AI and Machine Learning Advances
Volume 2, Issue 2
ISSN: 3067-3216

Engineering Excellence Starts with Quality: Why Global Product Teams Must Own It Together

[1]*Chandrani Gupta Chowdhury
[1]Colorado Technical University, USA.

Abstract

Even with the improvement of tools and technologies, defect leakage continues to be a problem for multinationals product engineering teams. As systems become more complex due to distributed development, cloud-native platforms and microservices, the traditional approach to quality assurance (QA) is becoming less effective. This paper examines the reasons for leakage of defects into global product teams and suggests the Shared Quality Ownership Framework for spreading the responsibility for quality throughout the life of the product. The aim is to move away from "reactive" QA and towards a more "proactive" and collaborative quality management. This conceptual review combines the theories behind software quality, Shift-Left Engineering, DevOps, and organizational behavior. The Shared Quality Ownership Framework is designed, focusing on team collaboration across engineering, product management, design, Quality Engineering (QE), DevOps/SRE, and product market activation teams. It's all about predicting future defects and ongoing feedback cycles. The framework facilitates early defect identification, improved product stability, and quicker release cycles. It creates better functional coordination, decreases the escape rate of defects, and increases release confidence, and encourages a culture of shared responsibility. When quality is built into the product throughout the design process, it not only stabilizes the product, but also streamlines the innovation process and decreases the number of defects. The Shared Quality Ownership approach creates engineering excellence and enhances product teams around the world.

Keywords: Engineering excellence; Shared quality ownership; Quality engineering; Shift-left testing; Defect leakage; Global product teams.

1. Introduction

The global product engineering environment has changed in recent years with the advent of virtual development teams, global collaboration and simultaneous releases of products for a global audience (Ferguson et al., 2022). Many companies are not building software for the domestic market; they are developing, integrating and pushing out products for the world, across several time zones, and across different regulatory, operational and customer scenarios. This globalization of software delivery has meant that things move faster, and it is difficult to coordinate. Adapting to the agile mindset, DevOps culture and continuous deployment has sped up innovation, requiring more rapid product delivery (Ayyash, 2024). Simultaneously, product technology has become more complex with cloud-native platforms, microservices, API-based connections and AI-based capabilities (Ogunmolu et al., 2026). These enhancements boost scalability and innovation opportunities, but a higher degree of interdependencies, interoperability and variability. In this new environment, quality assurance must move from isolated activities to a strategic capability that spans the whole life of the product. Many organizations are still practicing traditional forms of quality assurance in which testing is still performed after the design and development of the product has occurred. This downstream view of QA, a gatekeeper view of quality, can result in defects being made, delays in market release and a higher cost of defects. Defects found in Market User Testing (MUAT) or post-deployment are often due not only to technical but also communication, governance and ownership issues. No ownership across engineering, product, QA and marketing results in blind spots where problems are pushed. This means product delivery timelines are impacted, customers are not confident, and global teams are forced to rework, triage and fix issues instead of building product.

Despite the large emphasis on automation, shift-left and DevOps in software engineering literature, current

Chandrani Gupta chowdhury
Colorado Technical University, USA.
Email: gupta.chandrani@gmail.com

Received: 2-May-2026 Revised: 14-May-2026 Accepted: 26-May-2026

AI and Machine Learning Advances
Volume 2, Issue 2
ISSN: 3067-3216

debates are still focused on technology and not governance (Kavuri, 2024; Rani et al., 2023). Existing models tend to under-prioritize how to apportion quality ownership among product managers, designers, developers, Quality Engineering (QE), operations and the marketplace. This leaves a significant gap: organizations can treat their software testing as high-end testing tools, but still have quality issues due to poor quality governance. There is little conceptual integration between practices of engineering excellence and quality governance at the organization level, especially in global product ecosystems.

This paper contends that quality needs to shift from the domain of QA to a collaborative product ownership within an organization. The main aim is to create a conceptual Shared Quality Ownership Model that combines engineering, product, QE and market collaboration to provide a holistic perspective for global product teams. This research, which integrates software engineering, organizational behaviour and operational excellence, aims to redefine engineering excellence in terms of shared quality ownership, rather than individual testing efficiency.

2. Literature Review and Theoretical Foundation

2.1 Software Quality and Defect Leakage Models

Software quality has traditionally been viewed as a multi-dimensional attribute that includes functional correctness, reliability, usability, maintainability and operational resilience (Zakeri et al., 2025). Early software quality theories, such as defect lifecycle models, view defects as problems that are introduced in requirements, design, coding, integration or deployment, with distinct potential for defects to be prevented or magnified. Early lifecycle models postulated that defects are not exclusively introduced in the code, but are often introduced in the requirements, architecture or communication, prior to the technical implementation (Otieno et al., 2023). This understanding evolved software quality from defect detection to prevention. Defect leakage is when issues are not found during one cycle of product verification and are found in subsequent phases, such as system integration testing, user acceptance testing or production (Potla, 2021). Leakage is a critical issue as escaped defects tend to be compounded in the downstream chains, which increases the complexity and disrupts the business more.

The principle of rising cost of defect is a widely known model in the software industry (Thota et al., 2020). The model defines the cost of defects found later in the software life cycle to grow the software development or delivery cycle. If defects are detected early in the process (e.g. the design phase), they may only require slight corrections. On the other hand, defects found during deployment may lead to large rework, product launch delay, customer loss and brand degradation. This is empirically investigated in studies which demonstrate that the delay of the defect correction (causing technical debt) also results in organizational inefficiencies such as rework, disputes and suboptimal resource misallocation. In global product ecosystems, the consequence of defect leakage has increased to the product's strategic level, affecting launch schedules, product credibility and scalability (Ilufoye et al., 2023).

