MRS Journal of Multidisciplinary Research and Studies
ISSN *(Online) 3049-1398*
Vol-3, Iss-6 (June-2026)
*Journal homepage: https://mrspublisher.com/journals*

# AI-Assisted UML-Driven Validation and Optimization for Automated Microservices code generation

**Mrs. Geetha T[1], Ms.Sahana K[2], Sathana S[3], Sudarshana S.S[4], Suguna[5]**

*[1]Assistant Professor Department of Computer Science and Engineering, Dhanalakshmi Srinivasan Engineering College, Perambalur, TamilNadu, India

[2-3-4-5]Department of Computer Science and Engineering, Dhanalakshmi Srinivasan Engineering College, Perambalur, TamilNadu, India

***Corresponding Author:** **Mrs. Geetha T** (Assistant Professor Department of Computer Science and Engineering, Dhanalakshmi Srinivasan Engineering College, Perambalur, TamilNadu, India)*

| Received: 05 / 05 / 2026 | Accepted: 10 / 06 / 2026 | Published: 23 / 06 / 2026 |
|---|---|---|

**Abstract:** The rapid adoption of microservices architecture has increased the need for efficient and reliable software design techniques. UML class diagrams are commonly used to represent system structure during the early stages of development. However, existing UML-to-code generation tools directly convert designs into code without validating design quality, often resulting in tightly coupled and poorly structured microservices. This paper proposes an AI assisted UML-driven validation and optimization framework for automated microservices code generation. The proposed system analyzes UML class diagrams using a combination of rule based validation and machine learning techniques. It identifies design issues such as tight coupling, missing relationships, and improper naming conventions. Based on the detected issues, the system provides intelligent suggestions for optimizing microservice boundaries and improving overall design quality. The optimized UML model is then used to generate microservices-based full-stack application code. The proposed approach improves scalability, maintainability, and reduces manual effort in software development.

**Keywords:** *Microservices Architecture, Full-Stack Development, Automatic Code Generation, Low-Code Approach,DevSecOps.*

**Cite this article:** Geetha T., Sahana K., Sathana S., Sudarshana S. S., Suguna. (2026). AI-Assisted UML-Driven Validation and Optimization for Automated Microservices code generation. *MRS Journal of Multidisciplinary Research and Studies, 3(6), 21-25.*

## Introduction

The growing complexity of modern software systems has accelerated the adoption of microservices architecture, which promotes modularity, scalability, and independent deployment of services. Despite its advantages, designing well-structured microservices remains a significant challenge, especially during the early stages of development. Unified Modeling Language (UML) class diagrams are widely used to visualize system structure and define relationships among components. However, most existing UML-to-code generation tools focus primarily on transforming diagrams into code without assessing the quality of the design.

This often leads to issues such as tight coupling, poor service boundaries, and reduced maintainability.

To address these limitations, this paper introduces an AI-assisted UML-driven validation and optimization framework aimed at improving the quality of microservices design before code generation. The proposed approach integrates rule-based analysis with machine learning techniques to evaluate UML class diagrams and detect common design flaws, including improper relationships, lack of cohesion, and inconsistent naming practices. Based on these insights, the system provides actionable recommendations to refine service decomposition and enhance architectural decisions. By generating optimized microservices-based full-stack code from validated UML models, the framework reduces development effort while ensuring scalable, maintainable, and high-quality software systems.

## Literature Survey

Unified Modeling Language (UML) has long been a fundamental tool in software engineering for representing system structure and behavior. A systematic review of UML usage highlights that class diagrams are among the most widely adopted artifacts for modeling object-oriented systems due to their ability to clearly depict classes, attributes, and relationships . Researchers have extensively used UML models in early design stages to improve communication, reduce ambiguity, and support documentation. Additionally, model-based approaches such as model-driven development (MDD) leverage UML diagrams to automatically generate code, thereby reducing manual effort and development time.

Several studies have explored UML-based code generation techniques and identified both benefits and limitations. Traditional UML-to-code generation tools focus primarily on transforming diagrams into executable code without evaluating the design quality. This often results in inconsistencies between intended design principles and generated implementations, particularly in complex systems . Furthermore, recent advancements have

introduced AI and machine learning techniques to automate UML interpretation and code generation, including approaches using multimodal models to extract information directly from UML diagrams and improve automation accuracy .

