Proceedings of the 35th European Safety and Reliability & the 33rd Society for Risk Analysis Europe Conference
Edited by Eirik Bjorheim Abrahamsen, Terje Aven, Frederic Bouder, Roger Flage, Marja Ylönen
©2025 ESREL SRA-E 2025 Organizers. *Published by* Research Publishing, Singapore.
doi: 10.3850/978-981-94-3281-3_ESREL-SRA-E2025-P1191-cd

Ontology-driven Integration of System-Theoretic Process Analysis and Model-Based Safety Analysis for Comprehensive Safety Assessment

Max Chopart

Department of Air Transport, Czech Technical University in Prague, Czech Republic.
E-mail: chopamax@fd.cvut.cz

Julien Vidalie

Airbus Protect, France. E-mail: julien.vidalie@airbus.com

Xavier De Bossoreille

Airbus Protect, France. E-mail: xavier.de-bossoreille@airbus.com

This work aims to improve safety assessments in complex systems by presenting an ontology-driven approach to integrating Model-Based Safety Analysis (MBSA) and System-Theoretic Process Analysis (STPA). The ontology serves as an interface with two main goals: (1) to filter STPA scenarios and identify failure-based cases for MBSA, and (2) to automatically translate the filtered scenarios into MBSA-compatible feared events (i.e., “observers”). By bridging STPA’s hazard identification methodology with MBSA’s failure-based analysis, this approach offers a more efficient and comprehensive framework for system safety and reliability assessment.
Even though the improvement of overall safety is a goal shared by both STPA and MBSA, their approach to doing so is different. STPA provides a broad hazard identification framework capturing a range of scenario types, including non-failure cases driven by systemic, human, and organizational factors. In contrast, MBSA focuses on the study of failure propagations. These methodologies are thus complementary and can offer a more complete picture when used together; however, achieving such integration requires a systematic approach to filter and map relevant information between them.
To categorize STPA scenarios and eliminate those unsuitable for MBSA, the created ontology serves as a common framework by ensuring that only scenarios appropriate for MBSA’s failure-based approach are transferred. The ontology lessens the need for human intervention improving accuracy and enabling more thorough fault propagation analyses by automatically converting compatible STPA scenarios into MBSA’s format.
Early findings show that this strategy improves the effectiveness of the safety assessment process while also streamlining the integration of STPA and MBSA. The ontology-driven interface enables the utilization of both methodologies’ strengths through a methodical and structured mapping, providing a unified framework that can be tailored to the intricacies of contemporary safety-critical systems.

Keywords: failure propagation model, hazard identification, Model-Based Safety Analysis, ontology, safety assessment, System-Theoretic Process Analysis

1. Introduction

Modern safety-critical systems, such as those found in the aerospace and automotive industries, have reached levels of complexity that make it difficult to rely solely on traditional, failure-centric safety approaches. System-Theoretic Process Analysis (STPA) stands out for its holistic view of systems, capturing hazards that can vary from software, organizational structures, and human factors. Meanwhile, Model-Based Safety Analysis (MBSA) can go deeper, at a lower level of abstraction, mapping out the fault propagation of potential failures. These two methodologies each address different facets of system safety, suggesting that together they may provide a more complete and efficient approach to hazard identification and assessment.

Building on their previous work with the Cessna Citation Mustang’s electrical system Chopart et al. (2024), the authors showed that STPA could identify a wide range of hazardous scenarios, including issues that might remain hidden in a purely failure-driven framework. MBSA, in turn, can verify the scenarios proposed by simulating them directly, pinpointing how failures propagate and measuring their likelihood. While this harmonization showed valuable insights, there are also a few obstacles: many STPA scenarios do not reflect failure-based events, and transferring them into MBSA models has typically involved a manual, time-consuming process that is prone to human error.

The authors now introduce an ontology-driven interface that filters STPA scenarios for those pertinent to MBSA. By automatically translating these filtered scenarios into "observers" the approach cuts down on manual effort and avoids losing STPA's comprehensive scope. The end result is a more seamless workflow, one that preserves the broad coverage of STPA while taking full advantage of MBSA's detailed modelling strengths.

2. Background

This section provides details of the two approaches that being used in this paper.

2.1. *STPA*

STPA is a technique for spotting and averting possible losses in intricate systems, like software, aircraft, or even a combination of both. The foundation of STPA is the STAMP model, which sees accidents as the consequence of inadequate control or enforcement of safety constraints on the system's behaviour and interactions.

