A Study on Automated Fault Tree Generation Using Object-Oriented Modeling of Simplified P&ID Components

Jinok Lee[1], Gyunyoung Heo[1]*, Ho Seok[1],
[1]*Department of Nuclear Engineering, College of Engineering, Kyung Hee University, Yongin, Republic of Korea*
**Corresponding Author: gheo@khu.ac.kr*

Keywords : Fault Tree Analysis, Simplified Piping and Instrumentation Diagram , PSA, Automation

1. Introduction

As the development of advanced nuclear reactors progresses, Probabilistic Safety Assessment (PSA) is expected to play a critical role from the early design stage, requiring close collaboration between designers and PSA practitioners. In the initial design phases, the capability to rapidly construct and update PSA models in response to design changes is more important than building highly detailed models.

PSA consists of various technical elements; however, this study focuses on FTA (Fault Tree Analysis), which is considered the most feasible for automation during the design process. Traditionally, Fault Tree (FT) models have been manually constructed by domain experts, a process that is time-consuming and prone to human error [1]. In systems such as plants and power stations, Simplified Piping and Instrumentation Diagram (P&ID) has been primarily used to represent interconnections between components. However, these diagrams often exist in paper form or as unstructured digital files such as PDFs, making automatic FT generation based on them virtually impossible [2].

Consequently, FTA has long depended on manual work by experts, leading to persistent issues such as a lack of consistency in analysis, limitation of time and cost, and increased potential for errors. Recently, however, the adoption of CAD-based P&ID management and the database integration of design and operational information have opened new possibilities for automation. Existing studies have been limited by the unstructured and complex nature of P&ID data, and have not achieved full automation due to technical constraints in diagram interpretation, causal reasoning, and logical expansion. Moreover, most approaches still require significant manual or partial intervention from experts, presenting a high entry barrier for novice users.

In this study, we propose a methodology for the automatic generation of FTs by modeling the components of Simplified P&ID using an object-oriented class structure. This approach aims at improving the level of FTA automation, minimizing subjective intervention by analysts, and enabling reliable and timely risk assessment for complex systems.

2. Design and Implementation of the Proposed Methodology

In this study, only the back-end portion of the overall framework illustrated in Figure 1 has been implemented, while the ultimate goal is to develop a fully integrated system that encompasses both the front-end and back-end. The front-end is designed to assist users in easily creating Simplified P&IDs and, at the same time, to automatically generate corresponding component classes based on the drawn diagrams. This section focuses on the back-end, proposing an object-oriented class structure derived from the Simplified P&ID and presenting the automatic FT generation algorithm developed on this basis. Furthermore, a case study was conducted to validate the feasibility of the proposed algorithm.

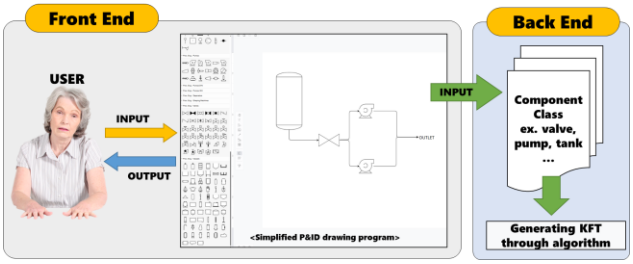

Figure 1. System Architecture Diagram

2.1. Data Structuring of Simplified P&ID

In this study, the standard procedure for constructing FT is followed by the Fault Tree Handbook published by U.S.NRC [3], which consists of the following main steps:

Step 1: System definition and boundary setting - Define the functions and scope of the target system for analysis.

Step 2: Top Event definition - Select the target event to be analyzed, such as system failure or loss of a safety function.

Step 3: System function analysis - Identify the roles and interactions of each function and component.

Step 4: Derivation of logical structure - Determine causal relationships and connection types (e.g., serial, parallel) between components.

Step 5: Assignment of gates and basic events - Apply appropriate logic operators such as AND, OR, and K-out-of-N gates to connect components.

