Volume 11, Issue 2, February – 2026 International Journal of Innovative Science and Research Technology
ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb900

Optimizing Software Project Performance Through Agile Metrics and System Development Life Cycle Evaluation

Oluwakoya John B.[1*]; Ogbonna Chibueze A.[2]; Maitanmi Stephen O.[3]

[1;2;3]Department of Computer Science Babcock University, Ilishan Remo

Corresponding Author: Oluwakoya John B.[1*]

Publication Date: 2026/02/27

Abstract: The evaluation of project performance has become important due to the persistent failure of software projects. Agile metrics provide essential insights for optimizing delivery by guiding teams towards iterative success through adaptive practices and continuous improvement. Metrics such as sprint burndown, velocity, control charts, and cumulative flow diagrams help track progress, identify risks, and maintain predictable delivery cadences. These tools enable teams to forecast workloads, manage scope changes, and ensure consistent throughput. Control charts and cumulative flow diagrams further enhance efficiency by visualizing cycle times and work status across development stages, helping to detect bottlenecks and workflow irregularities. Quality metrics – including defect counts, deferred defects, and automated test coverage – ensure reliability and customer satisfaction by identifying issues early and reducing rework. Additionally, integrating quality metrics ensure teams maintain high standards across all phases of development. Metrics such as defect counts, deferred defects, and automated test coverage are crucial for maintaining product reliability and addressing potential issues before customer release. Effective quality measurement also supports continuous improvement, emphasizing customer satisfaction and reducing the need for rework through timely identification of flaws. By systematically applying Agile metrics, teams can refine workflows, mitigate risks, and adapt to evolving priorities. This fosters a framework of collaboration, innovation, and sustained progress, ultimately enabling delivery of high – value products with greater predictability and efficiency.

Keywords: *Agile Metrics; Project Optimization; Sprint Burndown; Velocity; Quality Assurance; Cumulative Flow Diagram; System Development Life Cycle.*

How to Cite: Oluwakoya John B.; Ogbonna Chibueze A.; Maitanmi Stephen O. (2026) Optimizing Software Project Performance Through Agile Metrics and System Development Life Cycle Evaluation. *International Journal of Innovative Science and Research Technology,* 11(2), 1707-1719. https://doi.org/10.38124/ijisrt/26feb900

I. INTRODUCTION

The challenges and causes of software project failures can be addressed with a focus on solutions that target the requirement elicitation, communication gaps with stakeholders, changing requirements, and insufficient documentation. Project delays and schedule spillages caused by miss deadlines are challenges which often lead to schedule overruns and potential cancellations. Project performance can however be improved and project failure rate reduced by identifying and mitigating issues such as staff burnout, ambiguous requirements, and inadequate preliminary planning [27]. Therefore, to ensure improvement of project success, project performance evaluation throughout the Software Development Life Cycle phases cannot be overemphasized. Hence, numerous evaluation metrics are utilized to evaluate project performance across various phases of the Software Development Life Cycle (SDLC).

Software projects' performance is meticulously measured to ensure project success and client satisfaction, with the aim of aligning planned effort, time, and budget constraints. Achieving these goals requires an iterative and interactive approach throughout all the stages of the software development process, the stages include planning, analysis, design, coding, integration, testing, and delivery [17].

Metrics enhances software quality and productivity, hence accurate project success assessment will therefore depend on a range of factors, such as cost efficiency, stakeholder satisfaction, and the lessons learned throughout the project's lifecycle. By employing project evaluation metrics, organizations can measure, monitor, and manage performance, while identifying opportunities that necessitate improvement and aids necessary corrective actions that realign with projects goals [40].

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb900

Metrics are essential tools that ensure projects remain on schedule, meet defined objectives, and produce high-quality software outcomes. The System Development Life Cycle encompasses a series of fundamental processes which include specification, development, testing, deployment, utilization, and maintenance. Each process phase is found critical in achieving optimal project performance.

This paper aims to improve project success through an extensive literature review, yielding relevant insights and recommendations designed to enhance successful project delivery.

II. SOFTWARE CHALLENGES AND SOLUTIONS

Software project failure and abandonment have become rampant due to implementation challenges. These challenges can be addressed using software development life cycle which help minimizes redundancies and enhance technical clarity, despite several efforts to improve project delivery, software projects still face several inherent challenges. [26] noted that these challenges cause project delays and schedule spillages because of miss deadlines which often lead to schedule overruns and potential project cancellations. Non-conformance to requirements is a notable challenge which often results to inefficient software products and occurs when software product failed to meet customer needs or solve the intended problem.

Software implementation challenge is Staff burnout which occurs when team members experience burnout, especially if projects do not align with team areas of interest or if key personnel leave. It also happens when there are disproportionate efforts for minor changes, or when small modifications demand extensive effort, exacerbating project delays. These challenges appear in different forms, and the causes and appropriate solutions are further explained below.

- Inadequate Communication: Communication constraint with stakeholders is often caused by factors such as time constraints, geographical distances, and inadequate customer representation. Therefore, effective communication and involvement of stakeholders is essential for accurate requirement changes and validation processes; without this, prioritization and high-quality user interaction suffer, leading to incorrect or incomplete requirements [17, 31]. Hence, direct communication, surveys, and regular interviews with clients can mitigate this challenging issues [12].
- Incomplete Documentation: Lack of proper documentation have created challenges for development and quality assurance teams; it also complicates requirement management for large projects [38]. Issues such as requirement ambiguity and insufficient domain knowledge further compound the problem [20]. The need to establish documentation standards is emphasized to track, manage, and validate requirements effectively.
- Unclear, Conflicting, and Missing Requirements: Lack of clarity and inconsistency in user stories often lead to severe quality and schedule issues [2]. Also, missing interfaces between requirements can trigger significant rework [22]. Hence employing formal methods for requirements specification will address ambiguities, inconsistencies, and evolving needs [27].
- Prototyping challenges: Author [32] advocates the use of design thinking emphasizing empathy, ideation, prototyping, and testing with a view of preventing excessive implementation during initial prototypes. However, reusing code at this stage may lead to quality issues and stakeholder rejection of more robust cycles [17]. Therefore, utilizing paper prototyping will minimize premature implementations and provides an adaptable, quick-to-change approach [33].
- Insufficient Planning and Team Involvement: Insufficient planning often leads to scope misalignment and inaccurate schedule and budget estimations [25]. Therefore, engaging the entire team from the project's inception will ensures that all technical details and requirement changes are well-understood by the team. Author [2] noted that engaging development teams through questionnaires and interviews can aid in accurate estimations.
- Teams' skills and Experience: Miscommunication and inadequate analysis stemming from inexperienced teams may result in lower project success rates. Therefore, engagement of practitioners in training sessions, reference materials, and best-practice guidelines will enhance team expertise.
- Customers inability to articulate User Stories: When customers cannot properly articulate user's stories, the incomplete domain knowledge often leads to requirement prioritization issues, rework, and added costs among user groups [40]. Training customers and involving them in decision-making throughout the requirement engineering process can reduce these challenges.
- Tacit Knowledge Challenges: This is implied knowledge, which is unwritten, but experience-based and can be difficult to convey, leading to information gaps [34]. Direct communication, observation, and knowledge-sharing sessions within teams will help mitigate this challenge and promote knowledge dissemination [35].
- Issues of Changing Requirements: Evolving customer needs can introduce interface compatibility issues, which inflate project costs and cause system failures [12, 18]. The use of ARCM-RM (Agile Requirement Change Management Readiness Model) and tools like JIRA and RE-KOMBINE will helped manage these changes effectively [27].
- Requirement Prioritization Challenges: [23] affirmed that Stakeholder unavailability and market changes complicate requirement prioritization. A framework proposed by [29] characterized as encompassing identification, verification, estimation, and prioritization can streamline agile requirement management across software development stages.
- Non-Functional Requirements (NFRs) Issues: According to [12] Non-Functional Requirements often lack defined gathering and evaluation processes in agile methods, impacting overall product quality and project timelines. Author [41] propose NORMATIC, a Java-based

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb900

simulation tool for modelling NFRs in agile projects, which enable effective planning and implementation [30].