2.2 Shift-Left Testing and Continuous Quality

To avoid late-stage testing, shift-left testing is a predictive approach that prioritizes defect prevention in early stages of development (Patel, 2026). Shift-left transforms quality from a goal set at the end of the process to an integrated and continuous process that begins with the gathering of requirements and then spans design, development, deployment and operations. Rather than postponing quality, the team include quality tests at the time of defect creation. That is, architecture reviews, requirements verification, static and dynamic analysis, unit and integration tests. The theory is sound: the earlier the repair of defects, the lower the cost of correction, the time required to fix them, and the greater product stability.

The emergence of cultural and technological DevOps and CI/CD practices and platforms improved shift-left practices by embedding quality in the SDLC process(Heijstek, 2023; Hirsi, 2025). Continuous Integration (CI) enables regularly integrating, automatically rebuilding and testing software to avoid bugs. Continuous and Continuous Deployment (CD) processes to enhance efficiency further by automating pipelines and establishing quality gates (Sannapureddy et al., 2021). Automated test and verification (such as regression) testing, API contract testing and infrastructure-as-code testing reduce human intervention and increase repeatability. More importantly, shift-left is not a technology practice, but alters the culture and communication, as it makes developers, product managers and operations stakeholders think about quality earlier, in a more collective manner (Horshchar, 2025). The latest trend is to consider shift-left as an engineering and governance revolution, which "normalizes" responsibility.

2.3 Engineering Excellence Frameworks

Engineering excellence is not only about individual capabilities but also operational excellence, reliability and learning. The latest excellence models deal with quantitative measures of business/engineering performance. The most prominent is the DORA (DevOps Research and Assessment) model, with four metrics: deployment frequency, lead time for changes, change failure rate and mean time to recovery (MTTR) (Thason & Singh). These metrics have revolutionized discussions on software delivery because they offer measurable links between speed and quality. Top performing teams are not only quick at deployment, but also sustain high speed, low failure and quick recovery.

Reliability engineering, such as with principles of Site Reliability Engineering (SRE), complements DORA in focusing on system reliability, observability and availability (Saarinen, 2024). Reliability systems prioritize the upfront observability of the system, service level objectives (SLOs) and rigor. Overall, these approaches imply that engineering excellence is achieved when variability and variability are minimized, automation is applied, and quality processes are properly established. Operations maturity models place enterprises in an order of their progression from being reactive in quality to predictive and data-driven. These visions reinforce that engineering excellence is not simply engineering success but an integrated capability reliant on technical, governance and cultural maturity.

2.4 Organizational Behavior and Accountability Models

Technical models are important, but as research in organizational behaviour shows, they are dependent on governance, leadership and accountability (Bankins et al., 2024). Functional ownership approaches say quality not only fails to achieve its goals because of technical inadequacies but also because of distributed accountability. Most common functional structures (build, QA, product, marketing), with passing the baton from one to another, cause coordination problems. Research shows that ambiguity in role ownership results in higher coordination costs, lengthened feedback loops, and the tendency to subgame-play versus fixing problems.

Models of shared accountability emphasize shared accountability, as each function focuses on prevention in its sphere of influence (Saheb & Saheb, 2024). It is important in this model to ensure that executive incentives, measures, and decisions are in line with quality priorities. This concept is built on a quality culture transformation, which institutionalizes quality, becomes a culture and standard, not something that is limited to quality departments. This process is facilitated through the support of leaders, knowledge management and process re-engineering. As a consequence, the recent quality literature now increasingly focuses on one theme: engineering excellence can't be delivered through just better tools or automation, but by systems that guarantee technical excellence and account for collective ownership, governance consistency and quality culture.

3. Problem Diagnosis: Why Defect Leakage Persists

3.1 Product Ecosystems with a Fragmented Sense of Ownership

Today, product teams still have a "leaky" model for quality because of the way they work—functional areas. Product development ecosystems are often focused on product development (engineering), quality assurance (QA), business enablement (product) and market (market) (Tang, 2021). This separation might make sense from an operational perspective, but it leads to an isolated accountability model for quality. Engineering might emphasize speed of delivery, QA speed of testing, product teams' speed of release and market teams' speed of deployment. With fragmentation comes a lack of ownership at each stage and a chance for bugs to be missed through the cracks. This "handoff" approach creates an atmosphere of transactional passing of work, rather than collaborative refinement, leading to a disconnect in the process. Rather than working together to prevent defects, issues are often detected at handoff points. This creates a lack of shared awareness and permits quality problems to flow through development, integration and deployment.

3.2 Late-Stage Validation Dependency

The second key driver of quality leakage is that organizations still place a heavy focus on late-stage validation activities, primarily in the form of Market User Acceptance Testing (MUAT) and post-development quality reviews (Alphonce, 2024). Many enterprises still focus quality efforts at the end of the lifecycle, where testing is used as a check rather than an integral part of the process. MUAT often represents the first time that the business process, market or the readiness for operation is truly tested. This can cause bottlenecks as defects identified here are often ingrained in product architecture, integration or assumptions, and are more difficult to fix. Rather than functioning as strategic validation, MUAT can

AI and Machine Learning Advances
Volume 2, Issue 2
ISSN: 3067-3216

devolve into reactive defect discovery. This then requires a review of defects where priorities are assessed against the need to release the product. Triage can concentrate on the symptoms of the problems and not on the root issues, and leave the issues unresolved. In the end, there are multiple schedule slippages, unstable deployments and quality issues.