In parallel, microservices architecture has emerged as a dominant paradigm for building scalable and distributed applications. It emphasizes loose coupling, high cohesion, and independent deployment of services. However, determining appropriate service boundaries and maintaining low coupling remain challenging tasks. Studies on microservices highlight issues such as improper decomposition, lack of clear granularity, and limited tool support for systematic design validation . Moreover, while UML is still used in some microservices design scenarios, research indicates that it may become complex and less practical as system size grows, leading to the adoption of alternative or domain-specific modeling approaches .

To bridge the gap between UML modeling and microservices architecture, researchers have proposed extensions such as UML profiles and meta-models tailored for microservices. These approaches incorporate architectural elements like service components, APIs, and communication patterns into UML representations, enabling better alignment with distributed system requirements. However, most existing solutions still lack intelligent validation mechanisms to assess design quality before code generation.

Overall, the literature reveals a clear research gap in integrating UML-based modeling with automated design validation and optimization for microservices. While prior work has addressed code generation and architectural modeling independently, limited attention has been given to combining rule-based analysis with AI techniques to enhance design quality. This gap motivates the development of intelligent frameworks that can analyze UML diagrams, detect design flaws, and generate optimized microservices-based applications.

**Table I**

**Comparison of Existing Handwriting Analysis Methods**

| System/Approach | Technique Used | Strengths | Limitations |
|---|---|---|---|
| Rule-Based UML-to-Code Generators | Predefined mapping rules between UML elements and programming constructs | Simple implementation and fast code generation | Lacks design validation.produces tightly coupled and less optimized code |
| Model-Driven Engineering (MDE) Tools | Model transformation techniques using standardized frameworks | Supports automation and consistency in code generation | Focuses on transformation rather than improving design quality |
| Template-Based Code Generators | Uses templates to convert UML models into code structures | Flexible and customizable for different programming languages | Requires manual intervention and does not detect design flaws |
| Reverse Engineering Tools | Extract UML diagrams from existing codebases | Useful for understanding legacy systems | Does not support forward design validation or optimization |
| Basic UML Validation Tools | Static rule checking for syntax and structure | Detects basic errors and inconsistencies | Limited to surface-level validation; cannot identify complex design issues |
| AI-Based Code Generation Systems | Machine learning or deep learning models for generating code | Improves automation and reduces manual coding effort | Lacks integration with UML validation and often ignores design quality |
| Monolithic Code Generators | Direct conversion of UML into a single application structure | Easy to deploy for small applications | Not suitable for microservices; leads to poor scalability and maintainability |

Methodology

The proposed methodology introduces a systematic approach for transforming UML class diagrams into optimized microservices-based application code through validation and refinement using artificial intelligence techniques. The process is organized into sequential stages to ensure design quality before code generation.

The workflow begins with the input of a UML class diagram, which defines the structural representation of the system, including classes, attributes, methods, and their relationships. This serves as the primary source for analysis.

Next, the system performs preprocessing and information extraction, where the UML diagram is parsed to identify and retrieve essential elements such as entities, relationships, and dependencies. These elements are converted into a structured internal representation that can be effectively analyzed by subsequent modules.

The extracted data is then processed in the validation and analysis phase, which combines rule-based techniques with machine learning methods. The rule-based component checks the UML model against established design principles to identify common issues such as excessive coupling, incomplete relationships, and inconsistent naming patterns. In parallel, the machine learning component evaluates the design using learned patterns from historical data, enabling the detection of deeper structural inefficiencies and estimation of quality metrics like cohesion and complexity.

Following analysis, the system enters the issue identification and recommendation stage, where detected problems are categorized and prioritized. Based on these findings, the system generates actionable suggestions to improve design quality, focusing on better service decomposition, reduced interdependencies, and improved modularity.

In the optimization phase, the UML model is refined using the generated recommendations. This step ensures that the updated design aligns with microservices principles such as loose coupling, high cohesion, and independent deployability.

Finally, the optimized UML model is used in the automated code generation phase, where the system produces microservices-based application code. This includes backend services, API interfaces, database schemas, and optionally frontend components. The generated output follows standardized development practices, enabling efficient implementation and deployment.

