IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

An exploratory study of DevOps and it's future in the United States

Sikender Mohsienuddin Mohammad

KPMG LLP, Sr. Associate & Information Technology, Wilmington University
419 V street, Apt D, Sacramento, CA 95818

Abstract—*This document will explore Deposits building blocks, and its future in the U.S. Every organization is a software company nowadays, and large companies are keen to take over huge segments of the economy. This is a coordinated approach that would not only eradicate the endless series of concurrent activities but also maintain an effective implementation process in the life cycle of software development (SDLC). Currently, a very organized approach is called DevOps and is characterized by fewer errors, more efficiency, and remarkable coordination. With so much written about DevOps, it's easy to forget that it's still in its early stages of maturity. But it is expected to grow well beyond its current status over the next few years. Technologists have made great strides to benefit from it ever since the DevOps ideation. Notwithstanding advancements and the progress teams have seen in pockets, many of the core problems still prevail: scaling across the organization due to manual processes, inadequate coordination across dev and ops teams, and overlapping tools that hinder agility. Fortunately, there are hopes in the future especially for organizations hoping to get the most out of DevOps. Software engineers now can continue to increase their knowledge to be ready for the next DevOps stage, through growth, delivery, and quality assurance. At the same time, all team leaders need to be prepared to handle some major challenges. Over the years, DevOps has begun to evolve and gain recognition. This trend has picked up the exponential pace with the advent of mobile phones. Today, the DevOps cycle continues to thrive on the principles of enhanced teamwork and automation to create, test, and release apps faster. DevOps will continue to be seen clearly in the U.S., in which continuous updates are transforming the way software is delivered to a near-limitless market. In this intensely competitive world of technology, DevOps has now become a must for economic development. DevOps has been a primary feature and influenced the tech landscape and many analysts expect that DevOps will become the standard and will hit its peak this year. To make DevOps work and to promote agility over the next ten years, the trick is pretty simple: Have the right team together at the right moment.*

Keywords — *DevOps, Building Blocks of DevOps, Future of DevOps in the U.S., DevOps trends, DevSecOps.*

I. Introduction

While companies have different meanings of DevOps, we can identify DevOps as a way of thinking that a team takes to bring their engineering dynamics to new levels. DevOps is mainly concerned with eliminating technical barriers and mainly cultural barriers between concept and implementation, thereby allowing the software process to be enhanced, quicker, cheaper, and more efficient [1]. Whatever it might be called, it all will end up with automation, which in effect will allow businesses to easily grow, ship quickly, fail easily, recover swiftly, and adapt fast. Through the SDLC model to the present, things have greatly changed [2]. DevOps had been introduced in 2009, advocating a societal transition and other technical concepts where everything was viewed as code. The principles like CI / CD were then given a lot of limelight, but the software was written as a great monolith and there were many challenges [3]. Thus, in 2011 the architecture of microservices was introduced, this architecture of microservices promoted fine-grained and loosely coupled components with a special mission. The applications written after this loosely interconnected cloud-based architecture have been called cloud-native. Many companies have moved from VMs to Kubernetes to Serverless, based on their business needs and objectives. Building mobile app software needs a lot of computer programming [4]. Companies now have to provide users with unique, high-quality software so that they stay competitive in the technology market as much as possible [4].

The companies will concentrate on two key factors in this new approach: resilience and adaptability. The former describes the ability to code and recode, while the other defines the ability of DevOps to have changed along the SDLC [5]. Companies are actively working to integrate these attributes into their workflows to facilitate DevOps principles of continuous delivery, continuous integration, and sorting out glitches to enable users to benefit from an inconvenient software experience [6]. Despite large investments in the implementation of DevOps, most companies have made progress only on improvements and downstream applications [7]. Too often, developers have to log in, create requests for change, and manually complete forms – a frustrating, needless, and overall process. The reality is that DevOps manual operations literally cannot survive over the next decade. But the trend begins as companies try to automate activities such as change management by using complex controls via data and policy automation to achieve complete CI / CD. As automation becomes increasingly popular, developers can spend more time creating and delivering apps faster and more effectively. The main focus of this study is to explore DevOps in detail and it's future in the United States.

