Cureus Journal of Computer Science

Open Access Original Article

Driving Innovation Through Change: Release Management With a Focus on Nonfunctional Requirements and Emerging Technologies

Received 02/21/2025
Review began 04/18/2025
Review ended 06/30/2025
Published 07/10/2025

DOI: https://doi.org/10.7759/s44389-025-03354-5

Bishnu Shankar Satapathy [1] , Ashish Sharma [2] , Madhusmita Dash [3] , Siddhartha Sankar Satapathy [4] , S. Ibotombi Singh [4] , Joya Chakraborty [5]

1. Center for Multidisciplinary Research, Tezpur University, Tezpur, IND 2. Department of International Business, Birla Institute of Management and Technology, Noida, IND 3. Department of Electronics and Communication Engineering, National Institute of Technology, Jote, IND 4. Department of Computer Science and Engineering, Tezpur University, Tezpur, IND 5. Department of Mass Communication and Journalism, Tezpur University, Tezpur, IND

Corresponding authors: Bishnu Shankar Satapathy, bishnu_mdp21@tezu.ernet.in, Ashish Sharma, ashish.sharma.ei@gmail.com, Madhusmita Dash, madhusmita.dash81@gmail.com, Siddhartha Sankar Satapathy, ssankar@tezu.ernet.in, S. Ibotombi Singh, sis@tezu.ernet.in, Joya Chakraborty, jc@tezu.ernet.in

Abstract

The release management process is essential for solution providers and customers. It encompasses strategic planning, regression testing, exact deployment, and continuous monitoring of software updates. Rapid improvements in artificial intelligence/machine learning technologies and cloud-based software solutions have created unique challenges with specific reference to overseeing the nonfunctional requirements (NFRs) in the release management processes designed for the software industry. Considering the significance of release and organizational change management in digital transformation, specifically in moving from traditional on-premises solutions to enterprise cloud-based SaaS solutions, we have presented a software release management workflow by incorporating a change management module based on the responses received from software solutions and project stakeholders. The preliminary results on addressing the necessary adaptations in product-based software organizations when overseeing the NFRs over a limited period show that the revised workflow potentially improves the management of slippage by enhanced efficiency in addressing NFR-related issues. The outcome endorses a more conventional release management process and reliable release cycles to navigate change in the rapidly evolving ecosystem of the release management process.

Categories: Computational Science and Engineering, Software Development, Software Project Management
Keywords: release management, process management, change management, non-functional requirements (nfr), software as service

Introduction

Software release management is crucial for delivering software solutions systematically and ensuring quality. It involves planning, scheduling, and controlling updates across on-premises and cloud environments. This strategy is instrumental in preserving the robustness of the solution post-deployment to the customer. It has a broad plan guiding all stakeholders through the software development life cycle stages, ensuring a cohesive and predictable release process, and satisfying nonfunctional requirement (NFR) quality gateway parameters. The traditional release management workflow starts with design, where requirements are gathered and system architecture is planned. It is followed by coding, where engineers develop features and fix bugs. Before deployment, it moves through various release gateways, including quality, security, and stakeholder approval (Figure *1*). However, this approach often faces limitations like inflexible processes that can cause delays and misalignment with changing requirements, as well as bottlenecks that increase the risk of errors during handling a critical case [1].

The software industry faces significant challenges due to noncompliance with release gateways, which often impacts both the development and release cycles. During the release cycle, many issues are reported towards end of the cycle, delaying releases and putting others on hold as teams lack decision-making authority. With the rise of cloud-native architectures, microservices, and DevOps practices, the complexity of software systems has notably increased. Existing research identifies significant gaps in managing these complexities, particularly in dependency management, process automation, and communication across distributed teams. When an on-premises solution transitions to a native Software-as-a-Service (SaaS) solution, it involves complexities like customer focus, observability, and clear release roadmaps [2]. Study says a firm release management strategy in a cloud environment is essential to ensure NFRs of the solutions [3]. In addition, the associated release gateways ensure that the release of a solution complies with DevOps quality processes [4]. The rise of artificial intelligence (AI) and machine learning (ML) technologies adds complexity, necessitating a well-planned release management approach. Complexity includes dependency management, process automation, and communication across teams. These technological advancements offer flexibility and scalability, introducing coordination, dependency

management, and software quality challenges.

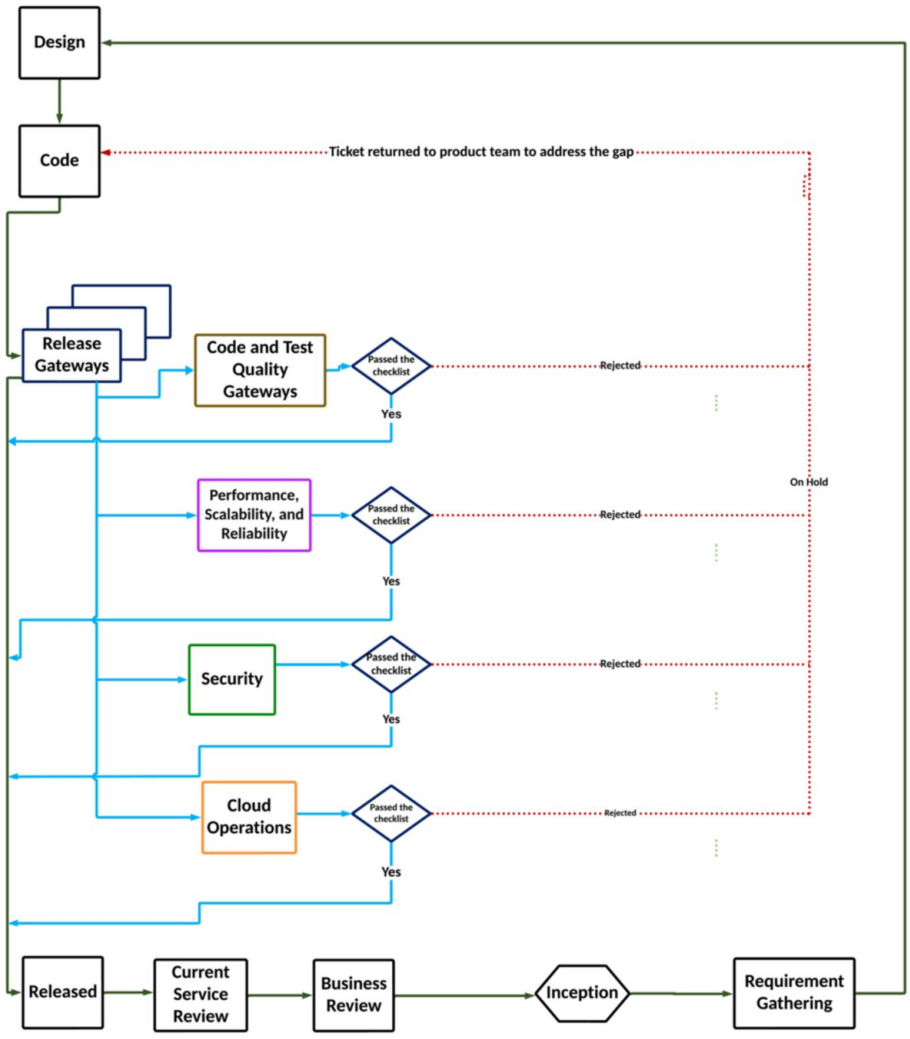

FIGURE 1: The figure illustrates the various gateways in the traditional software solutions before the release, without the change control board and with release gateway keepers in place. The golden box and text represent the code and test release gateway. The purple box and text denote the performance release gateway. The yellow box and text indicate the cloud operations release gateway. Finally, the security release gateway is detailed in the green box.