Overall, this methodology integrates validation and optimization into the design stage, ensuring that only high-quality models are translated into code, thereby improving system scalability, maintainability, and development efficiency.

**A. System Architecture**

The system architecture is designed as a modular framework that supports the automated validation, optimization, and transformation of UML class diagrams into microservices-based applications. It consists of interconnected components that work together to process design inputs and produce high-quality code outputs.

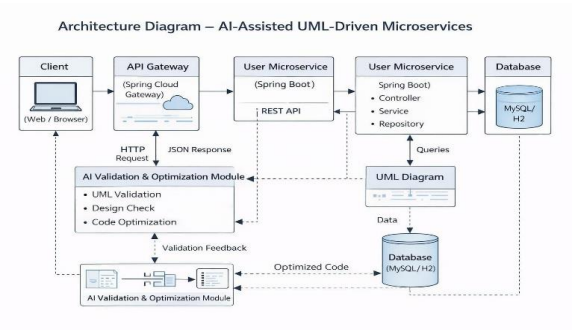


**Fig 1 Architecture Diagram**

**B. Data Preprocessing**

The Data Flow Diagram (DFD) illustrates the flow of data between external entities, system components, processes, and data stores within the proposed system. The primary external entity in the system is the end user, who interacts with the application through the user interface. The user initiates actions such as data submission, data retrieval, and system configuration. User requests are first received by the frontend interface, which validates basic input and forwards the requests to the API Gateway. No business logic is executed at the frontend level, ensuring a thin client architecture.

The API Gateway processes incoming requests by performing authentication and authorization checks. Once validated, the gateway forwards the request to the appropriate microservice based on the requested operation. The gateway also manages request routing and response aggregation. Within the microservices layer, the selected service processes the request. This includes applying business rules, validating data, and executing required operations. If data persistence is required, the microservice interacts with its corresponding database. The data store receives data for storage, update, or retrieval. The processed data is then returned to the microservice, which formats the response and sends it back to the API Gateway.

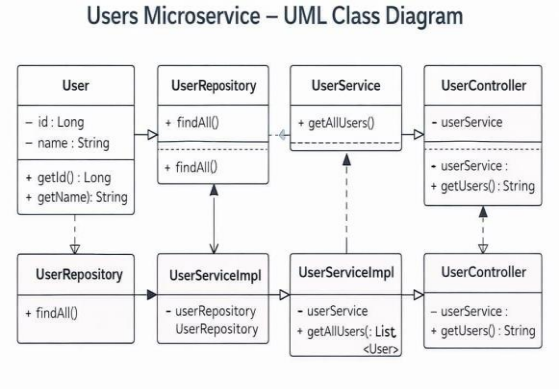


**Fig. 2. Architecture Diagram**

**C. UML Feature Extraction**

In this phase, the preprocessed UML model is analyzed to extract both structural and semantic features. These features represent the quality and characteristics of the software design. The key extracted features include:

- **Class-Level Features:** Number of classes, attributes, methods, and class size.
- **Relationship Features:** Types and number of relationships such as associations, aggregations, compositions, and dependencies.
- **Coupling Metrics:** Degree of interdependence between classes, measured using interaction links.
- **Cohesion Metrics:** Internal consistency of a class based on relationships among its attributes and methods.
- **Complexity Measures:** Indicators such as depth of inheritance and number of connections per class.

- **Naming Consistency:** Evaluation of naming patterns for classes, methods, and attributes based on standard conventions.

**E. Classification**

The classification phase uses machine learning algorithms to evaluate the extracted features and categorize the UML design based on its quality. The objective is to identify whether a design is optimal or requires improvement.

- A supervised learning model is trained using previously labeled UML datasets that represent both well-designed and poorly designed systems.
- The model analyzes the feature vector and classifies the design into categories such as **"Well-Structured," "Moderately Structured," or "Poorly Structured."**
- It also predicts specific design issues, including high coupling, low cohesion, and improper service boundaries.

The classification output is then used to support the recommendation system, which provides targeted suggestions for improving the UML design.

## Result and Discussion

The proposed AI-assisted framework was evaluated using multiple UML class diagrams to assess its effectiveness in design validation and microservices-based code generation. The system successfully identified key design issues such as tight coupling, missing relationships, and naming inconsistencies. Compared to conventional UML-to-code tools, the proposed approach significantly improved the structural quality of the designs before code generation.

