International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 07 Issue: 2
***Article Received:** 25 December 2019 **Revised:** 12 January2019 **Accepted:** 30 February 2019*

Frameworks for New Software Product Launch Adopting Data-Driven Approach

Mainak Mitra,
Manager, Cognitive and Analytics, Deloitte
mainakmitra1210@gmail.com

Soumit Roy,
Technical Architect, Analytics, TCS
soumit123@gmail.com

Abstract
This research paper explores the critical facets of introducing new products and managing them effectively within competitive markets. The study synthesizes contemporary methodologies in product development, launch strategies, and lifecycle management to propose a comprehensive framework for New Software Product Launch Cycle optimization (NSPLC) programs. By integrating theoretical models with empirical data from multiple industries, the paper delineates how organizations can leverage market insights, customer feedback, and technological advancements to optimize their product portfolios.
The research identifies key factors influencing successful product launches, including market segmentation, positioning strategies, and the alignment of product capabilities with consumer expectations. It further examines the role of cross-functional teams in enhancing the efficiency of New Software Product Launch Cycle optimization (NSPLC) processes and the importance of agile methodologies in adapting to changing market conditions. The analysis extends to post-launch strategies, emphasizing continuous improvement and the strategic use of analytics to refine product offerings.

Through a series of case studies, the paper illustrates practical applications of the proposed New Software Product Launch framework and evaluates its impact on organizational performance. The findings suggest that a well-structured New Software Product Launch Cycle optimization (NSPLC) program, combined with robust product management practices, significantly enhances the market success of new products and sustains competitive advantage.
This study contributes to the literature by providing a holistic view of product management that integrates strategic, operational, and tactical dimensions. It offers valuable insights for business leaders, product managers, and academics seeking to understand the complexities of product introduction and management in today's dynamic business environment.

Background

In the dynamic landscape of software development, the launch of new products is a crucial and complex process. Optimizing the launch of software products is pivotal for gaining a competitive advantage and ensuring the product's success. Software product launch optimization involves strategic planning and execution, incorporating elements from both software development and marketing disciplines. As technology continues to evolve at an unprecedented pace, organizations need robust strategies to manage and streamline their product launches effectively.

In today's rapidly evolving digital marketplace, the introduction of new software products is a critical juncture that demands strategic alignment between the software development life cycle (SDLC) and product life cycle (PLC) management. This research paper delves into the nuanced interaction between these frameworks to optimize the launch and ongoing development of software products. The failure of BlackBerry Messenger (BBM) serves as a poignant case study underscoring the consequences of misalignment in these areas. Despite its early success and a peak user base of 90 million in 2014, BBM's decline was precipitated by a failure to innovate and adapt in the face of rising competition from platforms like WhatsApp and Telegram.

Article Received: *25 December 2019* ***Revised:*** *12 January2019* ***Accepted:*** *30 February 2019*

The acquisition of BBM by the Emtek Group in 2016 and subsequent strategic missteps highlight critical vulnerabilities in its approach to product management, particularly in the innovation and adaptation of new features. Contrary to competitors who continuously enhanced user experience and functionality, BBM stagnated, maintaining outdated features that failed to resonate with a younger, more dynamic audience. This research aims to explore how effective synchronization of SDLC and PLC could mitigate such risks, using the BBM example to illustrate the broader implications for software product launches.

The premise of this paper is built upon the theoretical underpinnings of product life cycles, as posited by scholars like Anderson & Zeithaml (1984) and Day (1981), and software development models such as the Waterfall Model discussed by Royce (1970) and refined by Boehm (1976). These frameworks provide a foundational lens through which to examine the critical stages of product introduction and growth, and the strategic imperatives during the maturity and potential decline phases. By analyzing how BBM managed these phases poorly, this paper proposes a model for aligning SDLC and PLC that not only enhances the strategic deployment of new software products but also ensures their sustainable growth and adaptability in a competitive landscape.

Motivation

Numerous cases in the tech industry, such as the shutdown of BBM and its replacement with BBMe, have highlighted the significance of optimizing software product launches. These cases exemplify the challenges faced during a product's decline phase and underscore the necessity for effective management strategies during the launch phase to ensure longevity and success in the competitive market.

Previous Research

Previous studies have extensively discussed the software development life cycle (SDLC) and the product life cycle (PLC), focusing on their respective impacts on software quality and market performance. However, there is a lack of comprehensive research that integrates these two life cycles, especially in the context of optimizing new software product launches. This integration is crucial for understanding how strategic decisions during the development phase can influence the software product's market success.

