Journal of Information Systems Engineering and Management
2025, 10(37s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Harmonizing Product Strategy with Development Execution: A Comprehensive Study on Quality Engineering Excellence

Samuel Maniraj, Selvaraj1 * Dr.G. Arun Sampaul Thomas 2
[1]Engineering Manager, Test Automation, Veeva Systems, USA
2 Department of AI&ML, J.B. Institute of Engineering and Technology, Hyderabad, Telangana, India
saminkit@gmail.com*, arunsam.infotech@gmail.com

ARTICLE INFO

Received: 16 Dec 2024

Revised: 22 Feb 2025

Accepted: 28 Feb 2025

ABSTRACT

Quality Engineering (QE) plays a critical role in bridging the gap between product development processes and business objectives. This paper presents a structured, methodology-agnostic framework for effective Quality Engineering execution, applicable across Agile, Waterfall, and hybrid development models. It delineates key principles and best practices across five essential phases—understanding, planning, execution, reporting, and continuous improvement. By systematically addressing requirement gaps, establishing clear acceptance criteria, implementing robust testing strategies, and fostering a culture of continuous enhancement, QE professionals can significantly improve software quality, operational efficiency, and business value. The findings underscore the strategic importance of Quality Engineering in driving organizational success through proactive quality assurance and process optimization.

Keywords: Automation, Quality Engineering, Software Engineering, Software Quality.

INTRODUCTION

In the rapidly evolving digital landscape, organizations must ensure that their product development efforts are strategically aligned with business objectives to drive sustainable growth and competitive advantage. Quality Engineering (QE) plays a pivotal role in bridging the gap between product strategy and development execution by embedding quality-driven processes throughout the software development lifecycle. Unlike traditional quality assurance, which primarily focuses on defect detection, QE takes a proactive approach by ensuring that software products are designed, developed, and delivered in alignment with business goals, customer expectations, and industry standards.

Effective alignment between product strategy and development execution requires a structured approach to quality engineering that is adaptable across various software development methodologies, including Agile, Waterfall, and hybrid models. This paper presents a methodology-agnostic framework for Quality Engineering execution, emphasizing five critical phases: understanding, planning, execution, reporting, and continuous improvement. By systematically addressing requirement gaps, defining clear acceptance criteria, strengthening testing processes, and fostering a culture of continuous enhancement, QE professionals can ensure that product development efforts remain aligned with strategic objectives while delivering high-quality software solutions.

This study underscores the strategic role of Quality Engineering in modern software development, highlighting its impact on product success and organizational growth and highlights the essential role of Quality Engineering in optimizing product development efficiency, enhancing cross-functional collaboration, and ensuring that organizations achieve their desired business outcomes. Through a comprehensive exploration of best practices, this paper aims to provide valuable insights for QE professionals, software engineers, and business leaders striving to enhance quality standards and optimize development efficiency.

2.OBJECTIVE:

This article presents a flexible Quality Engineering (QE) framework to align product development with business goals, applicable across Agile, Waterfall, and hybrid methodologies. Structured into five phases—understanding, planning, execution, reporting, and continual improvement—it emphasizes requirement clarity, rigorous testing, and iterativerefinement. By integrating proactive quality assurance and process optimization, QE enhances software quality, operational efficiency, and commercial outcomes, positioning it as a strategic driver of organizational success.
In Summary, major contributions of this paper are described in the following,

- Ability to find the gaps in different layers of platform or application
- Balancing Structured and Exploratory Test Approach

- Achieving comprehensive Test coverage
- Importance of monitoring and producing metrics
- Identifying rooms for improvement

Our research resulted in below main findings which are summarized in Table 1

Numbers	Key Findings	Implications
F1	Breaking down the development tasks into Infrastructure, Database, API and UI changes	Detects gaps early and helps align development approach with product expectations while defining the requirements itself.
F2	Close collaboration between QE, developers, and product owners	Improves clarity and reduces rework.
F3	Ensuring APIs are documented	Ensures early Automation adoption
F4	Understanding on all layer's UI, API and Database	Enhances Test coverage
F5	Reviewing the QE strategy with stakeholders	Ensures alignment with business objectives and user expectations
F6	Ensuring test environment readiness	Prevents last-minute delays.
F7	A balanced approach to structured and exploratory testing.	Enhances test coverage and efficiency.
F8	Effective bug triage	Accelerates defect resolution, improving software quality and test reliability
F9	Real-time monitoring	Organizations should invest in advanced test management tools for real-time monitoring.
F10	Effective metric tracking	Standardized reporting formats improve transparency and accountability in testing outcomes.
F11	Automation coverage insights	Optimize test execution efficiency and cost.
F12	Systematic defect categorization	Enhances Quality Engineering process optimization
F13	Automation enhancements	Reduce redundancy and increase testing accuracy.

Table 1. Key findings and implications of this research.

3.PROPOSED MATHOD

The findings of this paper contribute general information for future research, as the quality of software platforms and applications will become increasingly important. They can also help product owners and developers to apply suitable development approach and make sure it aligns with product goals. Traditional Quality Assurance approach have certain effect on quality of software application but evolution of quality engineering model which this study emphasizes to align product expectation and development strategy increases quality standards.
The rest of the paper is structured as follows. The next section 2 describes The Foundation of Quality Engineering. Section 3 Structuring the Quality Engineering approach for Success. Section 4 Execution Excellence: Achieving Comprehensive Testing. Section 5 Realtime monitoring and producing meaningful metrics, Section 6 Continuous Improvement: Learning from the Process.

2. The Foundation of Quality Engineering

Quality engineering in software engineering is a critical discipline aimed at ensuring that products meet the specified requirements and deliver a positive user experience. However, achieving this goal requires a clear understanding of the end user's needs and expectations, along with an efficient process for tracking the execution of these needs in the development phase. The Quality engineer is central to this process, as they bridge the gap between product requirements and technical development.
This study focuses on two primary aspects of Quality engineering:

1. **Identifying and clarifying end user requirements**: A thorough understanding of what the end user expects is the foundation for the development process.
2. **Finding gaps to align product expectations with development executions**: Ensuring that the requirements are effectively implemented throughout the development cycle and that no discrepancies arise.

