IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 12, Issue 8, August 2023

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.423

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

|e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 8, August 2023 ||

| DOI:10.15680/IJIRSET.2023.1208077 |

Software Engineering: New Methodologies, Tools, and Best Practices in Software Development

Twinkle Bhowmick[1], Suraj Koner[2], Biraj Saha[3] , Ms. Debosree Ghosh[4] , Mr. Bablu Pramanik[5]

Student, Department of Computer Science and Technology, Swami Vivekananda Institute of Science and Technology, West Bengal, India[1,2,3]

Assistant Professor, Department of Computer Science and Technology, Shree Ramkrishna Institute of Science and Technology, West Bengal, India[4]

Assistant Professor, Department of Computer Application and Science, Swami Vivekananda Institute of Modern Science, West Bengal, India[5]

ABSTRACT *:* Software engineering is a dynamic field that is always evolving to meet the demands of difficult software development projects. This journal article investigates contemporary software engineering methodologies, tools, and best practices in order to enhance the efficiency, quality, and dependability of software development processes. We discuss the most recent advancements in quality assurance, DevOps, agile software development, and automated testing. We also look at the role that cooperation, communication, and project management have in efficient software engineering methods. By highlighting these advances, this article aims to promote the delivery of high-quality software products and improve software development processes.

KEYWORDS *:* Agile Software Development , Iterative and Incremental Practices, Agile Software Management and Collaboration Techniques, DevOps and Continuous Delivery Practices , Integration of Development and Operations Teams, Continuous Integration, Continuous Testing and Continuous Deployment Practices, Automated Software Testing and Quality Assurance, Methodologies: Test-Driven Development and Behaviour-Driven Development, Test Automation Frameworks and Tools, Code Reviews and Static Code Analysis, Continuous Testing and Quality Metrics.

I. INTRODUCTION

Software engineering has undergone incredible development and evolution over time due to the demands of a rapidly changing technical environment and the increasing complexity of software systems. To successfully manage the opportunities and challenges presented by this changing environment, software developers and organizations must adopt new methodologies, tools, and best practices. In order to give a thorough overview of the most recent developments in software engineering, this paper will concentrate on new methodologies, cutting-edge tools, and tried-and-true best practices that are reshaping the industry and altering the software development process.

Background :

Software engineering fundamentals are applied to provide dependable, scalable, and superior software solutions. In the past, software development followed the Waterfall paradigm, which prioritized planning, requirements gathering, design, coding, testing, and deployment. As software systems became more complex and market demands changed, traditional strategies, on the other hand, proved to be less effective in meeting customer expectations, maintaining flexibility, and delivering value quickly.

|e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 8, August 2023 ||

| DOI:10.15680/IJIRSET.2023.1208077 |

The Need for Innovation :

Software engineering must adopt new approaches, tools, and best practices due to the ever-increasing complexity of software systems and the emergence of disruptive technologies like blockchain, cloud computing, and artificial intelligence. Scrum and Kanban are two popular agile approaches that place a strong emphasis on iterative development, teamwork, and adaptability. Additionally, the emergence of DevOps has made it possible for businesses to close the gap between development and operations, fostering quicker delivery, increased dependability, and improved teamwork.

Emerging Methodologies :

This paper will analyze the most recent software development approaches that have found success. It will look at the core principles, benefits, and challenges of methods including Agile, DevOps, automated testing, and quality assurance. By implementing these strategies, organizations can enhance software quality, guarantee a shorter time to market, encourage customer collaboration, and boost team productivity.

Innovative Tools and Technologies :

The creation of a wide range of tools and technologies that aid different stages of the software development life cycle has been facilitated by advancements in software engineering. In order to demonstrate how these tools improve workflows, automate tedious processes, and facilitate effective team collaboration within development teams, this article will showcase the features, capabilities, and advantages of these tools.

Best Practices for Software Development :

This paper will explore new approaches and technologies while highlighting the value of using best practices in software development. It will look at case studies from the industry, empirical research, and expert opinions to find tried-and-true techniques that improve software engineering results. There will be discussions on issues including code reviews, automated testing, version control, documentation, and continuous improvement. Organizations may improve software quality, lower technical debt, and promote a culture of innovation and excellence by integrating these best practices into their operations.

