LLM-Based Retrieval-Augmented Construction of Dynamic Master Logic Models for System Diagnostics of Complex Systems

Saman Marandi[a], Yu-Shu Hu[b], and Mohammad Modarres[a]
[a] Center for Risk and Reliability University of Maryland, College Park, USA; {smarandi, modarres}@umd.edu
[b] DML Inc., Hsinchu, Taiwan, huyushu@dml.com.tw

Abstract: Functional modeling is widely used in reliability and safety assessment to describe how engineered systems achieve their objectives and how failures propagate through their functional structure. Dynamic Master Logic (DML) is a hierarchical functional modeling framework that links high-level objectives to supporting components through explicit logical relationships. However, constructing DML models from technical documentation remains a labor-intensive process, limiting its applicability to large systems. Recent advances in Large Language Models (LLMs) have created new opportunities to assist in extracting knowledge from unstructured engineering texts. In our prior work, LLMs were used to construct a KG-based representation of a DML model for a smaller-scale system; however, scalability to more complex systems and systematic evaluation were not addressed. In this study, we extend this approach to larger and more complex systems through a scalable retrieval-augmented LLM-based framework for constructing DML models from system descriptions. Retrieved document evidence is used to ground and constrain model generation. The resulting DML logic, including its hierarchical and logical relationships, is encoded as a Knowledge Graph (KG-DML). The approach employs hierarchical retrieval to incrementally build the model along the DML structure. The resulting KG-DML supports an interaction phase in which an LLM agent invokes predefined tools to generate diagnostic insights, enabling upward propagation of failure impacts and downward identification of success paths. A case study involving the Low-Pressure Coolant Injection system of a decommissioned nuclear power plant demonstrates consistent reconstruction of the DML model and stable structural performance across repeated runs. The results suggest that documentation-driven DML construction can scale to larger, document-intensive systems while maintaining the structural consistency required for reliability analysis.

1. INTRODUCTION

Modern engineered systems consist of numerous interconnected components operating across multiple hierarchical levels, requiring modeling approaches capable of capturing complex interactions and dependencies. Traditional approaches such as Fault Trees Analysis [1] and Event Trees Analysis [2] remain central to risk assessment practice; however, as engineered systems grow in structural complexity and documentation volume, constructing comprehensive models becomes increasingly labor-intensive and difficult to scale.

Functional modeling provides an alternative approach. Rather than enumerating failure events, functional models describe how system goals are achieved through hierarchical relationships among functions, subfunctions, components, and success conditions. Dynamic Master Logic (DML) offers an approach for representing these relationships and supports both forward and backward reasoning for diagnostic and reliability analysis [3]. Despite its advantages, DML model development typically requires detailed expert interpretation of system documentation, which limits broader deployment in large-scale or documentation-intensive systems.

Recent advances in Large Language Models (LLMs) have demonstrated strong capabilities in natural language understanding, semantic reasoning, and structured information extraction from unstructured text [4]. Large Language Models are built on transformer architectures that use self-attention mechanisms to model long-range dependencies in text [5]. However, directly applying LLMs to safety-critical modeling is challenging, as their outputs must remain logically consistent and grounded in the underlying system structure. In this work, we explore how retrieval-based LLM support can be combined with DML to help build functional models from system documentation. Rather than replacing expert judgment, the goal is to assist the modeling process by reducing the time and effort required to develop reliable representations for diagnostics and decision making.

The remainder of the paper is organized as follows. Section 2 provides background on DML and related LLM and Knowledge Graph (KG) approaches. Section 3 presents the research motivation. Section 4 describes the proposed framework. Section 5 presents the case study, including model construction, interaction, and diagnostic interface. Section 6 evaluates the framework. Section 7 discusses findings and future work, and Section 8 concludes the paper.

