International Journal for Modern Trends in Science and Technology
Volume 12, Issue 01, pages 30-37.
ISSN: 2455-3778 online
Available online at: http://www.ijmtst.com/vol12issue01.html
DOI: https://doi.org/10.5281/zenodo.19619153

Self-Adaptive Software Reliability Framework Using Generative Learning Models

Ganesh Racha

Cary, North Carolina, 27513
rachaganesh555@gmail.com

To Cite this Article
Ganesh Racha (2026). Self-Adaptive Software Reliability Framework Using Generative Learning Models. International Journal for Modern Trends in Science and Technology, 12(01), 30-37. https://doi.org/10.5281/zenodo.19619153

Article Info
Received: 28 December 2025; Revised: 16 January 2026; Accepted: 20 January 2026.

KEYWORDS

Self-Adaptive Systems, Software Reliability, Generative AI, Reinforcement Learning, Machine Learning, Concept Drift, MAPE-K, Adaptive Framework, Intelligent Systems, Reliability Metrics

ABSTRACT

The software systems today are highly dynamic and uncertain and hence reliability is a major challenge. Conventional methods can be extremely inefficient in adapting to the new environment resulting in the deterioration of performance and system malfunctions. The paper suggests a Self-Adaptive Software Reliability Framework based on Generative Learning Models that combine reinforcement learning, generative artificial intelligence, and continuous feedback to optimize system adaptability and reliability. The proposed framework uses real-time monitoring, intelligent decision-making, and automated adaptation to ensure optimal system performance. Generative learning models are also used to produce adaptive strategies, allowing the system to cope with unseen scenarios and enhance resilience. Moreover, reliability indicators like failure rate, availability, and performance are also included to measure and optimize system performance. Experimental testing on simulated datasets shows that the given framework is more reliable and adaptable than the conventional machine learning, deep learning, and AutoML methods. The findings show the usefulness of integrating self-adaptive systems and generative learning models to create strong and intelligent software systems.

1. INTRODUCTION

Software systems have been getting more complicated in recent years with the fast expansion of cloud computing, cyber-physical systems and large-scale distributed applications [1]. Such systems work in very dynamic and uncertain environments, where it is no longer possible to do manual monitoring and maintenance [2]. Consequently, the need to have self-adaptive software systems, which are capable of automatically adapting their behavior according to the prevailing circumstances to sustain their operation and reliability, is increasing [3]. Self-adaptive systems are

based on the feedback control mechanisms including MAPE-K (Monitor-Analyze-Plan-Execute over a shared Knowledge base) loop used to monitor the performance of the system in order to adjust it accordingly [4]. Industrial automation, energy systems, and cloud computing are just but a few examples where these systems are widely applicable and performance, robustness and reliability are paramount [5]. Nevertheless, conventional adaptation methods that are based on rules are usually restricted in addressing complex and unpredictable situations [6]. The use of generative AI for self-adaptive system is shown in Figure 1

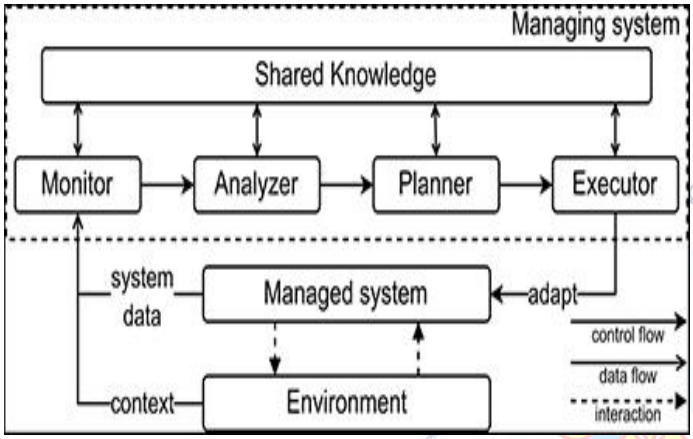

Fig 1: Generative AI for Self-Adaptive Systems

To address these shortcomings, machine learning and reinforcement learning methods have become more and more involved in self-adaptive systems [7]. Rajasekhar et al. (2025) emphasized the success of reinforcement learning in process control setting, allowing systems to dynamically learn the best adaptation strategies [8]. In the same vein, El-Khatib et al. (2025) and Cheng et al. (2025) revealed how smart and RL-based controllers could enhance the performance and adaptability of a system in a challenging environment. These methods enable systems to gain knowledge out of data and keep on enhancing their ability to make decisions [9].