According to Leveson (2018), STPA consists of four steps:

(1) Identify the system-level losses and hazards.
(2) Draw the control structure of the system, showing the controllers, control actions, feedback, and controlled processes.
(3) Identify the unsafe control actions that could contribute to the hazards, such as commands that are not given, given incorrectly, or given at the wrong time.
(4) Identify the causal factors and control flaws that could cause unsafe control actions, such as design errors, human errors, software errors, or environmental factors.

Compared to more conventional techniques like Fault Tree Analysis (FTA) or Failure Modes and Effects Analysis (FMEA), STPA is a proactive approach to safety engineering that can identify more accident causes James Elizebeth et al. (2023).

2.2. *MBSA*

Traditional safety tools such as fault trees, lack expressive power and are conceptually far to specifications Batteux et al. (2019). MBSA approaches aim to reduce the gap between specifications and safety models. They can consist of three different kinds of approaches Batteux et al. (2019):

- extensions of fault trees or reliability block diagrams;
- specialized profiles of MBSE tools;
- model-based safety and reliability analysis modelling languages.

We are interested in the latter, which encompasses formal languages such as SAML Gudemann and Ortmeier (2010), Figaro Bouissou et al. (1991), and AltaRica Data-Flow Boiteau et al. (2006). MBSA has been acknowledged as a tool for aircraft safety assessment in the ARP4761A guidelines ARP4761A (2023). MBSA is based on Failure Propagation Models. They represent the architecture of the system in a functional or organic view, close to the schematic views used in design. Physical or functional components of the system are represented as bricks that contain local behaviour. The global behaviour of the system is then computed through the combination of these local behaviours. The study case presented in this paper was modelled using the SimfiaNeo tool [a], based on AltaRica Data-Flow.

Failure conditions are taken as inputs of the MBSA model, and defined in AltaRica using observers, which are boolean variables. They are computed as formulae depending on the chosen variables of the system.

[a]https://www.protect.airbus.com/safety/simfianeo/

3. Methodology

This methodology uses ontology to bridge STPA outputs with MBSA tools, ensuring consistency and traceability. The approach includes formalizing Loss Scenarios and mapping them to MBSA variables. This ensures consistency, traceability, and scalability, enabling analysts to effectively use STPA results within the MBSA framework.

3.1. *Ontology*

The foundation of this approach is an ontology that establishes a structured representation of both the STPA artefacts and their relationship to MBSA elements. A selective view of the ontology is shown in Fig.1. Each instance of a Loss Scenario is linked to a Controller, a Control Action and a Context. Additionally, the ontology includes mappings that relate these concepts to their corresponding MBSA variables. This ensures a direct path from higher-level safety analysis to more formal, tool-specific representations, notably using AltaRica Dataflow language. By serving as a single, authoritative knowledge base, the ontology enables moving from STPA results to MBSA models.

3.2. *Loss Scenarios to Observers transformer*

The first step in the process is to generate the Loss Scenarios, as described by Thomas (2024) to get a formal Loss Scenario that explicitly denotes the controller, the control action, the feedback and the context. According to Thomas, four types of Loss Scenarios exist and can be generated as follows:

(1) Unsafe Controller Behavior:
   - `<Controller> provides <Control Action> when <Context>`
   - `<Input> to <Controller> correctly indicates <Context>`

(2) Unsafe Feedback Path:
   - `<Feedback> does not adequately indicate <Context>`
   - `<Process> is actually <Context>`

(3) Unsafe Control Path:
   - `<Controller> does not provide <Control Action> when <Context>`
   - `<Process> receives <Control Action> when <Context>`

(4) Unsafe Controlled Process Behaviour:
   - `<Process> does not receive a <Control Action> when <Context>`
   - `<Process> applies <Control Action> when <Context>`

Once the Loss Scenario text is generated, it is converted as an instance of the ontology. A Loss Scenario instance might look like this in a simplified Turtle syntax:

```
:LS_X a :LossScenario ;
   :hasController :<Controller> ;
   :hasControlAction :<ControlAction> ;
   :hasContext :<Context> .
```

It is then possible to query each Loss Scenario instance and use the ontology to make the interface between the STPA and the MBSA:

```
:<Controller> :hasStateVariable
    controllerVar .
:<ControlAction> :hasFlowVariable
    controlActionVar .
:<Context> :hasAssertion contextVar .
```

