

A UAV Case Study on an MBSE Workflow with Integrated Modular Safety and Reliability Analysis

Bernhard Kaiser, Ansys Germany

Michael Soden, Ansys Germany

Nils Heuermann, Ansys Germany

Key Words: Component Fault Tree Analysis, MBSE, MBSA, SysML, UAV

SUMMARY & CONCLUSIONS

This paper presents a model-based systems engineering (MBSE) workflow in compliance with aerospace safety standards, based on the new SysML v2 language and with particular focus on executable models. We extend the SysML modeling concepts by Component Fault Trees (CFTs) to form an integrated modular Model-Based Safety Analysis (MBSA) approach. We showcase our approach by an Unmanned Aerial Vehicle (search&rescue drone) and discuss how this approach can speed up the process and increase the consistency, modularity, and reuse of the design and safety analysis of the system.

1 INTRODUCTION

Unmanned Aerial Vehicles (UAVs) are fail-operational systems. This means that there is no immediate passive safe state, but operation must be upheld at least for a period of time, even in the presence of failures. Accordingly, safety cannot be judged separately from reliability and availability, and complex redundancy and fallback concepts are required. Modeling such complex architectures and analyzing the behavior under different degradation conditions demands model-based systems engineering (MBSE). The methods to analyze the system for single-point failures and to compute quantitative reliability figures must be tightly integrated with the MBSE approach and existing models. Moreover, it is desirable that these methods are modular to enable agile evolution of the system and to reduce the workload by exploiting the typically highly redundant nature of such systems.

In this article, we present an MBSE workflow based on the new SysML v2 language as a basis for all design and safety/reliability analysis efforts [13]. As a novelty we integrate the modular safety analysis technique Component Fault Trees (CFT) into the “define & use” concept of SysML v2 to describe fault logic of components. The workflow has been adapted from a similar approach we have successfully applied to an automated driving function in the automotive domain [Error! Reference source not found.]. We will go through the main phases of the development and safety assessment cycle, starting with CONOPS definition and requirements, then iterating between design activities such as functional and technical

architecture definition, requirements refinement, and safety activities such as Functional Hazard Assessment (FHA) and (Preliminary) System Safety Assessment. A main focus will be on CFT: we briefly introduce the approach, show how CFT component definitions and usages along with their (failure) ports can be derived from the SysML model, populated with failure modes attached to the component models using FMECA, and finally performing exemplary qualitative and quantitative analysis on them. We will explain how multi-instantiation of the same CFT component definition (e.g. for a rotor unit) saves time during creation, maintenance and variation of the models. We will conclude with a summary of the achievements in terms of effort reduction.

2 PRELIMINARIES

2.1 Model-Based Systems Engineering (MBSE)

Model-based Systems Engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases [Error! Reference source not found.]. Frequently used models for complex technical systems are expressed in the SysML modeling language. Being semi-formal graphical models, they are understandable for engineers and managers from different domains while providing a minimum of consensus among experts on how their semantics is to be interpreted. This latter point will profit from the ongoing introduction of SysML v2, which will have precise semantics. This makes formal support for requirements refinement (see [Error! Reference source not found.], Table 1), static checking of properties and the integration of behavior simulation feasible. SysML v2 will come with a recursive concept of definitions and usages that fits, as we will see, very well the approach of CFTs. Furthermore, SysML v2 will include analyses and trade studies as first-class language concepts, as well as variants.

