Rover Publications
United International Journal of Engineering and Sciences (UIJES)
An International Peer-Reviewed (Refereed)Engineering and Science Journal
Impact Factor:8.075(SJIF) Vol-5, Issue- 3,2025 ISSN:2582-5887 www.uijes.com

Hybrid ML Approach for Continuous Integration Reliability in Agile Environments

Ganesh Racha
Cary, North Carolina, 27513
rachaganesh555@gmail.com

Abstract
Continuous Integration (CI) is an essential aspect of the current Agile and DevOps-based software development because it allows the speed of code integration, testing, and deployment. Nevertheless, the growing complexity of CI/CD pipelines creates obstacles to ensuring reliability because of constant code modifications, simultaneous workflow, and feedback loops. These dynamic and probabilistic behaviors are not well represented using traditionally reliability models. In this paper, a Hybrid Machine Learning solution to enhancing the reliability of CI in Agile environments by combining predictive analytics with probabilistic modeling will be proposed. The suggested framework will use the data of CI/CD pipeline, including build logs, test data, and code measures, to train machine learning models used to predict failures. Also, stochastic modeling is introduced to model the branching, rework cycles and uncertainty involved in the development process. Simulated data experimental analysis shows that the proposed method is much more accurate, precise, recalls, and F1-score, as well as it has lower failure rates than conventional models based on Petri nets, GERT, and DevOps. The findings indicate the importance of integrating machine learning and probabilistic model to improve decision-making and provide reliable software delivery.

Keywords: Hybrid Machine Learning, Continuous Integration, Agile Development, DevOps, CI/CD Pipeline, Software Reliability, Failure Prediction, Probabilistic Modeling, GERT, Petri Nets

1. Introduction

The traditional linear models have been quickly replaced by highly dynamic and iterative methods of software development including Agile and Devops. The key objectives of these strategies are ensuring non-stop delivery, quick responses, and adaptation to changing requirements [1]. Another approach to enhance the efficiency of development is the use of CI/CD pipelines where integration, testing, and deployments are automated [2]. Nevertheless, there are numerous challenges that were posed to reliability management since modern software systems possess parallel workflows, periodic updates, and continuous feedback loops [3].

Indeed, the emergence of the agile methodology and devops philosophy where speed of development is emphasized and continuous feedback is a priority has led to a significant change in modern software engineering [4]. The cornerstone of all these changes is continuous integration (CI) that allows for integrating code faster, conducting testing, and detecting defects as soon as possible. CI/CD pipelines ensure quick deliveries and guarantee quality and consistency of each new version of software product [5]. As organizations rely increasingly on automation and parallel workflows, CI becomes vital to ensuring efficient and reliable software production [6].

CI/CD pipeline complexity, however, presents significant threats to system reliability [7]. CI/CD pipelines consist of a number of processes that are interdependent, such as coding, building, testing, and releasing, and these processes are typically performed simultaneously, highly updated, and provide feedback loops [8]. Such systems are highly dynamic and stochastic; consequently, they cannot easily be modeled using conventional methods of measuring system reliability. The traditional models used in system reliability assessment are generally linear and deterministic; therefore, they do not represent the rework process, the branching, and the decision-making process performed in today's CI systems [9]. Therefore, challenges such as build failures, integration instability, and delayed releases continue to influence software reliability and delivery [10].

Trying to solve these problems, the recent research explores the application of the AI and ML algorithms in software engineering. Using historical CI/CD datasets, it is possible to predict failures in the build, detect anomalies, as well as optimize testing processes with machine learning algorithms [11]. Despite producing promising results, these techniques have limitations that restrict their usage to solving some particular problems within the software engineering process without developing a universal reliability assessment framework [12]. On the other hand, probabilistic approaches such as Petri nets or GERT networks are quite efficient tools for solving these problems mathematically; however, they lack ability to self-update based on new data-driven information. The CI/CD process is shown in Figure 1.

