https://doi.org/10.1109/ISSE54508.2022.10005396

Model-Based STPA: Towards Agile Safety-Guided Design with Formalization

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

Wanja Zaeske
German Aerospace Center (DLR)
Institute of Flight Systems
Braunschweig, Germany
wanja.zaeske@dlr.de

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

***Abstract*—The competition for market entry in emerging segments such as Urban Air Mobility highlights the need for efficient and flexible development processes. This is accompanied by the trend towards software-intensive avionics systems due to the requirement for complex and computationally expensive algorithms. Considering the successful application of agile developments in the software domain, one might conclude that the agile paradigm would be the perfect fit to address these issues. However, for highly safety-critical domains such as aviation, multiple conflicts with the agile paradigm exist. Especially the constraint to follow rigorous and well documented processes contradicts the ideas of agile. To bridge this gap, a comprehensive and well documented, but still flexible process is necessary. Accordingly, this paper proposes a first step towards such an agile safety-guided design, by combining Model-Based Systems Engineering with the System-Theoretic Process Analysis. Particularly, focus is placed on enabling an iterative safety-guided design by providing functionality to track design changes to the corresponding safety artifacts. This automated functionality is enabled by a formalized execution of the safety analysis. At first glance, formalization sounds like a contradiction to the agile paradigm. However, we argue that formality and agility are not necessarily contradicting each other. Our theory is that moving the focus of formality from the human activities to the assisting functionality even increases overall agility. The iterative safety-guided design and resulting identification of safety improvements is demonstrated with examples of a flight assistance system.**

***Index Terms*—Safety, Agile, Formality, MBSE, STPA, SysML, Design Changes, Tracing, UAM, CPS**

I. Introduction

For the development of automated vehicles for Urban Air Mobility (UAM), various potential solution concepts compete for the market entry. Consequently, harsh time-to-market requirements are influencing developments in the safety-critical aviation sector. Simultaneously, the required automation of UAM vehicles increases the shift towards software-intensive airborne systems and introduces novel failure causes into the development [1]. This necessitates a paradigm shift for development approaches in safety-critical sectors. The goal would be to establish an efficient and effective Safety-Guided Design (SGD), where SGD stands for the proactive application of a safety analysis technique to guide the system development. A suitable safety analysis technique for establishing a SGD is the System-Theoretic Process Analysis (STPA) [2]. In addition, the system-theoretic approach of STPA also suits the analysis of software-intensive airborne systems, since the STPA was designed with the intention to identify novel failure causes such specification or interaction issues besides typical component failures.

Another interesting direction addressing the needs for future-oriented system developments is the movement towards agile and iterative Systems Engineering (SE) approaches [3]. At the same time, for safety-critical domains such as the aviation sector, rigorous and well documented safety assessments are unavoidable [4]. On the first glance, this seems to contradict with the required iterative agility. For instance, when iteratively changing the system, it is inefficient to repeat the complete safety assessment over and over again. Hence, a guidance is required indicating which safety artifacts have to be updated when iteratively changing parts of the developed system. To bridge this gap, a comprehensive safety traceability between design changes and their impact on safety artifacts has to be provided [5].

Definition 1. Safety traceability describes the ability to trace the impact of design decisions towards safety artifacts.

Definition 2. Safety artifacts encompass all elements that are either a part of the safety analysis execution or are created as a result. Examples are elements of hazard analysis tables, but also resulting requirements.

Subsequently, the main contribution of this paper is to outline and demonstrate an approach that shows how tracing and managing of design changes towards safety artifacts can be established. Therefore, two complementary functions will be explained and discussed. The first function informs the analysts when and where an update in the safety assessments is required by considering the recent design changes. The second function enables the creation of the corresponding safety artifacts that require an update. In the demonstration Section V, it will be shown how both can be applied during an iterative derivation of a simplified flight assistance system modeled with the Systems Modeling Language (SysML). Demonstration shows promising sings to enable efficient SGD for future-oriented safety-critical systems.

One prerequisite to establish such a traceability of changes is a firm linkage between the system development and the

© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

corresponding safety assessment. To outline how this can be established, the previously introduced and model-based STPA implementation of [6] is utilized. An important aspect is the systematic integration of model-based design and safety activities, since this lays the foundation for the envisioned change impact traceability. Therefore, it will be explained how safety analyses can be integrated into a systematic Model-Based Systems Engineering (MBSE) framework.

In summary, automatically tracing design change impact becomes necessary especially in agile and iterative development processes for future-oriented safety-critical systems. By laying the foundation for automatic change impact traceability, a first step towards agile SGD is described in this paper. In the following sections, the concept will be elaborated in more detail. First, required background information is introduced in Section II. Afterwards, the developed methodology will be outlined in Section III and demonstrated in Section IV. Then, related work will be described in Section V and the overall concept discussed in Section VI. Finally, the paper is concluded in Section VII.