II. Literature review

A. *Building blocks of DevOps*

According to Hüttermann, the DevOps framework to continuous software creation and deployment (also referred to as continuous software delivery) has made the IT world grow faster. This has become a valuable advantage for IT leaders and CIOs to identify key building blocks that are critical for a productive DevOps operation [8]. It is an approach that incorporates cultural values, organizational improvements, and the introduction of new technologies and practices. The main building blocks of DevOps are as follows:

1.) Total management of small teams

Code production historically involved several closely related teams, which depend on each other to complete their tasks. Processing time became an immense issue as people became potential bottlenecks. According to Kim et al., DevOps strategic choices promote software engineers who manage minor functionalities and own the entire chain from Dev to Ops [8]. This is because over-specialization leads to numerous handoffs and lines that lead to longer lead times. It does not mean that highly skilled engineers are outdated. On the contrary, technology is becoming more advanced and we need more skills than ever before. According to Gill et al., organizations must promote professional learning and growth to enable their personnel to become rounded experts [9].

2.) Automated continuous feedback checks

In IT, bad results are typically triggered by a lack of quick feedback. For software engineers working on waterfall projects, the techniques used before Agile were not uncommon in developing code before it was sent to QA testing. Having test findings late in the game means that developers can have worked on flawed code for a whole year without understanding it. It's not simple to find the root of the problem and repair it one year afterward. Nybom et al. mention that QA testing is at the core of DevOps to address these issues. It seeks to provide rapid feedback loops when work is done so that the development team can correct errors as and when they occur [11]. This means that developers must be able to carry out their experiments instead of relying on a different team. This can be accomplished by automated tests, which automatically identify problems, so that they can be fixed before further research is done and so that the developers do not replicate the same mistake. The bigger the device, the more improvements are checked or put into service, the easier it is to figure out where the problem comes from [11].

3.) Loosely Coupled Architectures

System adjustments or upgrades need close communication with everyone else working on the program and can, therefore, be affected. This leads to a dynamic cycle of change management. Gregory et al mention that minor changes in closely connected architectures can result in major errors [12]. But even then, once the technology is ready for development, new technology has already been added by other classes. This means that one will eventually release the code into a different device from the one checked. The bigger the device, the more improvements are checked or put into service, the easier it is to figure out where the problem comes from. According to Laan, any transition, regardless of how small, is an extremely frustrating risk in managing [13].

A DevOps-oriented architecture helps small teams to easily and rapidly develop, test, and deploy code to improve efficiency and performance. This is facilitated by loosely connected service-oriented architectures (SOA). The loose link ensures that components (or services) are enclosed into containers and are not influenced by changes around them — all they need is in the container. Bell et al. suggest that strictly communicating through application programming interfaces (APIs) does not share database structures or schemas [14]. The result: a small environment with well-defined interfaces that promote versatility and scalability even in large companies with thousands of developers who constantly deploy code to the same framework.

4.) Telemetry for reviews on production applications

When a code is in development, things will go wrong and there can be no way to solve it. To fix it, one needs to first define the cause, and data is needed to do so. Software developers would normally rebuild servers one by one in the past before the problem was solved [15]. Although that sounds nuts, particularly when it comes to mission-critical applications, that was the quickest way to get the systems up and running again. According to Bell et al. reporting and tracking systems were not as advanced, and IT had no telemetry (i.e. data) required for it to rapidly reach the source [15]. Rebooting was the fastest way to restore infrastructure, so it was. Currently, things have changed drastically. New tools to track, monitor, and warn help IT identify contributing factors and recognize problems early, preferably before customers are affected [16]. Such systems collect and submit measurements and metrics constantly and automatically to a tracking tool. When an anomaly occurs, the program warns IT to allow it to be investigated. IT will check with enough data points that the systems are working properly and flag when an issue arises such that corrections can be made immediately. DevOps is a crucial component of setting up these instruments and visualizing telemetry across dashboards for the entire team.

5.) Organizational continuous learning

While the goal is to identify problems before they are deployed, errors may slip. This is why developers need to be able to identify problems themselves, fix them, and share the lessons learned within the company. Smeds et al. suggest that for this to occur, one needs to build a learning mechanism where failures and mitigating measures are detected and made available in the IT department to become a resilient organization [16].

6.) DevSecOps: Security building in the DevOps approach

Security is a crucial factor that was initially overlooked when DevOps was introduced. The DevSecOps movement corrects it by applying the DevOps methodology and incorporating protection into the everyday work of developers and operations. Sacks state that a typical personal relationship between production, operations, and InfoSec are approximately 100:10:1[17]. When code is considered ready for use by dev and ops, InfoSec verifies it. When a problem is found – as it is always – work is returned for rework. The process of sending back and forth research is inefficient, costly, and causes friction between teams. To solve this, DevOps needs the same strategy. DevOps ensures operating specifications are implemented from day one. The same has to be done for protection needs — this is the core of DevSecOps. DevOps is expanded to include security through automation and incorporation into production and operations by automated InfoSec tests, which will be performed along with all other automated operating tests. Also, InfoSec and developers work early on to ensure that safety and enforcement goals are achieved [17].