3.3 Root Causes of Quality Failure

In addition to a lack of ownership and delayed testing, defect leakage is an issue of systemic problems (Mahmood, 2022). Deficiencies in communication between teams and blocking of requirements, acceptance criteria and readiness to ship, especially with remote product development. Environmental differences between development, testing and production environments lead to hidden defects by allowing software to have different behaviours in different environments. Incentives are the primary source of defect leakage, as teams with performance incentives to deliver fast, release, and productivity may forgo prevention efforts aimed at quality (Li & Hazelrigg, 2023). Under performance criteria, quality is neglected for local objectives. All this indicates that the defect is not a testing problem, but a bad governance and system design problem.

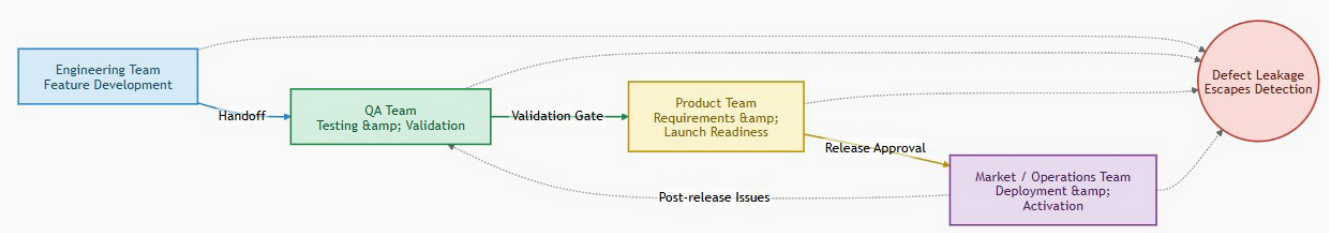

Figure 1. Traditional Silo-Based Quality Model and Defect Leakage Pathways

4. Shared Quality Ownership Framework

4.1 Framework Overview

To solve the problem of defect leakage, we need to rethink not just processes but the governance, practices and nature of global product quality. This research introduces the notion of collective quality ownership, where quality is no longer regarded as a back-verification, but rather is spread across the entire lifecycle and integrated into all product functions. Rather than quality being "owned" by a QA or Quality Engineering (QE) group, this approach sees quality as a corporate asset, with each entity having a responsibility to eliminate its defects and meet success criteria. product success. This approach embeds quality from inception to post-deployment activities in a field; each phase in the lifecycle is both product and process.

This will eliminate the governance leak of defects from gate to systems design. Instead of quality being "tested" after the fact, it is progressively built into requirements, architecture and design, development, deployment and enablement. This enables organizations to manage handover risks, improve integrity and develop resilience. The model stresses that defects are not just caused by software coding problems, but by problems with requirements, user experience, operational and market fit. Thus, all functions play an integral role in preventing defects.

4.2 Role-Based Quality Academy

The Shared Quality Ownership Framework defines quality in terms of role-based accountability and maintaining shared ownership. Specialized preventive controls by stakeholders are introduced to address their role-specific needs.

Engineers are accountable for code and system execution quality (Alami & Ernst, 2024). They are responsible for unit testing, integration testing, code reviews, architectural quality and secure design. Engineers are the first line of defence against defects by building quality into the product. Moreover, product managers are responsible for quality through vision (Mulla, 2023). They set acceptance criteria, clarify requirements, focus on business value and eliminate backlog confusion. Product requirements too often lead to downstream defects; thus, product quality starts with product clarity.

Similarly, Designers add value through user-friendliness, accessibility, user flow and human validation (Rechkemmer et al., 2025). Design issues can cause misalignment when code is correct, making design quality a trust factor.

Likewise, Quality Engineering (QE) evolves from test execution to enablement. QE is responsible for test architecture, test automation frameworks, test management, risk-based testing and release metrics (Bittla, 2025; Perla). It is a force multiplier, rather than just a gatekeeper.

Additionally, DevOps / Site Reliability Engineering (SRE) teams offer operational quality assurance in the form of infrastructure standardization, deployment automation, observability, resilience engineering and incident management services. They avoid drift and high variability of the environment and deployment.

Similarly, Markets / Regional Activation Teams verify location readiness, ensuring market regulatory, localization and operational readiness prior to activation. Their contribution evades activation issues.

Table 1. Shared Quality Responsibility Matrix

Function	Primary Quality Responsibility	Preventive Focus
Engineers	Code and architecture integrity	Unit, integration, API quality
Product Managers	Requirement precision	Acceptance criteria, backlog clarity
Designers	Experience quality	UX, accessibility, workflow logic
QE	Governance and automation	Test frameworks, metrics, strategy
DevOps/SRE	Operational resilience	CI/CD, observability, deployment stability
Markets	Activation readiness	Localization, compliance, market fit

4.3 Quality Feedback Loops

An important part of this practice is replacing handoffs with quality feedback loops. Traditionally, the delivery of products is done in a handoff manner, in which each group in the organization finishes their work and passes it on to the next group. This leads to a delay in feedback and escaped defect rates. However, with Shared Quality Ownership, there's continuous bi-directional validation. These loops are based on shared metrics. Defect escape rate, deployment stability, change failure rate, acceptance clarity, market readiness metrics and others are examples of shared metrics (Daraojimba et al., 2024). This drives aligned incentives and decisions.