2. BACKGROUND

2.1. Dynamic Master Logic for System-Level Modeling

For system-level modeling, DML represents system knowledge as a success logic organized in hierarchical form. Rather than focusing solely on failure events, DML describes how system objectives are achieved through supporting functions, sub-functions, system elements, and basic components. This structure enables analysis of the degree of success or failure, estimation of system-level performance, and the propagation of state changes among components. As such, DML provides an effective framework for describing causal relationships and tracing the propagation of failures or anomalies in complex engineered systems [6]. Figure 1 conceptually illustrates the DML framework, emphasizing the hierarchical decomposition from system objectives to basic components and the interaction between functional and structural perspectives.

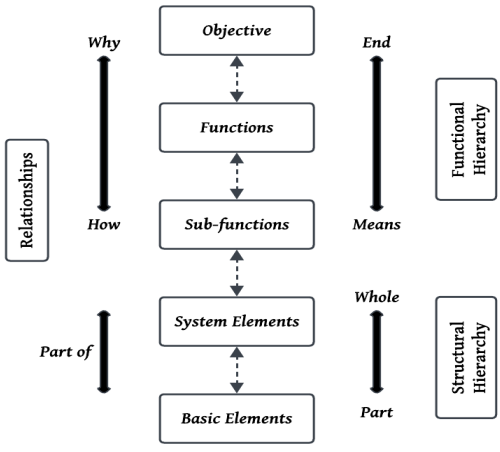

Figure 1. Conceptual DML framework. Adapted from [6].

The functional hierarchy proceeds top-down from objectives to functions and sub-functions, capturing the intended purpose and behavior of the system. In contrast, the structural hierarchy decomposes the system into elements and basic components, reflecting its physical or logical composition. Arrows indicate causal

"Why–How" relationships within the functional hierarchy and compositional "Part-of" relationships within the structural hierarchy. Together, these relationships form an integrated representation in which higher-level goals depend on lower-level means, and structural components collectively enable system objectives.

DML-based modeling has been applied in various safety-critical domains, including nuclear power plants [7], [8] and geared wind turbines [9]. Its ability to represent both hierarchical decomposition and logical interdependence makes it well suited for reliability analysis.

Despite these advantages, constructing DML models traditionally requires extensive expert interpretation of design documents and system descriptions. This manual effort can limit scalability when systems are large or documentation is extensive.

2.2. LLMs, Knowledge Graphs, and Diagnostic Modeling

Recent advances in LLMs have introduced new possibilities for assisting in diagnostics and knowledge extraction from technical documentation. However, they are also subject to well-documented limitations, including hallucination and inconsistent reasoning, which are particularly problematic in safety-critical contexts. To address these issues, researchers have increasingly combined LLMs with KGs. In such hybrid systems, the KG provides structured, external grounding that constrains and verifies LLM outputs. Integrated LLM–KG frameworks have shown promise in nuclear systems, industrial machinery, and sensor networks. These approaches support fault tracing, root cause analysis, and explainable diagnostics by linking language-based reasoning to explicitly encoded system relationships [10], [11], [12]. Recent work has also explored the use of LLMs to identify relationships among elements within functional modeling frameworks such as DML for fault detection and diagnosis [13]. In our prior work [14], LLMs were used to construct a KG-based DML representation for a relatively small-scale system. While the study demonstrated the feasibility of integrating LLM reasoning with DML structures, scalability to more complex systems and systematic evaluation of the generated models were not addressed. The present work extends this direction by introducing a retrieval-augmented and hierarchically constrained framework designed for larger, document-intensive systems.

However, most existing KG-based diagnostic applications focus on symptom-based fault retrieval or fault-cause mapping. In these systems, the KG primarily serves as a repository of fault knowledge rather than as a representation of the system's functional logic. As a result, the hierarchical relationships between system objectives, functions, and structural elements are not fully captured within the graph.