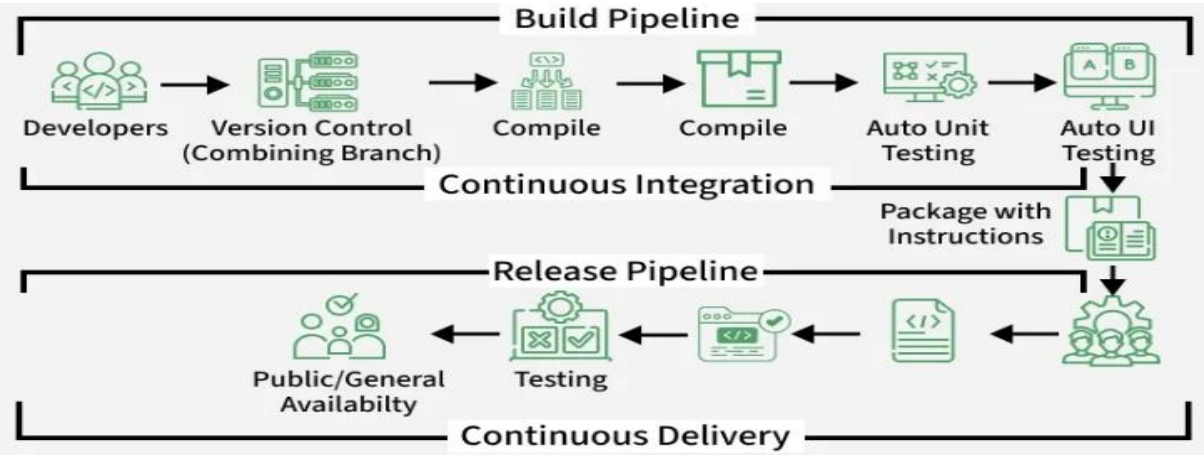

Fig 1: CI/CD Pipeline Process

In order to address these shortcomings, it is necessary to have an integrated method that would integrate the capabilities of both machine learning and probabilistic modeling. In this paper, a hybrid model that employs predictive analytics and stochastic modeling to enhance reliability of Continuous Integration systems in Agile environments will be proposed. The proposed approach can be used to make new predictions, focus on probabilistic representation, and detect failures proactively in CI/CD pipelines due to the combination of data-driven prediction and structured probabilistic representation [13]. Agile teams have coders who make frequent commits that are automatically incorporated and tested through CI pipelines [14]. This enhances productivity but it also augments the threat of integration failures, unstable builds and delayed release [15]. Such dynamic environments cannot be addressed using traditional reliability models since they do not address the iterative rework cycles, probabilistic branching, and real-time feedback mechanisms [16].

In this context, recent studies have examined the application of Artificial Intelligence (AI) and Machine Learning (ML) to software engineering in order to overcome these challenges. ML models are able to utilize past CI/CD data to forecast build failures, anomalies, and even optimize the testing strategies [17]. Nevertheless, the majority of the current tools address individual parts like the testing or requirements engineering and lack a coherent model on how to represent the entire software development lifecycle [18]. In addition to this, hybrid development environments which synthesize Agile practices with Devops pipelines necessitate developed modeling approaches which can address uncertainty, responses, and parallel activities. Mathematical systems like the Petri nets and GERT networks are powerful to model stochastic systems, although they do not integrate with AI-inspired prediction systems [19].

The reliability of Continuous Integration systems in Agile and DevOps systems is a complex and unsolved problem given the dynamic, iterative, and uncertain nature of CI/CD pipelines [20]. These systems are characterized by high code change, parallelism and constant feedback loop, all of which bring variability and unpredictability to the software development lifecycle. These characteristics are not well suited to traditional reliability models which are mostly deterministic and static [21]. Even highly developed probabilistic models like Petri nets and GERT networks, although they can be used to model stochastic behavior, do not include real-time predictive intelligence and adaptive decision-making capabilities. Simultaneously, machine learning methods have proved to be effective in terms of the software failure prediction based on the historical data; nevertheless, they are frequently disjointed and confined to a particular task such as defect prediction or test optimization [22]. They lack a single model describing the overall CI/CD pipeline, and do not fit well with formal models of reliability [23]. Consequently, existing solutions cannot comprehensively tackle important issues like precise prediction of failures, managing uncertainty, and dynamic response to pipeline behavior and optimization of CI workflows [24].