Four considerations are significant when competing with the elite of the DevOps industry.

i. Lead time
ii. Frequency of release
iii. Time for recovery
iv. Change rate shift

III. TABLE I

DevOps Performance Metrics				
	Elite	High	Medium	Low
Lead time for changes	<1 day	1 day-1 week	1week-1month	1month-6months
Release frequency	On-demand (>1 daily)	one daily-one monthly	one weekly-one monthly	once monthly-once every six months
Time to recovery	<1 hour	<24 hours	<24 hours	7 days-1 month
Change failure rate	0 to 15 percent	0 to15 percent	0 to 15 percent	46 to 60 percent

Fig i: DevOps Performance Metrics

IV. Future of DevOps in the U.S

There have been waves in the IT industry in the emergence of AI and automation, and DevOps is no exception. DevOps in the U.S has big potential projections. AI and automation will transform the development process of applications in the coming years. Apps will be automatically composed and coded. The AI, data, and machine learning will determine tasks such as integration, stability, testing, and deployment in all-new AI Ops. A data-driven approach will allow software, similar to other AI-driven technologies such as Apple's Siri and Alexa, to dynamically adjust its behaviours to its environment [18]. Besides, this critical change from programmed to non-programmed systems will require better security and development management and developers will be able to concentrate on value-added activities.

V. Practical steps the United States need to take in the future

The main focus for transformation into fully developed DevOps starts with the identification of the most troublesome and typical bottlenecks and the creation of a strategy with budgets, owners, and metrics. If the number of errors escaped into production is identified, the significance of this transformation is also conspicuous. This can then be replicated in other bottlenecks or areas which can be changed. Reducing challenges – and the desire to overcome them – will also lead towards greater tolerance by team leaders who can resist change [19]. For example, people who have been in the industry for many years may find it challenging to let go of the manual system, but to integrate them into a collaborative approach where data and perspectives help enhance the work-life tangibly.

VI. QA test

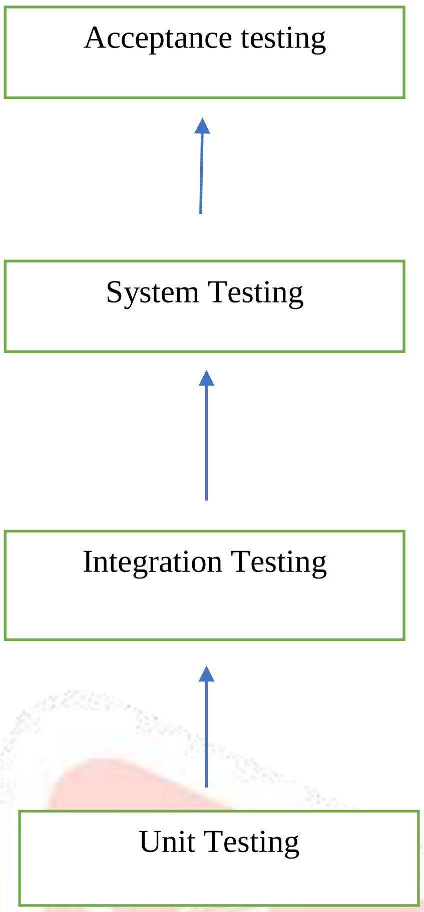

Fig ii: Unit tests play of software

The United States may have to recognize developing new competencies in DevSecOps, DevQualOps, and perhaps DevAPMOps (integrating APM activities into the build cycle). New technologies would also create new software development criteria beyond those listed above – 5 G and IoT apps, for example. The tech of the future changes rapidly and DevOps evolves quickly. These are exciting times, but management needs to concentrate on training teams in tools and expertise to take advantage of the opportunities available [19]. Likewise, each team member must accept this new environment and the opportunity for personal growth.

VII. Significance of this research to the United States

This research is vital to the United States in knowing its progress on DevOps implementation and what it needs to do to make sure that it is not left behind. The U.S. will gain more knowledge on some of the developing competencies that it must fully implement to meet the demands of the software development that can compete in the technological market. New competences in DevSecOps, DevQualOps, and perhaps DevAPMOps needs to be integrated into APM activities into its already the build DevOps cycle. Many companies are still lagging behind and therefore should build and make changes to their systems to improve the services that they provide using their technologies.

