https://doi.org/10.1109/DASC55683.2022.9925883

Model-Based STPA: Enabling Safety Analysis Coverage Assessment with Formalization

Alexander Ahlbrecht
Institute of Flight Systems
German Aerospace Center (DLR)
Braunschweig, Germany
alexander.ahlbrecht@dlr.de

Umut Durak
Institute of Flight Systems
German Aerospace Center (DLR)
Braunschweig, Germany
umut.durak@dlr.de

***Abstract*—Urban Air Mobility introduces safety-related challenges for future avionics systems. The associated need for increased autonomy demands novel functions based on high-performance algorithms. To provide such functionality in future air vehicles of all sizes, the trend is towards centralized and powerful computing platforms. That turns avionics into a complex, integrated, and software-intensive aircraft system. Simultaneously, this increases the need for adapted safety analyses. The System-Theoretic Process Analysis is a promising approach to analyze the safety of software-intensive systems. It enables consideration of interaction and specification issues additional to component failures. However, even when using state-of-the-art analyses such as STPA, claiming the sufficiency of the safety analysis efforts is a challenging tasks for systems with ever-increasing complexity. To address this issue, this paper extends the coverage analysis concepts known from the software development to safety analyses. This is achieved with the utilization of failure graphs, i.e., formalized analysis summaries that can be automatically created during the safety analysis. Failure graphs have two advantages: they provide the possibility for visual analysis state indication and can be used to calculate various statistical metrics. Thereby, they allow to improve the knowledge about the depth, breadth, and state of the safety analysis. Both visual and statistical consideration complement each other to enhance the safety analysis coverage assessment for future avionic systems. To show all capabilities, the analysis of a flight assistance system serves as demonstrator.**

***Index Terms*—Safety, Coverage, Metrics, MBSE, STPA, SysML**

I. Introduction

Safety is of utmost importance for systems in the aviation domain, since accidents entail huge consequences in physical harm, economic value [1], and public perception of involved companies [2]. Claiming sufficiency of safety analysis efforts is a challenging endeavor for future systems with ever-increasing complexity. Essentially, the same issue applies as for proving the absence of bugs by testing, where the difficulty is well recognized [3]. Especially for avionics systems in segments such as Urban Air Mobility (UAM), novel function based on high performance algorithms and increasing software complexity [4] complicate the claim of sufficient safety consideration. For future systems, reducing the unknown and unsafe scenario space will be required. This approach is also targeted in the automotive domain with standards such as *Safety Of The Intended Functionality (SOTIF)* [5]. To achieve this goal, suitable safety analyses are needed. The *System-Theoretic Process Analysis (STPA)* with its system-theoretic analysis approach can contribute in this area, as highlighted by the recent recommended practice SAE J3187 [6]. It allows identifying interaction and specification issues additional to component failures [7].

Even when using state-of-the-art analyses such as the STPA, identification of all potential accident causes is not guaranteed. In addition, the rising complexity also increases certification difficulty. Hence, safety engineers require assistance for efficient investigation and means to build sound safety arguments. To cope with these challenges, adapted and more formal practices to certify future safety-critical systems are required [8]. Both analysis execution and the certification processes could benefit from improved coverage assessments. That is why this paper targets to improve coverage assessment of safety analyses by providing statistical and visual indications that enable assistance during multiple phases of the safety analysis.

In the software-related context of DO-178C [9], *Coverage Analysis* is defined as "the process of determining the degree to which a proposed software verification process activity satisfies its objective". In this paper, we extend the coverage concept to the coverage assessment of safety analyses. Therefore, a methodology will be presented that enables the novel concept of *Safety Analysis Coverage Assessment (SACA)* for the STPA. This is achieved by applying a formalized and model-based STPA version on a system architecture modeled with the *Systems Modeling Language (SysML)* [10]. Previous work already demonstrated how a formalized STPA can be integrated into SysML [11], [12]. By slightly extending this approach, we demonstrate automated extraction of *Failure Graphs (FG)*, i.e., formalized analysis summaries, from the model-based safety analysis.

Definition 1. *Safety Analysis Coverage Assessment (SACA)* describes the process of determining the degree to which a safety analysis process activity satisfies its objective.

FGs provide the means for automated visualization of identified accident causes. Visualizing FGs in combination with the corresponding analysis model gives a quick overview of the safety analysis status. This visualization can even

incorporate information about which parts of the FG were assessed to be properly mitigated or approved by an authorized agency. Beyond visual analysis, FGs can also be exported and algorithmically analyzed for various properties. For instance, statistical metrics can be calculated, assisting to identify weaknesses of the analyst team towards the detection of specific accident causes. To demonstrate the methodology, a logical architecture model of a *Flight Assistance System* is used in this paper. Overall, the derivation and application of FGs shows promising signs to improve SACA. Both visual and algorithmic functionality complement each other to enhance the SACA for complex avionic systems.

The following sections will elaborate the FG concept and application in more detail. To introduce the topic, background information will be provided in Section II. Using this baseline, the FG concept will be outlined in Section III and demonstrated in Section IV. Finally, the concept is discussed in Section V and summarized in Section VI.

II. Background

In order to provide a better understanding for the FG concept, background knowledge will be introduced in this section. This includes a short introduction to existing SACA activities, the formalized STPA foundation, and a brief description of the use case that will be used to demonstrate the methodology within this paper.

A. Safety Analysis Coverage Assessment

Typically, guidance in terms of the aspects to be covered during safety analysis is provided by standards of the corresponding domains. Software-intensive avionic systems [13] will mainly need to consider guidance of documents such as DO-178C [9] and DO-297 [14]. Similarly, the STPA handbook itself proposes accident cause categories that should be evaluated during analysis [7]. This failure categorization can also be found in practical applications of the STPA [15], [16]. Other than the classical categorization of accident causes, it is argued that an important part of the safety analysis is the assurance of a well-defined analysis model [17]. This is especially important in a analysis such as STPA, where the model-based control structure is the centerpiece for the analysis execution. In this direction, [17] proposes ways to establish confidence in the safety assessment by improving the analysis model. Reviewing the STPA, [18] argues that the systematic top-down approach of STPA already lays a good foundation to identify many accident causes. However, in current literature, there is a lack of automated and scalable support for SACA. We argue that enabling such an automated SACA support will be more and more important to assist in the systematic analysis of systems with ever-increasing complexity. One way of achieving scalable SACA support could be the combination of *Model-Based Systems Engineering (MBSE)* with safety analyses. For example, [19] proposes to use SysML tables to support quick identification of element coverage for analyses such as *Failure Mode And Effects Analysis (FMEA)*.

