# A FRAMEWORK FOR QUANTIFYING SAFETY-CRITICAL THREAT SCENARIOS USING BAYESIAN NETWORKS

M. Jahn*
* Airbus Defence and Space GmbH, Airbus-Allee 1, 28199 Bremen, Germany

**Abstract**
Safety and cybersecurity are often addressed separately in system design, despite their increasing interdependence. This paper presents a quantitative framework for assessing safety-critical threat scenarios during early design stages. Based on Bayesian networks, the approach models attack propagation and system failures, enabling engineers to evaluate the likelihood of compromise and safety impacts across design alternatives. A key feature is dynamic updating: changes such as component failures or compromises can be introduced to reflect evolving system states and immediately update risk assessments. The framework integrates naturally with model-based systems engineering by providing a practical tool for identifying high-risk paths and, ultimately, allows for designs that are safer and more secure.

**Keywords**
Threat scenarios; Attack graphs; Fault Trees; Bayesian Network

## 1. INTRODUCTION

Safety has long been a fundamental principle in system design. Across industries, engineers perform hazard analyses, introduce redundancy, and comply with certification standards to minimize the likelihood of catastrophic failures. In domains such as aviation, this safe-by-design approach has proven highly effective. With increasing connectivity and reliance on commercial off-the-shelf components, however, traditional safety measures are no longer sufficient. Cybersecurity vulnerabilities can undermine safety-critical functions, making it necessary to address safety and security in an integrated approach from the earliest design stages.
Model-based systems engineering (MBSE) [1] offers a structured way to capture system requirements and dependencies, and languages such as SysML [2] and RAAML [3] enable safety and security information to be modeled within a unified framework. Yet hazard identification methods, such as System Theoretic Process Analysis (STPA) [4,5], remain largely manual and require significant expertise. Moreover, existing approaches usually do not account for the dynamic nature of systems, where vulnerabilities emerge, components degrade, and updates alter risk profiles.
Recent research combines system-theoretic safety analysis with formal verification and security modeling to identify safety-critical attack paths [5–7]. Building on this foundation, this article presents a framework that extends MBSE with dynamic threat analysis. The framework supports the quantification of risk, simulation of failures and attacks, and evaluation of design alternatives under changing conditions, thereby advancing integrated safety–security analysis for complex systems. As a result, it can be used to support engineers in performing rapid root-cause analysis and exploring “what-if” scenarios. In more detail, the article contributes:

- methods to ingest established models (fault trees, attack graphs, STPA-based scenarios) in practical formats and to represent them in a unified DAG for joint probability–time analysis,
- mechanisms for efficient updating as new evidence becomes available (e.g., compromised nodes, failures, emerging CVEs), ensuring that scenario assessments remain current, and
- decision-support for MBSE workflows through explainable outputs and threshold-based safety states, illustrated by case studies.

The remainder of this article is organized as follows: After recalling the most important concepts and approaches in Section 2, we present the framework in Section 3 including the assumptions and practical conventions used by the framework. We then provide numerical results in Section 4 using an abstract threat-scenario example and a real-world radio altimeter case. In Section 5, we comment on limitations of the approach and conclude the article in Section 6.

## 2. BACKGROUND

As we are focusing on safety-critical threat scenarios, we briefly first recall the most important concepts and approaches.

doi: 10.25967/650096

## 2.1. Hazard analysis

A hazard analysis is a systematic process to identify, evaluate, and control potential sources of harm in a process or system. In aviation and other safety-critical domains, the outcome typically includes the *likelihood*, i.e., the joint probability of a hazard occurring, impacting safety, and contributing to an incident. Standards such as SAE ARP4761, DO-178C, or ISO 26262 typically define mappings between likelihood and severity categories and specify acceptable levels of risk for systems and systems of systems.

## 2.2. System Theoretic Accident Model and Processes

The system theoretic accident model and processes (STAMP) [5] is a system-theoretic approach to accident causality. Unlike traditional reliability-based models, STAMP views safety as an emergent control problem. Accidents arise from inadequate application of safety controls, incomplete feedback, or dysfunctional interactions in hierarchical control structures, even when individual components operate as intended.

## 2.3. System Theoretic Process Analysis

System Theoretic Process Analysis (STPA) [4, 5] builds on STAMP to provide a top-down hazard analysis method. Therefore, it works at a *functional* level rather than requiring detailed item-level specifications. The method is used to identify unsafe control actions and causal scenarios, enabling hazards to be addressed early in the design process, when corrective actions are less costly and more effective.

## 2.4. STPA-Sec

An extension of STPA is the method STPA-Sec [7] which adapts the original method STPA to security. The approach applies the same control-theoretic reasoning to examine how malicious actions or vulnerabilities may lead to losses, thereby integrating safety and security analysis in a single framework.

## 2.5. Fault tree analysis

Fault Tree Analysis (FTA) [8] is a deductive, failure-oriented method widely used across safety-critical industries. It models how combinations of lower-level events may lead to system-level failures. FTA is most effective for well-defined architectures with item-level specifications, where failure probabilities of components can be quantified. This makes FTA a powerful tool for probabilistic risk assessment once the system architecture is sufficiently mature. However, its bottom-up focus can limit its usefulness in the early stages of design, when detailed specifications are not yet available. As such, FTA and STPA are often considered complementary: STPA supports early functional hazard identification, while FTA provides quantitative assessment of risk in later design phases.

## 2.6. Attack graphs

Attack graphs are graphical models that represent potential sequences of actions or events (i.e., attack paths) that an adversary may take to compromise a system. Each path within an attack graph illustrates how vulnerabilities, configuration-related mistakes, and privileges can be exploited in sequence to move from an initial state to a goal state (e.g., unauthorized access to a critical component).
In the literature, two main types of attack graphs are commonly distinguished [9]:

- **State-based attack graphs**, where each node represents a complete system state and edges represent transitions between states caused by exploits. This representation captures system evolution over time but tends to suffer from state-space explosion in large systems.
- **Logical attack graphs**, where the graph is bipartite with nodes representing either *exploits* or *security conditions*. Edges describe dependencies, such as which conditions must hold for an exploit to succeed and what conditions are enabled once it has executed. Logical graphs are more scalable and allow abstraction away from individual states.

Attack graphs are widely used for security analysis, since they allow systematic identification of attack paths, assessment of their likelihood or cost, and prioritization of defensive measures.

## 2.7. Threat scenarios

In cybersecurity standards, e.g., [10], a threat scenario is defined as a set of discrete threat events, associated with one or more threat sources, that are partially ordered in time. Such scenarios typically include a chronological and functional sequence of steps that could be used to exploit system vulnerabilities and, as a result, provide a structured description of how a system could be attacked under specific conditions.
In safety-critical domains, threat scenarios extend beyond the traditional focus and also address confidentiality, integrity, and availability of systems. They must capture cyber-physical attack paths that can degrade safety constraints and drive the system into hazardous states. By combining safety-oriented methods with security analysis, such as STPA-Sec, these scenarios describe how attacks unfold and how they can lead to unsafe control actions or failures in critical control structures. This dual perspective ensures that both the likelihood of attacks and their safety impact are systematically assessed.

## 3. PREDICTION ENGINE

The presented framework is designed for researchers to evaluate safety-critical threat scenarios for a system or system of systems. The goal is to predict the "safety state" of a system (or a system of systems). Fundamentally, the framework is based on Bayesian networks, a concept which is well-established in the field of cybersecurity and risk analysis.

## 3.1. Bayesian networks

Bayesian networks provide the probabilistic foundation for our framework, allowing heterogeneous evidence—such as component failure rates, exploit success probabilities, and monitoring data—to be combined within a single causal model [11]. As probabilistic graphical models, they represent conditional dependencies among random variables using a directed acyclic graph (DAG). Therein, nodes correspond to random variables and edges indicate direct influence, with each parent node affecting the probability distribution of its children.
Formally, given $\mathbf{X} = \{X_1, \ldots, X_n\}$, the joint probability distribution (JPD) can be factorized as:

$$p(\mathbf{X}) = \prod_{i=1}^{n} p(X_i \mid \mathbf{pa}_i),$$

where $\mathbf{pa}_i$ denotes the set of parent nodes of $X_i$ [12]. This factorization exploits conditional independence, avoiding the exponential growth of joint probability tables and enabling efficient inference.

