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
 Published by Research Publishing, Singapore.
doi: 10.3850/978-981-94-3281-3_ESREL-SRA-E2025-P1842-cd

Fault Tree Construction: A Survey of Knowledge-based, Model-Based and Data-Driven Approaches

Stanley Suan You Lim and Eric Gascard
Univ. Grenoble Alpes, CNRS, Grenoble INP ⋆, G-SCOP, 38000 Grenoble, France
⋆ Institute of Engineering Univ. Grenoble Alpes
E-mail: stanley-suan-you.lim@grenoble-inp.fr

Fault tree (FT) analysis is a widely used technique in system reliability engineering, providing a structured method for identifying the root causes of system failures. By visually representing failure pathways, FTs help engineers assess potential risks and implement preventive maintenance strategies. However, the construction of FTs remains a significant bottleneck, especially for large and complex systems due to the intricacies involved in manually modeling failure modes and capturing system behavior which increase the risk of incomplete failure analysis of the system. This paper proposes a survey of automatic construction methods of FTs. We classify related works into three well-known categories: knowledge-based, model-based, and data-driven. Our survey highlights the formalisms used to model the system under analysis, the algorithms used to build the ft, the scope of their application, and their limitations.

Keywords: Fault Tree Construction, Reliability Analysis, Model-based Safety Assessment (MBSA)

1. Introduction

System reliability is critical for preventing unexpected breakdowns that can lead to significant economic losses, including disrupted production and diminished client trust. Fault Tree (FT) analysis is a widely used method for system reliability assessment. An FT is a directed acyclic graph that illustrates the logical relationships between failure events and their contribution to a defined critical system failure, known as the top event (TE). It enables both qualitative analysis of failure propagation and quantitative analysis of the TE's probability. Despite these advantages, building an FT often requires substantial domain expertise, time, and effort, especially for complex systems. This challenge has sparked considerable research interest in automating its construction, which would greatly enhance the usability and scalability of FT analysis.

The scope of this paper is to provide a concise overview of various FT construction methods. For a comprehensive review of FT analysis frameworks and tools, readers may refer to (Zhu et al., 2023) and Baklouti et al. (2017). The most recent review paper on FT construction methods is Berres and Schumann (2016), covering methods from 1989 to 2015. Our paper provides an update on recent advancements and addresses the evolving trends and challenges in FT construction. The methods are categorized into three main approaches: knowledge-based, model-based, and data-driven. Knowledge-based and model-based approaches rely on expert knowledge of the system described by modeling languages or graphs and converting them into FTs. These methods are effective when domain knowledge is available and enables consistency across fault analyses. They were particularly popular in the latter half of the 1900s in the form of expert systems till the 2000s model-based approaches with the rise of model-based system engineering (MBSE) and safety assessment (MBSA) approaches to manage the growing complexity of systems. Meanwhile, data-driven approaches have seen rapid growth since the 2010s, coinciding with advancements in data science and the increasing availability of failure datasets. These methods construct FTs using patterns uncovered from extensive operational data, offering a scalable and adaptable solution for modern systems.

Our literature review summarizes key insights from previous studies, it is structured as follows: it begins in Section 2 with an overview of our research methodology. Section 3 presents a review of knowledge-based approaches, while Section 4 concerns model-based approaches, and Section 5 is dedicated to data-driven approaches. Then, the paper concludes in Section 6 with a discussion, highlighting the main findings and suggesting areas for future research.

2. Research methodology

The literature search used ScienceDirect, Web of Science, IEEE Xplore, and Google Scholar with these keywords: "Fault Tree" combined with "construction", "generation", "synthesis", "induction", "inference", and "derivation". Papers were initially screened by title and keywords, followed by an abstract review if relevant. Those passing this stage were saved for further review, including the introduction and conclusion. Papers deemed relevant were also checked for forward searches, and their references were used for backward searches to capture any missed-out papers.

3. Knowledge-based approaches

The first category, knowledge-based approaches, relies on formalizing domain expertise to represent the relationships between system components and their failure modes. From this knowledge, algorithms for FT construction or rule-based methods for FT deduction are elaborated.