The next step is to write a Boolean expression, now using MBSA artefacts, that represents the conditions to satisfy for the Loss Scenario to be triggered:

$$\begin{aligned}&\text{LossScenario} := \text{controllerVar}\\ &\wedge \text{controlActionVar}\\ &\wedge \text{contextVar}\end{aligned} \tag{1}$$

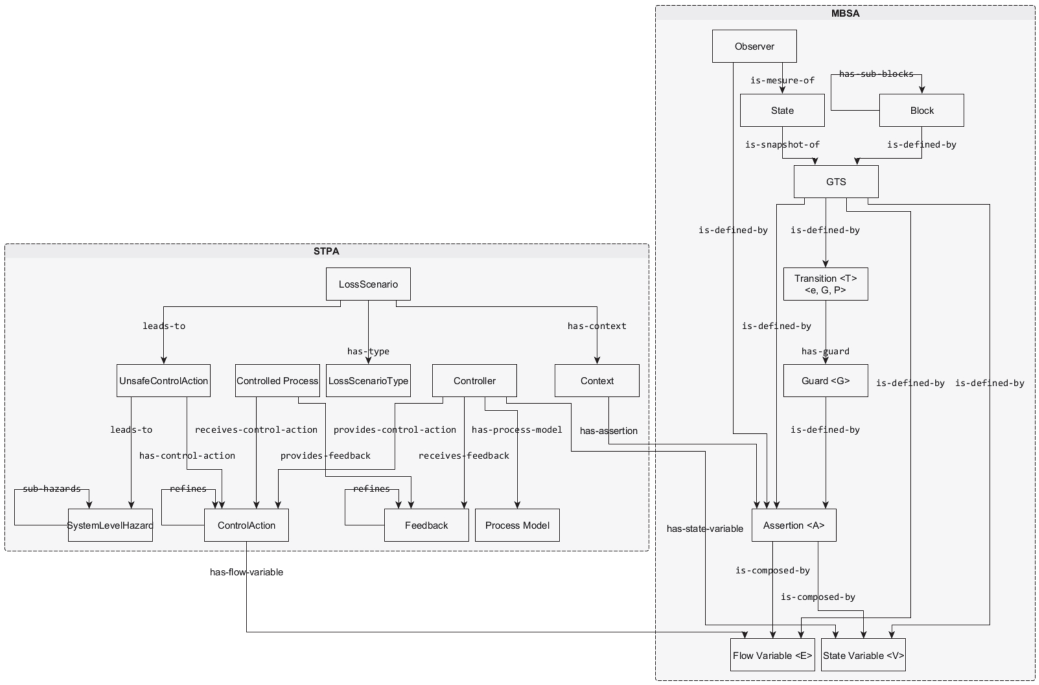

Fig. 1. STPA and MBSA's partial ontology

The authors are using the tool SimfiaNeo, which uses AltaRica-Dataflow under the hood. Therefore, Eq. (1) becomes:

```
observer LS : boolean;
LS = controllerVar
   and controlActionVar
   and contextVar;
```

3.3. *Filtering non-failure-based scenarios*

In addition to capturing all Loss Scenarios in the ontology, a filtering process can ensure that only those scenarios relevant to hardware or software failures are transformed to be supported by the MBSA. By default, STPA may generate scenarios rooted in a wide variety of factors, including organizational or human-element issues, which do not necessarily fit the scope of failure-based MBSA analyses. The following steps describe a generic method to systematically exclude these non–failure-based scenarios while retaining them in the ontology for reference or separate analysis.

The approach involves explicitly labelling each Loss Scenario within the ontology to indicate whether it is failure-based. This is achieved by introducing a property such as *:isFailureBased* or by categorizing scenarios into subclasses like *:FailureBasedScenario*, *:HumanFactorScenario*, or *:OrganizationalScenario*. For example:

```
:LS-X a :LossScenario, :
    FailureBasedScenario ;
     :hasController :<Controller> ;
     :hasControlAction :<ControlAction> ;
     :hasContext :<Context> ;
     :isFailureBased true .

:LS-Y a :LossScenario, :
    HumanFactorScenario ;
     :hasController :<Controller> ;
     :hasControlAction :<ControlAction> ;
     :hasContext :<Context> ;
     :isFailureBased false .
```

During the scenario generation phase, each Loss Scenario is evaluated and classified. This can be done manually by analysts or through predefined rules based on the scenario's attributes. For instance, scenarios involving sensor malfunctions, data corruption, or actuator failures are classified as failure-based, while those involving operator errors, procedural lapses, or organizational issues are classified as non–failure-based.