2.1 Identifying and Clarifying End User Requirements: The first step in the QE process is to clearly identify and clarify the product's requirements. This step is crucial in ensuring that the development process addresses all necessary components and aligns with business goals.

- **Active Engagement in Reviewing Requirements**: The Quality engineer should not rely on assumptions but actively engage in reviewing and analyzing high-level business requirements. The goal is to ensure

Journal of Information Systems Engineering and Management
2025, 10(37s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/
Research Article

a clear understanding of what needs to be tested and how it fits within the overall product vision.

- **Breaking Down High-Level Requirements**: High-level business requirements are broken down into more manageable elements such as Epics and User Stories. This allows for prioritization based on business goals, ensuring that essential features are addressed first.
- **Collaboration with Stakeholders**: Collaboration among key stakeholders—such as developers, testers, and business users—is critical. The Quality engineer must engage in discussions to define acceptance criteria for each user story. Clear communication ensures that everyone has a shared understanding of the requirements and expectations.
- **Categorization of Development Tasks**: The Quality engineer drives the conversation with product owners and development engineers to categorize development tasks into four main areas:
  - Infrastructure Changes,
  - Database Changes,
  - API Changes, and
  - UI Changes.

2.2. Finding Gap's to align product expectations and development executions:

In this study we identified and focused on below potential gap where Quality engineer should drive the conversation in aligning the end user requirements with the development approach, Quality engineer should establish a process to track and manage possible misalignment or any unresolved questions. Effective and close communication between product and dev is essential to succeed in this part. It is essential to close these gaps to avoid potential defects that may arise from incomplete or misunderstood requirements.

Potential Gaps:

- **Integrations**: The Quality engineer must analyze both internal and external integration points during the tech design review.
- **API and UI Consistency**: Ensuring consistency between API response messages and the UI is critical for a seamless user experience.
- **Unexpected Scenarios:** Identifying edge cases and unexpected scenarios is essential for comprehensive testing.
- **API Availability for Automation**: Ensuring that APIs are available early in the process to facilitate automation and reduce manual testing efforts.
- **User Experience and Interface Evaluation**: Working with UX experts to evaluate the application's user interface against mock flows, accessibility, and security standards.
- **Cross-Platform Compatibility**: Testing from the end user's perspective across different platforms and browsers.
- **API Documentation**: Verifying that new APIs are well-documented, for example, using Swagger for enhancing quality engineering productivity in automation.
- **Performance Impact**: Assessing the potential performance impact of new features or changes to the product.

The research reveals that Quality Engineering fosters engagement in understanding and clarifying requirements early in the process, can identify gaps in the product understanding phase and also help ensure that the final product aligns with user expectations.

3. Structuring the Quality Engineering Process for Success

The complexity of modern software applications demands a systematic quality engineering strategy. A robust quality engineering process minimizes defects, enhances software reliability, and ensures seamless integration across system components. This study explores the essential components of an effective quality engineering strategy and demonstrates its application through use case references.

3.1. Strategizing the Quality Engineering process for success

Test Strategy document provides best practices on, how to strategize test approach for Platform or an application. It describes the Goal, Test scope, Platform Assessment, testing classes, Test Environments, Test data approach, Success criteria and Defect Management process involved in intended Test process activities. The main purpose of a Test strategy is to outline the Test approach to achieve Organization's core objectives from a quality assurance perspective. Test Strategy should be frequently reviewed, challenged and updated as the organization and the product evolve over time. A detailed test plan provides an approach for the entire QE process, ensuring that all areas of the applications are covered. In this study we identified the areas to be considered for strategy and what testing class covers this area. Our research resulted in below areas which are summarized in Table 2.

Journal of Information Systems Engineering and Management
2025, 10(37s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/
Research Article

Table 2. Key Areas to focus and Testing classes applied

Area	Focus Area	Testing Class
A1	Understanding on all layer's UI, API and Database.	Integration
A2	UI Design and experience as an end user	Functional, Exploratory
A3	Data flow between Modules	Integration
A4	Internal and External Integration points.	Integration
A5	Error handling – Consistency between UI and API's	Security, Functional
A6	CRUD operations: Database validations through SQL's and API's	Database
A7	Logging	Functional, Security
A8	Automation for the benefit of Regression • API first approach, Limit UI Automation • Hybrid approach for UI Automation.	Automation
A9	Test Environment readiness	Smoke
A10	Test data needs	Functional, Automation
A11	Integration points for Connectivity	Production Smoke, Automation
A12	Performance impact	Performance

4.METHODLOGY

4.1Definition of Testing Classes:

Smoke Testing:

Smoke Testing will be performed by quality engineering team in a testing environment through Automated scripts to ensure that the environment is ready and stable for Functional Testing. Smoke testing will be scheduled to run through CI/CD pipeline daily. Smoke test scenarios will be added to the existing smoke test suite on a release basis. Smoke test helps in exposing integration and major problems early in the cycle.

4.1.1 Functional Testing:

The core functionality of **features** which is being built for **the application** will be validated based on the test cases written for each user story created out of requirements described. The main objective of the functional testing is to validate whether business requirements are aligned with software delivered and working as expected. Quality engineering team will execute the test cases and validate whether the user story is meeting the defined acceptance criteria or not. The test cases should prescribe any pre-requisites and a set of steps or actions to execute the test.

4.1.2 Integration Test:

Integration testing is done to test the integration between <System1> and the <System2> or more than one system. The purpose of this level of testing is to expose defects in the interfaces and interaction between integrated components. Unit level Integration Testing is usually done by developers and System level Integration Testing by testing team or independent testers.