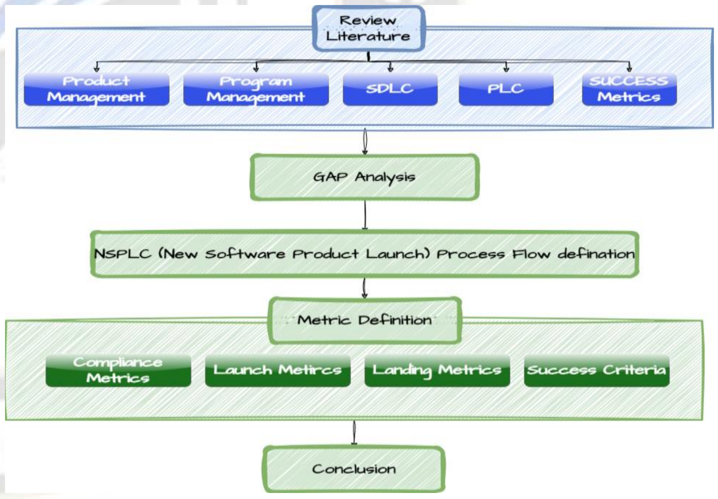
Objectives

This research aims to bridge this gap by aligning the SDLC and PLC, specifically focusing on the launch phase. We seek to identify critical factors in the software development process that can significantly influence the success of a product launch. By doing so, this study intends to provide actionable insights and strategic recommendations for software developers and project managers to enhance their launch strategies.

Methodology

The remainder of this paper is organized as follows: Section 2 reviews relevant literature on SDLC, PLC, Metrics, and their roles in software product launch. Section 3 discusses the methodology employed to integrate these models and analyze their impact on product launch success. Section 4 presents the findings, followed by a discussion in Section 5. Finally, Section 6 concludes the paper with implications for practice and suggestions for future research.

This introduction sets the stage for a detailed examination of how integrating the software development life cycle and product life cycle can optimize new software product launches, ensuring tremendous market success and sustainability.



Literature Review

In the rapidly evolving field of software development, especially with the integration of cloud computing, Metrics-Driven Development (MDD) has emerged as a crucial strategy to enhance the quality, performance, and efficiency of software projects. The literature review delves into several foundational and contemporary sources that elucidate the challenges and methodologies associated with implementing metrics-driven practices in software development, particularly within cloud-based environments.

Introduction to Metrics-Driven Development

Metrics have been highlighted as pivotal tools that enable the effective management of software development processes by providing quantitative insights into various aspects of software production and maintenance [3]. The adoption of metrics spans both process and product dimensions, making them versatile tools for improving software productivity and quality. The utility of software metrics, particularly in cloud environments, has necessitated a reevaluation of traditional metrics to accommodate the dynamic and scalable nature of cloud resources [6].

Challenges of Traditional Metrics in Cloud-Based Development

Traditional software metrics, while effective in conventional settings, often fall short in cloud computing scenarios due to the dynamic nature of the cloud services [6]. These challenges include adapting metrics to manage legacy software migration, maintaining service quality, and ensuring efficient resource utilization. The literature emphasizes the need to develop new metrics or adapt existing ones to reflect the complexities of cloud-based development better [3].

Metrics-Driven Development (MDD) Approach

MDD is proposed as an innovative approach tailored specifically for cloud-based enterprise software development, focusing on employing specific software metrics to guide development processes. This approach is aimed at enhancing operational efficiency and software quality by providing continuous measurement and analysis to inform strategic decisions throughout the Software Development Life Cycle (SDLC) [6].

Enterprise Metrics-Driven Development (EMDD)

Building on the concept of MDD, EMDD has been discussed as an essential extension particularly relevant to large-scale corporate software projects. EMDD utilizes metrics to assess how well software development processes are working, track those metrics, and use them to make informed decisions and streamline procedures [2]. This focus on metrics facilitates improved project management and operational efficiency by identifying and measuring the key effects of the development process.

Implementation and Empirical Validation

Empirical experiments and case studies play a crucial role in validating the effectiveness of MDD within a cloud computing context. The literature includes discussions on specific metrics that assess various aspects of software quality and performance, demonstrating how these metrics can lead to more informed decision-making and better resource management [6].
Future research in MDD is expected to refine the metric selection and analysis techniques, expand the range of metrics to include more cloud-specific indicators and explore the integration of automated tools to enhance the effectiveness of MDD further. This is aimed at ensuring that metrics-driven approaches keep pace with the technological advancements in cloud computing [6].

In the expansive discourse on Metrics-Driven Development within cloud computing environments, additional insights from references on the practical application and theoretical underpinnings of software metrics in agile and quality assurance contexts further enrich the understanding. This elaboration incorporates crucial perspectives from references [5] and [10], which explore the adoption of metrics in agile software environments and the broad implications of software quality assurance.