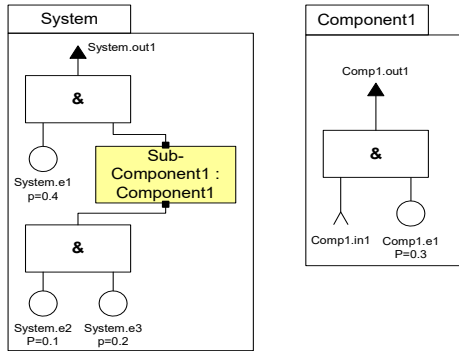
2.2 Model-Based Safety Analysis (MBSA)

Model-Based Safety Analysis (MBSA) is a term used to describe a set of formal and semi-formal techniques that are

centered around models with fault behavior (cp. [Error! Reference source not found.], Appendix N). The common idea is to describe the system structure and nominal behavior as well as the (potential) faulty behavior in the same modeling language instead of creating separated analysis aside the design. In this paper we refer to MBSA specifically by embedding fault tree logic into the SysML v2 language to specify failure propagation logic. That means that the structural and behavioral aspects are described using standard SysML models whereas the fault logic is defined in the combinatorial logic implied by fault trees (FTs).

2.3 Component Fault Trees (CFTs)

The traditional modularization concept of FTs (modules as independent subtrees) is not ideal for FTs of complex cyber-physical systems that are organized in a signal-flow oriented way: As parallel paths of an architecture can be fed from common sources, like power supplies or sensors, this leads to a complex FT structure with many repeated events. It is barely possible to cut the FT into separate entities that can be edited and maintained independently and that represent the actual components of a system, as defined by the architecture. If a system has multiple components of the same type (e.g. 4 or 8 rotor/motor units on a drone), the reuse in FTA is often restricted to copy&paste. Later modification, evolution, or



exchange of components by variants with a different implementation is difficult, as is well-managed reuse of components and their corresponding FTs in future projects.

Figure 1: The principle of Component Fault Trees

To overcome these limitations of traditional FTs, Component Fault Trees (CFTs) were proposed by [Error! Reference source not found.] as a new component concept. The idea is to cut out parts of FTs that correspond to technical components and turn them into reusable units with *input* and *output failure ports*, hence defining a *failure interface* of the component. At the same time, the topology was extended from a tree structure to a directed acyclic graph (DAG). This avoids the artificial splitting of common-cause faults into multiple repeated events

In the example in Figure 1 we see a system with one subcomponent that might come from a supplier. It has an input port where it depends on some input signals or supply from its surroundings and an output where it provides some service to

the superordinate system. Assuming that the failure propagation happens at these ports we can assign failure modes (e.g. supply voltage not provided at the input port, wrong signal value at the output port) to the ports. Then the supplier can model the isolated FT for the subcomponent and the integrator can model the surrounding failure propagation logic in his FT. Of course, there can be more than one subcomponent and, more importantly, there can be many occurrences of the same type of subcomponent in the system. The FT for the subcomponent type must be modeled only once, even if it is instantiated multiple times.

Since the early years many suggestions have been made to integrate CFTs with system modeling languages and ship them along with component models, see for instance [Error! Reference source not found.], where “Component-integrated CFTs”, short “C²FTs” were proposed. Later a circle of CFT contributors suggested to make this the regular case and firmly integrate CFTs with SysML Internal Block Diagrams (IBD), connecting the failure ports of CFTs with the “normal” ports of the components [Error! Reference source not found.]. The definition and usage concept of SysML v2 perfectly matches the similar concept of CFTs so that replication of components works the same in both worlds. Therefore, we use the basic idea of CFT and combine them with SysML v2 as a novel approach to MBSA.

The easy replacement of CFT components by variants with same interface enables product variants as well as agile approaches and frontloading: At first a preliminary CFT definition can be used to make the CFT complete and analyzable, later successively replaced by more accurate failure models as soon as design choices have been made and the actual technical solutions are known.

3 RUNNING EXAMPLE: A UAV FOR SEARCH&RESCUE

We will explain the suggested modular MBSE process with SysML and CFTs by the example of a UAV for search&rescue missions. For this purpose, we used our Ansys toolchain (SCADE Architect and medini analyze). We imagine a small (max. 10 kg) multicopter. It features a flight computer and a set of sensors (camera and radar) to each side, plus GNSS receiver for navigation and an IMU (inertial measurement unit, including a magnetic compass). Via radio transceiver it can communicate with its peers to self-organize formation flights with several drones to efficiently scan larger areas. It also communicates with a ground station to get its mission instructions and report any findings. For the search mission it is equipped with a suitable type of downfacing visible light or infrared camera to locate a person floating on the sea or lost in a forest or mountain terrain.

