IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

THE CHALLENGES AND MITIGATION STRATEGIES OF USING DEVOPS DURING SOFTWARE DEVELOPMENT

Dhaya Sindhu Battina

Sr. Data Engineer, DevOps & Software Automation SME

Department of Information Technology

USA

Abstract— *This paper looked at the challenges that DevOps brings to software development, as well as the techniques for mitigating such challenges. Every industry, from finance to retail manufacturing, relies on software. Individual desktop programs, large-scale web applications, and mobile apps are all examples of software [1]. Due to software's intangible and complicated nature, requirements change fast, and software development teams have found it difficult to generate software that adequately meets client expectations while also providing the desired functionality and software quality [1]. Innovations have advanced the software business fast. Researchers have developed new programming languages, new database designs, and new development technologies such as Cloud Computing, Crowdsourcing, APIs, and SOA as a burgeoning business that is just 50 to 60 years old [2]. Because of the introduction of new technologies, old software systems become out-of-date and must be improved for the company to remain viable. Because software is always evolving, several Software Development Life Cycle (SDLC) models have emerged, including the Waterfall approach, Iterative Model, Spiral Model, Agile, and derivative forms [2]. These models all cater to software development, deployment, and maintenance. With today's technological advancements, the industry is moving toward the "DevOps" paradigm of software development. To provide software and services quickly, reliably, and of better quality, DevOps uses a variety of methodologies that bring together developers and operations personnel. In a team empowered with complete responsibility for their service and its underlying technological stack, duties and responsibilities are shared from development through deployment and maintenance [3]. This study is meant to verify and assess whether the problems and mitigation measures of implementing DevOps from systematic literature research are widespread in business.*

Keywords: *DevOps, CI, CD, machine learning, CI/CD Pipeline, software development*

I. INTRODUCTION

Traditional software companies have distinct divisions for software development, IT operations, and quality assurance. Development and operations are often at odds when providing excellent software to clients regularly [4]. Developers are more interested in providing new features or modifications to consumers rapidly, while operations want greater reliability and security and suggest that they don't alter their products as regularly. It's difficult for operations to tolerate the frequent release of new versions [4]. These disagreements may impede the advancement of software [4,5]. To bridge the gap between development and operations, DevOps combines the words "development" and "operations," which are used to change incentives and share techniques throughout the whole development process." Development and operations must work together more effectively to tackle crucial problems throughout the software development process, and DevOps is a collection of methods for doing so [8]. Fear of change and unsafe deployment are key challenges [5]. There is less of a divide between developers and operations personnel and end-users as a result of DevOps [5]. DevOps may embrace and give strategies to handle the everyday issues of software delivery [8]. However, according to our research, there isn't a consensus on what DevOps means [6] [7]. DevOps requires an understanding of topics of software engineering other than DevOps to fully grasp its extent. There is no such thing as a DevOps department [8], and there is no DevOps approach or process either [7]. While DevOps and Agile have certain similarities, there are some key differences as well [7,8]. Agile is a metaphor for shifting perspectives. However, the use of DevOps results in cultural shifts inside the business [8]. Agile is more of a methodology, whereas DevOps is more of a framework. Businesses are beginning

to use DevOps throughout the development process as a result of the growth of DevOps in recent years. Using DevOps during software development poses some challenges, and this thesis seeks to identify and collect these issues as well as mitigating solutions. This is accomplished by looking at the problems that might arise while utilizing DevOps and the solutions that can be used to mitigate those problems in the literature and the industry.

The software industry must recognize its strengths and limitations to remain competitive and expand, as well as use the best software development method. When compared to other places where software is developed, Sri Lankan culture and values stand out. New software development approaches such as DevOps should be explicitly reviewed to see whether they can be employed in the country's current software development process models to provide high-quality products that meet customer expectations [8]. In other words, the goal of this study is to look at companies that use DevOps to see how it affects the growth of several quality matrices that have been defined. This study demonstrates the benefits of adopting DevOps methods in Agile software development processes, as well as the drawbacks. It also discusses the difficulties of making the switch to DevOps.

