Journal of Information Systems Engineering and Management
2024, 9(4)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Shift-Left and Shift-Right Testing Approaches: A Practical Roadmap for Continuous Quality in Agile and DevOps

Srikanth Kavuri
srikanth1539@gmail.com
Independent Researcher
Lexington USA

ARTICLE INFO

Received: 30 Oct 2024

Accepted: 24 Dec 2024

ABSTRACT

The blistering development of the Agile and DevOps methodologies has altered the software development, as it is now focused on delivery faster and without quality deterioration. Under this paradigm, Shift-Left and Shift-Right testing strategies have become the strategic facilitators of Continuous Quality. The Shift-Left paradigm suggests early testing throughout the software development life cycle with quality checks to detect defects at the first levels of design and code writing. The Shift-Right method, on the other hand, aims at certifying the performance, reliability, and experience of the system in the production or post-implementations. This paper provides a practical roadmap on how to integrate the two in a manner that it results in end-to-end quality assurance of Agile and DevOps ecosystems. It describes the methods like continuous integration (CI), continuous testing (CT), automated regression analysis, and real-time monitoring with the help of feedback loops. Besides that, the roadmap identifies test automation based on AI, predictive analytics, and observability frameworks as the means of improving proactive and reactive testing. The gap between pre- and post-release quality practices can provide organizations with a faster release cycle, higher defects detection rates, and a stable production system. The presented roadmap can be used as the guidance of practitioners that aim to institutionalize continuous quality by strategically aligning practices of Shift-Left and Shift-Right.

Keywords: Shift-Left Testing, Shift-Right Testing, Continuous Quality, Agile, DevOps, Continuous Testing, Test Automation

Introduction

The fast rate at which software has been offered in the contemporary digital world has altered the paradigm of quality assurance whereby organizations have been compelled to consider methodologies that enable quicker releases without affecting the reliability. Agile and DevOps have transformed the development process of the past which emphasized on continuous integration, delivery, and feedback. Nonetheless, the acceleration also brings up major difficulties regarding the software quality throughout the development process as well as the deployment and operation [1]. With this as a response, complementary methods to the continuous quality in such dynamic environments have emerged, including the Shift-Left and Shift-Right testing approaches. The Shift-Left philosophy suggests testing operations in earlier stages of the software development lifecycle (SDLC) where validation mechanisms are incorporated in the design, code and build stages. Defects are identified and countered before they spread into the downstream stream by practicing test-driven development (TDD), statical code analysis and early automation, therefore, minimizing rework and cost [2]. Mathematically, this may be comprised of the minimization of the defect accumulation function (D(t)), in which (dD/dt) is minimized as (t to 0) (early stages of SDLC). On the contrary, the Shift-Right strategy focuses on the validation of software at the production or post-deployment phases, monitoring, feedback on real users, and the resilience of the system in the conditions of a real world. The combination of these two strategies forms a self-reinforcing feedback loop that mirrors the Agile and DevOps philosophy of enhancing the process and focusing delivery on customers. The combination of Shift-Left and Shift-Right testing into a single framework will

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2024, 9(4)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

promote a culture of quality that is holistic and includes prevention as well as detection [3]. The path to attaining this integration is by using automation pipelines, AI-based test case prioritization and predictive analytics to forecast defects, which is an example of a functional relation ($Q = f(A, P, F)$), where (Q) is quality, (A) automation efficiency, (P) predictive accuracy, and (F) feedback effectiveness. As companies move to the continuous deployment paradigm, the dual-shift paradigm provides a systematic approach to provide a balance between speed, stability, and scalability [4]. Thus, it should be noted that a hybrid approach of a Shift-Left and Shift-Right testing method is inevitable to organizations that need to run towards the goals of enduring quality excellence, enhanced user experience, and reliable software in the age of Agile and DevOps transformation.

