Hemanth Swamy, 2022, 9:2
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology

An Open Access Journal

Software Quality Analysis in Edge Computing for Distributed Devops Using ResNet Model

Hemanth Swamy
Senior Software Engineer
Motorola Solutions

Abstract- The phrase "development operations" (DevOps) refers to a more modern trend in which several disciplines work together to speed up and enhance the delivery of IT solutions that provide value to businesses. The advantages of DevOps, such as faster development times, more stable environments, better teamwork, and better communication, are attracting many software companies. Despite the importance of DevOps principles for software firms, the management of these activities has received less attention in the literature. An efficient framework for managing DevOps activities is our study's overarching goal. In order to find out how to execute DevOps well, we ran an empirical research using the open-source HELENA2 dataset. In addition, we have developed a model for forecasting for DevOps implementation using the RESNET prediction algorithm, which is furthermore compared to machine learning models like SVM, ANN, and RF. Consequently, we determined that checking system log files, automated tests, making sure that automated tests cover a lot of ground, and using a continuous deployment plan are the most typical tasks. Additionally, the Extended Kalman Filter (EKF) is used to assess the quality. This research gives us the green light to keep digging for answers on how to improve quality of software in DevOps environments.

Keywords- Continuous Quality, DevOps, Feature Dependent, Software Quality, Edge and Machine Learning Model

I. INTRODUCTION

A philosophy and set of techniques known as "DevOps" encompasses both development and operations. Its primary goals are to increase software quality, decrease software development lifecycle, and remove obstacles to software evolution. Development, Security, along with Operations (DevSecOps) is an emerging version of DevOps that aims to include security principles into the process of DevOps. It was created in response to the growing need for safe software products. Software quality is often cited as an outcome of DevOps implementation in literature. Although there is some evidence that DevOps improves the quality of goods, there is less detail about the specific actions or best practices that do so [1]. Constant innovation is essential to meet the aspirations of corporations operating in today's industrial and market-driven business climate. Programmers, testers, DevOps engineers, and project managers are all going to be involved in this. In order to properly account for the most recent solutions, procedures, standards, and methods, it is necessary to mix whole software code, which is associated with the dispersion of different directions [2]. Many QA groups now use the software quality model as a framework for analyzing software products and ensuring their quality. For DevOps to operate, the CI/CD process

© 2022 Hemanth Swamy. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly credited.

builds a pipeline, testing jobs are added to it, and automating of testing is encouraged so that products may be developed more efficiently. However, traditional quality metrics like bug ratio and test coverage rate have been employed by most DevOps initiatives. The quality model allows for the visualization and management of the progress toward the released system's needed quality by classifying the quality assured through CI/CD pipeline testing according to quality attributes [3]. There is a pressing need to identify more effective development models due to the rapid evolution of software engineering. DevOps with its security-focused addition the goal of using DevSecOps was to increase code dependability and speed up the entire development cycle. Nonetheless, security holes can still exist due to infrastructure vulnerabilities and many third-party libraries. In addition, standards and regulatory compliance need thorough infrastructure hardening and secure software [4, 5]. Improved software efficiency, reliability, and quality may be achieved via DevOps, a method that brings together development and operations teams. During various phases of software development, DevOps employs a diverse collection of automation technologies [6, 7].

1. Contributions

In order to create high-quality products, software development companies are now embracing DevOps principles. Since the word "DevOps" has not been defined, the concepts, procedures, and measures used to evaluate its efficacy have evolved significantly. If the DevOps methods are used efficiently and successfully, various advantages of DevOps may be obtained. The purpose of this research was to catalog the DevOps methods that help bring about a high degree of DevOps. The HELENA2 dataset has been analyzed using a qualitative and interpretative method. Using degrees of characteristics, the ResNet model classifies DevOps practices into three categories: high, low, and medium. Each category is utilized to determine the quality calculation. Companies that consistently employ DevOps have a leg up when it comes to reaching their objectives, according to the quality score across all categories.