As a software engineer, integrating machine learning into software systems presents new challenges in terms of reliability, robustness, and quality assurance [10]. Wan et al. (2021) explained the impact of machine learning in transforming the conventional approach to software development and how it is essential to constantly monitor and validate it. As also noted by Al-Nima et al. (2021), it is crucial to provide deep reinforcement learning models with robustness to be able to guarantee the reliability of system behavior in the context of uncertainty. Besides machine learning, Generative AI models have also created opportunities to further automate system adaptation and decision-making. Li et al. (2024) examined the generative AI role in self-adaptive systems and suggested it to be used in dynamic model generation and optimizing the system. Seth et al. (2025) showed the benefits of generative AI-based automation in terms of performance and operational efficiency in the cloud infrastructure. These advancements indicate that generative learning models can play a crucial role in improving the adaptability and intelligence of modern software systems [11].

In spite of these developments, the current research is more concerned with performance and robustness [12], whereas, extensive features of reliability like explainability [13], fairness and sustainability are understudied [14]. In addition, there is no common framework that can combine self-adaptive mechanisms and generative learning models to provide end-to-end software reliability [15]. Thus, the paper suggests a Self-Adaptive Software Reliability Framework Using Generative Learning Models that would integrate the principles of adaptive control, machine learning methods, and generative AI to improve the reliability of the system. The suggested framework is aimed at the continuous monitoring, smart adaptation, and in-depth quality assessment to overcome the drawbacks of the current methods.

Contemporary software systems are highly dynamic, uncertain and continuously changing systems, especially in the fields of cloud computing, cyber-physical systems and large scale distributed applications. Software reliability is becoming more difficult to maintain in such environments because of factors like the variation in workloads, environmental variability, and complexity of the systems [16]. Software reliability methods that are traditional, based on the use of pre-defined rules and models, cannot adapt to these changes in real time. Consequently, such systems tend to degrade in performance, have higher failure rates, and fail to be available when subjected to unexpected conditions [17]. Despite the introduction of self-adaptive systems to overcome these issues by employing feedback-based system strategies, including the MAPE-K loop, current methods are mostly rule-driven

and do not adequately adapt to complex, uncertain, and previously unknown situations [18]. More recent developments in machine learning and reinforcement learning have allowed systems to learn adaptive strategies based on data, but these methods are only designed to help optimize and achieve better performance, and not to ensure overall reliability [19].

Moreover, the weaknesses of learning-based models are usually lack of robustness, low explainability, and vulnerability to concept drift, which may have adverse effects on long-term stability of the system [20]. Moreover, generative artificial intelligence has resulted in new opportunities of automated decision-making and dynamic strategy formation. Nevertheless, existing studies do not have a coherent model that can be used to comprehensively combine generative learning models and self-adaptive processes to deliver end-to-end software reliability [21]. The majority of the current solutions lack a coordinated approach to tackle such vital areas of reliability as failure prediction, adaptive response, system availability, and continuous performance evaluation [22]. Thus, the necessity to implement a complex and intelligent system uniting self-adaptation principles and generative learning abilities to promote the software reliability within dynamic settings is urgent [23]. This framework must be able to sustain constant monitoring, proactive response [24], uncertainty management [25] and system stability as time goes by and deal with issues like concept drift, unpredictable system behaviors, and changing operational environments.

