PLATFORM ENGINEERING AND SITE RELIABILITY ENGINEERING: THE PATH TO DEVOPS SUCCESS

SEREMET, Z. & RAKIC, K.

Abstract: *The platform enables teams to concentrate on solving real business problems by abstracting organizational complexities and effort. Recently, many organizations are building an internal development platform or development control layer. Humanitec surveyed 1,850 engineering organizations last year and found that most are already building or planning to build their own internal developer platforms. Reducing complexity while enabling self-service for developers is always a popular move, especially when it works. While most developer platforms and control layers are built with these goals at their core, striking the right balance can prove challenging for most developers. Site reliability engineering (SRE) teams apply software engineering principles to improve reliability. The combination of the platform team and the SRE team unlock the productivity of application development teams. In this paper, the simple model for a thousand-person engineering organization when building a platform will be presented.*

Key words: *platform engineering, site reliability engineering, cloud, DevOps, GitOps*

Authors´ data: Asst. Prof. Dr. Sc. **Seremet**, Z[eljko]*; Asst. Prof. Dr. Sc. **Rakic**, K[resimir]**, * University of Mostar, Matice hrvatske bb, Mostar, Bosnia and Herzegovina, zeljko.seremet@fsre.sum.ba, kresimir.rakic@fsre.sum.ba

This Publication has to be referred as: Seremet, Z[eljko] & Rakic, K[resimir] (2022). Platform Engineering and Site Reliability Engineering: The Path to DevOps Success, Chapter 13 in DAAAM International Scientific Book 2022, pp.155-162, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-34-1, ISSN 1726-9687, Vienna, Austria
DOI: 10.2507/daaam.scibook.2022.13

1. Introduction

Over the past decade, engineering and technology organizations have converged on a common set of best practices for building and deploying cloud-native applications. These best practices include continuous delivery, containerization, and building observable systems.

At the same time, cloud-native organizations have radically changed how they're organized, moving from large departments (development, QA, operations, release) to smaller, independent development teams. These application development teams are supported by two new functions: site reliability engineering (SRE) and platform engineering. SRE and platform engineering are spiritual successor of traditional operations teams and bring the discipline of software engineering to different aspects of operations. (Labs, A. 2021)

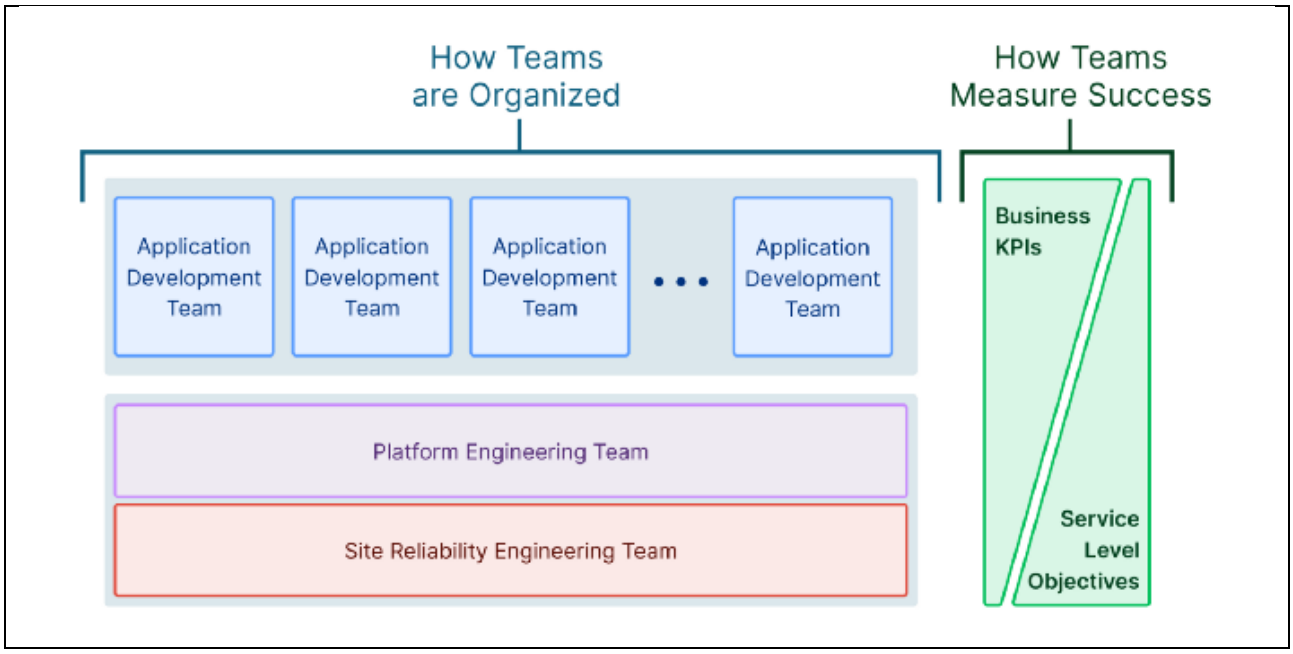

Fig. 1. Site reliability engineering and platform engineering (Labs, A. 2021)

