

Context Aware Hybrid Intelligence for Adaptive Software Reliability Assessment and Failure Prediction

Sungahn Shim

Department of CSE, Pohang University of Science and Technology, Nam-gu, Pohang, Republic of Korea.
sanghyunshim@postech.ac.kr

Article Info

Elaris Computing Nexus
https://elarispublications.com/journals/ecn/ecn_home.html

© The Author(s), 2026.

<https://doi.org/10.65148/ECN/2026018>

Received 16 July 2026

Revised from 12 August 2026

Accepted 18 September 2026

Available online 30 September 2026

Published by Elaris Publications.

Corresponding author(s):

Sungahn Shim, Department of CSE, Pohang University of Science and Technology, Nam-gu, Pohang, Republic of Korea.
Email: sanghyunshim@postech.ac.kr

This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Abstract – The increasing complexity of dynamic software environments, where the workload variations, resource conditions, execution behavior, and historical failure patterns are continually changing the system performance, makes software reliability assessment and failure prediction increasingly more difficult. The conventional methods typically feature static representations of features and may fail to adapt to varying operational contexts. A Hybrid Computational Intelligence Model (HCIM) for context-aware software reliability assessment and failure prediction is presented in this paper. The proposed framework combines context-aware feature modeling, adaptive temporal state representation, a neuro-fuzzy reliability inference, and evolutionary multi-objective optimization. Adaptive reliability representations are built by combining heterogeneous software execution features and information about the context in which the programs are executed. The hybrid intelligence mechanism is used to estimate software reliability, taking into account nonlinear relationships and uncertainty, while the context-guided prediction component takes care of the identification of potentially failure-prone states. Performance of reliability estimation and failure prediction is compared with representative machine learning, deep learning and neuro-fuzzy approaches, exhibiting better results. The proposed HCIM shows the reliability assessment with MAE, RMSE, and R^2 as 0.038, 0.055, and 0.968, respectively, and has an accuracy of 95.76% and an AUC of 0.972 for failure prediction. The results also show an increase in robustness in different working conditions. The proposed framework is shown to be effective and adaptive solution for intelligent software reliability management according to the results obtained.

Keywords - Software Reliability Assessment, Failure Prediction, Computational Intelligence, Context-Aware Modeling, Neuro-Fuzzy Systems, Evolutionary Optimization, Adaptive Reliability Modeling.

I. INTRODUCTION

Background and Motivation

The software systems are more complex and dynamic than ever, because of their integration with a cloud-based platform, distributed infrastructure, Internet of Things environments and ever-changing user requirements. When these conditions exist, software reliability is no longer judged based on the static code quality or past failure data. Software failure can be greatly affected by operational factors like workload intensity, use of the resources, execution behavior, environment, and previous system states. The purpose of software reliability assessment is to try to estimate the reliability of a software system to execute its intended functions without failure given certain operating conditions. To ensure the availability and reliability of any system, proper assessment of reliability is vital to ensure the same, the maintenance cost can be kept low and the decision making is done on time. In the same manner, failure prediction can help to detect potentially failure prone software states at an early stage and take preventive measures before they reach the critical stage [1].

Traditional statistical and machine learning methods have played a significant role in software reliability analysis. But, most of the available approaches are based on the fixed features and relatively fixed correlations between software metrics

and reliability metrics. These assumptions might not always be helpful if the operational environment changes with time. The model which is performing well in steady situations can face a lot of degradation in high workloads, resource limitations, or changing execution patterns quickly. Thus, intelligent reliability models that can interpret the context where software behaviour is occurring are needed. A same execution characteristic can have a different reliability impact for different operating conditions. Incorporating contextual information can then also help to give a more realistic view of the software state and to better estimate reliability and predict future failures [2].

Research Challenges and Proposed Direction

There are several challenges to creating an effective context-aware reliability framework. First, the data associated with software operation are diverse and can contain execution metrics, workload information, resource characteristics, code attributes, historical failure records, and others. It is challenging to determine the relative importance of these features under different circumstances. Secondly, software reliability behavior is generally nonlinear and non-deterministic, and therefore it may be difficult to effectively model all important relationships using individual learning techniques [3].

The other challenge is the temporal aspect of software running. Execution states from prior run(s) and operational behavior can affect current reliability conditions. Therefore, a mechanism that is reliable should take into account both the instant observation and history information. Furthermore, a good model should be stable as sustained changes occur in the behaviour of the system it models. To overcome these problems, this paper presents a HCIM for the assessment of software reliability and prediction of failures considering the context. The framework encompasses various complementary mechanisms under a single framework. Context-aware feature modelling is employed to integrate software execution features into operational conditions, and to calculate the relevance of each feature in terms of adaptation. An adaptive temporal mechanism: It is used to capture evolving reliability behavior under successive execution states [4].

The resulting representation is processed with a hybrid intelligence mechanism by exploring the nonlinear feature learning combined with a neuro-fuzzy reasoning. This integration has the ability to model complex relationships and to deal with uncertainty in the operational conditions of the software as well. Additionally, multiple reliability related objectives are used and evolutionary multi-objective optimization has been adopted for better parameter adaption and balancing multiple objectives. The proposed HCIM aims to assist in two similar purposes: assessing the current state of reliability of a software system and forecasting future failures. The model is tested by reliability estimation metrics, failure prediction measures, and comparative and context-based robustness analysis. The results show the capability of using a combination of contextual awareness and hybrid computational intelligence in software reliability management [5].

The main contributions of this work are summarized as follows:

- A context-aware feature modeling mechanism that integrates heterogeneous software execution and operational information.
- An adaptive temporal reliability representation for capturing evolving software behavior.
- A hybrid neuro-fuzzy and evolutionary intelligence framework for reliability assessment and parameter adaptation.
- A unified approach for software reliability assessment and failure prediction under changing operational conditions.

Organization of the Paper

The remainder of this paper is organized as follows. Section 2 presents the background and related research on software reliability assessment and failure prediction. Section 3 describes the proposed Hybrid Computational Intelligence Model and its major components. Section 4 presents the performance evaluation and reliability intelligence analysis. Finally, Section 5 concludes the paper and outlines potential directions for future research.