Figure 2: Combined MBSA and MBSE Process

Due to the nature of its mission, the UAV typically operates over areas where the risk of hitting humans is low; however, take-off and landing locations could be in a city, where a falling drone could injure or kill people and there is risk of colliding with buildings or disturbing other air traffic. During the development process flow the product will evolve in several iterations, introducing subsequent improvements driven by the results from safety analysis and simulation. Our goal is to use this UAV for customer demos and studies on methods and tools (e.g. simulation of autonomous systems and validation of ML for flight planning). Besides some requirements, architecture, and safety analysis we have created runnable flight control software and simulated different missions with an A&D autonomy toolchain [8]. Our focus is on the method itself, not the actual hardware, so reliability analysis and all quantitative results shown are just exemplary. Our aim is to show that our proposed workflow can help meet the requirements of aerospace standards like ARP4754A / ARP4761 and Regulation (EU) 2019/947.

4 A COMBINED MBSA AND MBSE PROCESS

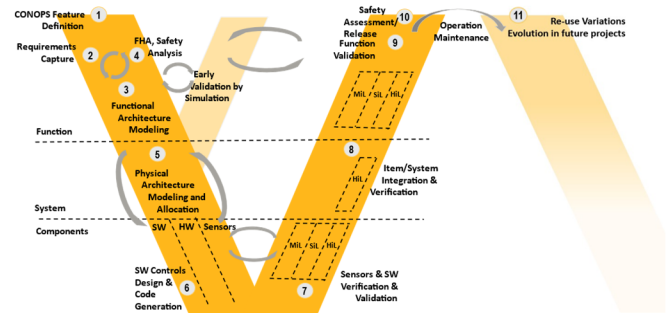
4.1 Process Overview

The process template we are presenting here (see Fig. 2) is an adaptation from a similar process we suggested for automated driving functions for road vehicles [4]. It can be extended with our process suggestions for AI-enabled autonomous aircraft [5]. Derived from a traditional V-model it features (a) usage of models from early phases, integrated with requirements refinement, (b) a separation of functional-level and technical-level development, (c) iteration / improvement loops on all levels, e.g. following the pattern requirements → modeling → analysis/simulation of the model → improved requirements → modeling of improved system and so on, and (d) integration of virtual testing / simulation as early as possible, evolved and refining by regression testing. The main steps of the process and a typical example of an iteration loop will be explained in the following.

4.2 Product Definition and CONOPS

In most cases, the development of a new system starts with informal concepts: A feature list, preliminary block diagrams, use cases or scenarios (e.g. a mission with several drones to locate a person floating on the water) along with their mission phases, and finally the Operational Design Domain (ODD), which is the set of operating conditions under which a given driving automation system or feature thereof is specifically designed to function [11]. In the aerospace domain, the term “Concept of Operations” (CONOPS) [9] is often used for such initial descriptions. We embrace this pragmatism but argue that formalized notations (as provided by SysML v2, in particular) should be used as early as possible to speed up the process to get to a first executable and analyzable model. These initial models form the starting point for both the further system

development and the first steps of the safety process, namely



the functional hazard analysis (FHA).

4.3 Functional Hazard Analysis (FHA)

We performed an FHA originally developed for manned civil aircraft compliant with ARP4754A / ARP4761. Some tailoring was applied for a small unmanned aerial vehicle: There is no risk for people on board, no crew awareness and catastrophic results of failure conditions are not to be expected (except for the hazard of collision or interference with a manned aircraft, which is normally prevented by operating in a geo-fenced area and defined altitude bands). Since the drone might take off and land in city areas, hazards to persons, vehicles and infrastructure on ground had to be considered. Thus, we took additional inspiration from Regulation (EU) 2019/947, specifically considering the SORA method [10].

