Model-based safety assessment for conceptual aircraft systems design

Simon Gradel¹ · Benedikt Aigner¹ · Eike Stumpf¹

Received: 25 January 2021 / Revised: 7 October 2021 / Accepted: 27 October 2021 / Published online: 23 November 2021
© The Author(s) 2021

Abstract

Traditional system technology modeling in conceptual aircraft design mainly relies on empirical knowledge and methods derived from conventional systems, for which valid system architecture designs are known. Since these systems have been proven valid especially from a safety perspective, detailed system safety analyses are usually not necessary. For unconventional systems and innovative technologies, on contrary, new architectures have to be designed and system safety has, therefore, to be taken into account. Therefore, the application of model-based safety assessment (MBSA) for designing system architectures in conceptual aircraft design studies is proposed. A MBSA approach based on a Simulink architecture model is presented which is tailored for use in conceptual design studies. It is applied to the cryocooling system of a hybrid-electric powertrain architecture from an already-published study. The original architecture as well as possible architecture alternatives are investigated. As a result, a safer architecture version with lower number of components can be proposed. The application example indicates that using MBSA in conceptual design benefits the latter by providing insights into safety properties of the system and by pointing out architecture safety weaknesses. This could result in safer, thus more realistic system architectures.

Keywords System safety · Conceptual design · PSSA · Model-based safety assessment · Fault tree analysis

Abbreviations

<i>I</i>	Importance
<i>P</i>	Probability
CFT	Component fault tree
FEM	Failure effects modeling
FLM	Failure logic modeling
FMES	Failure mode and effect summary
FTA	Fault tree analysis
HiP-HOPS	Hierarchically performed hazard origin and propagation studies
MBSA	Model-based safety assessment
MCS	Minimum cut set
PSSA	Preliminary system safety assessment
SysML	Systems modeling language
TLE	Top-level event

1 Introduction

The ambitious goals for the European aerospace industry, as formulated in the “Flightpath 2050” position paper of the European Commission [1] ask for new aircraft system technologies to reduce commercial aviation’s impact on the environment. The “Strategic Research and Innovation Agenda” (SRIA) of the Advisory Council for Aviation Research and Innovation in Europe (ACARE) identifies preliminary system architecture studies as a crucial field for today’s aviation research to investigate more-electric system designs and electric and hybrid propulsion approaches for future radically innovative aircraft configurations [2]. To ensure that only realistic, i.e., certifiable system architectures are considered, system safety becomes an important issue. Performing the complete preliminary system safety assessment (PSSA) process by hand, however, is time-consuming and seems inappropriate for conceptual aircraft and system research. General “redundancy guidelines”, i.e., empirical rules for redundancy allocation, present a possible solution to this challenge, but lack quantitative information and are subjected to individual judgment and experience. Model-based safety assessment (MBSA) presents an alternative to solve this problem. MBSA uses models to describe the fault behavior of a system. Consequently, safety analyses (e.g., the

✉ Simon Gradel
simon.gradel@ilr.rwth-aachen.de
Benedikt Aigner
benedikt.aigner@ilr.rwth-aachen.de
Eike Stumpf
eike.stumpf@ilr.rwth-aachen.de

¹ Institute of Aerospace Systems, RWTH Aachen,
Wüllnerstraße 7, 52062 Aachen, Germany

synthesis of fault trees) can be performed partly automatized with these models. Enabling the reuse of modeling elements, thus making possible architecture trade studies with respect to safety properties without extensive manual effort, MBSA seems suitable for use in conceptual design studies.

The contribution of this paper is twofold: First, an MBSA method suitable for conceptual system design studies is presented. Second, it is shown via an example system from an already published design study that applying MBSA to conceptual aircraft design provides benefits for the generation of system architectures.

This paper is structured as follows: Sect. 2 presents the terminology used within this publication. In Sect. 3, a brief overview of the PSSA and fault tree analysis, as well as several important concepts is given. Section 4 gives a systematic overview over existing MBSA approaches and gives some examples. In Sect. 5, the novel MBSA approach is presented and categorized according to the systematics from Sect. 4. In Sect. 6, an example system from a previous publication is presented which is consequently analyzed and optimized using a Matlab tool implementing the proposed approach. Conclusions on the validity and usefulness of the chosen approach are drawn in Sect. 7.

2 Terminology

In the following, the term *system* shall describe an entity which is part of the aircraft and provides one or more functions to the aircraft (e.g., the flight control system provides the function “attitude control” to the aircraft) and/or to other systems (e.g., energy supply systems). A *component*, on the other hand, is the smallest system element considered. It is not further decomposed into its parts and owns a certain behavior and performs functions for the system. *Subsystems* represent intermediate levels of abstraction, being part of a system and being aggregations of components and/or other subsystems. Systems and subsystems do not have proper behavior, instead, their behavior results from the interaction of their components.

3 PSSA

The preliminary system safety assessment process as defined by the Society of Automotive Engineers (SAE) in the Aerospace Recommended Practice (ARP) 4761 [4] is a standardized process for system architecture development and for establishing component safety requirements. The PSSA starts with an aircraft and a system functional hazard assessment (FHA), during which critical aircraft or system conditions are determined, respectively. These conditions are then used as a starting point for deductive analyses, e.g., the

fault tree analysis (FTA). The FTA is a qualitative, graphical and deductive analysis, relating a system-wide failure condition with its root causes. This analysis starts at system level with a top-level event (TLE). This event is then successively decomposed by finding its immediate causes and their logical relation. These causes are formulated as failure events. The events and the relations are depicted as boxes and gate symbols in a directed acyclic graph, the fault tree. The two main relations are the AND-gate (all subevents of the gate must occur for the top event of the gate to occur) and the OR-gate (one subevent occurrence is sufficient for the top event to occur). The symbols are shown in Fig. 1. Events, which are not further decomposed, are referred to as basic events. They are marked with a circle below the event box. The analysis is terminated if the ends of all branches are basic events.

A “smallest combination” of these basic events “which, if they all occur, will cause the top event to occur”, is called a Minimum Cut Set (MCS) [3]. They can be obtained from the fault tree via Boolean algebra operations. MCS are used for multiple analyses, both qualitatively and quantitatively [3]. Qualitative analyses comprise

- Qualitative importances of basic events.
- Common cause failure susceptibility.

Quantitative results include

- Top-level and intermediate event probability calculation.
- Quantitative importances.