Early collaboration is equally critical. PMs and designers collaborate with engineers, QE, DevOps and the market early in the cycle to experiment together. These are dynamically adapted as requirements, architecture and operational aspects are taken into account. This prevents defects from being created. Lifecycle continuity inhibits rebooting quality. Responsibility continuity is maintained between each phase: design to development, development to testing, testing to deployment and deployment to market (Ugwueze & Chukwunweike, 2024). Integrated dashboards, retrospectives and governance gates support this, ensuring continuous improvement from feedback.

4.4 Value of Shared Ownership

Beyond removing defects, Shared Quality Ownership has other advantages. First, decentralized accountability to early detect defects limits defect escapes (Makanda et al., 2023). With quality checks throughout the requirements, design, development and deployment, the earlier we can catch defects.

Second, release confidence improves. Firms that adopt shared quality controls benefit from increased predictability because quality is established over time, rather than in fits and starts. Deployment schedules become more predictable, MUAT delays are alleviated, and trust in deployment integrity is enhanced.

Third, innovation velocity increases. Contrary to the myth that increased governance increases delivery time, quality embedded into the process decreases the rework cycle, allowing more time for innovation. By reducing firefighting, delivery is sped up.

Finally, shared responsibility enhances corporate culture with systems thinking (Lapatoura, 2025). The outcome of this principle will help to drive a global culture of quality across ecosystems by changing the mindset of teams from reactive to proactive when dealing with faults.

5. Engineering Excellence Practices and Shift-Left Integration

5.1 Operationalizing Shift-Left Quality

Engineering excellence is operationalized when quality is built in as early as possible in the development process rather than left for later (Zonnenshain & Kenett, 2020). Shift-left quality is the operationalization of this concept as prevention is integrated into planning, design and development processes. Rather than testing being a downstream process, shift-left approaches implement quality at the source, where problems cost the least to fix and are least disruptive.

This approach uses a layer of unit testing to ensure that each element of the system works as designed within

AI and Machine Learning Advances
Volume 2, Issue 2
ISSN: 3067-3216

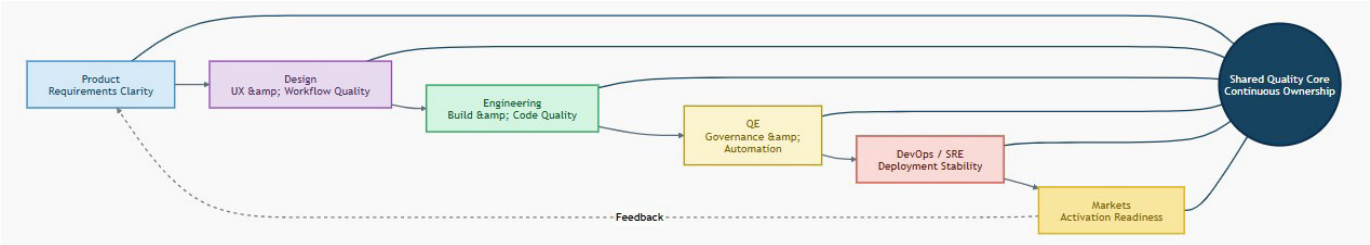

Figure 2. Shared Quality Ownership Ecosystem Model

the context of the system before integration. Through code module-level functional validation, defects can be immediately identified and their spread limited. Integration testing further safeguards quality by testing the interaction of services, APIs and modules across layered systems. In a world of increasingly distributed systems, with microservices and cloud-native platforms, integration issues can cause significant downstream disruption, so it's important to validate interfaces early in development (Harve et al., 2024; Raj et al., 2022). Contract testing further strengthens quality by ensuring that service-level expectations between systems remain consistent despite independent development cycles. This is crucial for global ecosystems with multiple teams working on interdependent services.

Unit testing is complemented with static analysis and architecture validation to shift-left quality control from testing to architecture (Horshchar, 2025; Patel, 2026). Static analysis software identifies security, maintainability, and other code issues before it runs, eliminating latent defects (Kader & Ahmed, 2025). Architecture validation verifies that the design of a system is made with consideration of scalability, resiliency and operational processes before the development of the system is undertaken. These measures will help to shift quality from a reactive to a proactive engineering practice.

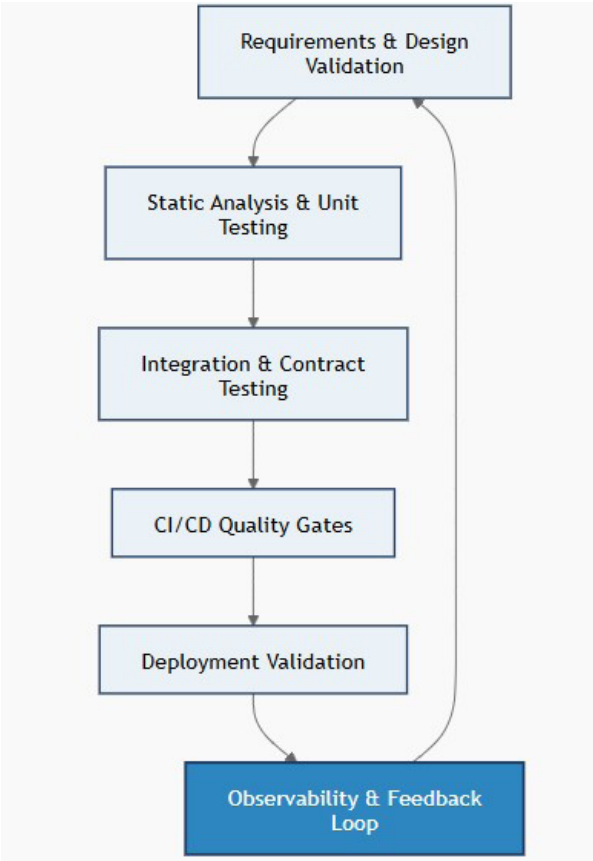

Figure 3. Shift-Left Engineering Excellence Pipeline

5.2 Automation: A Quality Stabilizer

Shift left helps to embed quality in the early phases of the process, while automation offers the capacity and repeatability to support enterprise-wide engineering quality. Automation is a quality stabilizer, reducing human error, increasing speed and enforcing consistent governance (Joseph et al., 2024). Continuous Integration and Continuous Delivery (CI/CD) pipelines are key to this in today's product ecosystems.

