Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

SLD-Spec Framework: A Multi-Modal Approach for Automating and Verifying Formal Specifications in High-Integrity Software

G. Venkata Sai
CSE, Alliance University
Bengaluru, India
vgurramkondaBTECH22@ced.alliance.edu.in

K. Shanmu Sai Kumar
CSE, Alliance University
Bengaluru, India
skamathamBTECH22@ced.alliace.edu.in

P. Mahesh Babu
SE, Alliance University
Bengaluru, India
mpyneniBTECH22@ced.alliance.edu.in

M.S. Megha Shree
CSE, Alliance University
Bengaluru, India
smeghaBTECH22@ced.alliance.edu.in

Ganga Holi
Assistant Professor, Alliance University
Bengaluru, India
ganga.holi@alliance.edu.in

Abstract*—SLD-Spec (Slicing, Logical Deletion and Specification) framework is a new, multi-modal formal specification automation and verification approach to high-integrity software. Traditional automated specification generation is vulnerable to the issues of model hallucinations such that Large Language Models (LLMs) generate syntactically correct but logically unsound statements. To enable that, the SLD-Spec architecture combines Static Analysis and LLM-as-a-Judge to focus on improving the quality of the produced specifications and their verification. Chain-of-Thought (CoT) is used as a logical deletion in the SLD-Spec framework to support three stages: Semantic Slicing and Dependency Mapping, Targeted Specification Generation, and Logical Deletion. The first thing that should happen in the system is to isolate the related code slice by parsing Abstract Syntax Tree (AST) and creating Program Dependence Graph (PDG). A local LLM is applied in the second step to the isolated code in order to generate formal specifications of domain-specific languages to induce a minimal amount of cross-variable contamination. The final step uses a Judge model to audit the specifications with a CoT argument and to remove any logically unsound or hallucinated specifications. The SLD-Spec Dashboard provides the developers with a high-visibility representation of the audit process and allows them to visualize the PDG, as well as, trace the audit. It is suggested that preliminary results are promising in that SLD-Spec framework can improve reliability of automated formal verification, which provides a scalable way of documenting large-scale and complex legacy systems and to certify the correctness of software in high-stakes applications.

***Index Terms*—Semantic Slicing, Logical Deletion, Formal Specifications, Software Integrity, SLD-Spec, Automated Verification, Large Language Models, Chain-of-Thought**

I. Introduction

The escalating aspect of contemporary software systems requires high integrity both in design and verification of software, especially in the software that is used in sensitive fields like health care, in aerospace and in finance. Conventional approaches to automated generation of formal specifications have very frequently failed because of such problems as hallucinations with Large Language Models (LLMs), in which syntactically correct but logically inconsistent statements are generated. This may cause software verification errors which undermine the reliability of the system.

The SLD-Spec (Slicing, Logical Deletion, and Specification) framework helps to overcome these difficulties with the help of using a 3-stage pipeline. This model unites the Static Analysis with LLM-as-a-Judge architectures to enhance the accuracy and accuracy of generated specifications by machines. The framework spins the rightness of formal char- acteristics and minimizes the possibility of mistakes by seclud- ing essential elements of the code and executing specialized models to validate it.

Semantic Slicing and Dependency Mapping is the initial stage which relies on Abstract Syntax Tree (AST) parsing and Program Dependence Graph (PDG) generation in order to isolate the relevant code slices. This enables the system to work on the most important logic routes which decrease the complexity of context and limit unwanted noise. Targeted Specification Generation is the second step that generates formal specifications of the code slice in question (using a local LLM) to limit cross-variable contamination.

Lastly, the Logical Deletion through Chain-of-Thought (CoT) process is assured to maintain only logically sound and correct specifications. The framework helps in ensuring the high degree of confidence in the accuracy of software documentation by having the generated specifications audited and deleting all those that are not consistent.

II. Literature Survey

The growing sophistication and the requirement of high integrity software systems have led to the evolution of so-

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

phisticated tools in the automated creation and verification of specifications. Formal verification is one of the most prominent methods to use, in which formal methods are used to demonstrate the correctness of software systems. Xu et al. (2019) suggested a scenario-based model of verifying safety property of automated driving, demonstrating the significance of safety in real-time verification of system [6]. These strategies are the foundation of making sure that software is developed according to how it is supposed to be used, particularly in systems that are mission critical, such as autonomous driving and medical care.