II. Background

A. Challenges and Requirements for Agile Safety

Current developments in the aviation domain follow strict safety-driven processes with guidance documents such as ARP4754A [7] and ARP4761 [8]. In addition, standards that are specific to the developed system have to be considered. For software-intensive systems, especially DO-178C [9] describes the typical aspects that need to be addressed to get a certification. When trying to integrate the concepts of agile in such processes, multiple challenges have to be dealt with [4]. For instance, traditional aviation processes require a strict documentation that on a first glance contradicts with the values of the agile manifesto[1].

To overcome these challenges and enable an agile SGD, we argue that a systematic but flexible development process is required. Accordingly, the concepts of agile SE provide a promising foundation. Simultaneously, the ability to use safety as one of the main design drivers will be important to subsequent certification of the developed systems. To achieve this, a tight integration of safety and development tasks will be required. Finally, we argue that although the agile principles shift the focus towards humans instead of tools, an increasing formality and automation in the tools has the ability to also facilitate the overall agility of the development. This specifically manifests in the supporting ability to allow shorter feedback cycles within a SGD. For the safety traceability concept presented in this paper, the previous statements are the main drivers of the corresponding implementation.

B. Model-Based STPA

As introduced, an important part for an agile SGD of future safety-critical systems is the usage of a suitable safety analysis. With its control-oriented and model-based approach, the STPA enables to identify complex accident causes related to the specification and interaction of software-intensive systems [2]. At the same time, the STPA provides a formal basis, allowing to automate parts of the process. For instance, [10] demonstrated the ability to automatically generate requirements from the analysis. This concept inspired the implementation of the customized SysML profile explained in [6] that allows to execute the STPA within an MBSE environment, while providing the ability to automate parts. Using this profile, it was explained in [11] how the integration of STPA and MBSE has the ability to assist in evaluating system architecture safety in early phases of development.

Considering the analysis execution, the STPA follows four main steps [12] that are also represented in the model-based SysML implementation of [6]. First, the analysis purpose is specified by defining hazards, losses, and the system to consider. In the second step, a control structure of the system is modeled including the subsystems with their internal and external interactions as well as important process variables of subsystems. Due to the control-based approach of the STPA, these subsystems are referred to as controllers while the interactions are split into control actions and feedback. In the third step, control actions are analyzed in specific context situations to identify unsafe control actions. Therefore, it has to be specified in which contexts the control actions shall be analyzed. Considering the model-based implementation of [6], this step is represented by the linking of contexts elements with one or more context values to the corresponding control actions. When the links are established, instances of the *AnalysisBlockSTPA* of Fig. 1 can be created automatically by calculating the cross product of all context values.

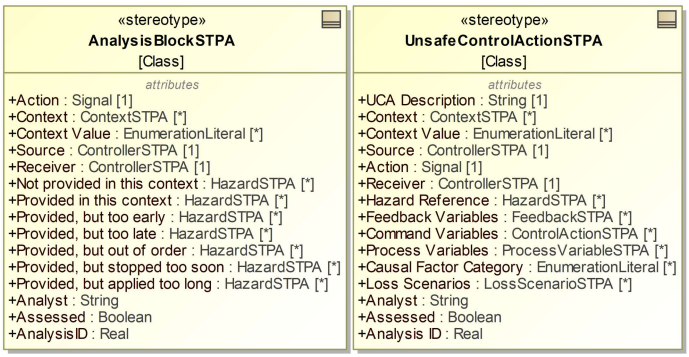

Fig. 1: Central Stereotypes of Model-Based STPA

Therefore, each *AnalysisBlockSTPA* instance represents one concrete context in which a control action shall be analyzed. Similar to the typical STPA execution, guiding sentences are used in the *AnalysisBlockSTPA* to identify and link hazards that can occur in the analyzed situation. Therefore, each *AnalysisBlockSTPA* instance essentially represents one row of a hazard analysis table. For instance, executing a control action in a context where it is dangerous would be an unsafe control action. Such an identification would be documented by linking the resulting hazard in the *Provided in this context* variable

[1]https://agilemanifesto.org/

of a *AnalysisBlockSTPA* instance. Within the model-based implementation of [6], *UnsafeControlActionSTPA* instances can be generated automatically using the inserted entries of the *AnalysisBlockSTPA* instances. During an automated creation, a lot of entries of the *UnsafeControlActionSTPA* instances can be pre-filled with the information available from the modeled control structure. Examplary, incoming *Feedback Variables*, *Command Variables*, and *Process Variables* of the analyzed controller can be inserted automatically. This pre-filled information assists the analyst in the identification of causal factor combinations that can lead to the execution of the analyzed unsafe control action. Essentially, this is the fourth and last step of the STPA, where loss scenarios are identified and mitigating safety requirements are derived to address the identified issues.

Since instances of the *AnalysisBlockSTPA* and *UnsafeControlActionSTPA* stereotypes can be created automatically under consideration of all relationships within the model, potential exists to trace the impact of related design changes. This potential is exactly what will be used in this paper to establish an important step towards an agile SGD.