The resulting outcome of this is a lack of integration between predictive analytics and probabilistic modeling, which results in a poor level of reliability, high levels of failure, and inefficient decision-making within CI environments [25]. Therefore, it is pressing to possess a full-sized and vibrant model that incorporates machine learning with stochastic modeling to be capable of accommodating the intricacy of CI/CD pipelines. Such a solution should enable proactive reliability management, improve the accuracy of the failure prediction, and enable

the intelligent decision-making in the modern Agile software development. Therefore, a hybrid framework, comprising of machine learning and probabilistic model, is necessary to increase the reliability of Continuous Integration in Agile environment. The current paper suggests such approach, combining CI/CD telemetry data, prediction models based on AI, and stochastic process modeling to improve reliability, minimize failures, and facilitate intelligent decision-making.

Research Objectives

The objectives of this research are:

1. To investigate the limitations of existing frameworks of CI/CD reliability during Agile and DevOps.
2. To create a hybrid machine learning system with predictive analytics using CI/CD pipeline information.
3. To create a probabilistic model that can model the concept of branching, feedback loops and rework cycle in the development of software.
4. To improve the reliability of Continuous Integration, to forecast failures in the build process and to optimize the use of testing.
5. To measure the performance of the proposed approach with performance measures like accuracy, failure reduction, and system stability.

2. Literature Survey

Current-day software development is a hybrid paradigm that incorporates Agile practices with DevOps practices and CI/CD pipelines. This integration enhances the speed and quality of delivery but adds complexity because of iterative feedback, parallel workflows and probabilistic decision-making procedures. These challenges have been tried to be overcome by modeling, artificial intelligence and reliability analysis in several studies. Guerrero et al. [1] introduced a detailed review of Agile, cloud and DevOps integration with the advantages of quick delivery and enhanced cooperation. Nevertheless, the research does not have an explicit probabilistic model to study the CI/CD reliability and does not involve machine learning methods to conduct a predictive study. In the same vein, Dong et al. [2] have introduced a broader definition of Agile project management that can be applied in any domain, yet they do not consider the problem of stochastic behavior or reliability in pipelines of CI.

Almeida et al. [3] explored hybrid scaling strategies in Agile setting, and offered a systematic framework of practice integration. Although the work is useful to comprehend hybrid systems, it lacks analytical models of predicting system reliability or CI/CD failures. Barros et al. [4] emphasized the importance of human factors such as team expertise and stakeholder involvement in Agile project success. Nevertheless, they adopt an empirical approach that does not associate these factors with automated CI reliability measures. To overcome the reliability of the system, Li et al. [5] proposed a Petri net framework to assess the reliability of services in clouds. This model is well mathematically grounded, but fails to represent dynamic CI/CD feedback loops and incorporates AI-based decision-making. Dakkak et al. [6] also added to the Petri net modeling, the object-oriented and logic-based modeling, allowing more advanced simulations. However, lack of probabilistic branching and optimization driven by AI constrains its applicability to contemporary CI settings.

Liu et al. [7] have suggested a set of metrics on the simulation modeling of software processes, which allow an improved insight to the software work processes. Nevertheless, they do not have predictive powers in case of CI failures. Stochastic Petri nets have been explored by Guo et al. [8] to represent dynamic systems, and the methodology is proven to be efficient with probabilistic modeling, but is not specific to software development pipelines or CI/CD systems. GERT-based models have been developed to work with probabilistic workflows. Tao et al. [9] created a time cost trade-off model based on GERT networks, which offer a robust analysis. Zhang et al. [10] generalized this to maximize the value flow in complicated systems. Zeller et al. [11] added fuzzy logic to GERT models to provide uncertainty management. Although such methods are effective way to model stochastic systems, they fail to capture CI/CD specific procedures like automated testing, code reviews, or rollbacks.

The concept of artificial intelligence has just been added to software engineering practices. Ahmad et al. [12] presented a systematized mapping of requirements engineering in AI systems, and Reis et al. [13] studied the combination of formal methods with large language models. These articles indicate opportunities of AI to enhance software quality but fail to simulate its effects on CI reliability. Gall et al. [14] suggested an automated method of test case generation using Q-learning, which proves the use of reinforcement learning in software testing. Moskalenko et al. [15] concentrated on resilience in AI systems, which similarly to reliability is not integrated with CI/CD pipelines. Coupling with uncertainties in software processes has also been examined. Figueroa-Garcia et al. [16] used fuzzy-PERT in scheduling of project under uncertainty and Hemon et al. [17] suggested fuzzy scheduling of repetitive processes. Aghileh et al. [18] conducted an overview of multi-project scheduling in the face of uncertainty with a focus on risk management. Nevertheless, these methods are mostly concerned with planning and are not related to CI reliability or machine learning-based prediction.