3.1. *Structured tables and diagrams*

Typically, rule-based languages in the form of tables or graphical system representations are used to encode the system's logical relationships. To construct FTs, trace-back algorithms are commonly employed to trace the defined TE back to the system boundaries, identifying the related components along the way. This method is used to construct FT from Salem et al. (1977); Wang and Liu (1993)'s decision tables, De Vries (1990)'s electrical circuits diagram, and Majdara and Wakabayashi (2009)'s function and state transition tables.

Wang and Liu (1993) refined Salem's method by minimizing decision tables, reclassifying event types, and using a virtual transfer component to reduce redundancies. The Automatic Fault Tree Construction Code (AFTC) is introduced in Han et al. (1989) to synthesize FT using decision tables, flow diagrams, Common Cause Failure models, and super component models to simplify repetitive components and subsystems.

Roth et al. (2015) captures failure propagation in a failure network derived from the multiple domain matrix, and FTs are extracted through structure analysis after identifying TEs using hierarchical function decomposition. Camarda et al. (1978) converts minimal path sets from Reliability Block Diagrams (RBD) into FTs. The TREE-EXPERT (Xie et al., 1993) uses a knowledge tree to map logical relationships and overcome limitations in Probabilistic Safety Analysis.

A directed graph (digraph) consists of nodes and edges used to represent fault propagation within a system. It models dependency paths and fault propagation, offering flexibility by capturing diverse behaviors. Lapp and Powers (1977) created an algorithm that constructs FTs from digraphs, capturing sequential behaviors with time-dependent edges, although it primarily models static fault modes. Nevertheless, this limitation is addressed by the Reliability Imbedded Design Language (RIDL) (Vemuri et al., 1999).

Control loops are mechanisms within control systems designed to continuously monitor and adjust system behavior through sensors, controllers, and actuators. These loops prevent process deviations but are challenging to model in FT formalism as they must be identified before constructing the FT and require consistency checks to ensure mutually exclusive events do not occur simultaneously. Taylor (1982) improved Fussell (1973)'s algorithm by adding support for loops, accepting graphical inputs, and using a library of minitrees. Though automated, the method still requires manual pruning and misses some hazards. In addressing process plant complexities like control loops and bypasses, Kelly and Lees (1986) developed FLTGEN which generates FTs from unit-based minitrees.

Causal trees with branches as fault paths and control/non-control loops as nodes are mapped into FTs in Bossche (1991). Wang et al. (2002) proposed an algorithm using mini cause-and-effect trees to model control loops between system components in the chemical process industry and construct FT. The algorithm works directly from a process block diagram but requires a TE library and user input to initiate events if they are not found in the library.

Digraph-based approaches are very suitable for modeling process systems and control loops, but modeling discrete quantities can be inconvenient, especially when some discrete parameters have wide ranges of variation, which could lead to very large digraphs with lots of directed connections among the connections (Majdara and Wakabayashi, 2009). Also, digraphs are limited to model static fault modes (Vemuri et al., 1999).

3.2. *Logical reasoning-based from knowledge*

The tool KB3 (Renault et al., 1999) enables users to describe systems using a rule-based language (FIGARO) or a GUI. The tool automatically generates FTs using a backward chaining method. By leveraging shared component libraries created using FIGARO, KB3 ensures consistency across FTs in different reliability studies. Elliott (1994) introduced the Reliability Analysis Expert System, generating FTs from functional block diagrams and simplifying them via Boolean algebra and modularization. State machines model system states and transitions, suitable for systems with sequential or conditional behaviors. Liggesmeyer and Rothfelder (1998) proposed a formal risk analysis where FTs are generated from system behavior described as finite-state machines. However, this approach is challenged by state explosion as system complexity grows.

3.3. *FT generation from model checkers*

Model checkers represent an alternative approach to backtracing a system model. For example, the NuSMV (Bozzano and Villafiorita, 2007) verifies temporal logic formulas that define the reachability of a faulty mode of a system model. The generated counterexample then serves as the basis for FT construction. Another model checker tool is Eucalypt (Rae and Lindsay, 2004), which derives FTs from process algebra.

