Automated Fault Tree Generation with a Constraint Driven Large Language Model

Taeyeon Kim [a], Man Cheol Kim [a*]
[a, a*] *Department of Energy Systems Engineering, Chung-Ang University, 84 Heukseokro, Dongjak-gu, Seoul 06974*
[*] *Corresponding author: charleskim@cau.ac.kr*

Keywords : fault tree analysis, large language model, automation, auxiliary feedwater system, validation

1. Introduction

Generating a fault tree (FT) for a given system requires accurate and detailed expert analysis and typically involves substantial manual effort. For this reason, Experts have long attempted to automate the process of Fault Tree Analysis. Recently, rapid advances in large language models (LLMs) have further accelerated attempts to use LLMs to assist or automate failure-logic generation.

Despite this growing interest, most published LLM based approaches have focused on demonstrating feasibility rather than delivering high fidelity, production-ready fault tree models. In practice, LLM outputs may vary across runs and can include spurious or weakly supported logic, which still requires expert review and correction before practical use. In addition, prior studies seldom provide rigorous, tool executed evidence demonstrating quantitative equivalence between LLM generated FT and expert developed FT. [1-4]

To address these limitations, this study proposes a constraint driven LLM pipeline that automatically generates a fault tree model that can be directly imported into a Probabilistic Safety Assessment (PSA) tool. The proposed approach is evaluated through a real case study by comparing the generated results against expert FTA results.

2. Methods and Results

This section presents the proposed constraint driven LLM pipeline for automated fault tree construction. We describe the pipeline workflow and implementation, specify the inputs required for model generation, and explain how the pipeline produces an import ready FT artifact for PSA tool. We then demonstrate the effectiveness of the approach on the Auxiliary Feedwater (AFW) system by comparing the generated fault tree and quantitative results against an expert developed FT model.

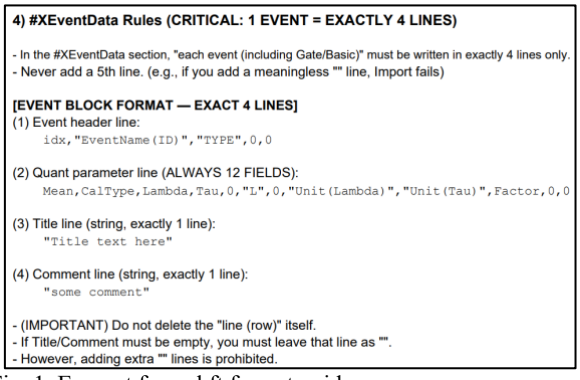
2.1 Overview of the proposed pipeline

The pipeline uses a GPT based model configured through OpenAI's Custom GPT framework, which allows additional constraints to be applied to the base model so that generation can be guided toward a predefined format. The primary goal of this work was to automatically generate a fault tree from inputs to make the output immediately usable in the commercial PSA tool AIMS-PSA without manual rework. To achieve this, we designed the pipeline so that, when a user provides a piping and instrumentation diagram, a written system description together with a failure mode list, and the quantitative parameters required for fault tree quantification, the model automatically produces a fault tree file in the kft format which is compatible with AIMS-PSA. The resulting kft artifact can be imported directly into AIMS-PSA, where the built-in graphical interface enables detailed fault tree review and inspection of quantification results.

2.2 Instruction driven export control

We constrained the model to construct the fault tree structure directly from the provided inputs, starting with simple structural rules that force serial relationships to map to OR gates and parallel relationships to AND gates for consistency, and then extending the constraints to finer details.

Moreover, we applied strict export control at the kft level. In the standard workflow, an analyst builds a fault tree directly in the AIMS-PSA graphical interface, and the tool internally records the construction steps to produce a kft file. Under this GUI-based process, syntax and structural formatting errors in the kft file are impossible because the tool itself enforces the format. In contrast, our pipeline generates the kft text from the ground up. As a result, even minor deviations in syntax or block structure can immediately cause import failure in AIMS-PSA. Therefore, controlling the output format is not optional but a necessary requirement for tool connectivity.


4) #XEventData Rules (CRITICAL: 1 EVENT = EXACTLY 4 LINES)

- In the #XEventData section, "each event (including Gate/Basic)" must be written in exactly 4 lines only.
- Never add a 5th line. (e.g., if you add a meaningless "" line, Import fails)

[EVENT BLOCK FORMAT — EXACT 4 LINES]
(1) Event header line:
`idx,"EventName(ID)","TYPE",0,0`

(2) Quant parameter line (ALWAYS 12 FIELDS):
`Mean,CalType,Lambda,Tau,0,"L",0,"Unit(Lambda)","Unit(Tau)",Factor,0,0`

(3) Title line (string, exactly 1 line):
`"Title text here"`

(4) Comment line (string, exactly 1 line):
`"some comment"`

- (IMPORTANT) Do not delete the "line (row)" itself.
- If Title/Comment must be empty, you must leave that line as "".
- However, adding extra "" lines is prohibited.

Fig. 1. Excerpt from .kft format guide

To address this, we constrained the model to follow fixed kft writing rules and coupled generation with integrated validation checks which screen for import-breaking issues before the final file is produced. Figure 1 illustrates one small portion of these controls, focusing on the event blocks must follow a rigid structure, such as requiring exactly four lines per event and disallowing extra blank lines. While the figure shows only a subset, detailed constraints was applied across the broader kft structure to ensure that the generated artifact can be imported into AIMS-PSA without errors.