Software project challenges and effective management through the application of software evaluation metrics will enhance project performance.

III. MANAGING SOFTWARE PROJECT FAILURE CAUSES

The true causes of software project failure are often poorly identified during software development, or more accurately, the numerous factors contributing to failure may be difficult to avoid hence different evaluation criteria are used by various stakeholders to resolve the problematic nature of project outcome which has been identified to be caused by their multi-dimensional nature.

Research on project failure have identified several key reasons for project failure as highlighted below.

➢ *Lack of Precision in Project Success Factors:*

Many factors recommended for improving project outcomes lack precision. Therefore, different success factors which include Adequacy of resources and training, appropriateness of development technology and methods, clarity of goals and system requirements, users' active involvement and qualifications are essential in achieving improvement of project success.

➢ *Difficulty in Implementing Critical Success Factors:*

Some critical success factors, like the need for top management support, are challenging to measure or implement. Although such support is often critical, obtaining it can be difficult in practice.

➢ *Organizational or Environmental Impacts:*

Organizational context may influence the implementation of recommended measures. For instance, a blanket recommendation for strong top management support may inadvertently lead to overcommitment to a failing project, thereby worsening existing problems.

Most software projects failure are caused by misleading generic factors used during software development across project life cycle and more often the changing nature of software system development are not considered [54]. In fact, human related factors often contribute to failure of most software systems, as well as institutional contexts and the complexity of stakeholders' interaction and interrelationships challenges during project development.

➢ *Use of Generic Factors*

Generic factors as noted in [54] often imply that all factors are independent, universally applicable, and equally important throughout a project's life cycle which is misleading and often result in project failure. In practice, the significance and influence of various factors change over time, requiring dynamic and context-specific management.

➢ *Evolving environment of Software Systems Development*

The rapid technological changes, complex and globalized business environment, and evolving development practices are shaping modern software development. The changes resulting from the changing environment of software system development according to [54] include:

- Devolution of Development Expenditures: More development costs are now absorbed by business units or user groups.
- Increased Packaged Software Acquisition and Customization: Organizations increasingly rely on off-the-shelf solutions that are then customized to meet specific needs.
- Outsourcing Trends: Outsourcing of systems development has grown, often reducing in-house capabilities.
- Emphasis on Vendor Relationships and System Configuration: Greater focus is placed on vendor selection, product features, system configuration, and necessary adjustments to business processes.
- Smaller Project Sizes and Incremental Delivery: Large projects are often delivered in smaller, manageable parts to improve chances of success.
- Role of Project Management: Effective project management has become crucial, particularly for large or complex enterprise-wide systems, to navigate the intricacies of development.

➢ *People and Process Factors*

The interaction and interrelationship of people and process often posed project challenges which stem not only from technical issues but also from organizational and human-related issues. Software project involves various stakeholders, developers, analysts, users, Management, external agents, project team in social interaction and issues will often arise during system development, which are enumerated below.

- Organizational Strategy Alignment: Development and implementation efforts can be undermined when organizational and business strategies misaligned.
- User Participation and Ownership: Insufficient user involvement can lead to poor system adoption and underperformance.
- Education and Training: Lack of adequate training hampers system utilization and performance.
- Resource Availability: Insufficient resources and support for organizational structures, processes, and cultures can derail projects.
- Attention to Job Design and Usability: Neglecting aspects such as job roles, usability, and user motivation can hinder system acceptance and effectiveness.

Participants in software development are diverse, with varying levels of technical, socio-political, and user-oriented expertise. Therefore, introducing new or modified systems may disrupt established roles and professional cultures, making implementation more challenging.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb900

➢ *Institutional Context and System Development*

Institutional context plays a significant role in shaping system development, this context includes social, political, and economic factors. The socio-economic environment in which the organization operates and the organizational context also have influence on software system development. Influencing factors such as labour market conditions, government regulations, industry pressures, and cultural dimensions can affect the direction and outcomes of development projects.

➢ *Complex Inter-Relationships and Interactions*

Software project outcomes are influenced by multiple, interacting factors whose effects may change over time. Successful project management requires understanding these dynamic interactions between people, processes, and technology within their unique contexts [54]. Empirical research focusing on the institutional dimensions of software development across various settings and stakeholder groups is essential for better understanding of these inter-relationships to improve project outcomes.

Consequently, the evolving nature of software development demands detailed analysis of processual, political, and behavioural factors, from diverse stakeholders and this emphasized the need for context-specific approaches to project management and execution to enhance project performance and success. The solution-based approach is the application of evaluation metrics across the System Development Life Cycle phases that will enhance better project performance.

IV. SOFTWARE DEVELOPMENT LIFE CYCLE PHASES

Software Development Life Cycle (SDLC) represents a structured methodology for software systems development through a sequence of well-defined phases that is required to manage analysis and design processes. The achievement of project success is enhanced through project management which defines project scope, resources, schedules, potential risks, and cost estimations [32]. Project performance should show a clear picture of project health and grant trust in the project team. Therefore, to effectively measure software project performance, system development life cycle is essential. The aim and objective of providing high-quality software that exceeds customer expectations can therefore be achieved through the Software Development Life Cycle, the process used to design, build and test high-quality software [39]. The phases of system development life cycle include:

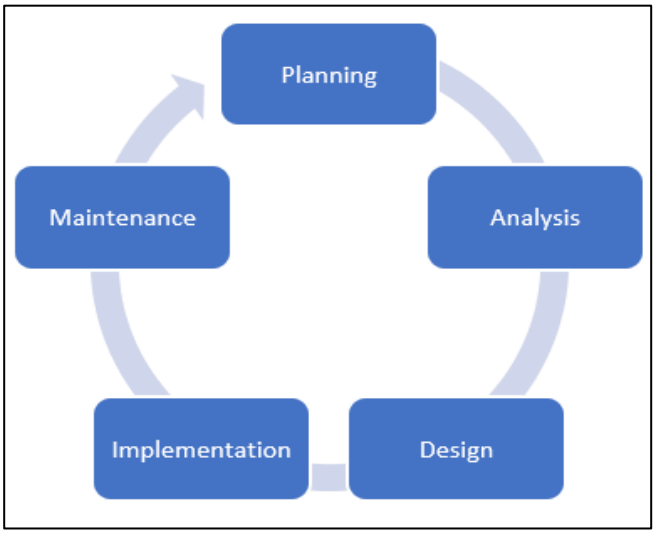

Fig 1 Software Development Life Cycle Phases

➢ *Planning Phase:*

The planning phase identifies problems within existing systems and establishes project priorities. The activities at planning phase include developing a comprehensive project work plan, determining system requirements, assigning team roles, and resource allocation. Hence, to mitigate budget variances and ensure stability, proper planning is important. Thus, if project objectives are aligned properly unrealistic timelines, budget constraints, and ambiguous goals will be avoided [54].