When contrasted with other models, our software quality approach stands out because

- Features-level non-functional requirements (such as deployability, containerization, virtualization, and elastic service provisioning) are defined, and non-functional requirements related to DevOps are also supported. Quality factors are operationalized using specified measures, instruments, and applicable constraints.
- Provides a collection of programming interfaces that are language-neutral and can be customized to fit typical DevOpstoolchains, enabling the implementation of the quality model.

Here is the structure of the paper: In Section 2, we go over the basics of DevOps, the Edge OPS architecture, and how to measure software quality. The research model and technique used in the study are described in Section 3. The data analysis, findings, and suggestions are all reported in Section 4. In Section 5, we draw conclusions and summarize the study on how DevOps might increase quality.

II. RELATED WORK

Investigating the DevOps processes that influence software quality is the aim of the ongoing effort [8]. To do that, they combed through the literature for references to software quality-related tasks. To further investigate the chosen activities' validity, authors examined specific instances of DevOps deployments. Consequently, they determined that checking system records, automating evaluations, making sure that automated tests cover a lot of ground, and using a continuous deployment plan are the most typical tasks. Having development progress measurements and making sure team members have the necessary abilities are the less common tasks. That research gives us the green light to keep digging for answers on how to improve quality of software in DevOps environments. Presented a multi-level edge computing–based distributed & anonymous data collection (DaaC) platform in [9]. To reduce end-to-end latency and packet loss, this system uses a network of level-one device edges (LOEDs) to share

the collected data and boost quality of service. In order to transfer data from LOEDs to servers in the cloud, mobile sinks are used. The portable sinks are safeguarded before data collecting begins by registering using a level-two edge device. Requests to acquire data from mobile sinks are protected from prying eyes by using group-based signatures. Our proposed approach enhances QoS via dispersed data transfer, as shown experimentally. By thinking about how to best deploy neural network models onto the edge nodes, suggest a task model offloading technique in [10]. Using the enhanced ant colony method, a flexible task scheduling method is also developed to optimize work assignment in an adaptable way. They are the basis of a distributed neural network-compatible collaborative cloud-edge computing architecture. The cloud and edge computing may operate together in harmony thanks to the procedures brought up by this architecture. The framework may increase task accuracy, decrease energy consumption, and delay, according to the simulation findings. The paper suggests an edge-computing-based method for detecting faults in distributed power distribution, which can improve power supply reliability, reduce power outages, increase user satisfaction, and speed up processing times for distribution faults [11]. The second part of the paper lays out the foundation for using wavelet transforms to identify signal singularities, and it suggests a way to analyze power signal fault signals using these techniques. By using wavelet transform to its fullest potential in fault signal evaluation, it not only overcomes the limitations of the conventional Fourier transform approach but also provides instances to back up its claims. An assessment system based on the edge computing CROSS index fault detection framework is provided, taking into consideration the important needs of edge computing such as agile connectivity, business real-time, data efficiency, application intelligence, security, and privacy protection. Work together on projects, create and monitor project deliverables such code in [12]. as well as monitor the project artifacts' deployment and release processes and build scripts. For administering your IoT Edge installations, Azure DevOps is a good option because of its support for containers and its interaction with Docker. On top of that, there are specialized processes for developing IoT Edges that streamline the release and build procedures. Cloud DevOps in Azure is a vast subject. Our emphasis in this chapter will be on the processes and workflows needed for developing and implementing Azure IoT Edge applications, once we have established some of the core principles. Future study may potentially investigate other methods or look at ways to improve the suggested hybrid model. Agile software application creation concerns pertaining to software testing are discussed in chapter [13]. The preferred tools for contemporary software development now fall under the Agile methodology umbrella, which includes Extreme Programming, Scrum, and Development and Operations, sometimes known as DevOps. The emphasis in these approaches is on incremental and iterative development, in which requirements and solutions are both changed by the work of interdisciplinary teams. Each level of development must provide high-quality results achieved via thorough testing for such approaches to be successful. Within the framework of a software development lifecycle that is built on Scrum and DevOps, that chapter explains the basics of software testing. Code versioning, ongoing integration, automated functional testing, static code analysis, and continuous deployment are some of the preventative quality techniques and recommendations for software development included in the [14] article. The goal of that software development case study is to showcase the best practices used by the Smart Campus Ecosystem. Method: software development efficient methods based on XP and DevOps are surveyed. The selection of implementation tools has a direct impact on the quality of software development. A number of technologies are used in their examination of the UNIAJC Smart campuses setting. In that piece, the results of the rollout are detailed. Resultant state: The results are shared after the preventative quality strategy has been exposed and tested. In conclusion, the preventive quality strategy aids in improving quality assurance results by giving development teams vital information for reworking and improving source code during developmental runtime or immediately thereafter. First, the research aims to learn more regarding the