Adoption of Metrics in Large-Scale Agile Environments

Reference [5] provides an in-depth analysis of the implementation of metrics in large-scale agile software development projects. The study elucidates how metrics such as story point estimation, velocity, and sprint burn-down are instrumental in managing progress and resource allocation. These metrics not only monitor the pace and quality of development but also facilitate strategic planning and execution crucial during the launch phases of new software products. The synthesis of findings emphasizes that while metrics foster transparency and control across organizational levels (team, program, and portfolio), they also face challenges such as resistance from development teams and the complexity of aligning metrics across diverse agile practices [5].

Software Quality Assurance and Metrics

Reference [10] discusses the role of software quality assurance (SQA) and metrics in enhancing the quality of software products. It highlights how metrics provide a systematic means to measure various software complexities, including size, control flow, and data flow metrics. The discussion extends to the categorization of metrics into review metrics, product metrics, project metrics, and process metrics, emphasizing their indispensability in assessing the effectiveness of quality control activities. These metrics, particularly product metrics, are vital for understanding the design and architecture before actual product development, thus playing a crucial role in the quality assurance processes [10].

Integrating Agile and Quality Assurance Metrics in Cloud-Based Development

The insights from references [5] and [10] suggest a convergence of agile methodologies and quality assurance metrics in cloud-based software development. This integration addresses the dual challenges of maintaining agile flexibility and ensuring product quality within the dynamic, scalable environments typical of cloud computing. The literature suggests that a combined approach can lead to more robust software development practices that not only align with but also enhance the strategic objectives of cloud-based enterprise development [5, 10].

Conclusion

The expanded literature review integrates critical perspectives from the adoption of metrics in agile environments and the role of metrics in software quality assurance, offering a holistic view of the metrics-driven development approach. These insights underline the multifaceted benefits of incorporating metrics into software development practices, particularly in cloud-based settings, where they aid in steering agile projects and ensuring the quality and reliability of software products. As cloud computing continues to influence software development paradigms, the integration of sophisticated metrics and quality assurance strategies will be crucial for developing high-quality, efficient, and competitive software products.

The literature underscores the critical importance of adopting a metrics-driven approach in cloud-based environments to handle the complexities of modern software development. It provides a framework for using metrics to manage software quality proactively and optimize performance throughout the SDLC, thereby aligning development efforts with organizational goals and market demands [1, 3, 6].

By systematically addressing these dimensions, the literature review offers a comprehensive perspective on the strategic role of metrics in modern software development, particularly in the context of cloud computing, thus providing valuable insights for both practitioners and researchers in the field.

Research Gap Analysis:

In the dynamic and technologically driven landscape of software development, particularly in cloud computing, Metrics-Driven Development (MDD) has emerged as a key strategy to improve software quality, performance, and efficiency. The integration of cloud computing has heightened the need for refined metrics that align with the scalability and dynamism of cloud resources, necessitating a shift from traditional metrics to more adaptive and comprehensive ones [6]. This gap analysis aims to pinpoint the unaddressed areas in the deployment of MDD across the stages of new software product launches, corresponding lifecycle processes, and compliance metrics.

New Software Product Launch and Lifecycle Process

While the literature extensively discusses the utilization of MDD in managing ongoing software development processes [3, 6], there is a notable gap in specifically tailoring these metrics to the unique challenges of launching new software products. The launch phase involves critical steps like market analysis, positioning, and the initial release, which can significantly benefit from targeted metrics to gauge market readiness, customer engagement, and initial usability feedback. The current literature does not sufficiently cover how metrics can be specifically adapted or designed to manage these launch-specific challenges effectively.

Process Compliance Metrics

The discussions around metrics in the reviewed literature primarily focus on performance and quality assurance [10]. However, there is a significant gap in addressing compliance metrics that align software development processes with regulatory and standards compliance, particularly in highly regulated industries. These metrics are crucial for ensuring that software products meet legal, security, and privacy standards, which is especially pertinent in cloud environments where data governance and compliance are paramount.

Launch and Landing Metrics

The application of metrics in the context of software product launches—often termed as "launch and landing metrics"—is underexplored. These metrics would specifically monitor the transition phases of software projects from development to market launch and then to post-launch adjustments. There is a lack of detailed frameworks or case studies in the literature that delineate which metrics are most effective during these critical phases and how they can predict and enhance the success of software launches in competitive markets.

Success Criteria and Metrics

While empirical studies validate the general effectiveness of MDD [6], there is a paucity of research focusing on defining and measuring success criteria for new software products in the context of MDD. Success criteria may include market adoption rates, customer satisfaction scores, and economic impact, which need to be clearly defined and measured through bespoke metrics. The literature lacks a detailed exploration of how these success metrics are developed, validated, and integrated into the broader metrics-driven frameworks for cloud-based software development.

The integration of agile methodologies and quality assurance metrics in cloud-based software development is advocated as a comprehensive approach to enhance software quality and manage development processes effectively [5, 10]. However, the specific adaptation of these strategies to the phases of new software product launches remains an underdeveloped area in current research. Bridging these gaps can significantly enhance the strategic deployment of metrics, thereby ensuring that new software products are not only launched with precision but also achieve sustained success in competitive markets.