➢ *Analysis Phase:*

System requirements are documented during analysis phase through structured detailed investigation to reduce redundancies. Requirement determination process includes studying user expectations, analysing existing systems, and defining its enhancements or replacements. Alternative design solutions are then considered based on cost and labour. At this phase, technical, economic, behavioural, operational, legal, and temporal feasibility studies are conducted to support decision-making [56]. The output is a recommended solution, which, upon approval, allows for the acquisition of hardware and software as well as detailed specifications development.

➢ *Design Phase:*

The design phase is when the chosen solution is converted into logical and physical system specifications without any coding. Logical design focuses on the business functionality of the system, while the physical design defines system-specific elements like data flows, input/output screens, and processing modules. The choice of programming language, database system, and technical environment is determined at this phase [42]. The output is a physical system specification ready for implementation.

➢ *Implementation Phase:*

System specifications are turned into working code at the implementation phase through activities such as coding, unit testing, integration, and system verification. Implementation includes the creation, testing, and deployment of the system, transitioning it into regular

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb900

organizational use [44]. The Installation methods include direct, parallel, or phased which are chosen based on project needs. User training and documentation finalization are also conducted, with its planning initiated during the earlier phases.

➢ *Testing Phase:*

Testing ensures software quality and identifies errors early to improve reliability and user satisfaction. The strategies employed during testing phase includes unit testing, integration testing, functionality acceptance testing, and regression testing. Delays in development often reduce testing focus, however early testing remains crucial for quality assurance [39]. Once defects are corrected, the software is ready for delivery to customer.

➢ *Maintenance Phase:*

Maintenance phase involves continuous support, updates, and issue resolution. It consists of four main types:

- Corrective Maintenance is required for system error correction.
- Adaptive Maintenance is required when system need to align with changing conditions.
- Perfective Maintenance is required for user-requested enhancements and added functionalities.
- Preventive Maintenance is required to minimize system failure risks.

Maintenance is required when adapting to changing business environments, regulations, and evolving user needs. Proper documentation is however critical in ensuring maintainability and cost-efficiency, the changes are deployed through structured releases, which include detailed release notes and system updates [56].

Adopting the system development lifecycle helps minimize redundancies and enhance technical clarity and where the critical stages were not properly managed software projects are often abandoned [32].

V. EVALUATION METRICS ACROSS DIFFERENT SYSTEM DEVELOPMENT LIFE CYCLE PHASES

Understanding software project complexity provide basis for the use of evaluation metrics to enhance project performance [56]. Evaluation metrics are essential for assessing project performance across the Software Development Life Cycle (SDLC) and it ensure that requirements objectives are met, design remains efficient, code quality is maintained, and the system fulfils its intended purpose [47]. The common metrics that can be applied at different development phases are discussed below.

➢ *Requirement Phase:*

- Requirement Coverage: This measures the percentage of documented requirements that are addressed in the software. Higher coverage indicates alignment between documented needs and implementation. Project performance will depend on how the functions and requirements expected from the software are defined and the extent of requirement coverage.
- Requirement Stability: This evaluates changes to requirements over time, highlighting project volatility and the stability of initial specifications to effectively manage requirements volatility.
- Requirement Traceability: This assesses the ability to trace each requirement through subsequent phases (design, implementation, testing, and deployment) to ensure that each requirement is accounted for therefore users' involvement in the requirement gathering is found necessary.

➢ *Design Phase:*

- Design Efficiency: This quantifies how effectively the design meets functional and non-functional requirements with minimal complexity and flaws.
- Modularity and Reusability: This evaluates the extent to which system components are modular, maintainable, and reusable across different projects, emphasizing long-term adaptability.
- Design Review Findings: This tracks the number and severity of issues discovered during design reviews, serving as an indicator of design robustness and adherence to standards.

➢ *Development Phase:*

- Code Quality: This is measured using metrics like cyclomatic complexity (complexity of the code), code coverage (percentage of code covered by tests), and static code analysis results (code adherence to coding standards).
- Defect Density: This represents the number of defects identified per unit of code (e.g., per thousand lines of code), highlighting code stability with a view of eliminating defects, code improvements, and managing requirement changes.
- Code Churn: This tracks the frequency of code changes, which can impact code stability and long-term maintainability.

➢ *Testing Phase:*

- Test Coverage: This measures the percentage of code executed by tests, indicating the thoroughness of testing efforts.
- Defect Detection Rate: This quantifies the number of defects found during testing per unit of time, serving as an indicator of testing efficiency.
- Regression Test Effectiveness: This assesses the ability of regression tests to detect new defects introduced by code changes, ensuring existing functionality is not broken.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb900

➢ *Deployment and Maintenance Phase:*

- Deployment Success Rate: This reflects the percentage of successful deployments without major issues, indicating deployment reliability.
- Mean Time to Repair (MTTR): This is the average time taken to resolve defects or issues identified after deployment, demonstrating maintenance efficiency.
- Customer Satisfaction: This is used to gathers user or stakeholder feedback on software usability, reliability, and performance to ensure the system meets user needs.

VI. PROJECT MANAGEMENT METRICS FOR PERFORMANCE EVALUATION

Effective evaluation metrics are crucial for tracking and measuring the success of ongoing and completed projects. Insights into various aspects of project performance are provided by the metrics, enabling teams to identify areas for improvement and ensure that project objectives align with business goals. The following are essential project evaluation metrics:

➢ *Schedule Adherence:*

This measures whether tasks and milestones are being completed as planned, ensuring timely project delivery. The metrics is used to monitor project execution, minimize re-work, and identify any deviations from the baseline schedule [13].

➢ *Quality Management:*

A comprehensive quality management plan guarantees that deliverables match the project's purpose and client expectations [26]. Quality Management metrics include the following:

- Planned Quality Metrics: Defines specific measurement parameters for deliverables, ensuring they align with agreed-upon standards and project goals.
- Quality Assurance: Involves proactive monitoring of processes to prevent defects and ensure that high standards are maintained throughout the project lifecycle.
- Quality Control: Focuses on detecting and correcting defects in deliverables to maintain customer satisfaction and product reliability.

➢ *Cost Management:*

The metrics tracks project expenses against the budget to avoid cost overruns and maximize financial efficiency. It ensures resource allocation aligns with project objectives and controls any unexpected spending [28].

➢ *Incident and Change Request Volume:*

The metrics measures the number of incidents or change requests raised during the project. This indicates project stability and flexibility, with fewer incidents reflecting a well-executed plan [24].

➢ *Stakeholder Satisfaction:*

The metrics evaluates stakeholder and client feedback regarding project deliverables and performance. It provides qualitative insights into whether the project meets user needs, expectations, and overall business objectives [38].

➢ *Performance Against the Business Case:*

The metrics assesses whether the project is achieving its intended business value and objectives. It evaluates the project's impact on overall business goals, including profitability, productivity, and strategic alignment [41].

➢ *Lessons Learned and Continuous Improvement:*

The metrics identifies successes, challenges, and improvement opportunities at post-project completion. It encourages iterative improvement for future projects and builds a culture of learning within the organization [28].