Objective and Scope of the Paper :

The goal of this paper is to give software engineers, researchers, and business experts a thorough grasp of the most recent approaches, resources, and industry best practices. We aim to provide readers with the information and insights required to drive successful software development projects by providing a thorough overview of these developments. The paper will discuss a variety of subjects, such as Agile approaches, DevOps practices, cutting-edge tools, and efficient software engineering methods.

II. AGILE SOFTWARE DEVELOPMENT

Agile software development stresses the delivery of high-quality software in a flexible method while simultaneously assuring customer satisfaction. It is an iterative and collaborative approach. It significantly contributes to the advancements in software development. It accepts responsibility for change and places a focus on the prompt and reliable supply of useful software. In order to create high-quality software that also closely meets the goals and expectations of the client, agile software development closely collaborates with the customer and the software development team.

Steps of Agile Software Development Process :

i. Requirements gathering: Collecting customer requirements, a crucial phase in the software development process. Any misleading information could render the software unusable for the client.

|e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 8, August 2023 ||

| DOI:10.15680/IJIRSET.2023.1208077 |

ii. Planning: All preparations for providing software to the customer were made by the software development team. This process also covers the planning of software development, testing, and software feature development.
iii. Development: The software is created throughout this procedure for the client's use. In this step, the software is developed through a frequent and quick iteration process.
iv. Testing: The software development team in this procedure meticulously verifies that the product satisfies all standards and specifications set forth by the client.
v. Deployment: The software development team deploys the software so that it can be used.
vi. Maintenance: It is both the most crucial and expensive step in the Agile Software Development process. The maintenance process is where the development team makes sure the program is in excellent working order and meets customer requirements.

Key Principles of Agile Development Process :
i. Customer collaboration: Agile software development teams prioritize getting to know their clients well, comprehending their needs for the program, and soliciting client input throughout the software development process.
ii. Individual and interaction over processes: Agile software development emphasizes the benefits of effective teamwork and communication in order to maximize team members' abilities and expertise.
iii. Comprehensive documentation: The delivery of functional software that meets the needs of the client is given precedence by the software development team over comprehensive documentation.
iv. Responds to change: Agile software development teams are able to understand how the software's requirements and priorities change over time and are able to adapt and react to such changes in a flexible way.

Values of Agile Software Development :
i. People and interactions over processes and tools: This process highlights the significance of excellent team member collaboration and communication.
ii. Delivering functional software: It satisfies user demands rather than producing voluminous documentation is the main goal.
iii. Customer cooperation takes precedence over contract negotiation: It emphasizes active customer participation and input throughout the development process.
iv. Adjusting to change instead of sticking to a plan: Being adaptable to changing demands or priorities rather than strictly adhering to predetermined plans.

Iterative and Incremental Practices :

Iterative development:
It entails continually making minor revisions to the software to refine and enhance it. Each iteration normally lasts for a predetermined amount of time (for instance, two weeks) and produces a working product increment. After each iteration, teams collect user and stakeholder feedback to inform future development.

Incremental development:
It entails developing the software in stages, with each stage adding new capability or improving current features. With this strategy, the team can quickly create a viable product and gradually add more features.

Agile Project Management and Collaboration Techniques :
Agile collaboration and project management methods are crucial for effective software development in this setting. Some methods include:-
i. Scrum: A common Agile framework that divides work into standardized time periods known as "sprints." To encourage cooperation and track progress, it incorporates rituals like Sprint Planning, Daily Standups, Sprint Review, and Sprint Retrospective.

|e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 8, August 2023 ||

| DOI:10.15680/IJIRSET.2023.1208077 |

ii. Kanban: A method of visual management that makes use of a board with columns for the various stages of work. Teams can monitor the progression of jobs and continuously improve their workflow thanks to it.
iii. User Stories: A method for communicating needs through the eyes of the user. These succinct, straightforward statements aid in keeping the emphasis on providing value to end users.
iv. Cross-functional Teams: Developers, testers, and designers are all included in agile teams to ensure that they can manage all project-related tasks and foster teamwork.
v. Product Backlog: This is a prioritized list of the features, upgrades, and problem fixes that need to be added to the product. Throughout the process, it changes.
vi. Pair programming: It is an approach where two engineers collaborate on the same piece of code to improve cooperation, knowledge exchange, and code quality.