The FHA is organized, as usual, in tabular form, relating supposed failure conditions to flight / mission phases and other operating conditions (like weather) where applicable. The worst-case accidents and losses are classified as Catastrophic, Hazardous etc. according to the existing standards, and resulting top-level safety requirements are derived, which add up to the stakeholder requirements from Product Definition phase. The FHA has been conducted on aircraft scope; internal failure causes (like wrong command given to rotor unit) or failure modes are derived during causal analysis via the CFT approach, and on technical level identified through FMECA.

4.4 Requirements elicitation and refinement

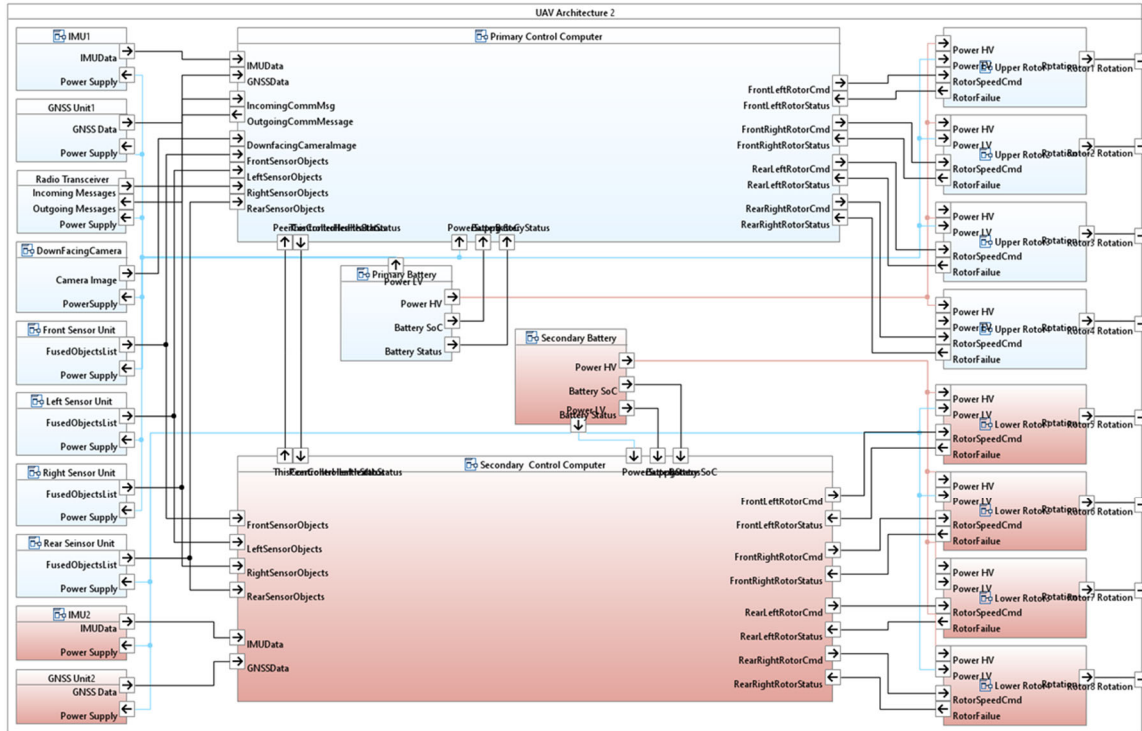
Unless there is a customer who provides a set of detailed requirements, the process of requirements elicitation can be based upon the initial CONOPS and experience with previous or similar systems, and obviously upon the list of safety objectives derived from the FHA. The first requirements are derived from the high-level objectives, the ODD, and the scenarios and use cases to be handled by the system. We recommend using a template language and, as soon as possible, pass to formal or semi-formal notations to derive acceptance criteria for the simulations. For constraints on values or the system structure it is possible to use the constraints provided by the SysML v2 language. Generally, all kinds of textual requirements can be attached to SysML model elements to ensure traceability. To the authors' experience it is advisable to develop requirements and first SysML models of the system (architecture and behavior) at the same time and in close interaction, other than suggested by the V model, where modeling starts only when requirements are accepted. One

example for a top-level requirement could be “Ensure reliable, safe and successful completion of the mission”, which can be further developed into requirements that call, for instance, for a sufficient battery capacity, the ability to always maintain a stable flight attitude, or a sensor set that provides all capabilities for the navigation and the search mission.