➢ *Quantitative Metrics:*

The metrics measures the percentile progress of manpower or other resources deployed during the project. It offers precise data on team output and resource allocation, aiding in capacity planning and optimization [14].

➢ *Practical Metrics:*

The metrics measures project resource efficiency, including team productivity and tool utilization. It assesses how effectively resources are being used to achieve project goals.

➢ *Directional Metrics:*

The metrics monitors project risks, trends, and potential challenges. It helps in proactive risk management and mitigation planning [27].

➢ *Milestone Metrics:*

The metrics tracks progress of key project milestones compared to the baseline schedule. It offers a clear view of project progress, ensuring critical activities remain on track [20].

➢ *Financial Metrics:*

The metrics measures financial gains, return on investment (ROI), and project profitability. It evaluates the overall economic impact and financial success of the project.

➢ *Actionable Metrics:*

The metrics tracks changes within the project, providing insights for necessary corrective actions. It helps in real-time decision-making and responsiveness to project challenges.

➢ *Value-Based Metrics:*

The metrics assesses whether the project meets client requirements related to quality, cost, features, safety, and delivery. It ensures the project delivers maximum value and aligns with client needs and objectives [11].

➢ *Alert Metrics:*

The metrics flags any instances where the project has exceeded acceptable tolerance levels. It acts as an early

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb900

warning system to avoid deviations from the project's scope, budget, or schedule.

➢ *Resource Utilization:*

The metrics evaluates the effectiveness of resource use, including time, budget, and personnel, to identify optimization opportunities [28].

These metrics provide a comprehensive view of project health and progress, facilitating data-driven decision-making and identifying areas for project performance improvement. By applying these metrics, project teams can ensure that project goals are met, risks are mitigated, and project outputs are aligned with initial requirements and expectations.

VII. EFFECTIVENESS OF EVALUATION METRICS FOR PROJECT PERFORMANCE ACROSS SYSTEM DEVELOPMENT LIFE CYCLE

The effectiveness of evaluation metrics for project performance considers critical factors that include adherence to standard methods, user participation, training, and change management approach. This requires a comprehensive planning approach, resource allocation, monitoring, and user collaboration throughout project lifecycle development [54]. Therefore, the essential elements for project success include effective project control, balanced team autonomy, coordination between business and technical needs, and robust stakeholder engagement [8]. Optimized project monitoring and evaluation during system development life cycle will also increase the effectiveness of these key evaluation metrics for system software projects to be delivered successfully and to meet user needs and organizational objectives.

➢ *Use of Standard Methods*

Adopting a standard method for system development provides structure and consistency across various project phases. These Standardized approaches are built on specific philosophies and development paradigms, supported by established techniques and tools that guide activities [12] such as:

- Sequencing Development Activities: This ensures consistent project management practices, reduces uncertainties, and promotes efficient task execution.
- Cost Management and Resource Allocation: This helps control financial aspects by optimizing resource distribution and effort across the development cycle, which is particularly beneficial for large-scale projects.
- Project Team Composition and User Participation: Standard methods facilitate collaboration between developers, project managers, and users. It aligns project goals with stakeholder needs, ensuring that all participants are appropriately engaged.

A standardized methodology streamlines the development process, reducing elapsed time, minimizing effort, and managing costs, thus striving for desirable economic outcomes.

➢ *User Participation*

User participation plays a crucial role in system software development and success [5, 45] through:

- Early Engagement: When users or their representatives are involved from the initial stages of system development, outcomes that align with their expectations are shaped, this enhances user acceptance and satisfaction.
- Expectations and Understanding: Active participation allow users to gain insights into the developing system, thus setting realistic expectations and fostering positive perceptions of the software's benefits.
- Communication and Collaboration: User involvement offers opportunities for mutual exchange between users and developers. Such interactions clarify user needs, improve the articulation of goals, and facilitate conflict resolution, ultimately boosting system usability and acceptance.
- External Knowledge Access: The inclusion of vendors or consultants introduces additional expertise, particularly concerning emergent technologies that may be outside an organization's domain.

The effectiveness of user participation hinges on meaningful and significant engagement, characterized by consultation, communication, personal autonomy, and shared decision-making [13].

➢ *User Training*

Users' training is pivotal for system adoption and long-term success [29, 49].

- Skill Development and Confidence: Proper training equips users with the skills and experience needed to utilize the system effectively, increasing their confidence and reducing resistance.
- Early Introduction: By beginning training during the development process, users can make more valuable contributions and integrate their expertise into system design and testing phases.
- User Attitudes: Comprehensive training minimizes negative attitudes toward the system and strengthens its acceptance, resulting in a smoother transition during implementation.

➢ *Management of Change*

Introducing new systems can lead to significant changes within an organization, affecting user roles, workflows, and relationships [27, 54].

- Change Resistance Management: Effective change management practices mitigate negative emotions such as fear and reluctance by ensuring that users understand the purpose and benefits of the new system.
- Organizational Alignment: Consideration of power structures, job roles, and organizational processes ensures alignment between system implementation and operational needs, reducing potential friction.
- Proactive Change Strategies: Enterprises-wide system changes require early intervention to address

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb900

organizational restructuring, data usage, process adaptation, and employee roles.

Change management practices should begin before system implementation and should involve effective communication, planning, and user support to foster a culture of acceptance and adaptability [28].

- *Optimizing Project Outcomes Through Evaluation of Metrics and System Development Life Cycle*

The consideration of evaluation metrics of people and action, project content, and development processes can result in the achievement of optimal project performance. These key metrics of system software projects, if effectively monitored, evaluated, and optimized through the application of system development life cycle, will enhance successful project implementations that align with user needs and organizational objectives [48].

People and action metrics that can influence optimal project performance include experienced developers with technical expertise and interpersonal skills, engaged and committed users/customers with realistic system expectations, strong supportive leadership, committed top management and effective team collaboration and communication [24].

Project content metrics that can influence optimal project performance include clearly defined and communicated project goals and objectives, adequate allocation of time, budget, and human resources and utilization of appropriate and advanced technology solutions [21].

Development processes metrics that can influence optimal project performance include well-defined and clearly stated user requirements, implementation of a suitable standard development methodology, active and continuous user participation throughout development and comprehensive user training that ensure successful system adoption and utilization [25].

VIII. AGILE METRICS REQUIRED FOR OPTIMAL SOFTWARE PROJECT PERFORMANCE

Agile metrics serve as valuable tools to monitor, measure, and optimize project delivery. The metrics ensures teams deliver incremental value effectively by continuously adapting to changing requirements. These metrics offer insights into project and team performance, highlight areas for improvement, and guide stakeholders in making informed decisions about the project's direction [53]. The key Agile metrics for project optimization also have some challenges with their potential implications.

- *Sprint Burndown:*

The completion of work throughout a sprint can be tracked with sprint burndown chart, allowing Scrum teams to visualize how much work remains and if teams are on track to meet sprint goals. Teams forecast their expected workload for each sprint, aiming to balance the planned and actual completed work [18]. The constrictions of sprint burndown include finishing sprints early due to under-commitment of work, consistently missing sprint goals due to over-commitment, steep drops in the burndown line, indicating work is not broken down into granular tasks and when scope changes are introduced mid-sprint by the product owner [52].