Advantages of Agile Software Development :
i. Flexibility and Adaptability: Agile software development permits modification and revision of software requirements, scope, and overall development process priorities. This enables the development team to adapt to changing client and market demands.
ii. Customer satisfaction: Agile software development works closely with customers and stakeholders to guarantee that they are continuously collaborating with the development team while the software is being developed and provided to customers, greatly increasing customer satisfaction.
iii. Improved quality: Using test-driven development and continuous integration, the agile software development method can produce software of higher quality.
iv. Faster time to market: The agile software development technique permits the release of functional software in shorter cycles than conventional methods.

Disadvantages of Agile Software Development :
i. Lack of predictability: Agile software development might be so distinct from others due to its adaptable and flexible nature. However, it could be a problem if a software has a set budget and timeframe.
ii. Client dependence: Agile software development is heavily reliant on the availability and input of the client. Therefore, all procedures can be suspended for a period of time if a customer is unavailable for a certain amount of time.
iii. Team experience and skill requirements: Agile software development demands a well-educated software development team. If the team does not adopt new capabilities, it will be challenging to remain competitive.
iv. Lack of structure: It can be a weakness of that team as Agile software development is typically less formal and structured than others in the current industry.

III. DEVOPS AND CONTINUOUS DELIVERY PRACTICES

Two key ideas in software engineering, DevOps and Continuous Delivery, concentrate on enhancing teamwork, effectiveness, and quality across the software development lifecycle. Organizations can achieve a quicker time to market, better software quality, enhanced collaboration, and more dependable and effective software delivery processes by implementing DevOps and Continuous Delivery methods.

DevOps :
Software development teams (Dev) and IT operations teams (Ops) are encouraged to work together, communicate, and integrate using the DevOps cultural and organizational approach. DevOps' primary objective is to provide quicker and more frequent software delivery while maintaining security, dependability, and stability. It entails dismantling team silos, automating procedures, and putting continuous feedback loops in place to allow for continual development.

|e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 8, August 2023 ||

| DOI:10.15680/IJIRSET.2023.1208077 |

Key aspects of DevOps :

i. Collaboration: Encouraging efficient dialogue and teamwork between development, operations, and other stakeholders.

ii. Automation: Automating manual procedures and operations to increase effectiveness and decrease mistakes.

iii. Continuous Integration(CI): Including code updates often to find and address problems.

iv. Continuous Deployment (CD): Automating the deployment procedure will enable prompt and reliable delivery of software updates.

v. Continuous Monitoring: Putting in place feedback and monitoring methods to learn more about the software's stability and performance.

Continuous Delivery :

The DevOps strategy is expanded upon by continuous delivery, which aims to make it possible for software to be sent to production quickly and reliably. Software should always be in a state that is ready for deployment, enabling teams to quickly add new features, patch bugs, and make other enhancements.

Key aspects of Continuous Delivery :

i. Automated Testing: To assure software quality, a full set of automated tests should be implemented.

ii. Deployment Automation: Automating the deployment procedure to reliably and consistently release software modifications.

iii. Version Control: Using version control systems to manage code repositories and keep track of changes.

iv.Configuration Management: Managing application configurations consistently and automatically is known as configuration management.

v. Continuous Integration: Frequently integrating code updates to find and fix integration issues.

Integration of Development and Operations Teams :

The term "DevOps" refers to the integration of the development and operations teams. To enhance the software development and deployment processes, it entails encouraging collaboration, communication, and shared responsibility across these teams. Organizations that use DevOps principles may produce software more quickly, more effectively, and with higher-quality results thanks to automated workflows and ongoing feedback loops.

Continuous Integration , Continuous Testing and Continuous Deployment Practices :

Continuous Integration (CI):

CI requires regularly automating the merging of code updates from several developers into a single common repository. Early detection and resolution of integration problems are desired in order to maintain consistency and stability of the codebase. Every time new code is added, automated builds and tests are initiated, assisting teams in finding errors promptly and maintaining a dependable development process.