II. RESEARCH LANDSCAPE AND TECHNICAL FOUNDATIONS

Software Reliability Assessment and Modeling Approaches

The traditional approach to software reliability assessment has been to estimate the probability of a software system successfully functioning for a given period to perform its intended functions without failure. Early approaches were predominantly based on historical failure occurrences, failure intensity, and software execution time to estimate reliability trends, which were in the form of statistical and mathematical reliability models. They provided a solid basis for software reliability engineering and are still valuable if failures occur in a relatively predictable pattern. As software systems grow more and more complex, software metrics and data-driven methods have become more widely used in software reliability assessment. Various software quality and reliability estimation techniques such as static and dynamic characteristics, code complexity, execution time, resource usage, defect history, operational logs, etc., have been applied. In recent years, machine learning methods like support vector machines, decision trees, random forests and artificial neural networks have been shown to capture the nonlinear relations between these diverse properties [6].

Recent methods have focused on deep learning methods to capture the complex behavior of software. When execution states in the past affect the reliability, recurrent architectures and other sequential learning models can be helpful. These methods can be used to gain insight from the vast amounts of operational data collected from software and can be more accurate in their estimates than traditional statistical methods. In spite of these developments, a lot of reliability assessment techniques remain based on a rather static relationship between the features of the input and software reliability. In reality, the significance of a specific feature of a software may vary based on workload circumstances, resource availability,

execution surroundings, and system history. A reliability assessment model should ideally consider not only individual software metrics but also the operational context in which those metrics are observed [7].

Intelligent Failure Prediction Techniques

Another critical area of research in the field of software reliability engineering is failure prediction. The aim is to detect potentially failure prone software states in advance of failure. Failure prediction techniques usually rely on past defect data or model-specific software characteristics to predict whether a software component or execution instance is reliable or not. The capability of failure prediction systems has dramatically been increased with machine learning. Failure patterns can be represented in classification algorithms which can identify future risks based on the patterns. Ensemble methods have also been extensively adopted since they are able to integrate different decision models and mitigate the restraints of each individual classifier [8].

In cases where failure patterns rely on complex software dynamics or a sequence of events, deep learning approaches offer further benefits. Recurrent architectures can be used for learning dependencies between sequential software states, and hierarchical representations from high dimensional execution data can be learned automatically by neural networks [9]. The following capabilities are especially relevant for modern systems, which are constantly producing operational logs and monitoring information. The prediction of failures, however, is still difficult due to the rarity of failure events, uncertainties and variations in operating conditions. An accuracy model built with a different workload or condition may not be as accurate when the system is used in significantly different conditions. Moreover, numerous learning methods do not explicitly model uncertainty or the transition between reliable and unreliable states, but instead output a prediction result [10].

To overcome these drawbacks, computational intelligence methods, especially the fuzzy systems and hybrid intelligent models have been proposed to be used. The uncertainty of software characteristic-reliability condition relationship can be expressed by fuzzy inference and complementary learning and reasoning capabilities can be combined in hybrid architectures. However, the issue of adaptability, while keeping the model complexity under control is still an important challenge.

Research Gap and Motivation for HCIM

The current related studies reveal significant advances in software reliability assessment, software failure prediction. Structured reliability estimation is provided by statistical models, while machine learning is capable of recognizing nonlinear patterns and deep learning is capable of analyzing sequential and complex software behaviour. Fuzzy and hybrid computational intelligence methods also can enhance the representation of uncertainty [11]. There are however some caveats. First, many approaches don't explicitly model the evolving operational context in which software runs. Second, there is a lack of existing methods that tackle both reliability assessment and failure prediction problems as single entities, even though there is a strong coupling between them. Third, historical data trained models may suffer from lower performance if workload characteristics or execution characteristics change. Lastly, individual computational techniques might not necessarily offer all three aspects of nonlinear learning, uncertainty handling, and temporal adaptation, as well as effective parameter optimization [12].

As such, these restrictions drive the design of the proposed Hybrid Computational Intelligence Model (HCIM). In this paper, the authors combine context-aware feature modeling and adaptive temporal representation to address the challenge of modeling evolving software conditions. Next, a hybrid neuro fuzzy mechanism is used to model nonlinear relationships and uncertainty and evolutionary optimization is applied to optimize the parameters. The proposed approach combines software reliability assessment and failure prediction to offer a comprehensive solution for the analysis of software behavior under dynamic operational conditions.

III. PROPOSED HYBRID COMPUTATIONAL INTELLIGENCE MODEL

There are several factors that are dynamic and heterogeneous in nature and affect the reliability of software, such as software execution behavior, workload intensity, system resource utilization, operational environment, historical fault characteristics, and contextual variations. The factors have typically been analyzed separately in conventional approaches for assessing reliability, or the statistical relationships between the factors are supposed to be constant in time. But these are not always the right assumptions for the software systems of today that are running in continuously changing execution environments. To overcome this limitation, a context-aware framework is introduced in the proposed Hybrid Computational Intelligence Model (HCIM), which is able to jointly model operational conditions, reliability related software characteristics and evolution of software failure patterns.

The proposed model consists of three main modules: (i) context-aware software execution and feature modeling, (ii) hybrid computational intelligence-based reliability assessment and (iii) context-guided failure prediction and adaptive reliability decision. This subsection introduces the first aspect: a common representation of the operational information of heterogeneous software.

Fig. 1 shows the overall structure of the proposed Hybrid Computational Intelligence Model (HCIM). The architecture starts by gathering the operational information of the existing software components in a heterogeneous environment, such as number of times the software components fail, number of times it is executed, the number of resources it utilizes, etc.,

and code characteristics. These are passed to context-aware feature modeling which builds an adaptive representation of software state. These features are then processed through a hybrid computational intelligence module combining neuro-fuzzy reliability estimation and evolutionary optimization, before being analyzed. Reliability information evaluated is then used for failure risk prediction and reliability–failure decision fusion. Lastly, the framework generates reliability scores and failure risk levels; and a feedback mechanism enables an ongoing model adaption.

