International Journal of Science and Research Archive

eISSN: 2582-8185

Cross Ref DOI: 10.30574/ijsra

Journal homepage: https://ijsra.net/

(REVIEW ARTICLE)

Check for updates

# Optimising Software Lifecycle Management through Predictive Maintenance: Insights and Best Practices

Abhishek Goyal *

*Independent Researcher.*

International Journal of Science and Research Archive, 2022, 07(02), 693–702

Publication history: Received on 18 November 2022; revised on 25 December 2022; accepted on 28 December 2022

Article DOI: https://doi.org/10.30574/ijsra.2022.7.2.0348

## Abstract

Predictive maintenance within SLM continues to grow as the revolutionary approach that optimises availability and enhances durability and performance of software systems while minimising the extent of downtime. This paper looks at the incorporation of predictive analysis in the SDLC mainly to forecast when software programs are likely to fail in an effort to minimise downtime. Using advanced technologies like machine learning, time series analysis, log mining, and automated testing, organisations can begin looking at ways to improve the ability to head off problems and improve the quality of software while decreasing maintenance costs. The paper focuses on correcting, adaptive, perfective, and preventive maintenance and explains the role of predictive maintenance in anticipating and preventing developing flaws. In addition, the advantages of adopting predictive maintenance in the software lifecycle are explained, which include safety, longer life span of assets, and a better fit with the keywords of Industry 4.0. The paper concludes with best practices for successfully incorporating predictive maintenance into SLM, emphasising data-driven decision-making, aligning maintenance strategies with business objectives, and ensuring continuous system optimisation.

**Keywords:** Software maintenance; Preventive maintenance; Software development lifecycle; Artificial intelligence; Machine learning

## 1. Introduction

The Software Development Lifecycle (SDLC) is a methodology for creating software that allows for detailed development, increases the likelihood of finishing the project on schedule, and ensures that the final result is consistent with standards. The SDLC is a methodology that software engineers and system designers use to create new software [1][2]. As the software industry continues to develop, an increasing number of software systems are reaching a point where maintenance is vital to the product's functionality [3]. The term "software maintenance" describes the steps used to update an operational system after it has been released and put into use. For hardware systems, the term "maintenance" means nothing more than returning the system to its factory settings[4][5]. Software systems, however, are always different from other systems. Software maintenance seeks to close the gap between the current functional system and evolving user requirements [6].

The first law of software evolution is that an operational software system changes continuously or loses usefulness over time [7][8]. The relevance of software maintenance and evolution is growing as a result of rising costs and slowing implementation speeds. Figure 1 shows the Software development lifecycle (SDLC).

* Corresponding author: Abhishek Goyal

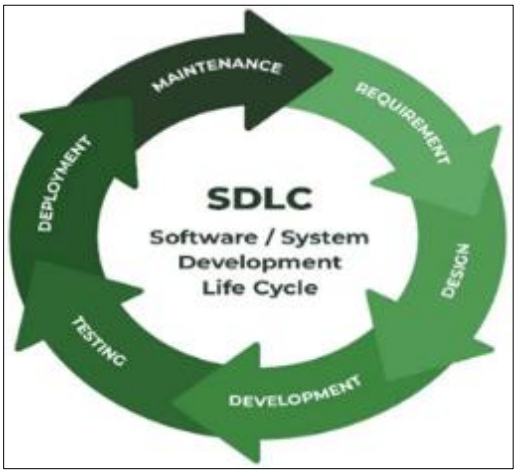


**Figure 1** SDLC

The development of the Process Reference Model for Predictive Maintenance is necessary to achieve the stated purpose. PReMMa offers practitioners thorough and detailed support as well as organised advice to help them implement predictive maintenance[9]. A PdM process model that takes into account the unique features and goals of PdM should be easily created using its reference viewpoint[10]. Using the existing theoretical literature as a foundation, the model is refined and then tested using real-world data gleaned from eleven in-depth interviews with industry professionals [11][12]. There are three tiers of granularity in the recognised process reference model's organisational structure [13]. The development of a phase model outlining general steps to be followed occurs at the highest level. Interrelated and interdependent fields, Systems Engineering and Project Management provide specific procedures for each of these stages at the second level[14][15]. Collectively, they have the potential to greatly aid in the creation of software-based solutions that fulfil the needs of businesses when carried out correctly [8].