4. Model-based approaches

The second category, model-based approaches, uses a representation of the structure and behaviour of the system studied in a specialized modeling language. Based on the model of the system defined in this particular modeling language, model-based approaches propose specialized methods to build FTs from undesired behaviour.

4.1. *UML and SysML*

The UML and SysML are former system description languages widely used in MBSE due to their flexibility and compatibility with various modeling tools. UML represents structural and behavioral aspects through various diagrams. It allows customization with constraints and stereotypes, making it adaptable for modeling failure-related properties in FT construction. In Pai and Dugan (2002), UML is used to model system dependencies and component failure attributes to construct Dynamic FT (DFT). SysML extends UML with additional diagrams for modeling hardware and software in system engineering. Xiang et al. (2011) used the Maude language to generate FTs from SysML diagrams. Yet their method is mainly suited for IT systems and in later design stages.

Mhenni et al. (2014) proposed a semi-automated approach to generate FMEA and FTs using SysML, where functional and internal block diagrams inform preliminary analyses, later refined by experts.

Baklouti et al. (2019) generates DFTs using SysML structural and class diagrams with redundancy profiles. A depth-first search traces failure propagation, creating sub-trees based on redundancy types and switch requirements, which are then concatenated to form the DFT.

A component FT (CFT) focuses on the reliability of smaller, reusable components that integrate into larger systems. The tool ALFRED (Höfig et al., 2015) extends CFTs across system layers using SysML safety models as input to address the lack of support for inter-layer dependencies between software and hardware in CFTs. This method enhances modularity but requires that all dependencies be defined for effective system-level analysis.

4.2. *Mode automata and Altarica*

Mode automata are high-level languages used to model complex systems. They decompose systems hierarchically into subsystems with finite states (modes) where only one state is active at a time. Each state defines outputs based on input flow values, and it can switch states in response to events, providing various model perspectives to emphasize different aspects.

AltaRica is a formal specification language based on mode automata designed to represent failure logic according to system structures and functions. It enables the generation of FTs that closely reflect real system designs. Rauzy (2002) derived Boolean equations from AltaRica for FT construction by deriving the system's behavior from state machines and computing a graph of all reachable states. Then, the logical expressions needed to reach each state are identified, and consistent expressions are retained to generate the FT.

Yakymets et al. (2013) proposed the Safety Modeling Framework for FT analysis, where

SysML diagrams are converted into AltaRica language, which enables minimal cut sets (MCSs) extraction using the ARC tool. The MCSs are then structured into FTs.

Li and Li (2014) refined AltaRica's fault modeling, which was designed by deriving failure logic from system nodes and component nodes destined for FT construction.

4.3. *MATLAB-Simulink*

MATLAB-Simulink is widely used for engineering system modeling and simulation due to its robust interface. Papadopoulos and Maruhn (2001) proposed a continuous safety assessment approach for programmable systems. Using Simulink to model the system, Hazard and Operability Studies (HAZOP) are conducted to identify hazards and an FT is generated by selecting subsystems that contribute to the TE. The downside of using HAZOP to uncover system dependencies and cause-effect relationships is that it demands significant modeling effort and expert analysis (Roth et al., 2015).

Similarly, Latif-Shabgahi and Tajarrod (2009) developed a method to construct FTs from MATLAB-Simulink block diagrams that include failure models. Their approach manually extends the model structure, enabling the FT construction code to generate either DFT or SFT. Components are connected based on their types to determine gate assignments, leveraging Simulink's toolbox and avoiding digraphs, transition tables, or knowledge-based rules. Nevertheless, it does not address cycles within models (Berres and Schumann, 2016). However, each subsystem has a limitation on the number of components it can contain in a Simulink model, limiting its usage on large and complex systems. Additionally, if the same component performs multiple functions, it must be treated as separate, virtually distinct components.

4.4. *AADL*

Architecture Analysis and Design Language (AADL) supports system architecture modeling for real-time systems, including component descriptions and failure data through its Error Model Annex. Unlike behavioral models such as AltaRica and Simulink with HiP-HOPS, which lack inherent support for architecture representation, AADL inherently integrates system components through its core language. Nevertheless, AADL is designed for embedded systems, making it less suitable for complex systems engineering, and it has limitations that restrict certain failure mode specifications.

Joshi et al. (2007) converted AADL models to SFT by extracting the error model, organizing it into graph nodes, and using a recursive algorithm to generate and optimize an intermediate FT for commercial FTA tool compatibility.

Dehlinger and Dugan (2008) proposed converting AADL models enriched with an Error Model Annex to DFTs, incorporating spares, functional dependencies, and failure sequences through dynamic event/failure trees to capture dynamic failure propagation.

4.5. *GO model*

The GO model is a success-based system probability analysis technique suited for modeling systems exhibiting multi-state and time-sequential characteristics. It is constructed directly from the system principle diagram, flowchart, or engineering diagram, thus closely resembling the system principle diagram. Zhang et al. (2015) defined mapping rules to convert the GO model into FT.

5. Data-driven approaches

The third category, data-driven approaches, is relieved of the constraint of having an expert of the system or a detailed model of the system. This approach only needs data sets relying on the nominal and failure-mode behaviour of the system. From these data sets, machine learning (ML) and/or heuristic algorithms provide FT inference.

5.1. *Decision tree and classification algorithms*

The Binary Decision Tree (BDT) is frequently employed to generate a tree formalism when Boolean data for FT construction is provided. Nolan et al. (1994) proposed the induction of FTs algorithm based on the ID3 decision tree induction algorithm. Their algorithm learns the FT structure by recording system states and undesired events, requiring no detailed system analysis.

Juričić and Žnidaršič (1996) integrated process models with ML to generate BDTs for continuous industrial processes. Using interval-based component models and physical laws, the process under fault conditions is simulated with a Monte Carlo simulation to generate training examples.

This hybrid method to model a system is proposed to merge knowledge about the process operation from various sources. A multi-strategy ML approach (ID3, CN2, KNN, Bayesian classification) was applied to identify thresholds and create decision trees linking symptoms to faults. While the approach was effective for simple cases, its applicability to dynamic systems and more complex cases remains uncertain.

Berikov (2004) proposed synthesizing FTs from stationary multivariate time series by using decision trees to forecast and identify events. Each decision path forms FT nodes, enabling a simple, generalizable approach that handles both qualitative and quantitative data, even with limited or low-quality datasets. However, expert input is needed to determine non-terminal nodes.

Contrasting to the correlation that BDTs capture, Causal Decision Trees (CDTs) describe causal relationships needed for cause-effect analysis. Nauta et al. (2018) introduced the LIFT algorithm inspired by the CDT to construct SFTs. Assuming that the TE, intermediate events (IEs), and basic events (BEs) are present in the dataset, the LIFT algorithm creates an SFT that contains solely the TE and then iteratively increases the depth of the FT by generating all combinations of IEs and BEs. The choice of gates is decided using the standard Mantel-Haenszel Partial Association test (PAMH). The advantage of this approach is that causality between the BEs is found, but an exhaustive test has to be carried out, resulting in exponential time complexity with increasing BEs, and this approach is sensitive to noise (Linard et al., 2019).

Verkuil et al. (2022) addressed the limitations of Boolean-based data-driven FT methods for continuous sensor data. They applied the C4.5 algorithm to discretize real-time data into Boolean values tailored to each TE and used the LIFT algorithm to construct the FT. The authors replaced the PAMH coefficient with the Phi coefficient, which adapts better to varying dataset dimensions at the cost of identifying correlations between failures instead of causality.

Niloofar and Lazarova-Molnar (2023) combines their approach of Data-Driven Fault Tree Analysis (DDFTA) (Lazarova-Molnar et al., 2020a) with a Naive Bayesian classification model (DDFTAnb) to predict system behaviors from combinations of failures unknown to domain experts. Their approach takes a multinomial time series and converts them into untimed boolean truth data. The unobserved TE states in the truth data are classified by the Naive Bayesian classification model fitted on data with known TE states, generating complete truth data for repairable FT extraction. The caveat is that the Naive Bayesian classification model assumes BEs are independent.

5.2. *Genetic algorithms*

Inspired by biological evolution, Linard et al. (2019) developed an evolutionary algorithm (EA) that iteratively evolves static FT structures using randomized genetic operators. This method generates FTs with or without expert knowledge, adding sub-trees to an existing structure if a partial FT is provided. The objective for each generation is to identify FTs that improve the accuracy of the TE value predicted from the given BE values. However, focusing solely on optimizing TE value prediction without considering the FT structure often results in inconsistencies in FT construction and long computational times. To address these limitations, Jimenez-Roa et al. (2022) proposed Multi-Objective Evolutionary Algorithms (MOEAs) that optimize conflicting objective functions, thereby finding optimal solutions in terms of accuracy and the generated FT structure. To further enhance the computational efficiency of EAs, the authors later introduced an optimization strategy that leverages the modularity and symmetries within the MCSs (Jimenez-Roa et al., 2022). This method is well-suited for systems with significant symmetries, such as truss structures. However, the scalability of the EA approach for large systems remains a significant challenge. In Linard et al. (2019), genetic operators are applied randomly, raising questions about whether alternative strategies could improve the likelihood of finding the global optimum and accelerate convergence. Colta et al. (2024) sought to benchmark genetic operators' influence on FT generation's success in the EA framework. While their results were largely inconclusive, they observed that the performance of genetic operators varied depending on the number of generations.

5.3. *Relationship mining*

Logical analysis of data (LAD) is a method for identifying patterns in datasets through enumeration. Boros et al. (2000) proposed a hybrid

LAD approach combining bottom-up and top-down methods. This approach generates minimal attribute patterns while ensuring each observation is associated with at least one pattern. Redundant patterns are removed using support sets for accurate and non-conflicting observation classifications. The Interpretable Logic Tree Analysis (ILTA) method uses the patterns extracted by LAD to construct one-level FT. The ILTA is later developed into Multi-level ILTA (MILTA) (Waghen and Ouali, 2021) to address ILTA's limitation in handling multiple cause-and-effect sequences. An improved "burn-and-build" algorithm which minimizes the number of patterns and their overlap while maximizing the coverage of same-class observations is also introduced to optimize pattern selection. These patterns are then connected via logic gates and assigned probabilities to form an ILTA model. If the ILTA model includes intermediate causes at the condition layer, a new sub-dataset is created to build another ILTA model based on those conditions, forming a MILTA model.

5.4. *Other methods*

The Siemens Corporate Research team (Mukherjee and Chakraborty, 2007) explored text mining techniques to analyze company knowledge repositories to identify failure events. To understand the meaning of words and their relationships, an online dictionary is used. They also looked at how failure events are described in a structured way, where sub-events leading to equipment failures are shown as indented text. The team then used a similarity detection technique to match these descriptions with the equipment list and categorized the words based on whether they referred to components or failures. New failure relationships discovered are added to update the existing FT.

Linard et al. (2019) proposed constructing SFTs from Bayesian Networks (BNs) using translation rules. BN learning involves structure learning by heuristic search and parameter learning which computes Conditional Probability Tables (CPT) to determine logic gates. Domain knowledge can be incorporated to reduce search space, thus minimizing computation time. The limitation of BNs is that the TE, BEs, and IEs need to be known.

Lazarova-Molnar et al. (2020b)'s algorithm constructs repairable SFTs by converting time-series data into Boolean data and identifying cut sets based on TE occurrences. These cut sets are refined into MCSs using set theory, with events in the same MCS linked by AND operators and all MCSs connected by OR operators. This process generates a Boolean equation representing the FT, which is then simplified using Boolean algebra. Using the same method, multi-state FT is constructed in Lazarova-Molnar et al. (2020a).

Qian et al. (2021) tackled the limitations of FT methods that rely on functional modules or mechanical structures, which often lack adaptability to changing operating frequencies and environments. They proposed a data-driven FT construction method using association analysis of failure data to discover rules and causality between failure modes. To address the sparsity of failure data in reliable systems, failure records are transformed into a hierarchical database, and a multi-dimensional partitioning method is applied. Combined with a failure mode intensity parameter, this partitioned data creates a transaction database for association rule mining, with the resulting rules used to construct the FT.

6. Discussion and Conclusion

FTs are important reliability assessment tools, but their construction remains a significant bottleneck to their broader application. This paper reviewed FT construction methodologies, classifying them into knowledge-based, model-based, and data-driven approaches. Knowledge-based and Model-based approaches depend on domain knowledge, where experts describe the system structure and behaviors in various formalisms. These approaches to FT construction offer the advantage of being implementable during the early stages of system development, provided that the system's representation and relevant domain knowledge are available. The main drawbacks of these methods are that they require significant manual effort and rely heavily on the expertise of domain experts, meaning their completeness and accuracy vary depending on the depth and precision of the expert-provided information.

On the other hand, data-driven approaches represent a more automated way to construct FT. By learning from sensor readings, data-driven FT can account for operating environments and frequencies, providing a realistic representation of system behaviors and even discovering hidden failure behaviors. This includes estimating the actual failure probabilities of BEs and automatically updating the FT as systems evolve or components are re-

placed. However, the primary limitation of these approaches is that they can only be applied after the system is operational and sufficient data have been collected. This delay makes them less suitable for early-stage reliability assessments, and data on failure behaviors are rare in highly reliable systems.

Another key observation is the limited availability of methods for constructing DFTs. Unlike static FTs, DFTs incorporate time-dependent behaviors and dynamic interactions between system components, such as sequence dependencies, functional dependencies, and spares. This capability makes DFTs essential for the reliability assessment of modern systems, especially for cyber-physical systems and real-time control systems. DFTs allow for a more realistic representation of failure scenarios and provide insights into how system behavior evolves, making them a critical tool for analyzing the reliability of next-generation systems. Despite their importance, automated methods for constructing DFTs remain underexplored, presenting a significant research gap in the field.

This review examines the strengths and limitations of various FT construction methods, emphasizing their complementary roles in system reliability and diagnostics. By providing a concise overview, this paper contributes to enhance the understanding of existing approaches and guides practitioners in selecting the most suitable method for their needs.

References

Baklouti, A., N. Nguyen, J.-Y. Choley, F. Mhenni, and A. Mlika (2017). Free and open source fault tree analysis tools survey. In *Annual IEEE International Systems Conference (SysCon)*. IEEE.

Baklouti, A., N. Nguyen, F. Mhenni, J.-Y. Choley, and A. Mlika (2019). Dynamic fault tree generation for safety-critical systems within a systems engineering approach. *IEEE Systems Journal 14*(1).

Berikov, V. (2004). Fault tree construction on the basis of multivariate time series analysis. In *International Symposium on Science and Technology*. IEEE.

Berres, A. and H. Schumann (2016). Automatic generation of fault trees: A survey on methods and approaches. In *European Safety and Reliability Conference (ESREL)*.

Boros, E., P. L. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz, and I. Muchnik (2000). An implementation of logical analysis of data. *IEEE Transactions on knowledge and Data Engineering 12*(2).

Bossche, A. (1991). Computer-aided fault tree synthesis i (system modeling and causal trees). *Reliability Engineering & System Safety 32*(3).

Bozzano, M. and A. Villafiorita (2007). The fsap/nusmv-sa safety analysis platform. *International Journal on Software Tools for Technology Transfer 9*(1), 5–24.

Camarda, P., F. Corsi, and A. Trentadue (1978). An efficient simple algorithm for fault tree automatic synthesis from the reliability graph. *IEEE transactions on Reliability 27*(3).

Colța, B., L. A. Jimenez-Roa, and M. Stoelinga (2024). Understanding the application of multi-objective evolutionary algorithms to the inference of fault tree models. B.S. thesis, University of Twente.

De Vries, R. C. (1990). An automated methodology for generating a fault tree. *IEEE transactions on reliability 39*(1).

Dehlinger, J. and J. B. Dugan (2008). Analyzing dynamic fault trees derived from model-based system architectures. *Nuclear Engineering and Technology 40*(5).

Elliott, M. S. (1994). Computer-assisted fault-tree construction using a knowledge-based approach. *IEEE Transactions on Reliability 43*(1).

Fussell, J. B. (1973). Synthetic tree model: A formal methodology for fault tree construction. Technical report, Aerojet Nuclear Company (United States).

Han, S. H., T. W. Kim, Y. Choi, and K. J. Yoo (1989). Development of a computer code aftc for fault tree construction using decision table method and super component concept. *Reliability Engineering & System Safety 25*(1).

Höfig, K., M. Zeller, and R. Heilmann (2015). Alfred: a methodology to enable component fault trees for layered architectures. In *Euromicro Conference on Software Engineering and Advanced Applications*. IEEE.

Jimenez-Roa, L. A., T. Heskes, T. Tinga, and M. Stoelinga (2022). Automatic inference of fault tree models via multi-objective evolutionary algorithms. *IEEE transactions on dependable and secure computing 20*(4), 3317–3327.

Jimenez-Roa, L. A., M. Volk, and M. Stoelinga (2022). Data-driven inference of fault tree models exploiting symmetry and modularization. In *International Conference on Computer Safety, Reliability, and Security*. Springer.

Joshi, A., S. Vestal, and P. Binns (2007). Automatic generation of static fault trees from aadl models. In *Dependable Systems and Networks Workshop on Architecting Dependable Systems*.

Juričić, Đ. and A. Žnidaršič (1996). Generation of fault trees by means of simplified process models and machine learning. *IFAC Proceedings Volumes 29*(7).

Kelly, B. E. and F. P. Lees (1986). The propagation of faults in process plants: 2. fault tree synthesis. *Reliability Engineering 16*(1).

Lapp, S. A. and G. J. Powers (1977). Computer-aided synthesis of fault-trees. *IEEE Transactions on Reliability 26*(1).

Latif-Shabgahi, G. and F. Tajarrod (2009). A new approach for the construction of fault trees from

system simulink. In *International Conference on Availability, Reliability and Security*. IEEE.

Lazarova-Molnar, S., P. Niloofar, and G. K. Barta (2020a). Automating reliability analysis: Data-driven learning and analysis of multi-state fault trees. In *European Safety and Reliability Conference Conference (ESREL)*.

Lazarova-Molnar, S., P. Niloofar, and G. K. Barta (2020b). Data-driven fault tree modeling for reliability assessment of cyber-physical systems. In *Winter Simulation Conference (WSC)*. IEEE.

Li, S. and X. Li (2014). Study on generation of fault trees from altarica models. *Procedia Engineering 80*.

Liggesmeyer, P. and M. Rothfelder (1998). Improving system reliability with automatic fault tree generation. In *International Symposium on Fault-Tolerant Computing*. IEEE.

Linard, A., D. Bucur, and M. Stoelinga (2019). Fault trees from data: Efficient learning with an evolutionary algorithm. In *International Symposium on Dependable Software Engineering: Theories, Tools, and Applications*. Springer.

Linard, A., M. L. P. Bueno, D. Bucur, and M. Stoelinga (2019). Induction of fault trees through bayesian networks. In *European Safety and Reliability Conference Conference (ESREL)*.

Majdara, A. and T. Wakabayashi (2009). Component-based modeling of systems for automated fault tree generation. *Reliability Engineering & System Safety 94*(6).

Mhenni, F., N. Nguyen, and J.-Y. Choley (2014). Automatic fault tree generation from sysml system models. In *International Conference on Advanced Intelligent Mechatronics*. IEEE.

Mukherjee, S. and A. Chakraborty (2007). Automated fault tree generation: bridging reliability with text mining. In *Annual Reliability and Maintainability Symposium*. IEEE.

Nauta, M., D. Bucur, and M. Stoelinga (2018). Lift: learning fault trees from observational data. In *International Conference QEST*. Springer.

Niloofar, P. and S. Lazarova-Molnar (2023). Data-driven extraction and analysis of repairable fault trees from time series data. *Expert Systems with Applications 215*.

Nolan, P. J., M. G. Madden, and P. Muldoon (1994). Diagnosis using fault trees induced from simulated incipient fault case data. In *International Conference on Intelligent Systems Engineering*. IEEE.

Pai, G. J. and J. B. Dugan (2002). Automatic synthesis of dynamic fault trees from uml system models. In *International Symposium on Software Reliability Engineering*. IEEE.

Papadopoulos, Y. and M. Maruhn (2001). Model-based synthesis of fault trees from matlab-simulink models. In *International Conference on Dependable Systems and Networks*. IEEE.

Qian, K., L. Yu, and S. Gao (2021). Fault tree construction model based on association analysis for railway overhead contact system. *International Journal of Computational Intelligence Systems 14*(1).

Rae, A. and P. Lindsay (2004). A behaviour-based method for fault tree generation. In *International System Safety Conference*.

Rauzy, A. (2002). Mode automata and their compilation into fault trees. *Reliability Engineering & System Safety 78*(1).

Renault, I., M. Pilliere, N. Villatte, and P. Mouttapa (1999). Kb3: computer program for automatic generation of fault trees. In *Reliability and Maintainability Symposium*. IEEE.

Roth, M., M. Wolf, and U. Lindemann (2015). Integrated matrix-based fault tree generation and evaluation. *Procedia Computer Science 44*.

Salem, S. L., G. E. Apostolakis, and D. Okrent (1977). A new methodology for the computer-aided construction of fault trees. *Annals of Nuclear Energy 4*(9-10).

Taylor, J. R. (1982). An algorithm for fault-tree construction. *IEEE Transactions on reliability 31*(2).

Vemuri, K. K., J. B. Dugan, and K. J. Sullivan (1999). Automatic synthesis of fault trees for computer-based systems. *IEEE Transactions on Reliability 48*(4).

Verkuil, B., C. E. Budde, and D. Bucur (2022). Automated fault tree learning from continuous-valued sensor data: a case study on domestic heaters. *International Journal of Prognostics and Health Management 13*(2).

Waghen, K. and M.-S. Ouali (2021). Multi-level interpretable logic tree analysis: A data-driven approach for hierarchical causality analysis. *Expert Systems with Applications 178*.

Wang, J. D. and T. S. Liu (1993). A component behavioural model for automatic fault tree construction. *Reliability Engineering & System Safety 42*(1).

Wang, Y., T. Teague, H. West, and S. Mannan (2002). A new algorithm for computer-aided fault tree synthesis. *Journal of Loss Prevention in the Process Industries 15*(4).

Xiang, J., K. Yanoo, Y. Maeno, and K. Tadano (2011). Automatic synthesis of static fault trees from system models. In *International Conference on Secure Software Integration and Reliability Improvement*. IEEE.

Xie, G., D. Xue, and S. Xi (1993). Tree-expert: a tree-based expert system for fault tree construction. *Reliability Engineering & System Safety 40*(3).

Yakymets, N., H. Jaber, and A. Lanusse (2013). Model-based system engineering for fault tree generation and analysis. In *International Conference on Model-Driven Engineering and Software Development*, Volume 2. SCITEPRESS.

Zhang, Y., Y. Ren, L. Liu, and Z. Wang (2015). A method of fault tree generation based on go model. In *International Conference on Reliability Systems Engineering*. IEEE.

Zhu, C., Y. Jiang, G. Liu, and T. Zhang (2023). Integration frameworks and intelligent research in dynamic fault tree: A comprehensive review and future perspectives. *Quality and Reliability Engineering International 39*(7).