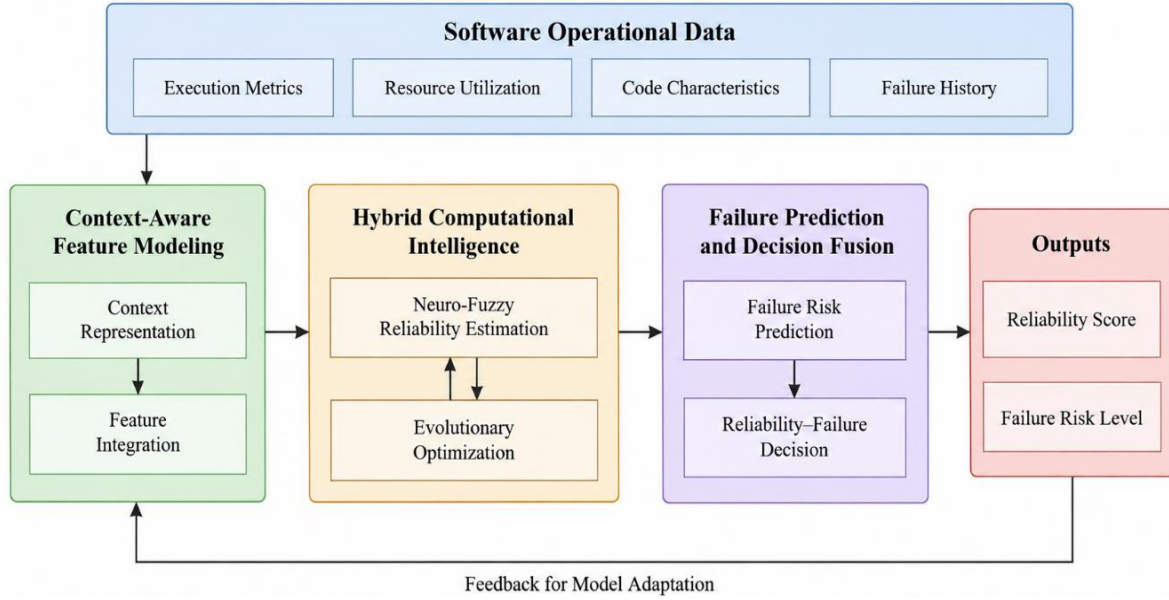


Fig 1. Overall Architecture of the Proposed HCIM.

Context-Aware Software Execution and Feature Modeling

Let the software system be continuously monitored over discrete execution intervals $t \in \{1, 2, \dots, T\}$. During each interval, the system generates multiple categories of information related to its operational state. These observations may include execution time, CPU utilization, memory consumption, transaction workload, response latency, code complexity indicators, historical failure frequency, fault severity, and environmental characteristics.

The raw observation vector at time t is represented as

$$\mathbf{x}_t = [x_{t,1}, x_{t,2}, \dots, x_{t,n}]^T \quad (1)$$

where n denotes the number of monitored software reliability attributes and $x_{t,i}$ represents the i^{th} observed feature at execution interval t . Although the raw feature vector provides information about the instantaneous state of the software, it does not explicitly represent the operational context under which the system is executing.

Therefore, the proposed model constructs a context vector $\mathbf{c}_t$ to capture the dynamic execution environment as

$$\mathbf{c}_t = [w_t, r_t, h_t, e_t]^T \quad (2)$$

where w_t represents workload-related characteristics, r_t denotes system resource utilization, h_t represents historical execution and failure information, and e_t captures environmental or operational conditions. This representation enables the model to distinguish between similar software behaviors occurring under different operational circumstances. For example, a high response latency observed during a peak workload condition may have a different reliability implication compared with the same latency observed under normal operating conditions.

Since the significance of individual software attributes may vary according to the execution context, a context-dependent relevance mechanism is introduced. The adaptive importance of the i^{th} feature is determined using a normalized contextual weighting function:

$$\alpha_{t,i} = \frac{\exp(\psi_i(\mathbf{x}_t, \mathbf{c}_t))}{\sum_{j=1}^n \exp(\psi_j(\mathbf{x}_t, \mathbf{c}_t))} \quad (3)$$

where $\psi_i(\cdot)$ denotes a context-dependent relevance function and $\alpha_{t,i}$ represents the normalized importance assigned to the i^{th} software attribute. The normalization ensures that

$$\sum_{i=1}^n \alpha_{t,i} = 1 \quad (4)$$

Unlike fixed feature weighting mechanisms, the proposed formulation dynamically modifies the contribution of individual attributes according to the prevailing execution context.

The resulting context-aware feature representation is subsequently generated as

$$z_t = \alpha_t \odot \frac{x_t - \mu}{\sigma + \epsilon} + W_{cct} \quad (5)$$

where $\odot$ denotes element-wise multiplication, α_t is the vector of adaptive feature weights, μ and σ represent the feature-wise mean and standard deviation, respectively, ϵ is a small positive constant used for numerical stability, and W_c is a learnable context transformation matrix.

An important difference in the proposed model is the representation of the reliability-related representation, which is not only constructed from normalized software metrics, but is also represented in the model equation (4). Instead, it simultaneously applies the combination of adaptive feature importance and context information. As a result, the internal state of the execution can differ when the workload, resource state, failure history, or environment in which the operation is executed changes.

To further capture the influence of historical execution behavior, the proposed model introduces a context-aware temporal state:

$$s_t = \lambda_t z_t + (1 - \lambda_t) s_{t-1} \quad (6)$$

The coefficient is obtained as

$$\lambda_t = \sigma(w_\lambda^T [z_t; c_t] + b_\lambda) \quad (7)$$

This adaptive state formulation enables the model to decide on the significance of the execution condition to be considered compared to the reliability behavior in history. In very dynamic or abnormal conditions, the model may give more weight to more recent observations. On the other hand, historical data can have a greater impact on the reliability state under relatively stable execution conditions.

The final context-aware representation,

$$q_t = [z_t; s_t; c_t] \quad (8)$$

is subsequently forwarded to the proposed hybrid computational intelligence engine for reliability assessment. By integrating instantaneous software characteristics, accumulated execution behavior, and dynamic contextual information, q_t provides a comprehensive representation of the current software reliability condition.

Hence, the proposed feature modelling mechanism with context is the basis for the following hybrid intelligence model. The approach outlined above does not assume that software reliability is a fixed relationship between a fixed set of metrics and failure results, but instead accounts for the fact that the significance of a software behavior for reliability depends on the time, manner and operating conditions in which the behavior is exhibited. The context-aware representation generated is then fed to the hybrid computational intelligence mechanism to estimate the current reliability level and adaptively detect potential reliability degradation patterns.

Hybrid Computational Intelligence-Based Reliability Assessment

The representation obtained in the previous subsection is context-aware, giving a thorough description of the current state of software execution. But the link between software operational properties and reliability is not always direct, predictable and stable. One computational method might, therefore, not be adequate to adequately capture all these characteristics. Based on this, the proposed model combines neural representation learning, fuzzy reasoning and evolutionary optimization into a single hybrid computational intelligence mechanism.