challenges of DevOps by conducting a methodical examination of literature and expert opinion surveys. Second, the study plans to rank the obstacles found throughout the review using a fuzzy analyzing hierarchy method. The study's findings provide a prioritization-based classification of the DevOps problems as well as a compilation of significant obstacles encountered by software companies while executing DevOps. Since FAHP helps clarify for practitioners what aspects of DevOps have an impact, it is a fresh approach to that field of study. It is their hope that scholars and practitioners in the software sector will be able to use that study's findings to inform future iterations of DevOps strategy development and revision.

III. DEVEDGEOPS

What we call "DevEdgeOps" is really the combination of traditional DevOps methods and concepts with the possibilities and threats that come with using computers at the edge. The goal is to maximize the benefits of edge computing by using DevOps best practices and modifying DevOps to work on the edge. The realization that a comprehensive strategy is required at the edge area is the driving force behind the rise of DevEdgeOps. Traditional DevOps practices, while highly effective in cloud-centric scenarios, fall short when applied directly to the edge.

Fig.1. ML based Devops for Edge Nodes

The "one-size-fits-all" approach no longer suffices in the face of the unique challenges posed by the edge, including scale, connectivity, security, and device diversity. DevEdgeOps serves as the bridge that connects these two technological powerhouses, allowing organizations to benefit from the agility and automation of DevOps while navigating the intricacies of edge environments. As an example, let's consider the DevOps concept "Shift Left" and its adaptation at the edge.

EdgeOps integrates DevOps, MLOps, and DataOps into a single, cohesive, extensible platform. EdgeOps manages continuous and reliable operation of AI/ML analytics at scale and across environments – edge, on-premise, and private cloud. Build data-centric applications, analytics, and insights that connects data from across your organization.

EdgeOps Features

Distributed Database

With an edge-first, data mesh architecture, maintain data close to the source allowing data and AI/ML features to remain geographically distributed with localized ownership while providing a single pane of glass view requiring no database administration expertise.

Integrated MLOps

Perform exploratory data analysis, track model experiments, train/version models, deploy them directly on edge hardware, and continuously monitor their performance.

Operational Pipelines

Extend beyond simply deploying models, run and manage end-to-end data flows in ultra-lightweight containers. Centrally managed but deployable anywhere, connect to any data source to unlock real-time analytics.

- Multi-Source distributed data collection, storage, and analytics
- Edge-native AI/ML inference with integrated MLOps
- Ultra-lightweight, low-latency, run-anywhere Operational Pipelines
- Low-code graphical pipeline builder
- Distributed Database with Data Mesh architecture
- Self-managed, self-contained, and user owned
- Seamless scaling with modern micro services architecture

- Centrally managed pipeline orchestration and versioning

IV. PROPOSED WORK

1. System Model

One of the primary challenges of any software development strategy is ensuring quality. An important part of DevOps is software quality assurance, which helps in finding and fixing bugs early on, which in turn allows for more dependable software releases. Finally, this article uses DevOps to forecast software quality in Edge Computing after exploring the link between different DevOps methods and their effects on software quality. In this context, DevOps procedures serve as independent variables, and software quality as a dependent variable. Utilizing the Pearson Correlation coefficient, we determined how well these variables were related to one another. The study's variables were shown to be positively and significantly correlated with one another. To put the theories to the test, we used the ResNet model. Software quality is most affected by automation in DevOps approaches, according to our findings.