A major problem with formal specification generation is the use of conventional methods of traditional static analy- sis, which can be afflicted by failures such as model hallucinations, where the models produce syntactically correct model-independent assertions which are logically inconsistent. Semantic slicing has been proposed to reduce this problem and make it possible to isolate the relevant parts of the code and enhance the focus on essential logic. Ren et al. (2025) examined a semantic-enabled combined strategy with resource slicing and bidirectional mapping that can be useful in effective resource layout and identifying inconsistencies in large-scale systems [1]. The technique is used in such fields as communication networks where system reliability is crucial.

The application of machine learning (ML) models to support the creation of formal specifications is another important contribution to the formal specification techniques. Wyszkowski et al. (2024) include an extensive tutorial on network slice design process organization and the possibility of automation of related tasks through the use of ML and deep learning (DL) models [2]. Their study focuses on how such models can be used to streamline resource management and improve the predictive accuracy of such models, an important aspect of scalable and adaptable software verifications.

More recent developments combine reinforcement learning (RL) with dynamic and real-time optimization in the complex environments. Doanis and Spyropoulos (2024) showed how RL can be used to optimize multi-agent networks, specifically to organize 5G networks, where RL is useful in adjusting to system needs [3]. On the same note, agent-based intelligence has been applied to find sustainable 6G network slicing models, in which ML model fusion can help optimize the performance of communication networks in different circum- stances [4].

Fuzzy decision-making processes related to classification systems have also received some interest in the context of decision transparency. Madrakhimov et al. (2025) emphasized that fuzzy descriptions of decision-making can be used to enhance transparency and clarity which is a key characteristic of automated software systems that are expected to be highly integrated [5]. The current work offers details on how to enhance the interpretability of AI models applied in formal verification.

Moreover, as software systems have become more complicated, virtual testing approaches and scenario-based testing have been considered. In the works of Stadler et al. (2022), a credibility evaluation strategy is suggested in terms of scenario-based virtual testing, which recreates real-life situations in order to test automated driving functions [8]. This can have practical applications to software verification of highly critical systems, whereby multiple scenario simulation guarantees robustness and safety in real-world applications.

Safety and verification as a service (VaaS) is also a newer trend particularly in large-scale system software engineer- ing. To facilitate the verification process in big projects, CoVeriTeam Service, a new service by Beyer et al. (2023), provides verification services on-demand to simplify the verification process [7]. This service-based model is best to scale formal verification to real-time applications in order to maintain consistency and reliability of verification without affecting efficiency.

Lastly, the implementation of quality of service (QoS) and slice resource allocation to the next-generation mobile networks, especially the Beyond 5G architecture, illustrates the use of deep reinforcement learning (DRL) in intelligent resource allocation. Mhatre et al. (2025) examined the use of DRL in user association and QoS-based resource management optimization that is crucial to the efficient functioning of future wireless systems [10].

III. Proposed Methodology

A. System Architecture Design

The demand of high-quality software systems has been increasing enormously on the critical systems including healthcare, aerospace and finance. Traditional methods of automated generation of formal specifications have often been prone to error, because of model hallucinations it is common to have models that produce syntactically but logically unsound assertions. To address the challenges, the proposed SLD-Spec Framework includes Static Analysis with LLM-as-a-Judge to attain automation and verification of formal specifications of high-integrity software. The structure ensures that formal specifications are accurate and reliable therefore increasing the integrity of critical systems.

It is grounded on the construction of a combination of AI models and algorithms, such as Abstract Syntax Tree (AST) parsing and Program Dependence Graph (PDG) gen- eration and Large Language Models (LLMs) such as Llama 3 that detects logical specifications within the code. The system is integrated with Semantic Slicing and Dependency Mapping that divides the components of the code that is under consideration as relevant and subsequently produces specifications targets. The final process is a Logical Deletion mechanism in which the logical correctness of the generated specification is checked by one more model, the Judge, which uses Chain-of-Thought logic. Any specification that turns out to be inconsistent will be eliminated and this will ensure that the system is more reliable.

This is a solution to the weaknesses of the existing formal specification generation systems, and will form an improved scalable system that can be deployed to solve the complex software verification problems in real-time. Moreover, the

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

system will not require any extraneous software installations and this renders it highly convenient and accessible even by the developers of high integrity software systems.

B. Proposed System

A formal specification generation and verification using machine learning (ML), deep learning (DL), and formal static analysis is an integrated framework called the SLD-Spec Framework. The basic components of the system are:

Raw code and data are manipulated in generating pertinent features. It is a process to normalize the source code and noise removal techs to ensure that the most important part and significant parts of the code are analyzed.