II. PROBLEM STATEMENT

The main problem that this paper will address is to explore the challenges and mitigation strategies when adopting DevOps during software development. It's not uncommon for development and operations to clash when a company is providing useful new software to its clients. To resolve the tension between the development and operations teams, the developing idea of DevOps has been presented. DevOps is being used by an increasing number of businesses and organizations. The notion of DevOps is still relatively new, so it's important to know what problems it might help solve as well as how to mitigate them.

III. LITERATURE REVIEW

A. DevOps: A Brief History

When Patrick Debois was working on a project involving development and operations teams, he realized that there had to be a better method to handle the disputes that arise between the two worlds of Dev and Ops. This was in 2007. Agile Infrastructure was a popular topic of discussion during the 2008 Agile Conference in Toronto, thanks to Patrick Debois[8,9]. A small group of people met after the conference to explore ways to bridge the development and operational gaps. During the Velocity conference 2009, John Allspaw and Paul Hammond presented their well-known lecture entitled 10 deployments per day Dev & ops cooperation at Flickr on June 23rd, 2009 [9]. Patrick, John, and Paul met together after exchanging messages on Twitter to talk about DevOps. Patrick concluded that the event needed a name that included both development and operations, therefore DevOpsdays was born [9]. DevOps attracted systems administrators, developers, and managers from all around the globe. Even though attendees dispersed to all regions of the world once the conference ended, the topic carried over to Twitter. Since Twitter only allows 140 characters per tweet, the DevOps hashtag is used instead of the DevOpsdays hashtag. Eventually, the conferences became a recurrent worldwide series produced by the DevOps community that is a key driving force in the field. The #DevOps Twitter hashtag develops into a rich and important information stream on the social media site. As DevOps grows, it enters the workplace, where well-known companies like Target, Nordstrom, and LEGO have embraced it [9].

B. *DevOps*

The concept of DevOps has received little attention in the academic literature [7]. Built on agile development principles, it allows for quick development and deployment cycles [19]. DevOps has no universally agreed-upon definition. According to researchers, DevOps is a software development technique that integrates quality assurance and operations into development methods [10]. DevOps is a corporate strategy that may be used to define more effective cooperation involving software development and infrastructure management experts [11]. According to IBM Cloud, adopting common DevOps technologies enables cooperation between developers, testers, and operators and facilitates the delivery of software continuously by allowing collaborative testing and continuous monitoring across the development, integration, and segmentation environments. Using the right tools may help with anything from version control to infrastructure setup to orchestrating to monitoring to containerization to automation. It's been since 2011 that the DevOps community has created open-source tools like Puppet (for automating the setup of virtual development environments) and Chef (for automating the setup of physical development environments).