CI/CD quality gates establish formal gates that automate checks against quality criteria (Esther, 2024). These gates can include test coverage and results from unit and integration testing, security scanning, compliance checks, and deployment checks. Through the automation of such controls, organizations will eliminate the chance of making varying decisions and the possibility of failing quality on the basis of subjective assessment due to deadlines.

Moreover, observability enhances quality by bringing testing to production (Abieba et al., 2025). Testing can confirm that things work as expected, but observability tools (which can include logs, metrics, tracing and performance monitoring) can track for anomalies. This visibility permits detection of hidden bugs, performance problems or service issues that could impact customers. Observability, as such, is a proactive and reactive quality strategy. Infrastructure consistency is equally critical. Variation between development, test, staging and production environments causes bugs. Infrastructure-as-code, containerization, and templates to deploy applications help minimize this by making environments consistent throughout the pipeline. Environment consistency improves software predictability, eliminates surprise integration problems and raises software confidence.

5.3 Quality Engineering as Strategic Enablement

With improvements to engineering systems, QA shifts from manual gatekeeping to a Quality Engineering (QE) enabler. Older models consider QA as a gatekeeper at the end of the development process, and perhaps just defect detection. This approach is not enough for the speed, complexity and global dynamics of today's product ecosystems. Quality Engineering restructures this role from being solely focused on testing to managing quality systems. QE is responsible for scalable automation platforms, governance policies, test architecture, and release-readiness metrics and integrated test ecosystems. QE facilitates "shift left", rather than being a blocker. These include the development of reusable automation frameworks, the establishment of enterprise-wide quality policies and the integration of governance into CI/CD.

Tools, governance, and quality systems make QE an enterprise capability, rather than a function. QE ensures engineering, product, operations and market stakeholders are aligned to standards and that quality systems grow with the complexity of the business (Bexheti, 2025). This approach makes quality a key enabler for business - delivering predictability, resilience, and speed to innovation. Engineering excellence is, therefore, not only achieved by improving product testing, but by embedding systems that continually engineer quality throughout the product lifecycle.

6. Organizational Transformation and Global Collaboration

6.1 Organizational Redesign

A long-term quality improvement program must involve leadership and governance redesign, in which quality is taken from the shop floor to the boardroom. In conventional organizations, quality is typically managed by QA departments with limited impact on the "big picture". But in global product ecosystems, this approach is inadequate because quality issues impact the predictability of releases, customer satisfaction, readiness for market, and overall business sustainability. To ensure quality remains a critical business capability, executive leadership should reframe quality and make it central to strategy, resource planning and governance(Bolatan et al., 2022). This involves redesigning governance to embed quality measures into corporate objectives along with speed, growth and innovation. Executives must set up integrated accountability mechanisms in which engineering, product, design, operations and market activities are held accountable for quality. Governance redesign correspondingly entails the creation of quality policies, enterprise-wide product release criteria and the creation of multiple integrated decision points, which match the ambition with the capability of the organization. Institutionalizing quality at the board level delivers coherence to avoid quality ownership gaps.

6.2 Cultural Transformation

More than just the governance system, engineering excellence demands cultural change, which ensures quality accountability across all levels of the organization. Silos pose one of the greatest challenges to quality maturity because they promote narrow priorities and local problem-

AI and Machine Learning Advances
Volume 2, Issue 2
ISSN: 3067-3216

solving (Siddiqui, 2024). Cultural change involves shifting quality "spoken" and "unspoken" norms from role-specific to company-wide, through norms in which all roles contribute to defect prevention. Collaborative cultures are focused on shared responsibility rather than handoffs and finger-pointing, and prevention rather than correction. This is particularly important in the traditionally siloed product, engineering, QA and operational cultures. Overcoming this requires process redesign, but likewise reinforcing new behaviours through messaging, incentives and workflows. Teams must understand quality is not "someone else's job" but rather an operational mindset (El Manzani, 2026). Companies that effectively ingrain this character develop greater trust and responsiveness in resolving issues and delivering cross-functional work.

6.3 Global Product Team Alignment

Global product teams pose an extra challenge due to remote teams across time zones, markets and operating environments (Singh & Bhaskar, 2025). The distance between teams can introduce delays in communication, inconsistencies in governance and readiness if quality systems are not carefully harmonized. To support collaboration across the globe, quality systems need to be consistent despite geographic distance. Collaborative documentation, consistent CI/CD processes, asynchronous collaboration models and global quality dashboards all ensure team coordination regardless of where they are located. Market teams for each region should be brought into the process earlier to proactively deal with localization, regulatory and activation needs. Global team integration guarantees that quality is consistent and yet reflects market realities.