Importances quantify the impact of a basic event or MCS on the top-level event probability. Hence, for system analysis, they help to find system safety “weaknesses” (e.g., the components with largest contribution to system failure events). To obtain a qualitative importance, the order of a MCS can be considered: Lower order MCS (i.e., those having few

Fig. 1 Fault tree symbols acc. to [3]

basic events) typically have a higher impact than higher order MCS, as the MCS probability decreases with every event. Assuming low probabilities, a change in MCS order leads to a change in MCS probability of several orders of magnitude. Quantitatively, the MCS importance can be calculated by dividing the MCS occurrence probability by the top-level event probability [3]:

$$I_{\text{MCS}} = \frac{P_{\text{MCS}}}{P_{\text{TLE}}}. \quad (1)$$

4 Model-based safety assessment approaches

Model-based safety assessment describes methods which use system models to determine system safety properties in a more or less automated way. Lisagor [5] classifies MBSA approaches with regard to two dimensions: The provenance of the system model and the semantics of component dependencies. The latter is classified on a scale between two extremes, the Failure logic modeling (FLM) approach and Failure effects modeling (FEM). They differ in how the components interact with each other.

In FLM models, components exchange only failure modes. As an example, consider a hydraulic pump supplying hydraulic pressure to a valve. If a failure occurs causing the pump to stop operation, it propagates a corresponding failure mode (e.g., an expression as “no pressure supplied”) to the valve. The pump then reacts to this failure information in a certain way, e.g., by propagating another failure mode to downstream components. The failure model is, therefore, completely written in failure space, which corresponds to the “classical” FTA approach. The complementary concepts of failure space and success space as defined in [3] categorize events with respect to two perspectives: The success space perspective describes to what degree a certain function of the system has been fulfilled, whereas the failure space perspective describes (the same situation) with respect to an undesired state (e.g., loss of human lives).

A disadvantage of FLM is the strong context dependency of component failure models: The failure mechanism description of one component strongly depends on the description of the upstream components’ failure modes. Therefore, reusing component models in a different context becomes difficult. In FEM models, on contrary, components exchange abstractions of their interactions. Considering again the pump/valve arrangement, the continuous pressure value could be abstracted by a Boolean value, being false (no pressure) or true (positive pressure value). Alternatively, more than two discrete values could be defined (e.g., no pressure, low, high). Evidently, FEM models are closer to

design models, which facilitates model construction. Also, dependencies between component failure modes are less strong than in FLM, so the reusability of component models is better. Also, FEM models are formulated in success space, which is contrary to the “traditional” FTA methodology as defined in [3]. In addition, component interaction only occurring in non-normal system state (e.g., interaction through physical proximity, electric shortcut etc.) are rarely considered in this approach.

The other classification dimension is the provenance of the model, with the two extremes of a complete usage of the system design model and construction of an entirely new model. When MBSA in a late phase of a systems design process, functional models of the system, e.g., in Simulink, are often available. These models can be partially (e.g., using the system structure) or completely (e.g., by integrating safety model aspects into the design model) reused for safety analysis. Whereas reusing the design model facilitates the modeling process, Lisagor [5] points out that a dedicated model construction with intent different from a design model (a design model describing how the system should work, a safety model describing how a system can fail) provides a certain “analytical redundancy” in the system analysis.

From the various methods for MBSA developed in the past decades, only a few prominent examples shall be presented in the following.

AltaRica is a safety modeling language first proposed in 1999 by Point and Rauzy [6]. Different versions have been published in the following, the latest being AltaRica 3.0 [7]. They have been used in a variety of publications (e.g., [8, 9]). Using a modified variant of state automata for component behavior description and flow equations called mode automata (for AltaRica 3.0: Guarded Transition Systems), it can be used for FEM models very well (e.g., consider the AltaRica 3.0 tutorial example [10]), but principally also supports FLM model edition (as demonstrated by Lisagor [5]), and is often used in hybrid approaches (e.g., [11]). In AltaRica 3.0, components are modeled with different states (state variables) and transitions between these states which are triggered by events. Components interact via flow variables. The flow variable propagation behavior depends on the component state, i.e., on the current values of the state variables. Variables can be, e.g., Boolean, but it is also possible to define user-specific “domains”.

Hierarchically performed hazard origin and propagation studies (HiP-HOPS) is a genuine FLM method first published in 1999 [12]. Its goal is the automation of traditional manual safety analysis, especially fault tree analysis. It consists of a structural model (in a later publication, this was replaced by a Simulink model [13]) and fault behavior tables called IF-FMEA (Interface-Focused Failure Modes and Effects Analysis) for each component. In these tables, a component’s reaction to input deviation as

well as internal failures are defined. Output failure modes are encoded with respect to a special scheme, indicating the deviation type (output omission, stuck at zero etc.). The component interactions are limited to nominal, i.e., intended interactions.

Component fault trees (CFT) presented by Kaiser et al. [14] can also be classified as an FLM method. In CFT, subtrees for each system component are constructed, which are then joined at their ports according to a system structure model. The subtrees and the synthesized fault tree are then not trees anymore, but directed acyclic graphs. Various extensions and elaborations of the method exist [15], among others an integration with a Simulink system model [16].

Joshi et al. [17] propose the extension of a Simulink design model with failure events. The failure events consist of component inputs which alter the component output behavior in a certain way. Component faults are then propagated via the nominal signal paths in most cases. This could be seen as a variation of the FEM approach. In addition, the authors propose to specify special failure signal paths for backwards fault propagation and other non-nominal interactions, which extends the methodology to a hybrid FEM/FLM model.

The systematics from [5] and the placement in this classification of the mentioned MBSA approaches, as well as our proposed approach which is presented in this publication, are graphically shown in Fig. 2.

Fig. 2 MBSA classification scheme after [5]

5 MBSA method description

The MBSA methods presented above were developed for system safety assessment purposes, not for conceptual design purposes. Therefore, these methods typically require a large effort for model construction. Also, architecture trade studies require major model modifications, as will be explained below. Therefore, in the following, a novel MBSA method for use in conceptual design studies shall be presented. To make it valuable in the conceptual design context, the following aspects are important:

- a fast and uncomplicated architecture definition,
- a graphical system representation,
- reusability of the component behavior models,
- fast architecture manipulation once the model is created.