Future research should aim to develop specific frameworks and tools for creating and implementing launch-specific metrics, with a strong focus on compliance and success measurement. This will not only fill the identified gaps but also enhance the practical application of MDD in the rapidly evolving landscape of cloud-based software development.

NSPLC Framework

The New Software Product Launch Cycle Framework is designed to guide organizations through the systematic development and launch of a software product, ensuring each stage is effectively managed and measured using specific metrics. Here's an expanded look at each stage of the framework:

Ideation

Purpose: To spark the creation of new software solutions that address unmet needs or improve on existing solutions.
Process: Teams engage in creative brainstorming processes to generate ideas, utilizing inputs such as emerging market trends, technology innovations, and customer feedback. Tools like mind mapping or ideation workshops are commonly used.
Metrics for Measurement: The number of viable ideas generated, engagement levels in brainstorming sessions, and initial feedback from stakeholder reviews.

Conceptualization

Purpose: To refine brainstormed ideas into workable project concepts that outline the software's purpose, target audience, and high-level features.
Process: This stage involves more detailed discussions and feasibility studies to select the most promising ideas for development. It may include prototyping potential solutions.
Metrics for Measurement: Concept approval rates by stakeholders, feasibility study results, and potential ROI assessments.

Success Criteria Metrics

Purpose: To establish clear, measurable goals that define what success looks like for the project.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 07 Issue: 2
***Article Received:** 25 December 2019 **Revised:** 12 January2019 **Accepted:** 30 February 2019*

Process: Setting specific, measurable, achievable, relevant, and time-bound (SMART) criteria that align with business goals, such as market share, revenue targets, and user engagement metrics.
Metrics for Measurement: Achievement of set KPIs alignment with strategic goals.

Discovery

Purpose: To gather detailed requirements and insights that will guide the development of the software.
Process: In-depth market research, user interviews, and competitive analysis are conducted to identify and understand the needs and expectations of the target audience.
Metrics for Measurement: Number of requirements validated, depth of market insight gained, user involvement level.

Planning and Design

Purpose: To plan the development process in detail and design the software architecture and user interfaces.
Process: Teams develop project timelines, budgets, and resources needed. Designers create wireframes and design prototypes.
Metrics for Measurement: Adherence to timelines, budget compliance, stakeholder satisfaction with designs.

Quality Metrics

Purpose: To ensure the product is developed to high-quality standards from the start.
Process: Integration of continuous testing, code reviews, and quality assurance protocols throughout the development phases.
Metrics for Measurement: Number of defects per coding cycle, pass rate of quality assurance tests, code quality scores.

Compliance Metrics

Purpose: To ensure the product meets all regulatory and legal standards applicable in the market.
Process: Regular compliance audits, security assessments, and adherence to data protection laws.
Metrics for Measurement: Number of compliance issues identified and resolved, audit outcomes, data breach incidents.

MVP (Minimum Viable Product)

Purpose: To quickly get a functional product to early adopters and gather feedback for further development.
Process: Development of a product with enough features to satisfy early customers and provide feedback for future product development cycles.
Metrics for Measurement: User adoption rates, initial user feedback quality, speed of MVP development.

Customer Validation

Purpose: To validate the product's fit in the market and its usability.
Process: Implement beta testing and pilot testing, as well as gather extensive user feedback.
Metrics for Measurement: Customer satisfaction levels, usability issue counts, feedback on product fit.

Iterative Release to GA (General Availability)

Purpose: To refine and enhance the product based on user feedback before full market release.
Process: Iterative updates to the product, improving features, and fixing bugs as needed based on user feedback from the MVP stage.
Metrics for Measurement: Adoption rates of new versions, user satisfaction with updates, time to release each iteration.

EOL (End of Life) and Deprecation Strategy

Purpose: To strategically phase out the product once it does not meet market needs or is overshadowed by newer technologies.
Process: Planning for the orderly wind-down of the product, including customer notifications, migration support to newer platforms, and service decommissioning.
Metrics for Measurement: Efficiency of migration processes, customer retention during transition, support request rates post-deprecation.
This elaborate framework provides a detailed road map for organizations aiming to manage software development processes effectively. It ensures that each stage is geared towards maximizing the quality, efficiency, and market fit of new software products.

MVP vs MSP

The concepts of Minimum Viable Product (MVP) and Minimum Shippable Product (MSP) are pivotal in modern software development methodologies, especially in environments embracing agile, lean startup, and iterative development frameworks. Both strategies aim to optimize the product development cycle, but they focus on different aspects of the product launch process. Here, we will explore what MSP and MVP entail and discuss their respective advantages and disadvantages.