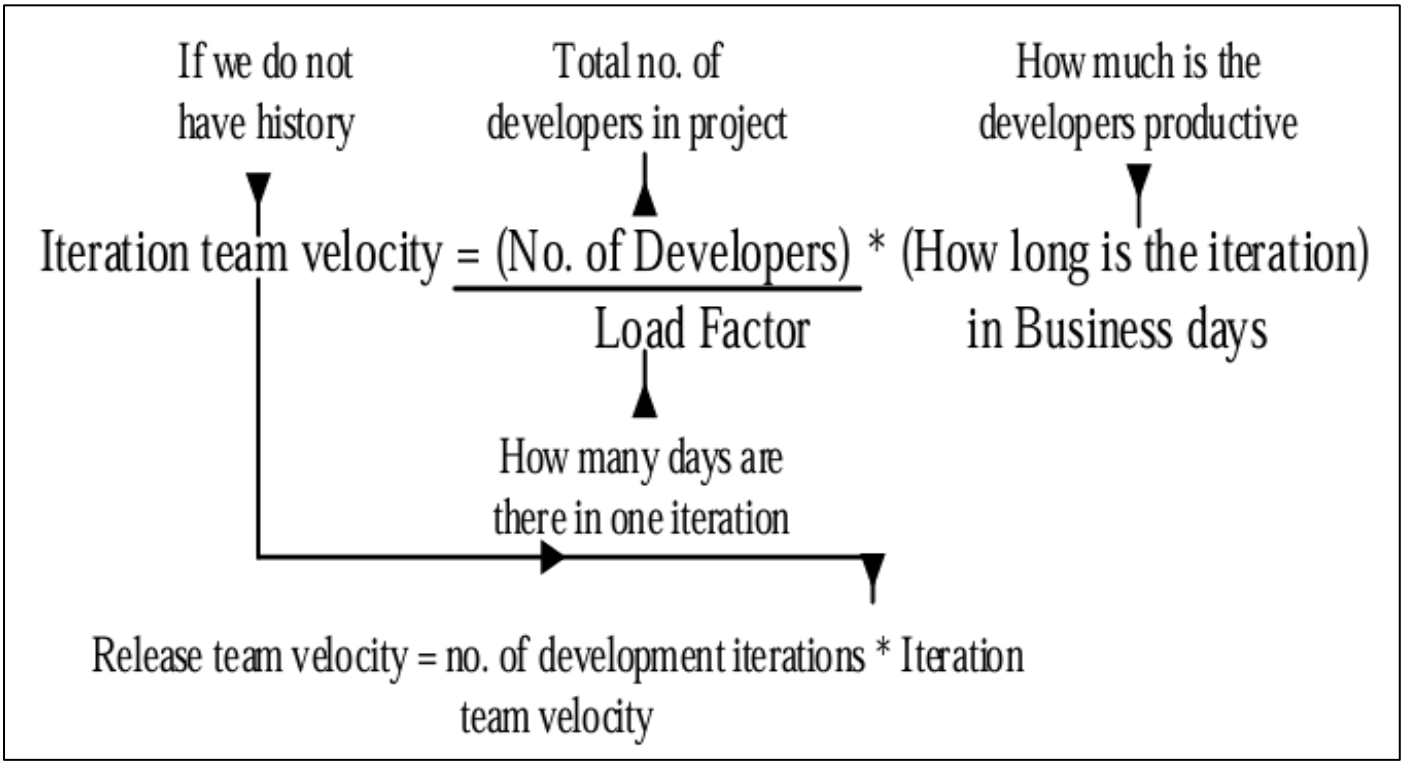

Fig 2 Agile Project Indicators, Based on Ostakhov, Artykulna & Morozov (2018)

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb900

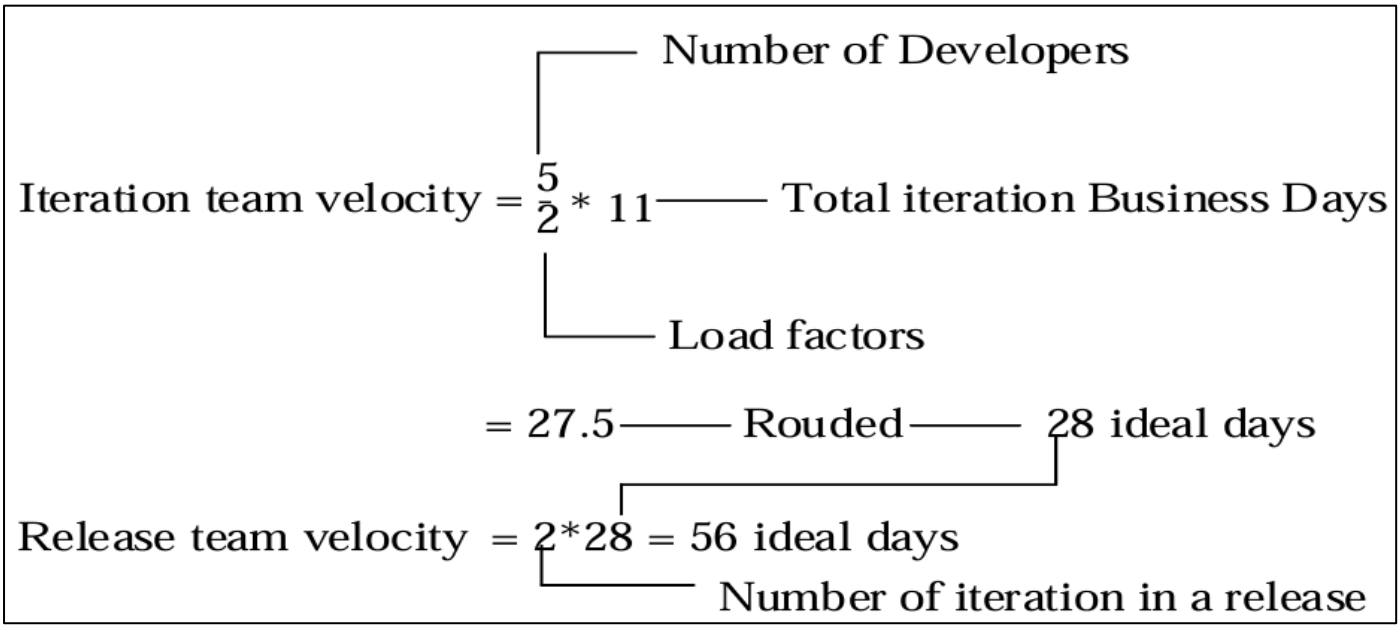

Fig 3 Agile Project Measurement, Based on Ostakhov, Artykulna & Morozov (2018)

➢ *Epic and Release Burndown Chart:*

Epic and release tracks progress over larger bodies of work (epics or versions) compared to a sprint. These charts are vital for monitoring long-term work completion, guiding both Scrum and Kanban teams. Tracking these metrics helps manage changes to scope as teams gain new insights and adjust their objectives. Epic and release burndown bottlenecks include outdated epic or release forecasts not reflecting current work progress, lack of progress over several iterations, chronic scope creep, potentially indicating unclear objectives or problem understanding work scope by the product owner, scope expansion outpacing the team's capacity to absorb it and teams failing to deliver incremental releases during epic development [53].

➢ *Velocity:*

Velocity represents the average amount of work a team completes during a sprint, measured in story points or hours. Velocity provides a reliable metric for forecasting future sprints, helping product owners predict backlog completion timelines. The flaws of velocity include erratic velocity over a prolonged period which indicate flaws in estimation or external pressures, team overestimation leading to missed sprint goals, development pressures causing a deviation from best practices [24]. There is need to focus on unique team dynamics and resist comparing velocities across different teams, as estimation cultures vary.