B. Formalized Model-Based STPA

STPA is a rather novel safety analysis based on system-theoretic principles and a control-based view of systems and their interactions. The premise of the STPA is to identify not only component-related failures, but also failures based on interactions and specifications [20]. This analysis focus is particular interesting for complex, software-intensive systems [7]. During previous work, a formalized STPA version, first introduced in [21], was integrated in a MBSE environment [11]. Formalization is achieved by using SysML stereotypes with precisely defined relationships that can be mapped to every part of the STPA. This not only allows to automate creating parts of the analysis, but also introduces the ability to automate verification and validation activities. A sample application of the analysis was published in [12]. In the SysML-based analysis, the main steps of the STPA are recreated as outlined in Fig. 1. In the first step, the analysis purpose has to be specified. This includes the definition of hazards and losses to be considered, as well as the definition of the system boundary. In the second step, the control structure of the system to analyze has to be modeled. The control structure is the centerpiece of the STPA and lays the foundation for the analysis execution. Important aspects of the control structure are the interacting systems called controllers, their corresponding *Control Actions (CA)*, and the associated *Process Variables (PV)*. In the third step, the control structure helps in combination with guidewords to identify potential *Unsafe Control Actions (UCA)*. In the final step, *Loss Scenarios (LS)* and their *Causal Factors (CF)* are derived for each UCA. After identification of LSs and their underlying CFs, mitigating requirements can be derived and used to improve the overall system safety.

Definition 2. *Control Actions (CA)* are actions provided by a controller to achieve a change in the controlled process. Set and element: $\mathcal{CA}, ca$.

Definition 3. *Process Variables (PV)* are variables that represent important properties about the controlled process or other relevant system- and environment-related aspects. They provide valuable insight about the important aspects of each controller. Set and element: $\mathcal{PV}, pv$.

Definition 4. *Unsafe Control Actions (UCA)* are CAs that, in a particular context and worst-case environment, will lead to a hazard [7]. Set and element: $\mathcal{UCA}, uca$.

Definition 5. *Loss Scenarios (LS)* describe the *Causal Factors (CF)* leading to the identified UCAs and ultimately to hazards [7]. Set and element: $\mathcal{LS}, \ell s$.

Definition 6. *Causal Factors (CF)* are underlying factors that in a particular context lead to a LS. Set and element: $\mathcal{CF}, cf$.

C. Flight Assistance System Use Case

Assuring non-functional properties for embedded systems is an important part of certification in safety-critical sectors [9], [14]. To assist in this process, an X-By-Construction toolchain

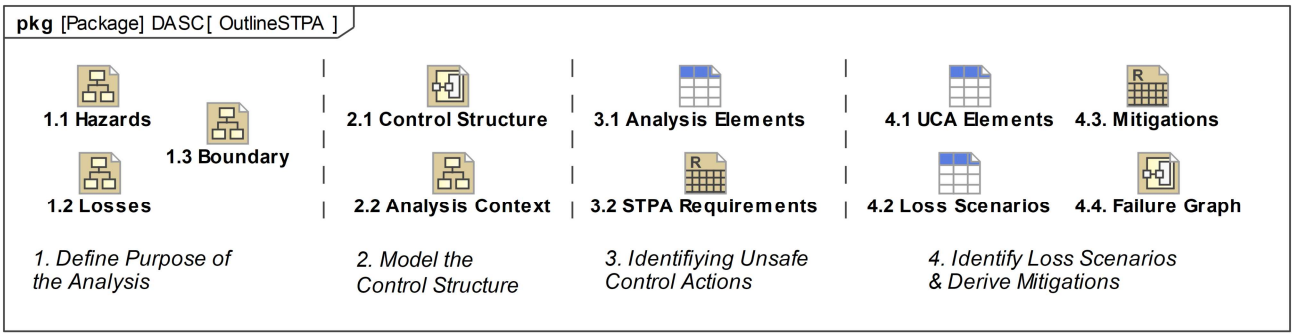

Fig. 1: STPA Outline with Corresponding SysML Diagrams

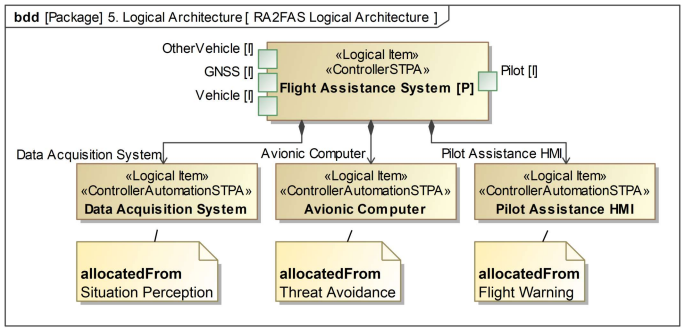

Fig. 2: Logical Assistance System Architecture

is developed in the European XANDAR project[1] [22]. The project focuses on enforcing non-functional properties in the design process of embedded systems. To validate the toolchain, two use cases will be utilized. One of the use cases is a *Flight Assistance System*, developed by the German Aerospace Center (DLR). Flight assistance will be increasingly important for pilots in urban air scenarios since pilots have to execute many tasks while monitoring multiple functionality at the same time. Airborne vehicles will have to cope with higher traffic densities while also avoiding tall structures. Hence, desired assistance includes advisories regarding traffic and terrain. Further, navigational advisories are also very important to guide the vehicle to the correct destination. To enable such functionality for UAM vehicles, a suitable assistance system needs to be developed. The corresponding high-level logical architecture is shown in Fig. 2. The main components of the architecture are the following: a *Data Acquisition System* enabling perception of the situation, an *Avionic Computer* providing all computational functionality, and a *Pilot Assistance HMI* that notifies the pilot in case of dangerous situations. Within this paper, the logical use case architecture will be used to demonstrate the FG concept in Section IV.