As organizations increasingly adopt Agile and DevOps methodologies, the need for more streamlined and adaptive change management processes within the traditional release management workflow has become apparent, prompting a reassessment of conventional release management methodologies [5]. Effective release management minimizes risks and disruptions, maintains customer satisfaction, plays a crucial role in the overall DevOps process by bridging the gap between development and operations, and applies Agile and Lean principles to improve delivery speed [6]. DevOps aims to bridge this gap by extending Agile and Lean principles to operations teams, thereby improving the delivery of software products and services [7,8]. However, rapid and frequent changes can lead to communication breakdowns, delays, and operational inefficiencies, ultimately impacting software stability and user satisfaction.

The manuscript investigates the empirical challenges of release management in adopting emerging technologies. The research aims to examine and improve release management practices by identifying and

addressing weaknesses in software delivery systems. This article focuses on the underexplored areas of release and change management, particularly as organizations migrate from on-premises to cloud platforms. It highlights release management and the industrial revolution as critical factors for successful software delivery. The study explores how the release management workflow and organizational change management (OCM) are essential in digital transformation, especially during the transition to cloud-based SaaS solutions. Through research and interviews, this article presents a workflow designed to ensure software release readiness.

The following research questions will guide the investigation:

· RQ1: What are the major concerns of stakeholders when delivering a software solution?

· RQ2: How can the release management process adapt to new requirements?

· RQ3: How does change management help align release management with changing technical needs and business goals?

· RQ4: How can change management help manage NFR slippage issues in the release management process?

Materials And Methods

Our research method consists of five phases (Figure *2*). In phases 1 and 2, we focused on preparing interview questions, planning the case study, and selecting stakeholders. Phase 3 involved pilot testing and gathering stakeholder input. In phase 4, real-time data was collected, and the findings were recorded. Finally, phase 5 centered on the evaluation and synthesis of the collected data. All research activities were conducted in accordance with relevant guidelines and regulations.

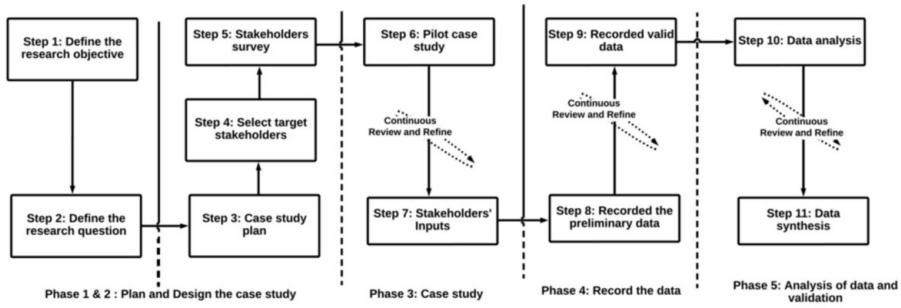

FIGURE 2: Figure illustrates the overview of the research processes.

Questionnaires used in surveying the stakeholders

To better understand the challenges stakeholders face in the software industry, we interacted with the stakeholders involved in various phases of the software development life cycle, from requirement analysis to maintenance. The aim was to understand the technology partners' pain points in full-stack software development. As part of the study, the stakeholders' interactions were designed to collect inputs for release planning and coordination issues, user training and support, system performance, security measures, feedback mechanisms, documentation, and continuous improvement. One hundred eighteen skilled stakeholders from various SaaS project teams from different countries, including India, the USA, Singapore, and Australia, responded, gathering pertinent information covering simple to complex multi-modal solutions. The detailed survey questionnaires are provided in the Google Forms and can be accessed using the https://forms.gle/sSeH5cTea6LEN5p1A and https://forms.gle/BaHcLiuBWCLL856n8 URLs. This interaction was strictly for academic research, and all data were treated with confidentiality. The stakeholders affirm no conflict of interest with the respondents or any other involved party, ensuring ethical data collection. The primary objective of this survey is to gain a comprehensive understanding of the software delivery strategies currently employed by product-based software organizations.

Software Organization Types and Select Participants

Thirty-four skilled stakeholders from various SaaS project teams from different countries, including India, USA, Singapore, and Australia, were interviewed for this study, gathering pertinent information covering simple to complex multi-modal solutions (Table *1*). We have collected data from three software organizations (company names are not disclosed due to a non-disclosure agreement), where we had access

to practitioners, products, testers, test cases, and processes to study the release management process. To ensure the participants can provide data on the release management process and NFRs, we defined specific sampling criteria, such as stakeholders' project work experience and role in the project, as shown in Table *1*, to increase the sample's representativeness. The working experience includes domain knowledge in both RM requirements and understanding the importance of the NFRs. At the time of this study, most selected participants had more than 8 years of work experience, which could indicate the participants' ability to answer our interview questions.

Participants	Participant Role	Industry Experience (Years)	Software Assets
L2	Engineer	3-6	Solutions A, B, C, D, E
L3	Quality Engineer	5-6	
L4	Cloud DevOps Engineer	5-8	
L5	Lead Engineer	8-10	
RM	Release Manager	10-12	
M1	Project Manager	10-15	
L5	Software Architect	8-12	
M2	Product Owner	15	
M5	Distinguished Engineer	20-25	
P4	Functional Architect	8-10	
M3	Delivery Manager	12-15	
M4	Change Management Professional	15-20	

TABLE 1: Participant's role, industry experience, and software assets.

Note: L1-L5: Stakeholders are the primary contributors actively involved in project delivery and have minimal decision-making power. M1-M5: Stakeholders are the ones who have decision-making power. This study collected data from three software organizations (company names are not disclosed due to a non-disclosure agreement), where we had access to practitioners, products, testers, test cases, and processes to study the release management process.

The project type includes various public and private cloud offerings, hybrid cloud setup, on-premises solutions migrating to the cloud platform, and various software management models for application development, support, and maintenance (Table *2*). In addition, we have gathered data points on the challenges the organization faces. Based on the input received, the details of how the business units' function have been summarized in the later sections. Software industry's process in the direction of SaaSification of their solution has also highlighted.

Data Collection

We have collaborated with the target stakeholders through personal communication and collected information with a non-disclosure agreement to ensure their privacy. We tried to uphold the study's internal validity in line with the case study guidelines established by Runeson and Höst [9]. The interview questions undergo rigorous assessment and validation to enhance the quality and address potential concerns. Industrial practitioners and researchers thoroughly reviewed and evaluated the interview questions to ensure precision. Additionally, we verified the study results by sharing them with practitioners.