Platform engineering teams apply software engineering principles to accelerate software delivery. Platform engineers ensure application development teams are productive in all aspects of the software delivery lifecycle. Site reliability engineering teams apply software engineering principles to improve reliability. Site reliability engineers minimize the frequency and impact of failures that can impact the overall reliability of a cloud (Celar et al., 2011) application.

These two teams are frequently confused, and the terms are sometimes used interchangeably. Indeed, some organizations consolidate SRE and platform engineering into the same function.

This occurs because both roles apply a common set of principles:

- Platform as product. These teams should spend time understanding their internal customers, building roadmaps, having a planned release cadence, writing documentation, and doing all the things that go into a software product.
- Self-service platforms. These teams build their platforms for internal use. In these platforms, best practices are encoded, so that the users of these platforms don't need to worry about it — they just push the button. In the Puppet Labs 2020 State of DevOps report, Puppet Labs found that High functioning DevOps organizations had more self-service infrastructure than low DevOps evolution organizations. (The 2020 State of DevOps Report, 2020)
- A constant focus on eliminating toil. As defined in the Google SRE book, toil is manual, repetitive, automatable, tactical work. The best SRE and platform teams identify toil, and work to eliminate it. (Beyer et al., 2016)

So, in the second chapter, we discuss the platform engineering, in the third chapter about site reliability engineering. The fourth chapter will describe the New Relic, in the five chapter, a DevOps and GitOps. The conclusion is given in the sixth chapter.

2. Platform Engineering

Platform engineers constantly examine the entire software development lifecycle from source to production. From this introspective process, they build a workflow that enables application developers to rapidly code and ship software. A basic workflow typically includes a source control system connected with a continuous integration system, along with a way to deploy artifacts into production.

As the number of application developers using the workflow grows, the need of the platform evolves. Different teams of application developers need similar but different workflows, so self-service infrastructure becomes important. Common platform engineering targets for self-service include CI/CD, alerting, and deployment workflows.

In addition to self-service, education and collaboration become challenges. Platform engineers find they increasingly spend time educating application developers on best practices and how to best use the platform. Application developers also find that they depend on other teams of application developers and look to the platform engineering team to give them the tools to collaborate productively with different teams.

The 2021 "State of DevOps" report by Puppet mentions this specific type of team and clearly notes that the level of DevOps maturity relates to the use of platforms built by platform teams and their crucial role in scaling up high-growth engineering organizations.

The common thread in everything a platform team does is:

- enabling developer self-service across the organization and
- keeping systems reliable and maintainable
- without compromising developers' (end users') experience of working with the infrastructure.

Platform teams need to create a cloud native platform consisting of high-quality building blocks that offer a paved path for development teams while still allowing those teams to deploy and operate their own application, which is what DevOps is about.

The platform empowers your teams to concentrate on solving real business problems by abstracting away organizational complexities and toil. This makes it easy for your teams to build, deploy and operate products. This, in turn, will allow your organization to accelerate its time to market, increase revenue, reduce costs, and create innovative products for your customers.

Another aspect is that the workers will become happier as their cognitive load lower. Platforms abstract away the usual infrastructure concerns, which leads to reduced onboarding times for new joiners, a mover between teams, a leaver, or the onboarding of a new development team. It will also become easier to attract new talent as your organization will have time to stay up to date on technology, which will encourage talented engineers to join your organization, and which will create more recruitment options.

Concentrating specialized skills in platform teams means recruitment efforts for development teams can focus on developers, testers, etc., without requiring more costly, specialized cloud or platform skills.

The Puppet survey results show that platform teams can be a building block of becoming a high-performing DevOps organization, but it's critical to note that success requires more than simply restructuring departments. It should also focus on how they work together.

In the same way platform teams prevent developer teams from reinventing the wheel through the paved path, platform teams should avoid falling into the same fallacy. Platform teams should focus on the specific needs of their organization and make use of off-the-shelf solutions.

The value of a platform team doesn't only come from the platform itself, but also from the support and adoption support the team provides to the development organization.

Platform teams are essential because this type of team acts as a major building block for enabling development teams; their role is unique and crucial to the rest of the organization's success. (The 2021 State of DevOps Report, 2021)

2.1. The simple model for organizing a platform team

Many organizations are on their journey to build platform teams and are struggling with it, some because of organizational and cultural topics, which aren't tackled in this post. Others have a more common problem: How many engineers should be working on "the platform"?

