as code, preferring to manually modify configuration files on target machines; Developers who look down on operational work, who feel that their job is done once the build completes and that whatever happens when the code hits bare metal is someone else's problem. Dev and Ops leadership who are at odds with each other because of a mismatch of missions (the number of features shipped vs. keeping downtime at a minimum.)

The thing is, what makes DevOps so attractive is that it's all about encouraging developers and Ops engineers to talk and learn from each other. At the end of the day, it's all about communication. About aligning goals. Once we start listening to each other, we will have taken the first step towards some sort of DevOps synergy.

So... what happens when you do the same thing with an engineering team's closest leadership figures?

What if we get the PO, TL and the AC to regularly talk about their concerns, their short and long-term goals for the team, and to teach each other the realities within which they live? Will we find similar synergy effects in these three roles? Will we not only eliminate conflicts between them, but also find something... more?

Potlac

That's what we did, six months ago. The three of us had never worked together in this particular constellation before, and we were willing to do some serious experimentation. Our aim was to try to minimise misunderstandings and to possibly get some sort of synergy effect.

So what did we do? How did we apply the lessons of DevOps into our work?

Weekly sync meeting

Half an hour every week, we discussed the current state, from each of our perspectives. Each brought at least one topic to the session, which we then discussed and digested together. Example topics would include increasing stakeholder involvement, upcoming conferences, or the theme of the next retrospective.

Quick chats and sync before key meetings

Before one of us held a critical meeting, we would have a quick chat with the others to get last-minute feedback. One would reiterate the purpose of the meeting, if the goal(s) were realistic, or what one should do to truly get the involvement of the meeting attendees.

Regular one-on-ones

Each of us also have 1:1s with each other once a week- either at the office, a quick phone call in the evening or over lunch. By decreasing the members of the discussion by one, the tone of the conversation and the problems raised became more personal, but all the while orbiting our common goals.

Mock meetings

If a meeting was special, or the agenda experimental, we would hold mock meetings with each other, practicing the tricky parts to check for holes in reasoning or for discovering unexplained assumptions.

These are the four most obvious ways in which we worked together. The emergent property of this group of three was that we started to think in each other's shoes, and in some sense each of us was suddenly wearing all three hats – albeit our original one still larger than the other two. It made us stronger as a group, and the more we discussed the further we deepened our cooperation.

A critical success factor for the setup described is that we shared a strong commitment to the team functioning as opposed to a single engineer functioning. In our opinion, this is a fairly large part of what made this work; the shared idea that the team is bigger than the sum of its parts.

Some time after we had worked in this manner, we accidentally came upon a name for ourselves. I had, at some point, placed photos of us on a board, with our role initials under the pics- it spelled "PO", "TL", "AC". It sounded like something pronounceable, and when Ingrid realised that Potlac can be pronounced Potluck, the name fastened itself and never came off. (As you might or might not know, a potluck is a meal to which everyone brings their own food for sharing with the others. A successful potluck requires some sort of coordination between the people who come to the meal, otherwise one will have all dessert and no mains.)

Benefits of Potlac

"That's all very fine, and sounds nice. But what do you get out of it?", you might ask. Well, it depends a bit on what responsibilities you have. Below we will each state the main unforeseen benefits we gained by working together in this fashion.

Mattias: The Team lead

Management can be a lonely job. While engineers can swarm around a problem, I often cannot do this- I can ask the team to do many things for me, but some things simply cannot be delegated or shared (I'm thinking career goals, personal confidences, salaries, etc). Though I can not share these things with my Potlac colleagues either, there are other topics I can and do share. Examples include discussing new and different ways of solving conventional problems, or discussing how to scale our team in a sustainable way. Often, it was through these discussions that I got a piece of the puzzle which helped me understand some problem I was working on.

Since we have an ongoing dialogue in Potlac, we have grown to know each other's visions and our respective views on the state and history of the team. Through this, together with our different networks (both inside and outside the company), I get an advantage in that I see things on the horizon long before I would have otherwise. I can then prepare in time, and snuff out many problems before they become big ones.

Ingrid: The Agile coach

The Potlac gave me an opportunity and platform to have discussions around agile. A stage to dialog about servant leadership. A forum to find a consensus on what it means. It became an interface for the leadership team to focus on results and to hold each other accountable. It soon was a sandbox where conversations of empowerment, impediments and conflict were hashed out. It was also a classroom where we talked about approaches to stakeholder meetings, planning meetings, retrospectives and one-on-ones. It also developed into coaching sessions where we talked about our failures and what we learned. Instead of three individuals, each working toward our respective goals, we became a team- a leadership team with a united mission of supporting our engineers.