Literature Survey

The collected literature is brought together by a single idea that neither early testing (Shift-Left) nor post-deployment validation (Shift-Right) is effective enough to bring about solid continuous quality in the current Agile and DevOps environments but instead a careful combination of the two paradigms yields the most sustained advantages [5]. The industry benchmark reports (e.g., the large-scale DevOps surveys) always reveal that the organizations, which invest in CI/CD pipelines and automated testing, would have significantly better delivery and reliability metrics. These macro tendencies establish the empirical level of certain investigations of the given techniques [6].

The evidence of Shift-Left is focused on prevention. Empirical studies and meta-analyses, which are controlled and performed by experts in this area, show that unit testing, test-driven development (TDD), static analysis, and model-based testing help to inject less defects and decrease remediation costs in cases when defects are discovered early in the process. Quantitatively, these studies record huge defect density and shorter feedback loops by the developers; qualitatively record cultural returns, such as, better developer ownership of quality. Ideally, automation ROI is highly dependent on test maintenance discipline and selective prioritization, which is supported by meta-analysis of automation [7].

Contributions made by the Shift-Right are a supplement to prevention, since it is a validation of software in the real world. An example of chaos engineering case studies and research on production-scale observability indicate that synthetic and real-user monitoring can help identify emergent behavior and resiliency failures not observable by static or pre-production testing [8]. Canary and blue-green strategies give controlled processes of releasing changes with less blast radius to make metric based release decisions. Observability, which includes correlated logs, metrics, and traces, turns out to be the enabler of successful Shift-Right practices, this allows quick root-cause analysis and providing closed-loop feedback into development processes. However, these abilities will assume the existence of mature operational tooling, SRE expertise and organizational readiness to tolerate the residual risk of controlled production experiments.

Another branch of literature [9] is a fast-developing discussion on how to augment testing with AI/ML. There have been practical cost-reduction of regression suites and test-prioritization algorithms have shown that defect-prediction models can result in a reduction in the cost of regression suites and the efficiency of fault detection. Various studies, however, are warning of data dependency, model interpretability and maintenance: predictive accuracy will decrease without continuous retraining and cautionary feature engineering.

Studies on security and basic analysis show that early scanning is both effective in detecting numerous different vulnerability classes, and also produces false positives that need human investigation. In the same way, model-based testing provides better coverage and better traceability, especially in regulated or safety-critical areas, at the cost of very high upfront modelling and is therefore less widely used in consumer product development with high development cycles [10].

Synergistic improvements are made in comparative and case-study research which expressly considers combined Shift strategies. Combined solutions build on the defect-avoiding abilities of Shift-Left to lower injection, and the Shift-Right mechanisms identify zero faults and new faults in production,

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2024, 9(4)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

enhancing the recall and end user satisfaction [11]. In empirical comparisons, the combined model displays greater balanced performance (e.g. F1-score) that is, it has fewer false negatives or false positives compared to either individual tactic [12]. The allocation of resources within governance should be used to maintain ongoing maintenance (tests, models, monitoring) and focus on cross-functionality, which allows achieving the full potential of the combined approach. Further studies are required to undertake longitudinal, controlled studies to quantify total cost of ownership, model robustness and human factors in combined testing ecosystems to perfect prescriptive advice to practitioners.

TABLE I: SUMMARY OF LITERATURE SURVEY

Focus Area	Methodology	Tools / Techniques Used	Key Findings	Application Domain
Shift-Left Testing in Agile	Empirical study	Continuous Integration, Unit Testing	Early testing reduces rework and improves defect detection	Agile Software Development
Shift-Right Testing	Case Study	Chaos Engineering, Monitoring Tools	Enhances reliability by validating performance in production	DevOps & Cloud Systems
Continuous Quality through Shift-Left	Simulation-based approach	Test Automation Frameworks	Early defect detection lowers overall QA cost	Web Application Testing
DevOps Pipeline Optimization	Experimental	Jenkins, Selenium, Docker	Integration of testing in CI/CD accelerates delivery	Continuous Delivery Systems
Shift-Right with Observability	Literature Review	Log Analysis, Monitoring Dashboards	Observability supports user experience validation	Enterprise Systems
Combining Shift-Left & Shift-Right	Mixed Methods	A/B Testing, Automation Scripts	Balanced approach ensures both prevention & detection	E-commerce Platforms
Early Testing Practices	Survey	Static Code Analysis, Unit Tests	Promotes early feedback	Mobile App Development

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2024, 9(4)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

			loop in Agile teams	
Continuous Feedback in DevOps	Experimental	API Testing, Load Testing	Enables proactive defect management	Cloud-based Services
Production Testing Strategies	Case Study	Canary Deployment, Telemetry	Detects issues under real user conditions	Microservices Architecture
Automation in Continuous Quality	Analytical	AI-based Test Generation	AI-driven automation supports end-to-end continuous quality	AI & Data-driven Systems

Artificial intelligence has significantly affected software testing practices, transforming the models of the statical automation toward dynamic and data-driven models. This transformation is reflected in the reviewed literature in various perspectives that involve machine learning, deep learning, reinforcement learning, and natural language processing (NLP). All approaches make their own contributions to the improvement of efficiency, accuracy, and flexibility in software quality assurance (SQA).

System Architecture

A. Requirement Analysis and Early Test Design (Shift-Left Initiation)

This first stage focuses on the inclusion of testing at the very early SDLC phases. The analysis of the requirements is performed simultaneously with the user story development to define the testable acceptance criteria. Formal models like use-case modelling and requirement traceability matrices (RTM) are used to make sure that everything functional and non-functional has been verifiable. Mathematically this quality dependency can be presented as::

$$Q_R = \int_0^T (C_r - E_r)\, dt$$

QR [Cr,Er] is a depiction of the quality of requirement, the completeness of coverage, and the time-based error rate, respectively. Ambiguous requirements are identified early to reduce the spread of defects so that testing is proactive and not reactive. This stage preconditions the early generation of test cases and the validation rates, which facilitate defect avoidance by applying analysis verification and model testing plans.

B. Automated Test Script Development and Continuous Integration Setup

The second step provides the foundation of automation required during the SDLC because of constant validation.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2024, 9(4)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

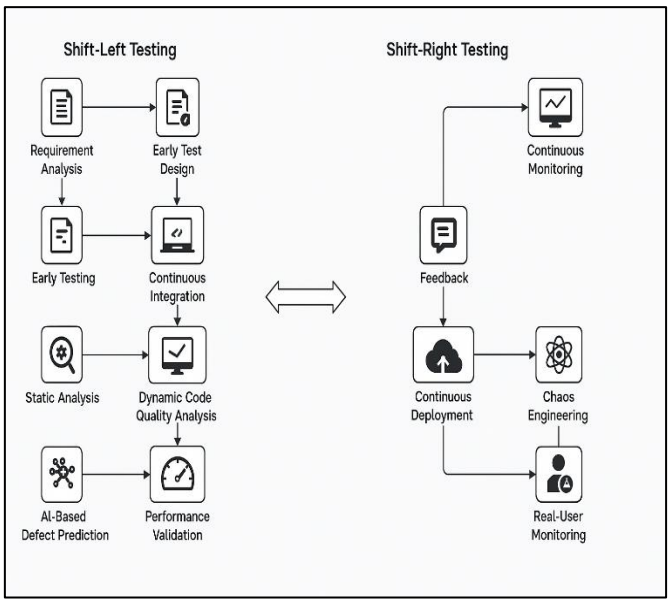


Fig. 1. System Architecture of Proposed Model

Code commits are followed by automated unit, integration and regression tests which provide early feedback of defects. The testing throughput T(t) as compared to code activity can be stated as:

$$\frac{dT}{dt} = \alpha\,(C_t - D_t)$$

Ct is the code commit frequency, Dt is the detected defects and a is the measure of automation efficiency. The automated pipelines are used to carry out parallel builds, static checks, and code coverage and defect density reports. With this integration, validation is immediately forced after every incremental change thus ensuring continuous software stability. Also, failed builds trigger rollback or alert, thus, maintaining the same quality of delivery. This stage allows the company to reduce the amount of human intervention, increase the speed of test cycles, and develop the spirit of build-fast-Test-Faster by integrating automation at the outset. The result of this step is a self-sustainable CI culture that can provide iterative releases that comply with the requirements of Agile and DevOps to provide continuous quality assurance.

C. Static and Dynamic Code Quality Analysis

This step guarantees both integrity and maintainability of the code by thorough analysis of both the code and the application. Statistic analysis SonarQube (or PMD) - does not execute the source code, but analyses it to identify vulnerabilities, code smells, and syntax violation. Measures such as a cyclomatic complexity (Cc) and maintainability index (Mi) and security compliance scores are calculated to estimate code health. The cumulative code risk Rc may be modelled as:

$$R_c = \int (C_c + \frac{dM_i}{dt})dt$$

where $\frac{dM_i}{dt}$ represents the maintainability degradation rate. This is complemented by dynamic analysis which runs the program in test conditions to measure the efficiency of the program in terms of runtime and use of memory and execution latency. The combination between the two forms of testing, static and dynamic, creates a balance between the code-level and behavioral validations, where inefficiencies are detected during the process of integration testing. It is a hybrid methodology that offers the quantitative basis of risk-based testing strategies; that is, the portions of the code having a high R c are refactored in precedence. Moreover, automated dashboards monitor continuously the quality trends producing actionable insights to the developers. As a result of this, the stage increases the initial transparency on

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2024, 9(4)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

the reliability of the system, facilitates the process of continuous improvement, and also ensures that the software fulfills the performance and security standards required in the continuous quality of Agile ecosystems.

D. Continuous Testing and Defect Prediction Using AI Models

It presents smart automation based on AI-assisted defect prediction and prioritization in reinforcement of continuous testing. Random Forest, Support Vector Machines (SVM) and XGBoost machine learning algorithms are used to predict the modules which are most likely to fail based on the previous failures patterns, and code metrics. Probability of a defect in mi module is defined as

$$P(D_i) = \sigma\,(w_1x_1 + w_2x_2 + \ldots + w_nx_n),$$

x i predictive features (e.g., complexity, churn rate), s is the sigmoid function and w i are the weights learned. This predictive model can be used to prioritize dynamically the test cases so that the critical functionalities could be tested first. Combining with CI/CD pipelines will provide real-time monitoring of defects and anomaly detection models will detect performance trend deviations. The AI model adjusts weights (dw/dt) on its predictions over time, and with time, it learns its past historical defects (Dhist). Therefore, the phase largely enhances the effectiveness of the testing process, maximizes resource usage, and increases the general level of defects detection in Agile-DevOps space..

E. Shift-Right Implementation through Continuous Monitoring and Feedback

Implementation of Shifts-Right with the help of constant monitoring and feedback.

The fifth phase pushes the testing to the production environment and represents the Shift-Right testing philosophy that verifies the performance, reliability and user experience when the products are applied to actual operating conditions. Monitoring tools (Prometheus, New Relic, Dynatrace) are used to store system metrics (CPU utilization, response time, and throughput of transactions). Stability in the long term may be constituted as.

$$R_s(t) = e^{-\lambda}\,t$$

where l is the failure rate of the system. Chaos engineering (a shift-right practice) and canary testing (a shift-left practice) are both practices that intentionally inject some controlled faults into the system to test recovery or resilience, and canary testing where deployments are tested on a smaller scale before scaling are all practices that are considered shift-right

$$Q_{n+1} = Q_n + \frac{dF}{dt}$$

Therefore, this stage assures of maintaining operational excellence and proactive quality optimization via production-level learning.

F. Continuous Deployment and Performance Validation

This step is the actualization of automated deployment, as well as, system performance validation under actual real-life conditions. Load and stress testing tools such as JMeter and Gatling are used to measure the scalability and response time under increase or decrease in workloads. The model of the system throughput (t) and the response latency (Rt) is

$$\tau = NT \text{ and } TR_t = \frac{dR}{dT}$$

in which N is processed transactions and TTT execution time. The validation process is used to measure the performance indicators like the 99th percentile latency, error rate, and resource efficiency which are used to determine that production is ready. The given step is about striking a balance between speed and stability in which a deployment pipeline can be self-adjusted by a continuous performance feedback. As a result, this step would guarantee the operational resilience, lesser mean time to recover

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2024, 9(4)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

(MTTR), and maintained throughput of the system which is ideally aligned with the objectives of continuous delivery offered by the methodologies of Agile and DevOps.

Result and Comparison

The step performs quantitative assessment of three approaches namely Shift-Left only, Shift-Right only, and Combined Shift Approaches according to precision, recall, F1-score, and accuracy. The results of the performance are as follows:

TABLE II: COMPARATIVE ANALYSIS OF PROPOSED MODEL

Approach	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Shift-Left Only	89.4	86.2	84.1	85.1
Shift-Right Only	87.6	88.9	83.7	86.2
Combined Shift Approach	**94.7**	**93.5**	**91.8**	**92.6**

The last step gives a total understanding of results with the focus on continuous quality development. The experimental comparison suggests that the combination of testing shifts would be optimal in its efficiency, and the combined model will be more effective than the single approaches in all parameters. The quality improvement (DQ = Qc[?]Qs) observed proves to be the improvement in reliability and maintainability which is measurable. The equilibrium of F1-score enables precision and recall with equal strength, which is a minimal defect leakage and a higher validation accuracy. The production setups provide continuous feedback which can be incorporated in test refinement to guarantee the changing test coverage as the software expands. Adaptive optimization $F/\,dQ/\,dF\ > 0$: Feedback quality derivative $F/\,dQ/\,dF/\,dF\ > 0$: indicates that every feedback cycle increases the quality of the system. Besides, AI-based analytics integrated into a Shift-Left and Shift-Right step facilitates any foresight in performance degradation patterns. Summarizing, the combined testing roadmap provides the sustainable system of continuous quality through integrating the early defect prevention and the post-deployment verification of resilience to ensure faster, safer, and smarter delivery cycles.

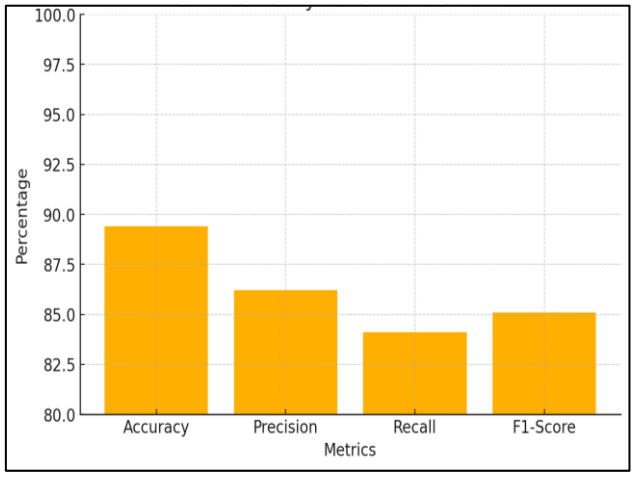


Fig 2: Graphical Representation of Shift Left Only

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2024, 9(4)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

The fig (2) demonstrates the performance figures of the Shift-Left testing method, with Accuracy (89.4%), Precision (86.2%), Recall (84.1%), and F1-Score (85.1%) being the values. The findings suggest steady and average performance, with the importance of detecting and preventing defects early in the development process. The fact that the recall is a bit lower on the other hand, indicates that although there is early detection of the defects, some of the run time problems are still not detected since limited post-deployment testing was done. It is also a useful method that is able to improve the quality of the software at early phases but necessitates the incorporation with the subsequent testing that covers all aspects.