4.1.3 Exploratory Testing:

Exploratory Testing is defined as a type of testing where Test cases are not created but quality engineering team will check the system on the fly. Exploratory testing will be performed on **application** without any formal Test cases to identify any potential issues in the delivered feature for each build. Exploratory testing will be done in unstructured and in freestyle way with random inputs or any other criteria aimed at identifying the unexpected behavior. Exploratory testing should also cover UI/UX experience of the app.

4.1.4 Unit Testing:

Unit Testing is done by development team as part of standard development practice where individual units/ components of an <application> is tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development of the feature by the developers.

4.1.5 Functional Regression (Automation):

Business critical test cases will be identified from each user story to include as part of automated regression suite. Automation team runs the regression testing cycle for the benefit of confirming the existing use cases are not impacted with the new changes introduced in the system.

4.1.6 API Testing:

User stories that involve new API's will be validated through API tool as part of API Testing. Identify API level test scenarios with respect to functionality and error handling. The purpose of API Testing is to check the functionality and reliability of the programming interfaces.

4.1.7 Backend Testing:

Requirements that require a database verification will be validated against databases with database test cases. CRUD operations will be performed through API and verified against database results.

4.1.8 Security Testing:
Security testing is performed to ensure all access and authorization rules are enforced correctly in the application.

4.1.9 Performance Testing:

Performance Testing may be needed for an app which is being built, and quality engineering team will identify scenarios for performance testing. Typical use case for performance testing is subjected to no of users and the volume of data being retrieved from the system or uploaded into the system.

4.1.10 Prod Smoke:

Production validation test is a smoke test performed in a production environment without affecting the data. Production smoke test suite should be reviewed and updated for each release. Prod smoke is critical for ensuring the systems are operational after every production releases.

4.3 Reviewing the Strategy

Reviewing the quality engineering strategy with stakeholders ensures alignment with business objectives and user expectations. Essential step in this framework is to review the strategy with stakeholders, including Development, business owners and product managers. The QE professional will ensure that the test cases align with the acceptance criteria and conduct a walkthrough of the plan to validate it with all stakeholders.
This study identified the below key benefits of conducting Test strategy reviews which are summarized in Table 3.

Table 3 Key benefits of conducting Test strategy reviews

Benefit Number	Key Benefit	Implications
B1	Improved Quality	Reduces the likelihood of defects due to poor test coverage.
B2	Cost-Effectiveness	Detecting issues in the review phase is cheaper than fixing bugs in production.
B3	Knowledge Sharing	Encourages collaboration and understanding across teams.
B4	Standardization	Ensures consistency in testing practices.

Our study identifies that, A structured Quality Engineering approach with a well-defined strategy significantly improves software quality. By incorporating various testing classes and stakeholder reviews, organizations can ensure robust, reliable, and user-friendly software solutions.

4.2 Execution Excellence: Achieving Comprehensive Testing
This study explores the critical aspects of achieving execution excellence in testing, focusing on test environment

readiness, balancing structured and exploratory testing, and systematic bug handling. Findings suggest that integrating these elements enhances software quality and Agile development efficiency. Implications for practice include improved defect detection, accelerated feedback loops, and enhanced test coverage.

4.3 Our study finds “Testing with the speed for Development” is crucial to achieve execution excellence.

It is very crucial to test the application being developed with the speed of development. Quality engineering approach takes a step to run through the test cases on the code being developed to prevent the defects rather finding defects in the test environment. Our study demonstrates that when you collaborate and test the application at developer’s environment, it enhances the quality of the software, and it helps align the product being developed with the business expectation. This is an effective method for being Agile since even before the test environment is ready, we can achieve comprehensive test exection with developer collaboration. Our study summarizes below workflow to execute at different environments to achieve execution excellence.
The figure 1 illuatrated that Annotations used in lifecycle used Quality Engineering execution lifecycle.

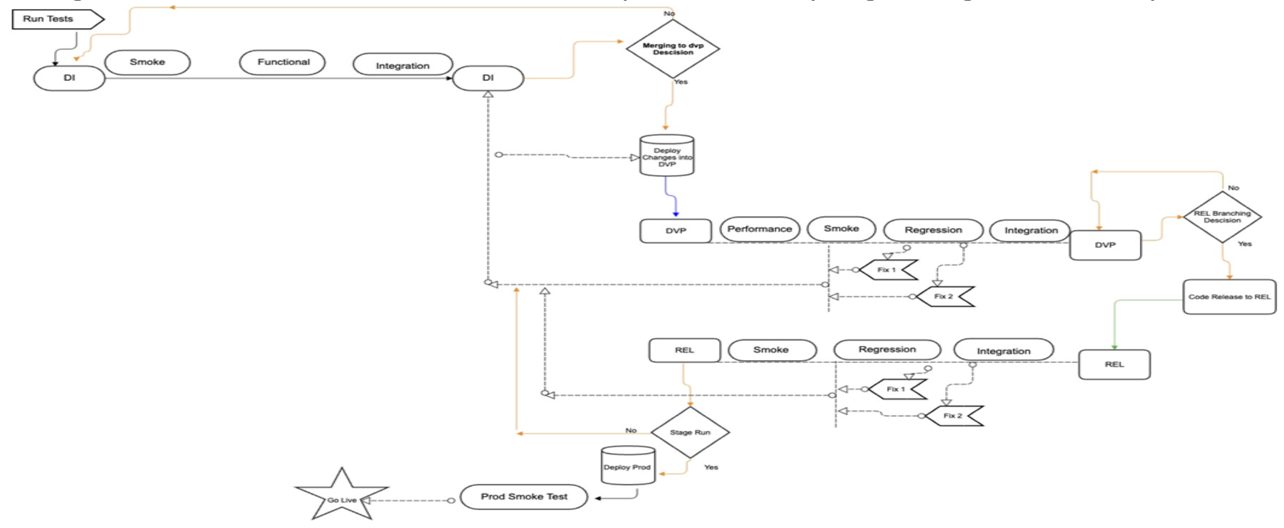

Figure 1. Quality Engineering execution lifecycle

Table 1: Annotations used in lifecycle

Annotation	Type of Environment	Built for
DI	Dev Integration	Developer testing
DVP	Develop	QE Smoke, Functional, Integration, Regression, Performance
REL	Release	QE Smoke, Functional, Integration, Regression, Performance
Stage	Staging (Pre-Production)	Stage Smoke
Prod	Production	Prod Smoke