Ramon: The Product owner

Pushing for delivery can be just as lonely as managing a team. By working so closely together with both a Team lead and an Agile coach I gained a multitude of benefits. One of the most important ones is focus, it allows me to focus on delivery of enhancements to the products I am responsible for because I know that the other two equally important aspects of team leadership are covered by my two colleagues. Mystifying incidents of the past such as for example a sudden lack of commitment by an engineer for some time became a lot clearer with the added information from Mattias on the personal situation of that engineer. Ingrid opened up entirely new ways of handling typical issues with the team which helped me a great deal.

The second most important benefit I see is the typical thorny problem of avoiding (or repaying) technical debt. An open discussion between people representing the different interests involved makes it easier to approach this problem. Otherwise, it's just an internal debate in a single person's head.

We have seen how our initial experiments with Potlac brought new insights into our day-to-day work dynamics; unforeseen, and yet somehow expected. This way of putting ourselves in each others shoes has broadened our horizons, and given each of us more context when considering a problem.

At the end of the day, it's really all about exposing context. Context around why a product is necessary now rather than later; Context on the background to a conflict in the team; Context to help select the right combination of agile methodologies for this particular team at this particular time.

DevOps helps the engineers practicing it to better understand the points of views of tangential groups of people. It exposes their needs, and the requirements set upon them. In a similar way, Potlac has helped the three of us by giving us context in a focussed high-bandwidth channel.

So... what now?

Though this has been an amazing ride, the environment within which we have worked is changing. And with it, we will need to adapt our methods in some way. The team, which consisted of nine people (plus us) will probably double in size during the coming year. The company is, as always, expanding and with this comes changing focus. Ingrid has been asked to work as an Agile coach elsewhere, and Ramon has taken on broader PO responsibilities. Mattias will have a flurry of new engineers in his team to manage. Each of us will need to form brand new Potlac groups in our new surroundings.

One question we ask ourselves is how Potlac will scale with the growth of an engineering team. With a larger number of people in the team, we will most likely have more products to develop and maintain. This implies more product owners. The Team lead will not be able to manage this many people- some sort of team split is on the horizon. Finally, we might need two Agile coaches, if the team grows to this size. Will Potlac function if its members grow from three to six?

Another important question we have considered is how difficult it would be to try to duplicate this leadership model. In software- if a hack helps solve an immediate problem, it is a good thing. However, to really get true value from the hack, it must be documented and portable. Is Potlac portable?

It would be convenient if we could produce a puppet recipe covering how to deploy Potlac in other teams. Alas, puppet does not cover this particular feature, and until that time comes, perhaps this article will help iron out what we did to get the results we described above.

Happy org-hacking!

PS: If you want to more about how we organise our whole tech organisation, see Henrik Kniberg and Anders Ivarsson's paper on Scaling Agile at Spotify (<http://blog.crisp.se/2012/11/14/henrikkniberg/scaling-agile-at-spotify>).

[Like](#) One person likes this. Be the first of your friends.

Share this:

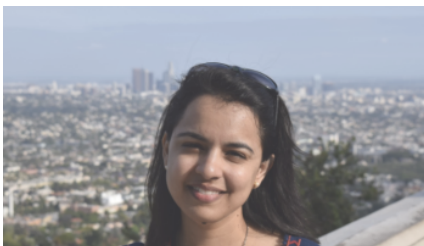
[Twitter \(https://labs.spotify.com/2013/05/17/devops-management/?share=twitter&nb=1\)](https://labs.spotify.com/2013/05/17/devops-management/?share=twitter&nb=1)

[Facebook 1 \(https://labs.spotify.com/2013/05/17/devops-management/?share=facebook&nb=1\)](https://labs.spotify.com/2013/05/17/devops-management/?share=facebook&nb=1)

[LinkedIn \(https://labs.spotify.com/2013/05/17/devops-management/?share=linkedin&nb=1\)](https://labs.spotify.com/2013/05/17/devops-management/?share=linkedin&nb=1)

[Email \(https://labs.spotify.com/2013/05/17/devops-management/?share=email&nb=1\)](https://labs.spotify.com/2013/05/17/devops-management/?share=email&nb=1)

Related



(<https://labs.spotify.com/2018/01/22/spotify-spotlight-interview-with-data-engineer-ankita-pawar/>)