After receiving the first survey inputs, we have identified key delivery stakeholders from the identified organization to understand more about the release management process and how the change management process works in the respective organization. To further interact with the key contributions, we have designed the following interview questionnaires (IQs):

• IQ1: What challenges are you facing during the release management process?

IQ1 aims to find the pain points during the release management process. These could include issues such as frequent delays in release, difficulty in coordinating with different teams, or challenges in keeping the quality of the released software.

• IQ2: What are the NFRs given the highest priority?

IQ2 expands on IQ1 and aims to find NFRs given the highest priority. These requirements, including security and quality processes, metrics, and tools, are vital for NFR testing. By adopting this higher-level perspective, we examine implied relationships between NFR metrics and the release management process.

• IQ3: How is the Release Readiness Workflow aligned to confirm the NFR slippage?

IQ3 extends the scope of IQ2 by explicitly focusing on how the reported NFR components are managed during software delivery. This includes understanding and managing any instances of NFR slippage, which refers to the situation where the actual performance of the software does not meet the defined NFRs.

• IQ4: What challenges are associated with handling NFR tracking and the role of the change management process?

IQ4 aims to find the challenges associated with managing the reported NFRs and how they are managed during the release process using the change management process.

NFRs slippage data collection

Deviations or delays in meeting NFRs, called NFR slippage, can pose significant challenges to project delivery and overall quality, impacting project outcomes and stakeholder satisfaction. We followed a structured approach to analyze NFR slippage to facilitate root cause analysis, prioritization, mitigation, communication, and iterative improvement of NFR slippages [10]. For our study, we have considered three mid-range software product-based organization (Table *2*: Solution D) that offers SaaS-based solutions in various regions. The team working on the solution is composed of 30 individuals. Aspects related to types of NFRs (Table *3*) are the primary focus of our analysis. During the software release process, the NFRs such as performance, quality, security, scalability, usability, and reliability are scanned using various tools. All released artifacts were scanned for any security issues remaining in the product. Further, tools like JMeter and LoadRunner were used for performance and load testing, ensuring the software could manage expected traffic. SonarQube and OWASP ZAP were used to scan code quality and security vulnerabilities. New Relic and Dyna-trace provide insights into scalability and reliability by monitoring real-time performance.

Components	Elements	Solution A	Solution B	Solution C	Solution D	Solution E
Project	Solution Domain	Supply Chain Retail	Supply Planning	Retail Finance Technology	NextGen AI	Logistics
	Project Team Size	Medium (51-100)	Medium (51-100)	Small (21-50)	Small (21-50)	Medium (51-100)
	Product Type	Retail Planning Services	Supply Planning and Forecasting	Retail Financial Planning Solutions	Supply Planning and Forecasting	Transport Optimization
	Maturity of Product	Matured Solution	Matured Solution	Matured Solution	Nextgen Cloud Solution	Matured Heritage Product
	Agile Adoption	Yes (global)	Yes (global)	Partially	No (startup culture)	Yes (global)
	Teams Involved	Development & test teams from distributed location. Decision makers are from different regions	Development & test teams from distributed location	Development teams co-located and managers from different geographic regions	Development & testing teams co-located	Development & test teams from different geographic regions
	Distributed Development	Yes	Yes	No	No	Yes
Market	Customer Base	Global Organizations	Global Organizations	Global Organizations	Global Organizations	Global Organizations
Development Process		Development plan is driven by scaled agile framework-based continuous development	Combination of agile and Kanban for incremental development	Combination of agile and Kanban for incremental development	Incremental development	Development plan is driven by scaled agile framework-based continuous development
Release Readiness Strategy	Maturity Level	Traditional and Matured Workflow	Traditional and Matured Workflow	Traditional and Matured Workflow	Fast-Paced and Dynamic Process	Traditional and Matured Workflow

TABLE 2: Software assets details considered for the study.

To understand the impact of change management's role in release management on the exception slippage, we collected inputs over a limited period from the first quarter of 2023 to the second quarter of 2024. In the projects, the new release management workflow with clear benchmarking of the release gateways and the change management process were introduced in the fourth quarter of 2023. The stakeholders involved during the release work towards fixing the reported issues in each release. When the activities of the major release are completed, the major release code base is branched out from the main. The team uses the branch code base to work towards the minor releases. During this period, the development team fixed the carry forward NFRs from the major release and the newly reported issues. Since the minor releases are more focused on fixing the bugs and technical stacks are not changed, the reported issues have been observed to be much fewer than the major releases [11].

NFR Category	Key Attributes	Description	Sourced Data Type
Observability	Testability, Changeability, Modifiability	Measures how well internal states of a system can be inferred from its outputs, how easily changes can be made, and how test-friendly the system is.	Internal data collected from testing environments, including logs, code change frequency, and automated test reports.
Security	Vulnerability, Confidentiality, Authentication, Access Control	Evaluates system robustness against threats and unauthorized access, ensuring data privacy and secure access mechanisms.	Internal vulnerability scans, authentication logs, and access control reports from the test environment.
Performance	Response Time, Accuracy	Assesses how quickly the system responds to requests and how accurately it processes data under varying loads.	Performance metrics, latency data, and test results from the staging/test environment.
Monitoring	Performance Tracking, Availability, Behavioral Insights	Focuses on real-time system health, service uptime, and tracking unexpected behaviors during operation.	Logs, dashboards, and monitoring tool outputs (e.g., Prometheus, Grafana) from the test environment.
Scalability	High Availability	Measures the system's ability to manage growth of more users, more data, or higher transaction volumes without performance degradation.	Assess environment data reflecting system behavior under simulated high-load conditions or failover testing.

TABLE 3: Nonfunctional requirement types considered for slippage analysis.

NFR, Nonfunctional Requirement

Data analysis

We documented all relevant NFRs for the software development project in consultation with the stakeholders, including end-users, business analysts, and domain experts, to ensure comprehensive coverage of NFRs. Once the NFRs were identified, baseline targets or criteria were established for each NFR. These baselines served as reference points against which the system's actual performance was to be measured. Any changes or updates to the NFRs are carefully tracked and documented. Deviation analysis involved comparing the system's actual performance against the established NFR baselines, considering quantitative metrics (e.g., response time, throughput) and qualitative assessments (e.g., user satisfaction, system usability). In the case of NFR slippage, a root cause analysis was conducted to find the underlying factors contributing to the deviations. For statistical analysis, we calculated the mean (μ) and standard deviation (σ) of the following formulae:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} a_i \qquad (1)$$

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2} \qquad (2)$$

where:

x_i = data points (e.g., number of NFR issues reported, NFR issues remain unresolved)

n = total number of data points (e.g., number of NFR issues reported, NFR issues remain unresolved)

Additionally, we conducted a hypothesis test using the null hypothesis ($H0$) and alternative hypothesis ($H1$) as part of the statistical analysis and calculated p-values using the Mann-Whitney U test.

Microsoft Excel analyzes the collected information and identifies patterns and trends. In addition, using the Lucid chart software (https://www.lucidchart.com), we designed workflow figures.