C. Flight Assistance Use Case

For demonstration purposes, an example system is utilized that was developed within the XANDAR [13] EU project. To be more precise, a high-level SysML model of a *Flight Assistance System* will be considered. The corresponding STPA control structure is displayed in Fig. 2.

It is visible how the logical subsystems of the architecture in form of the: *Data Acquisition System*, *Avionic Computer* and *Pilot Assistance HMI* are represented as controllers while the controllers process variables are displayed as values. Moreover, the internal and external interactions help to understand the control flow within the system as follows. The *Data Acquisition System* receives information, processes it, and forwards it to the *Avionic Computer*. The *Avionic Computer* computes advisories from this information, which are then displayed on the *Pilot Assistance HMI* to the *Pilot*. For simplicity, only the *Pilot* is further considered as an outside interaction. In the demonstration Section IV, this *Flight Assistance System* and the corresponding control structure are used to demonstrate the concepts proposed in this paper.

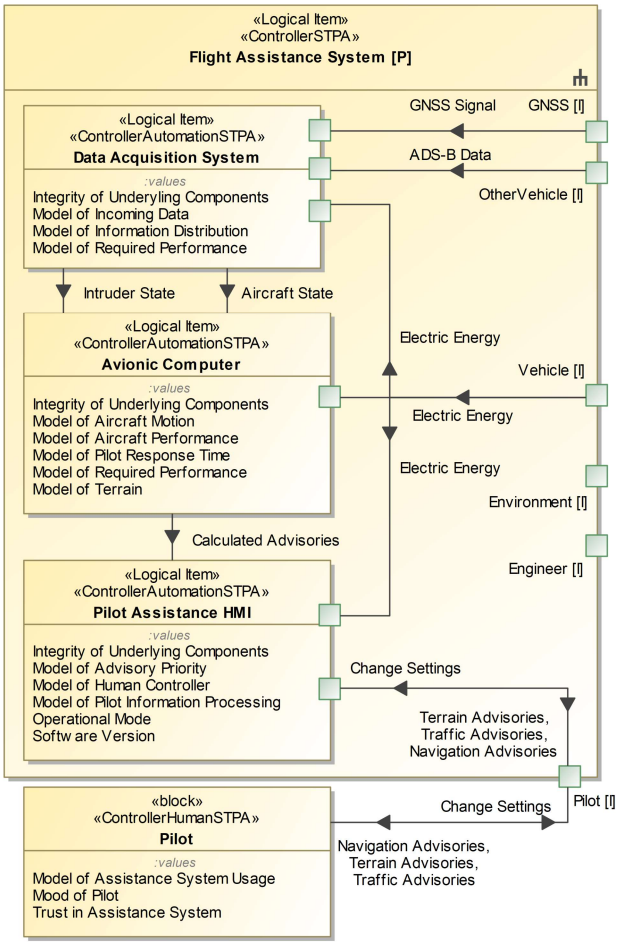

Fig. 2: Flight Assistance System Control Structure

III. Agile Safety-Guided Design

In this section, the envisioned concept will be introduced that lays the foundation for an agile SGD. The concept is divided into two main parts. The first part covers the integration of the safety and development activities into a cohesive modeling framework, allowing for a SGD approach. The second introduces concepts enabling to trace the impact of design changes towards safety artifacts by utilizing the model-based links established in the STPA execution.

A. Safety-Guided Design Framework

To establish a safety-guided decision making within the model-based system development, a thoughtful coupling of the safety and design activities has to be established. This can be achieved with the integration of the STPA as part of a suitable MBSE framework. Overall, MBSE frameworks provide a systematic layout of how the system design can be executed. In this paper, the MagicGrid [14] MBSE framework was selected due to its systematic but lean approach to system development. It divides the design into two main development phases where the system design is executed using the four pillars of SysML. The first *Problem* phase is a specification phase, where the system concept is modeled. In the second *Solution* phase, systems variants are designed that fit the specification and can be evaluated against each other. In Fig. 3, the framework is visualized and the different steps are displayed for each part of the two phases. Interestingly, the newest version of MagicGrid already includes a placeholder for a safety analysis in the framework structure that allows for a SGD [14].

Moreover, considering the parts that need to be modeled from MBSE and STPA side, multiple overlapping activities exist. For instance, the STPA requires a control structure model where the system boundary and outside interactions are clearly defined. Such a specification is already part of the modeling of the system context within the selected MBSE framework. Another overlap can be identified at the modeling

	Requirements	Behavior	Structure	Parametrics	Safety
Problem	Stakeholder Needs	Use Cases Functional Analysis	System Context Logical Systems	Measure-ments of Effectiveness	Concept STPA
Solution	Solution Requirements	Solution Behavior	Solution Assembly	Solution Parameters	Solution STPA

Fig. 3: Safety Analysis Integrated into MBSE Framework

of the architecture, because this can be used as a foundation to define the controllers and internal interactions of the STPA's control structure. As visible with these examples, an efficient integration of the STPA within a MBSE framework is possible.