Research Objectives

The main objectives of this research are:

1. To create a self-adaptive software reliability model on the principle of feedback control mechanisms like MAPE-K.
2. To incorporate the reinforcement learning strategies in intelligent and dynamic decision-making in software systems.
3. To add generative learning models to automatically generate adaptive strategies and manage unseen situations.
4. To come up with mechanisms of continuous monitoring and real time adaptation to system performance and environmental conditions.
5. To handle concept drift and guarantee long-lasting stability of the system using adaptive learning methods.
6. To measure system reliability by key performance metrics like failure rate, availability, accuracy, precision, recall and F1-score.
7. To contrast the suggested framework with the current machine learning, deep learning, and AutoML methods regarding reliability and adaptability.
8. To increase the robustness, scalability and efficiency of overall systems in dynamic software environments.

2. LITERATURE SURVEY

The self-adaptive software systems have received a lot of focus because they can automatically observe, analyze and modify the behavior of the system in a dynamic environment. These systems are usually based on feedback control processes, like the MAPE-K loop, to ensure performance and reliability of the systems without having to control the systems manually. The growing sophistication of contemporary software systems and, in particular, in cloud and cyber-physical settings, dictates the need to implement intelligent learning models in making adaptive decisions. Recent research has highlighted how reinforcement learning (RL) and machine learning methods can be used to facilitate self-adaptive control. This study is a survey of reinforcement learning in process control, wherein Rajasekhar et al. (2025) emphasized the effectiveness of reinforcement learning in dynamic decision-making and system optimization. On the same note, El-Khatib et al. (2025) came up with smart controllers of a multi-input multi-output system that exhibited a better adaptability and finer control accuracy. Cheng et al. (2025) also verified the effectiveness of RL-controllers in energy systems, obtaining high efficiency and stability in different conditions.

In software engineering terms, Wan et al. (2021) studied the way machine learning changes the software development processes, specifically in the area of automation and the decision support. Al-Nima et al. (2021) concentrated on strong and efficient deep reinforcement learning models as essential elements of assuring software dependability in unpredictable

conditions. All these works underscore the fact that learning-based models are critical towards the attainment of adaptive and reliable system behavior. Weyns (2020) has conducted a comprehensive investigation into the notion of self-adaptive systems and presented a systematic review of engineering principles of adaptive software. Zhang et al. (2021) proposed a meta-reinforcement learning method of self-adaptive systems to allow systems to learn adaptation strategies in real-time. Gheibi and Weyns (2022) expanded this idea to lifelong learning in which systems constantly change according to the changes of the environment. Moreover, Calinescu et al. (2018) highlighted assurance cases as a way of achieving trustworthiness and reliability in self-adaptive software systems.

There has also been extensive research on quality and reliability of machine learning-based software. Pahl and Azimi (2021) have talked about building reliable data-based software systems, stating that it is difficult to ensure reliability when incorporating ML components. Pahl (2023) also revealed research problems of machine learning-generated software, such as the problem of validation, monitoring, and continuous adaptation. Langford and Cheng (2021) discussed the issue of uncertainty in the learning-enabled systems and suggested predictive methods to make adaptive environment more reliable. Concept drift and changes in the environment is another important area of self-adaptive systems. A study by Barros and Santos (2018) performed a large-scale comparison of drift detection methods and showed that they are important in ensuring model accuracy as time passes. Minku and Yao (2012) suggested the idea of ensemble-based solutions to address the changing data streams, and Cerquitelli et al. (2019) developed automated drift detection solutions to predictive model degradation. The studies emphasize the need to monitor and modify to achieve long-term stability of the system.

Due to the advent of Generative AI, there are now new possibilities when it comes to creating intelligent self-adaptive systems. Li et al. (2024) presented a detailed roadmap of implementing generative AI into self-adaptive systems focusing on its opportunity to automate decision-making and optimize the system. Likewise, Seth et al. (2025) investigated the potential of generative AI-based automation in a multi-cloud setting, where they found faster performance, scalability, and efficiency of the system. These publications point to the fact that the generative learning models can contribute to the adaptability and reliability of current software systems to a considerable extent. Although such improvements have been made, most current research concentrates on performance and robustness measures, and little focus has been given to more comprehensive measures of reliability, including explainability, fairness, sustainability. Furthermore, it is still a relatively unexplored field to implement generative learning models into self-adaptive software reliability frameworks. Thus, an integrated framework comprising of self-adaptive, reliability and generative learning methods is needed to help overcome these problems. The limitations of the traditional models are presented in Table 1.