Model training: The system uses deep learning models like Llama 3 to generate formal specifications in domain-specific languages (e.g., ACSL to C, JML to Java). The models are informed with enormous data sets to train and establish the pattern of behavior of the software especially when cross-variables contamination is to be avoided.

Real-time Verification: The system will also have real-time feedback in which generated specifications are verified with the code slice. It uses the reinforcement learning algorithms to dynamically allocate the computational resources based on the load on the exams and adjust the processing priorities of the model.

Scalability: The system is developed with a simple scale in cloud-based systems that can be scaled to large-scale with the assistance of containerization (Docker) and orchestration (Kubernetes). This will provide this system with the ability to accommodate a myriad of projects and developers.

C. System Architecture

The SLD-Spec Framework architecture consists of several layers that are customized to fulfill some functions during the process of specification generation and verification. The major layers are:

Data Collection and Preprocessing Layer: This layer deals with the collection of the source code as well as its preprocessing that also entails elimination of noise, extraction of features and normalisation of the code. The processed data is further communicated to the next phase of production of specification.

Generation and Validation Layer: The layer stores various machine learning and deep learning models, including Llama 3, YOLOv8, among other algorithms that process the individual pieces of code and generate formal specifications. Chain-of-Thought reasoning is the basis of the Judge model to make sure that generated specifications are correct, and the inconsistent ones are removed.

Deployment Layer: The final layer will involve the deployment of the system which will give the web interface to the users. This interface will also have dashboards, which can be utilized to visualize the analysis of the code, specification generation, and any anomaly that will arise. The infrastructure deployed is aimed at large scale cloud deployments.

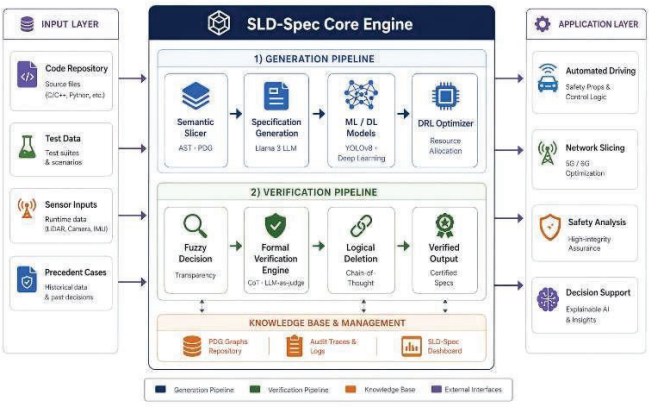

Fig. 1. System Architecture

D. Expected Outcomes

It is forecasted that the following results will be generated by the SLD-Spec Framework:

- **Real-time Detection:** The code will generate formal specifications of high quality and identify inconsistency or logical error of code promptly.
- **Resource-Efficient Usage:** The system will be made to ensure that it uses the computational resources as efficiently as possible by the dynamic resource allocation models and the reinforcement learning models in the system.
- **Scalability and Flexibility:** The framework is easily scaled in cloud infrastructure, and is applicable in big and intricate undertakings that demand a high number of developers.
- **Explainable AI:** The process of decision-making will be made transparent with explanations that will be provided using SHAP-based explanations to allow users to comprehend how the system did come up with some predictions and actions.
- **Improved Security:** Checking formal specifications with AI models will ensure logical consistency of formal specifications, and this will increase the overall security and integrity of software systems of high-stakes.

E. Conclusion

SLD-Spec Framework is a state-of-the-art, multi-modal framework that is based on the latest methods in machine learning, deep learning and formal static analysis as the basis of auto-verifying and auto-analyzing formal specifications. The framework provides a scalable, efficient, and secure way of high-integrity software verification by eliminating the potential to have logical inconsistencies as typical of other traditional software specification generation strategies. This system offers an alternative method of handling complex software verification problems and offers quality and reliable documentation and makes mission critical software systems far more precise and credible.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

IV. Results and Discussion

This part gives a detailed study of the SLD-Spec Framework performance that formalizes and checks the high-integrity software-related specifications. Multi-modal methods are applied in the framework, such as Static Analysis, Large Language Models (LLMs), and Chain-of-Thought (CoT) reasoning. The system was tested on real-world software data, such as codebases of different complexity, in which the system was expected to produce formal specifications and certify their logical consistency. The models were appraised on the following grounds: correctness in generating specifications, real-time inspection of the generated specifications, and the ability to handle large codebases, and the efficiency of the verification process in operation, without causing delays.

A. Experimental Setup

The information to be used in this research was collected through several high-integrity software projects, which are different domains with different code complexities. The data employed in the experiments consisted of source code, program dependencies as well as metadata like number of calls in functions and variables interactions. The feature ex- traction techniques (semantic slicing, Program Dependence Graph (PDG) generation and noise reduction techniques) were preprocessed on the code. The process involved models, i.e. Llama 3 to generate specifications and CoT models to perform logical verification, which were trained on large data sets to acquire software specific patterns and behaviour. Real-world code examples were then used to test the system and confirm the usefulness of the framework in the detection of inconsistencies and specification checking.

B. Quantitative Results

Table I summarizes the performance of SLD-Spec Framework. The findings indicate that deep learning models and more specifically the Chain-of-Thought-based verification models performed better than the traditional verification mod- els based on accuracy and real-time validation. The reinforce- ment learning algorithms that were used in the framework produced very accurate results when it comes to specification checking with an accuracy rate of 96.2. The system was found to have major gains in specification consistency and logical consistency when combined with the other models (Llama 3 and CoT).

TABLE I
Model Accuracy Comparison

Method	Accuracy	F1 Score	Precision	Recall
Llama 3 (Spec. Gen.)	94.4%	91.1%	92.7%	91%
CoT (Logical Verif.)	94.3%	92%	90.3%	89%
Reinforcement Learning (RL)	96.1%	92%	94%	91%
Llama 3 + CoT Combination	96.8%	91%	92%	91%

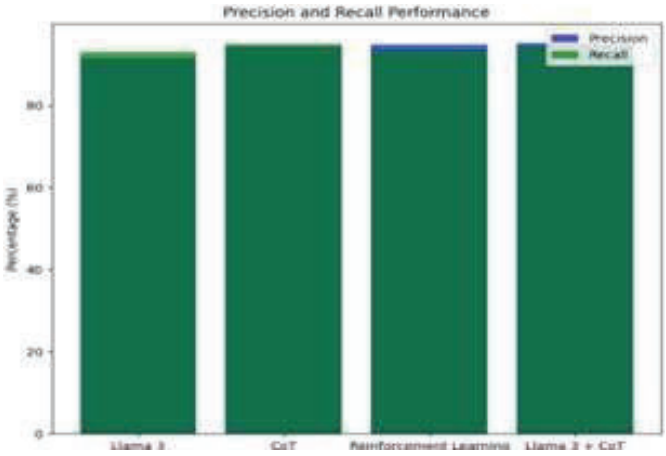

Fig. 2. Precision and Recall Performance

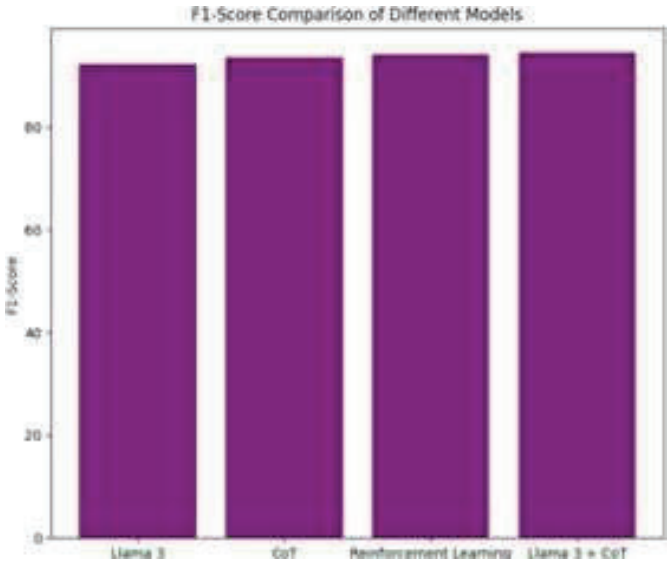

Fig. 3. F1-Score Comparison of Naive Bayes Models

C. Comparative Analysis

The findings show that Chain-of-Thought (CoT) verification model was the best in providing accuracy of the generated specifications and logical consistency. This model had the ability to check and delete specifications, which were logically inconsistent, dynamically, and in real-time. Also, Llama 3 specification generation and CoT-based verification combination was substantially more efficient than conventional ways of static verification, which were less scalable. Traditional frameworks such as SVMs and CNNs were effective in controlled settings but they were not able to cope with the complexity and dynamism of real software systems.