Since the premise was to establish a SGD with the described coupling, it is important to consider how this can be achieved. Due to the safety analysis being a part of the MBSE framework in every level of design, the results of the safety analysis can influence the system design. On the one hand, the results can be directly incorporated to improve the system design on the same level. On the other hand, the derived mitigations can also influence the design decisions for the following more detailed design levels. The described process even facilitates an iterative SGD, where each time a safety analysis is executed, the design can be adapted accordingly untill the safety and design goals are sufficiently aligned. A prerequisite for the feasibility of this agile decision making is the ability to track the impact of design changes towards the safety artifacts. Otherwise, all safety activities would have to be executed from scratch in every iteration cycle, which would be a barrier for real-world adoption. This is why the next section will lay out a process allowing to track the design decisions to the safety artifacts of the model-based STPA.

B. Safety Traceability Foundation & Implementation

As introduced, a traceability of design decisions towards safety artifacts is targeted in this paper. Considering the model-based STPA implementation introduced in Section II-B, examples of safety artifacts are instances of the *AnalysisBlockSTPA* and *UnsafeControlActionSTPA* stereotypes of Fig. 1. For all parts of the model-based STPA that can be created or inserted automatically, the foundation to trace related changes is available. In the model-based STPA of Section II-B, this encompasses both *AnalysisBlockSTPA* instances that are used to identify unsafe control actions and *UnsafeControlActionSTPA* instances that document the identified unsafe control actions and assist in identifying and linking reasons for their occurrence (loss scenarios). Therefore, the sets of required and current elements can be calculated at every point in time for each of the two safety artifacts.

Definition 3. The set of **required elements** $a \in \mathcal{A}$ encompasses all safety artifact elements a that are required at the moment of calculation.

Definition 4. The set of **current elements** $b \in \mathcal{B}$ encompasses all safety artifact elements b that are present at the moment of calculation.

These two sets already build the foundation to trace the impact of design changes in the following ways. First, by calculating the difference between the required and current element sets, the set of new and changed elements can be calculated.

Definition 5. The set of new and **changed elements** $x \in \mathcal{X}$ encompasses all safety artifact elements x that are calculated with the difference of the required and current element sets: $\mathcal{X} = \mathcal{A} \setminus \mathcal{B}$.

Second, the set of stable (unchanged) elements can be calculated with the intersection of the required and current element sets.

Definition 6. The set of **stable elements** $y \in \mathcal{Y}$ encompasses all safety artifact elements y that are calculated with the intersection of the required and current element sets: $\mathcal{Y} = \mathcal{A} \cap \mathcal{B}$.

Finally, the set of outdated elements can be calculated with the difference between the current and required element sets.

Definition 7. The set of **outdated elements** $z \in \mathcal{Z}$ encompasses all safety artifact elements z that are calculated with the difference of the current and required element sets: $\mathcal{Z} = \mathcal{B} \setminus \mathcal{A}$.

The relationships between the calculated sets can also be visualized as shown in Fig. 4. It is visible how the different sets can be extracted from the current and required elements. Using the defined sets, different supporting functionality can be implemented for the analysis. For instance, when a change in a model occurs, the analyst can be notified about the safety artifacts that are new/changed, stable, or outdated as a result. Moreover, functionality can be implemented that executes the updates in the safety artifacts while also notifying the analyst.

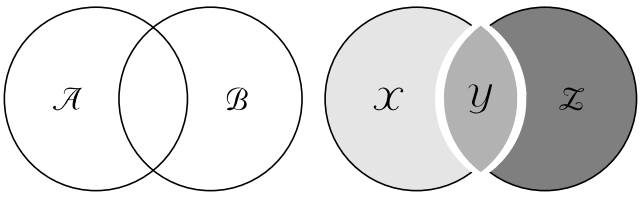

Fig. 4: Set-Relationships for Safety Traceability

After abstractly defining how the different sets can be calculated, the algorithm will be outlined for the *AnalysisBlockSTPA* safety artifact. In the model-based STPA, the *AnalysisBlockSTPA* instances are calculated considering the contexts that are linked to the control actions. These links specify in which situations the control actions shall be analyzed. Each context contains a list of related states that essentially can be viewed as a vector of context variables. When linking more than one context, the cross product of the context vectors is calculated and used to create all context combinations in which the control action shall be analyzed. This means that the set of

required *AnalysisBlockSTPA* instances can be calculated at every point in time under consideration of the links established within the model. Subsequently, it is therefore possible to calculate all sets introduced beforehand as shown in Alg. 1 and implement the envisioned change impact traceability functions.

Definition 8. The set of **control action elements** $ca \in \mathcal{CA}$ encompasses all control actions that shall be analyzed within the safety analysis.

Definition 9. The set of **context elements** $co \in \mathcal{C}o$ encompasses all context elements that are linked to control actions within the model.