A very important point is the architecture manipulation. To enable trade studies, it should be possible to add or remove components in the architecture model without having to change the component failure model. This is, however, not possible in many existing approaches, as the component behavior models largely depend on their context. For example, if it is investigated whether a motor should have fuel input from one or two pumps, the behavior model of the motor must be changed to account for the additional input. For conceptual design studies; however, it would be desirable to move the model complexity from the structural model to the component behavior description, in a way that these remain valid in many contexts (e.g., when adding redundancies). This enables a fast model manipulation once component behavior models are defined and a higher reusability of the component models.

On the other hand, the evaluation of the model should yield results similar to “traditional” analyses. As the notation and systematics is widely known and recognized, fault tree analysis is a suitable analysis method. As pointed out above, FTA enables qualitative and quantitative evaluation of the system safety properties, thus helping to identify system weaknesses, even when precise quantitative data is unavailable. Therefore, the model should support the automatic generation of fault trees and their automatic evaluation.

As models can be seen as a combination of structure and behavior [18], these two aspects are considered separately. In the following, the system structure model is presented first (5.1), then the component behavior model (5.2) is described.

5.1 System structure model

The system structure is defined via a Simulink model. Simulink has two main advantages: Its close coupling to

the Matlab programming environment used for the tool implementation and its intuitive and simple representation and edition of model structure and hierarchy via blocks. For several MBSA approaches, Simulink has already been used (e.g., [13, 17]). An alternative would be SysML, the Systems Modeling Language defined by the Object Management Group (OMG), which has also been employed for system modeling in MBSA (e.g., [19]). Definitively, SysML's modeling concepts exceed the expressiveness of a Simulink model, for example by providing the possibility to model several parallel hierarchies. Therefore, it is a very suitable candidate for alternative approaches and future developments. On the other hand, this would require the integration of a SysML modeling tool into the workflow, and would require the user to be acquainted with the SysML notation, which is, as to the authors' personal experience, less intuitive than a Simulink model.

Figure 3 shows parts of a Simulink model. "Subsystem" blocks are used to represent components and signals represent functional interactions. The labeling of blocks and signals is very important as it allows for the identification of component types and the attribution of failure mode propagation to component interactions. An example for a block label is

Fig. 3 Example model in Simulink

The first part ("PMP") identifies the component type. The component type relates a certain behavior (see below Sect. 5.2) to the block. The second part can be chosen arbitrarily. Signals must be labeled with the connection type (e.g., "HydraulicPressure", "Fuel"). System outputs (i.e., information/ energy/ mass flows the system provides to the aircraft) are modeled by signals ending in a terminator block. It is important to notice that the analysis method does only require the system structure, i.e., subsystem blocks and their connections, to perform the analysis. The component behavior models are defined separately, not in the Simulink model.

For large models, it may be useful to organize a model into several hierarchical levels. With the "Subsystem" block, this can easily be accounted for in Simulink. Currently, the analysis tool allows only for two hierarchy levels.

5.2 Component behavior model

As stated above in Sect. 2, failure behavior is attributed to components. Therefore, a so-called *FMES* (after the failure mode and effect summary [21]) table is defined for each employed component type. These are stored in a component type library file. The failure behavior description is based on the model of the failure cause/mode/effect terminology from [3]. A component has several defined failure modes, which are triggered either by an internal failure (these shall be referred to as *internal* failure modes) or by an external failure cause, i.e., a failure effect of an upstream component (*external* failure mode). Each failure mode has one or more failure effects, which are propagated along the defined connections. To identify the relevant upstream components for a certain failure mode, a *RelatedInput* parameter is defined for each failure mode. In this parameter, one or several signal labels are specified. To find the cause of an external failure mode, all upstream components connected to the current component by a signal with the defined label are considered. This is a crucial point and a major difference to existing MBSA approaches, as it allows for the modification of the number of (similar) inputs without changing the component behavior model. For an external failure mode, the failure mode identifier of the downstream component is identical to one of the failure effects of the upstream component (see Fig. 4), or is explicitly related to some other effect of the upstream component. The latter is done via a *CorrespondencyList*, on which correspondencies between external failure modes and failure effects of other components are defined. For example, it makes sense to relate the failure effect identifier "NOPO - No pressure output" to the failure mode identifier "NOPI - No pressure input" for all component types.

In case of only one upstream component relevant for the failure mode, this scheme can be used to trace component failure modes backwards to upstream components, until

Fig. 4 Component fault propagation modeling scheme

every branch has reached an internal failure mode, which is treated as a basic event. In redundant systems, on contrary, several input signals of the same type (i.e., with the same label) exist. Depending on the nature of the failure mode, all or just one of these inputs have to show a certain fault. This corresponds to the AND and OR gate of the fault tree methodology. This is stored in the second row of the *RelatedInput* for each signal label. The scheme shall be detailed with an example *FMES* for a hydraulic distribution system component, which is shown in Table 1.

The component has two failure modes: “LEKG” and “NOPI”. The first is an internal failure mode, i.e., a failure that originates in the component itself. This can be seen by the *RelatedInput* argument, which is ‘Internal’, and the definition of a failure rate for this failure mode. The latter is an external failure mode, caused by an upstream component. It is propagated to the pressure distribution via a signal labeled with “*HydraulicPressure*”. The second row of the

Table 1 *FMES* example content for internal failure modes and fault propagation

<i>CompID:</i>	HDST
<i>Name:</i>	Hydraulic pressure distribution system
Failure mode 1	
<i>FailureModeID:</i>	LEKG
<i>FailureModeDesc:</i>	Leakage of the system
<i>Failure rate:</i>	0.0001
<i>RelatedInput:</i>	{‘Internal’}
<i>FailureEffectID:</i>	NOPO
<i>FailureEffectDesc:</i>	No hydraulic pressure output
Failure mode 2	
<i>FailureModeID:</i>	NOPI
<i>FailureModeDesc:</i>	No hydraulic pressure supplied
<i>Failure rate:</i>	–
<i>RelatedInput:</i>	{‘HydraulicPressure’ ‘AND’}
<i>FailureEffectID:</i>	NOPO
<i>FailureEffectDesc:</i>	No hydraulic pressure output

RelatedInput indicates the gate type, in this case an AND-gate. If more than one signal labeled with “*HydraulicPressure*” is connected to a HDST component, an AND-gate is created and the upstream components with the respective failure effect are written as subevents. Both failure modes in Table 1 have the same effect: “NOPO”, no pressure output.