[1]https://xandar-project.eu/

III. Proposing Failure Graphs

A. Failure Graph Concept

As introduced, the trend towards complex and software-intensive systems enforces new challenges on the development processes and certification. This strengthens the need for a systematic safety assessment that enables SACA support during analysis execution and certification. Consequently, the FG concept will be introduced, allowing for various assistance.

Definition 7. *Failure Graphs (FG)* are formalized safety analysis summaries. They are created for every CA that was analyzed with the STPA and inherit the structure of the STPA execution. Thereby, they contain the entire accident cause information as well as the corresponding interconnection identified within the analysis. Set and element: $\mathcal{FG}, fg$.

In terms of processing, FGs provide the ability to extract important SACA-related metrics. Metrics will be especially essential for the systematic assessment of complex systems, where it is not obvious when the analysis is sufficiently executed. In such cases, metrics might be essential to create a safety case to certify future systems.

Looking at the safety analysis execution, a tight coupling between the development and analysis activities would allow various benefits. For instance, a combination of development activities and safety analysis execution would enforce the usage of the same system state for development and safety aspects. Considering the FG concept, the integration would enable a visualization of the identified accident cause relations in combination with the development model. These visualizations can be used to facilitate interdisciplinary discussions between safety and development engineers that would ultimately lead to improved system safety.

B. Failure Graph Structure

After explaining the concept behind FGs, a more concrete implementation structure will be presented. The idea was to build FGs upon a formal STPA structure. To outline how a FG will be extracted, an overview image of the underlying relationships is provided in Fig. 3. FGs shall be created for every analyzed CA. In Fig. 3, it is shown how every CA can be linked to one or more unsafe counterparts identified during the

initial analysis. Every identified UCA can be caused by one or multiple LSs, which again have their own CFs. Exemplary causes are inadequate incoming CAs, inadequate PVs or even multi-causal combinations of these factors. Considering these connections, a FG covers a complete analysis summary for one specific CA, including all related unsafe scenarios and their corresponding CFs. Due to the fact that a CA can again be a CF for a failure in a LS, a hierarchical linking of FGs can be established. For instance, in Fig. 3., fg_{ca_0} is also linked to fg_{ca_1} because of ca_1 being a CF identified for a ca_0 related LS as highlighted in red. With this connection, the depth of the FG is extended by one level in the example. This linking will be from now on referred to as depth-level:

Definition 8. *Depth-Level* refers to the level of repeated linking of FGs. Hierarchic linking is established when a FG is linked to another FG through a CF (e.g. a CA).

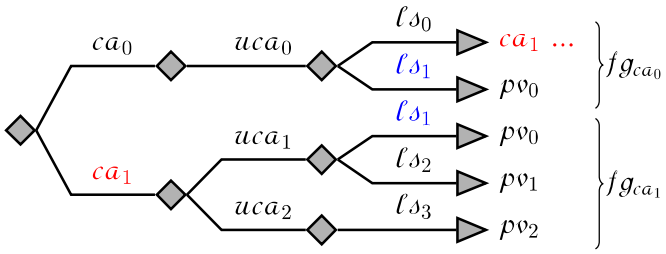

Fig. 3: Formalized Safety Analysis Structure

C. Foundation & Creation of Failure Graphs

To make the automatic creation and application of FGs possible, a sufficiently formalized analysis structure is required. Without this being the case, it is not possible to automate the creation of FG analysis summaries. The prerequisites can be met with the usage and slight extension of the formalized and model-based version of the STPA previously introduced in Section II-B. To create a FG automatically, every analysis step has to be linked and the results formalized. This was not the case for the LS description in the previous analysis implementation [11], [12], where only textual rationals were used to document the LSs in the UCA elements. To formalize the LSs, the textual rationals were extracted from the UCA elements and a separate LS element type was created. This new LS element type includes all required properties and thereby enables a linking of related model elements as visible in Fig. 4. LS properties that should be defined are: *Causal Factor Classification*, *Causal Factor Source*, *Causal Factor*, *Loss*, *Mitigations*, and *Approval*. Each of these values provides the ability to establish a link to one or more corresponding model elements (systems, interactions and requirements). In addition, the textual description of the LS is included in the form of the *Loss Scenario Description*. In the LS element, multiple CFs can be linked simultaneously. This consideration of multi-causal accident causes is an important property of the STPA and distinguishes it from bottom-up safety analyses such

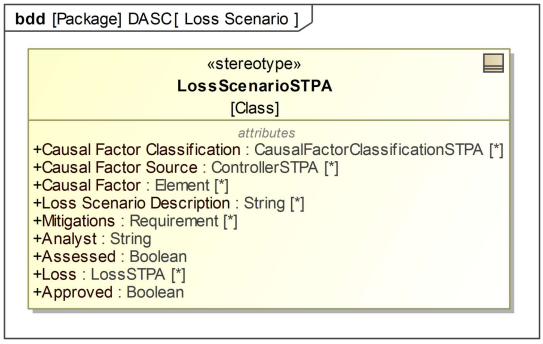

Fig. 4: Loss Scenario Element

as FMEA. Without the ability to consider multi-causal issues, non-trivial LSs can be missed [7].

In summary, the previously called rationale was extracted from the UCA elements and formalized in a separate LS element. In combination with the already established formalized and model-based STPA of Section II-B, this provides a sufficient basis to implement the FG concept. Finally, the split-up also enables the reuse of identified LSs in multiple UCA elements. A reuse of LSs can be useful to link accident causes that are relevant to more than one UCA. This removes repeated entries of the same failure cause and thereby reduces ambiguity. In addition, this can improve analysis efficiency in some cases, where the failure-related causation is very similar. One example could be that a power-related LS similarly effects multiple outputs of the *Avionic Computer*. In Fig. 3, the repeated linking of the same ℓs_1 element is highlighted in blue to exemplify such a structural relation.

Definition 9. *LS Repeat Rate* describes the ratio of LSs that are used more than once in a FG over the number of overall identified LSs linked to the FG.

Since FGs are based on the formal relationships shown in Fig. 3, they can be easily extracted after executing the model-based STPA. An exemplary algorithm layout to create FGs is provided in Alg. 1. Using the existing $\mathcal{UCA}$, $\mathcal{LS}$, and $\mathcal{CA}$ sets of the executed analysis, the overarching $\mathcal{FG}$ set can be created. The overarching $\mathcal{FG}$ set includes a fg_{ca} for every analyzed ca element.