### 1.1. Organization of the paper

The following paper is organised as follows: Section II provides an overview of software development lifecycle management, followed by Section III, which discusses predictive maintenance in software lifecycle management. Section IV explores current trends in predictive maintenance in software lifecycle management, highlighting advancements and emerging practices. Section V presents a literature review on this topic, and the paper concludes with future work that suggests potential directions for enhancing predictive maintenance in software development.

## 2. Overview of software development lifecycle management

Systematically completing software development within the allotted time while preserving software quality is the goal of SDLC. The SDLC (System Development Life Cycle) lays out the steps that need to be taken when creating a system; it is also known as the SDLC for software [16]. Any software development business can simply manage the software product since software development is organised into a series of activities. Models for the SDLC use a sequential strategy to finish the software development life cycle [17]. The project's success depends on the quality of the process, which in turn determines the quality of the finished result [18][16]. These models aid in the development of the software that developers are seeking. It is a comprehensive and diagrammatic representation of the software life cycle [19]. It encompasses everything that must be done to progress a software product through its life cycle stages.

### 2.1. Planning

It is at this phase of planning that the project's risks are identified, and the needs for quality assurance are considered. Various technical techniques might be defined to effectively perform the project with minimal risks, according to the technical feasibility assessment.

### 2.2. Defining

The next stage, after the completion of the requirement analysis, is to accurately describe and record the product needs, after which they must be approved by either the client or the market analyst.

### 2.3. Designing

All of the product's architectural modules, as well as their relationships with one another and with third-party modules, are defined in detail by a design methodology.

### 2.4. Building

Product development and construction begin at this phase of the SDLC. At this point, the code is created according to DDS.

### 2.5. Testing

The testing operations are often integrated into all phases of the SDLC in current models. Hence, this stage is typically a subset of all stages.

### 2.6. Maintenance

The product is officially launched to the relevant market after it has passed all necessary tests and is prepared for deployment. Deploying a product may occur in phases depending on the organisation's business plan.

#### *2.6.1. Benefits of software development lifecycle management*

Implementing a structured SDLC brings several advantages that enhance both the development process and the final product's quality. A formally defined SDLC helps establish a systematic approach to software development, which improves efficiency, clarity, and accountability across teams. Here are the key benefits[20]:

- Establishing a Common Vocabulary: Standard terms for each phase reduce misunderstandings, ensuring all stakeholders communicate clearly.
- Defining Communication Channels: Structured paths for updates and feedback keep everyone aligned, fostering transparency and minimising rework.
- Clarifying Roles and Responsibilities: Specific roles for developers, designers, analysts, and managers improve accountability and streamline contributions.
- Setting Clear Inputs and Outputs: Both phases are refined, have clear objectives, decrease uncertainty, and facilitate the transition from one phase to another.
- Providing a Deterministic “Definition of Done”: Business rules that guide each phase ensure that quality prevails, any issues that are left are not many, and the project will not deviate from course.

#### *2.6.2. Maintenance in Software Lifecycle Management (SLM)*

This is a key process in Software Lifecycle Management (SLM) as it is the way to ensure the software remains usable and capable of adjusting to six- new demands in order to deliver continual value to the user. There are four main types of maintenance, each serving a unique role:

- Corrective Maintenance: Some of the software that has had initial deployment may have errors that require correction during this kind of maintenance. Corrective maintenance is fixing bugs, errors and failures that may occur in the course of normal use so as to make the software to become stable and functional.
- Adaptive Maintenance: When an application's environment changes, such as an operating system, hardware, or external dependency, adaptive maintenance modifies the program to account for these changes. This category of maintenance makes the software compatible and responsive to new technological environments; the software runs in new or changing environments seamlessly.
- Perfective Maintenance: As a result of user feedback and increasing needs, perfective maintenance is designed to improve the functioning, interaction, and relevance of the software application. This might include enhancing already implemented functionalities, enhancing computational speed or enhancing the visible interface.
- Preventive Maintenance: The goals of preventative work are to fix possible future problems that may arise due to technical debt by making changes to the codebase. This may include code tidying up probably several times a day, algorithm optimisation improved security, etc.