3. RESEARCH MOTIVATION

The integration of LLMs and KGs for functional modeling remains relatively underexplored. In particular, limited work has examined how LLMs can assist in constructing formal system logic representations such as DML directly from design and operational documents. To address this gap, this paper presents a framework that combines LLMs and KGs to automatically build and interact with a DML model derived from system documentation. The proposed framework aims to:

a) Automate functional model creation from unstructured system descriptions.

b) Enable interactive fault analysis through hierarchical and probabilistic reasoning.

c) Support risk-informed decision-making using natural language explanations grounded in formal system logic.

By grounding LLM-assisted extraction within a DML representation encoded as a KG, the framework seeks to preserve the rigor required for probabilistic safety assessment while improving scalability of model development.

4. PROPOSED FRAMEWORK

The proposed framework converts unstructured system documentation into a DML representation through a retrieval-grounded and hierarchically constrained process. Rather than attempting to construct the entire model in a single generation step, the approach proceeds incrementally through the DML hierarchy to preserve structural consistency and logical coherence. The overall workflow consists of document preparation, layer-by-layer model construction, and synthesis of the validated structure into a KG representation. The overall workflow of the proposed framework is illustrated in Figure 2.

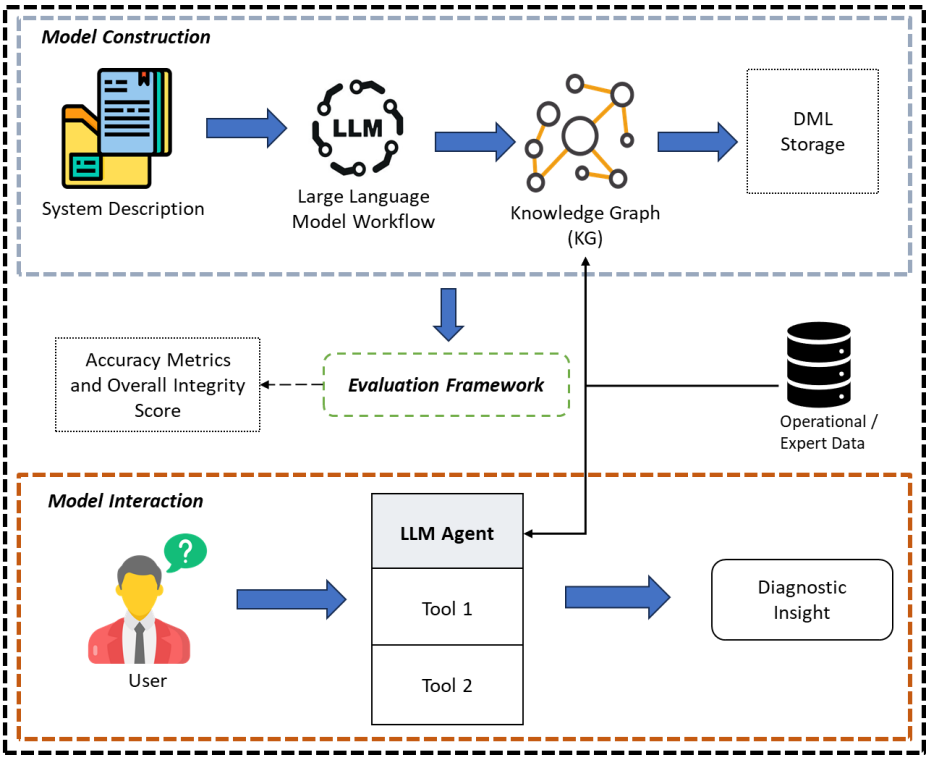

Figure 2. High-level overview of research. Adapted from [15].

The process begins with preprocessing of system design and operational documents to standardize terminology and ensure consistent identification of components and system elements. The documentation is then segmented into overlapping text sections and embedded in a vector database, enabling similarity-based retrieval of relevant passages during each modeling stage. By restricting generation to retrieved evidence associated with a specific hierarchical scope, the framework reduces unsupported inference and maintains traceability to source documentation.

