A comparison of DevOps and Site Reliability Engineering

Agnes Forsberg - agnesfo@kth.se

George Bassilious - gdba@kth.se

April 2021

1 Introduction

Efficiency and speed are vital in modern companies, and software engineering is no exception. In addition, reliability is also highly critical for companies and software engineering. Lack of reliability could result in terrible consequences, which could have the same impact as a failing service. The traditional way of handling this is with a development team who fight to get their code deployed as quickly as possible, and the operations team who would rather keep the safe, old code that is known to be reliable in deployment.

There are many solutions to this conflict, and two of the best known are site reliability engineering (SRE) and DevOps. They seem similar at first glance, and they do both combine development and operations. However they differ in a number of areas, and this essay plans to provide a thorough comparison of the terms, as well as an insight to how the approaches affect workflows and culture.

2 Background

This section will provide background on the concepts of DevOps and SRE respectively to give the reader an overview of the terms.

2.1 DevOps

DevOps is the organizational model that aims to find a cooperation between ***Dev***elopment and ***Op***erations. The work that traditionally was done in different teams is more collaborative and works more in synchronization.[1] The areas that are involved in the DevOps model are roles within development, IT operations, quality engineering, and security. What the model accomplishes is that it tries to make the teams cross-functional in a way that ensures that they can work continuously on developing new operational features, while at the same

time maintaining the upkeep of these features and monitoring them. [4]

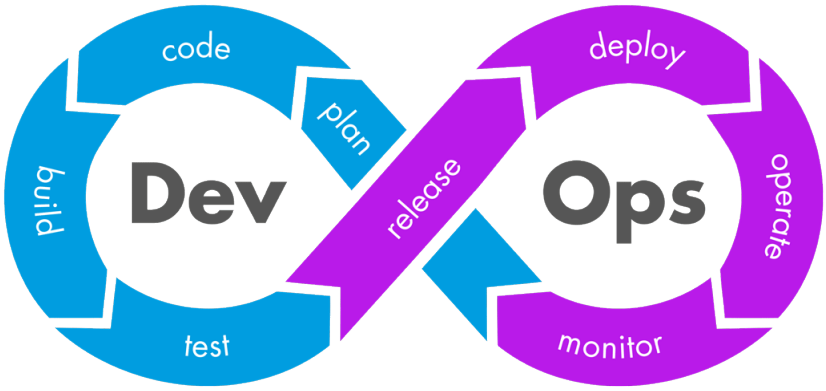
The mantra of the Model surrounds itself on 5 concepts. The first being **culture**, which has to do with the collaboration and communication between different areas in the application life-cycle. There is also a focus on continuously improving all different aspects of the workflow. The second concept is **Automation**, where teams should try to minimize the amount of repetitive work needed to progress both in development but also maintaining the features. The third concept is **Lean IT** which is a workflow that ensures to maximize the gain of customers, and minimize time waste and bad workflow. The fourth, **Measurement**, and fifth concept **Sharing** go hand in hand where Measurement has to do with using values and metrics that improve the work performance and sharing is the motivation of always communicating and collaborating between teams and leaders to improve the work. [10]

Following the 5 concepts will implement a good work ethics for a DevOps organization, however there are some practical steps that must be taken for the model to work. Devops is usually split into 4 main phases that surround application life-cycles or products [11]. The phases aforementioned are: **Plan**, **Develop**, **Deliver**, **Operate** where each phase usually depends heavily on the others. This Model is usually represented by a famously known infinity symbol as seen in figure 1. The four phases have been complemented with the workflow and link between each of the 4 phases. First of all the first phase *plan* where teams form ideas, define features and other capabilities that their product will have. After the first phase it has become time for the *develop* phase. This phase has everything to do with programming, testing, reviewing and building the product and in the model(figure 1) are represented as "code", "build" and "test" steps. When that has been done the workflow goes to the *deliver* phase where the main focus is to "release" and "deploy" applications in a production environment. Finally, the product is deployed and the final phase *operate* is the two steps of the model(figure 1) where the focus is to ensure that the system is reliable and is checked by monitoring and troubleshooting the products and maintaining the uptime. Once all this has been successful the teams can go back to the *plan* phase if they want to improve or develop new features [11]. Most of these phases are automated and there are tools to ensure that the work is efficient, however it is important to have the right processes for DevOps to work as intended. [1]