Algorithm 1: Calculating Sets for AnalysisBlockSTPA

Input: $\mathcal{B}$, $\mathcal{CA}, \mathcal{C}o$
Output: $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$
// go through each control action ca to calculate required elements $\mathcal{A}_{ca}$
forall $ca \in \mathcal{CA}$ **do**
  // create context combinations $\mathcal{CC}_{ca}$ using attached contexts $co_{ca} \in \mathcal{C}o_{ca}$
  $\mathcal{CC}_{ca} = co_{ca} \times co_{ca}$
  if $\mathcal{CC}_{ca} \neq \emptyset$ **then**
    // go through all related context combinations
    forall $cc_{ca} \in \mathcal{CC}_{ca}$ **do**
      // add required element a_{ca} to required set $\mathcal{A}_{ca}$
      $\mathcal{A}_{ca} = \mathcal{A}_{ca} \cup a_{ca}$
  // add calculated elements $\mathcal{A}_{ca}$ to overall required element set $\mathcal{A}$
  $\mathcal{A} = \mathcal{A} \cup \mathcal{A}_{ca}$
// calculate all element sets
$\mathcal{X} = \mathcal{A} \setminus \mathcal{B}$
$\mathcal{Y} = \mathcal{A} \cap \mathcal{B}$
$\mathcal{Z} = \mathcal{B} \setminus \mathcal{A}$
return $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$

In Alg. 1, the set of current *AnalysisBlockSTPA* instances $\mathcal{B}$, the set of control actions $\mathcal{CA}$, and the set of the linked context elements $\mathcal{C}o$ are used as input for the algorithm. By calculating all required elements using the annotated context combinations of each control action, the required element set $\mathcal{A}_{ca}$ can be calculated for each control action. Afterwards, the sets of new, stable, and outdated *AnalysisBlockSTPA* instances can be calculated and used to establish different supporting functionality as described earlier.

A similar process can be implemented for the *UnsafeControlActionSTPA* instances or other safety artifacts. In general, the process always requires the following steps: 1. Identify the set of current elements, 2. Calculate the set of required elements, 3. Calculate the new, stable, and outdated elements, 4. Use the information to support the analyst in various ways. Due to length limitations, the detailed algorithm is omitted for the *UnsafeControlActionSTPA* safety artifacts.

IV. Demonstrating Safety Traceability

In the demonstration section, the main focus is placed on the change impact traceability. Hence, the *Flight Assistance System* introduced in Section III-C is utilized to show how the tracing can enable a timely update of safety artifacts.

A. Tracing the Impact of Context Changes

During an agile and model-based development of the *Flight Assistance System*, it was identified through a coverage assessment as presented in [15] that the *Traffic Advisories* context shall be extended. For instance, instead of only analyzing a context where one intruder is nearby, also multiple intruders shall be considered. In addition, it was identified that the terrain related context might also have an influence on the safe issuing of *Traffic Advisories*. Therefore, the context is extended by also considering multi-intruder scenarios and the terrain related context as visible in Fig. 5.

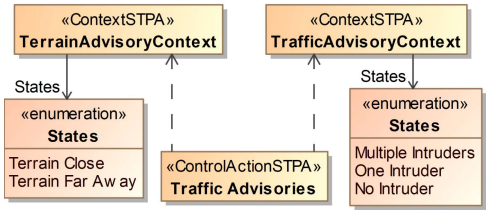

Fig. 5: Extended Traffic Advisory Context

After establishing the adapted relationships, automated functionality can immediately identify the changes in the context and calculate the adaptations in the *AnalysisBlockSTPA* instances. Using the set operations established in Section III-B, it is possible to calculate the new and changed, as well as the outdated *AnalysisBlockSTPA* instances. At the same time, an update functionality can be used to create the novel elements while highlighting the outdated ones. When executing the update functionality, six new *AnalysisBlockSTPA* instances can be automatically created according to the cross product of the two enumerations of Fig. 5. Two of the created *AnalysisBlockSTPA* instances are displayed in Fig. 6. All entries that are inserted automatically are highlighted in blue while the ones that are inserted automatically and have changed are highlighted in red. After being supported by the notifying and update functionality, the safety analyst can manually execute the required safety analysis update. Therefore, the typical steps of the STPA execution have to be followed. Under consideration of the context in combination with the guidewords, it is identified in *STPA Element 16* that providing unnecessary *Traffic Advisories* when being in close range of terrain might lead to a *Controlled Flight into Terrain (CFIT)* of the aircraft. Similarly, it is identified in *STPA Element 17* that out of order *Traffic Advisories* in a multi-intruder scenario could lead to failure to prevent a *Near Mid Air Collision (NMAC)* of the aircraft. After linking the corresponding hazards in the

AnalysisBlockSTPA instances, automated functionality can be used to create the related *UnsafeControlActionSTPA* instances displayed in Fig. 7, allowing a further analysis for potential causes. Again, all entries that are inserted automatically during creation are highlighted in blue and all entries that have changed in addition are highlighted in red. For *STPA UCA 0*, an incorrect *Model of Advisory Priority* is identified and linked as a potential reason for an inadequate issuing of *Traffic Advisories*. This loss scenario is further detailed in Fig. 8. For *STPA UCA 12*, it is identified that an incorrect intruder focus can be the reason for an out of order issuing of *Traffic Advisories* in a multi-intruder scenario. Subsequently, the identified causal information can be used to derive corresponding mitigations. Regarding the first loss scenario of Fig. 8, a mitigation is derived stating that a precise advisory priority shall be defined as shown in Fig. 9. Moreover, to prevent the second loss scenario, it is derived that a precise definition of how multiple intruders are handled has to be established.