In some cases, it is useful to describe a fault that is a combination of internal and external causes or of more than one external failure mode propagated to the component via different input signals. Those combined failure modes can be written with an “&” delimiter separating the partial failure modes. In this case, the related input is an array of several signals. The single partial failure modes are repeated as own failure modes, but with the failure effect “NODE”, no direct effect. An example is shown in Table 2. In the FTA construction, the combined failure mode (in the example “Failure mode 1”) is resolved to an event with an AND-gate. If the failure mode requires an OR-gate, it should be modeled as several independent “normal” failure modes. The partial failure modes (2 and 3) contain the gate type (AND/OR) of the subgates in the second row of the *RelatedInput* parameter.

Table 2 *FMES* example content for a combined failure mode with Priority-AND logic

<i>CompID:</i>	SVLV
<i>Name:</i>	Hydraulic pressure switching valve
Failure mode 1	
<i>FailureModeID:</i>	SWF& NPJA
<i>FailureModeDesc:</i>	Switch failure & No pressure supply on active port
<i>FailureMode Probability:</i>	– 1
<i>RelatedInput:</i>	{‘Internal’ ‘HydrActive’ ‘1’ ‘2’}
<i>FailureEffectID:</i>	NOPO
<i>FailureEffectDesc:</i>	No hydraulic pressure output
Failure mode 2	
<i>FailureModeID:</i>	SWF
<i>FailureModeDesc:</i>	Loss of switching capability
<i>FailureMode Probability:</i>	0.001
<i>RelatedInput:</i>	{‘Internal’}
<i>FailureEffectID:</i>	NODE
<i>FailureEffectDesc:</i>	–
Failure mode 3	
<i>FailureModeID:</i>	NPJA
<i>FailureModeDesc:</i>	No hydraulic pressure on active port
<i>FailureMode Probability:</i>	– 1
<i>RelatedInput:</i>	{‘HydrActive’ ‘AND’}
<i>FailureEffectID:</i>	NODE
<i>FailureEffectDesc:</i>	–

The logic of “standard” fault trees considers only the combination of events, but not their order. Many components, especially those with a reconfiguration behavior, e.g., a valve switching from a normal pressure supply line to an alternate line, can show a fault behavior which is dependent on the order of events. If the valve has a failure mode “SWF” which signifies that the valve can not change its position after this failure event has occurred, the valve shows a different failure effect if first the pressure decreases in the normal pressure supply and then the valve becomes stuck, or vice versa. In the first case, the valve switches to alternate pressure, and pressure output from the valve is maintained. In the second case, a switching to alternate is not possible, hence, output pressure can not be maintained. Therefore, an extension for the classical fault tree was proposed by Dugan [20]. Using (among others) the new Priority-AND gate (PAND-gate), the fault tree becomes dynamic, i.e., sequence-sensitive. Using the already mentioned combined failure modes, it is possible to define an additional “order” attribute for dynamic combined failure modes, representing a PAND-gate. This is done in the second row of the combined failure mode’s *RelatedInput*. In Table 2, for example, the failure mode “SWF&NPiA” occurs only if “SWF” occurs before “NPiA”, as indicated by their numbering.

As could be seen from the description above, all events are strictly described in failure space. All failures modes and effects are, therefore, written as “deviations from intent” [5]. This is convenient as it is in accordance with the “classical” FTA methodology; therefore, also the obtained fault tree corresponds more to a fault tree obtained by manual analysis, thus facilitating use of the method by users already experienced with the “classical” method.

The proposed scheme is in some respects similar to the IF-FMEA of HiP-HOPS. As both methods are basically FLM approaches, this can be expected. Both extend the classical FMEA/FMES scheme with fault propagation behavior. To a lesser extent, it is also comparable to CFT, which does not store component behavior in tables but in form of small subtrees. The main difference, however, is the greater flexibility of the proposed approach when adding or removing additional inputs to account for redundant configurations. A redefinition of the component *FMES* is not necessary in this case. In HiP-HOPS, on the contrary, the IF-FMEA of the respective component would have to be changed, as well as the subtree of a CFT. The system structure and the component behavior models are, therefore, more closely coupled in these approaches, which makes trade studies more time-consuming. Also, output deviations in HiP-HOPS are limited to a certain set of constructs [12], which is not the case in the presented approach.

Along with the advantages for conceptual design studies, there come some limitations of this approach: In more complicated contexts, adding or removing components might

not be possible without changing the component behavior in the *FMES*. If, for example, the switching valve with a normal and an alternate pressure input port mentioned above is considered, these inputs have very different meanings concerning the valve component behavior and must, therefore, be labeled differently, even if they describe the same physical connection. This is a direct consequence of the fact that failure mode propagation paths are specified via signal labels and not by ports. Adding a third input or removing one will, therefore, not be possible without changing the valve’s *FMES* table. Also, the connected components’ type descriptions have to account for the different signal labels, increasing context dependency and reducing reusability of the component type models. In addition, as failure modes are not directly mapped to output ports, analysis methods as stepwise simulation or stochastic simulation, which require a forward failure mode propagation (from inputs to outputs), are impractical with the presented model. Another point is the propagation direction of failure modes. In the presented approach, a reverse or bi-directional propagation of failure modes (e.g., when a pressure loss due to leakage propagates upstream as well as downstream) can not be modeled without adding a duplicate connection. This, for example, can easily be done in AltaRica 3.0.