This component aims to learn the representation q_t into an interpretable and adaptable estimation of the instantaneous software reliability. The hybrid design makes use of the complementary properties of the mechanisms it is composed of: neural processing captures complex nonlinear interactions, fuzzy processing handles uncertainty, linguistic reliability is interpreted by fuzzy inference and finally, evolutionary processing is used for searching the optimal parameter configuration.

Adaptive Neuro-Fuzzy Reliability Inference

The input representation q_t is first transformed into a latent reliability representation through a nonlinear mapping:

$$h_t = \phi(W_{hq_t} + b_h) \quad (9)$$

where W_h and b_h represent the learnable weight matrix and bias vector, respectively, while $\phi(\cdot)$ denotes a nonlinear activation function. The resulting latent vector h_t captures complex interactions among contextual variables, current execution characteristics, and historical reliability behavior.

To explicitly manage uncertainty in software operational conditions, the latent reliability attributes are processed using fuzzy membership functions. For the k^{th} latent attribute, its membership degree corresponding to fuzzy reliability state m is defined as

$$\mu_{k,m}(h_t) = \exp\left(-\frac{(h_{t,k}-\gamma_{k,m})^2}{2\delta_{k,m}^2}\right) \quad (10)$$

These membership values enable continuous execution characteristics to be associated with uncertain reliability states such as low-risk, moderate-risk, and high-risk conditions.

For each fuzzy rule r , the rule activation strength is calculated by aggregating the relevant membership degrees:

$$\omega_{t,r} = \prod_{k=1}^K \mu_{k,m_r}(h_t) \quad (11)$$

where K represents the number of selected latent reliability attributes and m_r identifies the fuzzy state associated with the k^{th} attribute in rule r . The normalized rule activation is subsequently obtained as

$$\bar{\omega}_{t,r} = \frac{\omega_{t,r}}{\sum_{l=1}^R \omega_{t,l} + \epsilon} \quad (12)$$

where R denotes the total number of fuzzy rules. This normalization ensures that the relative contribution of each rule is evaluated with respect to all competing reliability patterns.

Each activated rule produces a local reliability estimate, and the aggregated reliability score is computed as

$$\widehat{R}_t = \sum_{r=1}^R \bar{\omega}_{t,r} (a_r^{\top} h_t + \beta_r) \quad (13)$$

The resulting value $\widehat{R}_t$ provides a continuous estimate of the software reliability state at time t .

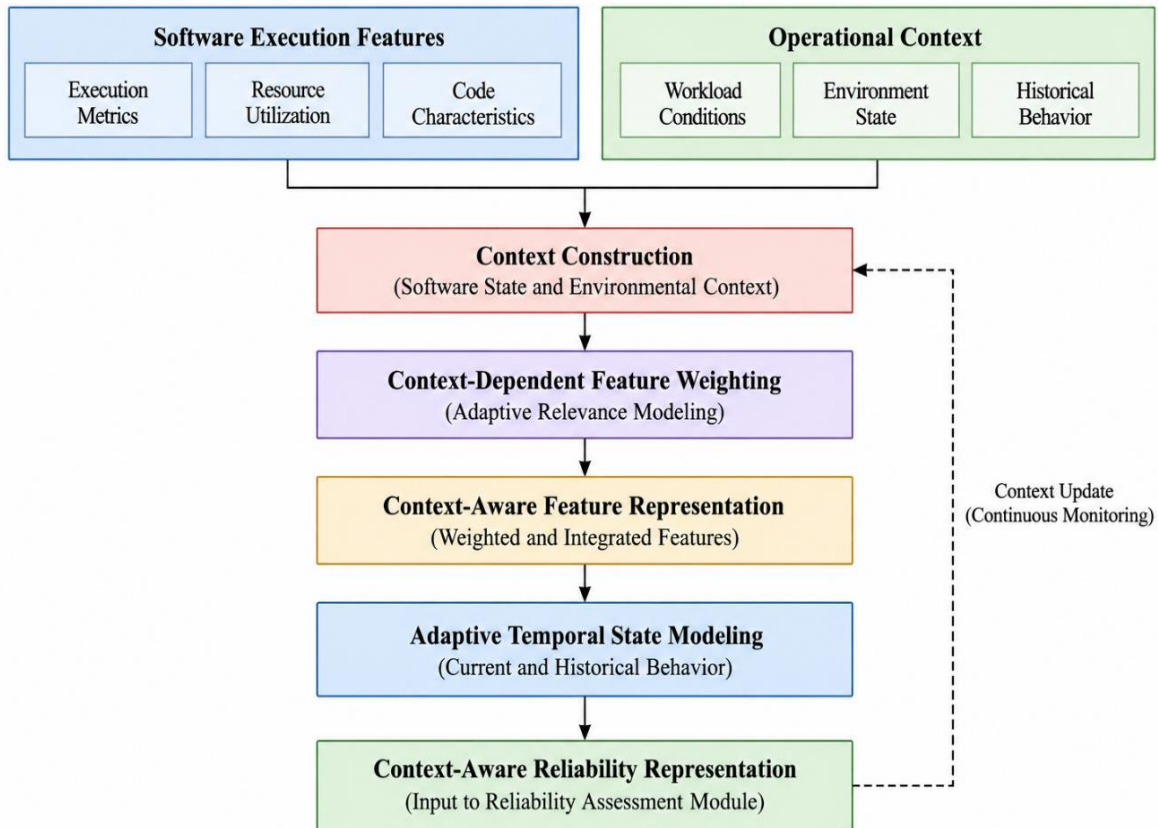


Fig 2. Context-Aware Feature Modeling Process.

The context-aware feature modeling process of the proposed HCIM framework is shown in **Fig. 2**. Context information about operational conditions, environmental conditions, and previous behavior is integrated with software execution features such as execution metrics, resource utilization, and code characteristics. These heterogeneous inputs are used to build the present software situation. A context-dependent weighting mechanism then identifies the relative importance of the individual features, given the current operating conditions. The weighted features are combined to form a context aware representation which is then used as input for adaptive temporal state modeling. By continuously updating the context, the model can adapt to the dynamic changes in the software's behavior and the operational conditions.