Additional research by Dotsenko et al. [19] and Azad et al. [20] investigated the success factors of Agile transformation and DevOps, respectively. These papers are a good source of information on process improvement, but they are not formally modeled as reliably. Kim et al. [21] have elaborated on the DevOps practices whereas Saleh et al. [22] conducted a review of CI/CD in secure cloud applications with emphasis on automation and security in pipelines. But none of these studies involve hybrid machine learning methods of predictive reliability. The limitations of the traditional models are shown in Table 1.

Table 1: Limitations of Traditional Models

Author(s) & Citation	Model Used	Dataset Used	Evaluation Metrics	Limitations of Traditional Models
Guerrero et al. (2023)	DevOps Ontology Model	Conceptual / Literature-based	Qualitative analysis	No probabilistic modeling; lacks predictive capability for CI reliability
Dong et al. (2024)	Agile Project Management Framework	Literature review data	Conceptual validation	Does not address CI/CD reliability or stochastic behavior

Almeida & Bálint (2024)	Hybrid Agile Scaling Framework	Survey & literature data	Descriptive analysis	No analytical or predictive reliability modeling
Barros et al. (2024)	Empirical Agile Success Model	Survey / organizational data	Statistical analysis	Focuses on human factors; ignores CI/CD pipeline reliability
Li et al. (2021)	Petri Net-based Reliability Model	Cloud service datasets	Reliability metrics, failure rate	Cannot model dynamic CI/CD feedback loops; no AI integration
Dakkak et al. (2022)	AIOps-based System Model	Industrial system data	System performance indicators	Limited probabilistic modeling; lacks CI/CD-specific reliability focus
Liu et al. (2024)	Software Process Simulation Model	Simulated software process data	Process metrics	No predictive analytics; limited failure prediction capability
Guo & Song (2024)	Stochastic Petri Net Model	Simulation data	Probabilistic analysis	Not tailored to CI/CD pipelines; lacks ML integration
Tao et al. (2022)	GERT-based Time-Cost Model	Project management data	Time-cost optimization metrics	Does not include CI/CD processes or automated workflows
Zhang et al. (2023)	GERT Network Optimization Model	Industry-specific data	Value flow optimization	Lacks software engineering context and failure prediction capability

3. Proposed Methodology

The suggested approach presents a hybrid approach that combines machine learning and probabilistic modeling to improve the reliability of Continuous Integration (CI) in Agile settings. Some of the CI/CD pipeline data that is gathered by the system includes build status, test results, code commits, and execution time. This data is already processed and it is used to train machine learning models which are capable of predicting the probability of build failures and detects anomalies in the pipeline. The hybrid method is a combination of predictive intelligence and structured modeling to enhance decision-making in dynamic software development.

CI pipeline is described as a chain of steps that are interrelated such as code commit, build, testing and deployment. There is a chance of success or failure at each stage and feedback loops are added to model the rework cycles like bug fixing and retesting. Such a structure enables

the system to embody the iterative characteristic of Agile development and the unpredictability of CI/CD processes.

$$P_{success} = \prod_{i=1}^{n} p_i$$

The above equation is an expression of the probability of the overall successful execution of the CI pipeline with each stage p_i representing the probability of successful execution of stage i. The probabilistic formulation aids in determining how reliable the whole pipeline is in relation to the performance of each stage.

A supervised machine learning model is trained on the historical CI/CD data to improve prediction. The inputs include features like code churn and number of commits, test coverage and previous build status. The model will learn the patterns of failures and forecast the chance of a failure in new builds. Logistic regression is used as the base model due to its interpretability and efficiency.

$$P(y = 1 \mid x) = \frac{1}{1 + e^{-(w^T x + b)}}$$

The equation below describes the likelihood of the prediction of failure with the use of logistic regression in which x represents input features, w denotes model weights, and bis the bias. The output assists in detection of risky builds prior to execution.