## 3.2. Bayesian attack graphs

Bayesian networks are widely applied in security modeling because they capture uncertainty in attacker behavior and support both static and dynamic analysis [13–15]. When combined with logical attack graphs, nodes represent resources or attacks and edges indicate causal relationships. Nodes are typically modeled as Bernoulli variables, with $p(X_i) = \Pr(X_i = T)$ representing compromise likelihood and a node's probability of compromise depends on its parents, with logical dependencies modeled via *AND* or *OR* relations. A logical *AND* can be expressed as [12]:

$$p(X_i|\mathbf{pa}_i) = \begin{cases} 0, & \exists X_j \in \mathbf{pa}_i | X_j = F \\ \prod_{j=1}^{l} p_{v_j}, & \text{otherwise} \end{cases}$$

while a logical *OR* is given by

$$p(X_i|\mathbf{pa}_i) = \begin{cases} 0, & \forall X_j \in \mathbf{pa}_i | X_j = F \\ 1 - \prod_{j=1}^{l} (1 - p_{v_j}), & \text{otherwise} \end{cases}$$

where $p_{v_j}$ denotes the probability of exploiting vulnerability $v_j$.

## 3.3. Bayesian fault graphs

Fault trees can be modeled analogously by using Bayesian networks, where nodes represent component failures and edges describe causal dependencies. Prior probabilities can encode fault likelihoods while unconditional transition probabilities are set to $1$. With this, Bayesian inference can be used to update failure risks as new evidence becomes available. This approach enables probabilistic fault analysis beyond traditional deterministic FTA and allows for performing root-cause analysis.

## 3.4. Assumptions

Aside from using Bayesian networks, the proposed framework relies on a set of conceptual and implementation-level assumptions.

### 3.4.1. Conceptual Assumptions

1) **Availability of threat scenarios.** A threat scenario captures combinations of system states that can lead to unsafe control actions (STPA) and, ultimately, hazardous conditions. Threat scenarios involve one or multiple target systems, e.g., in aviation the flight control management and connected systems [6]. Since there are various reasons for systems to be e.g. not available (failure or attack), we use multiple graphs per system and distinguish fault graphs from attack graphs. As we also need to consider combination of failures and attacks, the number of DAGs can be rather high.
2) **Availability of fault trees.** Fault trees are assumed to be available for all safety-critical systems under analysis. This assumption is consistent with current engineering practice, where Fault Tree Analysis (FTA) is widely used as part of certification and safety assurance processes.
3) **Availability of attack graphs.** We assume that a security engineer has conducted threat modeling and constructed attack graphs. Such graphs may be generated manually or by automated methods [6, 15]. These graphs capture feasible attack paths and provide the security-oriented counterpart to the fault trees.
4) **Attacker capabilities.** Following [12], we assume that the attacker capabilities do not change during an attack and that probabilities of successfully exploiting a vulnerability remain constant over time and do not increase if the same exploit could be used for a different system.
5) **Consideration of time.** Unlike probabilistic graphical models, there is no widely accepted formalism for representing temporal aspects. We consider time as transition time on edges. For nodes with multiple predecessors, *AND* takes the maximum incoming time and *OR* the minimum. When a node is compromised, its time is set to $0$ s and accumulated only for the successors.

### 3.4.2. Practical implementation aspects.

We adopt the following conventions in our framework:

- **Safety-state classification and scenario definition.** The current implementation classifies safety state as *safe*, *medium risk*, or *high risk* based on user-defined probability and time thresholds. A threat scenario is defined as a combination of node-states (targets) that result in unsafe control actions per STPA.

- **Canonical DAG representation.** All inputs are converted into DAGs. Nodes represent events or states (basic faults, compromise states, enabling conditions, logical gate outputs). Edges represent causal/transition relationships from which conditional probability distributions (CPDs) and temporal dependencies are constructed. Logical gates are implemented as deterministic CPDs. The overall structure must be acyclic on ingestion; cycles in sources have to be resolved during model preparation.
- **Scenario queries.** Scenario probabilities are evaluated via Bayesian inference, while the time to threat-materialization is computed as earliest-arrival in the time network.
- **Input formats.** Graphs are provided via CSV files for arcs and nodes (consistent with MulVAL exports), and experiments/scenarios via YAML, as detailed in Section 3.7
- **Consideration of probabilities and times:**
  - **Attack graphs (probabilities):** we primarily use an edge-based (condensed) encoding in which unconditional transition probabilities are attached to edges, representing the likelihood that an exploit succeeds along a dependency. An equivalent node-based encoding with explicit attacker/likelihood nodes and logical connectors (AND/OR) is possible but not required.
  - **Fault trees (probabilities):** failure rates/probabilities for basic events (typically hardware) are treated as priors on leaf nodes; gate outputs are deterministic. Transitions along fault-propagation edges are usually set to 1 for hardware. For software-related faults producing incorrect values, checks and plausibility filters may reduce the risk which can be considered by using probabilities that are less than 1.
  - **Timing:** transition times (or time distributions) are assumed available for edges to enable temporal analysis (earliest-arrival times to targets).

## 3.5. Architecture

The framework is designed for use in safety- and mission-critical domains, where rapid adaptation to new vulnerabilities, failures, or attack information is essential. We used a modular architecture where each component has a clear responsibility. This design simplifies updates, extensions, and independent validation. The modularity also enables scalable analysis by separating probability- and time-based reasoning while ensuring consistency through a central coordinator, the `FrameworkManager`.

`FrameworkManager`
- Central coordinator managing objects such as `GraphData`, `AttackAndFaultGraph`, and `ThreatScenarioManager`.
- Propagates updates (e.g., compromised nodes, failures, new CVEs) across all dependent components.

`GraphData`
- Builds DAGs from CSV input or synthetic data.
- Maintains the global structure of nodes, edges, and subgraphs, ensuring efficient partitioning when possible.

`AttackAndFaultGraph`
- Represents a single subgraph derived from `GraphData`.
- Interfaces with `BayesianNetwork` for probability computations and `TimeNetwork` for temporal evaluations.
- Propagates updates (e.g., priors, evidence, transition probabilities) to both associated networks.

`BayesianNetwork`
- Constructs a Bayesian network from an subgraph.
- Computes probabilities of reaching specified target nodes.

`TimeNetwork`
- Builds a temporal dependency model from an `AttackAndFaultGraph`.
- Estimates minimum times required to compromise target nodes.

`ThreatScenarioManager`
- Manages collections of `ThreatScenario` instances.
- Updates `ThreatScenario` and related safety states based on data updates
- Ensures consistency of scenarios.

`ThreatScenario`
- Combines multiple targets across different subgraphs.
- Aggregates probability and time metrics to provide overall scenario evaluations.

Figure 1 provides a conceptual overview of the main classes and their relations. Although the diagram resembles a UML dependency view, it serves as a high-level functional illustration rather than a formal specification.

## 3.6. Decision support and what-if analysis

The framework supports design-time exploration of alternatives by:
- Modifying priors and edge parameters (probabilities, times) to emulate design choices (e.g., added redundancy, rate limiting, authentication, hardened components).
- Injecting controls as nodes/edges (detectors, plausibility checks) to consider propagation.
- Comparing scenarios across designs via differences in target-node probabilities/times and threshold-based safety states.
- Identifying and ranking critical paths or root causes to guide engineers where to focus efforts.

Design iterations can be scripted via the YAML config or applied interactively through updates to nodes/edges via an API.