D. Failure Graph Metrics & Visualization

After the analysis execution, the created FGs can be used to support the SACA. This support is possible in multiple ways. Metrics can provide insight into the depth and breadth of the analysis execution, and also help to identify potential weak spots. More precisely, they can give an overview of how deep every CA was analyzed and how many LSs were identified. Similarly, the type of the identified CFs can give an insight into the potential biases of the analysis execution. For instance, if only causes of one specific type (e.g. environmental or component failure) were identified, other categories of accident causes might have been overlooked. A reason for this could be that the expertise of the safety engineer executing

Algorithm 1: Creating Failure Graphs

Input: $\mathcal{UCA}, \mathcal{LS}, \mathcal{CA}$
Output: $\mathcal{FG}$
// create a fg_{ca} for each ca
forall ca *in* $\mathcal{CA}$ **do**
  $fg_{ca} = \{\}$
  // uca connects ca to ℓs
  forall uca *in* $\mathcal{UCA}$ **do**
    // check if ca is cf in uca
    if ca *is* cf *in* uca **then**
      // add uca info to fg_{ca}
      $fg_{ca} = fg_{ca} \cup uca$
      // loop linked ℓs of uca
      forall ℓs_{uca} *in* $\mathcal{LS}_{uca}$ **do**
        // add ℓs_{uca} info to fg
        $fg_{ca} = fg_{ca} \cup \ell s_{uca}$
  // add fg_{ca} to overall $\mathcal{FG}$ set
  $\mathcal{FG} = \mathcal{FG} \cup fg_{ca}$
return $\mathcal{FG}$
// if cause of fg_{ca} is other fg_{ca},
// recursive linking is possible

the analysis lies in a specific field. The consideration of irregularities in causal categories relates metrics to SACA considerations described in Section II-A.

In this paper, the following metrics will be used to demonstrate how FGs can be leveraged to improve the SACA:

- Bias and weakness indicating metrics:
  - Classifications of LS causes per CA
  - Classifications of LS causes per UCA
- Analysis state indicating metrics:
  - Percentage of approved LSs
  - Percentage of mitigated LSs
- Depth and breadth indicating metrics:
  - Depth of FG per CA
  - Number of UCAs per CA
  - Number of LSs per CA
  - Repeat rate of LSs per CA

In addition to the extraction of metrics, FGs enable visual SACA support. This is especially helpful in combination with the corresponding control structure diagram. To support the SACA of specific CAs, the corresponding FG can be created and visualized in parallel to the safety analysis execution. Thereby, a simultaneous SACA is enabled. Furthermore, it is possible to integrate specific feedback about the status of the analysis. This is possible since the formalized analysis style already includes properties to link mitigations and indicate approvals in the LS elements of Fig. 4. Hence, when using these entries, the analysis state is always known and can be visualized. For a FG visualization in the system model, the use of a traffic light principle is proposed to color the background of elements in the following ways. Red highlights elements that were identified as accident cause in one or more LSs. Orange highlights elements that were not only identified, but also mitigated in all related LSs. Green highlights elements that were identified and approved in all related LSs. Creating this visualization is always viewed in context of a corresponding control structure diagram. Therefore, the state $\mathcal{s} = \{none, identified, mitigated, approved\}$ has to be evaluated and visualized for every element of the diagram. This can be achieved with an algorithm such as Alg. 2, where for each element e of the diagram's element set $\mathcal{E}$, the lowest state in any related LS is identified. In the following application Section IV, the visual and algorithmic features of FGs will be demonstrated.

Algorithm 2: Visualizing a Failure Graph

Input: $fg_{ca}, \mathcal{E}$
Output: $\mathcal{E}_{\mathcal{s}}$
// $\mathcal{E}_{\mathcal{s}}$ collects all element states $e_{\mathcal{s}}$
$\mathcal{E}_{\mathcal{s}} = \{\}$
// Evalute state for all e *in* $\mathcal{E}$
forall e *in* $\mathcal{E}$ **do**
  // $e_{\mathcal{s}}$ defines state of e
  $e_{\mathcal{s}} = none$
  // Evaluate if element e is cause
  // in any fg related to action ca
  if e *is* cf *in* fg_{ca} **then**
    // Find lowest state $\mathcal{s}$ of e
    if e *is approved*, $\forall\ \ell s$ *in* fg_{ca} **then**
      $e_{\mathcal{s}} = approved$
    else if $e \geq mitigated$, $\forall\ \ell s$ *in* fg_{ca} **then**
      $e_{\mathcal{s}} = mitigated$
    else
      $e_{\mathcal{s}} = identified$
  $\mathcal{E}_{\mathcal{s}} = \mathcal{E}_{\mathcal{s}} \cup e_{\mathcal{s}}$
return $\mathcal{E}_{\mathcal{s}}$

IV. Applying Failure Graphs

In this section, the application of FGs will be demonstrated using the *Flight Assistance System* introduced in Section II-C. Considering the number of steps used to execute the STPA as outlined in Fig. 1, only an excerpt can be shown in this paper. Hence, focus was directed towards diagrams containing the output of the analysis execution. Namely, diagrams 4.1 to 4.4 of Fig. 1 are used to provide the basis for the FG demonstration. Due to the length limitations, diagrams and tables can only be partly displayed.