Results

This section outlines the key findings derived from stakeholder interviews, process observations, and

system data analysis. The results delve into the primary concerns expressed by technology partners involved in delivering SaaS solutions, with a particular emphasis on how release management processes adapt to evolving requirements. Additionally, it underscores the vital role of change management in ensuring that technical changes align with business objectives. Special focus is given to the handling of NFRs, exploring the methods for identifying, managing, and mitigating potential slippage within the release workflow. These insights are supported by an experimental setup aimed at testing the effectiveness of change management strategies in addressing NFR slippage.

Primary concerns of technology partners' stakeholders regarding SaaS solution delivery

Inadequate software-release administration can create unnecessary pressure to address any arising risks quickly. However, introducing a change control board (CCB) into the release management process can mitigate these risks and prevent unauthorized changes that may cause disruptions. The release management team oversees coordinating project teams, and without proper CCB oversight, errors can occur, and system stability may be compromised, resulting in dissatisfied customers. To better understand the challenges faced by stakeholders in the software industry, we conducted a questionnaire-based survey with those involved in various phases of the software development life cycle, from requirement analysis to maintenance.

Inadequate software release administration can create unnecessary pressure to address any arising risks quickly. To better understand stakeholders' challenges in the software industry, we conducted a questionnaire-based survey with those involved in various software development life cycle phases, from requirement analysis to maintenance.

We did a survey among our technology partners between November 2023 and March 2024 and collected information on the pain points. We interviewed skilled stakeholders from various SaaS projects, and the information we gathered covered everything from simple solutions to complex, multi-modal ones. Responses from the technology partners about SaaS solution delivery are summarized in Figure 3. Our survey results discovered that stakeholders face challenges during development due to the need to adhere to release and change management policies. Lack of clarity in on-premises to cloud platform migration, communication gap between stakeholders and development team, misunderstanding in requirements, and lack of training materials on project requirements were among the significant concerns raised in the survey. Respondents emphasized the importance of greater transparency from higher management, particularly about release management and change management strategies in adopting modern technologies due to which the integration of change management and release management in software development has garnered significant attention in recent years [12]. In recent times, researchers and practitioners pursue to enhance the efficiency and effectiveness of software development processes [13,14].

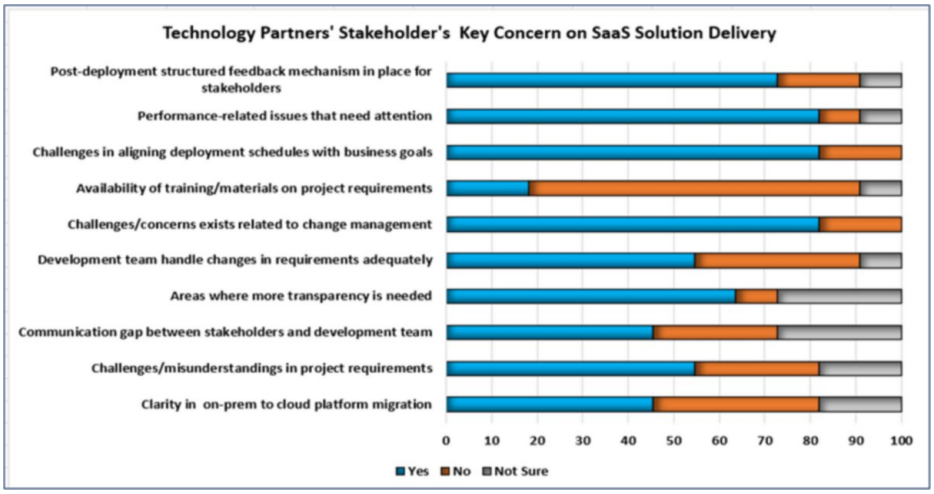

FIGURE 3: Figure depicts the technology partners' stakeholder's key concern on SaaS solution delivery. The length of the horizontal bars represents the proportion of responses aligning with the legends specified in the chart concerning the survey questionnaire.

Release management process accommodates the change aligning

with the new requirements

As technology and consumer trends increasingly support continuous releases, real-world scenarios often necessitate more novel solutions. Even stable systems may require minor adjustments, and significant risks can arise when a new team assumes responsibility without a comprehensive understanding of prior work. Centralized management is crucial in maintaining transparency and control in solution delivery, instilling a sense of security within the development team. It needs a robust framework that addresses all non-production requirements before a release. In addition, quality management must align with industry standards to ensure high release readiness. All cloud-deployed solutions are required to go through the architecture review and approval process. It ensures that the solution adheres to an automated deployment pipeline encompassing infrastructure, application code, and service configuration. To maintain the solution quality, a close collaboration among engineering teams, SaaS stakeholders, and architecture groups is essential for alignment with global cloud standard. An observability framework is critical, enabling the team to swiftly identify and resolve issues in new SaaS products while prioritizing monitoring and telemetry as key features.

Effective release management leverages AI, IoT, and collaborative robots to streamline continuous integration and deployment, reducing risks. While automating the DevOps cycle alleviates tedious tasks, the team's commitment remains essential for success [15]. The pandemic has prompted technology partners and customers to embrace more flexible and sustainable solutions. In software release management, well-defined release gates are vital for ensuring code readiness and deployment reliability before customer delivery. In the software release management process, release gates play a crucial role in ensuring that the software meets all key requirements before it is delivered to the customers by the technology partners. These gates need to be well defined to enforce checkpoints that ensure code readiness and deployment reliability.

• Code and test quality: A comprehensive code quality gate establishes systematic code reviews, automated testing, and continuous integration. Setting thresholds for test coverage and conducting static code analysis, these gates help identify errors early and reduce technical debt.

• Performance and scalability: Performance gates ensure the system can efficiently manage increased loads and large data volumes after deployment. Load, stress, and scalability testing are essential before production to prevent bottlenecks and meet user expectations.

• Reliability: Reliability gates involve running regression tests, high-availability checks, and failover scenarios to ensure continuous operation. These measures minimize downtime risks through strategies such as chaos engineering and disaster recovery drills.

• Security: Security gates are crucial for compliance and vulnerability management. They involve regular audits, penetration testing, and code scanning to address potential threats before deployment.

• Cloud operations: Gates for cloud operations ensure automated deployment pipelines and proper infrastructure implementation as code for cloud-native applications. They validate integration with cloud services and enhance operational flexibility.

• User experience: User experience gates require passing usability and user acceptance tests to ensure intuitive design and effective user interaction. This prevents poor experiences that could lead to user dissatisfaction.

Keeping the above scenario in view and to mitigate the technology partners' pain points for the stakeholders, we have updated the workflow described in the introduction to ensure better quality delivery (Figure *4*). The workflow is designed to provide end-to-end visibility to all the project stakeholders involved in the project. In addition, when any bottleneck is created, the decision-making authority plays the change management role to ensure seamless project delivery.