B. Tracing the Impact of System Changes

After demonstrating the ability to trace changes in the analysis context, it is explained in the following how changes in the systems control structure model can be traced. Therefore, the control structure of the *Flight Assistance System* in Fig. 2 will be considered. During the analysis of the operating environment of the *Pilot Assistance HMI*, it might be identified that the *Environmental Influences* also have to be considered. In practice, this would mean that the control structure diagram of Fig. 2 would be extended with a corresponding control action between the *Environment [I]* port and the *Pilot Assistance HMI*. When this connection is established, it also has to be evaluated if this additional relation could be a potential causal factor for the execution of an unsafe control action. Considering the *UnsafeControlActionSTPA* instance from Fig. 1, it is visible that incoming *Command Variables* are attributes of the stereotype. Moreover, the Variables are among others pre-filled during the automated creation of the stereotypes instances as explained in Section II-B. After adding the *Environmental Influences* control action, it is therefore known that all of the *UnsafeControlActionSTPA* instances related to the *PilotAssistanceHMI* have to be updated to include the new incoming *Command Variable*. Therefore, the safety engineers can be notified that an update is required in the *UnsafeControlActionSTPA* instances and the adapted instances can be created as displayed in Fig. 7.

Since the safety engineers are automatically notified after the change, they can then search for potential loss scenarios that were missed beforehand. Exemplary, the loss scenario could be identified that electromagnetic interference of the surrounding environment might be able to change the displayed advisory as shown in the third loss scenario of Fig. 8. Subsequently, a corresponding mitigation can be derived stating that the *Pilot Assistance HMI* has to cope with electromagnetic compatibility as displayed in Fig. 9. This is only one example of a system related change that can be traced towards the safety artifacts and used to update the safety analysis. In general, all changes in the properties that are inserted during the automatic generation of the safety artifacts are traceable.

V. Related Work

Since the agile paradigm stems from the software community, interesting work has been done to integrate the software considerations into the safety-critical aviation domain. For instance, [4] explains how agility can be compatible with DO-178C [9] and used to develop certifiable avionics software.

Another important area for this paper is the combination of agile developments with a systematic SE activities. An overview of the concepts and challenges of agile SE is provided in [3]. Considering the step towards combining MBSE with the agile paradigm, [16] proposes a framework that integrates the ideas of agile into MBSE by structuring the development process according to a DevOps approach.

Interestingly, not only SE and MBSE are linked to the agile paradigm in the literature. It is also shown in [17], [18] how the STPA itself can be integrated into agile development processes. Regarding the establishment of an agile STPA execution, [19] argues that a sufficient tool support is required that enables to track design changes.

Since safety and security are both emergent properties, the analysis of them can also be quite similar. This is supported by the fact that the STPA is also be applied for security concerns in form of the STPA-Sec [20]. Therefore, it is also interesting to consider related approaches in the security direction. Looking at the integration of agility with STPA, [21] demonstrates how security concerns can be efficiently evaluated with the the agile paradigm. At the same time, the combination of agility and MBSE was considered for the security analysis of IOT devices in [22].

After outlining work that combines topics such as agility, MBSE, and analyses, related work in the direction of tracing changes will be introduced. In [23], a model-based traceability of changes is proposed that allows automatic and iterative creation of parts for a "Dependent Failure Analyse (DFA)". Furthermore, [24] shows how traceability can be established in combination with SysML to trace the impact of requirement changes. On the overarching level of change management, [25] explains how MBSE can be used to facilitate the change impact analysis for product design processes. Finally, [26] lays out a process to establish a systematic change impact traceability for safety artifacts in software product lines.

VI. Discussion

To allow an agile SGD in future system developments, the ability to trace safety related system changes becomes inevitable. With the envisioned process presented in this paper, changes can be traced automatically and used to implement multiple supporting functionality. However, there are some limitations to the process that have to be considered. The current implementation enables powerful supporting functionality, but does currently only cover traceability of changes in the aspects that are related to the targeted safety artifacts. Hence, the provided information relating to updates has to

#	Name	Context	Context Value	Source	Action	Receiver	Provided in this context	Provided, but out of order
1	STPA Element 16	TerrainAdvisoryContext TrafficAdvisoryContext	Terrain Close Multiple Intruders	Pilot Assistance HMI	Traffic Advisories	Pilot	Flight Assistance System Contributes to a CFIT	
2	STPA Element 17	TerrainAdvisoryContext TrafficAdvisoryContext	Multiple Intruders Terrain Far Away	Pilot Assistance HMI	Traffic Advisories	Pilot		Flight Assistance System Fails to Prevent NMAC