Minimum Viable Product (MVP)

Definition: An MVP is a version of a new product which allows a team to collect the maximum amount of validated learning about customers with the least effort. The key objective is to test hypotheses about the market, target audience, and product features with minimal resources.

Pros of MVP:

1. Rapid Market Feedback: MVP allows developers and companies to understand customer needs and validate assumptions quickly and with minimal investment.
2. Cost Efficiency: Lower initial development cost due to focusing on core functionalities.
3. Adaptability: Quick iterations based on user feedback allow for flexible adaptations to the product, potentially leading to a better final product that is more aligned with user demands.

Cons of MVP:

1. User Dissatisfaction: Since MVPs focus on minimal features, early users might feel the product is incomplete or lacks functionality, negatively affecting their user experience.
2. Brand Risk: Launching a product that’s perceived as incomplete or below expectations can harm a company's brand reputation.
3. Scope Creep: Extensive iterations based on diverse user feedback can lead to deviation from the original purpose.

Minimum Shippable Product (MSP)

Definition: An MSP refers to a product that includes not just the core features needed to function and be legally compliant but also enough features to ensure customer satisfaction. It's the minimal version of the product that is fully functional, deliverable, and ready to provide a positive user experience.

Pros of MSP:

1. Higher User Satisfaction: MSPs can increase initial customer satisfaction by focusing on delivering a product that meets user expectations in terms of functionality and usability.
2. Market Competitiveness: MSPs are typically more polished than MVPs, making them more competitive in markets where users have higher expectations of product readiness and quality.
3. Reduced Post-Launch Work: Since MSPs address major bugs and user experience issues pre-launch, there is often less work required post-launch compared to MVPs.

Cons of MSP:

1. Higher Initial Cost: Developing an MSP can be costlier and more time-consuming as it involves more detailed work on additional features and polishing.
2. Delayed Feedback: The time to market is longer, which delays the feedback loop and potentially slows down the pace at which the product meets the market fit.
3. Risk of Overbuilding: There is a risk of investing in features that users do not need or value, leading to wasted resources.

Developing an MVP or an MSP largely depends on the organization's strategic goals, the nature of the product, and the market environment. MVPs are suitable for startups and companies looking to rapidly test ideas in the market and iterate based on real user feedback. In contrast, MSPs might be the better option for established companies or industries where users expect a higher degree of product completeness and sophistication upon release.
In scenarios where user base expectations are high, and the market is competitive, starting with an MSP might be necessary to ensure a positive user experience and credible brand image right from the launch. However, in highly innovative sectors where the market is yet to be defined or is undergoing rapid changes, launching an MVP can be a strategic approach to adapt quickly and efficiently.

Metrics

Defining precise and measurable compliance metrics is essential for monitoring and ensuring adherence to established processes during the Software Development Life Cycle (SDLC). Here are detailed definitions and mathematical formulations for key compliance metrics related to process and execution:

Article Received: *25 December 2019* ***Revised:*** *12 January2019* ***Accepted:*** *30 February 2019*

Compliance Metrics

1. PLC Compliance (Product Life Cycle Compliance)

Definition: Measures adherence to the defined product life cycle processes

This metric ensures that all predefined stages of the product launch, such as ideation, planning, design, and testing, are followed systematically. High compliance suggests disciplined project management and adherence to the development roadmap, which typically leads to more predictable and successful product launches.

$$PLC_{\ Compliance} = \left(\frac{N_{\ PLC\ Compliant\ Processes}}{N_{\ Total\ PLC\ Processes}}\right)$$

2. PLC Gating Compliance

Definition: Assesses adherence to critical decision points or 'gates' in the PLC.

This metric evaluates whether each phase of the PLC meets its defined exit criteria before moving to the next phase. This ensures that prerequisites for progression are met, thus maintaining the quality and feasibility of the project at each stage.

$$PLC_{\ Gating\ Compliance} = \left(\frac{N_{\ Successful\ Gates\ Passed}}{N_{\ Total\ Gates}}\right)$$

3. Artifact Compliance

Definition: Measures the completeness and quality of documentation against required standards.

Proper documentation is crucial for maintaining an audit trail, facilitating project continuity, and ensuring that all project requirements are met. This metric helps in tracking whether project documentation adequately covers the needs as specified in the compliance checklist.

$$Document_{\ Compliance} = \left(\frac{N_{\ Required\ Artifacts\ adhering\ to\ Policy}}{N_{\ of\ Required\ Artifacts}}\right)$$

4. Test Compliance:

Definition: Measures how well the system-level testing covers the defined requirements.

Testing is critical for verifying that the integrated system meets all specified requirements. This metric ensures that the system functions correctly in its intended environment, thus validating the end-to-end system, functional, and nonfunctional requirements.

$$Test_{\ Compliance} = \left(\frac{N_{\ Mandatory\ Test\ Cases\ Passed\ Successfully}}{N_{\ of\ Mandatory\ Test\ Cases}}\right)$$

5. Deployment Readiness Compliance

Definition: Evaluates readiness of the product for deployment based on predefined criteria.