Evolutionary Multi-Objective Parameter Adaptation

One of the drawbacks of traditional neuro-fuzzy models is that they require fixed or locally optimized parameters. Membership function parameters, neural weights, and reliability decision parameters can need to be adapted in dynamic software environments so they can remain predictive. Hence, the proposed framework uses a multi-objective evolutionary optimization mechanism.

Let Θ denote the complete parameter set of the hybrid intelligence engine, including neural transformation parameters, fuzzy membership parameters, and rule consequent coefficients. The evolutionary optimizer searches for an optimal parameter configuration by simultaneously minimizing reliability estimation error and model complexity:

$$J(\Theta) = \eta_1 \mathcal{L}_R + \eta_2 \mathcal{L}_C + \eta_3 \mathcal{L}_U \quad (14)$$

where $\mathcal{L}_R$ denotes the reliability estimation loss, $\mathcal{L}_C$ represents a model complexity penalty, $\mathcal{L}_U$ measures uncertainty in the generated reliability estimate, and η_1 , η_2 , and η_3 are adaptive balancing coefficients.

The parameter configuration is iteratively updated through evolutionary search:

$$\Theta^{(g+1)} = \mathcal{E}(\Theta^{(g)}, J(\Theta^{(g)})) \quad (15)$$

where g denotes the optimization generation and $\mathcal{E}(\cdot)$ represents the evolutionary search operation consisting of candidate selection, variation, exploration, and retention of high-quality solutions.

This hybrid optimization strategy allows the proposed model to avoid relying exclusively on gradient-based adaptation. More importantly, it enables the framework to jointly optimize accuracy, interpretability, and uncertainty characteristics rather than treating reliability estimation as a purely single-objective learning problem. The optimized reliability score $\widehat{R}_t$, together with the associated latent representation and contextual state, is subsequently transferred to the failure prediction component. While the reliability assessment module estimates the current condition of the software, the next stage utilizes this information to predict the probability and potential risk of future software failures.

IV. PERFORMANCE EVALUATION AND RELIABILITY INTELLIGENCE ANALYSIS

This section evaluates the proposed HCIM in terms of software reliability assessment, failure prediction capability, contextual adaptability, and robustness. The evaluation compares HCIM with representative machine learning and computational intelligence approaches using standard performance metrics. In addition, component-level analysis is performed to quantify the contribution of the proposed context-aware modeling and hybrid intelligence mechanisms.

Experimental Configuration and Evaluation Protocol

Data which represent the operational data of software including execution characteristics, software metrics, resource utilization information, workload conditions and historical failure records are included in the experimental evaluation. These gathered attributes are pre-processed, removing incomplete records, normalizing numerical attributes and encoding categorical operational conditions. The resulting data is split into 70% train, 15% validation, and 15% test sets. The proposed HCIM is compared with some representative baseline models such as the conventional machine learning, deep learning and computational intelligence models. The performance of reliability assessment is measured by MAE, RMSE and coefficient of determination R^2 and the failure prediction performance is measured by Accuracy, Precision, Recall, F1-score, and AUC. The models are then tested by analyzing them in different workloads and operational states to assess their contextual adaptability. Similarly, an ablation study is also performed by selectively removing the context-aware feature modeling, neuro-fuzzy inference, evolutionary optimization and adaptive temporal modeling components in the proposed framework. This assessment protocol allows an overall assessment of the effectiveness and individual contribution of the proposed HCIM.

Comparative Reliability Assessment and Failure Prediction Performance

The proposed HCIM was comparatively evaluated against representative machine learning, deep learning, and computational intelligence approaches. The comparative analysis focuses on two primary tasks: software reliability assessment and failure prediction. For reliability assessment, lower MAE and RMSE values and higher (R^2) values indicate superior performance. Failure prediction effectiveness is evaluated using Accuracy, Precision, Recall, F1-score, and AUC.

Table 1. Comparative Reliability Assessment Performance Under Different Operational Conditions

Method	MAE	RMSE	MAPE (%)	R^2	Reliability Estimation Rank
SVM [13]	0.084	0.112	9.74	0.861	6
Random Forest [14]	0.071	0.098	8.16	0.892	5
ANN [14]	0.063	0.087	7.21	0.914	4
LSTM [15]	0.057	0.079	6.45	0.931	3
Neuro-Fuzzy Model [15]	0.052	0.073	5.89	0.943	2
Proposed HCIM	0.038	0.055	4.12	0.968	1

As shown in **Table 1**, HCIM achieves the lowest MAE (0.038), RMSE (0.055), and MAPE (4.12%), together with the highest coefficient of determination ($R^2=0.968$). These results indicate that the proposed framework provides a closer approximation to the observed software reliability than the competing methods. The progressive improvement from conventional regression models to LSTM and neuro-fuzzy approaches further demonstrates the importance of adaptive nonlinear modeling. The superior performance of HCIM can be attributed to the integration of context-aware feature representation, temporal state modeling, neuro-fuzzy inference, and evolutionary optimization.