Fig. 6: New and Changed AnalysisBlockSTPA Instances

#	Name	UCA Description	Context	Context Value	Command Variables	Process Variables	Loss Scenarios
1	STPA UCA 0	The [Pilot Assistance HMI] applies [Traffic Advisories] to the [Pilot] when [TerrainAdvisoryContext] is in state [Terrain Close] and [TrafficAdvisoryContext] is in state [Multiple Intruders].	TerrainAdvisoryContext TrafficAdvisoryContext	Multiple Intruders Terrain Close	Calculated Advisories Change Settings Electric Energy Environmental Influences Update SW	Model of Advisory Priority Model of Human Controller Model of Pilot Information Processing Operational Mode Software Version Integrity of Underlying Components	AdvisoryPriorityMismatchResultsInWrongAdvisoryOutput ElectromagneticInterferenceChangesAdvisoryOutput
2	STPA UCA 12	The [Pilot Assistance HMI] applies [Traffic Advisories] to the [Pilot] out of order when [TerrainAdvisoryContext] is in state [Multiple Intruders] and [TrafficAdvisoryContext] is in state [Terrain Far Away].	TerrainAdvisoryContext TrafficAdvisoryContext	Multiple Intruders Terrain Far Away	Calculated Advisories Change Settings Electric Energy Environmental Influences Update SW	Model of Advisory Priority Model of Human Controller Model of Pilot Information Processing Operational Mode Software Version Integrity of Underlying Components	IncorrectIntruderFocusResultsInInadequateAdvisory ElectromagneticInterferenceChangesAdvisoryOutput

Fig. 7: New and Changed UnsafeControlActionSTPA Instances

#	Name	Causal Factor Classification	Causal Factor Source	Causal Factor	Loss Scenario Description	Loss	Mitigations
1	AdvisoryPriorityMismatchResultsInWrongAdvisoryOutput	Process Model	Pilot Assistance HMI	Model of Advisory Priority	An imprecise definition of the advisory priorities leads to a inadequate forwarding of a traffic advisory even though a more important terrain advisory is present. This leads to the pilot not executing the right terrain avoidance maneuver and a CFIT of the aircraft.	Loss of Aircraft	209.2.5 Precise Advisory Priority Definition
2	IncorrectIntruderFocusResultsInInadequateAdvisory	Algorithm	Avionic Computer	Multiple Intruders	During a multi intruder scenario the incorrect intruder is selected as the highest priority. This leads to a late traffic advisory and a failure to prevent a CFIT or NMAC of the aircraft.	Loss of Aircraft	209.3.2 Precise Definition of Intruder Priority
3	ElectromagneticInterferenceChangesAdvisoryOutput	Environment /External	Environment	Environmental Influences	Before the advisory is displayed, electromagnetic interferrence changes the advisory output. This leads to an inadequate advisory output and a contribution to a CFIT or NMAC of the aircraft.	Loss of Aircraft	209.5.2 HMI Electromagnetic Mitigation

Fig. 8: Loss Scenarios Identified due to Design Change Traceability

#	Name	Text
1	Precise Advisory Priority Definition	The Flight Assistance System shall use a precisely defined priority of all advisories.
2	Precise Definition of Intruder Priority	Th Flight Assistance System shall use a precise definition of intruder priority to correctly handle multi intruder scenarios.
3	HMI Electromagnetic Mitigation	The HMI design and installation shall consider and prevent electromagnetic interference of other components.

Fig. 9: Derived Loss Scenario Mitigations

be considered as valuable supporting functionality, but not as the all-encompassing solution. In the end, the human analyst has the final decision about the completeness of the traced changes. This also aligns with the ideas of the agile paradigm, since the automation mainly helps to facilitate discussions about the impact of design changes, while also reducing manual tracing efforts which are error-prone. A further limitation of the automation is that a formal analysis structure has to be followed to enable the supporting functionality which is always related to overhead. Still, we would argue that an overhead to defining the system and analysis analysis parts more precisely is not necessarily a bad thing.

In terms of the SGD aspect that integrates safety and design activities, the potential differences between the design and safety analysis model have to be discussed. In fact, even though the design and analysis model might overlap in a lot of ways, there is still the possibility that not the same set of model elements is used on purpose in both tasks. For the ability to trace design changes, this gap in the overlapping elements is the limiting factor. This limitation has to be considered during the establishment of the control structure within the MBSE framework. All elements that are not annotated with the required STPA related stereotypes and used within the safety considerations, are also not traceable in the current implementation.