5.3 Fault tree generation and evaluation

A coding scheme for fault tree events is required for automatic fault tree generation. The scheme employed is similar to the proposed scheme from [21]. It consists of three parts. The first two parts identify the component. The component type identifier presented in Sect. 5.1 in combination with a number are used for this purpose. The third part is the failure mode or effect identifier presented in Sect. 5.2. A typical event code is shown below:

PMP-001-FAIL

The component numbers and their respective names in the Simulink model are tracked via a matching table. With the structural information from the Simulink model and the component behavior model described in Sect. 5.2, it is possible to relate failure modes to their upstream component causes through the system. For a fault tree analysis, a top-level event must be defined. To automatically construct a fault tree, this system-wide failure condition must be decomposed into subevents, until component failure modes are reached, which can then be further decomposed as described in Sect. 5.2. To define a TLE, the following input has to be provided:

- *OutputSignals*: Labels of the output signals that show the fault (can be more than one)

- *TopGate*: The gate of the top-level event, if more than one of the *OutputSignals* exists or if there are several different *OutputSignals*
- *GateTypes*: Specification of the intermediate event sub-gates (for different *OutputSignals*)
- *FailureEffectIDs*: The failure effect for every output signal (for different *OutputSignals*)

A simple example TLE for the system shown in Fig. 3 would be “NOPRS”, no pressure output of the system. This could be defined as follows in the scheme:

- *OutputSignals*: HydraulicPressure
- *TopGate*: -
- *GateTypes*: ‘AND’
- *FailureEffectIDs*: NOPO

There is only one system output (signal ending in a terminator block), the “HydraulicPressure” output of the valve. Therefore, the TLE has only one subevent, which is the component providing the “HydraulicPressure” output showing the specified failure effect, “NOPO”:

VLV-001-NOPO

As a more complex example for a TLE definition, a braking system including a hydraulic wheel brake and two ground spoilers shall be considered. The system output, a brake force applied to the aircraft, can be produced in two different ways corresponding to two different output signals: A brake moment around the wheel axis, named “BrakeMoment”, and an aerodynamic brake force named “AeroBrakeForce”. Two spoilers (SPCS) provides the output “AeroBrakeForce”, one brake disc (BDSC) provides the output ‘BrakeMoment’.

The settings for the TLE definition of “NOBR” are then specified as follows:

- *OutputSignals*: AeroBrakeForce, BrakeMoment
- *TopGate*: ‘AND’
- *GateTypes*: ‘AND’, ‘AND’
- *FailureEffectIDs*: NOAB, NOBM

The two *FailureEffectIDs* identify the failure effects “No aerodynamic brake force” and “No brake moment”. These are written as AND-related (as specified in *TopGate*) subevents of the TLE in the form “SYS-001-<Failure mode ID>”. The upper part of the fault tree is depicted below in Fig. 5.

After the TLE has been decomposed into a component failure mode, this mode can be further decomposed using the structural and behavioral information described above. In an iterative, backward procedure, every event which is not a basic event (i.e., not an internal failure mode), is decomposed into its subevents. The algorithm (for the

Fig. 5 Top gate arrangement for the TLE NOBR

decomposition of one event) is shown as a flowchart in Fig. 6. If the event is a failure effect of the respective component, the component’s failure modes having this effect are written as OR-related subevents.

If, on the other hand, the event is a failure mode of the component, its *RelatedInput* is considered. If there is one related input, all upstream components connected with a signal of the specified label to the current component are determined. If there is a single upstream component, the failure effect which has to occur at this component is determined. Either, this failure effect of the upstream component is equal to the failure mode of the downstream component, or the corresponding failure effect is found on the *CorrespondencyList*. The failure modes of the upstream component causing this failure effect are then stored as subevents with an OR-gate relation. If more than one upstream component provides a signal to the component with the label specified in the *RelatedInput* of the failure mode, subevents are generated with the failure effects of the upstream components. The gate type is determined from the second row of the *RelatedInput* parameter.

For a combined failure mode, the partial failure modes are first written as AND-gate related subevents, which are then treated as “normal” failure modes.

In the example system, “NOPO” is a failure effect of component VLV, as can be seen in Table 3. Therefore, all corresponding failure modes of this component are determined. The two failure modes “BLCK” and “NOPI” are both associated with this effect. Therefore, these are written as subevents, related with an OR-gate. “BLCK” is internal, hence a basic event. “NOPI” is an external failure mode and its *RelatedInput* is “HydraulicPressure”. The only upstream component providing this input type is the Distribution component. “NOPI” has a corresponding failure effect on the *CorrespondencyList*: “NOPO”. Therefore, failure modes of the upstream component HDST having the effect “NOPO” are searched. Two failure modes are found: “LEKG” (internal, hence basic event) and “NOPI” (again external). These are written as subevents with an OR-gate. The *RelatedInput*

Fig. 6 Flowchart of the event decomposition algorithm. The algorithm is applied to each non-basic event of the currently lowest fault tree layer. The generated subevents form the next fault tree layer, on which the procedure is repeated in the next step

of “NOPI” is again “HydraulicPressure”, the gate type is AND. In this case, there are two upstream components providing this input. Therefore, two AND-gate related subevents are written to the fault tree with the failure effect “NOPO” (from the *CorrespondencyList*) for the components PMP-001 and PMP-002. These two effects can, in a last step, be decomposed into the single failure mode “FAIL” of PMP (see Table 4), which is a basic event. The complete fault tree of this example is shown in Fig. 7. After the fault tree has been constructed, the minimum cut sets are determined using the “bottom-up” algorithm described in [3]. For this algorithm, the laws of Boolean algebra are applied to gate

equations, so that cut sets are obtained for each non-basic event. This is done successively from the bottom to the top, until the top-level event is reached. These cut sets can then be used for a variety of analyses, as mentioned in Sect. 3.

If the fault tree is dynamic, MCS can not be used for the evaluation. Instead, an algorithm is provided to convert the fault tree into a Markov model (following the scheme sketched in [21]). The obtained TLE probability which corresponds to the probability of the system being in the TLE state can be approximated by the solution of the discrete Markov chain for small time steps using the Matlab-internal functions *dtmc* and *redistribute*. However,

Table 3 FMES example content for a valve component

<i>CompID:</i>	VLV
<i>Name:</i>	Valve
Failure mode 1	
<i>FailureModeID:</i>	BLCK
<i>FailureModeDesc:</i>	Valve blocked
<i>Failure rate:</i>	0.0002
<i>RelatedInput:</i>	{ 'Internal' }
<i>FailureEffectID:</i>	NOPO
<i>FailureEffectDesc:</i>	No hydraulic pressure output
Failure mode 2	
<i>FailureModeID:</i>	NOPI
<i>FailureModeDesc:</i>	No hydraulic pressure supplied
<i>Failure rate:</i>	-
<i>RelatedInput:</i>	{ 'HydraulicPressure' 'AND' }
<i>FailureEffectID:</i>	NOPO
<i>FailureEffectDesc:</i>	No hydraulic pressure output

Table 4 FMES example content for a pump component

<i>CompID:</i>	PMP
<i>Name:</i>	Hydraulic pump
Failure mode 1	
<i>FailureModeID:</i>	FAIL
<i>FailureModeDesc:</i>	Pump failure
<i>Failure rate:</i>	0.001
<i>RelatedInput:</i>	{ 'Internal' }
<i>FailureEffectID:</i>	NOPO
<i>FailureEffectDesc:</i>	No hydraulic pressure output

qualitative analyses are not possible, only probability calculations. In addition, this solution method requires significantly more computation time.