A. Assistance System STPA Execution

The *Flight Assistance System's* high-level control structure is displayed in Fig. 5. For now, the background coloring of the diagram parts can be ignored, since this feature will be covered at the end of this section. In addition to Fig. 2, it is now visible how the logical system architecture parts and their inner and outer CAs build up the control structure. Additionally, all controllers are enriched with information regarding internal

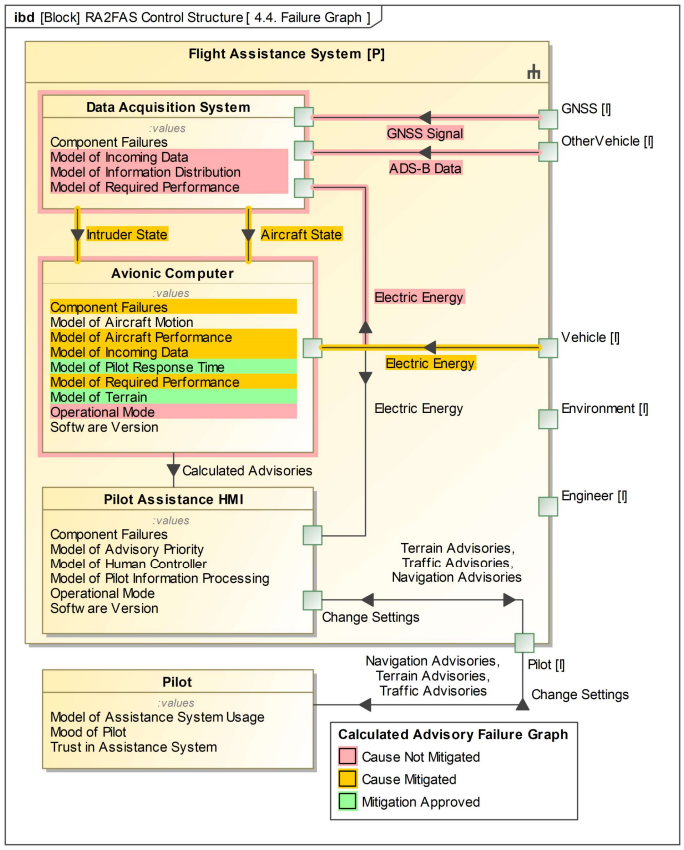

Fig. 5: Control Structure Diagram & Graph Visualization

process-related variables. For example, the *Model of Terrain* is an important PV of the *Avionic Computer* that has influence on calculating terrain, but also navigation-related advisories. In the following parts of this section, focus will be directed towards the *Calculated Advisories* provided by the *Avionic Computer* to the *Pilot Assistance HMI*.

Following the STPA execution of Fig. 1, the second part of *Identifying Unsafe Control Actions* lays the foundation for *Identifying & Mitigating Causal Factors*. An exemplary UCA entry is shown in Fig. 6, stating that it was identified to be dangerous, when the *Avionic Computer* applies unnecessary *Calculated Advisories* to the *Pilot Assistance HMI*. To support the engineer in the identification of LSs, information such as incoming *Command Variables*, *Feedback Variables* as well as internal *Process Variables* of the *Avionic Computer* are automatically provided. Using this information, LSs were derived and linked in the UCA element. Two of the five LSs linked in Fig. 6 are again shown in more detail in Fig. 7, where the newly introduced and formalized structure is applied. In the formal LS elements, a linking to other model elements is established with the *Causal Factor Source*, *Causal Factor*, *Loss*, and *Mitigations* entries. Additionally, a textual *Loss Scenario Description* is provided, as well as the ability to state sufficient compliance with the *Approved* entry. The first LS of Fig. 7 describes that the *Model of Terrain* used to generate advisories always needs to be up-to-date. Otherwise, inaccurate advisories can be generated as described in the *Loss Scenario Description*. In the worst case, unfitting advisories could contribute to a *Near Mid Air Collision (NMAC)* or *Controlled Flight Into Terrain (CFIT)*. As *Causal Factor Source*, the *Avionic Computer* itself is specified while the *Model of Terrain* is linked as corresponding *Causal Factor*. To mitigate the identified causation, a requirement was derived in Fig. 8 and linked in the LS. For this LS, it was exemplary shown with the *Approved* entry, how a certification agency could state sufficiency of introduced mitigations.

B. Failure Graph Interpretation

Even though the analysis process was only shown in more detail for the *Calculated Advisories* of Fig. 5, it should be executed for all CAs of the control structure. This includes CAs such as: *Traffic Advisories*, *Terrain Advisories*, *Change Settings*, *Intruder State*, *Aircraft State*. All of the information collected through the corresponding STPA execution can be used to automatically create FGs for each of these CAs. A segment of the FG created for the *Calculated Advisories* is shown in Fig. 9. It is structured according to the formal FG relationships presented earlier in Fig. 3. This relation is highlighted by the information annotated using curly braces. It is important to note that only the first depth-level of the FG is shown in Fig. 9. The depth-level refers to hierarchic linking of FGs, which can happen when CAs are the cause identified within a FG as described at the end of Section III-B. Similarly, for all metrics presented in the following, only the first depth-level will be used.

In TABLE I, extracted metrics are displayed that help to understand the depth, breadth and analysis state. For instance, the distinct count of UCAs and LSs helps to quickly get an idea of the extent to which the analysis was executed. When comparing the counts of the *Calculated Advisories* with the counts of the *Change Settings* CA, a big disparity regarding the identified LSs can be observed. This indicates that a lot more effort was placed on analyzing the *Calculated Advisories*. Similar observations can be extracted for all analyzed CAs. Additionally, the UCA to LS count ratio might be an interesting indicator in some instances. When very similar CAs show huge differences in the number of UCAs, LSs or even their ratio, it might be necessary to look into the reasoning. This could help to identify missing UCAs or LS reasons, leading to a more exhaustive analysis.

Since LSs can be reused in multiple UCAs as explained in the end of Section III-C, an interesting indicator is the LS repeat rate. The repeat rate in this paper shows the percentage of individual LSs that are used in more than one UCA. This is only one way of many to calculate the repeat rate and could be adapted if preferred. In TABLE I, we can see that the LS repeat rate is very high for the *Terrain Advisories*. This could lead to the false impression of a further advanced analysis state because every UCA might have many LSs linked to it. Furthermore, a correlation between the number of UCAs and the repeat rate is visible. This is reasonable, since the benefit

#	Name	UCA Description	Hazard Reference	Command Variables	Feedback Variables	Process Variables	Loss Scenarios
1	UCA 14	The [Avionic Computer] applies [Calculated Advisories] to the [Pilot Assistance HMI] when [AdvisoryNeed] is in state [Not Needed].	RA2FAS Contributes to a CFIT RA2FAS Contributes to a NMAC RA2FAS Increases Pilot Workload	Aircraft State Electric Energy Environmenta Influences Intruder State Update SW		Underlying Components Model of Aircraft Motion Model of Aircraft Performance Model of Required Performance Model of Incoming Data Model of Pilot Response Time Model of Terrain Operational Mode Software Version	InadequateTerrainModelLeadsToNc OtherAdvisory IncorrectAdvisoriesDueToInadequa teAircraftState IncorrectModelOfAircraftPerforman ceLeadsToUnnecessaryAdvisories IncorrectModelOfPilotDelayLeadsTc UnnecessaryAdvisories IncorrectAdvisoriesDueToInadequa teIntruderState