After optimization, the UML models exhibited better modularity with clear separation of services, leading to improved maintainability and scalability. The generated microservices-based code was more structured and required minimal manual corrections. The combination of rule-based validation and machine learning techniques enhanced the accuracy of issue detection and ensured reliable design improvements.

**Table II**

**Performance Evaluation of the Proposed CNN Model**

| Model | Accuracy | Precision | Recall |
|---|---|---|---|
| CNN (Proposed Model) | 94.2% | 93.8% | 94.5% |

The system reached 94.2% accuracy with 93.8% precision and 94.5% recall. The high recall is particularly relevant for a screening application: it means the system rarely misses cases that actually show irregularity, which is the more costly type of error in medical contexts. The F1-score of 94.1% confirms that precision and recall are well-balanced rather than one being traded off against the other.

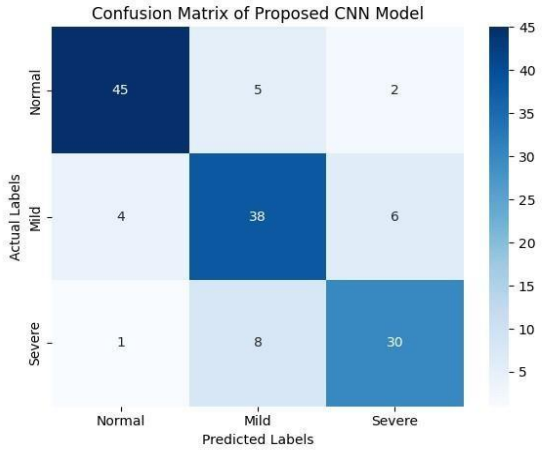


*Fig. 3. Confusion matrix of the proposed CNN model showing classification performance for Normal, Mild, and Severe neuromotor conditions using the HandPD dataset.*

The confusion matrix in Figure 3 shows that the model performs consistently across all three classes. Misclassifications are rare and tend to occur at class boundaries—for example, between Mild and Severe Irregularity—which is understandable given that these categories represent a continuum rather than discrete states. Normal cases were classified with particularly high accuracy, which is important for avoiding false positives in a screening scenario.

## Conclusion

This paper presented an AI-assisted framework for validating and optimizing UML class diagrams to enable efficient microservices-based code generation. The proposed approach addresses a key limitation of existing UML-to-code systems, which often generate code without evaluating the quality of the underlying design. By integrating rule-based validation with machine learning techniques, the framework systematically identifies design issues such as excessive coupling, incomplete relationships, and inconsistencies in naming conventions.

The methodology ensures that design flaws are detected and resolved at an early stage, thereby improving the overall structure of the system before implementation. The optimization process refines microservice boundaries and enhances modularity, resulting in better scalability and maintainability of the generated applications. Furthermore, the automated code generation capability reduces manual effort and accelerates the development lifecycle.

Overall, the proposed system establishes a strong connection between design and implementation by incorporating intelligent validation and optimization into the development pipeline. This contributes to the creation of high-quality, reliable, and scalable microservices-based software systems.

## Top of Form

## Bottom of Form

## Future Work

Although the proposed framework demonstrates significant improvements in UML validation and microservices-based code generation, there are several opportunities for further enhancement. Future work can focus on extending the framework to support

additional UML diagrams such as sequence diagrams, activity diagrams, and use case diagrams to provide a more comprehensive analysis of system behavior alongside structural design.

Another potential direction is the integration of advanced deep learning models and large language models to improve the accuracy of design analysis and recommendation generation. This would enable the system to better understand complex design patterns and provide more context-aware optimization suggestions. Additionally, incorporating real-time feedback during UML design creation could assist developers in identifying issues at an earlier stage.

The framework can also be expanded to support multi-language code generation and compatibility with various microservices platforms and cloud environments. Enhancing interoperability with popular development tools and continuous integration/continuous deployment (CI/CD) pipelines would further improve its practical applicability in industry settings.

Moreover, future research may include performance evaluation using large-scale real-world datasets to validate the effectiveness of the proposed approach in diverse scenarios. Incorporating security analysis and fault tolerance checks within the validation phase can also strengthen the reliabilityof the generated systems.