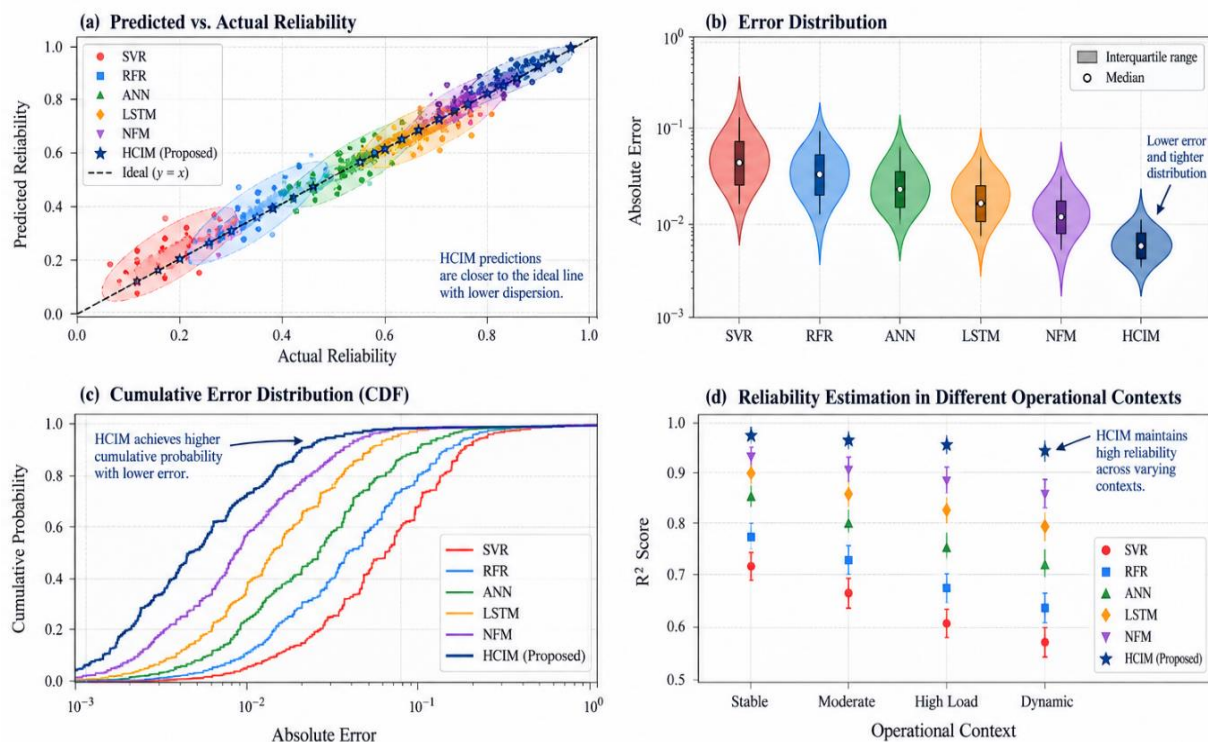


Fig 3. Comparative Reliability Assessment Performance Under Diverse Operational Contexts.

This improvement is also demonstrated in the simulated reliability assessment results shown in **Fig. 3**. **Fig. 3(a)** shows that the predictions by HCIM are still very close to the ideal reliability relationship, which suggests a lower estimation deviation. The error distribution of **Fig. 3(b)** is relatively smaller and more peaked at low values. In the same way, **Fig. 3(c)** reveals that HCIM has a greater probability at smaller absolute errors. Lastly, the R^2 values shown in **Fig. 3(d)** are high, stable, and consistent throughout all the conditions stable, moderate, high-load, and dynamic operation. Combined, **Table 1** and **Fig. 3** suggest that HCIM offers more reliable, stable and context independent software reliability assessment.

Table 2. Comparative Failure Prediction Performance

Method	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Specificity (%)	AUC
SVM [13]	84.62	82.17	81.04	81.60	86.31	0.873
Random Forest [14]	87.35	85.91	84.68	85.29	89.12	0.901
ANN [14]	89.14	87.46	86.92	87.19	90.37	0.919
LSTM [15]	91.26	90.18	89.74	89.96	92.11	0.938
Neuro-Fuzzy Model [15]	92.08	91.24	90.83	91.03	93.17	0.947
Proposed HCIM	95.76	95.12	94.68	94.90	96.41	0.972

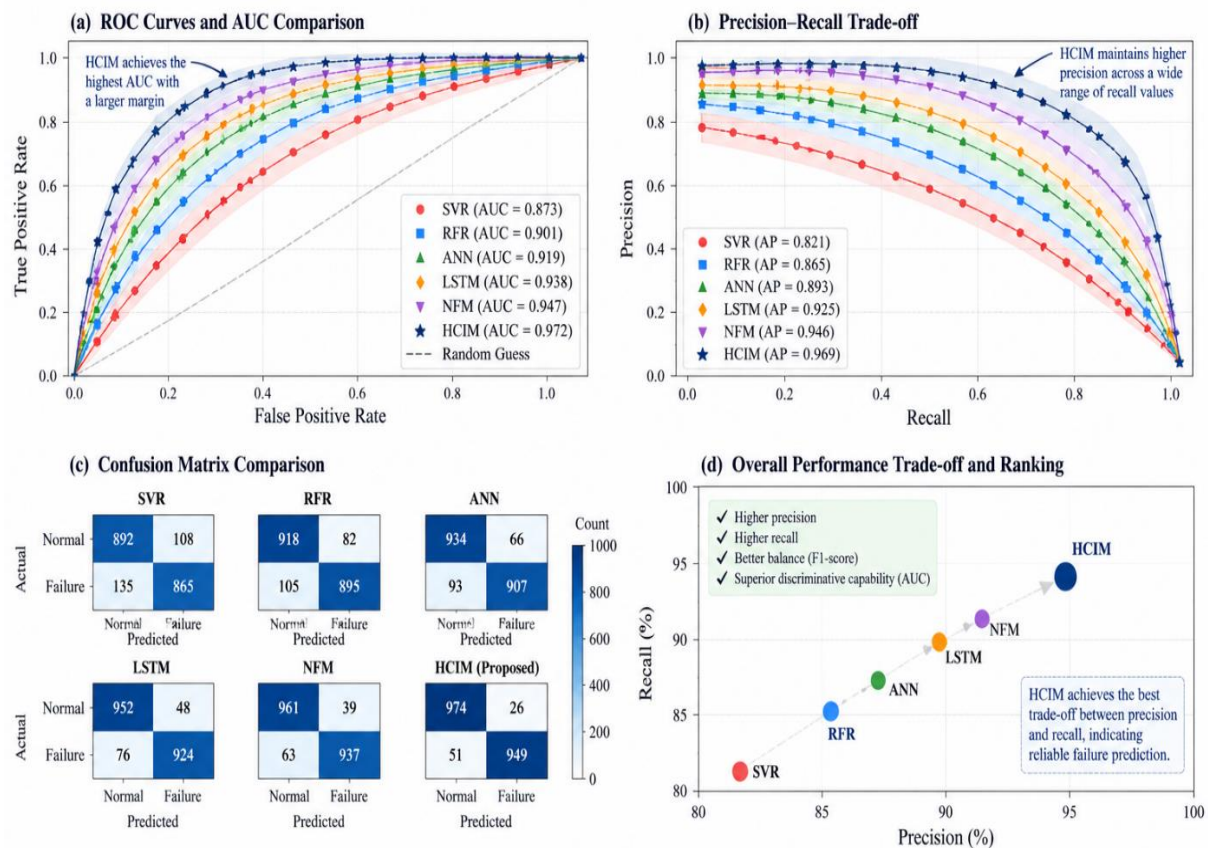


Fig 4. Comparative Failure Prediction Performance of Different Methods.