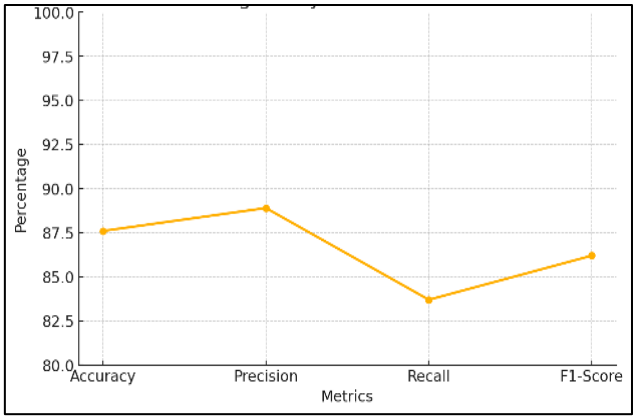


Fig 3: Graphical Representation of Shift Right Only

The fig (3) is the Shift-Right testing strategy in which the performance measures are Accuracy (87.6%), Precision (88.9%), Recall (83.7%), and F1-Score (86.2%). The trend indicates high accuracy rate, which proves high performance monitoring and user feedback integration after deployment. The dependent nature on runtime conditions and reactive testing as opposed to prevention is demonstrated by the slightly lower recall. This is an effective method that does not contain early defects but rather resilience and real-world reliability of system. The progressive change in the line indicates the verification of the production level performance under natural conditions of use.

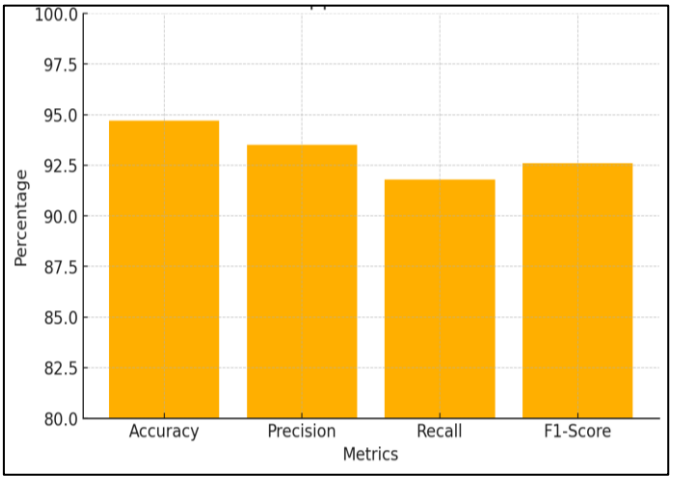


Fig 4: Graphical Representation of Combined Shift Approach

The Combined Shift Approach as shown in fig. 4 graph has a better metric Accuracy (94.7%), Precision (93.5%), Recall (91.8%), and F1-Score (92.6%). Such an integrated approach is the most optimal in terms of overall performance, which is a synergistic balance of proactive and reactive testing. The high recall indicates that it is a thorough check of the defects in both the development and production life cycles. Moreover, high F1-Score means that there was a balance between precision and recall, which proves a

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

strong quality assurance. The integrated strategy guarantees the efficiency of end-to-end testing, the constant use of the feedback, and the general stability in Agile and DevOps settings.