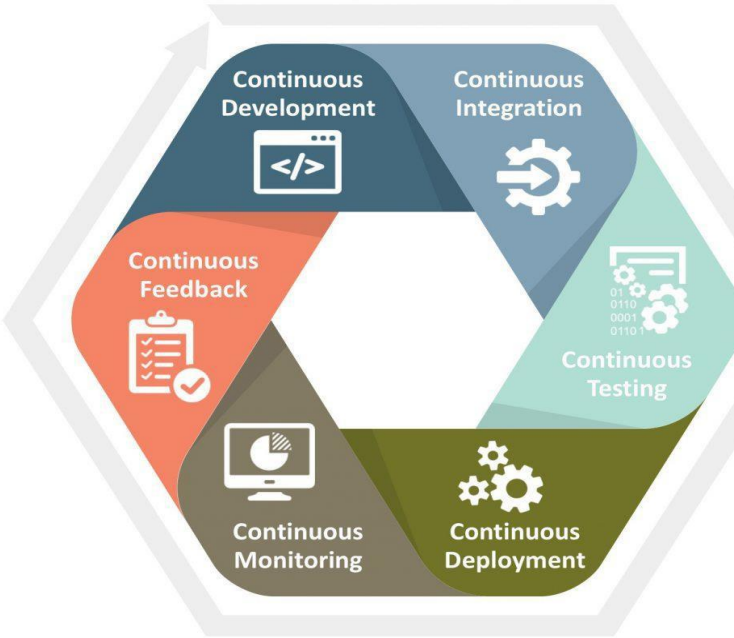

Fig i: Six Phases of DevOps Model

C. *Benefits of DevOps*

As a whole, the strategic goal of DevOps is to get the best possible return on investment while also ensuring high-quality software and meeting customer demands. To allow frequent and rapid software releases, DevOps strives to build a continuous pipeline [11, 12] that includes automated testing cycles to enable continuous software delivery. Using DevOps, you can also respond quickly to client needs that change [12]. With DevOps, developers and operators may collaborate to integrate all organizational processes, streamline testing and quality assurance, and ease the transition between development and operations. DevOps eliminates organizational and cultural problems by integrating development and operations [12] and reduces the cost of fault detection in the early stages [13]. Software defects are instantly fixed early on in the development lifecycle in the DevOps environment due to the continuous deployment of software builds [13].

D. Major DevOps Challenges

The genesis of DevOps may be traced back to the need to eliminate organizational silos to better own and collaborate on delivered products. Development and Operations are the two most important elements of the company area. DevOps is the practice of software development and operations teams working collaboratively from the beginning of the SDLC through deployment and operation. Increasing delivery speed and having greater ownership (and, as a result, higher quality) of the end product are both goals of this strategy [14]. With DevOps, businesses can better serve their consumers by delivering software continuously and with higher quality. Despite its many advantages, deploying DevOps may provide many obstacles as well. DevOps provides some problems to businesses, such as aligning objectives and priorities to encourage cooperation across functional boundaries or changing infrastructure models that are more than a decade old. Before implementing DevOps, it's critical to know what problems to expect and how to overcome them [14].

1. Issues in Security/Development Teams' Communication

Despite their apparent differences, developers and security professionals have similar objectives. As quickly as possible, developers want to get their code out into the wild. Security teams prioritize providing safe apps above speed, with security as their primary goal. This entails assessing a large number of apps before a new version is released. Confusion, delayed delivery, and irritation are all too common when security and development teams don't work together and communicate effectively. Security teams should be included early in the software development lifecycle (SDLC) according to DevOps. However, there is some tension in the initial stages between the development and security teams since developers typically do not understand security concepts or how to deal with security issues. To keep up with the rapid pace of new server deployment, both development and security teams must work together to ensure each server is adequately hardened, has correct logging in place, and so on. DevOps prioritizes high-quality service delivery, but it also needs more robust security measures. As a result, the development team is often unaware of the necessary security measures that should be done. Similarly, security teams often ignore DevOps' automated nature.

2. *The Security Team's Struggle to Keep Up with the DevOps Cycle*

Fast delivery and short development cycles are the emphases of DevOps. A single vulnerability may seriously jeopardize a business; thus, security teams strive to be as comprehensive as possible when examining the security of apps and their surroundings. Because of the need for thoroughness, evaluating the code and its environment might take significantly longer than developing or modifying it. While the goal of DevOps is quick continuous delivery, firms are often compelled to compromise on security to meet deadlines. Speeding up an application at any cost leaves it vulnerable to security breaches and malicious assaults due to ignored dangers and vulnerabilities in the code.

3. Cultural Resistance to Security

Testing for security has traditionally been done after the software development lifecycle (SDLC), just before release. Security teams, on the other hand, are interwoven across the SDLC thanks to DevOps. As the development teams are used to working alone throughout the development stage of the lifecycle, this early integration may cause friction. Management places enormous demands on development teams to produce updates as quickly as possible, and security teams typically see any involvement with the development team as a burden to providing the functionality that management wants to see delivered. Because of this, the development team frequently has to make compromises in terms of the application's security. When it comes to security, many people are concerned that incorporating it early in the process may lead to delivery delays and so ignore the issue.