Fig. 6: Identified Unsafe Control Action with Provided Model Information

#	Name	Causal Factor Classification	Causal Factor Source	Causal Factor	Loss Scenario Description	Loss	Mitigations	Approved
1	InadequateTerrain ModelLeadsToInco rrectAdvisory	Process Model	Avionic Computer	Model of Terrain	An inadequate terrain model leads to a incorrect advisory beeing generated. This leads to a contribution to a CFIT or NMAC of the aircraft.	Loss of Aircraft	209.2.3 Up-to-Date Terrain Data	true
2	IncorrectAdvisorie sDueToInadequat eAircraftState	Input	Data Acquisition System	Aircraft State	An inadequate aircraft state input results in the incorrect advisories being calculated. This leads to the assistance system contributing to a CFIT or NMAC of the aircraft.	Loss of Aircraft	209.1.2 Monitor Aircraft State Input	false

Fig. 7: Identified Loss Scenarios with Formal Linkage to Model

TABLE I: Extracted Depth, Breadth and Analysis State Metrics for Analyzed CA

Analyzed CA	Depth	UCA Count	LS Count	LS Repeat Rate	Mitigated	Approved
Calculated Advisories	2	3	14	21%	86%	21%
Traffic Advisories	3	6	13	31%	8%	0%
Terrain Advisories	3	6	12	75%	8%	0%
Change Settings	3	4	3	33%	0%	0%
Intruder State	1	3	10	0%	0%	0%
Aircraft State	1	3	3	0%	0%	0%

#	Name	Text
1	209.2.3 Up-to-Date Terrain Data	An up-to-date terrain model shall be used to calculate the related advisories.
2	209.1.2 Monitor Aircraft State Input	The aircraft state input to the avionic computer shall be monitored to ensure only reasonable states being transferred.

Fig. 8: Mitigating Requirements

for reuse increases with a higher number of potentially similar UCAs.

In terms of the analysis state, the counts only give a partial overview. To get a broader idea of the analysis state, it is also relevant to look at the percentage of mitigated and approved LSs. TABLE I shows that only the *Calculated Advisories* contain a high rate in terms of mitigated and approved LSs. This rightfully indicates that currently, the analysis is only further advanced in regard to the *Calculated Advisories*. Initial analysis focus in the project was directed towards the *Calculated Advisories*, because the *Avionic Computer* will be the main target for the use case driven evaluation of the XANDAR toolchain introduced in Section II-C.

The final metric-related examples introduced in this paper provide insight into bias and potential weaknesses of the executed analysis. Within the LSs, the ability to specify a main classification of the CF was introduced. When deliberately filling out the corresponding classification, valuable insight can be gained. In TABLE II, the classifications are shown for every analyzed CA. For the *Calculated Advisories*, the most LS causes were attributed to process-related issues. On the other side, no causes were attributed to human-related issues. Even if the *Avionic Computer* does not take direct input of the *Pilot*, still a bunch of human-related LSs affecting the *Avionic Computer* could be present. Exemplary, the *Change Settings* CA from the *Pilot* might also propagate through the *Pilot Assistance HMI* to the *Avionic Computer* and cause a selection of an inappropriate operational mode. Another example would be the maintenance crew that has the ability to cause malfunction of the *Avionic Computer*, when maintenance is not rightfully executed. Here, a reason for not identifying the potential issues could be the missing connection to the human-operators in the control structure of Fig. 5. As visible with this example, the classification amount allows identifying potential weaknesses of the analysis. In general, an interpretation of the metrics can be executed in multiple hierarchical levels. For instance, on a overarching analysis level, the summed number per cause classification can help to identify biases. In TABLE II, a substantial number of causes were attributed to process-related issues. Hence, it is possible to question this amount and

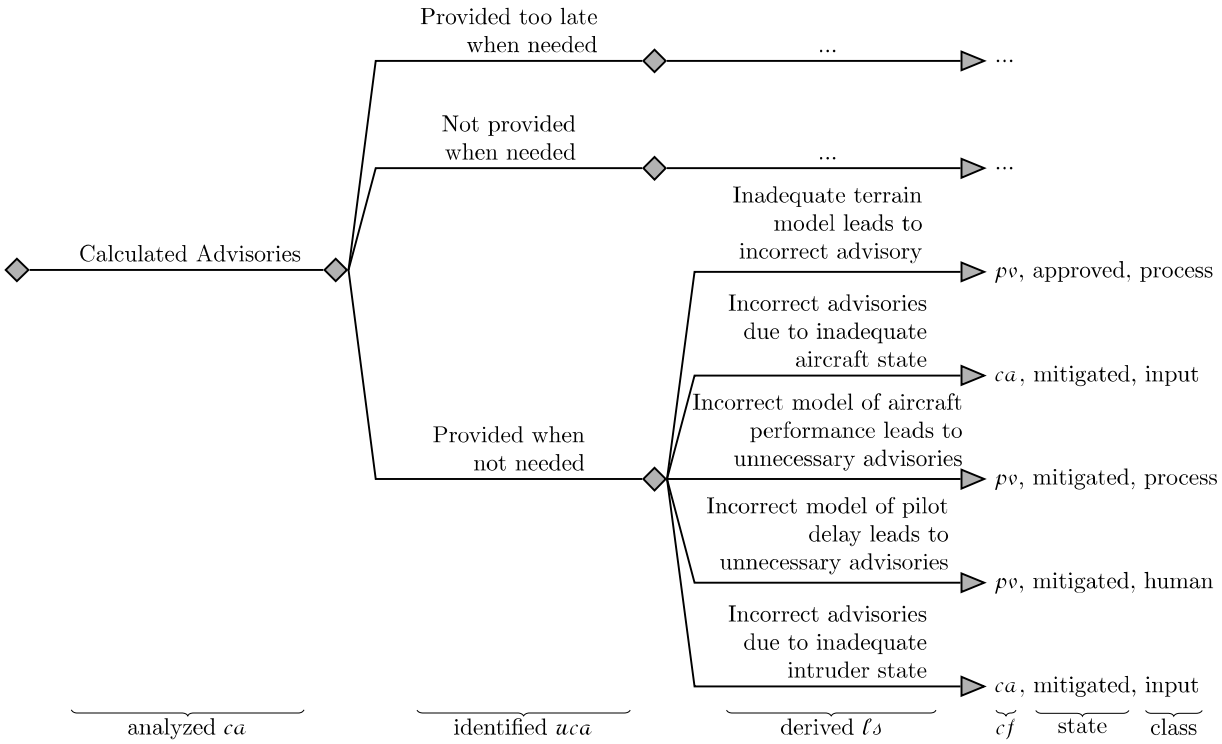

Fig. 9: Failure Graph Extract of Calculated Advisory (1st Depth-Level)