2.2 Site Reliability Engineering - SRE

Site reliability engineering (SRE) is a way of combining software engineering and operations, with the goal of creating highly reliable software. The term was coined by Ben Treynor Sloss, the senior VP overseeing technical operations at Google. Google describes SRE as operations when you treat it as a software problem [7]. In SRE the main goals are to create software systems that are scalable and highly reliable. Treynor Sloss was in 2003 tasked with creating a

Figure 1: The DevOps model representation



production team, but had up until that point only worked in software engineering. He therefore designed the group from that view point, and SRE at Google grew from that group. [7]

Site reliability engineers can, according to Google, be broken down into two main parts. 50-60% are Google software engineers - which means they, when they were hired, went through Google's standard software engineer procedure. The rest are people who almost matched Google's demands for software engineers, but also had some technical knowledge that most software engineers do not. According to Treynor Sloss, the most common additional expertise sets includes UNIX systems and networking. Treynor Sloss claims that this diversity in the teams results is better systems. [7]

In SRE it is important to be involved in many parts of the process, to make sure reliability is always a focus and also continuously being improved. A site reliability engineer needs to be multifaceted in order to analyze the workflow and product in several ways to improve reliability; he or she may even benefit from a knowledge of finance to be able to balance production costs with performance. [5]

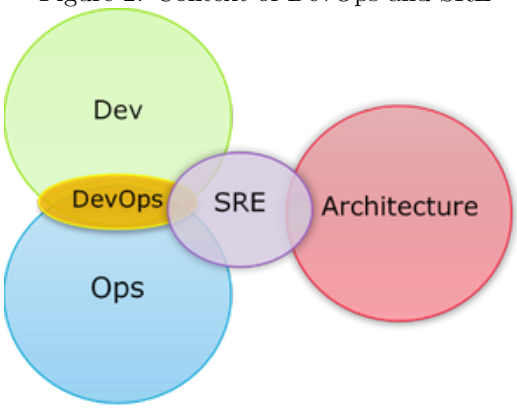
3 Comparison

This section will dive deeper into the different aspects of DevOps and SRE, and for each aspect compare the concepts.

3.1 Concepts and Philosophies

A site reliability engineering team often works on the entire IT infrastructure, and making it automated. DevOps on the other hand is more of an island operation, providing automated solutions on a need basis [9]. Figure 2 shows the relationship of DevOps' and SRE's contexts.

Figure 2: Context of DevOps and SRE



Google Cloud Tech published a video in 2018 describing the differences of DevOps and SRE, with site reliability engineer Liz Fong-Jones explaining that it can be seen as DevOps being an interface, and SRE a concrete class that implements DevOps. By this, Liz and her video co-host Seth Vargo, meant that if DevOps is more of a philosophy - SRE is a prescriptive way of accomplishing that philosophy. For example DevOps might help in formulating workflows and the work culture, and a site reliability engineer would find concrete ways to build this into the entire architecture. [8]

An example of this is the DevOps principle of measuring and quantifying everything. The way SRE would implement this principle would be to treat operations as a software problem, and in that provide specific ways for the company to measure availability, uptime, outages and toil. [8]

3.2 Workflow

Even though there might be similarities in the final products for both SRE and DevOps teams, the workflow differs a lot. For instance, in the software development life cycle. The main focus for a DevOps organization is to develop effective products and quick deliveries continuously. An important factor in that workflow is that the teams want a zero downtime deployment and therefore have to in each step of the process ensure that all issues are solved and thought of before carrying on the the next step. In a SRE organization the managing of the applications is most effective after the deployment has occurred. The focus

of the workflow is to ensure that the application has maximum uptime and stability for the application. In other words, DevOps teams aim to develop in a way that predicts and solves problems before deployment where SRE teams try to fix potential problems that might arise once they arrive. [6]