4.1. Test Environment Readiness:

Test execution begins with the preparation of the test environment during the development phase once the features are being delivered. Quality engineering professionals must coordinate with infrastructure teams to ensure that the environment is ready for testing. Connectivity tests should be conducted to ensure all systems are integrated and functioning as expected. Once the environment is validated, a smoke or sanity test suite is executed to ensure that the deployed code and environment are suitable for further functional testing.
The foundation of test execution is a well-prepared environment. This involves:

- **Collaboration with infrastructure teams** to ensure readiness.
- **Connectivity tests** to verify system integration.
- **Smoke testing** to validate the stability of the environment for further testing.

Findings indicate that early environment validation minimizes deployment issues and reduces testing bottlenecks, ensuring a smoother transition from development to testing.

4.2. Balancing Structured and Exploratory Testing:

Quality engineering professionals should execute planned test cases, which may include functional, integration, regression, and ETL tests. Exploratory testing is also an important practice, as it enables testers to uncover potential issues that were not foreseen in the original test cases. Quality engineering teams should ensure that issues are raised promptly, with clear documentation of any discrepancies between user stories and the user interface (UI).

4.2.1. Exploratory testing complements Agile in several ways:

- **Flexibility**: Allows testers to quickly adapt to changes in requirements and new functionalities.
- **Faster Feedback**: Identifies critical defects early in the development cycle without waiting for scripted tests.
- **Enhanced Test Coverage**: Encourages testers to think beyond predefined scenarios, uncovering hidden issues.
- **Empowered Testers**: Provides autonomy and creativity to testers, allowing them to leverage their experience and domain knowledge.

4.2.3 Benefits of Exploratory Testing,

- Encourages creativity
- Intuition in finding defects
- Adapts quickly to new changes
- Explore the unexpected
- Provides rapid feedback in fast paced agile environments.

4.3. Systematic Triage:

In quality engineering, defects are inevitable. However, the efficiency of addressing these defects determines the overall quality of a product. Systematic triage involves analyzing, categorizing, and prioritizing defects to streamline resolution. This study investigates how defect triage meetings, led by quality engineering professionals, impact defect resolution efficiency.
As defects are discovered, it is critical to manage them efficiently. Quality engineering professionals should lead defect triage meetings, where defects are analyzed, categorized, and corrective actions are identified. Effective Coordination with development teams and product team is also crucial during this phase to ensure that any issues found during testing are addressed promptly.
This study shows reveals steps to run an effective triage process.

4.3.1 Standardization of Triage Meetings: Organizations should implement regular defect triage meetings to accelerate issue resolution.

Cross-Team Collaboration: Establishing a seamless communication framework between quality, development, and product teams ensures timely corrective actions.
Data-Driven Decision Making: Leveraging defect categorization metrics can enhance predictive maintenance and proactive quality assurance.

5. Realtime monitoring and producing meaningful metrics

Effective real-time monitoring and metric tracking in software testing play a pivotal role in ensuring product quality and process efficiency. This study explores the significance of tracking meaningful metrics throughout the quality engineering lifecycle, providing insights into optimizing testing progress communication and reporting outcomes. The research highlights key tracking methods, reporting mechanisms, and their implications for agile development.

5.RESULTS

5.1. Metrics:

Effective tracking and communication of testing progress are crucial throughout the quality engineering lifecycle. Quality engineering professionals leverage various tools and metrics to monitor performance and execution.
To ensure comprehensive monitoring, the following metric points should be tracked:

- **Test Execution Progress**: Tracking the number of executed, pending, and blocked test cases.
- **Defect Dashboard**: A real-time visual representation of defect status, severity, and resolution progress.
- **Sprint Burn Down Charts**: Graphical representation of remaining tasks within a sprint, assisting in assessing sprint completion status.
- **Test Efficiency**: Measurement of the effectiveness of test cases based on defect detection rates.
- **Test Coverage**: Evaluation of the extent to which test cases cover the application's functional and non-functional aspects.
- **Automation Coverage**: Assessment of automated test execution versus manual testing efforts.
- **Production Defects**: Tracking defects reported post-release to evaluate pre-release testing effectiveness.

Real-time monitoring and meaningful metrics are essential for a robust quality engineering process. Implementing structured tracking methods and reporting frameworks enables quality engineering teams to enhance efficiency, optimize test execution, and improve software reliability. Organizations must embrace metric-driven testing strategies to achieve superior product quality and agile software delivery.

5.2. Continuous Improvement: Learning from the Process

This research explores the role of continuous improvement in Quality Engineering (QE) by analyzing defect categorization, root cause analysis, and automation enhancements. Findings indicate that categorizing defects and refining automation strategies significantly impact quality engineering efficiency. The study discusses the implications of these findings for quality engineering professionals and organizations striving for enhanced software reliability.

5.2.1. Root Cause Analysis and Defect Categorization:

Continuous improvement is a fundamental aspect of the quality engineering process. This study investigates how systematic defect categorization and automation enhancements contribute to quality engineering efficiency. Prior research highlights the need for structured post-project analysis and optimized testing methodologies to reduce defects and improve software quality. This study employs a mixed-methods approach, incorporating qualitative analysis of defect root causes and quantitative assessment of automation efficiency. Data was collected from system integration testing (SIT), user acceptance testing (UAT), and production validation processes.

A critical component of quality engineering improvement is analyzing and categorizing defects post-project. Categorization by root cause facilitates targeted process enhancements in areas such as training, communication, and documentation.

5.2.2. Defect Categories:

- Code Fix
- Data/Config Issue
- Requirement Gap
- Environment
- Works as Designed
- Duplicate

Another important aspect of Defect analysis is to group the defects based on modules or components with its severity and type.