## 3.7. Definition of experiments

Listing 1 shows how experiments are defined in the framework. The YAML structure specifies input data, analysis options, and scenario definitions. Some fields are self-explanatory (e.g., `name`, `results_path`), but others require more detail:

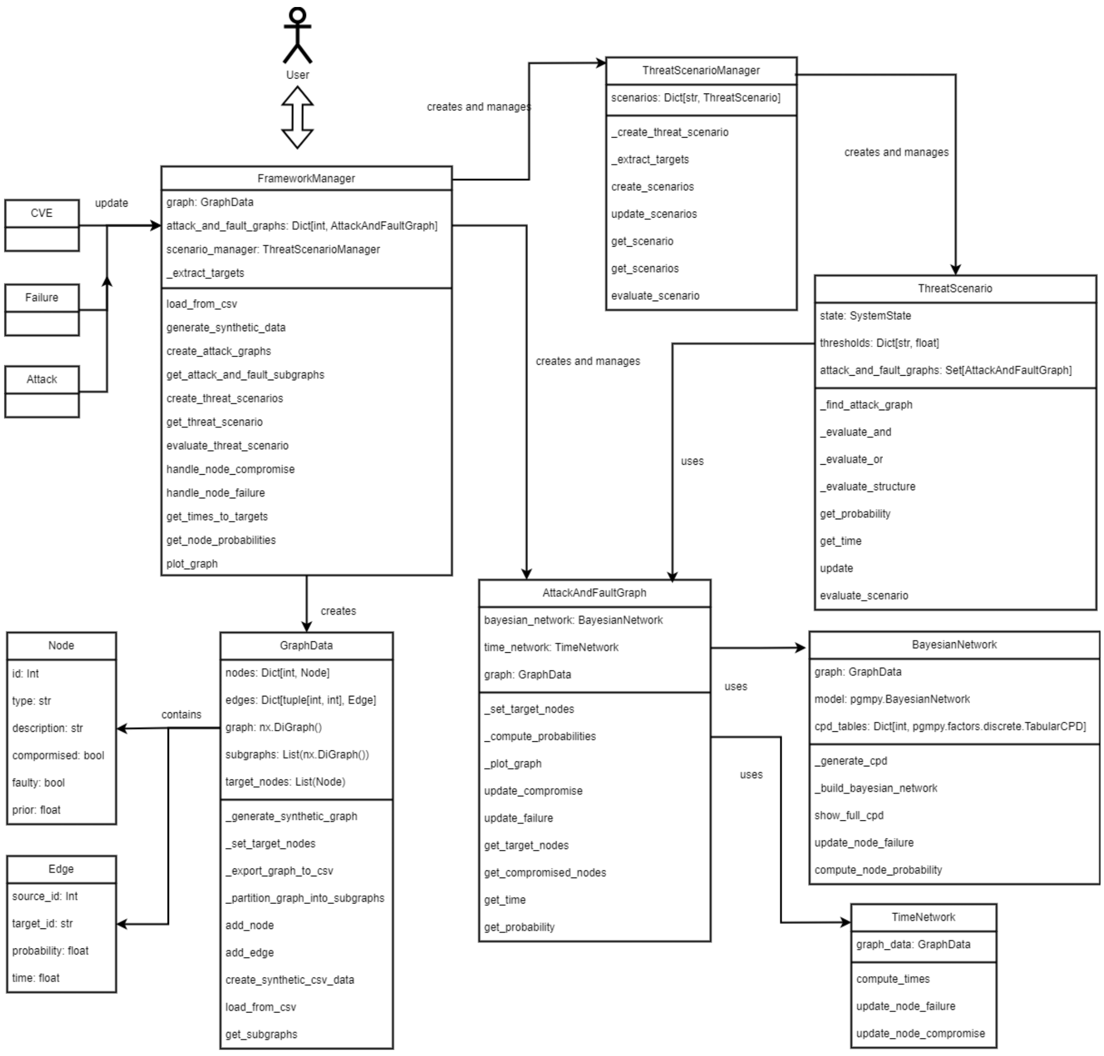

**FIG 1. Functional overview of the framework and its core components.**

- **Graph data:** The framework expects input in CSV format, consistent with the MulVAL export [16], a common attack-graph generator. Paths to these files must be provided in `arcs` and `vertices`.
- **Scenarios:** Defined under `scenarios`, each threat scenario combines target nodes using logical operators (`and`/`or`). This recursive structure allows for flexible composition of conditions.
- **Thresholds (optional):** If probability and time thresholds are defined, the framework evaluates results against them and returns system states (`safe`, `medium risk`, `high risk`). If omitted, raw values are reported and the state remains `unknown`.
- **Optional fields:** `compromised_nodes`, `failures`, and `cves` allow to use predefined configurations in experiments. During runtime, such updates can be injected dynamically via an API.

## 3.8. Dynamic updates

A central feature of the framework is the dynamic update of both, nodes and edges, during runtime. Nodes can be updated with new evidence or prior information, e.g., compromised components or failures, while edges can be modified by adjusting unconditional transition probabilities. This is essential for systems that require continuous threat monitoring, particularly in safety-critical domains where conditions evolve rapidly. Beyond run-time evidence ingestion, the same mechanism enables design-time "what-if" studies: engineers can alter priors, transition probabilities and time and immediately quantify their impact.

Dynamic updates serve several purposes:

- **Adaptability:** Attack graphs and combined fault–attack models often represent environments where vulnerabilities, failures, or adversarial behavior change over time. Propagating updates ensures the analysis reflects the current system state.
- **Evidence integration:** Bayesian inference allows new observations—such as detected compromises or system malfunctions—to be incorporated directly into probability calculations. This facilitates the identification of critical paths and supports prioritization of defensive measures.
- **Risk mitigation:** Real-time updates enable continuous reassessment of the likelihood of threats and their impact on safety. Without this capability, outdated models could lead to inaccurate risk estimates and delayed or ineffective responses.

```
name: <name of the example>
results_path: <path to results, e.g. "./results/">

bayesian_method: <"exact" or "approximate">

arcs: <path to csv file e.g. "./edges.csv">
vertices: <path to csv file e.g. "./nodes.csv">

scenarios:
  <name, e.g. "TS1">:
    condition:
      <"and" or "or">:
        - <"and" or "or"> : <list of target nodes>
        - <"and" or "or"> : <list of target nodes>
    thresholds: <optional>
      threshold_probability_high_risk: <float>
      threshold_probability_medium_risk: <float>
      threshold_time_high_risk: <number>
      threshold_time_medium_risk: <number>

  <name, e.g. "TS2">:
    [...]

compromised_nodes: <optional>
  - <node number or empty>

failures: <optional>
  - <list of node number and failure type,
  e.g. [3, "dos_failure"]>

cves: <optional>
  - <list of node number and CVE severity,
  e.g. [5, "high"]>
```

**Listing 1. Experiment definition in YAML.**

When an update occurs, the framework automatically propagates changes through all affected components. Specifically:

1) All attack or fault subgraphs containing the modified node or edge are identified.
2) Probabilities (via Bayesian inference, which can affect upstream and downstream nodes) and timing values (forward-only propagation) are recomputed for the affected subgraphs.
3) Dependent threat scenarios are updated, triggering re-evaluation of their system-level safety states.

The impact of updates is twofold: compromised nodes or system failures are treated as evidence within the Bayesian network, while new CVEs dynamically alter transition probabilities and times. Scenario evaluations are then recalculated on the basis of the updated subgraphs, ensuring that both ongoing attacks and emerging vulnerabilities are consistently reflected in the system's safety assessment.

## 4. NUMERICAL RESULTS

We demonstrate the functionality of the framework by considering an abstract example in Section 4.1 and then consider a real-world example [17] in Section 4.2.

### 4.1. Example: Threat scenario

To illustrate the use of the framework for analyzing threat scenarios, we manually design an abstract setup based on a system graph that includes multiple initial and target nodes and provide time-related information in addition to transition probabilities. We

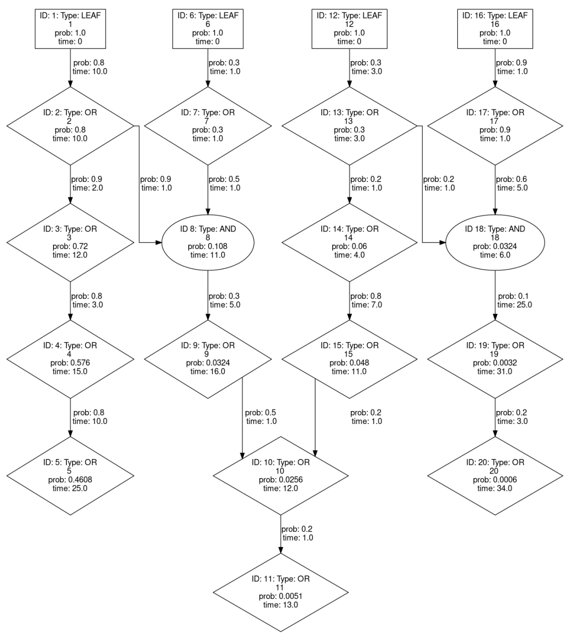
ID: 1: Type: LEAF 1 prob: 1.0 time: 0
ID: 6: Type: LEAF 6 prob: 1.0 time: 0
ID: 12: Type: LEAF 12 prob: 1.0 time: 0
ID: 16: Type: LEAF 16 prob: 1.0 time: 0
prob: 0.8 time: 10.0
prob: 0.3 time: 1.0
prob: 0.3 time: 3.0
prob: 0.9 time: 1.0
ID: 2: Type: OR 2 prob: 0.8 time: 10.0
ID: 7: Type: OR 7 prob: 0.3 time: 1.0
ID: 13: Type: OR 13 prob: 0.3 time: 3.0
ID: 17: Type: OR 17 prob: 0.9 time: 1.0
prob: 0.9 time: 2.0
prob: 0.9 time: 1.0
prob: 0.5 time: 1.0
prob: 0.2 time: 1.0
prob: 0.2 time: 1.0
prob: 0.6 time: 5.0
ID: 3: Type: OR 3 prob: 0.72 time: 12.0
ID 8: Type: AND 8 prob: 0.108 time: 11.0
ID: 14: Type: OR 14 prob: 0.06 time: 4.0
ID 18: Type: AND 18 prob: 0.0324 time: 6.0
prob: 0.8 time: 3.0
prob: 0.3 time: 5.0
prob: 0.8 time: 7.0
prob: 0.1 time: 25.0
ID: 4: Type: OR 4 prob: 0.576 time: 15.0
ID: 9: Type: OR 9 prob: 0.0324 time: 16.0
ID: 15: Type: OR 15 prob: 0.048 time: 11.0
ID: 19: Type: OR 19 prob: 0.0032 time: 31.0
prob: 0.8 time: 10.0
prob: 0.5 time: 1.0
prob: 0.2 time: 1.0
prob: 0.2 time: 3.0
ID: 5: Type: OR 5 prob: 0.4608 time: 25.0
ID: 10: Type: OR 10 prob: 0.0256 time: 12.0
ID: 20: Type: OR 20 prob: 0.0006 time: 34.0
prob: 0.2 time: 1.0
ID: 11: Type: OR 11 prob: 0.0051 time: 13.0

**FIG 2. System graph including probabilities and transition times.**

```
scenarios :
  "TS1":
    "condition":
      "or":
        - "and": [5, 20]
      "thresholds":
        "threshold_probability_high_risk": 0.04,
        "threshold_probability_medium_risk": 0.01,
        "threshold_time_high_risk": 8,
        "threshold_time_medium_risk": 15,
  "TS2":
    "condition":
      "or":
        - "and": [11]
      "thresholds":
        "threshold_probability_high_risk": 0.1,
        "threshold_probability_medium_risk": 0.04,
        "threshold_time_high_risk": 2,
        "threshold_time_medium_risk": 10,
```

**Listing 2. Definition of threat scenarios.**

define different threat scenarios that use the nodes $X_5$, $X_{11}$, and $X_{20}$ as target nodes (subsystems), see Listing 2.

**Initial state.** We compute the initial probabilities for the target systems (nodes); see Figure 2. The values for the target nodes are

$$\begin{aligned} p(X_5) &= 0.4608, \\ p(X_{11}) &= 0.0051, \\ p(X_{20}) &= 0.0006, \end{aligned}$$

with corresponding times $t(X_5) = 25.0$, $t(X_{11}) = 13.0$, and $t(X_{20}) = 34.0$. With respect to the threat scenarions `TS1` and `TS2`, this means it is

$$\begin{aligned} p(\text{TS1}) &= 0.0003, \\ t(\text{TS1}) &= 25, \\ p(\text{TS2}) &= 0.0051, \\ t(\text{TS2}) &= 13, \end{aligned}$$

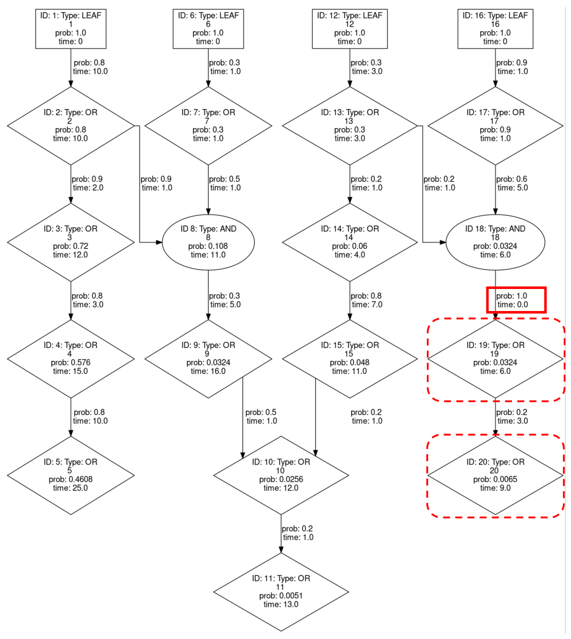
ID: 1: Type: LEAF 1 prob: 1.0 time: 0
ID: 6: Type: LEAF 6 prob: 1.0 time: 0
ID: 12: Type: LEAF 12 prob: 1.0 time: 0
ID: 16: Type: LEAF 16 prob: 1.0 time: 0
prob: 0.8 time: 10.0
prob: 0.3 time: 1.0
prob: 0.3 time: 3.0
prob: 0.9 time: 1.0
ID: 2: Type: OR 2 prob: 0.8 time: 10.0
ID: 7: Type: OR 7 prob: 0.3 time: 1.0
ID: 13: Type: OR 13 prob: 0.3 time: 3.0
ID: 17: Type: OR 17 prob: 0.9 time: 1.0
prob: 0.9 time: 2.0
prob: 0.9 time: 1.0
prob: 0.5 time: 1.0
prob: 0.2 time: 1.0
prob: 0.2 time: 1.0
prob: 0.6 time: 5.0
ID: 3: Type: OR 3 prob: 0.72 time: 12.0
ID 8: Type: AND 8 prob: 0.108 time: 11.0
ID: 14: Type: OR 14 prob: 0.06 time: 4.0
ID 18: Type: AND 18 prob: 0.0324 time: 6.0
prob: 0.8 time: 3.0
prob: 0.3 time: 5.0
prob: 0.8 time: 7.0
prob: 1.0 time: 0.0
ID: 4: Type: OR 4 prob: 0.576 time: 15.0
ID: 9: Type: OR 9 prob: 0.0324 time: 16.0
ID: 15: Type: OR 15 prob: 0.048 time: 11.0
ID: 19: Type: OR 19 prob: 0.0324 time: 6.0
prob: 0.8 time: 10.0
prob: 0.5 time: 1.0
prob: 0.2 time: 1.0
prob: 0.2 time: 3.0
ID: 5: Type: OR 5 prob: 0.4608 time: 25.0
ID: 10: Type: OR 10 prob: 0.0256 time: 12.0
ID: 20: Type: OR 20 prob: 0.0065 time: 9.0
prob: 0.2 time: 1.0
ID: 11: Type: OR 11 prob: 0.0051 time: 13.0