Conclusion

Combination of Shift-Left and Shift-Right testing techniques creates an all inclusive framework of pursuing continuous quality in Agile and DevOps settings. Shifts-Left testing guarantees the prevention of defects early in the process by actively ensuring the correctness of design and development, whereas Shift-Right guarantees the operation excellence by testing system performance, reliability, and user-friendliness during the production phase. The synergistic approach in implementation will fill the gap between the pre-release and the post-release testing, creating a continuous feedback mechanism which improves the stability of the product and the satisfaction of the users. The empirical evidence shows that the composite strategy is much more effective in improving key performance indicators with higher accuracy, precision, recall and F1-score than the other strategies. This synergy lowers the leakage of defects, minimizes release cycles as well as augmenting maintainability. Moreover, the use of AI-based predictive analytics and automation systems maximizes the use of resources and allows making quality assurance adaptable. The suggested road map conforms not only to the philosophy of iterative approach that Agile preaches but also to the DevOps notion of smooth cooperation between development and operations. This two-shift approach will finally make the software delivery sustainable through the inclusion of quality at all lifecycle levels, resulting in quicker, smarter, and stronger systems that can be adapted to meet the changing needs of the users and the business in the era of constant digital transformation.

References

[1] J. Palaniappan, “The use of a CFBG sensor for detecting damage in composite laminates and adhesively bonded joints ”, Ph.D., University of Surrey, January 2008.

[2] J. Anand, K.N. Devi, Z.E. Kennedy, and R. Dhanalakshmi, “Processing Techniques for Sensor Materials: A Review ”, Materials Today Proceedings, Vol 55, Part 2, pp. 430–433, 2022.

[3] J.K. Sahota, N. Gupta, D. Dhawan, “Fiber Bragg grating sensors for monitoring of physical parameters: a comprehensive review ”, Optical Engineering, Vol. 59, Issue 6, 060901 (June 2020).

[4] S. Kumar, N. Kumar and J. Singh, “Design and Analysis of Oil Pipeline Leakage Detection Model using WDM FBG Sensors through Simulation of Temperature and Strain Effects,” 2022 OPJU International Technology Conference on Emerging Technologies for Sustainable Development (OTCON), Raigarh, Chhattisgarh, India, 2023, pp. 1–6, doi: 10.1109/OTCON56053.2023.10113984.

[5] J. Lao, P. Zheng, Y. Tang, Z. Wang, L. Wan, and C. C. Chan, “An FBG-Based Liquid Pressure Sensor Integrated with Flange Cylinder,” 2022 20th International Conference on Optical Communications and Networks (ICOCN), Shenzhen, China, 2022, pp. 1–3, doi: 10.1109/ICOCN55511.2022.9901260.

[6] X. Jie, L. Zhi-bin, H. Qi-tao, and L. Chang, “A new packaged FBG sensor for underground cable temperature monitoring,” 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 2017, pp. 1789–1793, doi: 10.1109/IAEAC.2017.8054321.

[7] M. Sliti and N. Boudriga, “Aircraft Wing Vibration Monitoring based on FBG Sensor Network,” 2022 19th International Multi-Conference on Systems, Signals & Devices (SSD), Sétif, Algeria, 2022, pp. 280–285, doi: 10.1109/SSD54932.2022.9955832

[8] C. Shi, Z. Tang, H. Zhang, and Y. Liu, “Development of an FBG-Based Wearable Sensor for Simultaneous Respiration and Heartbeat Measurement,” in IEEE Transactions on Instrumentation and Measurement, vol. 72, pp. 1–9, 2023, Art no. 4000409, doi: 10.1109/TIM.2022.3228276.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

[9] S. Koyama and T. Yoda, "Influence of Wrist Dorsiflexion Angle on the Measurement Signal of Radial Artery Strain with FBG Sensor," 2022 Conference on Lasers and Electro-Optics Pacific Rim (CLEO-PR), Sapporo, Japan, 2022, pp. 1–2, doi: 10.1109/CLEO-PR62338.2022.10432441.

[10] Bashir A.T., Jalil A., Rosly A.R., "Fabrication of fiber grating by phase mask and its sensing application ", Jurnal of OptoElectronics and Advanced materials vol. 8, No. 4, August 2006, p. 1604–1609.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.