## 3. Techniques of predictive maintenance in software lifecycle management

In predictive maintenance, changes are made to software after delivery in order to find and fix hidden bugs before they affect the product's functionality. The program's operational tasks, such as backup, recovery, and system administration

[21], are often handled by the people who run the software and thus do not address their concerns. Know that PM is a preventative strategy that targets software obsolescence and is not an active player in the program's efficient functioning. The goal of software PM is to decrease the occurrence of unanticipated maintenance tasks while simultaneously increasing the system's dependability and maintainability [10]. Prospective problem-solving, or PM, entails thinking forward when issues may arise. Essentially, the goal of doing preventative maintenance is to make software more maintainable by making necessary modifications.

Some of the aforementioned maintenance approaches share the characteristic of initiating a service operation in the event of an impending or actual hard failure; this makes last-minute resource scheduling more challenging, increases the probability of equipment damage, and increases the repair cost[22]. In order to reduce the likelihood of unanticipated failures and maximise the time available for proactive resource planning, advanced failure prediction algorithms have been developed [23][23]. It is challenging to implement a predictive maintenance system that is both practical and dependable. The development of sophisticated statistical models for the analysis of various sensor data streams and the reliable prediction of trends and outcomes requires an in-depth familiarity with the equipment's typical operating parameters and failure mechanisms. There are three main types of software maintenance: correction (fixing bugs), adaptation (adjusting to new operating systems and business rules, for example) and enhancement (adding new features and expanding software beyond its initial functional requirements). The term "preventive maintenance" (PM) refers to software reengineering that is necessary to address degradation caused by changes [24][25]. Key points of predictive maintenance technique are covered in the below points.

### 3.1. Techniques

In software lifecycle management, predictive maintenance techniques aim to anticipate potential issues before they disrupt software performance, reduce technical debt, and increase overall system reliability. Here are several key techniques for implementing PM in the software lifecycle:

#### *3.1.1. Machine Learning for Failure Prediction*

ML techniques like classification and regression models are highly effective in predicting software failures and maintenance needs [26].

- Classification Models: Models such as LR, DT, and RF can classify whether a component or service might fail based on historical failure data[15].
- Regression Models: Regression techniques, including Linear Regression and Gradient Boosting, help estimate the time until a component might need maintenance, providing a timeline for intervention.
- Anomaly Detection: Methods such as Isolation Forests, One-Class SVM, as well as K-means clusters are employed with the objective of identifying any performance irregularity in an application or any user activity that may be potentially indicative of emerging faults [27].

#### *3.1.2. Time Series Analysis for Performance Monitoring*

Time series models are ideal for monitoring trends of specified software indicators, for instance, CPU utilisation, memory consumption, or requested response times [28].

- ARIMA Models: Another type of model is Autoregressive Integrated Moving Average, which can be used to literally predict, based on the timeline, the patterns that relate to failures and indicate that the memory usage or CPU load is going to grow.
- Exponential Smoothing: Both basic and advanced smoothing models can foretell increases in metrics that suggest impending problems so teams can solve them beforehand.
- Fourier Transforms: The general characteristics of Fourier analysis allow uncovering of cyclic behaviours and seasonality of metrics, which may indicate regular maintenance requirements at certain times.

#### *3.1.3. Log Mining and Event Analysis*

Manual and automated logging, as well as events, give deep insight into the software behaviour, and mining these resources can find problem symptoms.

- Log Mining with Pattern Recognition: Through clustering, like in the case of Weka, logs that are similar yet precede a failure can be recognised and through association rule mining, the association between pre-failure patterns can be mined.

- Sequence Modeling: Among them, HMMs and LSTM networks are useful for finding groups of events that may cause an error or system unavailability to prevent them.
- Error Correlation Analysis: For dependability and interaction between errors occurring in different areas of the system, the correlation analysis of various error logs is conducted to determine the potential cascading failure.

*3.1.4. Complexity and Dependency Analysis*

It becomes crucial to bear in mind how elements depend on one another in a software codebase in terms of maintenance requirements [29].

- Complexity Metrics: Cyclomatic complexity, coupling, cohesion, and code churn metrics that are used will make it easy to identify the "hot areas" of the code that are most likely to require frequent updates.
- Technical Debt Monitoring: There are ways of weighting and measuring technical debt and when written code is created, there would be features that would highlight or classify areas that are likely to cause maintenance problems.
- Dependency Mapping: When dependencies between the modules are detected, the teams will be able to identify which parts can be influenced by changes in certain areas and prevent them.

*3.1.5. Automated Testing with Predictive Prioritization*

Testing is critical in assuring software quality; predictive prioritisation improves its execution.

- Test Case Prioritization: There is also a method in machine learning that can rank test cases in order of defect occurrences enabling the testers to direct testing at areas in software that have high propensity to exhibit certain defects.
- Predictive Quality Assessment: Models can then use such values to determine which portions of the code base will tend to exhibit a higher density of defects and, therefore, will require attention from developers in the near future.
- Fault Localization: Predictive methods also enable automated testing tools to locate the most likely reasons for the problem and, therefore, decrease debugging time and increase software quality.

### 3.2. Benefits of Implementing Predictive Maintenance software in an organisation

The following Benefits of Implementing Predictive Maintenance software in an organisation:

- Minimize Downtime: PM enables an organisation to identify possible equipment defects mechanically and fix them before they lead to an unnecessary halt. Such a proactive approach is aimed at the prevention of failure, which helps to maintain the condition of machinery so that breakdowns and the subsequent loss of revenues are minimised.
- Reduces Maintenance Cost: That's because PdM avoids reactive maintenance, saves labour costs, and reduces spare part consumptions and emergency repairs, thus contributing greatly to cutting costs comprehensively [30].
- Extended Asset Lifespan: Maintenance optimisation in Predictive Maintenance improves efficiency of products with minimal replacement of the equipment or renewal of the assets.
- Improved Safety: Maintenance optimisation in Predictive Maintenance improves efficiency of products with minimal replacement of the equipment or renewal of the assets [31].
- Enhanced Equipment Reliability: Thus, monitoring level and analysis of the equipment effectiveness allow achieving and sustaining higher level of reliability in the organisation. This reliability, in turn, brings about product quality and, therefore, customer satisfaction.
- Data-Driven Decision-Making: As mentioned above, PdM relies on data analysis to provide organisations with knowledge of the health and performance of their equipment. These are helpful when addressing issues to do with the maintenance schedules, organising resources and making capital investments [32].
- Adaptability to Industry 4.0: PdM fits well into the current and upcoming trends of technology, such as Industry 4.0 and the Internet of Things. The adoption of sensors and data analytics can go a step further in an organisation's maintenance plan to ensure it stays relevant and competitive.

## 4. Best practices for predictive maintenance in software lifecycle management

While businesses work as the application of organisational software lifecycle management, predictive maintenance simply becomes a strategic means to minimise the rate of seats, maximise effectiveness and durability of applications and systems. The following sections discuss best practices for implementing and sustaining a predictive maintenance strategy effectively [33]:

### 4.1. Establishing a Robust Predictive Maintenance Strategy

As a result, the first step to arriving at a strategic approach to implementing a predictive maintenance plan is to evaluate its systems and put a system plan that meets the software lifecycle into practice. This includes:

- Identifying Key Performance Indicators (KPIs): It is important to choose right set of KPIs for tracking the issues related to the health of the software. Several KPIs related to the system's dependability, response rates, error frequencies, and user satisfaction should be used [13].
- Data Collection and Analysis: A robust predictive maintenance approach relies on collecting high-quality data from various sources, such as system logs, user feedback, performance metrics, and historical maintenance records. This data helps to train machine learning models, uncover patterns, and anticipate issues before they impact operations[19].
- Implementing Advanced Analytics and Machine Learning Models: Leveraging machine learning models enables real-time analysis of vast datasets, allowing early identification of issues. Models such as anomaly detection, regression, and clustering are commonly used to predict maintenance needs[34].

### 4.2. Aligning Predictive Maintenance with Business Objectives

Predictive maintenance should be seamlessly aligned with an organisation's overall business objectives. This requires:

- Defining Business Value: Understanding the business impact of predictive maintenance efforts ensures that strategies are targeted and justifiable. Maintenance should focus on areas that improve ROI, customer satisfaction, and product quality[35].
- Prioritizing Critical Systems and Processes: Not all software components require the same maintenance attention. By focusing predictive maintenance on high-impact or mission-critical systems, organisations can maximise resources and minimise disruptions.
- Involving Stakeholders Across Departments: Predictive maintenance initiatives benefit from cross-functional collaboration, bringing together IT, data science, and business teams to align on objectives[36], data requirements, and expected outcomes.