This metric ensures that all necessary steps, such as final testing, documentation completion, and stakeholder approvals, are completed before the product is released into the production environment or market.

$$Dep\ Readiness_{\ Compliance} = \left(\frac{N_{\ Deploytemt\ Check\ List\ Signed\ Off}}{N_{\ Toral\ Mandatory\ Deployment\ Checklist\ Items}}\right)$$

6. Code Hygine Compliance

Definition: Evaluates the completion and adherence to the established standards in code reviews within the development process.

Code reviews are critical for maintaining high code quality by ensuring adherence to coding standards, identifying potential bugs, and enhancing code readability and maintainability. This metric helps monitor the thoroughness and frequency of code reviews, ensuring that code meets organizational standards before it is merged into the main codebase.

$$CodeHygine_{\ Compliance} = \left(\frac{N_{\ Code\ Checkins\ Approved}}{N_{\ Toral\ Code\ Checkin\ \ submitted}}\right)$$

Launch Metrics

1. Launch Velocity

Definition: Measures the speed at which the product reaches market milestones, such as user acquisition or revenue goals, following its launch.

This metric assesses how quickly a product meets its initial market performance targets post-launch. Faster velocities indicate effective market penetration and strong initial demand

$$Launch_{\ velocity} = \left(\frac{TActual_{milestone2} - TActual_{\ milestone1}}{TGoal_{milestone2} - TGoal_{\ milestone1}}\right)$$

2. Customer Happiness

Definition: Evaluates the quality of the product at launch, typically measured by user feedback and initial product performance.

High-quality launches are crucial for setting the tone of the product's market presence. This metric helps understand the product's initial reception by end-users, focusing on aspects like bug reports, user satisfaction scores, and feature reception.

$$Cust_{\ happiness} = \left(\frac{NFeedback_{\ positive}}{NFeedback_{\ total}}\right)$$

3. GTM (Go-To-Market) Readiness Metrics

Definition: GTM Readiness Metrics assess the preparedness of a product and the organization to enter the market, ensuring that all aspects of the go-to-market strategy, including product, marketing, sales, support, and operational readiness, are aligned and fully prepared to execute the launch effectively.

GTM Readiness Metrics provide a quantitative measure of how ready the product and the organization are for the market launch. This metric encompasses various dimensions, each contributing to the overall readiness:

1. **Product Readiness:** Ensures that the product has met all the quality assurance checks, final user acceptance testing, and is compliant with market standards.
2. **Marketing Readiness:** Involves completion of all marketing materials, campaign strategies, and promotional content necessary to support the launch.
3. **Sales Readiness:** Measures whether the sales team is fully equipped with the necessary training, sales kits, and understanding of the product to sell it effectively.
4. **Support Readiness:** Ensures that the customer support team is trained on the new product and ready to handle customer inquiries and issues that may arise post-launch.
5. **Operational Readiness:** Assesses the supply chain, IT infrastructure, and logistical aspects to ensure the product release is smooth.

$$GTM_{\ Readiness} = \left(\frac{NRediness\ Checklists_{\ Complete}}{NReadiness\ Checklist_{\ total}}\right)$$

Landing Metrics

Landing Metrics are crucial indicators used to assess the stabilization and acceptance of a product post-launch. These metrics help organizations understand how effectively the product has been integrated into the target market and the level of satisfaction it achieves among the end users.

1. Customer Satisfaction Index (CSI): Measures how satisfied customers are with the product, typically gauged through surveys, feedback forms, and direct customer interactions. This metric is vital for understanding the user's perception and acceptance of the product.

$$CSI = \frac{\sum \quad Score_{\ Cust\ Surveys}}{\sum \quad Max_{\ Possible\ Scores}}$$

2. Product Stability Rate: Assesses the technical stability of the product post-launch, tracking issues like bugs, system crashes, or downtime that affect user experience. A higher stability rate indicates a smoother, more reliable product. The Product Stability Rate (PSR) is a metric used to measure the

***Article Received:** 25 December 2019 **Revised:** 12 January2019 **Accepted:** 30 February 2019*

reliability and stability of a software product over a given period, typically post-launch. It assesses how well the product performs under normal usage conditions without significant failures or disruptions. A common way to calculate the Product Stability Rate involves considering the ratio of stable operation time to the total observed time within a specified period.

$$PSR = \frac{\sum \quad Time_{\ \ without\ stability\ issues}}{\sum \quad Time_{\ observatoin\ period}}$$

3. Adoption Rate: Tracks how quickly and extensively the product is being adopted by the target audience. It reflects the effectiveness of the go-to-market strategies and the product's alignment with market needs.