Model construction proceeds sequentially from objectives to functions, subfunctions, system elements, and basic components. At each layer, previously validated parent elements define the scope for expansion. A retrieval query is formulated using the semantic definition of the current layer together with the relevant parent nodes, and the most relevant document segments are provided to the language model as contextual evidence. The model generates candidate elements and their logical relationships in a predefined format, which are then validated for structural coherence, identifier consistency, and logical correctness before

being integrated into the evolving representation. Boolean dependencies are explicitly identified and preserved, and nested logical groupings are maintained rather than flattened to retain the structural semantics of the system.

Once construction is complete, the representation is translated into a KG. Nodes correspond to DML elements, and edges encode hierarchical, causal, and compositional relationships, with logical operators represented explicitly within the graph structure. Interaction with the resulting model is separated from model construction: the LLM agent interprets user queries and selects appropriate operations, while structural traversal and probabilistic evaluation are performed using predefined functions on the graph.

5. CASE STUDY

The case study focuses on the Low-Pressure Coolant Injection (LPCI) system of the decommissioned Millstone Point Unit 1 nuclear power plant, as documented in the Interim Reliability Evaluation Program prepared for the U.S. Nuclear Regulatory Commission [16]. The LPCI system provides emergency core cooling following a Loss-of-Coolant Accident (LOCA) by delivering coolant to the reactor vessel under low-pressure conditions. The system consists of two redundant subsystems interconnected by a cross-tie line and includes pumps, heat exchangers, motor-operated valves, spray headers, and defined injection paths. Coolant is drawn from the suppression chamber and routed through alternative cooling and injection configurations depending on plant conditions. The presence of redundancy, multiple flow paths, and nested logical dependencies makes the LPCI system well suited for evaluating hierarchical DML construction and preservation of Boolean logic relationships.

5.1. Model Construction

Model construction began with preparation of the LPCI system documentation to support consistent extraction of structural information. Component names and identifiers were standardized to prevent inconsistent references to the same physical element, and abbreviations were clarified to reduce ambiguity. Where necessary, brief contextual descriptions were associated with component identifiers to distinguish elements with similar names but different roles. The documentation was then divided into overlapping text segments and stored in an indexed form to enable targeted retrieval of relevant passages during hierarchical model development. This preparation ensured that each modeling step remained traceable to specific portions of the source documentation.

The DML hierarchy was then developed sequentially, beginning with system-level objectives and progressing through functions, subfunctions, components, and success conditions. Each layer was constructed using previously validated parent elements as structural constraints. For a given layer, relevant portions of the documentation were identified and examined in relation to the defined scope of the parent nodes. Candidate elements and their logical dependencies were organized into a representation and reviewed for consistency, completeness, and clarity before being incorporated into the evolving model. By expanding the hierarchy incrementally and validating each stage prior to integration, the process limited structural inconsistencies and preserved the logical relationships necessary for downstream analysis.

To maintain model consistency, validation checks were performed after each stage of hierarchical expansion. These checks reviewed the representation for completeness of parent–child relationships, logical nesting consistency, and clarity of element naming. Where inconsistencies or ambiguities were identified, refinement was performed before proceeding to subsequent layers. This staged validation limited the propagation of structural errors and ensured that each level of the hierarchy remained logically coherent before integration into the overall model.

Once the hierarchical representation was finalized, it was translated into a KG and deployed in the Neo4j [17] environment. Each node in the graph corresponds to a DML element, and semantic relationships such as *Achieved by*, *Depends on*, *Requires*, and *Success through* define the hierarchical structure. Logical gates at each level determine how child nodes influence their parent, preserving the causal and compositional relationships central to the DML framework.

To enable analysis, component nodes were augmented with operational attributes representing possible system states, such as operational, degraded, or failed, along with associated likelihood values. Success probabilities tied to these states allow the graph to reflect changing system conditions based on real-time or simulated inputs. This structure supports probabilistic propagation and diagnostic reasoning directly on the encoded hierarchy.

Figure 3 illustrates the hierarchical organization of the resulting KG and the placement of logical gates within the DML structure.