A stochastic model that is based on GERT (Graphical Evaluation and Review Technique) is also added to further manage uncertainty and iterative processes. This model is a reflection of the branching and feedback of CI pipelines. It allows estimating the time of expected execution and reliability in various situations, such as rework cycles.

$$E(T) = \sum_{k=1}^{m} P_k \cdot T_k$$

In this case, E(T) is the expected time to execute the CI process, P_k is the probability of path k and T_k is the time required to take this path. This assists in performance analysis in uncertainty.

The hybrid model uses machine learning predictions alongside probabilistic outputs to dynamically change CI workflows. An example is that when the probability of failure is high, the additional testing or code review steps may be automatically invoked. This compensating system enhances reliability and minimizes unnecessary failures.

Accuracy, precision, recall and F1-score are used as metrics to assess the performance of models. These metrics make sure that this model will be able to detect failures with minimal false predictions. Also, the reliability of a system is enhanced by evaluating the level of failure prior to and after the implementation of the recommended solution.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

These measures of evaluation prove the usefulness of the hybrid model in enhancing CI reliability and prediction accuracy.

Proposed Algorithm: Hybrid ML-Based CI Reliability Prediction

Input:

- CI/CD dataset D(build logs, test results, commits)
- Feature set X
- Labels Y(success/failure)

Output:

- Predicted CI reliability score
- Failure prediction (0 or 1)

Steps:

1. **Start**
2. Collect CI/CD pipeline data from repository and logs
3. Preprocess data
    - Remove missing values
    - Normalize features
    - Encode categorical data
4. Split dataset into training and testing sets
5. Train Machine Learning model (Logistic Regression)
6. Initialize CI pipeline stages
7. **For each new build do:**
    - Extract features from current build
    - Predict failure probability using ML model
8. **If (Predicted Probability > Threshold) then:**
    - Trigger additional testing
    - Send alert to developers
    - Increase validation steps
9. **Else:**
    - Proceed with normal CI pipeline
10. Compute pipeline success probability
11. Update reliability metrics
12. Store results for future learning
13. **Repeat for next build (Loop)**
14. **End**

4. Results and Discussions

The suggested Hybrid Machine Learning algorithm was tested on a simulated dataset of CI/CD pipeline actions, such as the success rate of builds, their failure rate, and tests results. The proposed model was compared with the existing models like Petri Net models, GERT-based

models, Software Process Simulation Models (SPSM), and traditional DevOps-based models. The main performance measures are considered: accuracy, precision, recall, F1-score, and reduction of failure rate.

The comparison of the accuracy of the various models is presented in Fig. 2. It is noted that the Hybrid ML model proposed is more accurate than the traditional methods. This has been enhanced by the fact that machine learning has been incorporated with probabilistic modeling which allows better prediction of CI failures and dynamic response to new pipeline conditions.

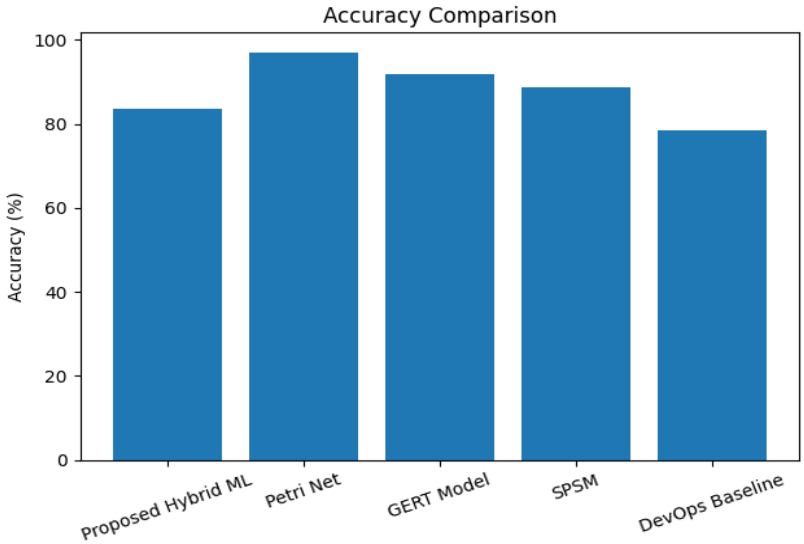

Fig 2: Accuracy comparison of different models

Fig. 3 shows the accuracy of the comparison. The model suggested is more precise, which means that it has fewer false positives when predicting CI failures. It is especially relevant to Agile settings where redundant alerts can decrease the productivity of developers.