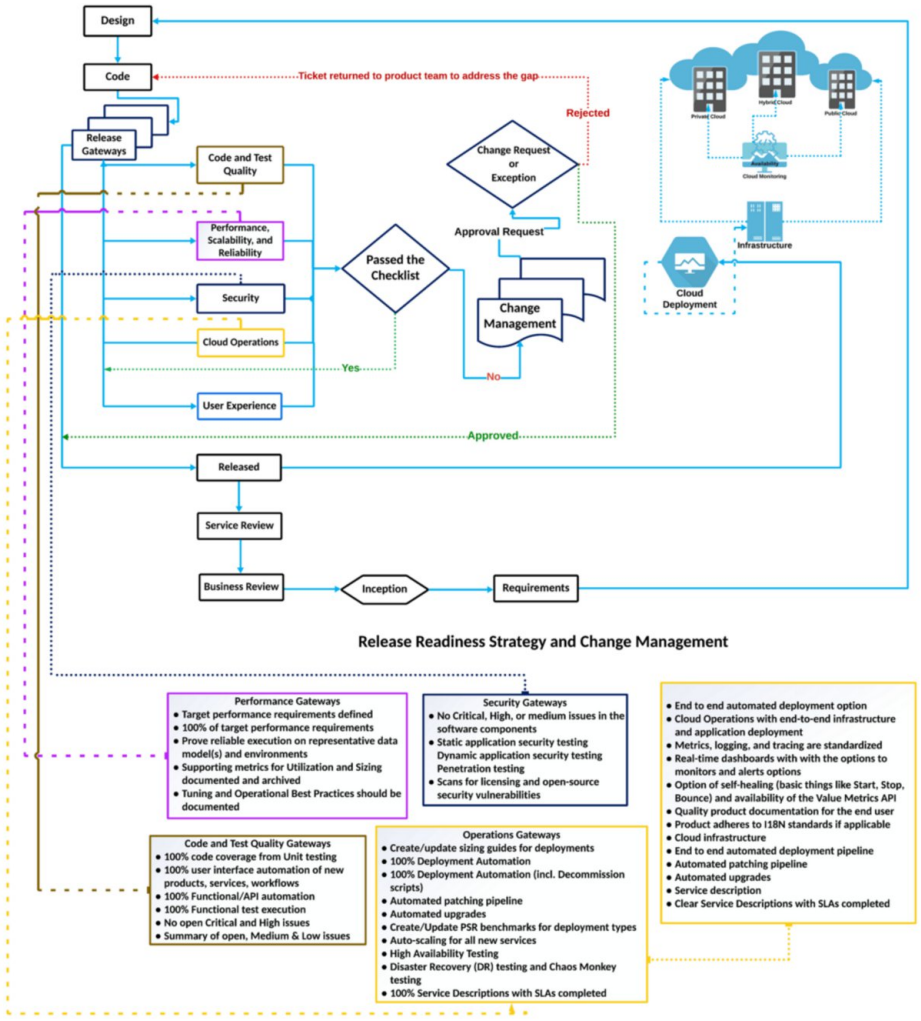

FIGURE 4: Illustrates the various gateways of the software solution before its release, along with change management and additional gatekeepers. The golden box and text represent the code and test release gateway. The purple box and text denote the performance release gateway. The yellow box and text indicate the cloud operations release gateway. Finally, the security release gateway is detailed in the deep blue box and text.

In this proposed workflow (Figure *4*), the software will undergo comprehensive release gateways to assess its readiness for deployment. The change management processes, designed to maintain accountability for any modifications, will provide security and confidence. In this new release management workflow, each release gateway has introduced the CCB role to evaluate the showstoppers. The CCB's participation is critical in prioritizing changes based on objectives and ensuring transparency and consistency in the process. With their supervision, compliance with regulations may be protected, and organizations may avoid facing penalties. The CCB can validate requests for release that are needed to clear the gates.

All requests submitted for review must include essential information for consideration by the CCB. This consists of the rationale for the change (e.g., a bug fix or a new feature), the business case or potential impact (e.g., increased efficiency or potential downtime), downstream effects (e.g., impact on other systems or processes), the platform involved (e.g., web application or database), and the product domain associated with the release (e.g., finance or customer service). During CCB meetings, assessments are conducted to evaluate the impact of changes on performance, security, and scalability. Mandatory evaluations and signoffs for NFRs occur during the release planning phase. The CCB uses critical parameters like the reason for the change, business case or impact, downstream impact, and the platform and domain of the product concerned to make its determinations. A standardized Change Request template is incorporated, enabling teams to document and track NFR implications from the

outset. The CCB will only evaluate requests that provide all required information. Unauthorized changes can introduce significant risks and disruptions. Failing to involve the CCB in the RM process can lead to such changes, impacting system stability and customer satisfaction. Upon successfully navigating these gateways, a business review will be conducted to ensure alignment with strategic objectives and stakeholder expectations. Lastly, a comprehensive post-release compliance review of NFRs confirms that solutions meet established quality standards and identify opportunities for continuous improvement, providing a thorough reassurance of the system's performance.

Role played by change management in aligning release management processes with evolving technical requirements and business goals

The role of release management is to ensure efficient software updates. This includes addressing stakeholder participation to avoid deployment disruptions and providing timely updates that are aligned with release schedules. However, changes mid-cycle may introduce disruption risks due to unforeseen deployment issues.

Change management process can enhance stakeholder engagement and manage interdependencies, especially when transitioning from on-premises to SaaS solutions. By integrating change management into the release process, alignment with strategic goals can be enhanced to minimize disruptions and support continuous improvement through business insights. While this integration may introduce complexity, require rigorous approvals, and extend timelines. Eventually, it balances innovation with stability, leading to more successful software updates.

Managing NFRs slippage through the release management workflow with the help of change management

To understand the impact of change management's role in release management, we collected inputs over a limited period for the exception slippage analysis, a crucial aspect of software release management. The insights gained from the NFR slippage analysis will be used to improve future projects and development processes. Lessons learned and best practices documented and incorporated into organizational knowledge repositories for continuous improvement [15]. Lack of proper coordination between solution providers, technology partners, and customers creates an additional challenge when development activities are managed by third-party vendors' full-stack developers who support from different geographic locations [16]. In addition, the technology partner or the customer cannot fix bugs without understanding how and where they are broken [17].

Experimental Setup: Managing NFRs Slippage Through RM Workflow With Change Management Practices Integration

Objective: This experiment is of utmost importance as it aims to evaluate and improve the effectiveness of integrating change management practices within the release management workflow. The goal is to manage and reduce the slippage of NFRs, such as performance, security, availability, scalability, and compliance, which are crucial for the success of any software development project.

Scope: This study focuses on software development projects that follow Agile or DevOps methodologies. It addresses the common challenge of NFRs being under-prioritized or slipping through release cycles due to a lack of visibility and traceability. The approach incorporates change management checkpoints and processes into the existing release management workflow.

Hypothesis**:** Integrating structured change management checkpoints into the release workflow will improve traceability, accountability, and alignment of NFRs with business goals, thereby reducing NFR slippage across releases.

Methodology:

- Experimental groups: The study involves two experimental groups: a Control Group that follows the existing release management workflow without formal change management, and an Experimental Group that adopts a modified workflow with formal change management checkpoints for NFR tracking. Key elements introduced in experimental group:

• NFR impact assessment during CCB meetings.

• Mandatory NFR evaluation and signoffs are part of release planning.