VII. Conclusion

By utilizing a model-based integration of MBSE and STPA, a step towards establishing an agile SGD is outlined in this

paper. This integration allows the ability to trace the impact of design changes to the changes required in related safety artifacts. Using this traceability, support functionality to inform the safety analyst and update the safety artifacts can be easily implemented. Current trends show that such a process will be required to establish efficient developments of future safety-critical systems. Considering the potential limitations of the process and the values of the agile paradigm, it is important to not forget the human component in such a SGD. With the supporting functionality highlighting and updating artifacts that are affected by design changes, the analysts are relieved of the error-prone parts and gain more time to focus on the creative tasks of the analysis. For instance, they can focus on identifying complex relationships that were missed by the automation and enhance the safety analysis even further. Essentially, we argue that this process facilitates the creative tasks and thereby increases the overall development agility.

VIII. 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.

References

[1] 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.

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

[3] C. Ebert and F. Kirschke-Biller, "Agile systems engineering," *IEEE Softw.*, volume 38, number 4, pages 7–15, Jul. 2021. DOI: 10.1109/MS.2021.3071806.

[4] J. Marsden, A. Windisch, R. Mayo, *et al.*, "ED-12C/DO-178C vs. Agile Manifesto – A Solution to Agile Development of Certifiable Avionics Systems," in *ERTS 2018*, series 9th European Congress on Embedded Real Time Software and Systems (ERTS 2018), Toulouse, France, Jan. 2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02156357.

[5] J.-P. Steghöfer, E. Knauss, J. Horkoff, and R. Wohlrab, "Challenges of scaled agile for safety-critical systems," in *Product-Focused Software Process Improvement*, Cham: Springer International Publishing, 2019, pages 350–366.

[6] 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.

[7] SAE, *Guidelines for Development of Civil Aircraft and Systems, ARP 4754A*, Standard, 2010.

[8] SAE, *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment, ARP 4761*, Standard, 1996.

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

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

[11] 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.

[12] 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 08/18/2022).

[13] 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.

[14] A. Aleksandraviciene and A. Morkevicius, *MagicGrid® Book of Knowledge, A Practical Guide to System Modeling using MagicGrid from No Magic*, 2nd. Vitae Litera: Kaunas, Lithuania, 2021.

[15] A. Ahlbrecht and U. Durak, "Model-based STPA: Enabling safety analysis coverage assessment with formalization," in *2022 IEEE/AIAA 41th Digital Avionics Systems Conference (DASC)*, 2022, to be published.

[16] V. Salehi and S. Wang, "Munich Agile MBSE Concept (MAGIC)," *Proceedings of the Design Society: International Conference on Engineering Design*, volume 1, number 1, pages 3701–3710, 2019. DOI: 10.1017/dsi.2019.377.

[17] Y. Wang, "System-Theoretic Safety Analysis in Agile Software Development," PhD, Universität Stuttgart.

[18] Y. Wang and S. Wagner, "Towards applying a safety analysis and verification method based on STPA to agile software development," in *2016 IEEE/ACM International Workshop on Continuous Software Evolution and Delivery (CSED)*, IEEE, 2016, pages 5–11.

[19] N. Ludvigsen, "Prototyping a digital support tool for an agile implementation of STPA," Master's Thesis, Norwegian University of Science and Technology.

[20] W. Young and N. Leveson, "Systems thinking for safety and security," in *Proceedings of the 29th Annual Computer Security Applications Conference*, series ACSAC '13, New Orleans, Louisiana, USA: Association for Computing Machinery, 2013, pages 1–8. DOI: 10.1145/2523649.2530277.

[21] G. Pope, "Systemic Theoretic Process Analysis (STPA) used for cyber security and agile software development," Apr. 2021. [Online]. Available: https://www.osti.gov/biblio/1779599.

[22] B. L. Papke, "Enabling design of agile security in the IOT with MBSE," in *2017 12th System of Systems Engineering Conference (SoSE)*, 2017, pages 1–6. DOI: 10.1109/SYSOSE.2017.7994938.

[23] P. Munk, *Model-based Depended Failure Analyse (DFA)*, Presentation, Bosch GmbH, 2021. [Online]. Available: https://safetronic.fraunhofer.de/wp-content/uploads/2021/11/Safetronic21-ModelBased-DFA-Munk.pdf (visited on 06/20/2022).

[24] S. Nejati, M. Sabetzadeh, C. Arora, L. C. Briand, and F. Mandoux, "Automated change impact analysis between SysML models of requirements and design," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, series FSE 2016, Seattle, WA, USA: Association for Computing Machinery, 2016, pages 242–253. DOI: 10.1145/2950290.2950293.

[25] R. Wilms, P. Kronsbein, D. Inkermann, T. Huth, M. Reik, and T. Vietor, "Using a cross-domain product model to support engineering change management," *Proceedings of the Design Society: DESIGN Conference*, volume 1, pages 1165–1174, 2020. DOI: 10.1017/dsd.2020.90.

[26] M. Käßmeyer, M. Schulze, and M. Schurius, "A process to support a systematic change impact analysis of variability and safety in automotive functions," in *Proceedings of the 19th International Conference on Software Product Line*, series SPLC '15, Nashville, Tennessee: Association for Computing Machinery, 2015, pages 235–244. DOI: 10.1145/2791060.2791079.