Continuous Testing (CT):

CT ensures that the CI/CD pipeline includes automated tests as a necessary component. Throughout the development process, automated tests are run continually to verify the functionality, performance, and security of the application. Teams can quickly find problems and make the necessary corrections before deploying the code to production by automatically performing tests.

Continuous Deployment (CD):

CD seeks to automate the deployment process, enabling frequent and reliable release of changes to production. Code can be automatically pushed to the production environment after passing through CI and CT. This method provides quick and continuous delivery of new features and bug fixes while minimizing manual intervention and reducing the time between code changes and deployment.The development and operations teams may work more productively,

|e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 8, August 2023 ||

| DOI:10.15680/IJIRSET.2023.1208077 |

achieve shorter release cycles, and preserve a high degree of software quality throughout the development and deployment processes by putting these strategies into effect.

IV. AUTOMATED TESTING AND QUALITY ASSURANCE

The overall dependability and efficiency of software systems are enhanced through automated testing and quality assurance, which are essential elements of software engineering. They play a crucial role in software engineering by ensuring that software systems adhere to the specified quality standards.

Automated Testing :
Automated testing is the process of running tests and validating the behavior and efficiency of software operations utilizing technical software tools and technologies.It enables the design of scripts and test cases that can be efficiently and consistently executed over and over again. This strategy ensures complete test coverage while saving time and effort. Software engineering employs a variety of automated testing methods, such as unit testing, integration testing, functional testing, performance testing, and regression testing. To find flaws, faults, and inconsistencies early in the development process, each technique focuses on a different component of the product. Software developers are able to find bugs, enhance the quality of their code, and provide end users with dependable and strong software applications by performing tests in a systematic and repetitive manner.

Quality Assurance :
Software must adhere to established quality standards and user needs, which are covered by quality assurance procedures and activities. It entails taking a methodical and proactive approach to flaw prevention and product quality enhancement. Not only testing but also requirement analysis, design reviews, code reviews, and documentation verification are included in it. Increased productivity, shortened time to market, better software quality, and higher customer satisfaction are all advantages of quality assurance.

Methodologies :
Test-Driven Development and Behaviour-Driven Development are the two methods. As a result, developers can identify errors and design problems earlier in the development lifecycle because of their encouragement of early and continuous testing.

i. Test-Driven Development :
In test-driven development (TDD), tests are written by developers prior to creating the real code. Three steps make up the TDD cycle: "red," "green," and "refactor." Developers create a test that initially fails in the "red" phase to draw attention to a particular behavior or capability that has to be added. Developers create the bare minimum of code necessary to pass the test during the "green" phase. The code is evaluated and refactored to increase readability, maintainability, and efficiency during the final "refactor" phase. By dividing the program requirements into manageable, testable chunks known as units or modules, it encourages a granular and progressive development approach. The emphasis is on creating unit tests that confirm how these isolated components behave.By adhering to TDD, developers may be sure that their code is proper and have a set of tests that can be run automatically anytime changes are made.

ii. Behavior-Driven Development :
By providing a common vocabulary for discussing and specifying software requirements and behavior, behavior-driven development (BDD), an extension of test-driven development (TDD), strives to close the communication gap between developers, testers, and stakeholders. "User stories" or "scenarios" are a concept introduced by BDD that describe the expected behavior of the product from the viewpoint of multiple stakeholders. These user stories are created in plain language that both technical and non-technical team members may understand.A structured framework called "Given-

|e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 8, August 2023 ||

| DOI:10.15680/IJIRSET.2023.1208077 |

When-Then" (GWT), commonly referred to as "Arrange-Act-Assert" (AAA), is used by BDD to express tests. Every scenario typically consists of a set of steps that describe the starting situation ("Given"), the action or event that is taking place ("When"), and the anticipated result or behavior ("Then"). Tools and libraries are provided by BDD frameworks to parse these scenarios and produce executable tests. It promotes teamwork and mutual understanding among participants. The emphasis is on ensuring that the overall system behavior conforms to the expectations of the stakeholders as well as confirming the functionality of particular system components. The focus of BDD tests is more on system integration and end-to-end functioning, and they are frequently written at a higher level of abstraction, frequently spanning many units or components.