Step 6: Model verification and modification - Review

logical consistency, completeness, and duplication, and make necessary revisions.

In this study, a data structure was designed to clearly represent the relationships between components in order to automatically derive a FT from a Simplified P&ID. To achieve this, the FT automation procedure was defined to follow the standard process presented in the Fault Tree Handbook, ensuring that all elements required in a conventional FT could be encompassed. Furthermore, the property of an event data structure from the AIMS-PSA in Figure 2 was referenced to define the type field, thereby ensuring compatibility.

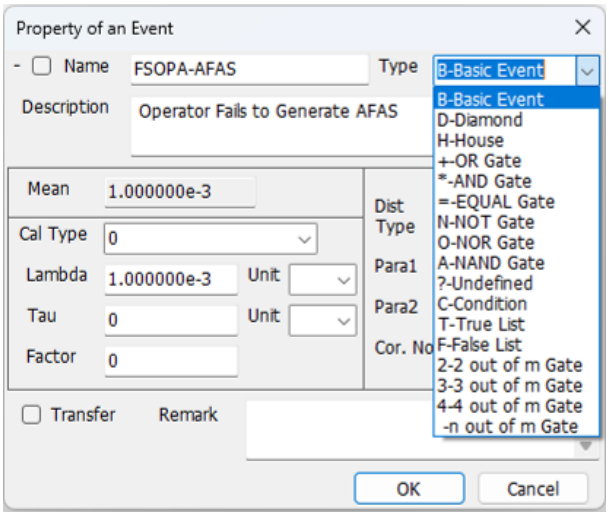

Figure 2. Property of an Event in AIMS-PSA

The defined component class structure is presented in Table 1, and each class is managed as an independent object. The relationships among components are explicitly defined through the parent and child fields. Furthermore, this class structure was designed to allow the algorithm to understand the causal relationships between gates and components, thereby enabling the automatic generation of the necessary logic gates and basic events.

When a user creates a simplified P&ID, dragging and dropping a component onto the diagram automatically generates a component class with the structure shown in Table 1. Once the user connects nodes between components, the corresponding parent and child fields are defined according to these relationships. The parent, child, component, and status fields are automatically defined by the software used for simplified P&ID creation, whereas the id, capacity, CCF, ccfID, and exception fields must be entered manually by the user. In addition, the failure modes for each component are predefined in a separate database. Based on the identified component type, the algorithm automatically generates the appropriate gates or basic events.

Table 1. Component Class Structure

Field Name	Data Type	Default	Description
No	Int	None	Unique number component
id	Optional[str]	None	Unique ID (Primary Key)
parent	Optional[List[Component]]	[]	List of parent components (Foreign Key)
child	Optional[List[Component]]	[]	List of child components (Foreign Key)
type	Str	"B"	Logic type (e.g., B, +, *)
component	Str	""	Equipment type (e.g., Valve, Pump)
status	Str	""	Status information (e.g., open, stop, TOP)
capacity	Float	0.0	Design capacity
ccf	Bool	False	Indicates Common Cause Failure (True = CCF)
ccfID	Optional[List[Component]]	[]	Related CCF component ID
exception	Bool	False	Indicates exclusion from analysis (True = excluded)

System-defined fileds, User-defined fields

2.2. Automated FT Generation Algorithm

The automatic FT generation algorithm proposed in this study is designed to automatically construct hierarchical fault logic by utilizing the information of component classes created as the user draws the simplified P&ID diagram. The main procedures of the algorithm are explained through the pseudo code presented in Table 2.