The automatic fault tree generation and the evaluation methods were implemented in a Matlab program. The user can define the FMES tables and TLE in a GUI. The interface for the FMES edition is shown in Fig. 7. The system structure, as already mentioned above, is edited in Simulink. The entire rest of the analysis is completely automatized. A graphical fault tree representation is generated and a report containing the results of the various analyses is written.

6 Application

In the following, the tool described above shall be applied to an example from an existing conceptual aircraft design study on a new and innovative system technology. The program's

Fig. 7 Fault tree for the example system

results shall be used to assess strengths and weaknesses of the system architecture concerning system safety and to investigate alternative architecture options. The main question is, as stated above, if the insight into the system properties provides significant benefit so that the additional effort of using the program is justified.

The example system is taken from [22]. The study investigated a hybrid-electric powertrain for CS-25 commercial aircraft. The basic idea was to use, in addition to two conventional turbofan engines, two ducted electric fans which receive their complete energy from batteries. To efficiently distribute and convert the high amounts of electric energy, superconducting power buses, electric motors, and motor controllers are employed. These require a cryocooling device. [22] proposes a distributed cryocooler architecture, i.e., every superconducting component is equipped with its own cooling unit. The complete architecture is shown in Fig. 8. The electric motors (SCEM) are each controlled by its own motor controller unit (MCNT). These, in turn, can receive their electric power by each of the two high-power bus systems, which also use the energy of each battery. This arrangement is similar to a conventional cross-feed fuel supply system.

In this publication, the cryocooler arrangement shall be studied. To do so, two top-level events are considered: OEI—One Engine Inoperative (one thrust-producing device does not deliver thrust), and AEI—All Engines Inoperative

Fig. 8 GUI for the Matlab program, *FMES* edition shown

(all thrust-producing devices do not deliver thrust). The switches are modeled without internal failure modes. For the superconducting components, it is assumed that a failure in the respective cryocooling unit leads to a component failure. However, there is evidence in the literature that for at least some systems, cryogenic temperature can be maintained for a longer timespan without coolant supply [28]. The internal failures of the modeled components and their failure rates are shown in Table 5. Where no reasonable failure rate data was available, values close to other components' failure rates were chosen not to obscure qualitative system properties by quantitative effects. Concerning the cryocoolers, only failure rates for Stirling cryocoolers were available to the authors, and these varied by a range of two orders of magnitude. A median value was chosen for this study.

As mentioned above, the ability to perform fast architecture trade studies is a crucial requirement. Therefore, the distributed architecture, as proposed in the original study, a completely centralized cryocooler arrangement, as well as

two mixed arrangements are to be studied. The resulting architectures are:

- Original, distributed system (6 cryocoolers)
- Cryocoolers of SCEM and its MCNT are merged (4 cryocoolers)
- One cryocooler for each system branch (SCEM, MCNT, HPBS, 2 cryocoolers in total)
- Completely centralized system (1 cryocooler)

These are analyzed with respect to the two top-level events. The results for “OEI” can be seen in Table 6.

It can be observed that with a decreasing number of cryocoolers, the number of minimum cut sets, thus the top-level event probability decreases. This could be expected, as the superconducting motor and its motor controller are in serial arrangement, hence, their cryocooler being disparate does not constitute a redundancy, but merely another source of

Table 5 Internal failure modes (i.e., basic events) and failure rates

Component name	Component ID	Internal failure mode	λ [1/FH]	Source
Ducted fan	DCTF	INOP	5.5096×10^{-6}	[23]
High-power electric bus	HPBS	INOP	10^{-5}	Assumed
Superconducting electric motor	SCEM	INOP	8.6775×10^{-6}	[23]
Cryocooling unit	CRCU	INOP	5×10^{-5}	Median of [24–26]
Motor control unit	MCNT	INOP	6.09×10^{-6}	[27]
Battery	BAT	FAIL	2.4885×10^{-4}	[23]
Engine	ENG	INOP	10^{-5}	Assumed

Table 6 Top-level event probabilities and number of MCS for different cryocooler arrangements for “OEI”

No. of cryocoolers	$P_{TLE}(1FH)$	No. of MCS
6	2.606197×10^{-4}	17
4	1.606197×10^{-4}	15
2	1.606162×10^{-4}	12
1	1.106162×10^{-4}	11

Table 7 Top-level event “AEI” probabilities and number of MCS for different cryocooler arrangements

No. of cryocoolers	$P_{TLE}(1FH)$	No. of MCS
6	7.999275×10^{-18}	30
4	7.046504×10^{-18}	21
2	6.796504×10^{-18}	20
1	5.006244×10^{-15}	12

failure. Reducing components thus leads to a reduction of minimum cut sets (corresponding to the logic of the very simple “Parts count” approach, see [3]). The same principle applies to the merging of the bus cryocoolers (a lower number of cryocoolers lead to a lower number of failure sources, hence lower number of MCS), but, as these do represent a redundancy already, with lower impact.

Table 7 shows the results for the top-level event “AEI”.

The same effect as observed for “OEI” leads to a reduction of TLE probability for the first three arrangements. The complete centralization of the cryocooling system, however, results in a drastic increase in the TLE probability. The CRCU has become a “single point of failure” for the entire electric propulsion part. Depending on the thrust requirements of the electric propulsion part, this could constitute a critical flight condition. Even if the probability of “AEI”, due to the modeling of the engines as complete independent components, is very low in all cases, the completely centralized system, under the assumptions made (failure of uncooled components within flight cycle) is not a suitable solution. The arrangement with two cryocoolers shall, therefore, be considered in more detail. The minimum cut sets are shown in Table 8.