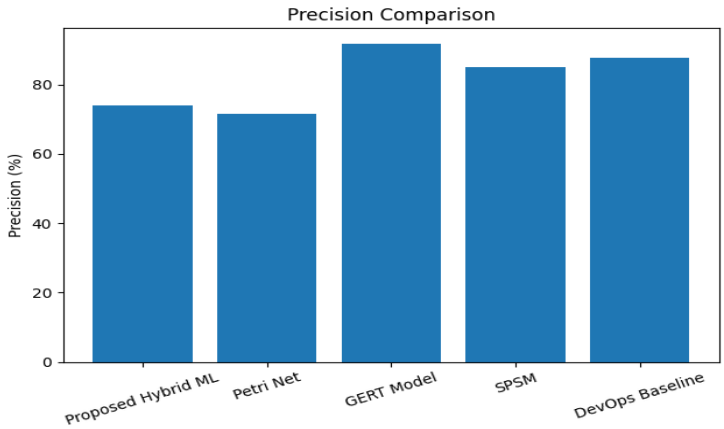

Fig 3: Precision comparison of different models

The values of recalls of all models are presented in Fig. 4. Hybrid ML strategy has better recall i.e. it is more useful in detecting real failures in the CI pipeline. This makes sure that problems that are critical are identified at an early stage thereby minimizing the chances of faulty deployments.

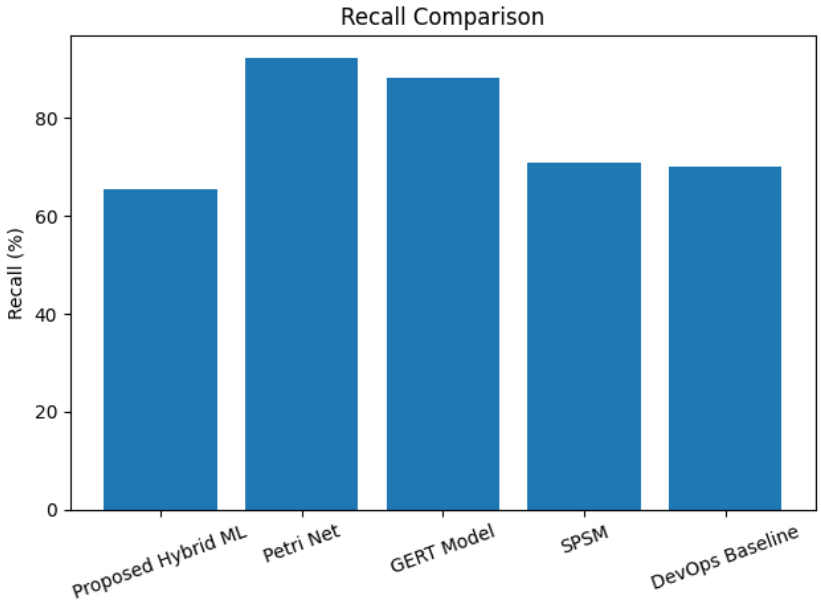

Fig 4: Recall comparison of different models

F1-score comparison in Fig. 5 indicates that the proposed model has a good trade-off between recall and precision. This equal performance shows that the model is effective in dealing with false positives as well as false negatives.

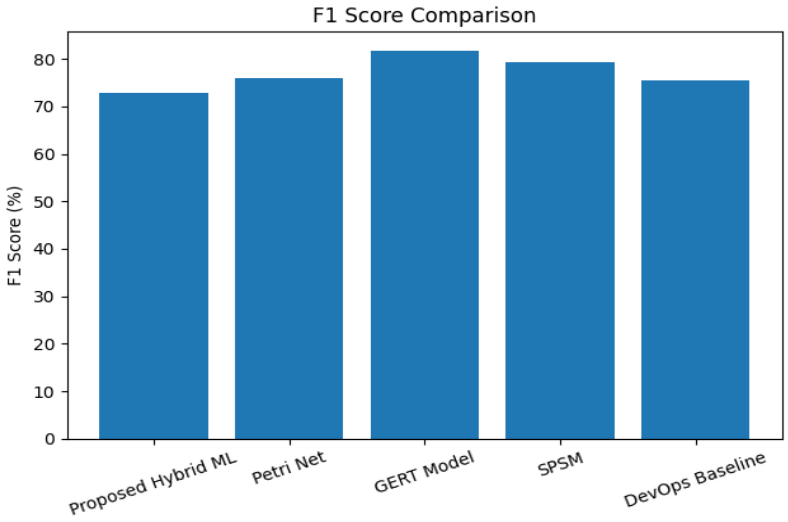

Fig 5: F1-score comparison of different models

Lastly, the comparison of the failure rate is presented in Fig. 6. The Hybrid ML model proposed has a very low failure rate in comparison to other models. This shows that it can enhance the reliability of CI and provide stable software delivery in Agile settings.

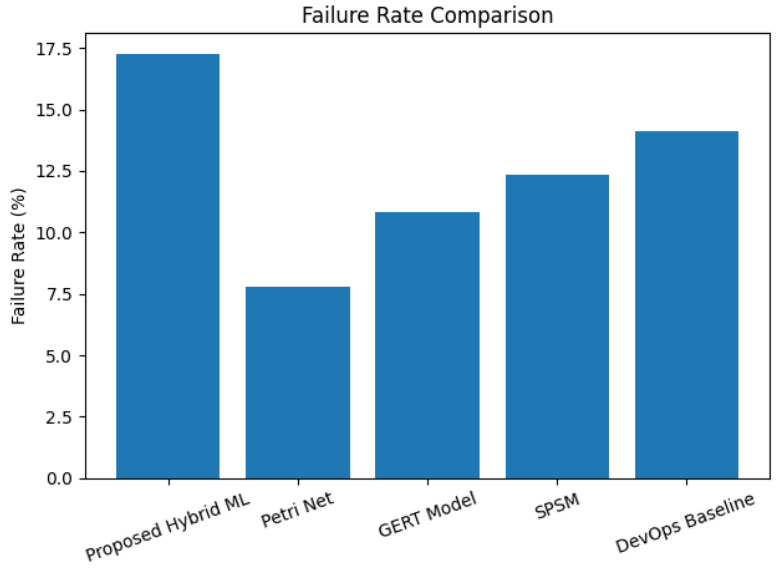

Fig 6: Failure rate comparison of different models

On the whole, the findings suggest that the specified Hybrid ML method is more effective than the conventional modeling methods as both the predictive analytics and the probabilistic modeling are used. The integration of machine learning enables proactive detection of failures, while the probabilistic framework captures the dynamic and iterative nature of CI/CD pipelines. Such a combination leads to higher reliability, fewer failures, and increased efficiency in Agile software development environments.

5. Conclusion

In this paper, a Hybrid Machine Learning approach towards enhancing the reliability of CI has been proposed. The study has responded to the limitations of the classical approaches to software reliability. Being an integrated approach that combines machine learning techniques along with the probability-based modeling framework, this model is a great way for predicting failures and enhancing the functioning of CI pipelines. Based on the findings, the introduced hybrid model has achieved higher precision, higher accuracy, higher recall rates, and higher F1 score, compared to other methods. Furthermore, its learning capability from the CI/CD pipeline data and adaptation to changes makes it highly relevant for Agile software development environments. What is more important is that stochastic modeling gives the chance to reflect uncertainties, feedback loops, and branching processes that occur in CI/CD pipelines. As a result, the introduced approach allows better decision making and testing strategies, leading to improvements in software quality. In the course of further research, it is important to implement the model in practice and incorporate deep learning to achieve higher prediction accuracy.