4.5 First Iteration: System Architecture Modeling

To demonstrate our iterative approach and the agility brought by modular and tightly integrated safety analysis techniques we start our first iteration with a naïve approach that

(e.g. battery or controller failure), or losing stability / maneuverability (e.g. loss of any of the four rotors). We will not discuss the modeling process in further detail, apart from that we mainly built on the SysML v2 ability to create part hierarchies by recursively applying part usage and port usages to create part definitions of higher-level components. Defining solution candidates (first on functional, implementation-independent abstraction level, later allocated to a technical / physical architecture with choices of concrete devices and technologies) is still a creative engineering process that cannot fully be put in a canonical recipe. What can be supported by a



normally no one would submit to safety analysis: We equip the UAV with only one control computer, only one battery unit and four rotors (see Figure 3 **Error! Reference source not found.**, but for now ignore all parts colored in red). Without much analysis it is obvious that there are a couple of single-point failures that would cause the hazards of crashing on the ground

rigid recipe is the decomposition of systems into subsystems, components, subcomponents etc. along with a decomposition of the high-level requirements into concrete and verifiable requirements on each level. We suggested such a proceeding in [12, Section 3.5] and applied it on the refinement of this system.

Figure 3: Top-Level SysML Block Diagram of UAV, 1st iteration (blue) and 2nd iteration (blue + red).

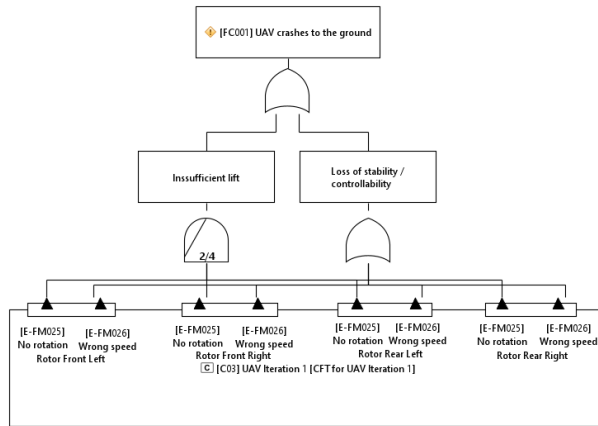
4.6 First Iteration: Early-phase behavioral simulation

It is advisable to use not just the static modeling capabilities of SysML but also the dynamic models. Sequence diagrams and activity diagrams can be used to explain the missions and use cases and describe cases for behavior simulation. State diagrams can be used to coarsely describe the behavior of the overall UAV and individual parts thereof, even if they are only approximately equivalent to the state machines used later to generate the actual embedded control software, which will contain much more details. Simulation tools that can simulate scenarios based on state machines are used to validate the correctness of the early design ideas against user

expectations and requirements. The continuous-state behavior of controllers and flight dynamics can be described by specific control or mission simulation tools that are based on ordinary differential equations or algebraic equations. Modular approaches allow a low-fidelity simulation as soon as a first skeleton of the system model is available and later exchange of models by more detailed models for different purposes as soon as they exist. This allows subsequent refinement and improvement of the system with back-to-back validation against the models from the previous stage.