These controls should be understood as a specification layer for AIMS-PSA compatible FT generation, not as an explicit prescription of the full fault-tree structure itself. The fixed gate-mapping rules, modeling scope and conventions, and .kft writing requirements constrained how the output had to be represented, but they did not explicitly define all components, failure relationships, or train-level organization in advance. Within those boundaries, the LLM had to interpret the P&ID together with the written system description and failure mode list, infer the relevant components and their failure relationships, and organize them into a coherent FT structure while incorporating the provided quantification information into the required format. In this sense, the role of the constraints was to control representation and tool compatibility, whereas the role of the model was to perform structured inference and integration from the inputs.

2.3 Case study and results

The proposed pipeline was evaluated using the AFW system from an OPR1000 plant, for which an expert developed fault tree was available as a baseline. As inputs, we provided the AFW P&ID, a written system description with a failure mode list, a task description that fixes the modeling scope and conventions, and the quantification data set such as basic event probability and common cause failure (CCF) grouping information. The evaluation focused on whether the generated kft artifact can be imported into AIMS-PSA without errors, whether the resulting FT structure is consistent with the expert baseline, and whether AIMS-PSA quantification results match the expert FT results under the same calculation settings.

In the AFW case, the generated kft file was imported into AIMS-PSA without import failures and produced an FT structure consistent with the expert developed FT, including the intended train level organization and CCF placement. Quantification in AIMS-PSA also matched the expert baseline, yielding the same top event probability (1.450e−5) and the same number of minimal cut sets (6,184). To assess robustness, we repeated the generation in 20 independent sessions and obtained importable kft outputs in all runs, with consistent structure and identical quantification results. The same comparison procedure was applied to multiple additional OPR1000 systems, and the pipeline consistently produced kft artifacts that were imported into AIMS-PSA without errors and yielded results consistently with the corresponding expert fault trees, confirming reliable operation beyond the AFW case.

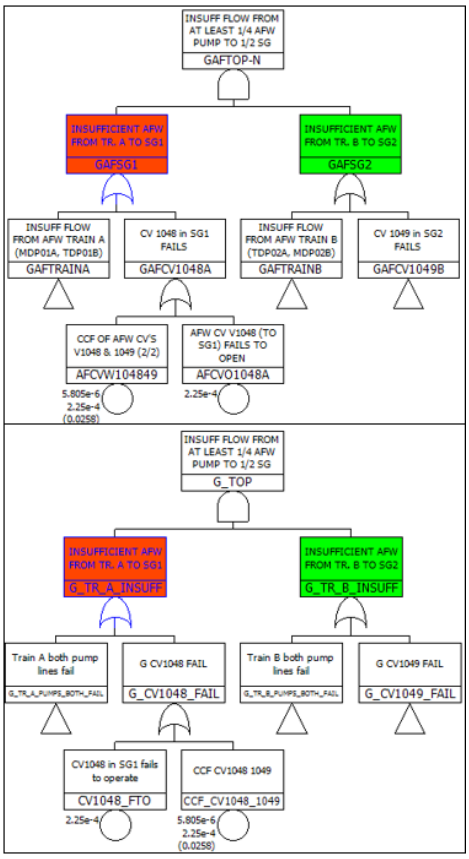


Fig. 2. Structural excerpt comparison of the AFW fault tree between the expert FT (top) and the automatically generated FT (bottom).

3. Conclusions

This study presented a constraint driven LLM pipeline that automates FT construction and directly outputs an AIMS-PSA compatible kft artifact. By constraining both FT logic generation and kft export, the proposed approach was designed to produce a tool importable FT model without manual rework while maintaining consistent modeling conventions. A case study on the OPR1000 AFW system demonstrated that the generated FT was consistent with the expert developed FT structure and, importantly, produced the same quantification results as the expert baseline. These results indicate that the proposed pipeline can move LLM based FT generation beyond feasibility demonstrations toward practical, verifiable FT construction.

The approach nevertheless depends strongly on the quality of the provided inputs and on how clearly the modeling scope and conventions are specified. Ambiguous system descriptions, incomplete failure mode definitions, or inconsistent parameter sets can propagate into the generated FT and reduce reliability.

Future work will extend the evaluation to more complex systems and larger systems to test scalability and robustness under realistic modeling complexity. We also plan to refine the constraint and validation layers to better handle edge cases and to further reduce the need for human intervention in defining inputs and checking model outputs.

REFERENCES

[1] Y. Shentu and M. Trapp, Facilitating Fault Tree Analysis with Generative AI, Computer Safety, Reliability, and Security: SAFECOMP 2025 Workshops, Lecture Notes in Computer Science, Vol. 15955, pp. 524-536, Springer, Cham, 2026.
[2] V. Rychkov, C. Picoco, and E. Caleca, Impact of Generative AI (Large Language Models) on the PRA Model Construction and Maintenance: Observations, arXiv preprint, arXiv:2406.01133, 2024.
[3] S. S. Shetiya, H. Garikapati, and A. Sohoni, FTA Generation Using GenAI with an Autonomy Sensor Usecase, arXiv preprint, arXiv:2411.15007, 2024.
[4] K. Clegg, I. Habli, and J. McDermid, Using GPT-4 to Generate Failure Logic, Computer Safety, Reliability, and Security: SAFECOMP 2024 Workshops, Lecture Notes in Computer Science, Vol. 14989, pp. 148-159, Springer, Cham, 2024.
[5] S. M. Shin, D. S. Kim, and H. G. Kang, Power-operated check valve in abnormal situations, Nuclear Engineering and Design, Vol. 330, pp. 28-35, 2018.