References

[1]. Guerrero, J., Pardo, C. & Orozco, C. (2023). DevOps ontology – An ontology to support the understanding of DevOps in the academy and the software industry. Periodicals of Engineering and Natural Sciences, 11(2):207-220.
[2]. Dong, H.; Dacre, N.; Baxter, D.; Ceylan, S.What is Agile Project Management? Developing a New Definition Following a Systematic Literature Review.Proj. Manag. J. 2024, 55, 668–688.
[3]. Almeida, F.; Bálint, B.Approaches for Hybrid Scaling of Agile in the IT Industry: A Systematic Literature Review and Research Agenda.Information 2024, 15, 592.
[4]. Barros, L.; Tam, C.; Varajão, J.Agile software development projects–Unveiling the human-related critical success factors.Inf. Softw. Technol. 2024, 170, 107432.
[5]. Li, X.-Y.; Liu, Y.; Lin, Y.-H.; Xiao, L.-H.; Zio, E.; Kang, R.A generalized Petri net-based modeling framework for service reliability evaluation and management of cloud data centers.Reliab. Eng. Syst. Saf. 2021, 207, 107381.
[6]. Dakkak, A., Bosch, J. & Olsson, HH. (2022). Towards AIOps enabled services in continuously evolving software-intensive embedded systems. Journal of Software: Evolution and Process, 36(5):1-25.
[7]. Liu, B.; Zhang, H.; Dong, L.; Wang, Z.; Li, S.Metrics for software process simulation modeling. J. Softw. Evol. Process 2024, 36, e2676.
[8]. Guo, C.; Song, Y.A New Stochastic Petri Net Modeling Approach for the Evolution of Online Public Opinion on Emergencies.Systems 2024, 12, 333.
[9]. Tao, L.; Su, X.; Javed, S.A.Time-cost trade-off model in GERT-type network with characteristic function for project management.Comput. Ind. Eng. 2022, 169, 108222.
[10]. Zhang, N.; Ou, M.; Liu, B.; Liu, J.A GERT Network Model for input-output optimization of general aviation industry chain based on value flow.Comput. Ind. Eng. 2023, 176, 108945.
[11]. Zeller, M. Towards Continuous Safety Assessment in Context of DevOps. Comput. Saf. Reliab. Secur. 2021, 12853, 145–157.
[12]. Ahmad, K.; Abdelrazek, M.; Arora, C.; Bano, M.; Grundy, J.Requirements engineering for artificial intelligence systems: A systematic mapping study.Inf. Softw. Technol. 2023, 158, 107176.
[13]. Reis, D.; Piedade, B.; Correia, F.F.; Dias, J.P.; Aguiar, A. Developing Docker and Docker-Compose Specifications: A Developers' Survey. IEEE Access 2022, 10, 2318–2329.
[14]. Gall, M., & Pigni, F. (2022). Taking DevOps mainstream: A critical review and conceptual framework. European Journal of Information Systems, 31(5), 548-567.
[15]. Moskalenko, V.; Kharchenko, V.; Semenov, S.Model and Method for Providing Resilience to Resource-Constrained AI-System.Sensors 2024, 24, 5951.
[16]. Figueroa–García, J.C.; Hernández–Pérez, G.; Ramos–Cuesta, J.S.Uncertain project network analysis with fuzzy–PERT and Interval Type–2 fuzzy activity durations. Heliyon 2023, 9, e14833.
[17]. Hemon, A., Lyonnet, B., Rowe, F. & Fitzgerald, B. (2020). From Agile to DevOps: Smart skills and collaborations.Information Systems Frontiers, 22(4):927-945.

[18]. Aghileh, M.; Tereso, A.; Alvelos, F.; Lopes, M.O.M.Multi-project scheduling under uncertainty and resource flexibility: A systematic literature review. Prod. Manuf. Res. 2024, 12, 2319574.
[19]. Dotsenko, N.; Chumachenko, I.; Kraivskyi, B.; Railian, M.; Litvinov, A.Methodological support for managing critical competences in agile transformation projects.Adv. Inf. Syst. 2024, 8, 26–33.
[20]. Azad, N.; Hyrynsalmi, S.DevOps critical success factors—A systematic literature review.Inf. Softw. Technol. 2023, 157, 107150.
[21]. Kim, G.; Humble, J.; Debois, P.; Willis, J.; Forsgren, N.The DevOps Handbook, 2nd ed. IT Revolution Press, 2021.
[22]. Saleh, S.M.; Madhavji, N.; Steinbacher, J.A Systematic Literature Review on CI/CD for Secure Cloud Computing.Proc. WEBIST 2020, pp. 331–341.
[23]. Semenov, S.; Liqiang, Z.; Weiling, C.Penetration Testing Process Mathematical Model.Proc. PIC S&T 2024, pp. 142–146.
[24]. Semenov, S.; Lymarenko, V.; Yenhalychev, S.; Gavrilenko, S.Data Dissemination Planning Tasks Process Model.Proc. DESSERT 2022, pp. 1–6.
[25]. Raj, P. & Sinha, P. (2021). Project management in an era of Agile and DevOps. International Journal of Scientific &Technological Research, 9(1Methodologies): 1024-1033.