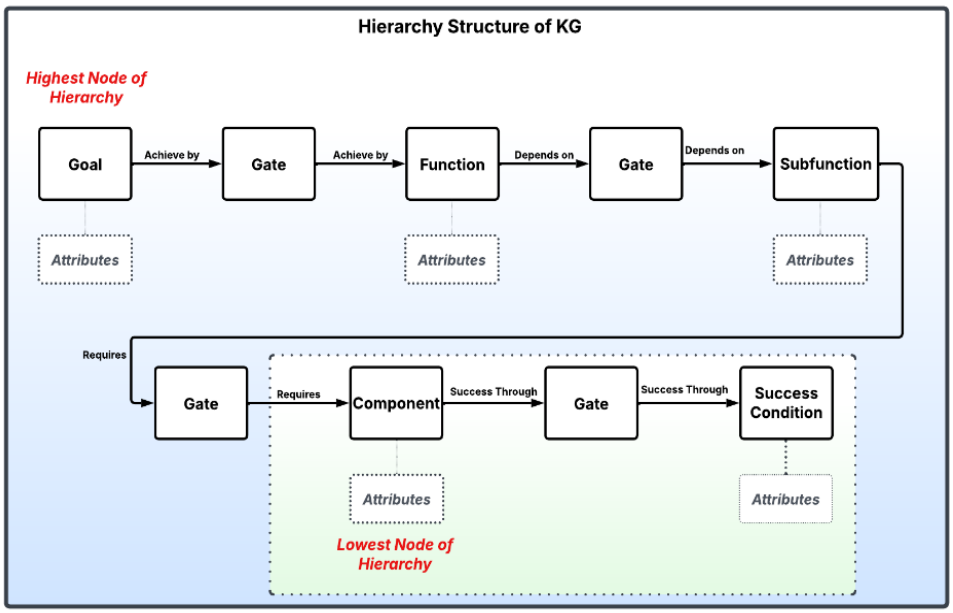

Figure 3. Hierarchy structure of KG [15].

5.2. Model Interaction

Following construction of the KG, the model supports interactive analysis of system behavior. User queries are interpreted and categorized according to their analytical intent, such as failure impact assessment, success path identification, or structural explanation. Based on the type of query, the appropriate reasoning procedure is invoked to operate directly on the KG. Responses are returned in natural language and include relevant structural context from the model, allowing users to explore dependencies, trace fault effects, and understand how specific components contribute to higher-level objectives.

For explanatory queries, relevant portions of the KG are retrieved and examined to ensure that responses are grounded in the encoded system logic. Rather than relying on general descriptions, the explanation is derived from the explicit hierarchical and logical relationships stored in the graph. This ensures that structural reasoning remains consistent with the constructed DML model.

5.3. LLM Reasoning Tools

5.3.1. Upward Propagation

The upward reasoning procedure evaluates how changes at the component level influence the likelihood of achieving higher-level functions and system objectives. Each component is associated with an operational state, such as operational, degraded, or failed. These states are assigned success probabilities that represent the likelihood of the component performing its intended function under current conditions.

Component-level success probabilities are propagated upward through the logical structure defined in the DML hierarchy. When a subfunction depends on multiple components, the propagation follows the Boolean logic encoded at that level. If the relationship is conjunctive (AND), all supporting components must perform successfully; degradation in any one component reduces the overall success probability of the subfunction. If the relationship is disjunctive (OR), alternative components provide redundancy, and the subfunction retains a higher probability of success unless all alternatives are compromised.

This propagation continues recursively through the hierarchy. Subfunction outcomes influence parent functions, and function-level results ultimately determine the likelihood of achieving system-level objectives. At each level, success probabilities are calculated based on the logical dependencies and the inherited values from lower layers. A node is identified as impacted when its success probability decreases below a defined threshold due to degradation or failure in dependent elements. The output of the analysis highlights affected nodes and reports their updated probabilities, allowing users to assess both the extent and severity of the impact.