Test Automation Frameworks and Tools :

Tools and frameworks for test automation are essential for automated testing and quality control in software engineering. They offer the tools, procedures, and infrastructure required to manage test cases, automate test execution, and produce reports.

A. Test Automation Frameworks :

Frameworks for test automation offer a standardized method for creating and running automated tests. They lay up the overall structure as well as rules for arranging and managing test scripts and test cases. Several well-liked test automation frameworks include:-

i. Keyword-driven frameworks: Tests are constructed using keywords and related activities, enabling for the creation of modular and reusable test components.

ii. Data-driven frameworks: Tests are created to work with various data sources, allowing for thorough testing and validation.

iii. Modular or library-based frameworks: Frameworks that are modular or library-based allow for the division of tests into smaller, reusable modules or libraries, which makes them easier to maintain and reuse.

iv. Behavior-driven development (BDD) frameworks: Facilitate communication between technical and non-technical stakeholders by offering a standard language for designing and implementing documents.

v. Hybrid frameworks: Combine several frameworks to take use of their own capabilities and meet the demands of a given project.

B. Test Automation Tools :

Software programs or libraries that help automate the testing process are known as test automation tools. They offer tools for creating, running, and reporting test scripts.

i. Selenium: It is a well-known open-source tool for testing web applications and is one of the most widely used test automation tools. It supports a variety of browsers and computer languages and offers a large selection of features for dealing with web elements.

ii. Appium: A free software program for testing mobile applications. It enables testing of mobile applications utilizing native and web-based testing techniques on a variety of platforms, including Android and iOS.

iii. JUnit : It is a framework for unit testing Java-based applications. For the purpose of creating and running unit tests, it offers annotations, assertions, and test runners.

iv. TestNG: It is an improved functional testing framework for Java-based apps.

Code Reviews and Static Code Analysis :

Static code analysis and code reviews are crucial procedures in automated testing and quality control that target assuring code quality, finding errors, and enhancing overall software reliability. By locating problems early in the development lifecycle, they aid in enhancing code quality and lowering faults.

|e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 8, August 2023 ||

| DOI:10.15680/IJIRSET.2023.1208077 |

Code Reviews :

One or more team members do systematic code reviews of the source code to find errors, enhance code quality, and make sure coding standards and best practices are being followed. Early on in the development phase, it aids in spotting flaws and issues.

lowering the cost and effort required to remedy problems later and enhancing the readability, maintainability, and by adhering to best practices, design patterns, and code standards.Team knowledge sharing and collaboration are encouraged. It guarantees adherence to team rules. Members and encouraging teamwork. Consequently, it confirms that the code implementation adheres to the project parameters and needs.

Static code Analysis :

The goal of static code analysis, an automated technique that examines source code without running it, is to find potential flaws, vulnerabilities, and compliance with coding standards. In order to find problems such as coding style violations, potential bugs, security vulnerabilities, performance bottlenecks, and adherence to coding standards, static code analysis tools scan code at the syntactic and semantic level. It assists in both locating common programming problems, such as race situations, memory leaks, buffer overflows, and null pointer dereferences, as well as security flaws, such SQL injection, cross-site scripting (XSS), and authentication flaws.In addition to upholding architectural and design patterns, it makes sure that coding standards, naming conventions, and best practices are followed. It also evaluates metrics for code complexity and maintainability. Through continuous integration (CI) and build systems, where code is automatically examined upon each code commit or build, static code analysis can be incorporated into the development process. SonarQube, PMD, Checkstyle, ESLint, and FindBugs are a few of the frequently used static code analysis tools. These tools each cater to certain programming languages and offer a variety of analysis criteria and customization possibilities.

Continuous Testing and Quality Metrics :

Important components of automated testing and quality assurance in software engineering are continuous testing and quality metrics. They aid in the quicker and more dependable delivery of software and assist in the ongoing improvement of software quality. Together, they both work to advance software quality.

Continuous Testing:

It is a methodology that continually runs automated tests throughout the course of the software development lifecycle to provide quick feedback on the software's reliability and quality.It entails incorporating testing tasks into the pipeline for continuous integration and delivery (CI/CD), where tests are launched automatically in response to code updates or deployments.It makes sure that errors and regressions are found early, allowing for quick correction and lowering the chance that problems will surface in production. As tests are constantly run and provide instant feedback, it enables developers to make essential improvements as soon as possible, allowing for quicker development iterations. It involves a variety of automated tests that are run automatically and repeatedly, including unit tests, integration tests, functional tests, performance tests, and security tests.