The analyst can finally query the relevant scenarios to apply in the MBSA analysis. Here is a query example:

```
SELECT ?scenario
WHERE {
?scenario a :LossScenario .
?scenario :isFailureBased true .
?scenario :hasController ?<Controller> .
?scenario :hasContext ?<Context> .
}
```

4. Case study

4.1. *The AIDA system*

The AIDA system (Aircraft Inspection by Drone Assistant) automates Pre-Flight Checks (PFCs) of civil aircraft using drones equipped with high-resolution cameras. Developed by IRT Saint Exupéry [b], it inspects hard-to-reach areas like upper surfaces and landing gear compartments, enhancing reliability and reducing human error while improving turnaround times. The system includes a quadrotor drone, a control desk for mission planning, and a communication link for coordination **?**.

AIDA operates in manual or automatic modes, using adaptive flight plans. It identifies anomalies such as structural damage or icing, stores inspection data in airline databases for traceability. Designed to meet aviation regulations, AIDA integrates into airport operations to boost safety, maintenance efficiency, and technology in aviation.

4.2. *STPA*

The full STPA analysis was performed to support this work, and a sample of each step is provided in this section.

4.2.1. *Scope of the analysis*

This step is about defining the boundaries of the analysis for the AIDA system. It involves identifying and categorizing potential losses and system-level hazards. Table 1 summarize the specific losses considered, ensuring that the analysis addresses all relevant safety concerns within the defined scope.

Table 1. Identified Losses for AIDA System

Code	Description
L-1	Loss/damage to the drone
L-2	Damage to the inspected aircraft
L-3	Damage to nearby aircraft/ infrastructures
L-4	Harm to airport staff/passengers
L-5	Loss of mission (pre-flight check)
L-6	Operational disruption

Table 2 shows the associated system-hazards that could lead to the associated losses in the worst case scenario.

Table 2. Identified System-Level Hazards for AIDA System

Code	Description
H-1	Operates outside geofenced boundaries [L1, L2, L3, L4 ,L5 ,L6]
H-2	Enters uncontrolled flight state [L1, L2, L3, L4 ,L5 ,L6]
H-3	Interacts with obstacles [L1, L2, L3, L4 ,L5 ,L6]
H-4	Produces invalid/incomplete inspection results [L5, L6]
H-5	Operates beyond design limitations [L1, L2, L3, L4 ,L5 ,L6]
H-6	Missions are delayed [L5, L6]

4.2.2. *Control Structure*

The control structure step outlines the relationships among the components of the AIDA System. By mapping how different controllers and controlled processes interact, this step provides a view of the flow of commands and information. Figure 2 focuses on the *Control Desk Controller*, which the authors selected to illustrate their approach.

[b]The AIDA drone is a public use case from IRT Saint Exupery and can be accessed at https://sahara.irt-saintexupery.com/AIDA

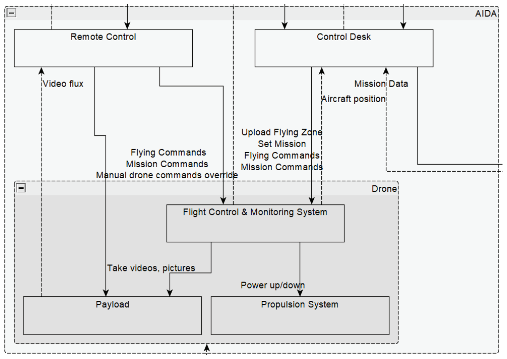

Fig. 2. Safety Control Structure of the AIDA system centered on the Control Desk Controller

4.2.3. *Unsafe Control Actions*

In this step, control actions that could lead to accidents are identified. As previously, the UCAs treated are associated with the *Control Desk* Controller, as shown in Table 3.