As the underlying idea of a cloud native platform is that it will make developers more effective, we can use the model created by Peter Seibel (tech lead for Twitter's **Engineering Effectiveness Group**). Seibel describes the following model:

$$E=(eng-ee)\cdot(1+(ee^{s}\cdot b))$$

where **E** is the total effectiveness, **ee** is the total number of platform engineers, **b** is the effectiveness boost for the first platform member and **s** is the scaling of the boost. The units for effectiveness are FTEs.

In your organization, the number of engineers is given. The interesting parameters to this model are the scaling factor, **s**, and the boost, **b**. For his model, Seibel uses 0.7 for **s** and 2% per FTE for **b**.

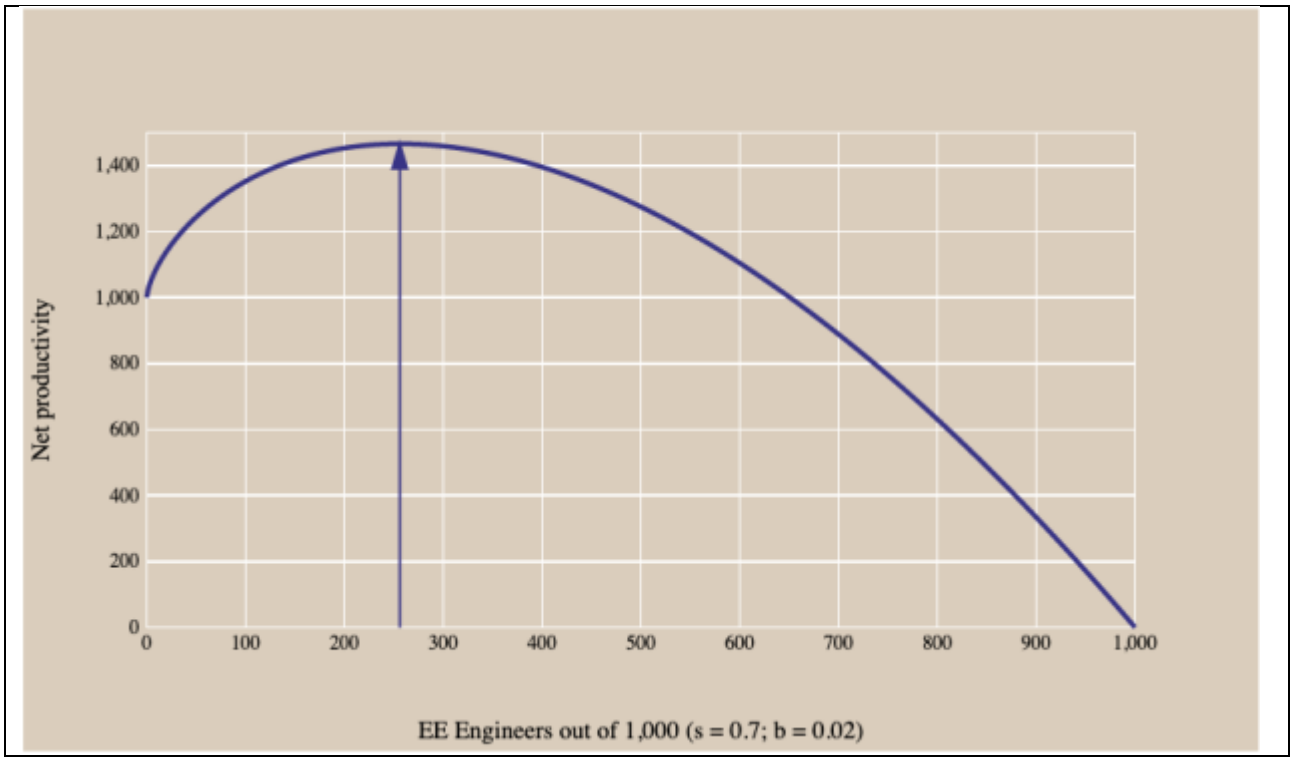

Fig. 2. Plot of the model for a thousand-person engineering organization

If these parameters are right, for a thousand-person engineering organization, we should devote over a quarter of our engineers — 255 — to engineering effectiveness, which is a big part of platform engineering. This yields a total effectiveness equivalent to 1,465 engineers for the price of 1,000.

Success comes with scale, which requires optimizing not for the individual and not for the team, but for the wider organization. Using the previous example of a thousand-person engineering organization, if the platform is only used by a hundred, 100 is the relevant number, not 1,000.