Table 2. Quality Ownership Maturity Model (Level 1–5)

Level	Maturity Stage	Characteristics
1	Reactive QA	Late-stage defect detection
2	Managed Testing	Structured QA processes
3	Shift-Left Integration	Early validation practices
4	Shared Ownership	Cross-functional accountability
5	Strategic Quality Ecosystem	Fully integrated global quality governance

7. Discussion and Practical Implications

7.1 Quality is a Strategic Asset

The lessons from this article help confirm an insight in today's approach to products: quality isn't just testing, it's a capability. Conventional views of approaches to quality limit it to "validation" activities like pre-release testing (De Klerk et al., 2025). While testing is important, this view understates the infrastructural role of quality in underpinning its ability to improve product delivery predictability and operational confidence. In the global product network of nowadays, it is more appropriate to conceptualize quality as the company, a governance, team, technology and culture model that drives the decision-making and the product development process (Bashan & Kordova, 2021; Malek et al., 2024). Strategically integrated quality enables requirements, design, manufacturing, implementation and time to market. These reframing shifts quality from a cost to a business investment that reduces waste, improves scalability, trust and innovation.

7.2 Comparative Analysis

The traditional QA Model and the Shared Quality Ownership Model differ in accountability and model. Traditional QA is based on a handoff approach involving design, QA, product, and marketing sign-off. This ensures quality is focused at the end of the cycle, resulting in reactive identification of quality defects and fixes. Ownership can be disparate, spurring fiefdom thinking and "throwing it over the wall". In the shared quality ownership model, quality is the responsibility of everyone. The product manager owns the requirement, the experience is owned by the designer, the governance quality engineer owns enablement, DevOps own resilience, and the market owns fit (Peters & Pallapa, 2025). This approach moves validation to quality integration. This results in a move from identifying defects to preventing defects. That is, rather than defect detection and correction by one team, systems and processes are designed to prevent defects. This contrast shows why collective ownership is a response to the complexity, speed and connectivity of global product development.

Table 3. Traditional QA vs Shared Quality Ownership Comparative Model

Dimension	Traditional QA Model	Shared Quality Ownership Model
Quality Responsibility	QA department	Cross-functional lifecycle ownership
Testing Timing	Late-stage validation	Continuous lifecycle integration
Defect Strategy	Detection-focused	Prevention-focused
Governance	Functional silos	Unified governance
Collaboration	Sequential handoffs	Continuous collaboration
Release Confidence	Variable/reactive	Predictable/proactive
Innovation Impact	Rework-heavy	Velocity-enhancing

7.3 Application for Global Teams

The model reinforces the need for engineering leaders to create technical systems that incorporate quality into the architecture, automation, and deployment rather than being relegated to testing (Buede & Miller, 2024). Product managers need to appreciate that "fuzzy" requirements pose quality risks and, as such, focus on clarity, accuracy and stakeholder alignment early in the development cycle. QE teams move from gatekeepers to quality enablers who own the automation framework, governance model and scalable quality validation ecosystem (Gupta, 2025). Market readiness teams must transition from lagging validators to early solution partners to ensure issues of localization, compliance and operational realities are considered early. Overall, these implications point to global teams being most effective when quality is driven across functions. Through the use of common quality systems, defect leakage can be reduced, release speed improved, and the right governance established for long-term engineering excellence.

8. Limitations

This study is conceptual, drawing primarily from synthesis of literature relating to Software Quality, DevOps, Organisational Behaviour and Engineering Governance not from empirical field validation. The Shared Quality Ownership Framework has not yet been applied in a way that involves longitudinal case studies, large-scale surveys or controlled implementation in a variety of organizational settings. The framework can thus marginalise the operation complexity of highly heterogeneous global product ecosystems, where industry regulations, team structures, and cultural differences may affect the implementation outcomes, and where the presence of organisations with varying degrees of maturity can be a determinant. Furthermore, the model suggests that the governance process be integrated throughout product functions, but does not quantify in terms of percentage reduction of defect reduction from each stakeholder group. The great multidisciplinary breadth can also make it difficult to quickly identify specific models to operate in an organization that needs them. Therefore, the ability to implement the strategy practically is likely to differ from enterprise to enterprise and depend on the technology architecture and leadership maturity to embrace cultural change.

9. Conclusion

This article has made the case that the key to engineering excellence in today's global interconnected product ecosystems is not agility, automation, or testing, but quality throughout the product lifecycle. In a rapidly evolving software environment increasingly marked by distributed teams, cloud-native infrastructure and continuous delivery, quality can no longer be simply QA-driven. Defects leak because quality is often seen as something that occurs after the fact, rather than a holistic, collaborative and integrated approach. Moving quality to a shared lifecycle capability alters the design, build, test and delivery practices of products.

The Shared Quality Ownership Framework makes both theoretical and practical contributions by conceptualizing quality as a hybrid technical, cultural and governance capability. The approach broadens the conversation about software engineering to include governance, leadership, cross-functional ownership and operational maturity in the context of quality. Companies that embed a shared sense of ownership across engineering, product, QE, DevOps and market teams have more stable, scalable and reliable production systems. In the end, quality not only does not belong to product, engineering, quality engineering, or DevOps - it is the engine for sustained engineering

AI and Machine Learning Advances
Volume 2, Issue 2
ISSN: 3067-3216

excellence.