identify biases of the analyst team. Biases could be introduced by the analysis process, the analyst's experience, and many other factors. If biases and weaknesses are identified after thorough consideration of the classification values, supportive measures can be introduced for long-term improvement of the safety assessment. Exemplary measures are the training of analysts in areas of identified weaknesses, the staff extension with corresponding experts or the purposeful adjustment of the analysis process.

Additional to the algorithmic analysis of FGs, the advantages of introducing visual SACA support will be demonstrated in the following. As explained at the end of Section III-D, a visual support is preferably established in combination with the control structure used for the analysis. Hence, in Fig. 5, a FG is visualized on top of the control structure diagram. Again, the FG for the *Calculated Advisories* is used for demonstration purposes. Unlike the previous FG cut-off after the first depth-level, the visualization benefits from using the full depth-level. All calculated depth-levels of the analyzed CAs are listed in TABLE I. Looking at the *Calculated Advisories* FG visualization in Fig. 5, the two depth-levels of analysis are emphasized through the background coloring. Since both *Intruder State* and *Aircraft State* were found to be a potential LS cause for an inadequate *Calculated Advisories* execution, also their respective LS causes have to be considered and mitigated to ensure correct functionality. Since no feedback or input is currently envisioned from the *Pilot Assistance HMI* to the *Avionic Computer*, no direct causal connection was identified. This is something that should be investigated and adjusted accordingly.

Not only the causal relationships of component CAs can be emphasized by visualizing FGs. Another important aspect is the coloring of control structure parts. Through the traffic light principle, the analysis state can be quickly observed. All parts that do not include any coloring were not identified as a part of the *Calculated Advisories* FG. In Fig. 5, this among other includes the *Model of Aircraft Motion*, which could surely be a CF for an inadequate advisory. Hence, parts that were not even considered once can be quickly identified. Furthermore, the traffic light coloring scheme underlines the analysis state. It is visible through the orange and green coloring that most of the identified accident causes of the *Avionic Computer* were at least mitigated and some even approved. This is not yet the case for the accident causes of the *Data Acquisition System*. Overall, during the analysis, the visualization proved to be a very valuable tool to identify missing analysis parts in multiple facets. In some cases, the LSs were even identified but not linked to the corresponding UCAs. Here, the missing coloring led to a quick identification of the missing establishment of formal links.

Using the SACA supporting concepts in combination with the STPA as presented in this paper, a more exhaustive safety analysis was enabled for the *Flight Assistance System* used in the XANDAR project. As a result, valuable safety-related requirements for the use case itself, but also the toolchain that is developed within the project were derived. Additional to

TABLE II: Extracted Loss Scenario Classification Count for Analyzed CAs

Analyzed CA	Input	Algorithm	Environment	Process	Component	Transfer	Human
Calculated Advisories	2	1	1	7	2	1	0
Traffic Advisories	0	3	2	3	1	2	2
Terrain Advisories	0	3	2	2	1	2	2
Change Settings	0	0	0	0	0	0	3
Intruder State	0	2	1	3	1	3	0
Aircraft State	0	0	0	0	1	2	0
Summation	2	9	6	15	6	10	7

the requirements, safety-guided architecture extensions were introduced through the results.

V. Discussion

Within this paper, the formalization of the previously introduced model-based STPA implementation was extended. These extensions allow novel functionality such as the SACA support by using automatically created FGs. With this example, it is visible that even a small extension of the formalization enables novel automated functionality. Since formalization in a model-based context can be easily established by linking model elements, there should not be a big barrier for practical application. Safety analysis formalization not only allows SACA support as presented in this paper, but enables a lot of helpful features. Other automatable examples include the creation of safety artifacts, the execution of verification activities [11] and the tracing of design changes towards safety artifacts [23]. It is therefore possible that the FG analysis summaries can enable a lot of functionality additional to the SACA-related support that was presented in this paper. For the efficient development and safety assessment of complex future systems, we would argue that both formalization and resulting automation will become increasingly necessary.

However, there are also some areas that require careful consideration and training when working with automation. Looking at the current visualization of FGs, it has to be considered that it is not represented how thorough each of the control structure parts was analyzed. Therefore, two diagram elements can appear in the same coloring even if one was approved in one and the other approved in hundreds of LSs. Reason being that the visualization only marks the lowest analysis state of all related LSs. Thus, conclusions should only be carefully derived from automated functionality such as visualization, which requires training. Thinking about the approval process, this information can be still really valuable. If the agency wants to make sure that every LS-related to one CA was approved, all parts of the related control structure have to appear with a green background coloring. To extend the SACA-related knowledge gained from the visualization with additional information about the depth of the analysis execution, the introduced metrics can be used. This highlights the potential synergy of combining the FG visualization and the related metrics to improve the overall SACA. For instance, the background coloring could be adjusted in terms of the darkness depending on the number of LSs calculated by the metrics.

After considering limitations of the visualization, also the limitations of metrics have to be discussed. Metrics can indicate analysis weaknesses, but are not the panacea. They are not able to provide full insight into the quality of the analysis execution and results. Hence, interpretation always requires caution. In a company or for a certification agency, they should not be used as a sole indicator for the analysis quality. This could lead to deliberate manipulation to adhere to company or agency-related requirements. Additionally, metrics always need to be interpreted in context of the analyzed system, since the classification of accident causes can be highly depended on the corresponding context. For instance, if a system under analysis directly involves a human operator, the likelihood of human-related accident causes should be a lot higher than if no humans are directly involved. Similarly, the classification into groups can be ambiguous sometimes and should be clearly defined beforehand.