Uncertainty is inherent in real-world system operation due to incomplete data, measurement variability, and changing operational conditions. The framework accounts for this uncertainty by embedding probabilistic attributes directly within component nodes and propagating these values through the hierarchical logic structure. Unlike purely deterministic models, this approach provides a continuously adjustable estimate of system status that can be updated as new information becomes available. The model also supports partial updates, enabling incorporation of new operational data or engineering insights without reconstructing the entire hierarchy. This adaptability enhances the practical value of the framework in long-term operational environments, including safety-critical and mission-driven systems.

5.3.2. Success Path Generation

The downward reasoning procedure provides a complementary perspective by identifying the structural requirements necessary to achieve a selected objective or function. Rather than tracing the consequences of failure, this analysis determines what must operate successfully for the target node to be satisfied. Starting from a specified goal or function, the procedure traverses the DML hierarchy downward, examining the associated functions, subfunctions, components, and success conditions.

At each level of the hierarchy, the logical relationships encoded in the model determine how child elements contribute to their parent. If a relationship is conjunctive, all supporting elements are required for success. If the relationship is disjunctive, alternative configurations may exist, reflecting redundancy or parallel pathways within the system. In such cases, multiple structurally valid success paths are identified. Each path represents a minimal set of elements that, if operational, are sufficient to achieve the target objective.

The result of this process is a collection of success paths expressed in terms of specific components and their associated conditions. These paths clarify what is minimally required for system success and provide insight into redundancy and dependency structure. Such information supports recovery planning, mission assurance, and evaluation of alternative operating configurations. When considered together with upward propagation analysis, the framework enables assessment of both failure impact and operational recovery within a unified hierarchical structure.

5.4. Diagnostic Example Results

The diagnostic interface supports three primary types of interaction: upward tracing, success path identification, and structural explanation. These modes allow users to examine failure impacts, determine what is required for successful operation, and understand the role of specific elements within the hierarchy.

For upward tracing, the system evaluates how a component failure affects higher-level elements. For example, if component C1 is assumed to fail, the analysis identifies impacted subfunctions, functions, and the associated system goal (e.g., G1), along with their updated success probabilities. This provides a clear view of how localized degradation influences overall system performance.

For downward analysis, users may request how a specific function, such as F2, can be achieved. The system responds by identifying a minimal success path, for example requiring Pump 1, Pump 2, Valve 1, Valve 2, and Tank 1 to operate successfully. Where redundancy exists, alternative valid paths are generated.

Explanatory queries provide structural clarification. A query regarding the role of SF1, for instance, returns its relationship to F1, the supporting component (e.g., C2), and the governing logical gate structure. Similarly, the interface can identify the primary system functions, such as F1 through F4, directly from the encoded hierarchy.

6. EVALUATION

The constructed KG-DML models are evaluated against a manually developed reference model to assess element-level accuracy and structural consistency across all layers of the DML hierarchy, including goals, functions, subfunctions, components, and success conditions. At each layer, nodes in the constructed model are compared with those in the reference model after normalization to account for naming differences, and classified as correctly identified, missing, or extraneous. Relationship correctness is assessed by comparing predicted parent–child links against the reference, with errors categorized as missing, incorrect, or extra. A gate accuracy metric evaluates whether the logical operator (AND or OR) at each grouping matches the reference. An aggregate integrity score combining layer-level accuracy and penalties for missing elements that disrupt hierarchical paths is used to assess overall structural completeness.

The framework achieves high structural accuracy across all layers. Higher-level elements such as goals and functions are identified with near-perfect accuracy, and most relationships and logical dependencies are correctly recovered. Minor omissions are observed at lower levels of the hierarchy, particularly in densely connected regions where logical dependencies are more complex. The framework also demonstrates consistent performance across repeated runs, with only small variation between executions.