➢ *Control Chart:*

Control charts measure cycle time (the time from "in progress" to "done"), providing insights into a team's efficiency and consistency in delivering work. Shorter and more predictable cycle times indicate higher throughput. Its primarily used by Kanban teams, Scrum teams can benefit as well [21]. The shortcomings include consistently increasing cycle time, indicating lost agility and erratic cycle times for similar work items, which suggest estimation issues [46].

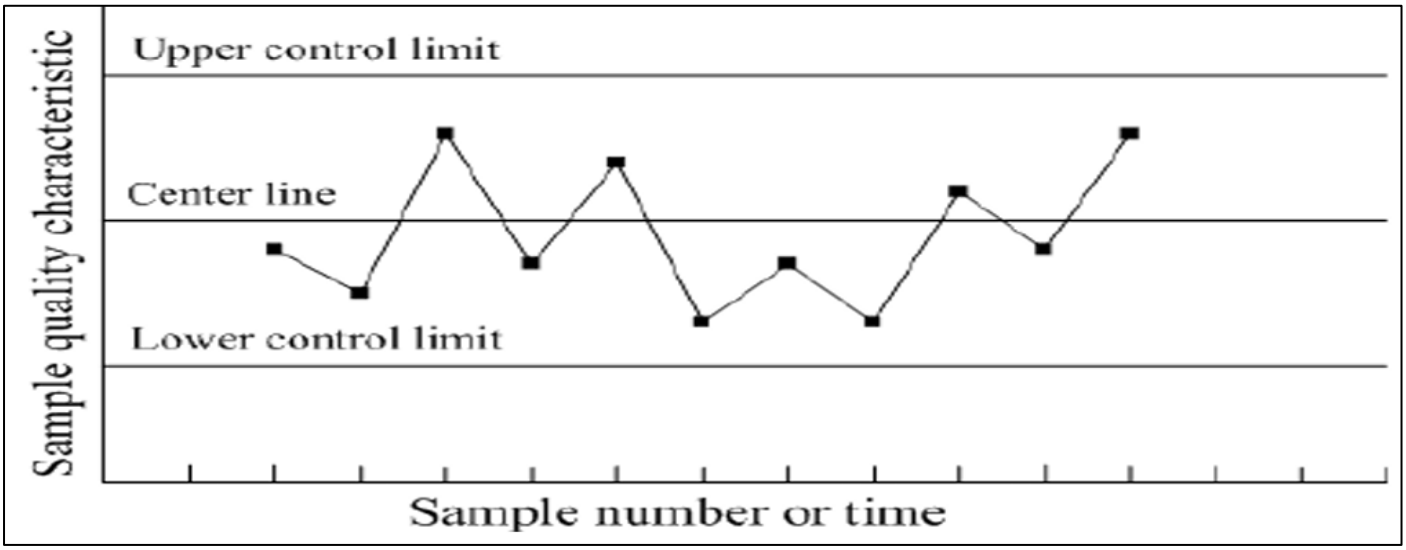

Fig 4 Control Chart Diagram

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb900

➢ *Cumulative Flow Diagram (CFD):*

Cumulative flow diagram provides a visual representation of work status across different stages, with a smooth flow indicating stable progress [30]. Bubble points and bottlenecks reveal areas needing improvement. The bottlenecks of CFD include blocking issues causing backups or starvation in the workflow and backlog growth from obsolete or low-priority issues left unresolved [53].

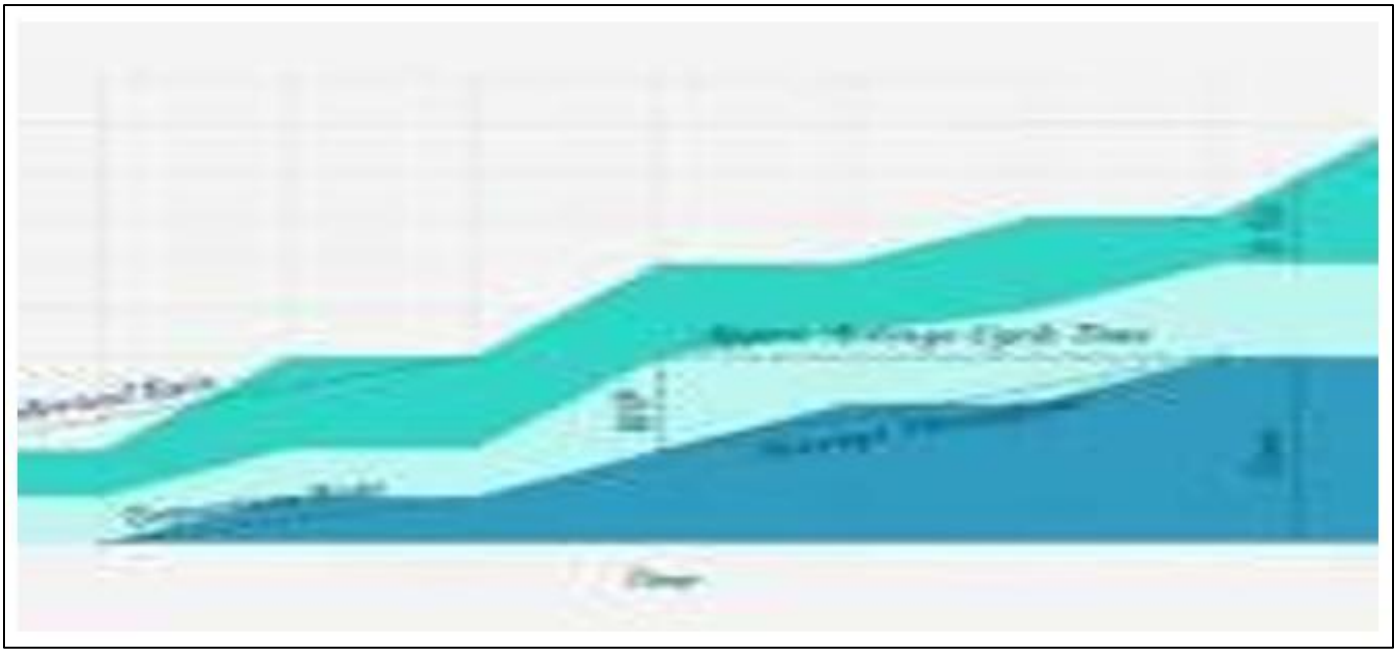

Fig 5 Cumulative Flow Diagram

➢ *Additional Quality Metrics:*

Quality metrics ensure Agile teams maintain high standards throughout development [40] and these includes:

- Defect Count: Tracks defects found during development, after release, or by external parties.
- Deferred Defects: Measures defects delayed to a future release, reflecting team prioritization.
- Customer Support Requests: Indicates customer satisfaction and product stability.
- Automated Test Coverage Percentage: Highlights the extent of automated testing for features, boosting reliability and reducing manual testing efforts.