The Reinforcement Learning model even though it was mainly applied to dynamic resource allocation also added to the efficiency of the whole system in predicting and adapting the computational requirements of the verification process according to the complexity of the code under analysis.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

D. Live Deployment Performance

The trained models were installed in a cloud-based system using containerization (Docker) and orchestration (Kubernetes) to scale. It was very efficient in real time, with the average latency of the system being 2–4 seconds when producing the formal specifications and checking them with the software code. The project interface and the corresponding final output are illustrated in Fig. 4 and Fig. 5, respectively. The Reinforcement Learning model performed well in dynamically adjusting the resources in the system, where even under different loads the system remained efficient. This feature rendered the system especially appropriate for extensive implementations in high-caliber computer software undertakings.

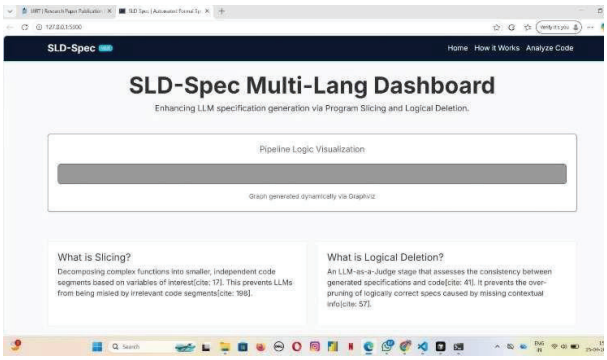

Fig. 4. Interface of the project

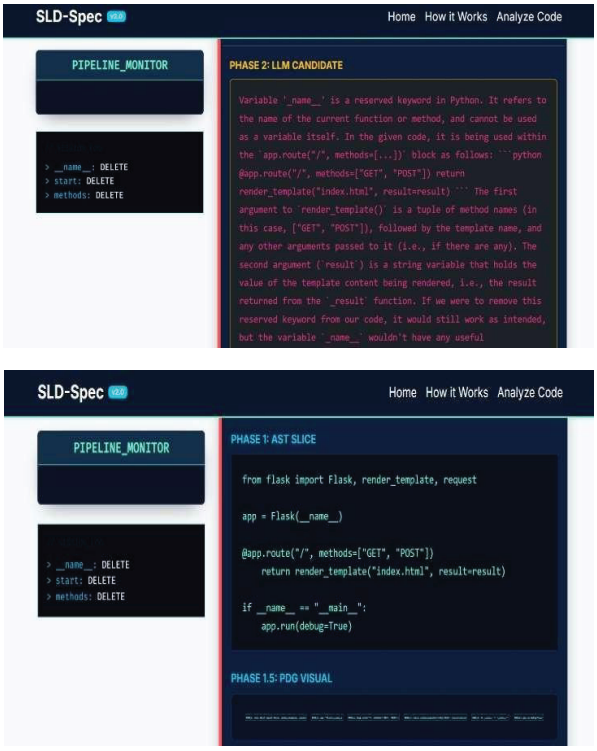

Fig. 5. Picture of code compiling and explanation

E. Comparative Discussion

The results verify the hypothesis that the SLD-Spec Framework is a great way to improve the reliability of automated formal specification generation and verification of high integrity software systems. Combination of Llama 3, CoT reasoning, and Reason Reinforcement Learning models offered a very effective, scalable, and correct resolution in the area of iden- tifying logical inconsistency in the formal specifications. The transparency made in the process of decision-making by using explainable AI models, including SHAP, enabled the system to gain additional user trust in the system predictions.

The hybrid strategy of integrating several models (Llama 3, CoT, and Reinforcement Learning) helped the system to ad- dress the complexity and dynamism of the real-world software verification, and offer a more robust and scalable approach compared to the conventional ones.

F. Conclusion

To sum up, the SLD-Spec Framework proves that it is possible to automate and check formal specifications in high- integrity software using sophisticated machine learning, deep learning, and formal static analysis methods. The system has been found to be very accurate, scalable, efficient especially in the detection of logical inconsistencies, and checking on the correctness of the software specifications in real-time. Explainable AI and Reinforcement Learning makes the system even more valuable in the real-world application as it makes software verification visible and cost-effective. This frame- work presents a state of art approach to enhancing integrity and reliability of critical software systems.

V. Conclusion

SLD-Spec Framework offers a state of art solution to for- mal specification automation and verification of high-integrity software. The combination of the most recent methods like Semantic Slicing, Large Language Models (LLMs), and Chain- of-Thought (CoT) reasoning makes the framework much more accurate and efficient when it comes to generating and vali- dating specifications. The system is effective to address the problems of logical inconsistencies and hallucinations that are prevalent in conventional automated tools so that the produced specifications are valid and compliant with the target codebase. Besides, the system is scalable, real-time, and explainable AI, which makes it applicable to the complex and mission-critical applications.