The cryocooler failures have the largest impact on the TLE and contribute together to more than 60% of the TLE probability. Of course, the importance of the single-event MCS only depends on the chosen failure rate. As the input data, especially for the cryocooler, was chosen from a wider span of values in the literature (see Table 5), the value could, in reality, vary significantly, possibly even by one order of magnitude. Therefore, it is desirable to reduce the sensitivity of the TLE probability to the CRCU failure rate by design of the system architecture. Especially important are

Table 8 MCS for top-level event “OEI”, system architecture with two cryocoolers

Basic events	$P_{MCS}(1FH)$	I_{MCS} (%)
CRCU-001-INOP	5×10^{-5}	31.130105
CRCU-002-INOP	5×10^{-5}	31.130105
ENG-001-INOP	10^{-5}	6.226021
ENG-002-INOP	10^{-5}	6.226021
SCHEM-001-INOP	8.6775×10^{-6}	5.402630
SCHEM-002-INOP	8.6775×10^{-6}	5.402630
MCNT-001-INOP	6.09×10^{-6}	3.791647
MCNT-002-INOP	6.09×10^{-6}	3.791647
DCTF-001-INOP	5.5096×10^{-6}	3.430289
DCTF-002-INOP	5.5096×10^{-6}	3.430289
HPBS-001-INOP HPBS-002-INOP	10^{-10}	0.000062
BAT-001-FAIL BAT-002-FAIL	6.192617×10^{-8}	0.038555

Fig. 9 System architecture from [22]. Cryocoolers shown in green**Table 9** Top-level event “AEI” and “OEI” probabilities and number of MCS for the improved cryocooler arrangement

TLE	P_{TLE}	No. of MCS
OEI	6.061973×10^{-5}	13
AEI	6.593733×10^{-18}	14

the cryocoolers for the SCHEM and the MCNT. Introducing a redundancy for those could significantly decrease the TLE probability. Due to the “cross-feed” design of the HPBS arrangement, their coolant supply does not significantly impact system safety, as long as these are not supplied by the same cryocooler unit. With this knowledge, the architecture shown in Fig. 9 can be proposed.

In Table 9, the analysis results for this new configuration are shown. It can be seen that the “OEI” probability has

Table 10 Improved hybrid power train minimum cut sets for TLE “OEI”

Basic events	$P_{MCS}(1FH)$	$I_{MCS} (%)$
ENG-001-INOP	10^{-5}	16.496280
ENG-002-INOP	10^{-5}	16.496280
SCEM-001-INOP	8.6775×10^{-6}	14.314647
SCEM-002-INOP	8.6775×10^{-6}	14.314647
MCNT-001-INOP	6.09×10^{-6}	10.046235
MCNT-002-INOP	6.09×10^{-6}	10.046235
DCTF-001-INOP	5.5096×10^{-6}	9.088791
DCTF-002-INOP	5.5096×10^{-6}	9.088791
BAT-001-FAIL BAT-002-FAIL	6.192617×10^{-8}	0.102155
CRCU-002-INOP CRCU-001-INOP	2.5×10^{-9}	0.004124
CRCU-001-INOP HPBS-002-INOP	5×10^{-10}	0.000825
HPBS-001-INOP CRCU-002-INOP	5×10^{-10}	0.000825
HPBS-001-INOP HPBS-002-INOP	10^{-10}	0.000165

decreased significantly. The MCS shown in Table 10 reveal that the CRCU failures appear now only in second-order MCS with an accumulated importance less than 0.01%, leading to a drastically decreased sensitivity of the TLE probability to the failure rate of the CRCU.

7 Conclusion

In this publication, the use of model-based safety assessment for system architecture analysis in conceptual aircraft design was proposed. Criteria were named for a suitable method, among others the ability to fastly perform architecture trade studies and the reusability of behavior models. An approach using a Simulink system structure model was proposed. Unlike other MBSA approaches (e.g., AltaRica, HiP-HOPS), it is designed such that adding or removing redundant components does typically not require a revision of the component behavior description. This facilitates trade studies. The method, implemented as a Matlab program, was applied to an example propulsion system from a previous publication. It could be shown that the program can reveal safety weaknesses. After the basic system architecture and the component behavior have been modeled, the generation of alternative architectures by adding or removing components from or to the Simulink model takes only a few minutes. The computation of a system of this size and complexity also takes less than a minute on a standard desktop computer. Using qualitative and quantitative results from the trade study, an improved system architecture can be proposed. Even if the detailed implementation of the cross-connection of the cooling system has not been studied in detail, it can definitely be stated that the application of the program provides increased system insight without

Fig. 10 Improved architecture

requiring extensive efforts. It can be used to assess proposed architectures and can help to develop new ideas for architecture designs. We, therefore, believe that using MBSA in conceptual aircraft design studies can be useful for architecture generation and may yield better system designs. We also believe that the ability to alter the system architecture without changing the component behavior models makes our proposed approach more suitable for this task than other MBSA approaches, e.g., HiP-HOPS. However, with increasing component interaction complexity, architecture changes can not be implemented as easily. Also, the failure mode propagation direction is fixed. For larger systems, the current program implementation might also be too slow. This point could certainly be overcome by more efficient algorithms (Fig. 10).

As stated above, the use of a more expressive model, e.g., SysML, could be investigated. Also, a failure effect modeling approach could be investigated and compared with the employed failure logic modeling approach. To this point, an architecture optimization is not possible, as only safety, not mass or power consumption of a system can be modeled. Optimizing for safety alone would not yield a realistic system architecture. Currently, an improved system model is under development, enabling a holistic system analysis and making possible an automated system optimization.

Funding Open Access funding enabled and organized by Projekt DEAL. The work had no third-party funding.