**FIG 3. System graph including updated probabilities and transition times. The misconfiguration at node $X_{19}$ is highlighted in red, showing the increased transition probability ($1.0$) and reduced transition time ($0.0$).**

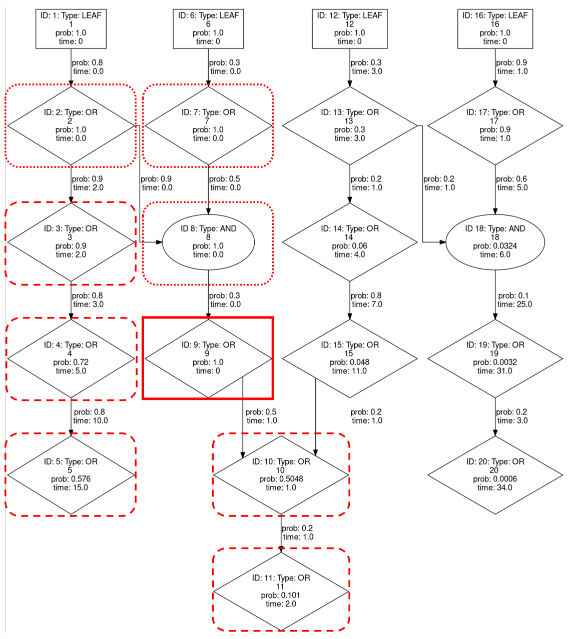
ID: 1: Type: LEAF 1 prob: 1.0 time: 0
ID: 6: Type: LEAF 6 prob: 1.0 time: 0
ID: 12: Type: LEAF 12 prob: 1.0 time: 0
ID: 16: Type: LEAF 16 prob: 1.0 time: 0
prob: 0.8 time: 0.0
prob: 0.3 time: 0.0
prob: 0.3 time: 3.0
prob: 0.9 time: 1.0
ID: 2: Type: OR 2 prob: 1.0 time: 0.0
ID: 7: Type: OR 7 prob: 1.0 time: 0.0
ID: 13: Type: OR 13 prob: 0.3 time: 3.0
ID: 17: Type: OR 17 prob: 0.9 time: 1.0
prob: 0.9 time: 2.0
prob: 0.9 time: 0.0
prob: 0.5 time: 0.0
prob: 0.2 time: 1.0
prob: 0.2 time: 1.0
prob: 0.6 time: 5.0
ID: 3: Type: OR 3 prob: 0.9 time: 2.0
ID 8: Type: AND 8 prob: 1.0 time: 0.0
ID: 14: Type: OR 14 prob: 0.06 time: 4.0
ID 18: Type: AND 18 prob: 0.0324 time: 6.0
prob: 0.8 time: 3.0
prob: 0.3 time: 0.0
prob: 0.8 time: 7.0
prob: 0.1 time: 25.0
ID: 4: Type: OR 4 prob: 0.72 time: 5.0
ID: 9: Type: OR 9 prob: 1.0 time: 0
ID: 15: Type: OR 15 prob: 0.048 time: 11.0
ID: 19: Type: OR 19 prob: 0.0032 time: 31.0
prob: 0.8 time: 10.0
prob: 0.5 time: 1.0
prob: 0.2 time: 1.0
prob: 0.2 time: 3.0
ID: 5: Type: OR 5 prob: 0.576 time: 15.0
ID: 10: Type: OR 10 prob: 0.5048 time: 1.0
ID: 20: Type: OR 20 prob: 0.0006 time: 34.0
prob: 0.2 time: 1.0
ID: 11: Type: OR 11 prob: 0.101 time: 2.0

**FIG 4. System graph after compromise of node $X_9$. The compromised node is highlighted with a solid red box, while affected nodes are shown with dashed red boxes. The updated probabilities and transition times reflect both back- and forward-propagation effects.**

As the values remain below (probability) or above (time) the respective thresholds defined in Listing 2, the system state is considered *safe* for both scenarios.

**Misconfiguration of $X_{19}$.** Next, we assume that the system denoted by $X_{19}$ is misconfigured and, as a result, the probability for an attacker to successfully compromise the system has increased while the time to exploit this is reduced. This change is highlighted in red in Figure 3. The transition probability at node $X_{19}$ is now $1.0$, while the corresponding transition time is reduced to $0.0$. As this node $X_{19}$ is only related to the target system (node) of the first threat scenario, we verify that the data for the second threat scenario does not change and that the probabilities and times for the first threat scenario are updated accordingly:

$$\begin{aligned} p(X_5) &= 0.4608, \\ p(X_{11}) &= 0.0051, \\ p(X_{20}) &= 0.0065, \end{aligned}$$

with corresponding times $t(X_5) = 25.0$, $t(X_{11}) = 13.0$, and $t(X_{20}) = 9.0$. Looking at the threat scenarions TS1 and TS2, it is

$$\begin{aligned} p(\text{TS1}) &= 0.0030, \\ t(\text{TS1}) &= 9, \\ p(\text{TS2}) &= 0.0051, \\ t(\text{TS2}) &= 13, \end{aligned}$$

When comparing the results to the thresholds, we can see that the probability of the threat scenario is still below the defined thresholds, however, due to the time threshold the state of the system in scenario TS1 is no longer *safe* but *at medium risk*.

**Compromise of $X_9$.** Finally, we assume that the system described by node $X_9$ has been compromised. The compromise of $X_9$ sets its probability to $1.0$ and the time to reach the node is set to $0.0$. Due to the set-up, we have to update all nodes and paths that are connected to this node. Consequently, the resulting probabilities of the target nodes are:

$$\begin{aligned} p(X_5) &= 0.576, \\ p(X_{11}) &= 0.101, \\ p(X_{20}) &= 0.0006, \end{aligned}$$

with times $t(X_5) = 15.0$, $t(X_{11}) = 2.0$, and $t(X_{20}) = 34.0$. Here, the results for the threat scenarios are

$$\begin{aligned} p(\text{TS1}) &= 0.00037, \\ t(\text{TS1}) &= 15, \\ p(\text{TS2}) &= 0.101, \\ t(\text{TS2}) &= 2, \end{aligned}$$

which means that for TS1 the system has to be considered to be at *medium risk* while for TS2 the system state is *high risk*.

The situation is highlighted in Figure 4. In the visualization, $X_9$ is marked with a solid red box, and all affected nodes are marked with dashed red boxes. The different styles also indicate the propagation semantics: probabilities are updated via Bayesian inference and can affect both upstream and downstream nodes (back- and forward-propagation), whereas transition times propagate forward along edges only.