Table 2. Pseudo-code for FT Generation

```
# Step 1. Component Class Creation
FOR each component ADDED in Simplified P&ID:
    CREATE new ComponentClass
    user DEFINES component.id
    component.component ← automatically assigned (based on type: valve,
pump, etc.)
    component.status ← automatically assigned (based on current state)

# Step 2. Node Connection Definition
IF user CONNECTS nodes:
    UPDATE component.parent
    UPDATE component.child

# Step 3. User-Defined Fields
FOR each component:
    user INPUTS component.capacity
    IF component is in a CCF relation:
        component.ccf ← True
        component.ccfID ← related component.id

# Step 4. Parent-Child Re-definition
FOR each component WHERE component.child IS NULL:
    TRACE parent upward
    IF parent.type == Basic Event:
        REDEFINE parent field to reference the higher-level component

# Step 5. Gate Generation Rules
FOR each component:
    IF component.child COUNT ≥ 2:
        total_capacity ← SUM(child.capacity)
        IF total_capacity > 100:
            CREATE AND gate
        ELSE:
            CREATE OR gate
        IF child COUNT > 2:
            K ← COUNT(combinations of child where SUM(capacity) > 100)
            N ← child COUNT
            CREATE K-out-of-N AND gate

# Step 6. Database-Based Component Handling
    IF component.component EXISTS in Database:
        CREATE gate
        CREATE basic events (based on predefined failure modes)
```

```
# Step 7. Output Generation
IF all gates and basic events are defined:
    PERFORM logic operations
    EXPORT result in FT (AIMS KFT format)
```

2.3. Validation and Case Study

When the user constructs the simplified P&ID shown in Figure 3, the *parent*, *child*, *component*, and *status* fields of each component are automatically defined as presented in Table **3.** Component classes of Figure 1Table 3. The user then additionally specifies information such as *id*, *capacity*, *CCF*, *ccfID*, and *exception*. Based on these completed component classes, the algorithm automatically generates the FT.

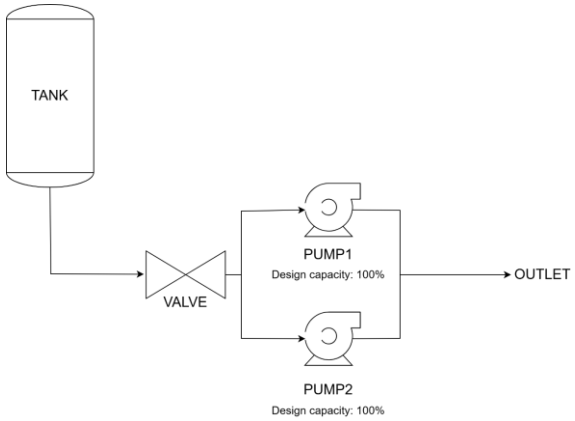

Figure 3. Simplified P&ID example

Tank: Starting point where the fluid is stored and supplied through the lower piping.

Valve: Located between the tank and pumps, it controls fluid flow; its open/closed state is used for failure mode determination.

Pump 1 & Pump 2: Arranged in parallel, with either pump sufficient to satisfy system performance requirements. Depending on the performance information, they are represented by AND/OR gates in the FT. As identical components, they may also be treated under a Common Cause Failure (CCF) relationship.

Outlet: The final delivery point of the fluid, which determines whether the system performs its intended function. This is defined as the Top Event in the FT.

Table 3. Component classes of Figure 1

Class Field	TOP EVENT	TANK	VALVE	PUMP1	PUMP2
no	**Auto**	**Auto**	**Auto**	**Auto**	**Auto**
id	TOP	TANK	VALVE	PUMP1	PUMP2
parent	**Null**	**TOP**	**TOP**	**VALVE**	**VALVE**
child	TANK	VALVE	PUMP1 PUMP2	Null	Null
type	*	**B**	**B**	**B**	**B**
component	Null	TANK	Valve	Pump	Pump
status	**TOP**	**Null**	**Open**	**Run**	**Run**
capacity	Null	Null	Null	100	100
ccf	**False**	**False**	**False**	**True**	**True**
ccfID	Null	Null	Null	PUMP2	PUMP1
exception	**False**	**False**	**False**	**False**	**False**

System-defined fileds, User-defined fields

The algorithm verifies the connection relationships by checking the parent and child fields of each component. After verifying the connection relationships, it examines the combined capacity of the parallel pumps. If the total exceeds 100%, an AND gate is automatically generated, with the basic events of each pump assigned as its child nodes.