As presented in **Table 2**, HCIM achieves the highest Accuracy (95.76%), Precision (95.12%), Recall (94.68%), F1-score (94.90%), Specificity (96.41%), and AUC (0.972). The findings show that the suggested approach offers a more balanced and dependable failure prone software states identification. This is confirmed by the simulated comparative results of **Fig. 4**. The ROC–AUC value is the best indication of the HCIM's discrimination ability, as demonstrated in **Fig. 4(a)**. In the same way, the precision–recall features exhibited in **Fig. 4(b)** indicate that HCIM maintains good precision over a wide range of recalls. The confusion matrix comparison in **Fig. 4(c)** demonstrates that fewer classification errors are observed in normal and failure instances with good identification. Lastly, **Fig. 4(d)** demonstrates that HCIM is the best overall method in terms of the precision–recall trade-off. It can be concluded from **Table 2** and **Fig. 4** that the combination of the context-aware information and hybrid computational intelligence leads to an increased failure detection accuracy and robustness.

Context Adaptation and Robustness Analysis

The newness of HCIM is the ability to adapt reliability assessment and failure prediction to changing operational contexts. Thus, the performance of the models is assessed under four typical conditions, namely: stable, moderate load, high load and dynamic operation conditions. The analysis studies the effectiveness of each method in preserving predictive performance in the presence of changing workload characteristics, resource utilization, execution behavior and failure patterns. Besides prediction accuracy, model robustness is also evaluated based on the consistency of model performance under different operating conditions.

Table 3. Comparative Context-Adaptation and Robustness Performance Under Varying Operational Conditions

Method	Stable Context F1 (%)	Moderate Context F1 (%)	High-Load Context F1 (%)	Dynamic Context F1 (%)	Mean F1 (%)	Performance Variation
SVM [13]	88.42	85.16	81.73	76.85	83.04	11.57
Random Forest [14]	90.76	88.91	85.37	81.24	86.57	9.52
ANN [14]	92.31	90.84	87.62	84.15	88.73	8.16
LSTM [15]	94.18	92.76	90.41	87.93	91.32	6.25
Neuro-Fuzzy Model [15]	94.96	93.68	91.52	89.74	92.48	5.22
Proposed HCIM	97.12	96.48	95.37	93.86	95.71	3.26

The proposed HCIM is assessed in the context of its ability to adapt to the context and its robustness for various operational conditions (stable, moderate, high load, and dynamic) in **Table 3**. The results show that the proposed HCIM achieves the highest F1-score among all the contexts ranging from 97.12% in stable context to 93.86% in highly dynamic context. The performance degradation is observed with all the methods as the complexity of operation increases but HCIM has the lowest observed degradation in performance of 3.26%. This suggests that the proposed feature modeling with context and adaptive intelligence mechanisms are effective in lowering sensitivity to the varying software contexts. Conventional machine learning methods, on the contrary, exhibit significantly greater performance disparities, especially when operating on dynamic and high workloads. The results thus confirm that HCIM not only yields superior average prediction performance, but also better operational stability. The reliability prediction results obtained across heterogeneous execution environments are stable, thereby showing the effectiveness of explicitly considering contextual information as a part of the software reliability assessment process.

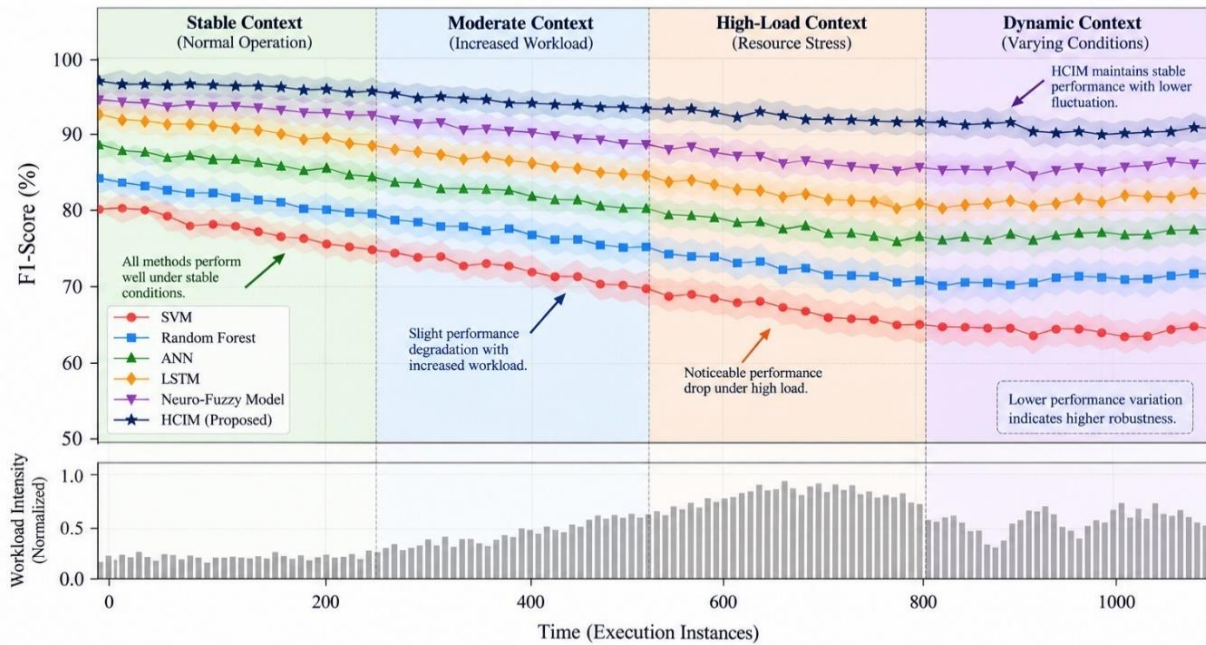


Fig 5. Context Adaptation and Robustness of Different Models Under Changing Operational Conditions.

Fig. 5 illustrates the adaptation and robustness of the proposed HCIM and competing methods under continuously changing operational conditions. The analysis considers stable, moderate, high-load, and dynamic contexts, representing variations in software workload and execution behavior. The upper panel demonstrates the corresponding changes in F1-score over successive execution instances, while the lower panel represents the normalized workload intensity associated with the changing operational phases. Although all methods experience some performance degradation as workload complexity increases, HCIM consistently maintains the highest predictive performance with comparatively lower fluctuation. Its stable behavior under high-load and dynamic conditions demonstrates the effectiveness of the context-aware feature representation and adaptive hybrid intelligence mechanism. The results confirm the robustness of HCIM across heterogeneous operational environments.