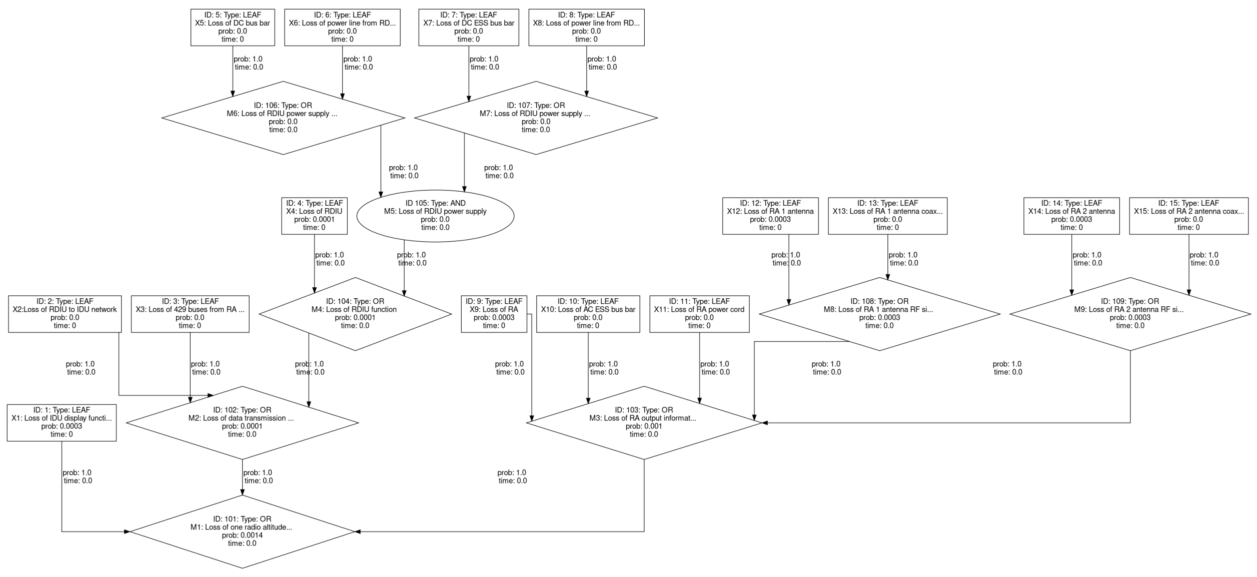
ID: 5: Type: LEAF X5: Loss of DC bus bar prob: 0.0 time: 0
ID: 6: Type: LEAF X6: Loss of power line from RD... prob: 0.0 time: 0
ID: 7: Type: LEAF X7: Loss of DC ESS bus bar prob: 0.0 time: 0
ID: 8: Type: LEAF X8: Loss of power line from RD... prob: 0.0 time: 0
ID: 106: Type: OR M6: Loss of RDIU power supply ... prob: 0.0 time: 0.0
ID: 107: Type: OR M7: Loss of RDIU power supply ... prob: 0.0 time: 0.0
ID: 4: Type: LEAF X4: Loss of RDIU prob: 0.0001 time: 0
ID 105: Type: AND M5: Loss of RDIU power supply prob: 0.0 time: 0.0
ID: 12: Type: LEAF X12: Loss of RA 1 antenna prob: 0.0003 time: 0
ID: 13: Type: LEAF X13: Loss of RA 1 antenna coax... prob: 0.0 time: 0
ID: 14: Type: LEAF X14: Loss of RA 2 antenna prob: 0.0003 time: 0
ID: 15: Type: LEAF X15: Loss of RA 2 antenna coax... prob: 0.0 time: 0
ID: 2: Type: LEAF X2:Loss of RDIU to IDU network prob: 0.0 time: 0
ID: 3: Type: LEAF X3: Loss of 429 buses from RA ... prob: 0.0 time: 0
ID: 104: Type: OR M4: Loss of RDIU function prob: 0.0001 time: 0.0
ID: 9: Type: LEAF X9: Loss of RA prob: 0.0003 time: 0
ID: 10: Type: LEAF X10: Loss of AC ESS bus bar prob: 0.0 time: 0
ID: 11: Type: LEAF X11: Loss of RA power cord prob: 0.0 time: 0
ID: 108: Type: OR M8: Loss of RA 1 antenna RF si... prob: 0.0003 time: 0.0
ID: 109: Type: OR M9: Loss of RA 2 antenna RF si... prob: 0.0003 time: 0.0
ID: 1: Type: LEAF X1: Loss of IDU display functi... prob: 0.0003 time: 0
ID: 102: Type: OR M2: Loss of data transmission ... prob: 0.0001 time: 0.0
ID: 103: Type: OR M3: Loss of RA output informat... prob: 0.001 time: 0.0
ID: 101: Type: OR M1: Loss of one radio altitude... prob: 0.0014 time: 0.0
prob: 1.0 time: 0.0

**FIG 5. DAG incl. initial failure probabilities based on [17]**

### 4.2. Fault tree and root cause analysis

The next example considered here is based on the use case described in [17], which, in contrast to the previous examples, represents a pure fault tree using a priori failure probabilities that are defined for all leaf nodes. The scenario models failures in a generic radio altimeter (RA) system, with the top-level event being the loss of one radio altitude display, represented by node $X_{101}$. The system is modeled using a fault tree and subsequently visualized as a directed graph to allow detailed probabilistic reasoning, including identification of failure paths and critical components. A graphical representation of the fault tree is given in Figure 5. All node identifiers, basic event names, and prior probabilities used in this example are taken directly from [17]. For completeness, the full data for all nodes are listed in Appendix A.

Each box in the graph corresponds to a leaf node, for which an initial failure probability is specified. These leaf events represent elementary hardware or system failures. Logical gates (OR and AND) are represented by diamonds and ellipses, and they aggregate the probabilities from their children nodes according to standard fault tree semantics.

To analyze potential failure scenarios, we consider a fault at node $X_{101}$ and trace the backward propagation of failure probabilities. This allows for root cause analysis, helping to identify the most likely sources of failure that could result in the top-level event. The experiment and the results are visualzed in Figure 6. The node $X_{101}$ is shown with a solid red border, indicating that this is the root event of interest. As highlighted in dashed red boxes, the most likely root causes (i.e., leaf events with the highest contributions to the failure of node $X_{101}$) are:

- $X_1$: Loss of IDU display functionality,
- $X_9$: Loss of RA,
- $X_{12}$: Loss of RA 1 antenna,
- $X_{14}$: Loss of RA 2 antenna.

These nodes are part of different failure propagation paths that converge at node $X_{101}$. Their relatively high prior probabilities and structural positions within the fault tree make them key contributors to the risk of a radio altitude display failure.

```
name: "gao_radio_altimeter"
results_path: "./results/"
bayesian_method: "exact"
arcs: "./data/Edges_Gao_RadioAltimeter.csv"
vertices: "./data/Nodes_Gao_RadioAltimeter.csv"
scenarios:
  "TS1":
    condition:
      "or":
        - "and": [ 101 ]
```

**Listing 3. Definition of scenario for the Radio Altimeter Configuration.**

## 5. COMMENTS ON THE METHOD AND RESTRICTIONS

In this section, we highlight the main restrictions and challenges of the proposed framework in its current form, with a focus on its applicability in practice. These limitations concern the following aspects:

1) **Availability of information:** For the framework to operate as intended, input data must be provided. This includes the identification of hazards and related unsafe control actions, as well as the derivation or manual specification of corresponding threat scenarios. Furthermore, the threat analysis requires attack and fault graphs with associated transition probabilities and transition times. While probabilities can often be estimated from vulnerability databases or expert knowledge, obtaining realistic transition times is considerably more difficult. Especially for new designs, this data can have a large error margin. If this information cannot be provided with sufficient accuracy, the overall usefulness of the framework is reduced.