7. DISCUSSION

The evaluation results indicate that the proposed framework reconstructs the functional structure of the system with high fidelity. High node-level precision and recall suggest that the model reliably identifies the relevant elements of the hierarchy, with only minor omissions, reflecting the effectiveness of the retrieval and schema-constrained extraction process in grounding the generated structure.

At the structural level, high link-level precision and recall indicate that most relationships between elements are correctly established, demonstrating that hierarchical dependencies in the DML representation can be recovered from textual descriptions with strong consistency. The slightly lower recall compared to precision suggests that remaining errors are primarily due to missing relationships rather than incorrect ones.

Gate accuracy is lower than node- and link-level metrics, highlighting the increased difficulty of recovering logical dependencies. Correct identification of AND/OR relationships requires not only detecting relevant elements but also interpreting their interactions within the system. As a result, errors are more likely to occur in regions with higher structural density and more complex dependency groupings. These results indicate that while the framework performs reliably overall, recovering complex logical dependencies remains more challenging, particularly in densely connected portions of the hierarchy.

7.1. Model Selection

In this work, GPT-4o was used to support the extraction and structuring tasks required for building the KG-DML. The model was selected because it performs reliably in following prompts and handling the multi-step reasoning needed across the DML hierarchy. However, relying on a large model accessed through an external API has practical drawbacks. In many engineering settings, system documentation can be confidential, which makes sending data to external services problematic. In addition, the layer-by-layer construction process requires multiple model calls, which can lead to relatively high API usage and cost.

The framework itself reduces the burden on the model by constraining the extraction through a predefined schema and outputs. This suggests that smaller or locally hosted language models may be sufficient for parts of the process, especially for more routine extraction steps. Exploring such configurations could improve cost efficiency and make the approach easier to deploy in practice.

7.2. Future Work

Several directions can be explored to extend this work. First, the framework was evaluated on a single system, and applying it to additional systems with different structures and documentation styles would help assess its generality. In practice, system information is often distributed across multiple documents, so extending the approach to handle multi-source retrieval is an important next step.

Another area for improvement is the handling of more complex and densely connected portions of the hierarchy. While the framework captures most elements and relationships, some errors remain in lower-level layers and in logical dependencies. Refining the retrieval strategy or introducing additional validation steps could help improve consistency in these areas.

Future work will also explore the use of smaller or locally hosted language models to reduce reliance on external APIs. This would make the framework more practical for real-world applications where cost and data confidentiality are important considerations. Hybrid approaches, where different models are used for different stages of the process, may offer a useful balance between performance and efficiency.

Finally, the current evaluation focuses on structural accuracy. Extending the evaluation to include functional validation, such as comparing success paths or diagnostic outcomes, would provide a stronger assessment of how well the constructed model supports reasoning tasks.

8. CONCLUSION

This work presents a framework for constructing DML models from system documentation and representing them as a KG-DML to support diagnostic reasoning. By combining retrieval with generation and schema constraints, the approach converts unstructured text into a hierarchical representation that preserves functional dependencies and logical relationships. The results show that the framework can recover most elements and relationships of the reference model with high accuracy, while maintaining consistency across the hierarchy. Remaining differences are mainly associated with more complex and densely connected portions of the model, particularly in logical dependencies. Overall, the approach

provides a practical way to support DML model construction with reduced manual effort, while keeping the structure interpretable and suitable for further analysis. Although additional validation across systems is needed, the results suggest that retrieval-based construction can be used to generate representations from technical documentation in a consistent and scalable manner.

References

[1] E. Ruijters and M. Stoelinga, "Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools," *Computer Science Review*, vol. 15–16, pp. 29–62, Feb. 2015, doi: 10.1016/j.cosrev.2015.03.001.

[2] M. Čepin, "Event Tree Analysis," in *Assessment of Power System Reliability*, London: Springer London, 2011, pp. 89–99. doi: 10.1007/978-0-85729-688-7_6.

[3] Y.-S. Hu and M. Modarres, "Evaluating system behavior through Dynamic Master Logic Diagram (DMLD) modeling," *Reliability Engineering & System Safety*, vol. 64, no. 2, pp. 241–269, May 1999, doi: 10.1016/S0951-8320(98)00066-0.