5.2.3 Defect Grouping:

- Modules/Components
- Type
- Severity

Defect Analysis Use case: 20% of critical defects under module Ex: Admin has a type of Regression which indicates, code quality of this module.

Action: Improve Unit Test coverage with increased level of code reviews.

5.3. Enhancing Automation for Regression Coverage:

Continuous refinement of automated testing frameworks contributes to increased regression coverage and reduced manual effort. Findings suggest that identifying critical test flows and analyzing failure patterns are key to improving automation efficiency.

- Expanding automation suite coverage to include critical workflows
- Analyzing test failure patterns to refine reliability of the automation
- Optimizing pipelines to improve speed and stability.
- Incorporating lessons learned into automation strategy

6. CONCLUSION

Effective testing is a multifaceted process that requires a strategic blend of structured methodologies and exploratory flexibility. By identifying gaps at different layers of a platform or application, teams can proactively address vulnerabilities and improve overall system reliability. Achieving comprehensive test coverage ensures that both

functional and non-functional aspects of the application are thoroughly validated, reducing the risk of unexpected failures. A systematic triage process enables teams to prioritize defects efficiently, ensuring that critical issues are addressed promptly without delaying development timelines.
Moreover, the importance of monitoring and producing meaningful metrics cannot be overstated, as these insights drive informed decision-making and continuous improvement. Metrics help teams track progress, assess risk, and refine testing strategies over time. Ultimately, fostering a culture of continuous assessment and identifying areas for improvement leads to a more resilient and high-performing software ecosystem. By integrating these best practices into the testing lifecycle, organizations can enhance software quality, optimize testing efficiency, and deliver a more seamless user experience.
Looking ahead, future initiatives should focus on leveraging emerging technologies such as AI and machine learning to enhance test automation, making defect detection more predictive and efficient. Additionally, continuous integration and continuous deployment (CI/CD) pipelines should be further optimized to integrate real-time monitoring and adaptive testing approaches. The development of industry-wide benchmarks for quality metrics can standardize assessment methods, ensuring consistency across different platforms and applications.
Furthermore, fostering stronger collaboration between development, operations, and testing teams through DevOps and shift-left testing approaches will help identify issues earlier in the software development lifecycle. Expanding exploratory testing techniques with domain-specific insights can also lead to more effective defect discovery. Lastly, investing in continuous skill development for testers will ensure that they stay ahead of evolving technologies, methodologies, and challenges in modern software testing.

REFERENCES

[1] Bilal Gonen; Dipali Sawant "*Significance of Agile Software Development and SQA Powered by Automation*". https://ieeexplore.ieee.org/document/9092149.

[2] Ogundipe, D.O., Odejide, O.A., & Edunjobi, T.E (2024). *Agile methodologies in digital banking: Theoretical underpinnings and implications for custom satisfaction.* Open Access Research Journal of Science and Technology, 2024, 10(02), 021-030. https://doi.org/10.53022/oarjst.2024.10.2.0045

[3] Leonor Barros a, Carlos Tam a, João Varajão b "Agile software development projects–Unveiling the human-related critical success factors", June 2024

[4] Boehm B. W., Brown, J. R., & Lipow, M. (1976). *Quantitative evaluation of software quality*. In *Proceedings of the 2nd International Conference on* Softw Eng *(ICSE), San Francisco (CA), USA, 13–15 October, 1976* (pp. 592–605). IEEE Computer Society Press

[5] Philipp Hohl, Jil Klünder, Arie van Bennekum, Ryan Lockard, James Gifford, Jürgen Münch, Michael Stupperich & Kurt Schneider *"Back to the future: origins and directions of the "Agile Manifesto" "*

[6] Wilson Rosa , Sara Jardine "A Novel Method to Early Agile Effort Estimation through Functional Initiatives", Nov 2024

[7] Marta Adzgauskaite , Carlos Tam , Ricardo Martins "What helps Agile remote teams to be successful in developing software? Empirical evidence" January 2025

[8] Delina Ly a b, Michiel Overeem a, Sjaak Brinkkemper b, Fabiano Dalpiaz b "The Power of Words in Agile vs. Waterfall Development: Written Communication in Hybrid Software Teams", Jan 2025

[9] Hao Dong , Nicholas Dacre, David Baxter, and Serkan Ceylan "What is Agile Project Management? Developing a New Definition Following a Systematic Literature Review" May 2024

[10] Chakraborty, T., Awan, T., Ashok, N. Kamran, M. *"Agile leadership for industry 4.0: An indispensable approach for the digital era" 2023*

[11] Bustard D, Wilkie G, Greer D (2013) *The diffusion of agile software development: insights from a regional survey. In: Pooley R, Coady J, Schneider C, Linger H, Barry C, Lang M (eds) Information systems development: reflections, challenges and new directions. Springer New York, New York, NY, pp 219–230*

[12] Mubarak Albarka Umar 1*, Chen Zhanfang 2 *"A Study of Automated Software Testing: Automation Tools and Frameworks" , 2019*

[13] Curtis B, Krasner H, Iscoe N *(1988) A field study of the software design process for large systems. Commun ACM 31(11):1268–1287*

[14] *IEEE (2006) Standard for developing a software project life cycle process. IEEE Standard:1074–2006*

[15] *ISO. (2011). ISO/IEC/IEEE 26515:2011 Systems and Softw Eng - Developing user documentation in an agile*

Environment

[16] A Fuggetta and Di Nino, "*Software process," in Future of Software Engineering, FOSE 2014 - Proceedings*. Association for Computing Machinery, Inc, May 2014, pp. 1-12

[17] Yunfeng Lu; Chao Li Song Wang ; Yang Liu; Jie Dai "*A Quality Evaluation Method for Software Testing about Safety-Critical Software" 2022*

[18] FATIH GURCAN 1 , GONCA GOKCE MENEKSE DALVEREN 2 , NERGIZ ERCIL CAGILTAY 2 , DUMITRU ROMAN 3 , AND AHMET SOYLU "Evolution of Software Testing Strategies and Trends" 2022

[19] G. Demartini and E Nasrabadi , "Quality Assurance of Software Products" 2017

[20] A. Agarwal, N.K. Garg, and A. Jain "Quality assurance for product development using Agile" 2014

[21] Beck, K. et al. (2001). Manifesto for Agile Software Development. Agile Alliance.