VI. Conclusion & Outlook

To support the SACA for software-intensive, cyber-physical systems, analysis summaries in the form of FGs are introduced in this paper. When applying a formalized safety analysis, FGs can be automatically extracted and enable algorithmic as well as visual SACA indications. Both algorithmic and visual indications complement each other to reach the final goal of improving the SACA for future safety-critical systems. Additional to the presented techniques, there are many ways to leverage formalization for SACA support just waiting to be discovered. This is why we want to encourage people to think about formalizing parts of their safety assessment. Formalization can be achieved without a lot of effort or required expert knowledge in a model-based context. In our opinion, focus should be directed towards automating error-prone and facilitating creative parts of the analysis. Especially for future and iterative developments of complex software-intensive systems, the need for automation also within the safety analysis will rapidly increase. However, integrating automation in the assessment process will still demand proper training to avoid intentional and unintentional misuse. Moving forward, it should be further evaluated how the presented derivation of SACA fits together with the overall assessment process of safety-critical systems. One area of synergy could be the combination with systematic and model-based safety assurance cases.

VII. Acknowledgment

This work is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 957210 - XANDAR: X-by-Construction Design framework for Engineering Autonomous & Distributed Real-time Embedded Software Systems.

References

[1] V. Shrivastava, "A study on the crash of Boeing 737 MAX," *International Journal of Science and Research (IJSR)*, volume 9, page 411, Aug. 2020. DOI: 10.21275/SR20805210709.

[2] V. K. Li Chen-Wei, S. Mio, and T. Yai, "The effects of aviation accidents on public perception toward an airline," *Journal of the Eastern Asia Society for Transportation Studies*, volume 11, pages 2347–2362, Dec. 2015. DOI: 10.11175/easts.11.2347.

[3] O. J. Dahl, E. W. Dijkstra, and C. A. R. Hoare, Eds., *Structured Programming*. GBR: Academic Press Ltd., 1972.

[4] J. Athavale, A. Baldovin, S. Mo, and M. Paulitsch, "Chip-level considerations to enable dependability for eVTOL and Urban Air Mobility systems," in *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, 2020, pages 1–6. DOI: 10.1109/DASC50938.2020.9256436.

[5] ISO, *ISO/PAS 21448:2019, Road vehicles – Safety of the intended functionality*, Standard, 2019.

[6] SAE, *J3187_202202, System Theoretic Process Analysis (STPA) Recommended Practices for Evaluations of Automotive Related Safety-Critical Systems*, Standard, 2022. [Online]. Available: https://www.sae.org/standards/content/j3187_202202/.

[7] N. G. Leveson and J. P. Thomas, *STPA Handbook*. 2018. [Online]. Available: https://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf (visited on 12/27/2020).

[8] D. Cofer, "Taming the complexity beast," *ITEA Journal*, volume 36, pages 313–318, 2015.

[9] RTCA, *Software Considerations in Airborne Systems and Equipment Certification, RTCA/DO-178C*, Standard, 2011.

[10] S. Friedenthal, A. Moore, and R. Steiner, *A Practical Guide to SysML, The Systems Modeling Language*. Morgan Kaufmann, 2015.

[11] A. Ahlbrecht and U. Durak, "Integrating safety into MBSE processes with formal methods," in *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*, 2021, pages 1–9. DOI: 10.1109/DASC52595.2021.9594315.

[12] A. Ahlbrecht and O. Bertram, "Evaluating system architecture safety in early phases of development with MBSE and STPA," in *2021 IEEE International Symposium on Systems Engineering (ISSE)*, 2021, pages 1–8. DOI: 10.1109/ISSE51541.2021.9582542.

[13] T. Gaska, C. Watkin, and Y. Chen, "Integrated Modular Avionics - past, present, and future," *IEEE Aerospace and Electronic Systems Magazine*, volume 30, number 9, pages 12–23, 2015. DOI: 10.1109/MAES.2015.150014.

[14] RTCA, *Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations, RTCA/DO-297*, Standard, 2005.

[15] C. Becker, J. Brewer, D. Arthur, F. Attioui, and L. Yount, "Functional safety assessment of a generic steer-by-wire steering system with active steering and four-wheel steering features," John A. Volpe National Transportation Systems Center (U.S.), Tech. Rep. DOT HS 812 576, 2018.

[16] C. Becker, D. Arthur, and J. Brewer, "Functional safety assessment of a generic, conventional, hydraulic braking system with antilock brakes, traction control, and electronic stability control," John A. Volpe National Transportation Systems Center (U.S.), Tech. Rep. DOT HS 812 574, 2018.

[17] L. Sun, "Establishing confidence in safety assessment evidence," University of York, Jul. 2012. [Online]. Available: https://etheses.whiterose.ac.uk/3183/.

[18] E. Harkleroad, A. Vela, and J. Kuchar, "Review of Systems-Theoretic Process Analysis (STPA) method and results to support NextGen concept assessment and validation," Lincoln Laboratory Massachusetts Institute of Technology, Tech. Rep. ATC-427, 2013.

[19] G. Biggs, T. Juknevicius, A. Armonas, and K. Post, "Integrating Safety and Reliability Analysis into MBSE: Overview of the New proposed OMG standard," *INCOSE International Symposium*, volume 28, pages 1322–1336, Jul. 2018. DOI: 10.1002/j.2334-5837.2018.00551.x.

[20] N. G. Leveson, *Engineering a Safer World, Systems Thinking Applied to Safety*. The MIT Press, 2016.

[21] J. Thomas, "Extending and Automating a Systems-Theoretic Hazard Analysis for Requirements Generation and Analysis," PhD, Massachusetts Institute of Technology.

[22] L. Masing, T. Dörr, F. Schade, *et al.*, "XANDAR: Exploiting the X-by-Construction paradigm in model-based development of safety-critical systems," in *2022 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2022, pages 1–5. DOI: 10.23919/DATE54114.2022.9774534.

[23] A. Ahlbrecht, U. Durak, and W. Zaeske, "Model-based STPA: Towards agile safety-guided design with formalization," in *2022 IEEE International Symposium on Systems Engineering (ISSE)*, 2022, to be published.