VIII. Conclusion

A decade after the great DevOps experiment, the numbers are clear: DevOps is here to stay — and for some excellent reasons. Many found this unlikely, but DevOps managed to integrate business users, developers, test engineers, and security engineers into a single workflow that focuses on customer needs. Developers and system managers avoid fighting and start helping each other, which minimizes the tensions all around. Business managers are pleased to obtain the software tools they need to market goods and services. Managers are continually watching the preferred dashboard indicators – sales, consumer satisfaction, machine confidence – travel north. This means that everybody should achieve the best performance and overall customer experience. However, improvements like these don't quickly get implemented. To deploy code more efficiently while your processes remain smooth, one needs the flexibility of tracking all adjustments to the technological environment accurately. The New Relic program provides developers, as well as operations, total visibility — from a digital transformation through apps and complex connectivity, to automated alerts and workflows — that can help those within an enterprise learn how software is implemented and functions in real-time.

References

[1] Herring, M,2015, “Continuous everything in DevOps,” Accenture. Retrieved from: https://www.accenture.com/us-en/blogs/software-engineering-blog/hering-continuous-everything-in-devops

[2] Virmani, M.,2015, “Understanding DevOps & bridging the gap from continuous integration to continuous delivery. *Fifth International Conference on Innovative Computing Technology (INTECH 2015)*. DOI:10.1109/intech.2015.7173368

[3] Boehm, B.,2006, "A view of 20th and 21st-century software engineering," in Proceedings of the 28th international conference on software engineering, pp. 12–29.

[4] Riungu-Kalliosaari, L., Mäkinen, S., Lwakatare L. E., Tiihonen, J. and Männistö, T.,2009, “DevOps Adoption Benefits and Challenges.

[5] Humble, J. and Farley, D., 2010, “Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation (Adobe Reader). Pearson Education.

[6] Limoncelli, T., Chalup, S., Hogan, C., and Limoncelli, T., 2014, The practice of cloud system administration.

[7] Wettinger, J., Vasilios, A., and Leymann, F.,2015, "Automated Capturing and Systematic Usage of DevOps Knowledge." Proceedings of the IEEE International Conference on. IEEE Computer Society.

[8] Hüttermann, M, 2012, "Building Blocks of DevOps. DevOps for Developers," 33-47. DOI:10.1007/978-1-4302-4570-4,3.

[9] Kim, G., Love, P., and Spafford, G., 2014, Visible Ops Security: achieving common security and IT operations objectives in 4 practical steps, Eugene, OR: IT Process Institute.

[10] Gill, A.Q., Loumish, A., Riyat, I., and Han, S.,2010, "DevOps for information management systems," VINE Journal of Information and Knowledge Management Systems, vol. 48, no. 1, pp. 122–139.

[11] Nybom, K., Smeds, J., and Porres, I.,2006, “On the impact of mixing responsibilities between devs and ops,” in International Conference on Agile Software Development, pp. 131–143.

[12] Gregory, J., and Crispin, L., 2015, More Agile Testing: Learning Journeys for the Whole Team, Addison-Wesley, Upper Saddle River, N.J.

[13] Laan, S.2011, “IT infrastructure architecture: Infrastructure building blocks and concepts,” U.S.A: Lulu Press.

[14] Bell, T. E., and Thayer, T. A.,2006, "Software requirements: Are they a problem," in Proceedings of the 2nd international conference on Software engineering, pp. 61–68.

[15] Huttermann, M., 2012, “DevOps for developers. Apress.

[16] Smeds, J., Nybom, K., and Porres, I.,2015, “DevOps: a definition and perceived adoption impediments,” in International Conference on Agile Software Development, pp. 166–177.

[17] Sacks, M., 2012, Pro website development and operations: Streamlining DevOps for large-scale websites, Apress, [New York].

[18] Sikender Mohsienuddin Mohammad, "CONTINUOUS INTEGRATION AND AUTOMATION", International Journal of Creative Research Thoughts (IJCRT), ISSN:2320-2882, Volume.4, Issue 3, pp.938-945, July 2016, Available at :http://www.ijcrt.org/papers/IJCRT1133440.pdf or http://doi.one/10.1729/Journal.24061

[19] Tessem, B., and Iden, J.,2008, “Cooperation between developers and operations in software engineering projects,” in Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering, pp. 105–108.