Table 3. Unsafe Control Actions for Mission Commands Controller

Not providing causes hazard	Providing causes hazard	Too early, too late, out of order
UCA-9: Control Desk does not provide the Mission Commands action when the mission is set to automatic inspection [H-4] **UCA-24:** Control Desk does not provide the Mission Commands action when the commands from the last mission are still uploaded when the new mission starts [H-4]	**UCA-10:** Control Desk provides the Mission Commands action when data is erroneous [H-4]	**UCA-11:** Control Desk provides the Mission Commands action too late when the mission has already started [H-4] **UCA-12:** Control Desk provides the Mission Commands action too early when another mission is still being processed [H-4]

4.2.4. *Loss Scenarios*

Loss scenarios detail the pathways through which UCAs can lead to the identified losses. Table 4 provides an example by illustrating how a specific UCA (i.e., *UCA-10*) can propagate through the system by generating the scenarios as described earlier.

4.3. *MBSA*

The MBSA model is represented in Fig. 3. It contains representations of the drone and the different components of the AIDA system with which it interacts.

At the component level, behaviour is represented with state machines representing the possible dysfunctional states of components (OK, LOST, ERRONEOUS), and some of its functional behaviours (e.g. AUTO/MANUAL modes).

4.4. *Integration*

For the integration of the solution, the authors selected *UCA-10* and *LLS-25* as examples to showcase their approach.

To formalize this Loss Scenario within the ontology, an instance of the Loss Scenario class is created, and categorized under *UnsafeController-Behavior*. The ontology represents the relationships between the controller, the control action, and the contextual conditions that render the action unsafe.

```
:LS_25 a :LossScenario, :
    UnsafeControllerBehavior ;
   :hasController :ControlDesk ;
   :hasControlAction :MissionCommands ;
   :hasContext :ErroneousData .
```

Each component of the scenario is mapped to corresponding MBSA elements through dedicated properties:

```
:ControlDesk :hasStatevariable "
    controlDeskState" .
:MissionCommands:hasFlowVariable "
    missionCommands" .
:ErroneousData :hasPossibleValue "
    dataValue" .
```

Table 4. Loss Scenarios of *UCA-10*

Unsafe Controller Behavior	**Unsafe Feedback Path**	**Unsafe Control Path**	**Unsafe Controlled Process Behavior**
LS-25: Control Desk provides the Mission Commands action - Control Desk received feedback that indicated that data is erroneous	**LS-26:** Feedback received by Control Desk does not adequately indicate that data is erroneous - it is true that data is erroneous	**LS-27:** Control Desk does not provide the Mission Commands action when data is erroneous - Flight Control & Monitoring System receives Mission Commands when data is erroneous	**LS-28:** The Mission Commands action is not received by Flight Control & Monitoring System when data is erroneous - Flight Control & Monitoring System responds

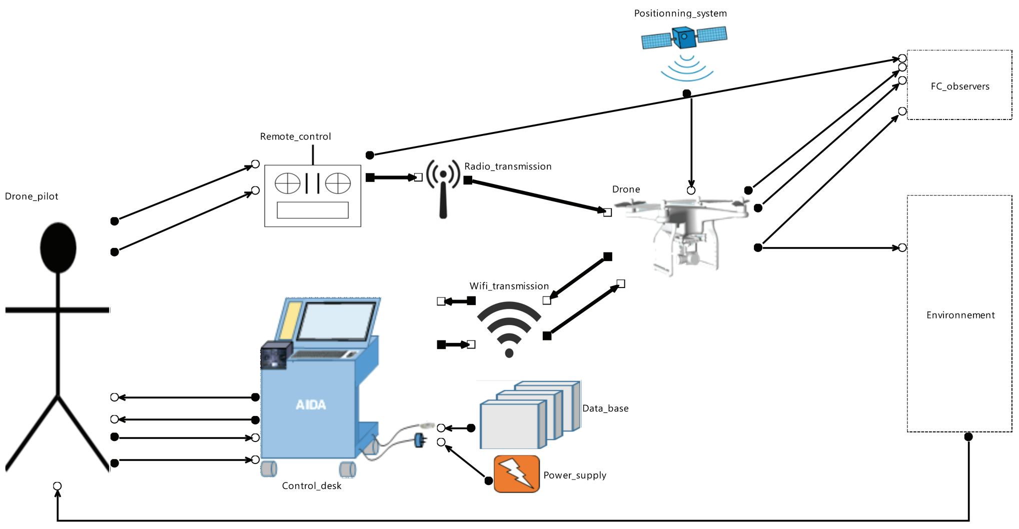

Fig. 3. MBSA physical model of the AIDA system

We can now compose the Boolean expression. The scenario *LS-25* is considered to occur when the Control Desk provides the Mission Commands and the data is erroneous:

```
LS-25 := "controlDeskState" AND "
    missionCommandsAction" AND "data" =
    Erroneous
```