Spotify Spotlight: Interview with Data Engineer Ankita Pawar

(<https://labs.spotify.com/2018/01/22/spotify-spotlight-interview-with-data-engineer-ankita-pawar/>)

In "People"

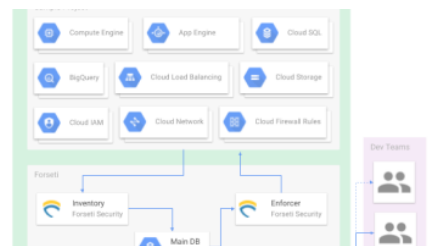


(<https://labs.spotify.com/2017/11/29/spotify-spotlight-interview-with-data-engineer-dara-ellars/>)

Spotify Spotlight: Interview with Data Engineer Dara Ellass

(<https://labs.spotify.com/2017/11/29/spotify-spotlight-interview-with-data-engineer-dara-ellars/>)

In "People"



(<https://labs.spotify.com/2017/09/15/stepping-up-the-cloud-security-game/>)

Stepping Up the Cloud Security Game

(<https://labs.spotify.com/2017/09/15/stepping-up-the-cloud-security-game/>)

In "Labs"

This entry was posted in [LABS \(HTTPS://LABS.SPOTIFY.COM/CATEGORY/LABS/\)](https://labs.spotify.com/category/labs/). Bookmark the permalink (<https://labs.spotify.com/2013/05/17/devops-management/>).

← ANALYTICS AT SPOTIFY ([HTTPS://LABS.SPOTIFY.COM/2013/05/13/ANALYTICS-AT-SPOTIFY/](https://labs.spotify.com/2013/05/13/analytics-at-spotify/))

INCIDENT MANAGEMENT AT SPOTIFY → (HTTPS://LABS.SPOTIFY.COM/2013/06/04/INCIDENT-MANAGEMENT-AT-SPOTIFY/)

0 Comments

Sort by Newest



Add a comment...

[Facebook Comments Plugin](#)

Recent Posts

Introducing Coördinator: A new open source project made at Spotify to inject some whimsy into datavisualizations (https://labs.spotify.com/2018/03/02/introducing-coordinator-a-new-open-source-project-made-at-spotify-to-inject-some-whimsy-into-data-visualizations/)
Spotify Spotlight: Interview with Continuous Delivery Engineer DonnieThompson (https://labs.spotify.com/2018/02/05/spotify-spotlight-interview-with-continuous-delivery-engineer-donnie-thompson/)
Spotify Spotlight: Interview with Data Engineer AnkitaPawar (https://labs.spotify.com/2018/01/22/spotify-spotlight-interview-with-data-engineer-ankita-pawar/)
Testing of Microservices (https://labs.spotify.com/2018/01/11/testing-of-microservices/)
Spotify Spotlight: Interview with Engineer DaenneySluijters (https://labs.spotify.com/2018/01/08/spotify-spotlight-interview-with-engineer-daenney-sluijters/)
Spotify Retro Kit (https://labs.spotify.com/2017/12/15/spotify-retro-kit/)
Spotify Spotlight: Interview with Data Engineer DaraElass (https://labs.spotify.com/2017/11/29/spotify-spotlight-interview-with-data-engineer-dara-elass/)
Spotify Spotlight: Interview with Product Manager SabrinaAronson (https://labs.spotify.com/2017/11/27/spotify-spotlight-interview-with-product-manager-sabrina-aronson/)
Autoscaling Pub/Sub Consumers (https://labs.spotify.com/2017/11/20/autoscaling-pub-sub-consumers/)
Big Data Processing at Spotify: The Road to Scio (Part2) (https://labs.spotify.com/2017/10/23/big-data-processing-at-spotify-the-road-to-scio-part-2/)

@SpotifyEng on Twitter

Tweets by @SpotifyEng

Spotify Engineering Retweeted



Matthias Grüter

@mattgruter

Dave from @SpotifyEng about engaging with the open-source community "The software thing is easy, talking to people is really the hard part."#KubeCon #CloudNativeCon



Embed

View on Twitter

(<http://instagram.com/spotify>)

(<https://www.facebook.com/Spotify>)

(<https://twitter.com/spotify>)

Legal (<https://www.spotify.com/legal/>) Cookies (<https://www.spotify.com/legal/privacy-policy/#cookies>)

AdChoices (http://info.evidon.com/pub_info/1120?v=1)

© 2018 Spotify AB | Powered by WordPress.com VIP (https://vip.wordpress.com/?utm_source=vip_powered_wpcom&utm_medium=web&utm_campaign=VIP%20Footer%20)