Quality Metrics :

Software quality is evaluated using quality metrics, which are quantitative indicators that offer information on the software's performance, dependability, maintainability, and other characteristics.

Quality metrics are useful for assessing the efficiency of testing and quality assurance procedures, pinpointing problem areas, and monitoring the advancement of quality objectives. Dashboards and reports can be used to track it and display it, giving stakeholders the ability to keep an eye on it and make data-driven decisions about the quality of their program. Additionally, it can be used to develop benchmarks, set quality goals, and assess how well various software releases perform.

|e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 8, August 2023 ||

| DOI:10.15680/IJIRSET.2023.1208077 |

Typical quality metrics include:-

i. Defect density: The quantity of flaws found in a unit of code or testing.

ii. Test coverage: It is a percentage indicating how thoroughly tests have used the software.

iii. Mean Time Between Failures (MTBF): The average amount of time between software failures is known as the Mean Time Between Failures (MTBF).

iv. ***Mean Time to Failure (MTTF):*** The amount of time the software typically lasts before failing.

v. Code complexity: Metrics that measure the complexity and readability of the code, such as cyclomatic complexity, lines of code, and maintainability index.

vi. Test case effectiveness: Effectiveness of test cases is the proportion of test cases that find bugs or confirm functionality.

vii. Customer satisfaction: User or stakeholder comments on how happy they are with the software.

V. CONCLUSION

Software engineering continues to be a crucial field, driving innovation and changing industries all over the world in the rapidly evolving technological landscape. This essay has explored the world of modern software development processes, tools, and best practices, illuminating the dynamic character of this industry and its far-reaching consequences for the future. Agile Software Development, DevOps and Continuous Delivery Practices, as well as Automated Testing and Quality Assurance, are among the approaches covered in this article. The software development landscape has been transformed by the fusion of Agile approaches, adaptable technologies, and strict best practices, driving it toward increased productivity, dependability, and creativity. As a result, to keep up with the dynamic growth of software engineering, new approaches, tools, and best practices must be continuously explored and adopted. Additionally, this evolution must be supported by a firm commitment to ethical issues and societal effects. The future of software engineering will be shaped by our capacity to embrace change, collaborate with others, and adapt as we travel into technologically unexplored territory. We must also make sure that the software we develop is a force for good in the world.

REFERENCES

[1] Pressman, R. S. (2014). "Software Engineering: A Practitioner's Approach." McGraw-Hill Education.

[2] Cohn, M. (2004). "User Stories Applied: For Agile Software Development." Addison-Wesley Professional

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

[4] Bass, L., Clements, P., & Kazman, R. (2012). "Software Architecture in Practice." Addison-Wesley Professional.

[5] Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Kern, J. (2001). "Manifesto for Agile Software Development." Retrieved from http://agilemanifesto.org/

[6] Martin, R. C. (2008). "Clean Code: A Handbook of Agile Software Craftsmanship." Prentice Hall.

[7] McConnell, S. (2004). "Code Complete: A Practical Handbook of Software Construction." Microsoft Press.

[8] Brooks Jr, F. P. (1995). "The Mythical Man-Month: Essays on Software Engineering." Addison-Wesley Professional.

[9] Ambler, S. W. (2010). "Agile Modeling: Effective Practices for Extreme Programming and the Unified Process." John Wiley & Sons.

[10] Sommerville, I. (2010). "Software Engineering." Addison-Wesley.

[11] Larman, C., & Basili, V. R. (2003). "Iterative and Incremental Development: A Brief History." IEEE Computer, 36(6), 47-56.

[12]Praveen Kumar Maroju, "Optimizing Mortgage Loan Processing in Capital Markets: A Machine Learning Approach, " International Journal of Innovations in Scientific Engineering, 17(1), PP. 36-55 , April 2023.

[13] Brooks Jr, F. P. (1995). "No Silver Bullet: Essence and Accidents of Software Engineering." IEEE Computer, 20(4), 10-19.

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor: 8.423

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details