### 4.3. Regular Model Updates and Retraining

Predictive models for maintenance require continuous monitoring, evaluation, and adjustment to remain accurate and effective over time. Key practices for regular model updates include:

- Monitoring Model Performance: Predictive models should be assessed continuously for accuracy, precision, and relevance. Performance metrics help detect model drift, ensuring models remain aligned with evolving system behaviours and software changes[37].
- Retraining Models with Updated Data: As software systems and user behaviours change, so do the patterns of potential failures. Regularly retraining models with recent data allows predictive maintenance systems to stay responsive to new trends and avoid outdated predictions.
- Automating Model Retraining Processes: Automating parts of the retraining pipeline can streamline the process, allowing quick model updates without significant manual intervention. This is particularly effective in environments with frequent software updates or changes[38].

## 5. Literature review

This section provides a literature review of Optimizing Software Lifecycle Management through Predictive Maintenance: Insights and Best Practices, a summary shown in Table 1.

This, Radaideh (2021) Using the PMBOK Guide and the SWEBOK-V3.0, two standards developed by the IEEE, this article details a senior software project management course. It also aims to address a number of research questions, such as

how software project management differs from more conventional methods of project management and whether or not the course satisfies the requirements set out by the IET and the ABET for accreditation[39].

The article, Kravets, Orudjev and Salnikova (2019) explains the procedure that service staff follow while servicing the business's office equipment. Creating software for predictive maintenance and repairs of office equipment is the focus of this article. This includes algorithms for tracking office equipment and repair cartridge incomings, as well as algorithms for locating and monitoring networked office equipment and creating maintenance cycles. Class relationships and data flow within the program are shown in the primary diagrams. The results of the generated algorithms and the selected approaches are presented in a generalised form [40].

In this study, da Silva, de Sá and Menegatti (2019) This research employed PicoScope6 software to detect system problems; subsequent predictive maintenance compared with traditional maintenance approaches; and finally, it allowed for the estimation of repair time. Findings demonstrated that software was more accurate in identifying malfunctioning mechanical parts. Repair time was cut by 88%, and repair cost was lowered by up to 93% when failed components were identified, and a list of components to be repaired or exchanged was estimated based on that. The list of components used in the comparison was created by examination without diagnostic software. Less time spent fixing machines meant more time for machines to be used in the field, which meant more equipment availability [41].

This, Klespitz, Biro and Kovacs (2016) Using a case study approach, this article lays out a set of criteria to assist a corporation in enhancing its current lifecycle management system. The report zeroes in on specifics of software development for medical devices. By offering a quantitative comparison or by emphasising the reasons for exclusion, the answers to these questions enable attaining an ideal choice among the available management systems [42].

To, Rehman et al. (2018) employed both theoretical and practical methods to identify criteria for successful agile maintenance, such as: before planning; on-site client presence; iterative maintenance; post-phase documentation updates; and testability of maintenance [43].

Here's Table 1 summarising the related works mentioned, focusing on their main areas of software lifecycle management through predictive maintenance.

**Table 1 S**ummary of the related work for software lifecycle management through predictive maintenance

| Ref | Focus | Approach | Key Findings | Challenges | Future Work |
|---|---|---|---|---|---|
| [39] | Senior software project management course | Based on PMBOK Guide and IEEE SWEBOK-V3.0 | Compares traditional project management vs. software project management. The course aligns with IET & ABET accreditation criteria. | Ensuring compliance with accreditation criteria, adapting traditional methods to software projects. | Further integration of software project management principles into specialised courses with evolving standards. |
| [40] | Maintenance and repair of office equipment | Development of predictive maintenance software algorithms | Software helps find networked equipment, track maintenance cycles, and record office equipment and cartridges for repair. Algorithms improve maintenance efficiency. | Developing a universally applicable maintenance system across various equipment types | Extending algorithms to cover more equipment types and improving the efficiency of predictive maintenance software. |
| [41] | Predictive maintenance for tractor clutch system | Use of PicoScope6 software for diagnostics and predictive maintenance | Diagnostic software predicted failures more accurately, reducing repair time by 88% and costs by 93%, improving equipment availability. | Implementing the software on diverse tractor systems and ensuring compatibility | Expanding the scope of diagnostic software to other agricultural machinery and evaluating broader cost-benefit outcomes. |

| | | | | | |
|---|---|---|---|---|---|
| [42] | Improving lifecycle management system for medical device software | Case study for developing system requirements for software management | Identified requirements for optimising lifecycle management in medical device software, leading to better system choice. | Addressing the complexity of medical device regulations and ensuring software adaptability | Exploring more efficient lifecycle management systems for medical devices and other high-regulation industries. |
| [43] | Agile maintenance management | Theoretical and empirical analysis of agile maintenance factors | Key factors for successful agile maintenance include customer presence, iterative processes, documentation updates, and testable maintenance. | Ensuring alignment between agile methodologies and maintenance tasks across different industries | Investigating further integration of agile maintenance practices with predictive maintenance for continuous improvement. |

## 6. Conclusion

The integration of Predictive Maintenance (PM) into Software Lifecycle Management (SLM) offers significant advantages in ensuring software reliability, reducing maintenance costs, and extending system longevity. Through advanced predictive techniques like machine learning models, time series analysis, and log mining, organisations can proactively address software issues before they affect operations. By focusing on the four key types of maintenance—corrective, adaptive, perfective, and preventive—PM enables a more efficient and effective maintenance process. The benefits of implementing PM include minimised downtime, improved safety, better decision-making, and an enhanced ability to adapt to emerging technologies like IoT and Industry 4.0. To fully realise these advantages, organisations must adopt a comprehensive predictive maintenance strategy that aligns with their business goals, ensuring that resources are focused on critical systems and performance metrics. Overall, optimising software lifecycle management through predictive maintenance not only enhances software quality but also aligns software maintenance efforts with organisational priorities, resulting in a more agile and responsive software development process.

## Compliance with ethical standards

### *Disclosure of conflict of interest*

No conflict of interest to be disclosed.

## References

[1] B. Tarika, "A Review of Software Development Life Cycle Models," Int. J. Adv. Res. Comput. Sci. Softw. Eng., 2019.

[2] R. Goyal, "THE ROLE OF REQUIREMENT GATHERING IN AGILE SOFTWARE DEVELOPMENT: STRATEGIES FOR SUCCESS AND CHALLENGES," Int. J. Core Eng. Manag., vol. 6, no. 12, pp. 142–152, 2021.

[3] V. Kumar and F. T. S. Chan, "A superiority search and optimisation algorithm to solve RFID and an environmental factor embedded closed loop logistics model," Int. J. Prod. Res., vol. 49, no. 16, 2011, doi: 10.1080/00207543.2010.503201.

[4] V. V. Kumar, F. T. S. Chan, N. Mishra, and V. Kumar, "Environmental integrated closed loop logistics model: An artificial bee colony approach," in SCMIS 2010 - Proceedings of 2010 8th International Conference on Supply Chain Management and Information Systems: Logistics Systems and Engineering, 2010.

[5] V. V. Kumar, A. Sahoo, and F. W. Liou, "Cyber-enabled product lifecycle management: A multi-agent framework," in Procedia Manufacturing, 2019. doi: 10.1016/j.promfg.2020.01.247.

[6] M. M. Lehman and L. A. Belady, Program evolution: processes of software change. USA: Academic Press Professional, Inc., 1985.

[7] A. P. A. Singh, "Streamlining Purchase Requisitions and Orders : A Guide to Effective Goods Receipt Management," J. Emerg. Technol. Innov. Res., vol. 8, no. 5, pp. g179–g184, 2021.

[8] V. K. Y. Nicholas Richardson, Rajani Pydipalli, Sai Sirisha Maddula, Sunil Kumar Reddy Anumandla, "Role-Based Access Control in SAS Programming: Enhancing Security and Authorization," Int. J. Reciprocal Symmetry Theor. Phys., vol. 6, no. 1, pp. 31–42, 2019.

[9] R. Goyal, "THE ROLE OF BUSINESS ANALYSTS IN INFORMATION MANAGEMENT PROJECTS," Int. J. Core Eng. Manag., vol. 6, no. 9, pp. 76–86, 2020.

[10] V. V. Kumar, M. K. Pandey, M. K. Tiwari, and D. Ben-Arieh, "Simultaneous optimisation of parts and operations sequences in SSMS: A chaos embedded Taguchi particle swarm optimisation approach," J. Intell. Manuf., 2010, doi: 10.1007/s10845-008-0175-4.