10. Future Directions

Future research should empirically validate the Shared Quality Ownership Framework through cross-industry case studies, comparative organizational analyses, and quantitative modeling of defect leakage reduction across maturity stages. Longitudinal research studies in the pre and post governance transformation would enhance causal understanding of how the impact of shared ownership on release stability, innovation velocity, and engineering excellence measures (DORA, MTTR) would be. Industry-specific adaptations, like in SaaS, fintech, healthcare, and regulated enterprise systems, could also be further explored by additional research, as compliance needs could mean a different type of ownership model. Another exciting path forward for a transformation into Quality 4.0 ecosystems is the combination of AI-powered quality intelligence, predictive defect analytics, and automated governance dashboards. Further, future studies of leadership behaviors, rewards, and models for cross-cultural collaboration that best promote shared quality ownership in globally dispersed teams should be explored through research and scholarship.

Declarations

Conflict of Interest

The author declares no conflict of interest

Acknowledgment

None

Funding

No funding was received.

Data Availability

All data has been provided within this manuscript.

Ethical Approval

Not Applicable

Consent for Publication

Not Applicable

Reference

Abieba, O. A., Ubamadu, B. C., Hassan, Y. G., Owobu, W. O., Daraojimba, A. I., & Gbenle, P. (2025). Advanced Observability and Monitoring Practices for Enhancing Production Environment Reliability.

Alami, A., & Ernst, N. (2024). Understanding the building blocks of accountability in software engineering. Proceedings of the 2024 IEEE/ACM 17th International Conference on Cooperative and Human Aspects of Software Engineering,

Alphonce, M. (2024). Enhancing software quality through early-phase of software verification and validation techniques. Available at SSRN 4611404.

Ayyash, M. A. I. A. (2024). Implementing Agile and DevOps at Scale: Identifying Best Frameworks, Practices, and Success Factors Al-Quds University].

Bankins, S., Ocampo, A. C., Marrone, M., Restubog, S. L. D., & Woo, S. E. (2024). A multilevel review of artificial intelligence in organizations: Implications for organizational behavior research and practice. Journal of organizational behavior, 45(2), 159-182.

Bashan, A., & Kordova, S. (2021). Globalization, quality and systems thinking: integrating global quality Management and a systems view. Heliyon, 7(2).

Bexheti, L. A. (2025). Shifting from Quality Control to Quality Enhancement: Insights from QA-SURE Project Partner Institutions. International Scientific Conference on Business and Economics,

Bittla, S. R. (2025). The Future of QE with AI-Driven Testing. In AI-Driven Software Testing: Transforming Software Testing with Artificial Intelligence and Machine Learning (pp. 455-472). Springer.

Bolatan, G. I. S., Golgeci, I., Arslan, A., Tatoglu, E., Zaim, S., & Gozlu, S. (2022). Unlocking the relationships between strategic planning, leadership and technology transfer competence: the mediating role of strategic quality management. Journal of Knowledge Management, 26(11), 89-113.

Buede, D. M., & Miller, W. D. (2024). The

engineering design of systems: models and methods. John Wiley & Sons.

Daraojimba, A. I., Kisina, D., Adanigbo, O. S., Ubanadu, B. C., Ochuba, N. A., & Gbenle, T. P. (2024). Systematic review of key performance metrics in modern devops and software reliability engineering. International Journal of Future Engineering Innovations, 1(1), 101-107.

De Klerk, C., Ker-Fox, J., & Steenekamp, L. (2025). Enhancing critical thinking through collaborative learning: The impact of a partial pre-release assessment format. Accounting Education, 34(4), 499-532.

El Manzani, Y. (2026). Quality mindset: applying implicit theories to quality management. International Journal of Quality & Reliability Management, 43(3), 784-815.

Esther, D. (2024). Automated Testing Strategies for Quality Assurance in CI/CD Pipelines. In: URL:(PDF) Automated Testing Strategies for Quality Assurance in CI/CD Pipelines.

Ferguson, S., Lai, K., Chen, J., Faidi, S., Leonardo, K., & Olechowski, A. (2022). "Why couldn't we do this more often?": exploring the feasibility of virtual and distributed work in product design engineering. Research in engineering design, 33(4), 413-436.

Gupta, E. (2025). ENABLING ANALYTICS GOVERNANCE IN AGILE PRODUCT TEAMS: A SCALABLE TAGGING AND QA FRAMEWORK. International Journal of Applied Mathematics, 38(7s), 1161-1172.

Harve, B. M., Bidkar, D. M., Krishnappa, M. S., Pandy, G., Jayaram, V., Veerapaneni, P. K., & Mehta, G. (2024). The cloud-native revolution: Microservices in a cloud-driven world. 2024 International Conference on Intelligent Cybernetics Technology & Applications (ICICyTA),

Heijstek, A. (2023). Bridging theory and practice: insights into practical implementations of security practices in secure devops and ci/cd environments Ph. D. thesis, Universiteit van Amsterdam].

Hirsi, A. (2025). Integration of DevSecOps practicesin the CI/CD Pipelines. In.

Horshchar, K. (2025). Implementation of the Shift Left paradigm: impact on software quality and testing efficiency. Global Prosperity, 5(3).

Ilufoye, H., Akinrinoye, O. V., & Okolo, C. H. (2023). A Global Reseller Ecosystem Design Model for Software-as-a-Service Expansion. International Journal of Multidisciplinary Research and Growth Evaluation, 3(6), 107-113.

Joseph, S. A., Kolade, T. M., Val, O. O., Adebiyi, O. O., Ogungbemi, O. S., & Olaniyi, O. O. (2024). AI-powered information governance: Balancing automation and human oversight for optimal organization productivity. Asian Journal of Research in Computer Science, 17(10), 110-131.