4.7 First Iteration: CFT Analysis

The hierarchical static SysML architecture on functional abstraction level is the basis for a formalized safety analysis. For this purpose, we extend each model in the hierarchy by a CFT. These CFTs will be used as a modular and maintainable failure genesis and propagation models under regular top-level



fault trees that we set up to investigate the causes for each identified hazard from FHA (deductive safety analysis), see

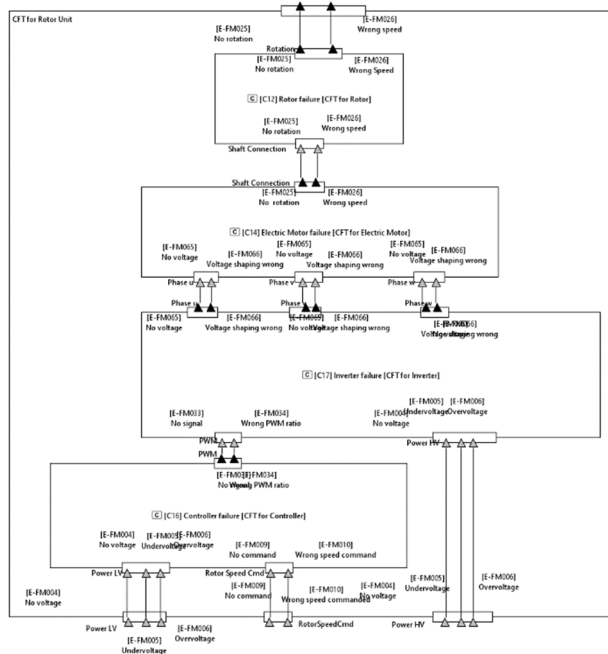


Figure 4Error! Reference source not found.. The top-level fault tree for a sample hazard looks in the upper half like a traditional fault tree, using logical gates to model the fact “The UAV may crash to the ground in case of (a) Insufficient lift, i.e. no rotation at two or more rotors, or (b) Loss of stability / controllability, i.e. at least one rotor rotating at a speed other than adequate to control the attitude”. Below this familiar structure we see a CFT component usage as a rectangle with the failure output ports (black triangles), which represent the error propagations at the ports of each component. The output ports are grouped into four rectangular output port groups, representing the output ports of the top-level SysML model of

the UAV, which are Rotor Front Left, Rotor Front Right, Rotor Rear Left, Rotor Rear Right.

Figure 4: Fault Tree for Hazard "UAV crashes to the ground"

Figure 5: CFT Definition of Rotor Unit with sub-component CFTs for Rotor (C12), Electric Motor (C14), Inverter (C17), and Controller (C16)

The define&use philosophy of SysML v2 matches exactly the idea behind CFTs: For instance, there are 4 usages of the component definition “rotor unit”. We need to model its internal failure genesis and propagation from input to output ports only once and can then make 4 usages of it. And we don’t have to model much on the level of “rotor unit” because each rotor unit is again composed of four sub-components (rotor, motor, inverter, controller) such that on the rotor unit level there aren’t any logical gates, just propagation of failure flows from component outputs to the connected inputs of an upstream component (see Figure 5Error! Reference source not found.).

The innermost component defs (e.g. for inverter) model the failure genesis and propagation inside with logical gates and basic failure events, which are traced to failure modes of the function blocks or technical components by which the SysML architecture model was augmented. In an early project phase, it might be sufficient to use stand-ins like “general hardware failure” to get the CFT model ready for analysis as quickly as possible. In later phases the CFT component definitions can be replaced by more accurate ones that show the actual failure modes of technical components, typically derived through FMECA.

We conducted a qualitative analysis of the CFT to identify minimal cut sets, which give hints on single-point failures which need to be eliminated according to our requirements, or dual-point failures which need to be subject to common-cause failure analysis. Not surprisingly, we find a long list of single-point failures leading to the hazard of “falling down” in this intentionally naïve design, ranging from exhausted battery to failures in the motor controller of any of the four rotor units. Having such results in an early phase enables us to do quick iteration cycles to improve and concretize the design.