Another aspect that affects the workflow is the speed of producing new updates/features. In DevOps organizations the philosophy is all about having quick release cycles where automation effectivizes the process and can give quicker deployment and continuous development. This leads to expensive costs if there is failure. Therefore as mentioned earlier, there is a heavy focus on not overlooking mistakes in the pre-deployment workflow [9] On the other hand for SRE teams, robustness and resilient solutions are important when producing new updates/features.This might take longer time and requires more efficient testing after deployment, however heavily reduces the cost of failure. The workflow therefore might be more intensive in reliability and maintaining maximum upkeep than producing quick and new features. [6]

3.3 Work culture

One major difference between SRE and DevOps is how companies applies the strategies on the workforce. Site reliability engineer is a job title, and while a DevOps engineer exist as a job description it is not as common. There are also some people who think DevOps is more of a culture that should envelop the entire organization than something that should be confined into a job or team title [3]. However at Google, who *invented* SRE, there are plenty of site reliability engineers and teams. In the SRE book [2] that was written by several people at Google, this is how SRE is explained and proposed.

On the other hand, both terms can be used quite vaguely to describe a way of working, and don't follow a strict manual. Google has published several books on SRE, but most companies are nothing like Google. SRE is therefore what the company makes of it, and can look very different depending on the team that utilizes it.

4 Conclusion

It can at first glance seem like DevOps and SRE are two names for the same concept. Hopefully, however, this essay has proved that is not the case. DevOps is a philosophy, or rather a set of principles, that can be implemented in many different ways depending on the organization and its size, structure and culture. SRE provides a way of implementing DevOps, although SRE can also differ from company to company. SRE also extends DevOps when it comes to building the IT architecture and infrastructure.

It is also more common for SRE to be a job title, while DevOps can take form

in DevOps engineers, teams, or simply as a philosophy for the entire software team.

References

[1] Atlassian. *DevOps Principles*. URL: https://www.atlassian.com/devops/what-is-devops.

[2] Betsy Beyer. *Site reliability engineering : How Google runs production systems*. Sebastopol, CA: O'Reilly Media, 2016. ISBN: 978-1491929124.

[3] Kevin Casey. *10 DevOps 'Secrets' for Job Seekers—and Everyone Else*. New Relic. Apr. 2018. URL: https://newrelic.com/blog/nerd-life/devops-secrets.

[4] Christof Ebert et al. "DevOps". In: *IEEE Software* 33.3 (May 2016), pp. 94–100. DOI: 10.1109/ms.2016.68. URL: https://doi.org/10.1109/ms.2016.68.

[5] Jan Fletcher. *What Does a Reliability Engineer Do?* wisegeek. Feb. 2021. URL: https://www.wise-geek.com/what-does-a-reliability-engineer-do.htm.

[6] Sunny Raskar. *Site Reliability Engineering (SRE) 101 with DevOps vs SRE*. Apr. 2021. URL: https://www.msystechnologies.com/blog/site-reliability-engineering-sre-101-with-devops-vs-sre/.

[7] Benjamin Treynor Sloss. "Introduction". In: *Site Reliability Engineering*. Ed. by B Beyer et al. O'Reilly, 2017.

[8] Google Cloud Tech. *What's the Difference Between DevOps and SRE? (class SRE implements DevOps)*. Youtube. Mar. 2018. URL: https://www.youtube.com/watch?v=uTEL8Ff1Zvk.

[9] Sean Washington. *DevOps vs Site Reliability Engineering*. LinkedIn. June 2016. URL: https://www.linkedin.com/pulse/devops-vs-site-reliability-engineering-sean-washington/.

[10] *What is DevOps?* Netmind. Mar. 2021. URL: https://netmind.net/en/devops-best-practice/.

[11] *What is DevOps? DevOps Explained: Microsoft Azure*. Microsoft Azure. URL: https://azure.microsoft.com/en-us/overview/what-is-devops/.