Kader, M. A., & Ahmed, R. (2025). A Comprehensive Review of Recent Advances in Program Analysis Techniques for Software Reliability and Security. Scientia. Technology, Science and Society, 2(10), 72-79.

Kavuri, S. (2024). Shift-Left and Shift-Right Testing Approaches: A Practical Roadmap for Continuous Quality in Agile and DevOps. Journal of Information Systems Engineering and Management, 9(4), 1-10.

Lapatoura, C. (2025). The Role of Leadership in Shaping Ethical Culture and Practices of Excellence: A System Thinking Approach. In Leadership Studies in the Turbulent Business Ecosystem. IntechOpen.

Li, W., & Hazelrigg, G. (2023). The potential for incentive structures to prevent significant engineering failures. International Design Engineering Technical Conferences and Computers and Information in Engineering Conference,

Mahmood, K. (2022). Manufacturing problem solving methodology analytic. A case study of leakage current in a production company.

Makanda, I. L. D., Jiang, P., Yang, M., & Shi, H. (2023). Emergence of collective intelligence in industrial cyber-physical-social systems for collaborative task

AI and Machine Learning Advances
Volume 2, Issue 2
ISSN: 3067-3216

allocation and defect detection. Computers in Industry, 152, 104006.

Malek, R., Yang, Q., & Dhelim, S. (2024). Toward sustainable global product development performance: Exploring the criticality of organizational factors and the moderating influence of global innovation culture. Sustainability, 16(10), 3911.

Mulla, M. (2023). Strategic Roles Of Product Managers In Agile Teams: A Key To Iterative Product Development. International Journal of Business Quantitative Economics and Applied Management Research, 7(8), 46-57.

Ogunmolu, A. M., Aroh, I. S., Onyii, H., Adeyinka, P. D., & Olutimehin, A. T. (2026). AI-Driven Observability for Managing Security Complexity in Cloud-native Microservices and Containerized Environments. Journal of Engineering Research and Reports, 28(2), 1-17.

Otieno, M., Odera, D., & Ounza, J. E. (2023). Theory and practice in secure software development lifecycle: A comprehensive survey. World Journal of Advanced Research and Reviews, 18(3), 053-078.

Patel, J. S. (2026). The Evolution of Shift-Left Testing in Modern Software Development. Journal of Computational Analysis & Applications, 35(1).

Perla, S. A Framework for AI-Powered Test Automation: Redefining QA Efficiency and Coverage.

Peters, M., & Pallapa, G. (2025). Mastering Enterprise Platform Engineering: A Practical Guide to Platform Engineering and Generative AI for High-Performance Software Delivery. Packt Publishing Ltd.

Potla, R. B. (2021). Reducing Test Cycle Time by 70% in ERP Transformations: A DevTestOps Blueprint for Manufacturing Enterprises. J Artif Intell Mach Learn & Data Sci, 4(1), 3242-3249.

Raj, P., Vanga, S., & Chaudhary, A. (2022). Cloud-Native Computing: How to design, develop, and secure microservices and event-driven applications. John Wiley & Sons.

Rani, V. S., Babu, A. R., Deepthi, K., & Reddy, V. R. (2023). Shift-left testing in devops: A study of benefits, challenges, and best practices. 2023 2nd International Conference on Automation, Computing and Renewable Systems (ICACRS),

Rechkemmer, D., Korth, M., May, M. C., & Lanza, G. (2025). Development of a concept for the design of user-friendly simulation models. Procedia CIRP, 132, 110-115.

Saarinen, M. (2024). Evaluation and implementation of Site Reliability Engineering (SRE) practices in an organization: an action research study M. Saarinen].

Saheb, T., & Saheb, T. (2024). Mapping ethical artificial intelligence policy landscape: A mixed method analysis. Science and engineering ethics, 30(2), 9.

Sannapureddy, R., Nelavelli, S., & Kovvuri, V. K. R. (2021). Optimizing Continuous Integration and Continuous Deployment (CI/CD) Pipelines: Strategies, Tools, and Performance Metrics. International Journal of AI, BigData, Computational and Management Studies, 2(4), 117-129.

Siddiqui, A. (2024). Breaking Down Silos: Strategies for Effective Multidisciplinary Collaboration. Kashf Journal of Multidisciplinary Research, 1(08), 411-420.

Singh, A., & Bhaskar, A. K. (2025). Global teams, local challenges: Cultural diversity in remote work. In Organizational Sociology in the Digital Age (pp. 63-86). IGI Global Scientific Publishing.

Tang, H. (2021). Quality planning and assurance: Principles, approaches, and methods for product and service development. John Wiley & Sons.

Thason, J. R. M., & Singh, R. K. Measuring Devops Success With The DORA Metrics: A Comprehensive Analysis Of Key Performance Indicators And Their Impact On Software Delivery.

Thota, M. K., Shajin, F. H., & Rajesh, P. (2020). Survey on software defect prediction techniques.

International Journal of Applied Science and Engineering, 17(4), 331-344.

Ugwueze, V. U., & Chukwunweike, J. N. (2024). Continuous integration and deployment strategies for streamlined DevOps in software engineering and application delivery. Int J Comput Appl Technol Res, 14(1), 1-24.

Zakeri, M., Abdi, F., & Bagheri, F. (2025). Enhancing Software Quality Attributes Through Multi-Dimensional Refactoring at Source-Level. Science of Computer Programming, 103434.

Zonnenshain, A., & Kenett, R. S. (2020). Quality 4.0—the challenging future of quality engineering. Quality engineering, 32(4), 614-626.