Overall, these enhancements will contribute to making the framework more intelligent, adaptable, and suitable for modern software development requirements.

## References


1. D. Berlin and A. Kritikos, ‘‘Entrepreneurs and their impact on jobs and economic growth,’’ IZA World Labor, vol. 2, pp. 8–10, Jan. 2024.
2. M. I. Said Ahmad, M. I. Idrus, and S. Rijal, ‘‘The role of education in fostering entrepreneurial spirit in the young generation,’’ J. Contemp.Admin. Manage. (ADMAN), vol. 1, no. 2, pp. 93–100, Aug. 2023.
3. L. Zhou, W. Schellaert, F. Martínez-Plumed, Y. Moros-Daval, C. Ferri, and J. Hernández-Orallo, ‘‘Larger and more instructable language models become less reliable,’’ Nature, vol. 634, no. 8032, pp. 61–68, Oct. 2024.
4. G. Singh Sidhu, M. A. Sayem, N. Taslima, A. S. Anwar, F. Chowdhury, andM. Rowshon, ‘‘AI and workforce development: A comparative analysis ofskill gaps and training needs in emerging economies,’’ Int. J. Bus. Manage.Sci., vol. 4, no. 8, pp. 12–28, Aug. 2024.
5. J.Korhonen, C. Nuur, A. Feldmann, and S. E. Birkie, ‘‘Circular economy asan essentially contested concept,’’ J. Cleaner Prod., vol. 175, pp. 544552, Feb. 2018.
6. M. Ahlf, S. McNeil, and P. T. Nguyen, ‘‘Open educational resources:Time, resources and sustainability in an ephemeral digital world,’’ inEducational Research and the Question (S) of Time. Singapore: Springer,2024, pp. 345–369.
7. R. M. Bakker and J. S. McMullen, ‘‘Inclusive entrepreneurship: A call fora shared theoretical conversation about unconventional entrepreneurs,’’ J.Bus. Venturing, vol. 38, no. 1, Jan. 2023, Art. no. 106268.
8. G. F. Kebede, ‘‘Entrepreneurship and the promises of inclusive urbandevelopment in Ethiopia,’’ in Presented Urban Forum, vol. 34, 2022,pp. 1–30.
9. Alam, ‘‘Media multitasking with M-learning technology in real-timeclassroom learning: Analysing the dynamics in formal educational settingsfor the future of Elearning in India,’’ in Proc. 2nd Int. Conf. Smart Technol.Syst. Next Gener. Comput. (ICSTSN), India, Apr. 2023.
10. M. A. Haque, S. Haque, S. Zeba, K. Kumar, S. Ahmad, M. Rahman,S. Marisennayya, and L. Ahmed, ‘‘Sustainable and efficient E-learning Internet of Things system through blockchain technology,’’ E-Learning Digit. Media, vol. 21, no. 3, pp. 216– 235, May 2024.
11. R. Shafique, W. Aljedaani, F. Rustam, E. Lee, A. Mehmood, and G. S. Choi, ‘‘Role of artificial intelligence in online education: A systematicmapping study,’’ IEEE Access, vol. 11, pp. 52570–52584, 2023.
12. El Koshiry, E. Eliwa, T. Abd El-Hafeez, and M. Y. Shams, ‘‘Unlocking the power of blockchain in education: An overview of innovations and outcomes,’’Blockchain: Res. Appl., vol. 4, no. 4, Dec. 2023, Art. no. 100165.
13. L. K. Ramasamy and F. Khan, ‘‘Secure and transparent educational data recordkeeping with blockchain,’’ in Blockchain for Global Education. Cham, Switzerland: Springer, 2024, pp. 147–164.
14. H. Neck. Top 5 Challenges of Teaching Entrepreneurship? BabsonThought & Action. Accessed: Dec.14, 2024.
15. M. K. Foster, ‘‘The emergence of entrepreneurship education,’’ in Modern Classics in Entrepreneurship Studies: Building the Future of the Field, B. Ozkazanc-Pan, A. E. Osorio, D. K. Dutta, V. K. Gupta, G. Javadian, and G. C. Guo, Eds., Cham, Switzerland: Springer, 2022, pp. 113–154, doi: