AUTOSAFE 2.0: Improved and Optimized Automation of Fault Tree Analysis

Kamilla Shagazatova
Delta System Solutions
München, Germany
0009-0008-7305-8007

Aigerim Bakytzhanova
Delta System Solutions
München, Germany
0009-0000-0445-8340

Winfried Friedl
Delta System Solutions
München, Germany

Robert Muirhead
Delta System Solutions
München, Germany

Brad Sams
Delta System Solutions
München, Germany

Abstract—Fault Tree Analysis (FTA) is widely used for safety certification of complex cyber-physical systems, yet its manual application is time-consuming and error-prone. This paper presents AUTOSAFE 2.0, a web-based tool for automated fault tree generation from model-based system descriptions, extending the Open Avionics Architecture Model with explicit failure propagation semantics. Key improvements include optimized synthesis using Binary Decision Diagrams, automated propagation-combination generation, integrated model verification, and XML-based interoperability with industrial tools such as Isograph. Work in progress targets extended industrial validation, deeper regulatory alignment, and automated hazard analysis integration.

Keywords—fault tree analysis; safety-critical systems; model-based systems engineering; automation; Binary Decision Diagrams

I. INTRODUCTION

Safety assessment is fundamental to the certification of safety-critical systems in aerospace, automotive, and industrial automation. Fault Tree Analysis (FTA) is one of the most widely applied methods for identifying combinations of component failures that lead to system-level hazards [1]. Despite its maturity, FTA practice remains largely manual: engineers interpret system documentation and expert knowledge to construct trees by hand. As complexity in system architectures grows, this process becomes increasingly time-consuming, error-prone, and difficult to maintain under design change [2].

Model-Based Systems Engineering (MBSE), combined with Model-Based Safety Assessment (MBSA), provides a framework for integrating safety analysis directly into the system development process. While MBSE establishes a single, authoritative system model, MBSA leverages that model to automatically derive, maintain, and validate safety artefacts such as fault trees, improving traceability, consistency, and engineering efficiency throughout the lifecycle [3]. Within this context, AUTOSAFE 2.0 implements an integrated MBSE–MBSA approach by automatically generating fault trees from model-based system descriptions while introducing significant improvements in performance, model verification, and interoperability over its predecessor [4].

II. TOOL OVERVIEW

A. Architecture and Modelling Framework

AUTOSAFE is grounded in the Eclipse Modelling Framework (EMF) ECORE standard and built on the Open Avionics Architecture Model (OAAM), extended with explicit failure propagation semantics [5]. The tool runs entirely in a web browser via the Essential Object Query (EOQ) framework, enabling collaborative, real-time editing without local installations. System models are structured across four layers: (1) Library—hardware types, task types, and failure mode sets; (2) Functions—logical functions and signal exchanges; (3) Hardware—physical architecture; (4) Allocation—function-to-hardware mapping.

B. Fault Tree Generation Workflow

Generation begins with the OAAM-based graphical editor, where engineers model architecture and failure propagation using Boolean expressions or matrix-based formulations. AUTOSAFE performs backtracking analysis over failure propagation paths, constructs Binary Decision Diagrams (BDDs) to represent failure logic, derives minimal cut sets, and translates these into fault tree structures [6]. A Boolean simplification pass (distributive and absorption laws) removes redundancy and improves readability.

C. Key Enhancements in AUTOSAFE 2.0

Performance and scalability. Backend optimizations—intelligent caching, query optimization, and refined memory management—substantially reduce computation time for large models and fault trees.

Automated propagation combinations. Engineers define failure propagation rules at a higher abstraction level in a logical matrix; all valid combinations are generated automatically, eliminating manual enumeration and reducing modelling errors.

Model verification. An integrated check analyses completeness and consistency before Fault Tree generation, detecting missing failure links, undefined parameters, and unresolved allocations with traceable diagnostic feedback.

XML interoperability. Fault trees can be exported and imported in XML, enabling structured exchange with established tools such as Isograph and supporting heterogeneous safety tool chains.

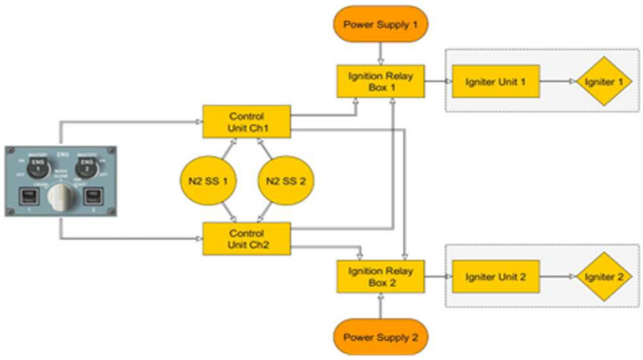
III. ILLUSTRATIVE CASE STUDY

To demonstrate AUTOSAFE 2.0, an engine ignition system is modelled: the system ignites the fuel/air mixture in a combustion chamber under control unit command, involving igniters, speed sensors, control boards, monitoring boards, a fuel metering valve, and associated power supplies. The system is a dual-lane architecture, with a single operating igniter sufficient to provide ignition in the combustion chamber. The analysis target is the failure condition “Total Loss of Ignition.”

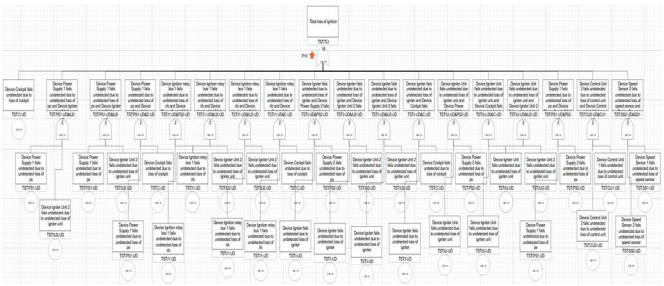
Modelling follows the four OAAM layers. The new propagation-combination feature automatically populates the import columns of the failure matrix, eliminating manual permutation. The model verification mechanism confirms completeness before generation. The resulting fault tree is exported to Isograph via XML and loads without modification, demonstrating end-to-end interoperability.

Library Layer forms the foundation of the system model, as all subsequent modelling steps are based on the data defined within it. Allocations Layer is the interface between the physical Hardware Layer and the logical Functions Layer, allows editing the allocation of tasks from the functions layer to devices from the hardware layer. It shows the devices from the hardware layer as compartments for the tasks from the functions layer. Hardware Layer is used to specify the physical architecture of the system. Architectures can be composed by devices and connections. Function Layer represents the logical view of the system. It shows allocable tasks connected

by signals. Each task and signal from the Functions Layer must be allocated to the hardware from the Hardware Layer.



Ignition system



Ignition System Fault Tree

IV. WORK IN PROGRESS

AUTOSAFE 2.0 represents ongoing work. The following areas are currently under active development and will be addressed in forthcoming iterations:

Extended industrial validation. The tool was demonstrated on a representative ignition system case study; validation on larger, multi-domain industrial systems is in progress to assess scalability and coverage of more complex real-world failure dependencies.

Process improvement for certification compliance. AUTOSAFE 2.0 improves the fault-tree development process by automating the translation of the defined system architecture into its corresponding failure logic. This reduces manual effort and the potential for human error while improving consistency and traceability. The resulting fault tree therefore maintains a direct relationship with the defined architecture and its failure propagation, supporting the safety analysis activities associated with certification.

Quantitative analysis and probability computation. Integration of probabilistic failure data into the BDD-based synthesis to support automatic cut-set probability calculation and top-event probability estimation is planned for the next release.

V. CONCLUSION

AUTOSAFE 2.0 improves on its predecessor in scalability, verification, and interoperability, and demonstrates automated fault tree generation for a representative safety-critical system. Ongoing work targets extended industrial validation, regulatory alignment, and automated hazard analysis integration.

The authors welcome discussion and early feedback from the SAFECOMP community.

References

[1] E. Ruijters and M. Stoelinga, "Fault tree analysis: A survey of the state-of-the-art," Computer Science Review, vol. 15–16, pp. 29–62, 2015.

[2] P. Liggesmeyer and M. Rothfelder, "Improving system reliability with automatic fault tree generation," in Proc. 28th FTCS, Munich, 1998, pp. 90–99.

[3] A. Schamai, "Model-based design verification," Linkoping Univ., Tech. Rep., 2013.

[4] M. Wumaierjiang et al., "Autosafe: Automatic fault tree synthesis for cyber-physical systems," in Proc. ESREL, 2021.

[5] B. Annighoefer, "An open source domain-specific avionics architecture model," SAE AeroTech Americas, 2019.

[6] K. A. Reay and J. D. Andrews, "A fault tree analysis strategy using binary decision diagrams," Reliability Eng. & System Safety, vol. 78, no. 1, pp. 45–56, 2002.