4.8 Second Iteration: System Architecture Modelling

For the demo of the iterative approach, we also show a second iteration of the design, which avoids the single-point failures that we found, see Figure 3Error! Reference source not found., this time including the components printed in red. We added a second controller, a second battery unit, a second IMU and increased the number of rotor units from 4 to 8 (packaged in pairs, coaxially mounted at the four corners of the aircraft). The upper deck of rotors is powered by a different battery and controlled by a different computer than the lower deck. Each deck has enough power to keep the altitude or at least damp the fall to permit a soft landing. The two computers are connected by cyclic exchange of all relevant state variables and self-diagnosis results so that a device at fault can be identified and put to a passive state, allowing the other

controller to control the aircraft's flight attitude, softly and it or even make it fly back to the home base at limited speed.

4.9 Second Iteration: CFT analysis

The top-level fault tree had to be adapted to the new hazard causalities: "*The UAV may crash to the ground in case of (a) insufficient lift, i.e. no rotation at five or more rotors or (b) loss of stability / controllability, i.e. both rotors of at least one corner rotating at a speed other than adequate to control the attitude*". Also, the top-most component definition had to be remodeled, again derived from the SysML part usages of the 2nd iteration (8 x 2 output failures instead of previously 4 x 2). Most of the other components and subcomponents, in particular the rotor units, IMUs, battery units and other sensors not explained here can simply be carried over as they are still the same as in the first iteration. Maintaining / updating a complex traditional fault tree would have caused much more work and error risks. The qualitative analysis was repeated and confirmed that all initially existing single-point failures were removed.

4.10 Concretization from Functional to Physical Architecture

We recommend to continue the hierarchical refinement and the iterative improvement on the functional abstraction until a satisfactory solution is reached and all perceived risks have been mitigated and only after that proceed to technical concretization, i.e. passing from functional to logical to technical (sometimes called physical) models. In the concretization phase, design decisions on technical equipment like sensors, controllers, and motors are made. Trade studies, which can evaluate different selections against requirements compliance and optimize for given KPIs can support this selection. The cycle of designing, simulating, and analyzing (in particular for safety and reliability) continues at the technical level, now including technical errors and limitations. Between the functional, logical, and technical/physical level the traceability needs to be ensured, which is achieved by "allocation" relations between function blocks on the more abstract / implementation-independent level and logical units and finally technical components. In analogy to this concretization, the CFT approach permits replacing abstract component definitions by more concrete component definitions that come closer to the real technical implementation.

4.11 FMECA and Quantitative Safety/Reliability Analysis

On the concrete technical item level (usually under responsibility of the supplier) the inductive analysis techniques like Failure Modes, Effects and Criticality Analysis (FMECA) according to ARP4761 complement the CFT analysis and feed the CFT component failure models with quantitative failure frequency or reliability data. They are first based on the list of subsystems/components already known. Later FMECA would analyze every single item of the Bill of Material (BOM).

In the MBSA tool we are using, the FMECA is directly connected to the SysML model structure, so the list of components is represented and always synchronized between FMECA and system model. Failure modes are stored, effects are linked within the system model, and failure rates are taken

from the reliability predictions of the system components (e.g. from NPRD/EPRD databases). We directly assigned the effects defined in the FMECA to CFT base events, which automatically take over the summed-up failure rate fractions from the different failure modes leading to these effects.

4.12 Further Development Process

The remainder of the development process was only partially performed for the case study as there was no intention to actually build the drone. A remarkable ingredient of the reference process is the round-trip synchronization of the function-block architecture in SysML with the software architecture of the application software architecture in the tool SCADE Suite [**Error! Reference source not found.**]. This allows further refinement on the software architecture level, modeling of control algorithms (including import from other controller design tools such as Simulink) and finally generation of efficient embedded C code, which is certified up to DAL A. While continuing the development, the models are constantly validated against the requirements and compared to the early-stage simulations by co-simulations that integrate the actual control software into a simulated aircraft and maneuvering environment (see [8]). Although testing of actual hardware items and sensors and later the integrated aircraft are still mandatory, the simulation provides faster feedback and allows testing more scenarios and more environmental conditions.