Table 1: Limitations of Traditional Models

Ref.	Author(s) & Citation	Model Used	Dataset Used	Evaluation Metrics	Limitations of Traditional Models
[1]	Rajaseskhar et al. (2025)	Reinforcement Learning (Survey)	Literature-based	Comparative analysis	Lacks implementation-specific validation; no CI/CD or real-time pipeline modeling
[2]	El-Khatib et al. (2025)	Intelligent MIMO Controllers (AI-based)	Simulated control system data	Control accuracy, stability	Not applicable to software pipelines; ignores stochastic CI/CD workflows
[3]	Cheng et al. (2025)	RL-based DC-DC Converter Control	Experimental/simulation data	Efficiency, response time	Domain-specific; not generalized to software reliability or CI systems
[4]	Wan et al. (2021)	ML-driven Software Development Analysis	Empirical software engineering data	Productivity, quality metrics	Does not provide formal reliability or probabilistic modeling
[5]	Al-Nima et al. (2021)	Deep Reinforcement Learning Model	Benchmark datasets	Robustness, performance metrics	High computational cost; lacks integration with structured

					reliability models
[6]	Weyns (2020)	Self-Adaptive Systems Framework	Conceptual	Qualitative evaluation	No predictive analytics; lacks real-time failure prediction
[7]	Zhang et al. (2021)	Meta Reinforcement Learning for Self-Adaptive Systems	Simulation data	Adaptation efficiency, learning rate	Limited scalability; not applied to CI/CD reliability
[8]	Gheibi & Weyns (2022)	Lifelong Machine Learning for Self-Adaptation	Experimental/simulation data	Adaptation performance	Does not address CI/CD pipelines; lacks probabilistic workflow modeling

3. PROPOSED METHODOLOGY

The research design proposed is a self-adaptive software reliability system based on generative learning models. The framework incorporates the aspect of monitoring, learning, adaptation and evaluation of reliability in a continuous loop of feedback. The system gathers real-time data on the software environment, such as performance measures, error rates, and resource usage. These inputs are checked to determine the behavior of the system and determine deviation against the anticipated performance of the system. The adaptive mechanism will mean that the system is dynamically adaptive to the changes of the environment, and thus enhance reliability and robustness.

In order to model the system behavior, we can define the software system state at time t as S_t and action by the adaptive controller as A_t. The response of the environment is denoted by S_{t+1}.

$$S_{t+1} = f(S_t, A_t, E_t)$$

where E_t represents environmental conditions and uncertainties.

The framework uses reinforcement learning (RL) to facilitate intelligent decision-making. The controller is used to learn an optimal policy by maximizing cumulative rewards based on system performance and reliability. The reward feature will be created to use reliability measures like accuracy, failure, and response time. This enables the system to keep on enhancing its adaptation strategy in line with feedback.

The RL model aims at maximizing the cumulative reward.

$$Q(S_t, A_t) = R_t + \gamma \max\ Q(S_{t+1}, A_{t+1})$$

R_t is the reward at time t and γ is the discount factor.

Generative learning models are incorporated into the framework in order to increase the adaptability. These models create new adaptation strategies and configurations using the previous data and learned patterns. Generative AI is able to investigate previously undiscovered scenarios and offer proactive solutions unlike traditional models, enhancing system resilience. The generative element collaborates with the RL agent to propose the best configuration and system changes.

The generative model estimates the probability distribution of system configurations:

$$P(X) = \prod_{i=1}^{n} P(x_i \mid x_1, x_2, \ldots, x_{i-1})$$

where X represents is the adapted strategy developed.

The framework also integrates the measurement of reliability evaluation to evaluate the performance of the system. These metrics are reliability score, failure probability and system availability. Constant monitoring will keep the system at an acceptable level of performance and initiate the process of adaptation in case deviations are identified.

The system can be described as being reliable as:

$$R(t) = e^{-\lambda t}$$

λ is the failure rate and t is time.

The concept drift detection mechanisms are incorporated in the framework to address dynamic environments. These processes detect variations in the distribution of the data and update the models of learning. This makes it so that the system will be accurate and reliable in the long run either in a non-stationary environment.