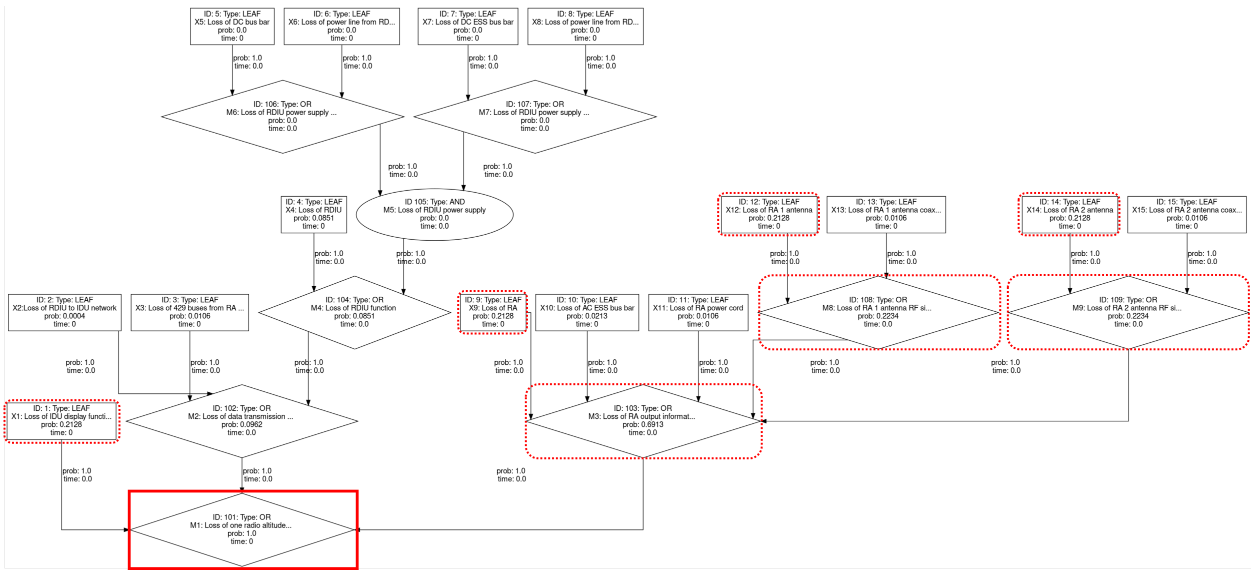
ID: 5: Type: LEAF X5: Loss of DC bus bar prob: 0.0 time: 0
ID: 6: Type: LEAF X6: Loss of power line from RD... prob: 0.0 time: 0
ID: 7: Type: LEAF X7: Loss of DC ESS bus bar prob: 0.0 time: 0
ID: 8: Type: LEAF X8: Loss of power line from RD... prob: 0.0 time: 0
ID: 106: Type: OR M6: Loss of RDIU power supply ... prob: 0.0 time: 0.0
ID: 107: Type: OR M7: Loss of RDIU power supply ... prob: 0.0 time: 0.0
ID: 4: Type: LEAF X4: Loss of RDIU prob: 0.0851 time: 0
ID 105: Type: AND M5: Loss of RDIU power supply prob: 0.0 time: 0.0
ID: 12: Type: LEAF X12: Loss of RA 1 antenna prob: 0.2128 time: 0
ID: 13: Type: LEAF X13: Loss of RA 1 antenna coax... prob: 0.0106 time: 0
ID: 14: Type: LEAF X14: Loss of RA 2 antenna prob: 0.2128 time: 0
ID: 15: Type: LEAF X15: Loss of RA 2 antenna coax... prob: 0.0106 time: 0
ID: 2: Type: LEAF X2:Loss of RDIU to IDU network prob: 0.0004 time: 0
ID: 3: Type: LEAF X3: Loss of 429 buses from RA ... prob: 0.0106 time: 0
ID: 104: Type: OR M4: Loss of RDIU function prob: 0.0851 time: 0.0
ID: 9: Type: LEAF X9: Loss of RA prob: 0.2128 time: 0
ID: 10: Type: LEAF X10: Loss of AC ESS bus bar prob: 0.0213 time: 0
ID: 11: Type: LEAF X11: Loss of RA power cord prob: 0.0106 time: 0
ID: 108: Type: OR M8: Loss of RA 1 antenna RF si... prob: 0.2234 time: 0.0
ID: 109: Type: OR M9: Loss of RA 2 antenna RF si... prob: 0.2234 time: 0.0
ID: 1: Type: LEAF X1: Loss of IDU display functi... prob: 0.2128 time: 0
ID: 102: Type: OR M2: Loss of data transmission ... prob: 0.0962 time: 0.0
ID: 103: Type: OR M3: Loss of RA output informat... prob: 0.6913 time: 0.0
ID: 101: Type: OR M1: Loss of one radio altitude... prob: 1.0 time: 0
prob: 1.0 time: 0.0

**FIG 6. Root cause analysis of a failure scenario in a radio altimeter system, adapted from [17]. The fault tree identifies possible failure propagation paths leading to the top-level event $X_{101}$ (Loss of one radio altitude display), shown with a red border. Boxes represent basic failure events (leaf nodes) with predefined probabilities. Diamonds and ellipses represent logical gates (OR resp. AND). Red dashed boxes indicate the most likely root causes of system failure.**

2) **Technical risks related to the inference algorithm:** We performed numerical studies which demonstrated that the framework can handle a significant number of nodes and edges, particularly when distributed across multiple (sub)graphs, e.g. $n_{\text{graphs}} = 50$, where each graph consists of $n_{\text{nodes}} = 80$ and $n_{\text{edges}} = 120$. However, some system designs may require highly complex graphs with many inter-dependencies within a single model. In such cases, the current inference approach may become inefficient and alternative inference algorithms such as junction-tree–based message passing may be necessary.
3) **Methodological restrictions:** The current implementation requires user-defined thresholds to classify system states into one of three categories: *safe*, *at medium risk*, and *at high risk*. While this provides a clear and intuitive classification, it also introduces subjectivity and may oversimplify complex risk landscapes. More nuanced classification schemes, adaptive thresholds, or continuous risk metrics could improve the fidelity of the analysis.
4) **Integration with MBSE tools:** The framework is currently implemented as a stand-alone tool that relies on external data inputs. This results in additional efforts, e.g. the data has to be exported and preprocessed, and might result in loosing the link with design processes over time. For now, the integration of the framework into existing MBSE toolchains remains a challenge for future work.
5) **Future extensions of the framework:** In addition to the previous point, several other avenues for further development exist. These include handling attack graphs and fault graphs as separate but interacting models, supporting parallel processing to improve performance on large-scale scenarios, and incorporating additional failure modes beyond those currently implemented. Such extensions would enhance the scalability and applicability of the framework in diverse safety-critical domains.

## 6. CONCLUSION

This paper presented a framework for the quantitative analysis of safety-critical threat scenarios. Building on concepts such as attack graphs and fault trees, the framework leverages Bayesian networks to jointly reason about probabilities, timing, and evolving system safety states. Unlike traditional static approaches, it enables dynamic updates that reflect real-time evidence, such as compromised nodes, component failures, or newly disclosed vulnerabilities.

The framework provides three main capabilities:

- systematic modeling and analysis of (safety-critical) threat scenarios involving one or more attack graphs and/or combined fault–attack models represented as directed acyclic graphs (DAGs),
- computation of probabilities and times required for the realization of such safety-critical threat scenarios, and
- continuous integration of new evidence through dynamic updates, ensuring that risk assessments remain accurate and actionable under changing conditions.

Through illustrative examples, we demonstrated how the framework can quantify the likelihood and progression of threat scenarios, thereby supporting the identification of critical vulnerabilities and informing the design of effective countermeasures. This contributes to bridging the gap between abstract risk assessments and operational decision-making in safety-critical environments.

Beyond analysis, the framework enables design iteration: engineers can pose “what-if” questions, quantify risks for different configurations (additional security measures, redundancy, segmentation, checks), and select designs that most effectively improve safety states relative to thresholds.
Future work will focus on extending the framework’s scalability to larger and more complex systems. The field of activity considered within this article is the support of design processes such as MBSE. However, the developed technology could, in principle, also be integrated with existing monitoring infrastructures such as intrusion detection and health management systems. These extensions will further enhance its applicability for real-world safety–security co-analysis.

**Contact address:**

mischa.jahn@airbus.com

## References

[1] A Wayne Wymore. *Model-based systems engineering*. CRC press, 2018.

[2] Matthew Hause et al. The sysml modelling language. In *Fifteenth European systems engineering conference*, volume 9, pages 1–12, 2006.

[3] Object Management Group (OMG) Standards Development Organization. Risk analysis and assessment modeling language (raaml). This version is made available for informational purposes. The formal version is the final approved specification and is the version that should be followed for compliance with this specification.