5 EVALUTATION AND FURTHER RESEARCH

The concept of modular safety/reliability models of the functional blocks and technical components at all hierarchical levels in close integration with the "regular" system models has shown clear advantages even for this rather primitive example: The risk of losing the overview was perceived as lower and the clearer structure was appreciated by the reviewer. In this example with many reused parts, a significant reduction in modeling time was observed compared to traditional FTA, which could be further increased in the future with additional automation features, such as automatic connection of corresponding fault ports of CFT components where signal connections are known from the original SysML model. Self-contained units allow managed reuse across projects and faster impact analysis and variant comparison.

Further research will focus on aligning the CFT concepts with further SysML v2 language features, such as feature inheritance, subsets, redefinitions, and variants, striving for a deeper integration of the fault logic into the advanced formal semantics of SysML v2.

REFERENCES

1. INCOSE SE Vision 2020 (INCOSE-TP-2004-004-02)
2. B. Kaiser, B. Dion, I. Tolchinsky, T. Le Sergent, M. Najork: An AEBS Use Case for Model-Based System Design Integrating Safety Analyses and Simulation. In: *Model-Based Safety and Assessment. IMBSA 2022*.
3. SAE Aerospace Recommended Practice: Guidelines for Conducting the Safety Assessment Process on Civil Aircraft, Systems, and Equipment (ARP 4761, Rev. A).

Draft Standard, Dec. 2022.

4. B. Kaiser, P. Liggesmeyer, O. Mäckel: A new component concept for fault trees. *Proceedings of the 8th Australian workshop on Safety critical systems and software*, 2003
5. D. Domis, Integrating Fault Tree Analysis and Component-Oriented Model-Based Design of Embedded Systems. *PhD Thesis, University of Kaiserslautern*, 2012
6. B. Kaiser et al: Advances in Component Fault Trees. *28th European Safety and Reliability Conference ESREL 2018*
7. K. Beck et al: Manifesto for Agile Software Development. Website agilemanifesto.org, retrieved on July 11th, 2023
8. B. Dion et al: Mission-driven and Safety-critical Software Development for Aerospace and Defense. NAFEMS World Congress 2023, Tampa, Florida, USA
9. ISO/IEC/IEEE 15288:2015 Systems and software engineering -- System life cycle processes
10. EASA: [Specific Operations Risk Assessment \(SORA\)](https://easa.europa.eu/en/sora) | [EASA \(europa.eu\)](https://easa.europa.eu), retrieved on June 16th, 2023
11. BSI PAS 1883
12. B. Kaiser, R. Weber, M. Oertel, E. Böde, B. Monajemi, J. Zander: Contract-Based Design of Embedded Systems Integrating Nominal Behavior and Safety. *Complex Systems Informatics and Modeling Quarterly*, Oct 2015

13. Object Management Group: Systems Modeling Language (SysML) Version 2.0, Release 2023-02

BIOGRAPHIES

Dr. Bernhard Kaiser works as a Principal Safety Consultant with Ansys Germany. He has a long experience record as a developer, consultant, and researcher. At Ansys he specializes in Safety Analysis, Simulation, and MBSE.

Michael Soden works as Lead Product Manager at Ansys, Germany. He holds a Ph.D. in computer science from the Systems Analysis and Simulation Department of the Humboldt Universität Berlin. For more than 15 years, his focus has been on the implementation of model-based methods for safety and reliability in various industries.

Nils Heuermann works as Senior Application Engineer at Ansys, Germany. He is an Aerospace Engineer with many years of experience in the development of HiL test systems and hardware and software for aircraft components and engines as a systems engineer and project manager. Over the last ten years, he has specialized in safety analyses and is an expert for aviation safety and cybersecurity at Ansys.