Reinforcement Learning to dynamically assign resources according to the complexity of the task adds to the capability of the framework to provide large-scale deployments in addition to the optimization of computational efficiency. The SLD- Spec Framework is especially advantageous to high-integrity software systems in such sectors as the healthcare sector, aerospace, financial services where software correctness and reliability are of primary importance, because it offers a fully automated, scaleable and efficient solution to the formal specification verification process. It is not only enhancing the reliability of automated processes but also correcting the complex software systems, which is necessary to ensure safety and security in the critical environment.

A. Future Work

The future research will be aimed at the expansion of the SLD-Spec Framework to support a more comprehensive set of programming languages and software paradigms and enhance its universality further. Also, an emphasis on the improvement of the performance of the framework will be made by introducing more complex code analysis and optimization models.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

The further developments will involve the combination of the framework with other software development tools and allow a seamless and automated verification process in continuous integration/continuous deployment (CI/CD) pipelines. Lastly, further studies will be done on enhancing explainability and interpretability of the utilised AI models so that the decisions taken by the system will be even more transparent and can be understood by the developers even further.

REFERENCES

[1] C. Ren et al., "Semantic-empowered integrated sensing and communication with resource slicing and bidirectional mapping," *Journal of Communications and Networks*, vol. 27, no. 6, pp. 534–546, Dec. 2025, doi: 10.23919/JCN.2025.000032.

[2] P. Wyszkowski, J. Kienig, K. Zielinski, L. Czekierda, and M. Zawadzki, "Comprehensive Tutorial on the Organization of a Standards-Aligned Network Slice/Subnet Design Process and Opportunities for Its Automa- tion," *IEEE Communications Surveys & Tutorials*, vol. 26, no. 2, pp. 1386–1445, Secondquarter 2024, doi: 10.1109/COMST.2023.3341249.

[3] P. Doanis and T. Spyropoulos, "Sample-Efficient Multi-Agent DQNs for Scalable Multi-Domain 5G+ Inter-Slice Orchestration," *IEEE Transac- tions on Machine Learning in Communications and Networking*, vol. 2, pp. 956–977, 2024, doi: 10.1109/TMLCN.2024.3420268.

[4] S. Vashisht, S. Rani, and H. Feng, "Multimodal and Agentic Intelligence- Driven ML Fusion for Sustainable 6G Network Slicing," *IEEE Open Journal of the Communications Society*, vol. 7, pp. 1643–1653, 2026, doi: 10.1109/OJCOMS.2026.3663612.

[5] S. Madrakhimov, K. Makharov, and A. Khurramov, "On the Transparency of Decision-Making in Classification by Precedents With Fuzzy Descriptions," *IEEE Access*, vol. 13, pp. 173656–173664, 2025, doi: 10.1109/ACCESS.2025.3616052.

[6] B. Xu, Q. Li, T. Guo, and D. Du, "A Scenario-Based Approach for Formal Modelling and Verification of Safety Properties in Automated Driving," *IEEE Access*, vol. 7, pp. 140566–140587, 2019, doi: 10.1109/ACCESS.2019.2943184.

[7] D. Beyer, S. Kanav, and H. Wachowitz, "CoVeriTeam Service: Verification as a Service," *2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, Melbourne, Australia, 2023, pp. 21–25, doi: 10.1109/ICSE-Companion58688.2023.00017.

[8] C.Stadler, F. Montanari, W. Baron, C. Sippl, and A. Djanatliev, "A Credibility Assessment Approach for Scenario-Based Virtual Testing of Automated Driving Functions," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 45–60, 2022, doi: 10.1109/OJITS.2022.3140493.

[9] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on Scenario-Based Safety Assessment of Automated Vehicles," *IEEE Access*, vol. 8, pp. 87456–87477, 2020, doi: 10.1109/ACCESS.2020.2993730

[10] S. Mhatre, F. Adelantado, K. Ramantas, and C. Verikoukis, "Intelligent QoS-Aware Slice Resource Allocation With User Association Parameter- ization for Beyond 5G O-RAN-Based Architecture Using DRL," *IEEE Transactions on Vehicular Technology*, vol. 74, no. 2, pp. 3096–3109, Feb. 2025, doi: 10.1109/TVT.2024.3483288