4. Avoiding Risks Associated with Containers and Other ToolS

When working in a DevOps environment, cloud infrastructure and deployments are regularly used, which makes the application vulnerable to possible security risks if suitable security measures are not implemented. In the DevOps environment, a large number of open-source, immature, and innovative technologies are used [14]. Simple bugs or misconfigurations in the fast-paced delivery pipeline of DevOps may result in spectacular failures, such as corporations disclosing their administrative consoles for their orchestration software, as Tesla did, or even the failure of a whole company. There are several tools that a DevOps team will use, including Ansible, Salt, Chef, Puppet, and many more similar programs and tools. Containers are one of the most regularly used tools/technologies by DevOps teams, and they are also one of the most widely distributed. Containment systems are ultra-lightweight portable packaging platforms that make it easy to install and use software programs. Unfortunately, determining the security of these containers might be challenging for security teams to do so accurately. The use of safe libraries, as well as the creation of adequately protected services, are being considered. Are secrets being kept and maintained securely? As a result of not completely addressing these concerns or providing satisfactory answers, the usage of containers may bring new dangers into a company [14]. While containers constitute a significant problem, it is also important to address and protect all of the tools connected with deployment, since they are critical in developing and maintaining the deployed application and environment. All too frequently, the keys to the kingdom are connected with orchestration software, and this software must be thoroughly evaluated to verify that it is safe to use and secure to operate.

5. Poor access controls and secret management procedures

Secrecy management and strict access constraints are critical in the context of highly automated development and deployment processes. API tokens, SSH keys, privileged account credentials, and other such information are examples of secrets. Containers, services, personnel, and a plethora of other entities might make use of them [14]. These kinds of essential passwords and keys are commonly mismanaged (and hence exposed), making them a popular target for attackers on the internet. Additionally, to maintain a seamless and efficient workflow, DevOps teams often provide practically unfettered access to privileged accounts such as admin, root, and so on. It becomes substantially more likely that these excessive rights may be exploited when numerous users use and exchange passwords for confidential accounts, as well as when processes execute with high access.

E. DevOps Mitigation Strategies in Software Development

DevOps may introduce security risks and cause compatibility concerns across multiple teams involved in the software development lifecycle (SDLC), however, there are techniques to overcome these difficulties. Consider applying the following strategies in your business to

increase DevOps security while maintaining a balance between diverse teams and the need for agility [14,15].

1. Enforce Security-Oriented Policies

When it comes to creating comprehensive security environments, the establishment of governance and effective communication are essential. You should develop a set of cybersecurity processes and rules that are simple, easy to understand, and transparent, covering topics like access restrictions, code review, firewalls, and configuration management, among others. Security rules should be adhered to by the DevOps teams, and they should work together constructively to create a secure application. In addition, the notion of "infrastructure as code (IaC)" is a foundation of the DevOps methodology and practice. Virtual machine setup and configuration, network configuration, load balancing, and connection topology are all defined as code that is versioned in the same way that the DevOps teams do for their application code [15]. IaC stands for Infrastructure as Code. While this may appear to be a frightening prospect, it has the potential to be extremely beneficial because code (the infrastructure, the servers, the routers, the configurations, and so on) can be reviewed and assessed more easily to ensure that the environment is in the proper hardened configuration for the situation15] [. Similar to the premise that the same code produces the same binary when an IaC model is applied, it creates the same environment as it did before it was implemented.

IaC overcomes the issue of environment drift in the delivery pipeline by using distributed computing. Teams would have to manually manage the parameters of each deployment environment if there was no IaC. Inconsistencies across various environments may cause problems during the release process. With the integration of IaC, DevOps teams can more simply administer and maintain the security of their apps and environments than they could before [15,16]. To maintain infrastructure and deliver applications safely, quickly, and at scale, DevOps teams that incorporate IaC work together and use a consistent set of security policies and tools to collaborate.