[22] North, D. (2006). Introducing BDD. Retrieved from https://dannorth.net/introducing-bdd

[23] Humble, J., & Farley, D. (2010). Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley.

[24] Fowler, M. (2019). The State of DevOps. Thoughtworks.

[25] Yang, C. Cardie. Context-aware Learning for Sentence-level Sentiment Analysis with Posterior Regularization. In Proceedings of ACL 2014.

[26] Xin Ye, Hui Shen, Xiao Ma, Razvan Bunescu, and Chang Liu. From word embeddings to document similarities for improved information retrieval ni software engineering. In International Conference on Software Engineering, pages 404-415, 2016.

[27] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025, 2015.

[28] John Anvik, Gail C Murphy. Reducing the effort of bug report triage: Recommenders for development-oriented decisions. ACM Transactions on Software Engineering, 2011.

[29] Zhang ,T Yang G, Le B, et al. ANovel Developer Ranking Algorithm for Automatic Bug TriageUsing Topic Model and Developer Relations [C]. Proceedings of the 21st Asia-Pacific Software Engineering Conference (APSEC'14), 2014, 1: 223-230.

[30] Cubranic D, Murphy G C. Automatic bug triage using text categorization [C]. Proceedings of the Sixteenth International Conference on Software Engineering &Knowledge Engineering, 2004.

[31] Ahsan S N, Ferzund J, Wotawa F, et al. Automatic Software Bug Triage System (BTS) Based on Latent Semantic Indexing and Support Vector Machine [C]. Proceedings of the Fourth International Conference on Software Engineering Advances (ICSEA'09), 2009: 216-221.

[32] Jifeng Xuan, He Jiang, Yan Hu, Zhilei Ren, Weign Zou, Zhongxuan Luo, and Xindong Wu. Towards effective bug triage with software data reduction techniques. IEEE Transactions onKnowledge and Data Engineering, 27(1):264-280, 2015.

[33] Yang G, Zhang T, Lee B, et al. Towards Semi-automatic Bug Triage and Severity Prediction Based on Topic Model and Multi-feature of Bug Reports [Cl. Proceedings of the IEEE 38th Annual Computer Software and Applications Conference (COMPSAC' 14), 2014: 97-106.

[34] Philip J Guo, Thomas Zimmermann, Nachiappan Nagappan, and Brendan Murphy. Characterizing and predicting which bugs get fixed: an empirical study of microsoft windows. In International Conference on Software Engineering, volume 1, pages 495-504, 2010.

[35] Ali Sajedi Badashian, Abram Hindle, and Eleni Stroulia. Crowdsourced bug triaging. In International Conference on Software Maintenance and Evolution, pages 506-510. IEEE, 2015.

[36] Osvaldo, S.S., Jr.; Lopes, D.; Silva, A.C.; Abdelouahab, Z. Developing software systems to Big Data platform based on MapReduce model: An approach based on Model Driven Engineering. *Inf. Softw. Technol.* **2017**, *92*, 30–48. [CrossRef]

[37] Enes, J.; Exposito, R.R.; Tourino, J. BDWatchdog: Real-time monitoring and profiling of Big Data applications and frameworks. *Future Gener. Comput. Syst.* **2018**, *87*, 420–437. [CrossRef]

[38] Zareian, S.; Fokaefs, M.; Khazaei, H.; Litoiu, M.; Zhang, X. A big data framework for cloud monitoring. In Proceedings of the 2nd International Workshop on BIG Data Software Engineering, Austin, TX, USA, 16 May 2016; pp. 58–64.

Journal of Information Systems Engineering and Management
2025, 10(37s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/
Research Article