The final step involves translating the Boolean expression into an AltaRica observer. The observer serves as a monitoring mechanism within the MBSA model, flagging when the specified unsafe conditions are met during simulation or analysis. Note that in the following sample, *"controlDeskState"*, *"missionCommandsAction"*, and *"data"* were replaced by the actual variables used in the SimfiaNeo tool:

```
observer LS-25 : boolean;
equation
LS-25 =
(Control_desk.Control_desk_common.
    Control_desk_common_state)
   = Nominal
AND (Control_desk.
WifiTransmissionUplink_FlightPlanAndZone)
   = Nominal
AND (Control_desk.iFlightPlanAndZone)
   = Erroneous;
```

5. Discussion

The results of this study provide valuable insights into the integration of STPA and MBSA using an ontology-driven approach. The methodology successfully demonstrated its ability to streamline the translation of STPA scenarios into MBSA-compatible artefacts. This process, applied with the AIDA system case study, significantly reduced manual effort and improved the precision of the scenario generation.

The structured filtering of STPA scenarios to identify those relevant to MBSA was another key outcome. By focusing only on failure-based scenarios, the approach ensured that MBSA models were not overwhelmed by irrelevant data, improving the efficiency and focus of the analysis.

However, the study also revealed certain weaknesses. The expression generation process has not yet been rigorously tested across all the different types of Loss Scenarios. Similarly, the filtering process, although effective, has a manual approach, opening a window for errors.

6. Conclusion and future work

This study introduced and tested an ontology-driven framework for integrating STPA and MBSA, with a focus on improving the efficiency and comprehensiveness of safety assessments in complex systems. By combining the systemic hazard identification of STPA with MBSA's detailed fault propagation capabilities, the framework offers a promising approach to addressing the challenges posed by modern safety-critical systems.

Despite its demonstrated potential, the framework has limitations that require attention. The logic behind the expression generation must be enhanced to ensure robustness across a wider range of scenarios and systems. Similarly, the reliance on manual filtering for failure-based scenarios poses challenges for scalability and consistency. While preliminary efforts to automate this process through text extraction and ontology mapping are encouraging, further work is needed.

In conclusion, while this work represents a significant step toward improving safety assessment methodologies, it also highlights the complexity of harmonizing STPA and MBSA. The proposed framework provides a solid foundation, and with continued refinement, it has the potential to make safety assessments more efficient, reliable, and scalable for complex, safety-critical systems.

References

ARP4761A (2023). Arp4761a: Guidelines for conducting the safety assessment process on civil aircraft, systems, and equipment. Technical report, SAE International.

Batteux, M., T. Prosvirnova, and A. Rauzy (2019). AltaRica 3.0 in ten modelling patterns. *International Journal of Critical Computer-Based Systems 9*(1-2), 133–165.

Boiteau, M., Y. Dutuit, A. Rauzy, and J.-P. Signoret (2006). The altarica data-flow language in use: modeling of production availability of a multi-state system. *Reliability Engineering & System Safety 91*(7), 747–755.

Bouissou, M., H. Bouhadana, M. Bannelier, and N. Villatte (1991). Knowledge modelling and reliability processing: Presentation of the figaro language and associated tools. *IFAC Proceedings Volumes 24*(13), 69–75. IFAC Symposium on Safety of Computer Control Systems 1991 (SAFECOMP'91), Trondheim, Norway, 30 October-1 November 1991.

Chopart, M., J. Vidalie, A. Lališ, and K. Grötschelová (2024). Harmonizing safety perspectives: Integrating stpa outputs into mbsa for comprehensive aircraft safety assessment. In *2024 New Trends in Civil Aviation (NTCA)*, pp. 261–267.

Gudemann, M. and F. Ortmeier (2010, 12). A framework for qualitative and quantitative formal model-based safety analysis. pp. 132 – 141.

James Elizebeth, M., S. Khastgir, I. Babaev, S. Chen, and P. Jennings (2023, March). Comparison of FTA and Stpa Approaches: A Brake-by-Wire Case Study.

Leveson, N. (2018). STPA Handbook.

Thomas, J. (2024). STPA Step 4 Building Scenarios A Formal Scenario Approach. *MIT STAMP Workshop*.