2. Adopt a DevSecOps model of operation

Effective DevOps security may be accomplished by promoting cross-functional partnerships across the full DevOps lifecycle, from development to operations. When it comes to achieving similar objectives such as increased security, DevOps teams should not only work together but also actively engage throughout the development lifecycle. Security should not be the primary responsibility of a single team, but rather should be a deeply ingrained part of the organization's whole culture. The term "DevSecOps" refers to the practice of embedding security into the culture of an organization [16]. It is a culture inside businesses in which everyone accepts responsibility for complying with security rules and procedures.

DevSecOps is a collection of cybersecurity functions and governance that work together to lower the likelihood of security breaches caused by lax account restrictions and other security vulnerabilities. It goes well beyond the use of technological tools and software to ensure that security is seen as a fundamental organizational concept. DevSecOps encourages teams of all sizes to become familiar with fundamental security concepts. It is recommended that all members of a team get some basic security training [16]. In addition to formal training, developers should get familiar with the usage of automated tools and software to do rapid security checks on their code. Security experts should also be able to write code and interact with APIs, which will allow them to script and automate security checks, which will be very useful in IaC environments. Involvement of security teams in the production of certified and hardened versions of infrastructure for usage by the development team is possible. They may also enforce the settings by automatically monitoring the infrastructure code, which they can do using automated tools [16].

3. Automate processes to increase speed and scalability

When it comes to developing safe apps and secure environments, automation is critical to success. Automation helps to decrease the risks associated with human mistakes, as well as the vulnerabilities and downtime that are connected with them. It becomes more difficult for the security team to keep up with the DevOps team if they do not have access to automated security tools and procedures. There are a variety of activities that may be automated, including continuous integration, vulnerability assessments, privileged credentials/secrets monitoring, and code analysis, to name a few. Selecting automated tools and procedures is as important to automating your DevOps as deploying them.Automated technologies that are used to create a secure DevOps process should have the following features:

- Must be simple to comprehend and manage
- Does not need the use of security expertise
- Not have a high proportion of false positives when it comes to concerns
- Be incorporated into the continuous integration and delivery pipeline

The objective is to make it simpler for the DevOps team to work more efficiently and effectively, not to overwhelm them with hundreds of tools or procedures that are unfamiliar to them from their current working environment. With a narrower gap between security and DevOps teams' speeds, it will be simpler to establish security as a key concept in your organization's development process.

4. Identify and Manage Vulnerabilities Successfully

Incorporating security into the software development lifecycle (SDLC) from the outset makes it easier to spot defects and vulnerabilities early on. To deal with the detected vulnerabilities, you'll need an effective vulnerability management system that can monitor and prioritize the many approaches that should be taken to fix each issue (remediation, acceptance, transfer, etc) [17]. In a vulnerability management program, there are four major steps to consider:

- Determine the criticality of an asset, the owners of the asset, the frequency of scanning, and the schedule for repair that can be achieved within a reasonable timeframe.
- Locate and inventory assets on the network as they are discovered.
- Identify any vulnerabilities in the assets that have been detected.
- Identify and address any vulnerabilities that have been discovered.

When you first start working with a vulnerability management application, you may realize that you have a rather high vulnerability score, which results in time-consuming repair cycles [17]. The important thing is to demonstrate development from quarter to quarter and year to year. With increased familiarity and education about the vulnerability management program, teams should be able to reduce the amount of time spent remediating vulnerabilities and lowering their vulnerability scores. These programs continually embrace and comply with the organization's newest risk reduction targets.

5. Implement a Successful DevOps Secrets Management Strategy

Passwords, keys, and other private information are examples of secrets that need to be guarded with extreme care. DevOps teams have regularly resorted to bad secret management practices in the pursuit of rapid automated deployment, such as keeping passwords in files inside containers. Team members might often take shortcuts that expose highly sensitive passwords and keys in the rush to install automated software as quickly as possible. For efficient DevOps secrets management, you should delete sensitive data such as credentials from code, files, accounts, services, and other platforms and tools, as well as from other platforms and tools [17]. To do this, the passwords are removed from the code and stored in an off-site password safe when they are not in use by the user. When you are not using your passwords, you may store them in solutions such as Cyberark, Azure key vault, AWS secrets manager, Thycotic Secret Server, and other similar products. Access to passwords may be controlled by using a centralized password safe. APIs may also be used to acquire control over code, scripts, files, and embedded keys by incorporating them into the system.

6. Implement an effective Privileged Access Management system

It is possible to dramatically minimize the number of opportunities by limiting privileged access to the account. End-user PCs will no longer be able to access administrative or privileged accounts as a result of this change. One should keep a close eye on all sessions using privileged accounts to make sure they're real and compliant with your company's policies. In addition to restricting access for development teams to select development, production, and management systems, enforcing a privileged model involves [18]. Even so, they should still be able to generate images and machines from authorized templates, deploy them, change them, and fix vulnerabilities in the system with the access and permissions necessary. Consider the implementation of a cutting-edge privileged access management system like OpenIAM, which can automate the control, monitoring, and auditing of privileged access all through the development life cycle [17, 18]. Privileged credentials and secrets should be tracked throughout their entire existence.

IV. FUTURE IN THE U.S

The future of DevOps and AI in the U.S is going to focus more on intelligent systems that train on data and learn to complete different functions on their own. Artificial intelligence (AI) has the potential to make DevOps more efficient. It may improve performance by allowing quick development and operation cycles, and by providing a great user experience with these features as well. In the DevOps system, machine learning algorithms may make it easier to gather data from diverse sources [18]. As artificial intelligence (AI) and data science become more widely used in the business, DevOps will continue to grow in importance [19]. To help enterprises accelerate and enhance their AI solutions, DevOps for AI offers a viable approach. It facilitates the preparation of data and the building of models while also introducing standardized methods to enable AI on a large scale [18]. Despite this, AI operationalization is seldom addressed, despite the many advantages it offers over traditional approaches. The moment has come to prioritize the implementation of artificial intelligence as a strategic goal for the company.

V. ECONOMIC BENEFITS IN THE UNITED STATES

This study will aid the United States by shedding light on how DevOps issues may be overcome to the advantage of a wide range of enterprises' operations. According to CB Insights' most recent estimates, the open-source services market will be worth more than $17 billion in 2019 and approximately $33 billion by 2022. Non-open-source firms like Microsoft and Google routinely contribute to GitHub projects, as do Intel and Facebook [18]. In 2018, Google workers contributed a total of 5,500 hours of volunteer work. Smaller, independent initiatives have benefited greatly from many of these donations, as well. Google's open-source software projects such as Kubernetes, Istio, and Knative, which are in great demand, have the most support, according to the survey [19]. Independent developers will continue to participate as corporate-sponsored initiatives gain popularity. This demonstrates the importance of the giants stepping up to assist the open-source movement in expanding. Over 19,000 people have contributed to the Visual Studio Code project at Microsoft. Because of the free developer input and direct user feedback provided by the hundreds of contributors, these tech titans can profit [20]. Because of this, businesses can produce better software more quickly. The use of open-source software is already commonplace, and it has a promising future.

VI. CONCLUSION

This research paper conducted a review of the literature and industry to determine the problems and mitigation techniques associated with using DevOps during software development. One of the main goals of this paper is to offer a thorough analysis and description of the problems that will arise while using DevOps. Using literature review, this study develops evaluations that highlight the problems of implementing DevOps throughout the software development process, as well as the methods for overcoming these challenges. In adopting DevOps, the two checklists might assist individuals to avoid issues, manage risks effectively, and budget for the unavoidable obstacles of utilizing DevOps. A brighter future has been ushered in by DevOps, which provides effective solutions that help in speedier delivery, boost cooperation among teams, and cultivate an Agile working environment. While the advantages of DevOps are many, it also brings with it new obstacles. The challenge many firms have in incorporating security into the DevOps approach is one of the most obvious problems with DevOps. However, security must be considered throughout the process. Implementing security in DevOps as early as possible can aid in the rapid identification of vulnerabilities and the remediation of operational flaws before they become a problem. Embedding security early in the DevOps lifecycle helps to keep it in place and operational throughout the product's lifespan. By using this security measure, the code will be protected against cybersecurity threats and data breaches.

REFERENCES

1. V. Gupta, P. Kapur and D. Kumar, "Modeling and measuring attributes influencing DevOps implementation in an enterprise using structural equation modeling", Information and Software Technology, vol. 92, pp. 75-91, 2017.
2. J. Humble, D. Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation (Adobe Reader). Pearson Education; 2010 Jul 27.
3. V. Lalsing, S. Kishnah, and S. Pudaruth, People factors in agile software development and project management. International Journal of Software Engineering & Applications, 3(1), 2012, p.117.
4. A. Benoist, 2013 "Influence of release frequency in software development.", [available at] https://hal.archives-ouvertes.fr/hal-00832011v2 [17] Iso.org, "ISO/IEC 9126-1:2001 - Software engineering -- Product quality -- Part 1: Quality model", 2016. [Online]. Available: http://www.iso.org/iso/catalogue_detail.htm?csnumber=22749. [Accessed: 05- Feb- 2016]
5. M. Ammar., "Application of Artificial Intelligence and Computer Vision Techniques to Signatory Recognition", Information Technology Journal, vol. 2, no. 1, pp. 44-51, 2002.
6. H. Izadkhah, "Transforming Source Code to Mathematical Relations for Performance Evaluation", Annales Universitatis Mariae Curie-Sklodowska, sectio AI – Informatica, vol. 15, no. 2, p. 7, 2015.
7. H. Papadopoulos, A. Andreou and M. Bramer, Artificial Intelligence Applications and Innovations. Berlin, Heidelberg: IFIP International Federation for Information Processing, 2010.
8. V. Sugumaran, Distributed artificial intelligence, agent technology and collaborative applications. Hershey, PA: Information Science Reference, 2009.
9. L. Lopes, N. Lau, P. Mariano and L. Rocha, Progress in Artificial Intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.
10. G. Simov, "Artificial intelligence and intelligent systems: the implications", Information and Software Technology, vol. 32, no. 3, p. 229, 1990.
11. M. Huttermann, "Beginning DevOps for Developers", DevOps for Developers, vol., 2012 , pp. 3-13.
12. H. Salzman, "Engineering perspectives and technology design in the United States", AI & Society, vol. 5, no. 4, pp. 339-356, 1991.
13. L. Rendell, "A new basis for state-space learning systems and a successful implementation", Artificial Intelligence, vol. 20, no. 4, pp. 369-392, 1983.
14. R. Conejo, M. Urretavizcaya and J. Prez-de-la-Cruz, Current topics in artificial intelligence. Berlin: Springer, 2004.
15. T. Bradley and T. Bradley, "Why DevOps means the end of the world as we know it", TechSpective, 2016. [Online]. Available: https://techspective.net/2015/08/16/why-devops-means-the-end-of-the-world-as-we-know-it/.
16. D. Linthicum, "What is DevOps? DevOps Explained | Microsoft Azure", Azure.microsoft.com, 2016. [Online]. Available: https://azure.microsoft.com/en-us/overview/what-is-devops/.
17. L. Iliadis, I. Maglogiannis and H. Papadopoulos, Artificial intelligence applications and innovations. Berlin: Springer, 2012.
18. Y. Jiang, "Analysis on the Application of Artificial Intelligence Technology in Modern Physical Education", Information Technology Journal, vol. 13, no. 3, pp. 477-484, 2014. [11] Y. Nakajima, M. Ptaszynski, H. Honma and F. Masui, "Automatic extraction of future references from news using morphosemantic patterns with application to future trend prediction", AI Matters, vol. 2, no. 4, pp. 13-15, 2016.
19. K. Hirasawa, "Trend on application of AI technologies to industry. From the latest international workshop on AI applications.", IEEJ Transactions on Industry Applications, vol. 108, no. 10, pp. 868-871, 1988.
20. L. Bass, I. Weber and L. Zhu, DevOps: A Software Architect's Perspective. Pearson Education,Inc., 2015.