The Adoption Rate is a key performance metric used to assess the rate at which a new software product is being accepted and used by its intended user base over a specific period. This metric helps determine the effectiveness of the product launch and the initial user engagement. The Adoption Rate is particularly important in gauging early success and potential long-term viability of software products.

$$PSR = \frac{\sum \quad N\ users_{\ \ Onboarded}}{\sum \quad N\ Users_{\ targetted}}$$

Success Criteria

Success criteria for a new software product launch refer to predefined goals and benchmarks that determine the effectiveness and impact of the product in the market. These criteria are crucial for assessing whether the launch meets the strategic objectives set by the company, and they typically encompass a variety of performance indicators that cover market reception, financial performance, user engagement, and technical stability. Setting clear success criteria helps stakeholders understand what needs to be achieved to consider the launch successful and guides post-launch strategies.

Measuring Success Criteria

To measure success criteria effectively, a blend of quantitative and qualitative metrics is used:

Quantitative Measures:

- Sales and Revenue: Track the number of units sold and revenue generated compared to forecasts.
- Adoption Rates: Percentage of the target market that has adopted the product within a certain timeframe.
- Performance Metrics: System uptime, page load times, and other relevant IT infrastructure metrics.
- Customer Satisfaction Index (CSI): A score that reflects the overall satisfaction level of customers, often derived from survey data.

Qualitative Measures:

- User Feedback and Reviews: Analysis of sentiments and opinions expressed in user feedback and product reviews.
- Stakeholder Feedback: Insights from internal stakeholders and partners regarding the product's impact on business processes and market positioning.
- Market Analysis Reports: In-depth reviews from industry analysts that may provide external perspectives on the product's market fit and competitive edge.

Implementing Measurement Strategies

- Set Baseline and Targets: Before the product launch, establish clear baseline values and targets for each metric to measure growth and impact post-launch accurately.
- Regular Monitoring: Use dashboards and real-time analytics to monitor these metrics continuously. This allows for agile responses to any issues or opportunities that arise.
- Post-Launch Reviews: Conduct regular review meetings to assess progress toward meeting the success criteria. These reviews can help adjust strategies or product features based on real-world feedback and performance data.

By diligently setting, monitoring, and analyzing these success criteria [Fig1], companies can not only gauge the immediate impact of their new software product but also make informed decisions to enhance future product development and marketing strategies.

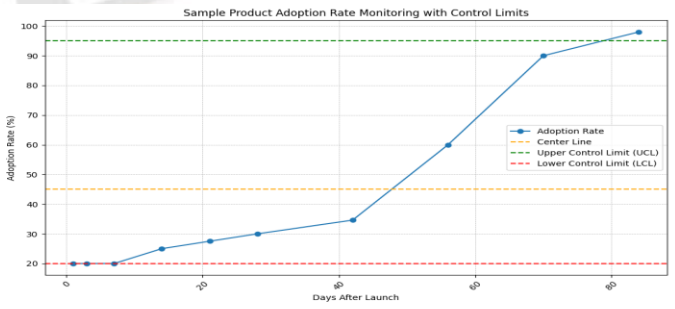


Fig1: Sample Monitoring chart for success Criteria

Future Scope

The current research has laid the groundwork for optimizing new software product launches by integrating and aligning Software Development Life Cycle (SDLC) and Product Life Cycle (PLC) processes within the framework of Metrics-Driven Development (MDD). However, several areas remain ripe for further exploration and development to enhance the efficacy of software product launches in the rapidly evolving technology landscape. Here are some potential directions for future research:

Advanced Metrics Development:

● Real-time Predictive Metrics: Future studies could focus on developing real-time predictive analytics tools that integrate AI and machine learning algorithms. These tools would predict outcomes based on ongoing metrics, allowing for preemptive adjustments during the product launch phases.

● User Engagement and Behavior Metrics: Further research could explore the development of more sophisticated metrics that capture user engagement and behavior in real-time, particularly in response to new features or updates in software products.

Integration of Emerging Technologies:

● Blockchain for Compliance Tracking: Exploring the use of blockchain technology to enhance transparency and traceability in compliance metrics could be a significant advancement. Blockchain could securely log all compliance checks and results, providing an immutable audit trail.

● IoT in Product Feedback Loops: The integration of Internet of Things (IoT) technology could be studied to provide continuous feedback from products in use, directly informing product teams of real-world usage and performance issues.

Cross-Domain Application of MDD:

● Beyond Software: Extending the principles of MDD to other domains such as hardware development or integrated systems (software-hardware) could provide insights into multi-faceted product development challenges.

● Global Deployment Strategies: Investigating how MDD can be tailored for global markets, considering different regulatory and cultural environments, could help multinational corporations streamline their product launches across borders.

Sustainable and Ethical Product Development:

● Sustainability Metrics: Future research could develop sustainability metrics for software products, measuring the environmental and social impacts of software development and usage.