[11] V. Kumar, V. V. Kumar, N. Mishra, F. T. S. Chan, and B. Gnanasekar, "Warranty failure analysis in service supply Chain a multi-agent framework," in SCMIS 2010 - Proceedings of 2010 8th International Conference on Supply Chain Management and Information Systems: Logistics Systems and Engineering, 2010.

[12] V. V Kumar, "An interactive product development model in remanufacturing environment : a chaos-based artificial bee colony approach," 2014.

[13] S. K. R. Anumandla, V. K. Yarlagadda, S. C. R. Vennapusa, and K. R. V Kothapalli, "Unveiling the Influence of Artificial Intelligence on Resource Management and Sustainable Development: A Comprehensive Investigation," Technol. \& Manag. Rev., vol. 5, no. 1, pp. 45–65, 2020.

[14] C. Wagner and B. Hellingrath, "Supporting the implementation of predictive maintenance – a process reference model," Int. J. Progn. Heal. Manag., 2021.

[15] V. K. Yarlagadda, "Harnessing Biomedical Signals: A Modern Fusion of Hadoop Infrastructure, AI, and Fuzzy Logic in Healthcare," Malaysian J. Med. Biol. Res., vol. 2, no. 2, pp. 85–92, 2021.

[16] J. Thomas, K. V. Vedi, and S. Gupta, "Enhancing Supply Chain Resilience Through Cloud-Based SCM and Advanced Machine Learning: A Case Study of Logistics," J. Emerg. Technol. Innov. Res., vol. 8, no. 9, 2021.

[17] J. Thomas, K. V. Vedi, and S. Gupta, "The Effect and Challenges of the Internet of Things (IoT) on the Management of Supply Chains," Int. J. Res. Anal. Rev., vol. 8, no. 3, pp. 874–879, 2021.

[18] G. P., "Software Development Lifecycle Model (SDLC) Incorporated With Release Management," vol. 3, no. 4, pp. 1536–1543, 2016.

[19] K. Patel, "Quality Assurance In The Age Of Data Analytics: Innovations And Challenges," Int. J. Creat. Res. Thoughts, vol. 9, no. 12, pp. f573–f578, 2021.

[20] G. Gurung, R. Shah, and D. P. Jaiswal, "Software Development Life Cycle Models-A Comparative Study," Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol., 2020, doi: 10.32628/cseit206410.

[21] B. Goel, "Integrating Preventive Maintenance within the Product Life Cycle," Int. J. Comput. Appl., vol. 25, no. 8, pp. 14–22, 2011, doi: 10.5120/3053-4156.

[22] N. Agarwal, S. Gupta, and S. Gupta, "A comparative study on discrete wavelet transform with different methods," in 2016 Symposium on Colossal Data Analysis and Networking (CDAN), Mar. 2016, pp. 1–6. doi: 10.1109/CDAN.2016.7570878.

[23] M. Z. Hasan, R. Fink, M. R. Suyambu, and M. K. Baskaran, "Assessment and improvement of intelligent controllers for elevator energy efficiency," in IEEE International Conference on Electro Information Technology, 2012. doi: 10.1109/EIT.2012.6220727.

[24] B. Boehm, "Get ready for agile methods, with care," Computer (Long. Beach. Calif)., 2002, doi: 10.1109/2.976920.

[25] M. Z. Hasan, R. Fink, M. R. Suyambu, M. K. Baskaran, D. James, and J. Gamboa, "Performance evaluation of energy efficient intelligent elevator controllers," in IEEE International Conference on Electro Information Technology, 2015. doi: 10.1109/EIT.2015.7293320.

[26] M. Gopalsamy, "Artificial Intelligence (AI) Based Internet-ofThings (IoT)-Botnet Attacks Identification Techniques to Enhance Cyber security," Int. J. Res. Anal. Rev., vol. 7, no. 4, pp. 414–420, 2020.

[27] J. Thomas, "The Effect and Challenges of the Internet of Things (IoT) on the Management of Supply Chains," Int. J. Res. Anal. Rev., vol. 8, no. 3, pp. 874–878, 2021.