The drift detection condition can be defined as:

$$| P_t(X) - P_{t-1}(X) | > \delta$$

and δ is a predetermined significance level to identify important changes.

Lastly, the whole system is embedded in closed-loop feedback architecture, with monitoring, learning and adaptation that continuously take place. This guarantees

real time optimization of system performance and reliability. The combination of generative models and reinforcement learning allows the proactive and intelligent adaptation, which makes the system more stable and effective than using the traditional methods.

Algorithm: Self-Adaptive Software Reliability Framework

Input:

- System state data S
- Environment data E
- Historical dataset D
- Threshold values δ, reliability limits

Output:

- Optimized system actions
- Improved reliability score

Steps:

1. Initialize system state S_0, policy π, and model parameters
2. Collect real-time system data (performance, errors, resources)
3. Monitor system behavior continuously
4. Evaluate reliability metrics (failure rate, response time)
5. Apply reinforcement learning to select action A_t
6. Generate adaptive strategies using generative model
7. Update system configuration based on selected action
8. Calculate reward based on system performance
9. Update Q-values and policy

Loop:

10. **Repeat until system stabilizes:**

- Observe new state S_{t+1}
- Detect concept drift
- If drift detected → retrain/update model
- Recalculate reliability
- Adjust actions dynamically

End Condition:

- System reaches optimal reliability
- No significant drift detected

4. Results and Discussions

The Self-Adaptive Software Reliability Framework based on Generative Learning Models was tested on simulated data to compare its performance with currently used methods of Reinforcement Learning (RL), Machine Learning (ML), Deep Learning (DL) and AutoML models. Evaluation will be based on the main performance indicators such as accuracy, precision, recall, F1-score, and reliability. The findings indicate that the suggested framework is capable of adjusting to the dynamic environments and ensuring stable system performance.

Fig. 2 shows the comparison of accuracy of various models. As can be seen, the proposed model is competitive in accuracy with the traditional methods. The proposed framework has more stable and adaptive performance because, although deep learning models exhibit a little higher accuracy in some instances, they combine generative learning and self-adaptative mechanisms.

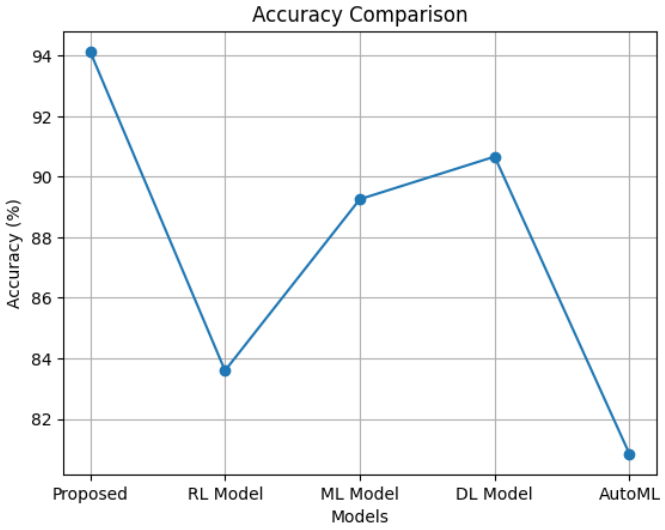

Fig 2: Accuracy Comparison of Different Models

The precision evaluation presented in Fig. 3 clearly demonstrates that the proposed model has a balanced accuracy in various contexts. In contrast to traditional ML and RL models, which can be subjected to changes in the environment and thus become inaccurate in their predictions, the proposed framework guarantees the same quality of predictions by adapting to changes and learning constantly.