In Figure 3, Pump 1 and Pump 2 are arranged in parallel with identical design specifications and share a CCF relationship. Since the design capacity of each pump is 100%, the system can meet its performance requirements as long as at least one pump operates normally.

In particular, for pump components, failure modes such as failure to start and failure to run are considered. When the algorithm recognizes a component as a pump, it first generates a gate and then automatically creates basic events representing each failure mode based on predefined rules. This process is guided by a rule-based database for each component type, allowing systematic generation of failure-mode-specific basic events without additional user input.

For CCF handling, the user must set the CCF field of the component to True and specify the related class ID in the ccfID field. Based on this information, the algorithm automatically generates the corresponding gate and CCF event.

Once all component classes are fully defined, the algorithm performs logical operations and generates a Fault Tree file in the KFT format compatible with AIMS-PSA. When this KFT file is imported into AIMS-PSA, the resulting Fault Tree is obtained, as shown in Figure 4.

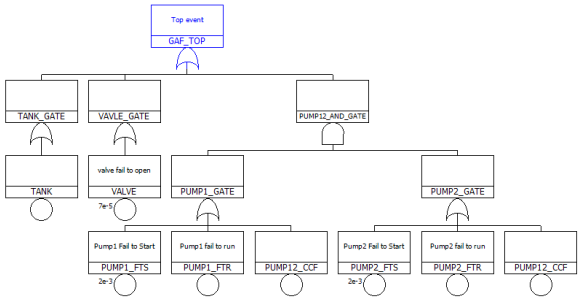

Figure 4. Resulting FT generated through the automation algorithm

3. Conclusion

This study proposes a methodology for the automatic generation of FTs based on Simplified P&ID. To this end, key components such as tanks, valves, and pumps were modeled using an object-oriented class structure, enabling systematic definition of their attributes, states, performance, and hierarchical relationships. Based on this framework, an algorithm was developed to automatically construct standardized FT generation procedures and a consistent hierarchical FT logic.

The proposed algorithm demonstrated its capability to automatically derive the logical gates required for FT

construction—such as AND, OR, and K-out-of-N gates—as well as basic events, by utilizing component attributes including capacity, state, and parent-child relationships. In addition, by incorporating a CCF field, the algorithm was shown to automatically generate gates and events that reflect dependent failure modes. The outputs were produced in the KFT format compatible with AIMS-PSA, thereby confirming the practicality of the proposed methodology.

Nevertheless, several limitations remain. The current framework requires users to manually input CCF relationships, as the algorithm is not yet able to autonomously identify them solely from component information. Developing the capability to generate CCF events automatically from Simplified P&ID data alone, without reliance on user input, represents an important area for future improvement. Furthermore, since the algorithm was validated using a simplified example system, additional extensions will be necessary to enable its application to more complex systems.

In conclusion, the proposed methodology offers value by reducing the time consumption and human error potential inherent in the traditionally manual FT construction process, while enabling rapid and consistent development of PSA models at the design stage. This approach enhances the level of automation in PSA model development and is expected to provide a foundation for efficient and consistent FT configuration, particularly during the early design phases of advanced reactors.

Acknowledgments

This work was supported by the Human Resources Development of the Korea Institute of Energy Technology Evaluation and Planning (KETEP) grant funded by the Korea government Ministry of Knowledge Economy (No. RS-2023-00244330) and the Nuclear Safety Research Program through the Regulatory Research Management Agency for SMRS (RMAS) and the Nuclear Safety and Security Commission (NSSC) of the Republic of Korea (No. 1500-1501-409).

REFERENCES

[1] Arroyo, Esteban, et al. "Automatic derivation of qualitative plant simulation models from legacy piping and instrumentation diagrams." Computers & Chemical Engineering 92 (2016): 112-132.

[2] Bäckström, Ola, et al. "Flexibility of analysis through knowledge bases." Proc. 31th Eur. Saf. Rel. Conf. 2021.

[3] U.S. Nuclear Regulatory Commission (NRC). "Fault Tree Handbook." NUREG-0492, Washington, DC, 1981.