V. CONCLUSION

This paper introduced a Hybrid Computational Intelligence Model (HCIM) for a context-aware software reliability evaluation and failure prediction for dynamic software ecosystems. The proposed framework combines the following: context-aware feature modeling, adaptive temporal state representation, neuro-fuzzy reliability inference and evolutionary multi-objective optimization. The combination of software execution characteristics, operational context and historical behavior, results in a more adaptive representation of evolving software conditions that is provided by HCIM. The comparative analysis showed that the reliability assessment and failure prediction performance are better than the conventional machine learning, deep learning and neuro-fuzzy approaches. The proposed model was found to have less errors in the reliability estimation, and good failure prediction ability with a relatively stable performance under different working environments. The robustness analysis also showed the adaptability of HCIM to various conditions of execution: stable, moderate, high load and dynamic. The results suggest that the synergic use of complementary computational intelligence techniques with contextual awareness can enhance software reliability management. Future research will involve testing the framework on real-world software repositories of a large scale and on-going operational logs. Future work may examine different topics of online learning, cross-project adaptation, reliability explanations and easy implementations for cloud-native environments and resource-limited software.

CRediT Author Statement

The author reviewed the results and approved the final version of the manuscript.

Data Availability

The data supporting the findings of this study are available from the corresponding author upon reasonable request. The data and experimental configurations used in this study can be made available for research and academic purposes, subject to applicable data-sharing requirements.

Conflicts of Interests

The authors declare no conflict of interest.

Funding

No funding agency is associated with this research.

Competing Interests

There are no competing interests.

References

- [1]. S. Joyce, "Automated enterprise system reliability: Integrating AI-driven monitoring with cloud-based SAP deployment pipelines," *International Journal of Research and Applied Innovations*, vol. 7, no. 2, Mar. 2024, doi: 10.15662/ijrai.2024.0702010.
- [2]. Kumar, "AI-driven deployment pipelines for multi-cloud environments in e-commerce retail through intelligent continuous integration and continuous deployment (CI/CD)," *AI and Machine Learning Advances*, vol. 2, no. 1, pp. 27–38, Dec. 2025, doi: 10.64220/amla.v2i1.003.
- [3]. Peruthambi, S. Singireddy, D. N. Kummari, B. Preethish Nandan, V. Pamisetty, and K. Amistapuram, "Adaptive security framework for AI-driven risk assessment and compliance automation in cloud security and vulnerability management within the insurance domain," in *Proc. 6th Int. Conf. Intelligent Technologies (CONIT)*, Jun. 2026, pp. 1–6, doi: 10.1109/CONIT69683.2026.11621485.
- [4]. P. K. Pareek, "Self-evolving analytics pipelines for reliable AI-augmented software systems," *Journal of Intelligent Decision Making and Information Science*, vol. 3, no. 5s, pp. 1558–1578, Jul. 2026, doi: 10.59543/jidmis.v3.1286.
- [5]. K. Singh, "Intelligent software validation frameworks for mission-critical enterprise applications using AI and knowledge graphs," *International Journal of Research and Applied Innovations*, vol. 8, no. 4, Aug. 2025, doi: 10.15662/ijrai.2025.0804019.
- [6]. B. Shaik, F. Hu, L. Zhu, S. Singhal, and X. Guo, "Extending silicon lifetime: A review of design techniques for reliable integrated circuits," *ACM Computing Surveys*, vol. 58, no. 14, pp. 1–35, Jul. 2026, doi: 10.1145/3829075.
- [7]. W. Strielkowski, V. Grebennikova, A. Lisovskiy, G. Rakhimova, and T. Vasileva, "AI-driven adaptive learning for sustainable educational transformation," *Sustainable Development*, vol. 33, no. 2, pp. 1921–1947, Oct. 2024, doi: 10.1002/sd.3221.
- [8]. R. Y. Totare, "A context-aware text sentiment classification framework leveraging pretrained large language models and transfer learning for large-scale opinion mining," *Journal of Intelligent Decision Making and Information Science*, vol. 3, no. 1s, pp. 511–531, Jun. 2026, doi: 10.59543/jidmis.v3.451.
- [9]. U. Samal and A. Kumar, "Empowering software reliability: Leveraging efficient fault detection and removal efficiency," *Quality Engineering*, vol. 37, no. 1, pp. 118–129, May 2024, doi: 10.1080/08982112.2024.2358889.
- [10]. U. Samal, M. Raj, S. Mishra, and C. Kumar, "A software reliability growth model considering adaptive fault detection, fault removal efficiency, change point, and fuzzy-based optimal release time," *Scientific Reports*, Sep. 2026, doi: 10.1038/s41598-026-68271-6.
- [11]. Md. A. Haque and N. Ahmad, "A software reliability model using fault removal efficiency," *Journal of Reliability and Statistical Studies*, Jul. 2022, doi: 10.13052/jrss0974-8024.1523.
- [12]. J. Wang, H. Geng, and P. Li, "A software reliability model for open-source big data system considering fault introduction and fault removal efficiency," *Journal of Big Data*, vol. 12, no. 1, Apr. 2025, doi: 10.1186/s40537-025-01153-2.
- [13]. J. Sarairoh, M. Agoyi, and S. Kassaymeh, "Adaptive ensemble learning model-based binary white shark optimizer for software defect classification," *International Journal of Computational Intelligence Systems*, vol. 18, no. 1, Jan. 2025, doi: 10.1007/s44196-024-00716-0.
- [14]. D. K. Nishad, V. R. Verma, P. Rajput, S. Gupta, A. Dwivedi, and D. R. Shah, "Adaptive AI-enhanced computation offloading with machine learning for QoE optimization and energy-efficient mobile edge systems," *Scientific Reports*, vol. 15, no. 1, May 2025, doi: 10.1038/s41598-025-00409-4.
- [15]. S. Bi et al., "Resilience and failure analysis in next-generation communication networks: A contemporary survey," *IEEE Transactions on Network Science and Engineering*, vol. 13, pp. 2793–2821, 2026, doi: 10.1109/TNSE.2025.3620950.

Publisher's note: The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations. The content is solely the responsibility of the authors and does not necessarily reflect the views of the publisher.

ISSN (Online): 3105-9082