• Use of a Change Request template that includes NFR fields.

• Post-release NFR compliance review.

- Metrics for evaluation:

• NFR visibility score: The NFR Visibility Score, a pivotal metric in planning, indicates how visible and documented the non-functional requirements are during a software release cycle. It reflects how NFRs are identified, reported, and acknowledged, underscoring the need for comprehensive planning and execution stages.

• NFR slippage rate: The NFR Slippage Rate, a metric of urgency, measures the ration of NFRs that remain unresolved at the end of a release cycle. It provides insight into how many non-functional expectations were not met within the planned timeline, highlighting the need for timely resolution.

Expected outcomes: The expected outcomes include a reduction in NFR slippage across release, improved cross-functional collaboration and accountability for addressing NFRs, and the development of a scalable process framework that can be standardized across teams.

The process entails evaluating the progress of a software project by identifying any deviations from the established schedule and effort. This analysis is crucial for determining necessary actions to achieve reliability objectives within the specified timeframe or, if required, for rescheduling project delivery. Our primary focus was on assessing compliance with NFRs, demonstrating our commitment to quality, and examining the extent of delays associated with unresolved NFR issues.

Discussion

From the collected data over the four quarters of the year 2023 and two quarters from the year 2024, the software development cycle has included 6 major and 26 minor releases. Tables *4* and 5 present consolidated data supporting the slippage analysis section. These tables provides aggregated NFRs slippage data collected prior to the implementation of the change management practices, as well as comparative data showcasing NFR slippage outcomes after integrating change management practices. In general, major releases were those releases with new product features, whereas minor release is the patch releases mostly related to bug fixes. Top and bottom panels of Figure *5* represent the NFR compliance scenario before and after the introduction of change management, respectively. We first present NFR compliance scenario before the introduction of the change management in the first three quarters of the year 2023. It can be observed that a substantial number of issues remain unresolved over the major releases. Forty-eight NFR issues were reported in the first major release in the first quarter of the year 2023, out of which 28 were left unresolved. The same slippage trend continued in the next two major releases also. Out of 68 and 43 reported issues, 21 and 16 issues remain unresolved in these two release periods. The minor releases that support specific functionalities and bug fixes have fewer NFR issues reported compared to major releases. Out of these issues, around 50% issues remain unresolved across the releases.

Quarter	Release Type	Version	No. of NFR Issues Reported	No. of NFR Issues Remaining Unresolved
1	Major Release	23.1.0	48	28
	Minor Release	23.1.1	12	9
		23.1.2	21	11
		23.1.3	18	8
		23.1.4	19	10
		23.1.5	16	10
		23.1.6	12	8
2	Major Release	23.2.0	68	21
	Minor Release	23.2.1	12	8
		23.2.2	19	7
		23.2.3	11	7
		23.2.4	14	9
		23.2.5	12	9
		23.2.6	12	6
3	Major Release	23.3.0	43	16
	Minor Release	23.3.1	13	12
		23.3.2	57	9
		23.3.3	11	2
		23.3.4	12	3
		23.3.5	34	11

TABLE 4: NFR compliance and slippage analysis without change management practices

NFR, Nonfunctional Requirement

Quarter	Release Type	Version	No. of NFR Issues Reported	No. of NFR Issues Remaining Unresolved
4	Major Release	23.4.0	11	4
	Minor Release	23.4.1	3	1
		23.4.2	2	0
		23.4.3	1	0
1	Major Release	24.1.0	3	1
	Minor Release	24.1.1	5	1
		24.1.2	4	0
2	Major Release	24.2.0	10	2
	Minor Release	24.2.1	9	3
		24.2.2	2	1
		24.2.3	1	0
		24.2.4	3	0

TABLE 5: NFR compliance and slippage analysis with change management practices

NFR, Nonfunctional Requirement

From the data collected over four quarters of 2023 and two quarters of 2024, the software development cycle included 6 major and 26 minor releases. The major releases, which introduced new features, have significantly enhanced the software's capabilities, making the investment more valuable. Minor releases focused on bug fixes. Figure *5* shows NFR compliance before and after introducing change management, starting with the scenario from the first three quarters of 2023. A total of 48 NFR issues were reported in the first major release in the first quarter of 2023, out of which 28 were left unresolved. The same slippage trend continued in the subsequent two major releases. Of 68 and 43 reported issues, 21 and 16 issues remain unresolved in these two release periods. Minor releases have fewer reported NFR issues than major releases, with about 50% remaining unresolved.

We calculated the average number of NFR issues reported per release (23.2) and standard deviation (16.78) to analyze the provided data statistically. The average number of unresolved NFR issues per release was calculated as 10.20 with a standard deviation of 5.70. The average number of NFRs reported per release is higher than the mean number of unresolved NFRs, suggesting that a sizable portion of reported NFRs is being addressed within the development cycle. However, it also indicates a consistent workload in addressing reported NFRs across releases. The higher standard deviation for the number of NFRs reported (16.78) compared to unresolved NFRs (5.70) indicates greater variability in the former. While there may be consistency in addressing NFRs, the number reported can fluctuate significantly between releases, presenting challenges for resource allocation and project planning. This variability can affect project timelines and overall success if not managed properly. It is the responsibility of the development team to explore the root causes and establish strategies to mitigate the impact on project delivery, demonstrating their commitment to the project's success.

The analysis highlights the need for a tactical evaluation of NFR management to align planned and actual efforts, minimizing the impact of unresolved NFRs on release schedules. Implementing a Change Management Board (CMB) can enhance project timelines and reduce slippage by overseeing and approving changes, ensuring each release is assessed for safety, quality, and customer satisfaction.

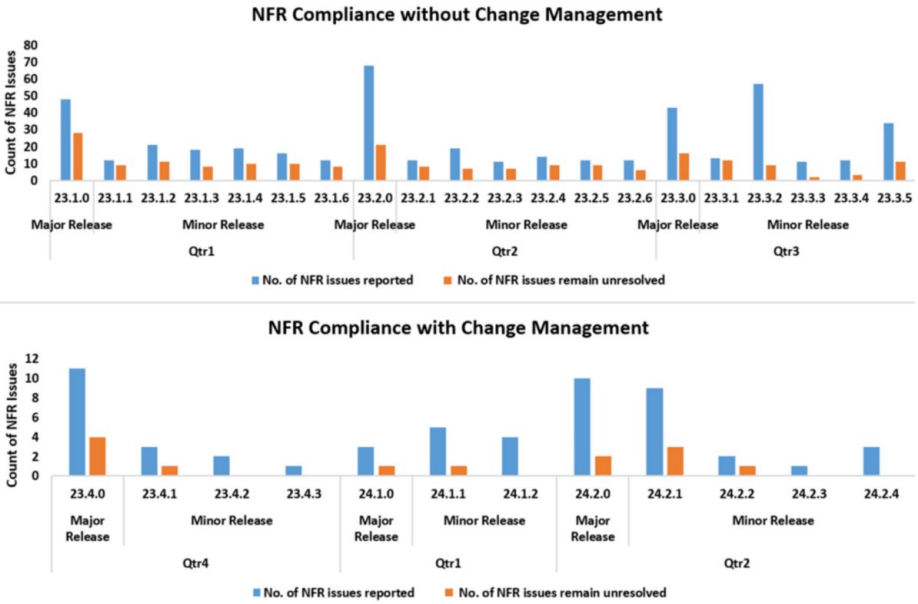

FIGURE 5: The figure presents NFR compliance scenarios before and after the introduction of change management. It is evident from the figure that the number of NFR issues that remain after the introduction of change management is comparatively less than that before.