This is obviously a very simple model, and like all models, it is wrong. But that doesn't make it less useful. It shows that investing in platforms and platform engineering pays off. For platform engineers to be able to boost effectiveness across all of engineering, things need to be standardized. (Seibel, 2015)

3. Site Reliability Engineering

Site reliability engineers create and evolve systems to automatically run applications, reliably. The concept of site reliability engineering originated at Google and is documented in detail in the Google SRE Book (Skelton et al., 2011). Ben Treynor Sloss, the SVP at Google responsible for technical operations, described SRE as "what happens when you ask a software engineer to design an operations team."

SREs define service level objectives and build systems to help services achieve these objectives. These systems evolve into a platform and workflow that encompass monitoring, incident management, eliminating single points of failure, failure mitigation, and more.

A key part of SRE culture is to treat every failure as a failure in the reliability system. Rigorous post-mortems are critical to identifying the root cause of the failure, and corrective actions are introduced into the automatic system to continue to improve reliability.

4. SRE and Platform Engineering at New Relic

New Relic by 2015 as it grew from a handful of customers to tens of thousands of clients, all sending millions of requests per second to the cloud. The company had independent SRE and platform engineering teams that followed the general principles outlined above.

One of the reasons these teams were built separately was that the people who thrived in these roles differed. While both SREs and platform engineers need strong systems engineering skills in addition to classic programming skills, the roles dictate very different personality types. SREs tend to enjoy crisis management and get an adrenaline rush out of troubleshooting an outage.

SRE managers thrive under intense pressure and are good at recruiting and managing similarly minded folks. On the other hand, platform engineers are more typical software engineers, preferring to work without interruption on big, complex problems. Platform engineering managers prefer to operate on a consistent cadence. (Labs, A. 2021)

5. DevOps and GitOps

Over the past decade, DevOps has become a popular term to describe many of these practices. More recently, GitOps has also emerged as a popular term. How do DevOps and GitOps relate to platform and SRE teams?

Both DevOps and GitOps are a loosely codified set of principles of how to manage different aspects of infrastructure. The core principles of both philosophies — automation, infrastructure as code, application of software engineering — are very similar.

DevOps is a broad movement that began with a focus on eliminating traditional silos between development and operation. Over time, strategies such as infrastructure automation and engineering applications with operations in mind have gained widespread acceptance as ways better build highly reliable applications.

GitOps is an approach for application delivery. In GitOps, declarative configuration is used to codify the desired state of the application at any moment in time. This configuration is managed in a versioned source control system as the single source of truth. This ensures auditability, reproducibility, and consistency of configuration. (Myrbakken et al., 2017)

6. Conclusion

This paper presents the site reliability engineering and platform engineering as two functions that are critical to optimizing engineering organizations for building cloud-native applications. The SRE team works to deliver infrastructure for highly reliable applications, while the platform engineering team works to deliver infrastructure for rapid application development.

Together, these two teams unlock the productivity of application development teams. We have created simple model for a thousand-person engineering organization. Which shows that investing in platforms and platform engineering pays off. For platform engineers to increase efficiency across engineering, things need to be standardized. In future work, we will develop a simple model for organizing the SRE team.

7. References

Celar, S.; Seremet, Z. & Turic, M. (2011): Cloud computing: definition, characteristics, services and models, Annals of DAAAM for 2011 & Proceedings of the 22nd International DAAAM Symposium / Katalinic, Branko - Vienna: DAAAM International Vienna, 2011, 0001-0002

Skelton, M. & Pais, M. (2019): Team Topologies: Organizing Business and Technology Teams for Fast Flow, IT Revolution Press

The 2021 State of DevOps Report, https://puppet.com/resources/report/2021-state-of-devops-report/ Accessed: 2022-12-06

The 2020 State of DevOps Report, https://puppet.com/resources/report/2020-state-of-devops-report/ Accessed: 2022-12-06

Beyer, B., Murphy, N. R., Petoff, J. & Jones, C. (2016): Site Reliability Engineering: How Google Runs Production Systems, O'Reilly

Seibel, P. (2015): How to think about engineering effectiveness, https://gigamonkeys.com/flowers/ Accessed: 2022-12-07

Labs, A. (2021): SRE vs Platform Engineering, https://blog.getambassador.io/the-rise-of-cloud-native-engineering-organizations-1a244581bda5 Accessed: 2022-12-06

Myrbakken H. & Colomo-Palacios R. (2017): DevSecOps: A Multivocal Literature Review. In: Mas A., Mesquida A., O'Connor R., Rout T., Dorling A. (eds) Software Process Improvement and Capability Determination. SPICE 2017. Communications in Computer and Information Science, vol 770. Springer, Cham.