[4] Takuto Ishimatsu, Nancy G Leveson, John Thomas, Masa Katahira, Yuko Miyamoto, and Haruka Nakao. Modeling and hazard analysis using stpa. 2010.

[5] Nancy G Leveson. *Engineering a safer world: Systems thinking applied to safety*. The MIT Press, 2016.

[6] Luca Maria Castiglione, Philipp Stassen, Cora Perner, Daniel Patrick Pereira, Gustavo de Carvalho Bertoli, and Emil C Lupu. Don’t panic! analysing the impact of attacks on the safety of flight management systems. In *2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC)*, pages 1–10. IEEE, 2023.

[7] Cody Fleming. Introduction to stpa-sec. *Systems Engineering for the Digital Age: Practitioner Perspectives*, pages 489–505, 2023.

[8] Enno Ruijters and Mariëlle Stoelinga. Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer science review*, 15:29–62, 2015.

[9] Paul Ammann, Duminda Wijesekera, and Saket Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 217–224, 2002.

[10] Ronald Ross. Guide for conducting risk assessments, 2012-09-17 2012.

[11] Irad Ben-Gal. *Bayesian Networks*. John Wiley & Sons, Ltd. ISBN: 9780470061572. 2008.

[12] Luis Muñoz-González, Daniele Sgandurra, Martín Barrère, and Emil C Lupu. Exact inference techniques for the analysis of bayesian attack graphs. *IEEE Transactions on Dependable and Secure Computing*, 16(2):231–244, 2017.

[13] Yu Liu and Hong Man. Network vulnerability assessment using bayesian networks. In *Data mining, intrusion detection, information assurance, and data networks security 2005*, volume 5812, pages 61–71. SPIE, 2005.

[14] Marcel Frigault and Lingyu Wang. Measuring network security using bayesian network-based attack graphs. In *2008 32nd Annual IEEE International Computer Software and Applications Conference*, pages 698–703. IEEE, 2008.

[15] Lu Chen, Tao Zhang, Yuanyuan Ma, Yong Li, Chen Wang, Chuan He, Zhuo Lv, and Nuannuan Li. A bayesian-attack-graph-based security assessment method for power systems. *Electronics*, 13(13):2628, 2024.

[16] Xinming Ou, Sudhakar Govindavajhala, Andrew W Appel, et al. Mulval: A logic-based network security analyzer. In *USENIX security symposium*, volume 8, pages 113–128. Baltimore, MD, 2005.

[17] Junru Gao, Bin Yao, and Tianfei Shen. Application of bayesian network based on fault tree in fault diagnosis of radio altimeter system. In *Journal of Physics: Conference Series*, volume 2551, page 012014. IOP Publishing, 2023.

## Appendix

### A. RADIO ALTIMETER DISPLAY FAILURE DATA

This section contains the basic events, identifiers, and failure probabilities used in the radio altimeter fault-tree example presented in Section 4.2. All values are taken directly from [17] without modification. Table 1 lists the basic events and their failure rates (scaled by $10^{-4}$), while Table 2 reproduces the event sequence with prior and posterior probabilities as reported in [17]. Event labels (M1, X1, ...) correspond to Figure 5; node identifiers $X_i$ used in the main text refer to the same events.

| Numbers | Basic event names | Failure rates |
|---|---|---|
| M1 | Loss of one radio altitude display | $14.1 \cdot 10^{-4}$ |
| M2 | Loss of data transmission paths from RA to IDU | $1.36 \cdot 10^{-4}$ |
| M3 | Loss of RA output information | $9.75 \cdot 10^{-4}$ |
| M4 | Loss of RDIU function | $1.2 \cdot 10^{-4}$ |
| M5 | Loss of RDIU power supply | $2.02e - 5 \cdot 10^{-4}$ |
| M6 | Loss of RDIU power supply 1 | $0.45 \cdot 10^{-4}$ |
| M7 | Loss of RDIU power supply 2 | $0.45 \cdot 10^{-4}$ |
| M8 | Loss of RA-1 antenna RF signal | $3.15 \cdot 10^{-4}$ |
| M9 | Loss of RA-2 antenna RF signal | $3.15 \cdot 10^{-4}$ |
| X1 | Loss of IDU display function | $3 \cdot 10^{-4}$ |
| X2 | Loss of RDIU to IDU network | $0.006 \cdot 10^{-4}$ |
| X3 | Loss of 429 buses from RA to RDIU | $0.15 \cdot 10^{-4}$ |
| X4 | Loss of RDIU | $1.2 \cdot 10^{-4}$ |
| X5 | Loss of DC bus bar | $0.3 \cdot 10^{-4}$ |
| X6 | Loss of power line from RDIU to DC bus bar | $0.15 \cdot 10^{-4}$ |
| X7 | Loss of DC ESS bus bar | $0.3 \cdot 10^{-4}$ |
| X8 | Loss of power line from RDIU to DC ESS bus bar | $0.15 \cdot 10^{-4}$ |
| X9 | Loss of RA | $3 \cdot 10^{-4}$ |
| X10 | Loss of AC ESS bus bar | $0.3 \cdot 10^{-4}$ |
| X11 | Loss of RA power cord | $0.15 \cdot 10^{-4}$ |
| X12 | Loss of RA-1 antenna | $3 \cdot 10^{-4}$ |
| X13 | Loss of RA-1 antenna coaxial cable | $0.15 \cdot 10^{-4}$ |
| X14 | Loss of RA-2 antenna | $3 \cdot 10^{-4}$ |
| X15 | Loss of RA-2 antenna coaxial cable | $0.15 \cdot 10^{-4}$ |

**TAB 1. Fault event numbers and probability values adapted from [17].**

| Basic events | Prior prob. | Posterior prob. |
|---|---|---|
| **Loss of RDIU function** | | |
| Loss of DC bus bar | $0.3 \cdot 10^{-4}$ | $0.0031 \cdot 10^{-2}$ |
| Loss of power line from RDIU to DC bus bar | $0.15 \cdot 10^{-4}$ | $0.0015 \cdot 10^{-2}$ |
| Loss of DC ESS bus bar | $0.3 \cdot 10^{-4}$ | $0.0031 \cdot 10^{-2}$ |
| Loss of power line from RDIU to DC ESS bus bar | $0.15 \cdot 10^{-4}$ | $0.0015 \cdot 10^{-2}$ |
| Loss of RDIU | $1.2 \cdot 10^{-4}$ | $8.512 \cdot 10^{-2}$ |
| Loss of RDIU to IDU network | $0.006 \cdot 10^{-4}$ | $0.04256 \cdot 10^{-2}$ |
| Loss of 429 buses from RA to RDIU | $0.15 \cdot 10^{-4}$ | $1.064 \cdot 10^{-2}$ |
| **Loss of RA output information** | | |
| Loss of RA-1 antenna | $3 \cdot 10^{-4}$ | $21.28 \cdot 10^{-2}$ |
| Loss of RA-1 antenna coaxial cable | $0.15 \cdot 10^{-4}$ | $1.064 \cdot 10^{-2}$ |
| Loss of RA-2 antenna | $3 \cdot 10^{-4}$ | $21.28 \cdot 10^{-2}$ |
| Loss of RA-2 antenna coaxial cable | $0.15 \cdot 10^{-4}$ | $1.064 \cdot 10^{-2}$ |
| Loss of RA | $3 \cdot 10^{-4}$ | $21.28 \cdot 10^{-2}$ |
| Loss of AC ESS bus bar | $0.3 \cdot 10^{-4}$ | $2.128 \cdot 10^{-2}$ |
| Loss of RA power cord | $0.15 \cdot 10^{-4}$ | $1.064 \cdot 10^{-2}$ |
| **Other events** | | |
| Loss of IDU display function | $3 \cdot 10^{-4}$ | $21.28 \cdot 10^{-2}$ |

**TAB 2. Event sequence for loss of the radio altitude display adapted from [17].**