NFR, Nonfunctional Requirement

The chart in the bottom panel of Figure *5* shows the positive effect of CMB involvement on the slippage of major and minor releases over the fourth quarter of 2023 and quarters 1 and 2 of 2024. Post-CMB, the average number of NFR issues reported per release was calculated as 4.50, with a standard deviation of 3.38. The average number of unresolved NFR issues per release was calculated as 1.08, with a standard deviation of 1.26. It is evident from these values that the average number of NFR issues reported, and remaining unresolved per release post-CMB were reduced significantly, suggesting significant improvement in resolving issues because of the introduction of CMB. Further, the lower standard deviation for the number of unresolved NFRs (1.26) indicates lesser variability. It suggests consistency in resolving NFRs and reducing resource allocation and project planning challenges.

	Before Introduction of CM		After Introduction of CM	
	Mean	Std. Dev.	Mean	Std. Dev.
NFR issues reported	23.2	16.78	10.20	5.70
NFR issues remaining unresolved	4.50	3.38	1.08	1.26

TABLE 6: NFR compliance scenario before and after introduction of CM

CM, Change Management; NFR, Nonfunctional Requirement

The compliance scenarios for NFRs both before and after implementing change management are presented in Table *6*. Preliminary findings indicate that the involvement of the CMB in the release process significantly enhances the management of NFR slippage by effectively addressing critical concerns, which leads to more reliable release cycles. This suggests that having the appropriate authority within the CMB positively impacts project management, with strategic decision-making regarding NFRs being essential in minimizing project delays. The reduction in project delays is a clear indication of the positive impact of the CMB's involvement. To evaluate this relationship, we conducted a Mann-Whitney U test to assess whether there is a significant difference in the mean number of reported and unresolved NFR issues before and after the CMB's appointment, assuming a normal distribution among associates.

Evaluate for the number of reported NFR issues:

· Null Hypothesis (*H*0): There is no significant difference in the number of reported NFR issues before and after the appointment CMB.

· Alternative Hypothesis (*H*1): There is a significant difference in the number of reported NFR issues before and after the appointment.

· The p-value (<.00001) is less than the significance level (0.01). Therefore, there is sufficient evidence to conclude that there is a significant difference in the number of reported NFR issues before and after the appointment of CMB.

Evaluate the number of NFR issues that remain unresolved:

· Null Hypothesis (*H*0): There is no significant difference in the number of NFR issues unresolved before and after the appointment of CMB.

· Alternative Hypothesis (*H*1): There is a significant difference in the number of NFR issues unresolved before and after the appointment.

· The p-value (<.00001) is less than the significance level (0.01). Therefore, there is sufficient evidence to conclude that there is a significant difference in the number of unresolved NFR issues before and after the appointment.

Evaluate the ratio of the unresolved NFR issues per reported NFR issues:

· Null Hypothesis (*H*0): There is no significant difference between the ratio of the unresolved NFR issues remaining per reported NFR issues before and after the appointment CMB.

· Alternative Hypothesis (*H*1): There is a significant difference between the ratio of the unresolved NFR issues that remain per reported NFR issues before and after the appointment.

· The p-value (0.00035) is less than the significance level (0.01). Therefore, there is sufficient evidence to conclude that there is a significant difference between the ratio of the unresolved NFR issues remaining per reported NFR issues before and after the appointment CMB.

Key takeaways for practitioners

Transitioning from legacy on-premises solutions to native SaaS delivery presents significant complexities due to the interdependencies among development, cloud operations, and release management teams. Effective collaboration between change management experts and integrating NFR is essential. Change management professionals ensure a smooth transition and align stakeholders with the organization's goals. Concurrently, NFR integration addresses the handling of NFRs during release management, safeguarding the quality and performance of the SaaS solution.

A lack of knowledge regarding cloud infrastructure further complicates the release process, particularly about requirements such as customer-centricity and comprehensive documentation. Engineers developing SaaS products must continue supporting legacy systems until the novel solutions are fully operational to maintain stakeholder confidence and morale. Integrating change management into the release process requires a delicate balance. While it can potentially enhance stakeholder engagement, it may also introduce delays due to the need for additional approvals. Therefore, striking the proper equilibrium between robust change management and agile practices is vital for fostering innovation. This balance will reassure all parties that careful consideration is given to every transition facet.

Successful implementation of change management process is an ongoing procedure that demands resolute professionals to coordinate efforts across various departments. Though resource-intensive, this continuous collaboration and communication are crucial for preventing stakeholder misalignment and ensuring everyone remains informed and actively involved in decision-making.

Validity threats and limitations of the study

In this manuscript, we emphasize the importance of user feedback in the software development process, advocating for its integration into development cycles to enhance software quality and user satisfaction. Our investigation into release readiness practices during software delivery and the pilot study on the role of change management in managing NFR slippage have significant implications for practitioners and researchers. We acknowledge that there are constraints associated with this research. Our study is focused on software development companies in different countries, so the findings may not be unanimously

applicable to the companies operating in the service industry or still operating through heritage applications. Although we included diverse projects, the sample size was limited. Additionally, the rapid pace at which technology advances make it challenging to draw firm conclusions. The ever-evolving nature of emerging technologies also presents a challenge, as studies may quickly become outdated. Furthermore, due to the diverse nature of software solution providers' change management processes, the inflexible nature of these technologies complicates research efforts, as each platform may require unique integration strategies.

Organizational roadmaps and industry sectors also affect the effectiveness of integration strategies and may limit the generalizability of research findings. On top of that, dealing with vendor bias, organizational resistance, and scalability concerns can add complexity to understanding the impact of integrating emerging technologies in release management. Failure to address these issues may lead to inefficient release management processes, increased risk of system failures, and missed opportunities for innovation. Interdisciplinary research approaches and a focus on long-term sustainability in the future may offer more comprehensive insights into effective integration strategies.

Industrial revolution implementation challenges

Implementing industrial evolution strategies presents various challenges, including integrating heterogeneous technologies, increased vulnerability to cybersecurity threats, and resistance to organizational change. To mitigate these challenges, organizations must adopt modular system architectures that support scalability and flexibility. Moreover, implementing stringent cybersecurity protocols and developing structured, organization-wide training initiatives is vital to equip stakeholders with the necessary skills and facilitate smoother transitions. These strategies are crucial in managing NFRs, particularly in system performance and security, thus ensuring that software systems comply with regulatory and quality standards and meet user expectations. The emphasis on structured, organization-wide training initiatives should make the stakeholders feel prepared and capable for the changes ahead.