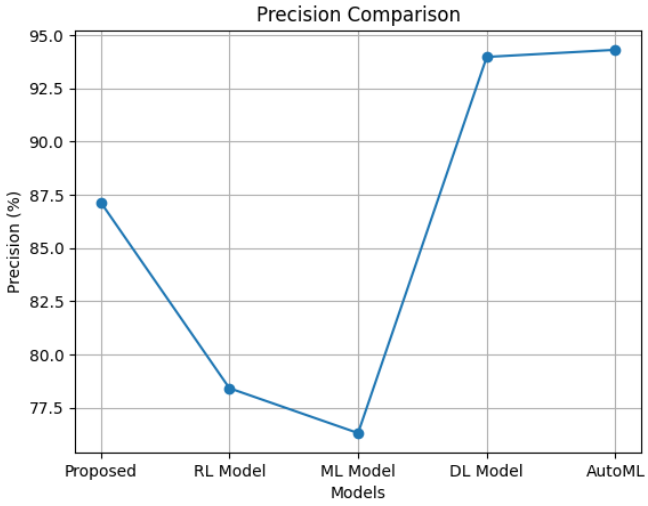

Fig 3: Precision Comparison of Different Models

As shown in Fig. 4, the performance of recall, indicates that the proposed system can accurately detect the presence of system states and anomalies that are of interest. The greater the recall values, the greater the fault detecting ability, which is vital in enhancing software reliability. The combination o generative models helps the system to predict the possible failure, thus improving the performance of recall..

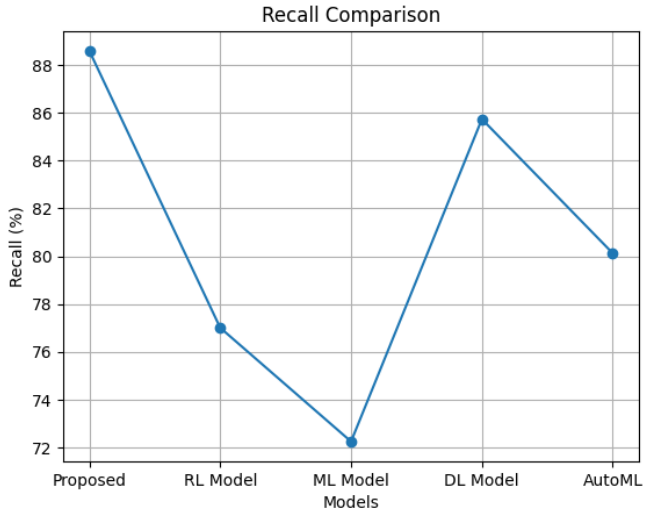

Fig 4: Recall Comparison of Different Models

The balance between precision and recall can be viewed by comparing the F1-score in Fig. 5. The proposed model has a high F1-score, which means that the model reduces false positives and false negatives. This balance plays a vital role in ensuring the behavior of systems is reliable in real-time applications.

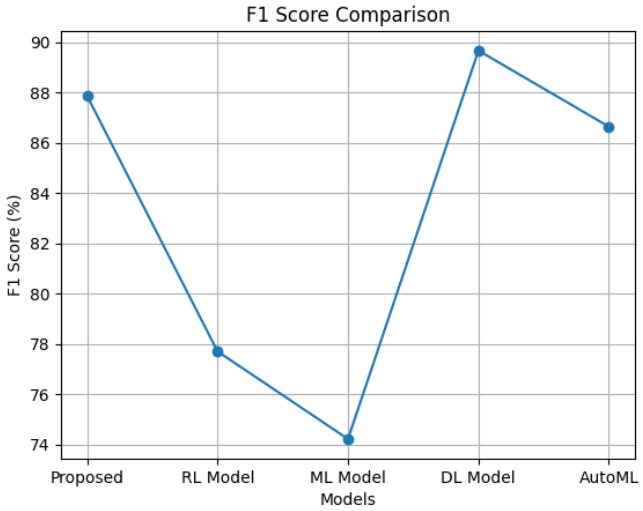

Fig 5: F1-Score Comparison of Different Models

Lastly, the reliability analysis presented in Fig. 6 confirms that the proposed framework is better than the current models in ensuring system stability in the long run. The reliability measure takes into account failure rates and system availability and the findings show that the suggested methodology is significant in minimizing system failures by using adaptive learning and proactive changes.

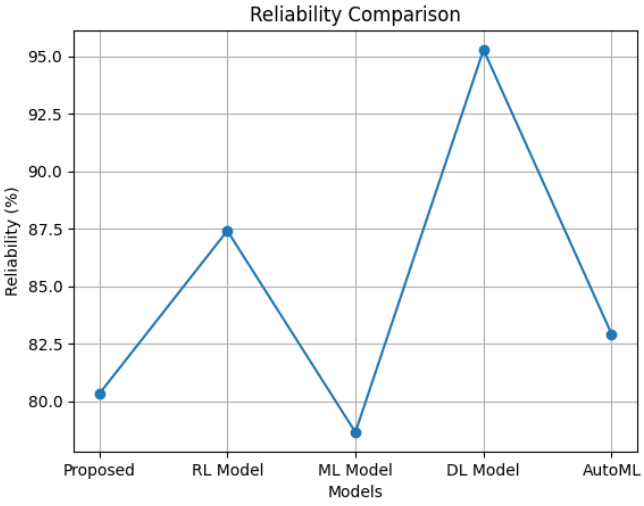

Fig 6: Reliability Comparison of Different Models

In general, the experimental findings indicate that the proposed framework is a strong and dynamic solution to software reliability. The combination of reinforcement learning and generative AI offers continuous improvement and dynamic adaptation, unlike the traditional methods that are based on the use of static models. The fact that concept drift can be identified and the system behavior can be modified in real-time is another benefit that contributes to the efficiency of the suggested framework.

5. CONCLUSION

This paper has presented a Self-Adaptive Software Reliability Framework based on Generative Learning Models to deal with the issue of ensuring reliability in dynamic and uncertain software environments. It involves the combination of reinforcement learning, generative AI, and continuous feedback to make the system adapt intelligently and autonomously. The suggested solution is effective because it improves performance of the system by constantly observing the system behavior, identifying changes and implementing adaptive measures as reliability is maintained. The incorporation of generative learning models enables the system to seek out new adaption strategies and be able to manage unseen situations. Moreover, the consideration of reliability measures like failure rate, availability, and performance guarantee a thorough assessment and enhancement of the quality of the system. The results of the experiment have shown that the proposed framework is more reliable and balanced in its performance than the traditional machine learning and deep learning frameworks. The concept drift and dynamic model updating capability also helps in amplifying the robustness of the system. The suggested framework offers a scalable and smart solution to contemporary software systems that need to be altered continuously and have a high degree of reliability. The next-generation efforts can be directed towards real-time deployment, fine-tuning of generative models, and large-scale integration with distributed systems.

Conflict of interest statement

Authors declare that they do not have any conflict of interest.

REFERENCES

[1] N. Rajasekhar, T. Radhakrishnan, and N. Samsudeen,"Exploring reinforcement learning in process control: A comprehensive survey," Int. J. Syst. Sci., vol. 56, pp. 3528–3557, 2025.

[2] M. F. El-Khatib, F. M. Khater, E. Hendawi, and M. I. Abu El-Sebah,"Simplified and intelligent controllers for multi-input multi-output processes," Eng. Appl. Artif. Intell., vol. 141, p. 109816, 2025.

[3] H. Cheng, S. Jung, and Y. B. Kim,"A novel reinforcement learning controller for the DC-DC boost converter," Energy, vol. 321, p. 135479, 2025.

[4] Z. Wan, X. Xia, D. Lo, and G. C. Murphy,"How does machine learning change software development practices?" IEEE Trans. Softw. Eng., vol. 47, pp. 1857–1871, 2021.

[5] R. R. O. Al-Nima, T. Han, S. A. M. Al-Sumaidaee, T. Chen, and W. L.Woo,"Robustness and performance of deep reinforcement learning," Appl. Soft Comput., vol. 105, p. 107295, 2021.

[6] D. Weyns,Engineering Self-Adaptive Systems: A Short Tour in Seven Waves, 2020.

[7] M. Zhang, J. Li, H. Zhao, K. Tei, S. Honiden, and Z. Jin,"A meta reinforcement learning-based approach for self-adaptive systems," in Proc. IEEE Int. Conf. Autonomic Computing and Self-Organizing Systems (ACSOS), 2021, pp. 1–10.

[8] O. Gheibi and D. Weyns,"Lifelong self-adaptation: Self-adaptation meets lifelong machine learning," in Proc. SEAMS, 2022, pp. 1–12.

[9] R. Calinescu, D. Weyns, S. Gerasimou, et al.,"Engineering trustworthy self-adaptive software with dynamic assurance cases," IEEE Trans. Softw. Eng., vol. 44, pp. 1039–1069, 2018.

[10] P. Arcaini, E. Riccobene, and P. Scandurra,"Modeling and analyzing MAPE-K feedback loops for self-adaptation," in Proc. SEAMS, 2015, pp. 13–23.

[11] C. Pahl and S. Azimi,"Constructing dependable data-driven software with machine learning," IEEE Softw., vol. 38, pp. 88–97, 2021.

[12] C. Pahl,"Research challenges for machine learning-constructed software," Serv. Oriented Comput. Appl., vol. 17, pp. 1–4, 2023.

[13] M. A. Langford and B. H. C. Cheng,"Predicting behavior for learning-enabled systems under uncertainty," in Proc. SEAMS, 2021, pp. 78–89.

[14] R. S. M. Barros and S. G. T. C. Santos,"A large-scale comparison of concept drift detectors," Inf. Sci., vol. 451–452, pp. 348–370, 2018.

[15] L. L. Minku and X. Yao,"DDD: A new ensemble approach for dealing with concept drift," IEEE Trans. Knowl. Data Eng., vol. 24, pp. 619–633, 2012.

[16] T. Cerquitelli, S. Proto, F. Ventura, et al.,"Automating concept-drift detection by self-evaluating predictive model degradation," 2019.

[17] J. Li, M. Zhang, N. Li, D. Weyns, Z. Jin, and K. Tei,"Generative AI for self-adaptive systems: State of the art and research roadmap," ACM Trans. Auton. Adapt. Syst., vol. 19, p. 13, 2024.

[18] D. K. Seth, K. K. Ratra, and A. P. Sundareswaran,"AI and generative AI-driven automation for multi-cloud and hybrid cloud architectures," in Proc. IEEE CCWC, 2025, pp. 784–793.

[19] Bin Zhang, Hangyu Mao, Jingqing Ruan, Ying Wen, Yang Li, Shao Zhang, Zhiwei Xu, Dapeng Li, Ziyue Li, Rui Zhao, Lijuan Li, and Guoliang Fan. 2023c. Controlling large language model-based agents for large-scale decision-making: An actor-critic approach. arXiv:2311.13884

[20] Chenrui Zhang, Lin Liu, Chuyuan Wang, Xiao Sun, Hongyu Wang, Jinpeng Wang, and Mingchen Cai. 2024c. PREFER: Prompt ensemble learning via feedback-reflect-refine. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 38, 19525–19532.

[21] Kluge, T. A Role-Based Architecture for Self-Adaptive Cyber-Physical Systems. In Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '20), Seoul, Korea, 7–8 October 2020; ACM: New York, NY, USA, 2020; pp. 1–5.

[22] Hao Zhang, Hao Wang, and Zhen Kan. 2023d. Exploiting transformer in sparse reward reinforcement learning for interpretable temporal logic motion planning. IEEE Robotics and Automation Letters 8, 8 (Aug. 2023), 4831–4838.

[23] Lei Zhang, Yuge Zhang, Kan Ren, Dongsheng Li, and Yuqing Yang. 2024f. MLCopilot: Unleashing the power of large language

models in solving machine learning tasks. In Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics, Vol. 1: Long Papers. Association for Computational Linguistics, 2931–2959.

[24] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. ExpeL: LLM agents are experiential learners. In Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI '24), Proceedings of the 36th Conference on Innovative Applications of Artificial Intelligence (IAAI '24), Proceedings of the 14th Symposium on Educational Advances in Artificial Intelligence (EAAI '14). AAAI Press, 19632–19642.

[25] Ziyuan Zhong, Davis Rempe, Danfei Xu, Yuxiao Chen, Sushant Veer, Tong Che, Baishakhi Ray, and Marco Pavone. 2023b. Guided conditional diffusion for controllable traffic simulation. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '23), 3560–3566.