[39] Casale, G.; Ardagna, D.; Artac, M.; Barbier, F.; Di Nitto, E.; Henry, A.; Iuhasz, G.; Joubert, C.; Merseguer, J.; Munteanu, V.I. DICE: Quality-driven development of data-intensive cloud applications. In Proceedings of the 2015 IEEE/ACM 7th International Workshop on Modeling in Software Engineering, Florence, Italy, 16–17 May 2015; pp. 78–83.
[40] Fang, K.; Li, X.; Hao, J.; Feng, Z. Formal modeling and verification of security protocols on cloud computing systems based on UML 2.3. In Proceedings of the 2016 IEEE Trustcom/BigDataSE/ISPA, Tianjin, China, 23–26 August 2016; pp. 852–859.
[41] Klein, J.; Gorton, I.; Alhmoud, L.; Gao, J.; Gemici, C.; Kapoor, R.; Nair, P.; Saravagi, V. Model-driven observability for big data storage. In Proceedings of the 2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA), Venice, Italy, 5–8 April 2016; pp. 134–139.
[42] Amato, F.; Moscato, F. Model transformations of mapreduce design patterns for automatic development and verification. *J. Parallel Distrib. Comput.* **2016**, *110*, 52–59. [CrossRef]
[43] Majd, A.; Troubitsyna, E. Data-driven approach to ensuring fault tolerance and efficiency of swarm systems. In Proceedings of the 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, USA, 11–14 December 2017; pp. 4792–4794.
[44] 100. Li, H.; Wu, J.; Jiang, Z.; Li, X.; Wei, X. Minimum backups for stream processing with recovery latency guarantees. *IEEE Trans. Reliab.* **2017**, *66*, 783–794. [CrossRef]
[45] Shang,W.;Jiang,Z.M.;Hemmati,H.;Adams,B.;Hassan,A.E.;Martin,P.Assistingdevelopersofbigdata analytics applications when deploying on hadoop clouds. In Proceedings of the 2013 35th International Conference on Software Engineering (ICSE), San Francisco, CA, USA, 18–26 May 2013; pp. 402–411.
[46] Guo, C.; Zhu, S.; Wang, T.; Wang, H. FeT: Hybrid Cloud-Based Mobile Bank Application Testing. In Proceedings of the 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), Lisbon, Portugal, 16–20 July 2018; pp. 21–26.
[47] C. Jones, 'Software Quality in 2012: a Survey of the State of the Art', 2012.
[48] NIST Report, 'The Economic Impacts of Inadequate Infrastructure for Software Testing', 2002.
[49] L. Luo, 'A Report on Software Testing Techniques', Pittsburgh, USA.
[50] A. Dennis, B. H. Wixom, and D. Tegarden, Systems Analysis and Design with OOP Approach with UML 2.0, 4th Editio. USA: John Wiley & Sons, Inc., 2009.
[51] A. Dennis, B. H. Wixom, and R. M. Roth, Systems Analysis and Design 5th Edition, 5th Editio. USA: John Wiley & Sons, Inc., 2012.
[52] C.Padmini,'BeginnersGuideToSoftwareTesting',pp.1–41,2013.
[53] Automated Software Testing - International Software Test Institute'. [Online]. Available: https://www.test-institute.org/Automated_Software_Testing.php. [Accessed: 18-Nov-2019].
[54] M.Polo,P.Reales,M.Piattini,andC.Ebert,'Testautomation',IEEESoftw.,vol.30,no.1,pp.84–89,2013.
[55] M. A. Umar, 'Comprehensive study of software testing : Categories , levels , techniques , and types', International Journal of Advance Research, Ideas and Innovations in Technology, vol. 5, no. 6, pp. 32–40, 2019.
[56] D.HuizingaandA.Kolawa,Automateddefectprevention.Hoboken,N.J.:Wiley-Interscience,2007.
[57] difference between unit functional acceptance and integration testing - StackOverflow'. [Online]. Available: https://stackoverflow.com/questions/4904096/whats-the-difference-between-unit-functional-acceptance-and-integration-test.
[58] 'FunctionalTestingWikipedia'.[Online].Available:https://en.wikipedia.org/wiki/Functional_testing
[59] Stackify,'CodeCoverageTools:25ToolsforTestinginC,C++,Java'.[Online].Available:https://stackify.com/code-coverage-tools/. [Accessed: 29-Sep-2019].
[60] Atlassian, 'About Code Coverage - Atlassian Documentation', 2017. [Online]. Available:https://confluence.atlassian.com/clover/about-code-coverage-71599496.html. [Accessed: 29-Sep-2019].
[61] SeaLights.io, 'Code Coverage vs. Test Coverage: Pros and Cons | SeaLights'. [Online]. Available: https://www.sealights.io/test- metrics/code-coverage-vs-test-coverage-pros-and-cons/. [Accessed: 29-Sep-2019].

[62] K. Sneha and G. M. Malle, 'Research on software testing techniques and software automation testing tools', 2017 Int. Conf. Energy, Commun. Data Anal. Soft Comput. ICECDS 2017, pp. 77–81, 2017.
[63] 'PerformanceTestingWikipedia'.[Online].Available:https://en.wikipedia.org/wiki/Software_performance_testing.
[64] E. Khan, 'Different Forms of Software Testing Techniques for Finding Errors', Int. J. Comput. Sci. Issues, vol. 7, no. 3, pp. 11–16, 2010.
[65] Aebersold Kirsten, 'Software Testing Methodologies'. [Online]. Available: https://smartbear.com/learn/automated-testing/software- testing-methodologies/. [Accessed: 10-May-2019].
[66] H.Bajaj,'CHOOSINGTHERIGHTAUTOMATIONTOOLANDFRAMEWORK',Bengaluru,India.
[67] Aebersold Kirsten, 'Test Automation Frameworks'. [Online]. Available: https://smartbear.com/learn/automated-testing/test- automation-frameworks/. [Accessed: 26-Aug-2019].
[68] Bhargava, Designing and implementing test automation frameworks with QTP : learn how to design and implement a test automation framework block by block. Packt Publishing, 2013.
[69] 'A Guide to Automation Frameworks |Smartsheet'. [Online]. Available: https://www.smartsheet.com/test-automation-frameworks- software. [Accessed: 30-Aug-2019].
[70] W.E.Perry,EffectiveMethodsforSoftwareTesting:IncludesCompleteGuidelines,Checklists,andTemplates,Third Edit.2007.
[71] S.NidhraandJ.Dondeti,'BlackBoxandWhiteBoxTestingTechniques',Int.J.Embed.Syst.Appl.,vol.2,no.2,pp.29–50,2012.]
[72] C.Abraham,'TestAutomationToolEvaluation',TamilNadu,India.
[73] C.Oriat,'Jartege:AtoolforrandomgenerationofunittestsforJavaclasses',Springer-VerlagBerlinHeidelb.,pp.242–256,2005.
[74] 'JUnit-Wikipedia'.[Online].Available:https://en.wikipedia.org/wiki/JUnit.[Accessed:19-Nov-2019].
[75] Altexsoft.com, 'Comparing Automated Testing Tools: Selenium, TestComplete, Ranorex, and more | AltexSoft', 2018. [Online]. Available: https://www.altexsoft.com/blog/engineering/comparing-automated-testing-tools-selenium-testcomplete-ranorex-and-more/. [Accessed: 19-Nov-2019].
[76] 'SeleniumHQBrowserAutomation'.[Online].Available:https://selenium.dev/.[Accessed:19-Nov-2019].
[77] 'A Comparison of Automated Testing Tools | Katalon Studio'. [Online]. Available: https://www.katalon.com/resources- center/blog/comparison-automated-testing-tools/. [Accessed: 17-May-2019].
[78] Brian, 'Best Automation Testing Tools for 2020 (Top 10 reviews)', 2017. [Online]. Available: https://medium.com/@briananderson2209/best-automation-testing-tools-for-2018-top-10-reviews-8a4a19f664d2. [Accessed: 20-Nov- 2019].
[79] Dai, H.N.; Wong, R.C.W.; Wang, H.; Zheng, Z.; Vasilakos, A.V. Big data analytics for large-scale wireless networks: Challenges and opportunities. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 1–36. [CrossRef]
[80] Chen, C.P.; Zhang, C.Y. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Inf. Sci.* **2014**, *275*, 314–347. [CrossRef]
[81] Allam, Z.; Dhunny, Z.A. On big data, artificial intelligence and smart cities. *Cities* **2019**, *89*, 80–91. [CrossRef]
[82] Tao, C.; Gao, J. Quality Assurance for Big Data Applications: Issues, Challenges, and Needs. In Proceedings of the Twenty-Eighth International Conference on Software Engineering and Knowledge Engineering, Redwood City, CA, USA, 1–3 July 2016.
[83] Jan, B.; Farman, H.; Khan, M.; Imran, M.; Islam, I.U.; Ahmad, A.; Ali, S.; Jeon, G. Deep learning in big data analytics: A comparative study. *Comput. Electr. Eng.* **2019**, *75*, 275–287. [CrossRef]
[84] Hilbert, M. Big Data for Development: A Review of Promises and Challenges. *Dev. Policy Rev.* **2016**, *34*, 135–174. [CrossRef]
[85] Laranjeiro, N.; Soydemir, S.N.; Bernardino, J. A Survey on Data Quality: Classifying Poor Data. In Proceedings of the IEEE Pacific Rim International Symposium on Dependable Computing, Zhangjiajie, China, 18–20 November 2015.
[86] Anagnostopoulos, I.; Zeadally, S.; Exposito, E. Handling big data: Research challenges and future directions.