[28] J. Thomas and V. Vedi, "Enhancing Supply Chain Resilience Through Cloud-Based SCM and Advanced Machine Learning: A Case Study of Logistics," J. Emerg. Technol. Innov. Res., vol. 8, no. 9, 2021.

[29] V. V. Kumar, S. R. Yadav, F. W. Liou, and S. N. Balakrishnan, "A digital interface for the part designers and the fixture designers for a reconfigurable assembly system," Math. Probl. Eng., 2013, doi: 10.1155/2013/943702.

[30] R. Bishukarma, "The Role of AI in Automated Testing and Monitoring in SaaS Environments," IJRAR, vol. 8, no. 2, 2021, [Online]. Available: https://www.ijrar.org/papers/IJRAR21B2597.pdf

[31] S. Gupta, N. Agrawal, and S. Gupta, "A Review on Search Engine Optimization: Basics," Int. J. Hybrid Inf. Technol., 2016, doi: 10.14257/ijhit.2016.9.5.32.

[32] K. Dixit, P. Pathak, and S. Gupta, "A new technique for trust computation and routing in VANET," in 2016 Symposium on Colossal Data Analysis and Networking, CDAN 2016, 2016. doi: 10.1109/CDAN.2016.7570944.

[33] S. Dixit, P. Pathak, and S. Gupta, "A novel approch for gray hole and black hole detection and prevention," in 2016 Symposium on Colossal Data Analysis and Networking, CDAN 2016, 2016. doi: 10.1109/CDAN.2016.7570861.

[34] S. Rathi, V. S. Rajput, and S. Gupta, "An improved evolution based optimization algorithm originated from the concept of SFLA and simulated annealing," in 11th International Conference on Industrial and Information Systems, ICIIS 2016 - Conference Proceedings, 2016. doi: 10.1109/ICIINFS.2016.8262933.

[35] C. Chen, C. Wang, N. Lu, B. Jiang, and Y. Xing, "A data-driven predictive maintenance strategy based on accurate failure prognostics," Eksploat. i Niezawodn., 2021, doi: 10.17531/EIN.2021.2.19.

[36] P. Pathak, A. Shrivastava, and S. Gupta, "A survey on various security issues in delay tolerant networks," J Adv Shell Program., vol. 2, no. 2, pp. 12–18, 2015.

[37] R. Arora, "Mitigating Security Risks on Privacy of Sensitive Data used in Cloud-based Mitigating Security Risks on Privacy of Sensitive Data used in Cloud-based ERP Applications," 8th Int. Conf. "Computing Sustain. Glob. Dev., no. March, pp. 458–463, 2021.

[38] A. S. Ramakrishna Garine, Rajeev Arora, Anoop Kumar, "Advanced Machine Learning for Analyzing and Mitigating Global Supply Chain Disruptions during COVID-19," SSRN, pp. 1–6, 2020.

[39] M. A. Radaideh, "Shifting the paradigms from teaching project management to teaching software project management at Jordan university of science and technology based on the IEEE software engineering management knowledge area," in Proceedings - 2021 International Conference on Computational Science and Computational Intelligence, CSCI 2021, 2021. doi: 10.1109/CSCI54926.2021.00227.

[40] A. G. Kravets, N. Y. Orudjev, and N. A. Salnikova, "Software for Predictive Maintenance and Repair of the Enterprise Office Equipment," in 2019 International Multi-Conference on Industrial Engineering and Modern Technologies, FarEastCon 2019, 2019. doi: 10.1109/FarEastCon.2019.8934186.

[41] C. A. G. da Silva, J. L. R. de Sá, and R. Menegatti, "Diagnostic of Failure in Transmission System of Agriculture Tractors Using Predictive Maintenance Based Software," AgriEngineering, 2019, doi: 10.3390/agriengineering1010010.

[42] J. Klespitz, M. Biro, and L. Kovacs, "Aspects of improvement of software development lifecycle management," in CINTI 2015 - 16th IEEE International Symposium on Computational Intelligence and Informatics, Proceedings, 2016. doi: 10.1109/CINTI.2015.7382943.

[43] F. U. Rehman, B. Maqbool, M. Q. Riaz, U. Qamar, and M. Abbas, "Scrum Software Maintenance Model: Efficient Software Maintenance in Agile Methodology," in 21st Saudi Computer Society National Computer Conference, NCC 2018, 2018. doi: 10.1109/NCG.2018.8593152.