As the paradigm of customer-centric software development continues to evolve, the software industry is increasingly required to adapt to the dynamic demands arising from successive phases of industrial transformation. However, empirical studies and industry surveys consistently reveal persistent difficulties meeting project timelines, especially during the transition to SaaS delivery models. In this context, Industry 4.0 emphasizes the importance of automation and data exchange, while Industry 5.0 expands this focus to include human-centric automation, environmental sustainability, and system resilience. Across both paradigms, the role of effective release management cannot be overstated. It is essential for ensuring operational continuity, enhancing product quality, and aligning technological capabilities with strategic organizational goals.

Conclusions

The manuscript examined the maturity of the release management (RMW) workflow regarding innovative technology adoption and managing NFRs through structured change management. Our findings highlight the critical role of RMW in ensuring compliance, mitigating NFR slippage, and improving SaaS delivery. The proposed framework offers a structured approach to tracking NFRs, with benefits such as improved compliance and reduced slippage. However, challenges such as compliance gaps, debugging constraints, and overlooked NFRs persist, indicating an urgent need for further refinement in release management. Effective software release management is essential for delivering product updates to end users. A systematic and coordinated approach to release management is crucial for ensuring the quality, reliability, and security of software products and services. These findings can guide practitioners in making decisions about managing release gates and changes during the RMW or improving existing RMW processes to address slippage. We also discuss the limitations of our study, providing a comprehensive view of our research. By understanding the complexities of integrating change management into RMW, practitioners can enhance real-time practices to mitigate the empirical challenges of handling change requests.

Future research should focus on (i) strengthening the role of NFRs in software quality and delivery, (ii) optimizing release management strategies for enterprise software, and (iii) leveraging AI and machine learning for predictive NFR management. Expanding case studies across various industries will provide deeper insights, while advanced change management techniques can enhance automation, compliance, and adaptability in SaaS environments. Addressing these areas will drive innovation in release management, leading to improved software quality and seamless delivery.

Additional Information

Author Contributions

All authors have reviewed the final version to be published and agreed to be accountable for all aspects of the work.

Concept and design: Bishnu Shankar Satapathy, Siddhartha Sankar Satapathy

Acquisition, analysis, or interpretation of data: Bishnu Shankar Satapathy, Ashish Sharma, Madhusmita Dash, S. Ibotombi Singh, Joya Chakraborty

Drafting of the manuscript: Bishnu Shankar Satapathy, Ashish Sharma, Madhusmita Dash, Joya Chakraborty

Critical review of the manuscript for important intellectual content: Bishnu Shankar Satapathy, Madhusmita Dash, Siddhartha Sankar Satapathy, S. Ibotombi Singh

Supervision: Siddhartha Sankar Satapathy, S. Ibotombi Singh, Joya Chakraborty

Disclosures

Human subjects: All authors have confirmed that this study did not involve human participants or tissue. **Animal subjects:** All authors have confirmed that this study did not involve animal subjects or tissue. **Conflicts of interest:** In compliance with the ICMJE uniform disclosure form, all authors declare the following: **Payment/services info:** All authors have declared that no financial support was received from any organization for the submitted work. **Financial relationships:** All authors have declared that they have no financial relationships at present or within the previous three years with any organizations that might have an interest in the submitted work. **Other relationships:** All authors have declared that there are no other relationships or activities that could appear to have influenced the submitted work.

References

1. Gräßler I, Oleff C, Preuß D: Proactive management of requirement changes in the development of complex technical systems. Applied Sciences. 2022, 12:1874. 10.3390/app12041874
2. Paulsson V, Johansson B: Cloud ERP systems architectural challenges on cloud adoption in large international organizations: A sociomaterial perspective. Procedia Computer Science. 2023, 219:797-806. 10.1016/j.procs.2023.01.353
3. Lavin A, Gilligan-Lee CM, Visnjic A, et al.: Technology readiness levels for machine learning systems. Nature Communications. 2022, 13:6039. 10.1038/s41467-022-33128-9
4. Kaiser KA: Release management in DevOps. Reinventing ITIL® and DevOps with Digital Transformation. Apress, Berkeley, CA; 2023. 279-303. 10.1007/978-1-4842-9072-9_10
5. Trzeciak M: Factors of success in the change management process of IT programs. Journal of Organizational Change Management. 2024, 37:58-74. 10.1108/jocm-04-2023-0110
6. Suescún-Monsalve E, Pardo-Calvache C-J, Rojas-Muñoz S-A, Velásquez-Uribe A: DevOps in industry 4.0: A systematic mapping. Revista Facultad de Ingeniería. 2021, 30:e13314. 10.19053/01211129.v30.n57.2021.13314
7. Smeds J, Nybom K, Porres I: DevOps: A definition and perceived adoption impediments. Agile Processes in Software Engineering and Extreme Programming. Lassenius C, Dingsøyr T, Paasivaara M (ed): Springer, Cham; 2015. 212:166-177. 10.1007/978-3-319-18612-2_14
8. Grande R, Vizcaíno A, García FO: Is it worth adopting DevOps practices in global software engineering? Possible challenges and benefits. Computer Standards & Interfaces. 2024, 87:103767. 10.1016/j.csi.2023.103767
9. Runeson P, Höst M: Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering. 2009, 14:131-164. 10.1007/s10664-008-9102-8
10. Werner C, Li ZS, Ernst N, Damian D: The lack of shared understanding of nonfunctional requirements in continuous software engineering: Accidental or essential. 2020 IEEE 28th International Requirements Engineering Conference (RE), Zurich, Switzerland. 2020, 90-101. 10.1109/RE48521.2020.00021
11. Phillips J, Klein JD: Change management: from theory to practice. TechTrends. 2023, 67:189-197. 10.1007/s11528-022-00775-0
12. Jayatilleke S, Lai R: A systematic review of requirements change management. Information and Software Technology. 2018, 93:163-185. 10.1016/j.infsof.2017.09.004
13. Satapathy BS, Sharma A, Dash M, Satapathy SS, Chakraborty J, Singh SI: The influence of dynamic technologies on the software release management and SaaS roadmap: a survey. International Journal of Business Process Integration and Management. 2025, 12:135-148. 10.1504/ijbpim.2025.146538
14. Satapathy BS, Satapathy SS, Singh SI, Chakraborty J: Continuous integration and continuous deployment (CI/CD) pipeline for the SaaS documentation delivery. Decision Intelligence Solutions. InCITe 2023. Hasteer N, McLoone S, Khari M, Sharma P (ed): Springer, Singapore; 2023. 1080:41-50. 10.1007/978-981-99-5994-5_5
15. Hustad E, Stensholt J: Customizing ERP-systems: A framework to support the decision-making process. Procedia Computer Science. 2023, 219:789-796. 10.1016/j.procs.2023.01.352
16. McCreery J, Moranta V: Mapping team collaboration in software development projects. PICMET '09 - 2009 Portland International Conference on Management of Engineering & Technology, Portland, OR, USA. 2009, 1088-1102. 10.1109/PICMET.2009.5262038
17. Jagtiani J, Bach C, Huntley C: Leveraging big data from open source to improve software project management. IEEE Engineering Management Review. 2018, 46:65-79. 10.1109/emr.2018.2809903