J. Supercomput. **2016**, *72*, 1494–1516. [CrossRef]
[87] Gudivada, V.N.; Baezayates, R.; Raghavan, V.V. Big Data: Promises and Problems. *Computer* **2015**, *48*, 20–23. [CrossRef]
[88] Montagud, S.; Abrahão, S.; Insfran, E. A systematic review of quality attributes and measures for software product lines. *Softw. Qual. J.* **2012**, *20*, 425–486. [CrossRef]
[89] Nguyen-Duc, A.; Cruzes, D.S.; Conradi, R. The impact of global dispersion on coordination, team performance and software quality–A systematic literature review. *Inf. Soft. Technol.* **2015**, *57*, 277–294. [CrossRef]
[90] Wang, H.; Xu, Z.; Fujita, H.; Liu, S. Towards Felicitous Decision Making: An Overview on Challenges and Trends of Big Data. *Inf. Sci.* **2016**, *367–368*, 747–765. [CrossRef]
[91] Bagriyanik, S.; Karahoca, A. Big data in software engineering: A systematic literature review. *Glob. J. Inf. Technol. Emerg. Technol.* **2016**, *6*, 107–116. [CrossRef]
[92] Schulmeyer, G.G.; McManus, J.I. *Handbook of Software Quality Assurance*; Van Nostrand Reinhold Co.: New York, NY, USA, 1992.
[93] Gao, J.; Xie, C.; Tao, C. Big Data Validation and Quality Assurance —Issuses, Challenges, and Needs. In Proceedings of the 2016 IEEE Symposium on Service-Oriented System Engineering, Oxford, UK, 29 March–2 April 2016; pp. 433–441.
[94] Lai, S.T.; Leu, F.Y. Data Preprocessing Quality Management Procedure for Improving Big Data Applications Efficiency and Practicality; In Proceedings of the International Conference on Broadband and Wireless Computing, Communication and Applications, Asan, Korea, 5–7 November 2016; pp. 731–738.
[95] Garg, N.; Singla, S.; Jangra, S. Challenges and techniques for testing of big data. *Procedia Comput. Sci.* **2016**, *85*, 940–948. [CrossRef]
[96] Zhou, H.; Lou, J.G.; Zhang, H.; Lin, H.; Lin, H.; Qin, T. An Empirical Study on Quality Issues of Production Big Data Platform. In Proceedings of the IEEE/ACM IEEE International Conference on Software Engineering, Florence, Italy, 16–24 May 2015; pp. 17–26.
[97] Juddoo, S. Overview of data quality challenges in the context of Big Data. In Proceedings of the International Conference on Computing, Communication and Security, Pamplemousses, Mauritius, 4–5 December 2016.
[98] Zhang, P.; Zhou, X.; Gao, J.; Tao, C. A survey on quality assurance techniques for big data applications. In Proceedings of the IEEE BigDataService 2017—International Workshop on Quality ASsurance and Validation for Big Data Applications, San Francisco, CA, USA, 6–9 April 2017.
[99] Ge, M.; Dohnal, V. Quality Management in Big Data. *Informatics* **2018**, *5*, 19. [CrossRef]
[100] worldqualityreport.com,‘WorldQualityReport2019’,2018.
[101] International Conference on Software Maintenance and Evolution, pages 506-510. IEEE, 2015.
[102] Bhattacharya P, Neamtiu I, Shelton CR, et al. Automated,highly-accurate, bug assignment using machine learning and tossing graphs (J. Journal of Systems and Software, 2012, 85(10): 2275-2292.
[103] Anvik J, Murphy G C. Reducing the effort of bug report triage: Recommenders for development-oriented decisions (JJ. ACM Transactions on Software Engineering and Methodology, 2011, 20(3): 10:1-10:35.
[104] Park J, Lee M, Kim J, et al. COSTRIAGE: a cost-aware triage algorithm for bug reporting [C]. Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI11), 2011:139-144.
[105] Sentil Mani, Anush Sankaran, Rahul Aralikatte, Exploring the Effectiveness of Deep Learning for Bug Triaging, 2016
[106] G. Jeong, S. Kim, T. Zimmermann, Improving Bug Triage with Bug Tossing Graphs, in: FSE, 2009.