Availability of data Not applicable.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Code availability Matlab code can be requested from the authors.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- European Commission: Flightpath 2050. Europe's Vision for Aviation—Maintaining Global Leadership & Serving Society's Needs. Report of the High-Level Group on Aviation Research. Publications Office of the European Union, Luxembourg (2011). <https://doi.org/10.2777/50266>
- Advisory Council for Aviation and Innovation in Europe: Strategic Research & Innovation Agenda. vol. 1. (2017). <https://www.acare4europe.org/sites/acadereurope.org/files/document/ACARE-Strategic-Research-Innovation-Volume-1.pdf>. Accessed 18 Nov 2021.
- Vesely, W.E., Goldberg, F.F., Roberts, N.H., Haasl, D.F.: Fault Tree Handbook. Office of Nuclear Regulatory Research, Washington D.C. (1981)
- Society of Automotive Engineers: ARP4761. Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment. SAE International, Warrendale (PA) (1996). <https://doi.org/10.4271/ARP4761>
- Lisagor, O: Failure logic modeling. A pragmatic approach. Ph.D. thesis (2010)
- Point, G., Rauzy, A.: AltaRica. Constraint automata as a description language. *J. Eur. Syst. Autom.* **33**(8–9), 1033–1052 (1999)
- Batteux, M., Tatiana Prosvirnova, T., Rauzy, A.: AltaRica 3.0 Language Specification. <https://www.openaltarica.fr/docs/AltaRica%203.0%20Language%20Specification%20-%20v1.1.pdf>. Accessed 14 Dec 2020
- Bieber, P., Blanquart, J. P., Durrieu, G., Lesens, D., Lucotte, J., Tardy, F., Turin, M., Seguin, C., Conquet, E.: Integration of formal fault analysis in ASSERT. Case studies and lessons learnt. Embedded Real Time Software and Systems ERTS (2008). <https://hal.archives-ouvertes.fr/hal-02270317>. Accessed 14 Dec 2020
- Bernard, R., Aubert, J.-J., Bieber, P., Merlini, C., Metge, S.: Experiments In model based safety analysis. *Flight Controls IFAC Proc.* **40**(6), 43–48 (2007)
- [https://www.openaltarica.fr/docs/The%20OpenAltaRica%20Platform%20-%20Example%20\(Simple\)CoolingSystem.pdf](https://www.openaltarica.fr/docs/The%20OpenAltaRica%20Platform%20-%20Example%20(Simple)CoolingSystem.pdf). Accessed 30 Jun 2020
- Kehren, C., Seguin, C., Bieber, P., Castel, C., Bougnol, C., Heckmann, J., Metge, S.: Advanced multi-system simulation capabilities with AltaRica (2004). <http://altarica-association.org/ressources/ARBib/KehrenEtAl2004-AdvancedSimulationCapabilities.pdf>. Accessed 14 Dec 2020
- Papadopoulos, Y., McDermid, J.A.: Hierarchically performed hazard origin and propagation studies. In: Felici, M., Kanoun, K., Pasquini, A. (eds.) *Computer Safety, Reliability and Security*. 18th international conference, proceedings/SAFECOMP, vol. 99, pp. 139–152. Springer, Berlin (1999)
- Papadopoulos, Y., Maruhn, M.: Model-based synthesis of fault trees from Matlab-Simulink models. In: *Institute of Electrical and Electronics Engineers: Proceedings of the International Conference on Dependable Systems and Networks*, pp. 77–82. IEEE Computer Society, Los Alamitos (CA) (2001). <https://doi.org/10.1109/DSN.2001.941384>
- Kaiser, B., Liggesmeyer, P., Mäkel, O.: A new component concept for fault trees. In: Lindsay, P., Cant, T. (eds.) *Proceedings of the 8th Australian Workshop on Safety Critical Systems and Software*, vol. 33, pp. 37–46. Canberra (2003)
- Kaiser, B., Schneider, D., Adler, R., Domis, D., Möhrle, F., Berres, A., Zeller, M., Höfig, K., Rothfelder, M.: Advances in component fault trees. In: Haugen, S., Barros, A., van Gulijk, C., Kongsvik, T., Vinnem, J. (eds.) *Safety and Reliability. Safe Societies in a Changing World*. Proceedings of ESREL 2018, June 17–21. CRC Press, London (2018)
- Ramich, V.: Teilautomatische Erstellung von Component-Fault-Trees aus Simulink-Modellen. Master Thesis (2014)
- Joshi, A., Heimdahl, M.P.E., Miller, S.P., Whalen, M.W.: Model-based safety analysis (2006). <https://shemesh.larc.nasa.gov/fm/papers/Joshi-CR-2006-213953-Model-Based-SA.pdf>. Accessed 14 Dec 2020
- Batteux, M., Prosvirnova, T., Rauzy, A.: From models of structures to structures of models. In: 2018 IEEE International Systems Engineering Symposium (ISSE), pp. 1–7 (2018)
- Nguyen, N., Mhenni, F., Choley, J.-Y.: AltaRica 3.0 code generation from SysML models. In: Haugen, S., Barros, A., van Gulijk, C., Kongsvik, T., Vinnem, J. (eds.) *Safety and Reliability. Safe Societies in a Changing World*. Proceedings of ESREL 2018, June 17–21. CRC Press, London (2018)
- Dugan, J.B., Bavuso, J.B., Boyd, M.A.: Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Trans. Reliab.* **41**(3), 363–377 (1992)
- Vesely, W.E., Dugan, J., Fragola, J., Minarick, J., III., Railsback, J.: *Fault Tree Handbook with Aerospace applications*. National Aeronautics and Space Administration, Washington D.C. (2002)
- Aigner, B., Nollmann, M., Stumpf, E.: Design of a hybrid electric propulsion system within a preliminary aircraft design software environment. In: *Luft- und Raumfahrt—Digitalisierung und Vernetzung*. Deutscher Luft- und Raumfahrtkongress (2018). <https://doi.org/10.25967/480153>
- Denson, W., Chandler, G., Crowell, W., Wanner, R.: *Nonelectronic Parts Reliability Data* (1991). <https://apps.dtic.mil/dtic/tr/fulltext/u2/a242083.pdf>. Accessed 14 Dec 2020
- Nisenoff, M., Edelsack, E.A.: US Navy programme in small cryocoolers. *Cryogenics* **23**(7), 353–356 (1983)
- Chen, X., Xia, M., Wang, L.Y., Gan, Z.H.: The reliability development of miniature stirling cryocoolers. In: Miller, S., Ross, R. (eds.) *Cryocoolers 18*. ICC Press, Boulder (2014)
- ter Brake, H., Wiegner, G.: Low-power cryocooler survey. *Cryogenics* **42**(11), 705–718 (2002)
- Department of Defense: MIL-HDBK-217F - Notice 2. Reliability Prediction of Electronic Equipment. Department of Defense, Washington D.C. (1991). <https://www.quantarion.com/wp-content/uploads/2014/09/MIL-HDBK-217F.pdf>. Accessed 18 Nov 2021.
- Masuda, T., Yumura, H., Watanabe, M., Takigawa, H., Ashibe, Y., Suzawa, C., Kato, T., Okura, K., Yamada, Y.O.-K., Hirose, M., Yatsuka, K., Sato, K., Isojima, S.: High-temperature superconducting cable technology and development trends (2004). <https://global-sei.com/technology/tr/bn59/pdf/59-2.pdf>. Accessed 14 Dec 2020