IX. CONCLUSION

The challenges of project performance can be improved through effective use of Agile metrics and system development life cycle to increase project success. These metrics are vital for project optimization, guiding teams toward better collaboration, more predictable delivery, and the creation of high-value solutions. Through consistent monitoring of Agile metrics such as sprint burndown, velocity, control charts, and cumulative flow diagrams, teams can identify bottlenecks and make necessary course corrections. Optimized project monitoring and evaluation during system development life cycle increases the effectiveness of evaluation metrics for system software projects to meet users' needs and organizational objectives in enhancing successful project delivery.

Integrating quality-focused measures further ensures that customer needs are met, and the product remains robust and reliable. Continuous assessment and adaptation based on these metrics empower Agile teams to consistently improve and deliver exceptional value. Therefore, by systematically leveraging Agile metrics, teams can continuously refine their workflows, mitigate risks, and adapt to changing priorities, ultimately delivering higher-value products with greater predictability and efficiency. Through accurate measurement, informed decision-making, and consistent feedback loops, Agile teams create a framework that encourages collaboration, innovation, and sustained progress toward achieving project goals and objectives. Project performance evaluation metrics are required in all the stages of System Development Life cycle, and all efforts should be towards having software projects that meet stakeholders' satisfaction hence further future works are encouraged to further improve the success rate of software project development.

➢ *Definition of Terms:*

- Agile – A framework that concentrates on client value, refined delivery, small team, self-enterprise and continuous enhancements.
- Automated test coverage – Highlights the extent of automated testing for features, boosting reliability and reducing manual testing efforts.
- Control charts – Measure the cycle time of work in progress to done, providing information on team's efficiency and consistency in delivering work.
- Cumulative flow diagrams – Provides a visual representation of work status across different stages, using

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb900

different color bands to represent stages like “To Do”, In Progress”, and “Done”, a smooth flow indicate stable progress.

- Cycle time – Software project application development time.
- Deferred defects – Measures defects delayed to a future release, reflecting team prioritization.
- Defect counts – Tracks defects found during development, after release, or by external parties.
- Project performance – Measures the project health meticulously and ensure project success and client satisfaction, with the aim of aligning planned effort, time, and budget constraints.
- Project success – This is much about success measurement both short-term and long-term, short-term assess implementation within schedule, cost, quality and scope while long-term assess the effects of the project results.
- Sprint burndown – Tracks the completion of work throughout a sprint
- Velocity – The average amount of work a team completes during a sprint, measured in story points or hours

REFERENCES

[1]. S. Balaji and M. S. Murugaiyan, “Waterfall vs. Agile: a comparative study on SDLC,” *International Journal of Information Technology and Business Management,* vol. 2, no. 1, pp. 26-30, 2012

[2]. H. Hajjdiab and A. S. Al Shaima Taleb, “Adopting agile software development: issues and challenges,” *International Journal of Managing Value and Supply Chains,* vol. 2, no. 3, pp. 1-10, 2011

[3]. H. Berger and P. Beynon-Davies, “The utility of rapid application development in large-scale, complex projects,” *Information Systems Journal,* vol. 19, no. 6, pp. 549 -570, 2009.

[4]. B. Hobbs and Y. Petit, “Agile methods on large projects in large organizations,” *Project Management Journal,* vol. 48, no. 3, pp. 3-19, 2017.

[5]. S. Alam, S. A. A. Shah, S. N. Bhatti, and A. M. Jadi, “Impact and challenges of requirement engineering in agile methodologies: A systematic review,” *International Journal of Advanced Computer Science and Applications,* vol. 8, 2017.

[6]. A. Moniruzzaman and D. S. A. Hossain, “Comparative study on agile software development methodologies,” 2013, http://arxiv.org/abs/13073356.

[7]. R. Knaster and D. Leffingwell*, SAFe 4.0 Distilled: Applying the Scaled Agile Framework for Lean Software and Systems Engineering,* Addison-Wesley Professional, 2017.

[8]. A. Putta, M. Paasivaara, and C. Lassenius, “Benefits and challenges of adopting the scaled agile framework (SAFe): preliminary results from a multivocal literature review,” in *International Conference on Product-Focussed Software Process Improvement,* pp. 334-351, Springer, Berlin, Germany, 2018.

[9]. T. Hafterson, “Incorporating agile methods into the development of large-scale systems,” *In Proceedings of the UMM CSsci Senior Conference,* Moris, MN, USA, July 2011.

[10]. K. Beck, M. Beedle, A. Van Bennekum et al. *Manifesto for Agile Software Development,* Agile Manifesto, 2001.

[11]. J. Highsmith and A. Cockburn, “Agile software development: the business of innovation,” *Computer,* vol. 34, no. 9, pp. 120 – 127, 2001.

[12]. M. U. Malik, N. M. Chaudhry, and K. S. Malik, “Evaluation of efficient requirement engineering techniques in agile software development,” *International Journal of Computer Applications,* vol. 83, no. 3, 2013.

[13]. T. Jokela and P. Abrahamsson, “Usability assessment of an extreme programming project: close co-operation with the customer does not equal to good usability,” in *Product Focussed Software Process Improvement,* pp. 393 – 407, Springer, Berlin, Germany, 2011.

[14]. T. Dingsoyr, T. E. Faegri, and J. Itkonen, “What is large in large-scale? A taxonomy of scale for agile software development,” in *Proceedings of the International Conference on Product-Focussed Software Process Improvement,* pp. 273 – 276, Springer, Rovaniemi, Finland, December 2014.

[15]. M. F. Abrar, M. S. Khan, S Ali et al., “Motivators for large-scale agile adoption from management perspective: a systematic literature review,” *IEEE Access,* vol. 7, pp. 22660 – 22674, 2019.

[16]. B. Kovitz, “Hidden skills that support phased and agile requirements engineering,” *Requirements Engineering,* vol. 8, no. 2, pp. 135 – 141, 2003.

[17]. L. Cao and B. Ramesh, “Agile requirements engineering practices: an empirical study,” *IEEE Software,* vol. 25, no. 1, pp. 60 – 67, 2008.

[18]. M. Amir, K. Khan, A. Khan, and M. Khan,” An appraisal of agile software development process,” *International Journal of Advanced Science & Technology,* vol. 58, no. 56, p. 20, 2013.

[19]. F. Paetsch, A. Eberlein, and F. Maurer, “Requirements engineering and agile software development,” in *Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises,* pp. 308 – 313, IEEE, Washington, DC, USA, June 2003

[20]. N. Ganesh and S. Thangasamy, “Issues identified in the software process due to barriers found during eliciting requirements on agile software projects: insights from India,” *International Journal of Computer Applications,* vol. 16, no. 5, pp. 7 – 12, 2011.

[21]. J. Cho, “Issues and Challenges of agile software development with SCRUM, “*Issues in Information Systems,* vol. 9, no. 2, pp. 188 – 195, 2008.

[22]. W. Helmy, A. Kamel, and O. Hegazy, “Requirements engineering methodology in agile environment,” *International Journal of Computer Science Issues (IJCSI),* vol. 9, no. 5, p. 293, 2012.

[23]. M. S. Rahim, A. E. Chowdhury, and S. Das, “Rize: a proposed requirements prioritization technique for agile development,” in *Proceedings of the 2017 IEEE*

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb900

Region 10 Humanitarian Technology Conference (R10-HTC), pp. 634-637, IEEE, Bangladesh, Asia, December 2017.

[24]. D. Mishra and A. Mishra, "Complex software project development: agile methods adoption," *Journal of Software Maintenance and Evolution: Research and Practice,* vol. 23, no. 8, pp. 549-564, 2011.