[4] T. B. Brown *et al.*, "Language Models are Few-Shot Learners," 2020, *arXiv*. doi: 10.48550/ARXIV.2005.14165.

[5] A. Vaswani *et al.*, "Attention Is All You Need," 2017, *arXiv*. doi: 10.48550/ARXIV.1706.03762.

[6] M. Modarres and S. W. Cheon, "Function-centered modeling of engineering systems using the goal tree–success tree technique and functional primitives," *Reliability Engineering & System Safety*, vol. 64, no. 2, pp. 181–200, May 1999, doi: 10.1016/S0951-8320(98)00062-3.

[7] J. O'Brien, D. Marksberry, and M. Modarres, "Reactor Safety Assessment System (RSAS). Development and Lessons Learned." Sep. 1997.

[8] F. Antonello, J. Buongiorno, and E. Zio, "A methodology to perform dynamic risk assessment using system theory and modeling and simulation: Application to nuclear batteries," *Reliability Engineering & System Safety*, vol. 228, p. 108769, Dec. 2022, doi: 10.1016/j.ress.2022.108769.

[9] Y. F. Li, S. Valla, and E. Zio, "Reliability assessment of generic geared wind turbines by GTST-MLD model and Monte Carlo simulation," *Renewable Energy*, vol. 83, pp. 222–233, Nov. 2015, doi: 10.1016/j.renene.2015.04.035.

[10] Y. Ma, S. Zheng, Z. Yang, H. Pan, and J. Hong, "A knowledge-graph enhanced large language model-based fault diagnostic reasoning and maintenance decision support pipeline towards industry 5.0," *International Journal of Production Research*, pp. 1–22, Feb. 2025, doi: 10.1080/00207543.2025.2472298.

[11] L. Li, A. Haruna, W. Ying, K. Noman, and Y. Li, "Knowledge graph-driven fault diagnosis for aviation equipment: Integrating improved joint extraction with large language model," *Journal of Industrial Information Integration*, vol. 50, p. 101039, Mar. 2026, doi: 10.1016/j.jii.2025.101039.

[12] W. B. M. Razaq, H. Chen, M. R. Machado, and J. L. R. Moreira, "How can the integration of AI large language models and knowledge graph enhance fault diagnosis? A systematic literature review," *Applied Soft Computing*, vol. 194, p. 114908, May 2026, doi: 10.1016/j.asoc.2026.114908.

[13] Y.-S. Hu, S. Marandi, and M. Modarres, "DML–LLM Hybrid Architecture for Fault Detection and Diagnosis in Sensor-Rich Industrial Systems," *Sensors*, vol. 26, no. 6, p. 2008, Mar. 2026, doi: 10.3390/s26062008.

[14] S. Marandi, Y.-S. Hu, and M. Modarres, "Complex System Diagnostics Using a Knowledge Graph-Informed and Large Language Model-Enhanced Framework," *Applied Sciences*, vol. 15, no. 17, p. 9428, Aug. 2025, doi: 10.3390/app15179428.

[15] S. Marandi, Y.-S. Hu, and M. Modarres, "Integrating Large Language Models and Knowledge Graphs for System Diagnostics," in *2026 Annual Reliability and Maintainability Symposium (RAMS)*, Miramar Beach, FL, USA: IEEE, Jan. 2026, pp. 1–6. doi: 10.1109/RAMS50514.2026.11424548.

[16] J. J. Curry, D. W. Gallagher, M. Modarres, J. A. Radder, and M. (USA) Science Applications Inc. ,. Bethesda, "Interim reliability-evaluation program: analysis of the Millstone Point Unit 1 nuclear power plant. Volume I. Main report." May 1983.

[17]Neo4j, Inc., “Neo4j GitHub Repository,” GitHub. [Online]. Available: https://github.com/neo4j/neo4j