2. Software Quality Prediction Using ResNet Model

By providing insights and recommendations based on data analysis along with predictive modeling, AI helps DevOps teams make better choices. Instead of relying on gut feelings or wild guesses, businesses may now make choices backed by hard facts. To identify patterns and anomalies that may indicate problems, AI algorithms may examine log files, system efficiency indicators, and user activity data. These systems are able to extract subtle relationships from massive, complicated datasets by using machine learning. Testing and deployment are two examples of repetitive tasks that may be greatly improved with the help of AI-driven automation. Software releases are now faster and more reliable thanks to this method, which also saves time by reducing the number of errors made. The capacity to automate these mundane operations is a major strength of AI DevOps. Consider the field of software testing. Algorithms powered by AI can independently create test cases and evaluate their results. Thanks to this automation, teams are free to concentrate on other aspects of the project while testing takes much less time and effort. In addition, it improves program quality by finding mistakes and bugs that humans may miss. To automate decision-making, optimize performance, and anticipate system breakdowns, DevOps heavily use machine learning algorithms for predictive analytics. They are adept at looking for trends in past data and using that knowledge to make educated guesses or choices.

Software Quality Measures

- require measure('Metrics.techDebt', mobileImageUpload) is less than or equal 10 days
- require measure('File.size', 'test.jar') is less than 10 megabytes
- prevent measure('ServiceInstances.count', mobileImageUpload) is larger than 10 per hour
- prevent measure('Deployment.time', mobileImageUpload) is larger than 1 minute
- require measure('ServiceInstances.failOverTime') is less than 5 seconds
- require event('Errors.general', mobileImageUpload) occurred less than 10 times within the last 2 hours,
- require event('Errors.severe', mobileImageUpload) has occurred less than 2 times in the last 2 hours,
- require execution(mobileImageUpload) is completed within 1 second in 5% of executions
- Feature mobileImageUpload { operation from endpoint '*/uploadImage/', source class 'at.jku.se.galleryApp.services.ImageUploadService'.

The accuracy of the procedure is compromised due to the presence of several irrelevant and redundant characteristics in the retrieved set of features, which must be eliminated. In order to do this, we are using a feature selection method that eliminates unnecessary and duplicate functionality. The attention module learns the feature weights that are associated with software quality and is hence responsible for feature extraction. It is possible to calculate the attention layer's output as,

$$Atn_i(Z) = Q_i(Z).F_i(Z) + F_i(Z) \qquad (1)$$

Where, $Q_i(Z)$illustrates the weight assigned to concentrate on and $F_i(Z)$show the characteristics. In the attention layer, we calculate the relationships between the characteristics so that we may extract more useful information from them,

$$R(F_1; F_2) = \int_{F_1} \int_{F_2} p(F_1, F_2) \log \frac{p(F_1, F_2)}{p(F_1)p(F_2)} dF_1 dF_2 \qquad (2)$$

Where $p(F_1, F_2)$to represent the probability function including the attributesF_1 and $F_2$and$p(F_1)$, $p(F_2)$standing for the distinct marginal density functions, respectively. By decreasing the size of the original patch, which impacts the original information needed for categorization, the inception layer discovers certain traits thoroughly.

Classification

The characteristics that were chosen are then used to classify the data. The purpose of this procedure is to determine whether the provided unknown log is benign or malicious. This categorization is carried out by the softmax layer, which computes the cross entropy to ascertain the output loss, which may be expressed as,

$$Loss\ (x) = -\sum_k \log(st(x)_k)\, q_i \qquad (3)$$

Where the softmax$st(x)$ of a vector can be formulated as,

$$st(x)_k = \frac{e^{x_k}}{\sum_m e^{x_m}} \qquad (4)$$

Increasing the score that corresponds to the logs is how the categorization is done via this. The trimming of logs is done to move them to the next step of categorization after they are designated as log quality.

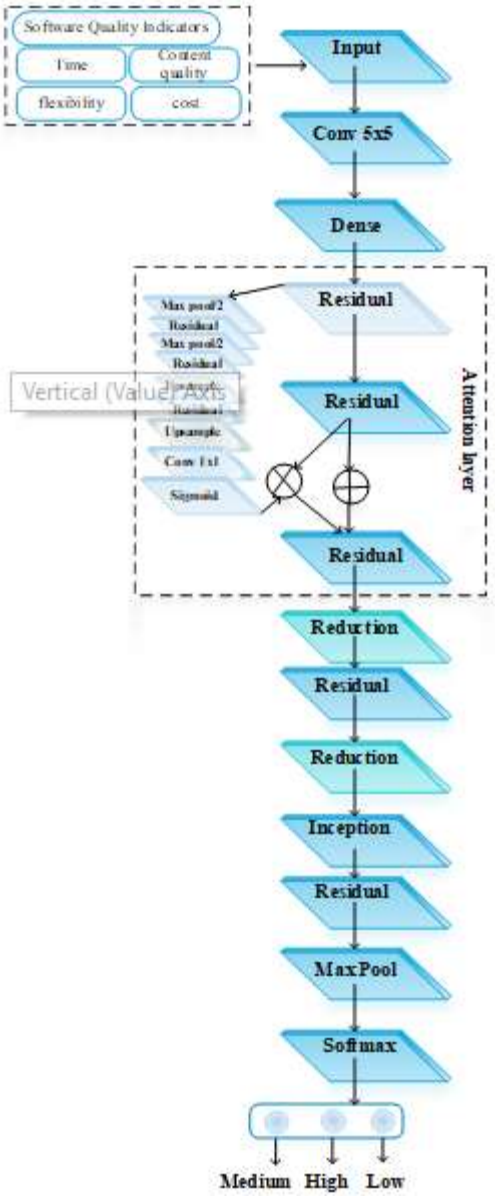

Fig.2 Attention based ResNet

This article provides an overview of our ongoing research on how to operationalize software quality models, a problem that prevents DevOps-driven projects from conducting continuous quality checks and evaluations. To add insult to injury, existing software quality models don't explain software quality within its pertinent functional scope since they don't clearly combine non-functional and functional criteria. Our study will address these restrictions by developing a DevOps-specific quality model that can be automatically operationalized utilizing standard toolchains. We have already

interviewed experts from 11 different firms to gather the most non-functional needs connected to DevOps. Hence in this paper, Ensemble Kalman filter is detected which is defined as follows,

$$N_n = \sum_{i=0}^{n} M_d(N) \quad (5)$$

Where, M_d represent the minimum distance between the nodes which is known as nearest the node. First, the various instruments are used to operationalize the various measurements in our DevOps-centered quality model. This allows us to automatically analyze nonfunctional criteria that are based on features. The next step is to link the actual measurements to the source code along with deployment artifacts, and then aggregate and summarize them according to the quality model. Each non-functional criterion defined in the quality framework for a piece of software is reviewed for completion and the level of completion is maintained in a centralized database for the continual assessment of software quality.

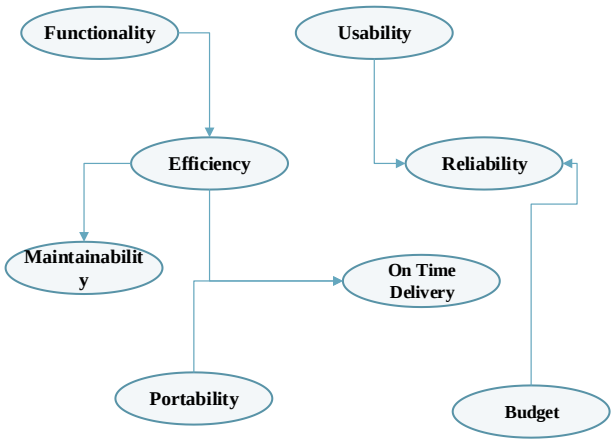

Fig.3 Indicators to Differentiate Software Quality Levels

First project the state which is defined as follows,

$$= f(x_{k-1}, v_k, 0) \quad (6)$$

Where, y_k represent the initial node of the environment. And then calculate the error covariance using the EKF which is defined as follows,

$$E_k = B_k Q_{k-1} B_k^T + WkP_{k-1} W_k^T \quad (7)$$

The kalman gain is calculated as follows,

$$K_g = Q_k H_k^T (H_k Q_k H_k^T + V_k S V_k^T)^{-1} \quad (8)$$

After calculating the kalman gain then the nearest node is estimated as follows,

$$y_k = y_k + K_g(Z_k - g(y_k) \quad (9)$$

The following updates the error covariance,

$$Q_k = (1 - K_g(y_k) Q_k' \quad (10)$$

This equation is representing the updation of error covariance updation. It is also detecting the intrusion severity. If the error covariance is high then the intrusion occurs frequent. If it is low then it will represent the intrusion severity level is rare. The severity is known by using the training dataset. Check whether the given intrusion is occurring frequent or rare. If it is frequent then the intimation or alarm is send to the entire software for ensuring the security. If it is rare the intimation is only send to the nearest node. For frequent quality alarm is defined as follows,

$$AG = \frac{\sum_{S>Th} Th_j H_j A(E_n)}{\sum_{S>Th} Th_j H_j} \quad (11)$$

Pseudocode for EKF
1. INPUT: Log quality software(M_n)
2. OUTPUT: Alarm generation
3. Begin
4. Initialize (M_n)
5. Perform state prediction using eqn
6. for every M_n do
7. Calculate error covariance using eqn
8. Compute kalman gain using eqn
9. Update the state estimation using eqn
10. end for
11. Calculate severity level
12. if (S>Th)
13. Generate alarm to every user in the network
14. else
15. Generate alarm to the user
16. end

Where, AG represents alarm generation and Th represent the threshold value and $A(E_n)$ represent

the alarm generated to entire node in the network and H_j represent the number of nodes in the network and s represent the severity level. For rare intrusion, the alarm is generated to the nearest node (N_n) which is define as follows,

$$AG = \frac{\sum_{S>Th} Th_j H_j(N_n)}{\sum_{S>Th} Th_j H_j} \quad (12)$$

V. RESULTS & DISCUSSION

We will do a comprehensive case study using a business associate and expert interviews to verify our approach to assessing non-functional needs that are reliant on features. To determine if the technique is suitable and applicable for continuous software enhancement for ineffective needs on feature level, this validation primarily seeks to collect empirical data.

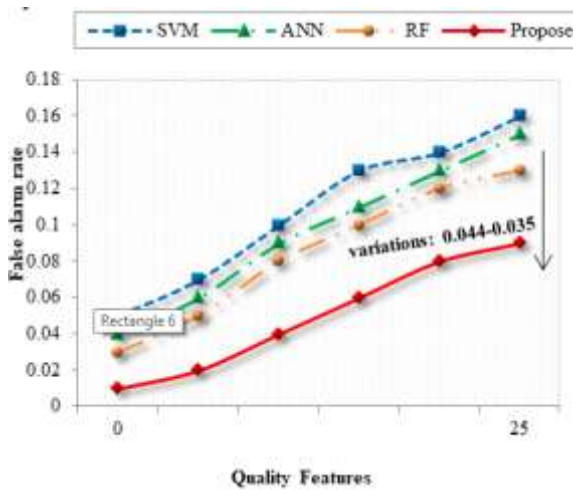

Fig.4. False Alarm Rate Performance

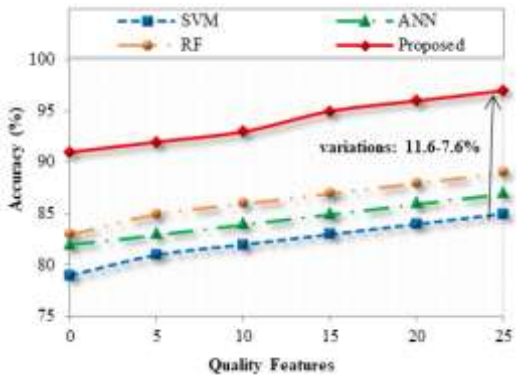

Fig.5. Accuracy vs. Quality Features

In this paper, we begin with the DevOps conceptual structure, conduct research using machine learning, and layout an intelligent DevOps platform. This platform will help engineers analyze large amounts of multifarious method notifications, promote software quality development toward EdgeDevops, and increase the effectiveness of DevOps engineers.

VI. CONCLUSION

Our study's key contribution is a DevOps-centered software quality framework that can be operationalized, allowing us to define and assess non-functional requirements that rely on features. We will verify it via actual instances and conversations with our industry partner, and it can be incorporated into popular DevOps toolchains. Software creation and delivery have taken a giant leap ahead with the integration of AI and DevOps. A competitive advantage in today's fast-paced digital market may be yours with this integration, which improves software quality and dependability. Improved software lifecycle efficiency, accuracy, and reliability are the results of transforming DevOps with AI technologies such as Robotic Process Automation and Machine Learning.

REFERENCES

1. Meedeniya, D.A., Rubasinghe, I.D., & Perera, I. (2020). Artefact Consistency Management in DevOps Practice. Advances in Systems Analysis, Software Engineering, and High Performance Computing.
2. López-Peña, M.A., Díaz, J., Pérez, J.E., & Humanes, H. (2020). DevOps for IoT Systems: Fast and Continuous Monitoring Feedback of System Availability. IEEE Internet of Things Journal, 7, 10695-10707.
3. Díaz, J., Pérez, J.E., López-Peña, M.A., Mena, G.A., & Yagüe, A. (2019). Self-Service Cybersecurity Monitoring as Enabler for DevSecOps. IEEE Access, 7, 100283-100295.
4. Li, Y., Qi, F., Wang, Z., Yu, X., & Shao, S. (2020). Distributed Edge Computing Offloading Algorithm Based on Deep Reinforcement Learning. IEEE Access, 8, 85204-85215.

5. Rouf, Y., Mukherjee, J., Litoiu, M., Wigglesworth, J., & Mateescu, R. (2021). A Framework for Developing DevOps Operation Automation in Clouds using Components-off-the-Shelf. Proceedings of the ACM/SPEC International Conference on Performance Engineering.
6. Batra, P., & Jatain, A. (2020). Measurement Based Performance Evaluation of DevOps. 2020 International Conference on Computational Performance Evaluation (ComPE), 757-760.
7. Lin, R., Zhou, Z., Luo, S., Xiao, Y., Wang, X., Wang, S., & Zukerman, M. (2020). Distributed Optimization for Computation Offloading in Edge Computing. IEEE Transactions on Wireless Communications, 19, 8179-8194.
8. Domínguez-Acosta, M., & García-Mireles, G.A. (2021). Identifying Activities for Enhancing Software Quality in DevOps Settings. 2021 10th International Conference On Software Process Improvement (CIMPS), 84-89.
9. Usman, M., Jan, M.A., Jolfaei, A., Xu, M., He, X., & Chen, J. (2020). A Distributed and Anonymous Data Collection Framework Based on Multilevel Edge Computing Architecture. IEEE Transactions on Industrial Informatics, 16, 6114-6123.
10. Xu, S., Zhang, Z., Kadoch, M., & Cheriet, M. (2020). A collaborative cloud-edge computing framework in distributed neural network. EURASIP Journal on Wireless Communications and Networking, 2020.
11. Huo, W., Liu, F., Wang, L., Jin, Y., & Wang, L. (2020). Research on Distributed Power Distribution Fault Detection Based on Edge Computing. IEEE Access, 8, 24643-24652.
12. Jensen, D. (2019). Azure DevOps for IoT Edge Solutions. Beginning Azure IoT Edge Computing.
13. Pal, K., & Karakostas, B. (2021). Software Testing Under Agile, Scrum, and DevOps. Advances in Systems Analysis, Software Engineering, and High Performance Computing.
14. Pastrana Pardo, M.A., Ordoñez Erazo, H.A., & Cobos Lozada, C.A. (2021). Documenting and implementing DevOps good practices with test automation and continuous deployment tools through software refinement. Periodicals of Engineering and Natural Sciences (PEN).
15. Akbar, M.A., Naveed, W., Mahmood, S., Alsanad, A.A., Alsanad, A., Gumaei, A.H., & Mateen, A. (2020). Prioritization Based Taxonomy of DevOps Challenges Using Fuzzy AHP Analysis. IEEE Access, 8, 202487-202507.