[25]. E. Bjarnason, K. Wnuk, and B. Regnell, "A case study on benefits and side effects of agile practices in large-scale requirements engineering," in *Proceedings of the 1st Workshop on Agile Requirements Engineering,* pp. 1-5, Lancaster, UK, June 2011.

[26]. G. Lee, W. DeLone, and J. A. Espinosa, "Ambidextrous coping strategies in globally distributed software development projects," *Communications of the ACM,* vol. 49, no. 10, pp. 35-40, 2006.

[27]. N. A. Ernst, A. Borgida, I. J. Jureta, and J. Mylopoulos," Agile requirements engineering via paraconsistent reasoning," *Information Systems,* vol. 43, pp. 100-116, 2014.

[28]. T. Kamal, Q. Zhang, M. A. Akbar, M. Shafiq, A. Gumaei, and A. Alsanad, "Identification and prioritization of agile requirements change management success factors in the domain of global software development," *IEEE Access,* vol. 8, pp. 44714-44726, 2020.

[29]. K. AbdElazim, R. Moawad, and E. Elfakharany, "A framework for requirements prioritization process in agile software development," *Journal of Physics,* vol. 1454, 2020.

[30]. W. M. Farid and F. J. Mitropoulos, "Visualization and scheduling of non-functional requirements for agile processes," in *Proceedings of the 2013 IEEE Southeastcon,* pp. 1-8, IEEE, Jacksonville, FL, USA, April 2013.

[31]. M. Kapyaho and M. Kauppinen, "Agile requirements engineering with prototyping: a case study," in *Proceedings of the 2015 IEEE 23rd International Requirements Engineering Conference (RE),* pp. 334-343, IEEE, Ottawa, ON, USA, August 2015.

[32]. F. A. Mir and D. Rezania, "From interactive control to IT project performance: examining the mediating role of stakeholder analysis effectiveness," *Journal of Accounting & organizational Change,* 2021. Doi:10.1108/JAOC-04-2021-0048

[33]. C. Snyder, Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces, Morgan Kaufmann, Burlington, MA, USA, 2003.

[34]. R. McAdam, B. Mason, and J. McCrory, "Exploring the dichotomies within the tacit knowledge literature towards a process of tacit knowing in organizations," *Journal of Knowledge Management,* 2007.

[35]. S. Ryan and R. V. O'Çonnor, "Acquiring and sharing tacit knowledge in software development teams: an empirical study," *Information and Software Technology,* vol.55, no. 9, pp.1614-1624, 2013.

[36]. A. R. Asghar, S. N. Bhatti, A. Tabassum, and S. Shah, "The impact of analytical assessment of requirements prioritization models: an empirical study, "*International Journal of Advanced Computer Science and Applications (IJACSA)* vol. 8, no. 2, pp. 303-313, 2017.

[37]. D. Turk, R. France, and B. Rumpe, "Limitations of agile software processes," 2014, http://arxiv.org/abs/14096600.

[38]. P. E. D. Love, G. D. Holt, and H. Li, "Stakeholder perspective measurement (SPM) framework for construction projects, "*Construction Management and Economics,* vol.18, no.3, pp. 321-331, 2002.

[39]. G. Sawarkar and D. Rajput, "Comparative Analysis of Various Software Development Life Cycle," *International Journal of Computer Science and Mobile Computing,* vol. 11, no. 8, pp. 1-8, 2022. Doi: 10.47760/ijcsmc. 2022.v11i08.001

[40]. L. Zhang, C. Baron, P. Esteban, R. Xue, Q. Zhang and S. Yang, "Using leading Indicators to improve project performance measurement*," Journal of Systems Science and Systems Engineering,* vol. 28, no. 5, pp. 529 – 554, 2019.

[41]. S. A. Devaux, "Managing Projects as Investments: Earned Value to Business Value," CRC Press, 2014.

[42]. M. K. Sharma, "A study of SDLC to develop well engineered software," *International Journal of Advanced Research in Computer Science,*2017. doi: 10.26483/IJARCS.V813.3045

[43]. A. Garg, R. K. Kaliyar and A. Goswami, "PDRSD- A systematic review on plan-driven SDLC models for Software Development," 2022. Doi: 10.1109/ICACCS541555559.2022.9785261.

[44]. S. Gupta, J. Banga, S. Dabas and M. K. Bhatia, "A Comprehensive Study of Software Development Lifecycle Models," *International Journal for Science Technology and Engineering,* 2022. Doi: 10.22214/ijraset.2022.47868.

[45]. A. A. Amaefule and F. N. Ogwueleka," Criteria for choosing the Right Software Development Life Cycle Method for the Success of Software Projects," *IUP Journal of Information Technology,* vol. 16, no. 2, pp. 40-65, 2020.

[46]. O. Gun, P. Y. Kumru and Z. Aladag," Developing a model for Measuring Project Performance with Software Life Cycle Process Metrics and Calculating Project Success Score," *Sakarya University Journal of Science,* vol. 24, no. 3, pp. 536-554, 2020.

[47]. P. J. Haried and C. C. Claybaugh," Evaluating Information System Offshore Project Success: Can Success and Failure Co-exist," *Journal of Global Information Technology Management,* vol. 20, no. 1, pp. 8 – 27, 2017.

[48]. J. Pereiva, J. Varajao and N. Takagi, "Evaluation of Information Systems Project Success – Insights from Practitioners, *Information Systems Management,* vol. 39, no. 2, pp. 138-155, 2022. doi: 10.1080/10580530.2021.1887982.

[49]. A. Al-Shaaby and A. Ahmed, "How Do We measure Project Success? A Survey, *Journal of Information Technology & Software Engineering,* vol. 8, no. 2, 2018. Doi: 10.4172/2175-7866.1000229

[50]. L. ITai and S. Avraham, "Measuring the Success of Lean and Agile Projects: Are cost, time, scope and quality equally important? *Journal of Modern Project*

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb900

Management, vol. 7, no. 1, 2019. Doi: 10.19255/JMPM01908.

[51]. C. C. Villazon, L. S. Pinilla, J. R. O. Olaso, N. T. Gandaria and N. Lopez de Lacalle, "Identification of Key performance Indicators in Project-based Organization through Lean Approach, 2020

[52]. K. Chopra and M. Sachdeva, "Evaluation of Software Metrics for Software Projects, " 2015 or 2018 doi: 10.1109/DSMP.2018.8478464

[53]. V. Ostakhov, N. Artykulna and V. Morozov, "Models of IT Projects KPIs and Metrics," *International Conference on Data Stream Mining & Processing,* 2018

[54]. L. McLeod and S. G. MacDonell, "Factors that affect software systems development project outcomes: a survey of research, *ACM Computing Surveys,* vol. 43, no. 4, pp. 24-56, 2011. Doi: 10.1145/1978802.1978803.

[55]. S. Alsulamy, S. Wamuziri and M. Taylor, "Evaluation of key metrics for measurement of project performance *In:* Smith, S. D (Ed) Procs 28th *Annual ARCOM Conference,* 3-5 September 2012, Edinburgh, UK, Association of Researchers in Contruction Management, 1101 – 1110.

[56]. H. Aminu and F. N. Ogwueleka, "A Comparative Study of System Development Life Cycle Models, *Journal of Emerging Technologies and Innovative Research (JETIR),* vol. 7, no. 8, 2020.