● Ethics in AI-based Metrics: As AI becomes more integrated into metric systems, ethical considerations must be explored, particularly concerning data privacy, bias in AI algorithms, and the potential impacts on employment in the software industry.

Methodological Enhancements:

● Hybrid Methodological Approaches: Research could explore hybrid approaches that combine agile and waterfall methodologies in the context of MDD, aiming to leverage the strengths of both to enhance product launch strategies.

● Case Studies and Longitudinal Studies: Conducting longitudinal studies to follow the long-term impacts of MDD strategies on software products post-launch would provide deeper insights into their effectiveness and areas for improvement.

Tool and Platform Development:

● Custom MDD Platforms: There is a need for the development of customizable MDD platforms that can be tailored to specific industry needs or company sizes, incorporating specific metrics and compliance requirements.

● Integration Tools for SDLC and PLC: Developing tools that seamlessly integrate data and workflows between SDLC and PLC systems could significantly reduce overhead and errors, ensuring smoother transitions between development phases.

Educational and Training Programs:

● Educational Curricula: Integrating MDD concepts into software engineering curricula in educational institutions would prepare the next generation of developers for advanced product management challenges.

● MDD Certification Programs: Establishing certification programs for Metrics-Driven Development could standardize practices and enhance skill sets across the industry.

Conclusion

This research has comprehensively addressed the intricacies of launching new software products through the innovative lens of the New Software Product Launch Optimization (NSPLO) framework. By amalgamating theoretical insights with empirical evidence across diverse industries, the study has illuminated the multifaceted processes involved in product

development, launch strategies, and lifecycle management. It has identified critical success factors such as market segmentation, strategic positioning, and the alignment of product capabilities with consumer expectations, which are pivotal for the triumph of new software products in competitive markets.
The exploration of cross-functional team dynamics and agile methodologies underscores the necessity of adaptive processes that respond to rapid market changes. This approach not only enhances the efficiency of NSPLO processes but also contributes to a robust framework capable of supporting continuous product improvement and the strategic application of analytics to refine and perfect product offerings.

The case studies presented within this research illustrate the practical implementation of the NSPLO framework and validate its efficacy in improving organizational performance. These examples demonstrate that a well-orchestrated approach, grounded in rigorous product management practices, can significantly amplify the market success of new products while fostering a sustainable competitive advantage.

This study enriches the academic and practical understanding of product management by integrating strategic, operational, and tactical perspectives into a cohesive framework. It offers a valuable resource for business leaders, product managers, and scholars, providing them with deep insights into the complex dynamics of product introduction and lifecycle management in today's ever-evolving business landscape.

In conclusion, this research not only contributes to the existing body of knowledge but also sets the stage for future explorations aimed at refining and expanding the NSPLO framework. As the market continues to evolve, the principles outlined in this paper will serve as a guiding framework for organizations striving to optimize their product launch strategies and achieve enduring success in the marketplace.

References:

[1] Wicaksono, Soetam. (2019). Revisiting Requirement Analysis in Product Life Cycle of Software Product. International Journal of Advances in Social and Economics. 1. 10.33122/ijase.v1i3.46.

[2] Chadha, Divya & Singh, Narender. (2014). Product Lining of Cloud Services.

[3] Chopra, Kunal & Sachdeva, Monika. (2015). EVALUATION OF SOFTWARE METRICS FOR SOFTWARE PROJECTS. INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY. 14. 5845-5853. 10.24297/ijct.v14i6.1915.

[4] Rui, Gustavo & Crespo, & Cardoso, Ana & Kokol, Peter. (2004). Complexity-based metrics for the evaluation of the program organization.

[5] Philipp, Pascal & Tobisch, Franziska & Matthes, Florian. (2022). Investigating the Adoption of Metrics in Large-Scale Agile Software Development.

[6] Babu, Gomathi & Santhanam, Srivatsan & Krishnaprasad, Ravikiran. (2023). Streamlining Software Development in Enterprises: The Power of Metrics-Driven Development. International Journal on Recent and Innovation Trends in Computing and Communication. 11. 1610–1617. 10.17762/ijritcc.v11i9.9146.

[7] Saif Himayat, Mohammad & Ahmad, Jameel. (2023). Prediction Systems for Process Understandability and Software Metrics. 10.13140/RG.2.2.11785.21605.

[8] Farid, Shahid. (2017). Gauging Quality of Software Products using Metrics. Technical Journal.

[9] Slhoub, Khaled & Nembhard, Fitzroy & Carvalho, Marco. (2019). A Metrics Tracking Program for Promoting High-Quality Software Development. 10.1109/SoutheastCon42311.2019.9020395.

[10] Avinash, Sivapriya & George, Soumya & Abraham, John. (2013). Brief Analysis on Product Metrics.