31st Annual INCOSE international symposium
virtual event
July 17 - 22, 2021

Integrating Safety Analysis into Model-Based Systems Engineering for Aircraft Systems: A Literature Review and Methodology Proposal

Kimberly Lai
University of Toronto
Toronto, ON
kimberly.lai@mail.utoronto.ca

Thomas Robert
Safran Landing Systems
Ajax, ON
thomas.robert@safrangroup.com

David Shindman
Safran Landing Systems
Ajax, ON
david.shindman@safrangroup.com

Alison Olechowski
University of Toronto
Toronto, ON
olechowski@mie.utoronto.ca

Abstract. Model-Based Systems Engineering (MBSE) has become increasingly popular within the aircraft industry in recent years. However, this model-based approach presents a challenge as traditional safety analysis practices are unable to keep up, resulting in inconsistency between the system and safety domains. This paper proposes a methodology tailored towards aircraft systems that addresses this issue by integrating safety analysis into MBSE. This is achieved by extending the Systems Modeling Language (SysML) profile to account for safety data in the system model and utilizing an Application Programming Interface (API) to automate the generation of safety analysis artefacts. The proposed methodology also allows for requirements management integration to increase the efficiency of the system development process.

Introduction

In recent years, the development of large, integrated systems within the aerospace industry has become increasingly complex and competitive (Gabbai 2005). Aircraft manufacturers and Tier 1 suppliers are expected to develop innovative systems in less time while maintaining a high degree of flexibility to keep up with clients' changing demands. In response to these needs, many companies within the industry are adopting Model-Based Systems Engineering (MBSE) to develop their systems (Howard 2018), hence moving from a document-centric approach to a model-centric one. Moreover, for a safety critical platform such as an aircraft, safety analysis is an essential part of the development cycle to ensure the integrity, quality and robustness of a design. However, as defined in the SAE Aerospace Recommended Practices, ARP4754A (SAE Aerospace 2010) and ARP4761 (SAE International 1996), system development and safety assessment processes are interdependent and iterative. Problems arise as safety analyses are typically performed separately by safety engineers and extraction of information from the system model is done manually. Designs tend to evolve continuously, resulting in analyses being performed on outdated versions of the model, and thus wasting time and money. Thus, to remain consistent with the design while remaining error-free, the integration of safety analysis into MBSE is essential. System and safety engineers can then work on the same model concurrently, allowing for improved efficiency, reliability and traceability.

The purpose of this paper is to provide insights into the various available methods for integrating safety analysis into MBSE and to propose a clear methodology that can be used directly in industry. The objectives are to: 1) allow for the automatic generation of FHA, FMEA and FTA, 2) enable manual edits made in these generated safety analysis artefacts to be propagated back into the model and 3) achieve traceability of model elements to requirements. With this approach, the increased efficiency and reliability of the development processes will allow companies to become more competitive and deliver safer systems, ultimately resulting in the production of safer aircraft.

Current Status of Safety Analysis

In practice, safety analysis is often performed independently with different tools by safety engineers and occurs relatively late in the design process. In cases where the design has already been finalized before safety analysis is complete, the opportunity to influence design decisions and make the required modifications is missed (Leveson 2018). These late changes can be extremely costly, as the removal of design faults detected during the design phase is up to six times more costly than it would be in the early concept phase (INCOSE 2015). This multiplier can even increase to 100 at the development stage and 1000 at the production/testing stage. In addition to this, in order to perform safety analyses, safety engineers must first extract information from the system model, and as the system model evolves continuously, this results in safety analyses being performed on obsolete designs (Baklouti et al. 2019). Therefore, early integration of safety assessment and earlier validation of system safety requirements is crucial.

The Purpose of Model-Based Safety Analysis

Model-Based Safety Analysis (MBSA) provides an approach to tackle these problems by having system and safety engineers work on a common model as shown in Figure 1. The system model on which the safety analysis is performed can be updated in real-time, thus preventing work from being done on outdated information. The use of a common model also encourages safety requirements to be considered at the early conceptual design stages which can help reduce the number of iterations and design changes further down the line (Mhenni, Nguyen & Choley 2013). Furthermore, the use of models allows for automation, such as the automatic generation of Fault Tree Analysis (FTA) and Failure Modes and Effects Analysis (FMEA), and increased traceability between requirements and design components. The automatic generation of safety analysis artefacts reduces the overall development time, decreases error proneness and thus enables safety engineers to become more efficient (Mhenni 2014).

The deployment of MBSA can greatly improve the reliability and efficiency of safety analysis activities, and this is the main reason why many industry practitioners have started employing or investigating this approach (Li, Gong & Su 2014). By integrating safety analysis into the MBSE processes, companies are able to develop complex and innovative systems in less time, at a lower cost and can respond quickly to clients' changing demands. However, it is worth noting that despite using a shared model in this model-based approach, the system design and safety data are organized in a way which allows system and safety engineers to edit the model simultaneously without causing conflict.

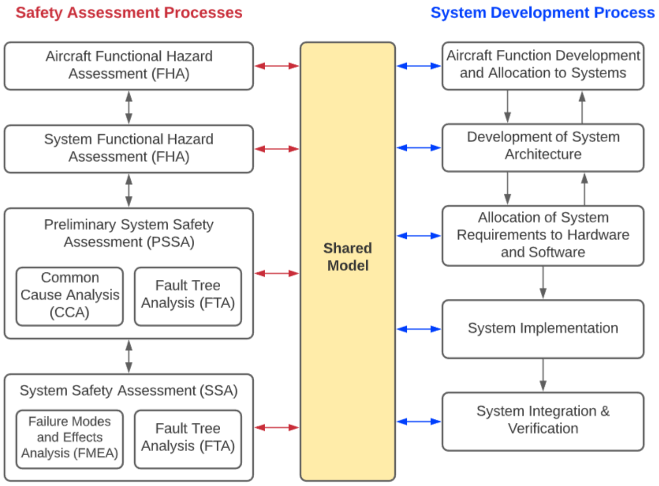

Safety Assessment Processes
System Development Process
Aircraft Functional Hazard Assessment (FHA)
System Functional Hazard Assessment (FHA)
Preliminary System Safety Assessment (PSSA)
Common Cause Analysis (CCA)
Fault Tree Analysis (FTA)
System Safety Assessment (SSA)
Failure Modes and Effects Analysis (FMEA)
Fault Tree Analysis (FTA)
Shared Model
Aircraft Function Development and Allocation to Systems
Development of System Architecture
Allocation of System Requirements to Hardware and Software
System Implementation
System Integration & Verification

Figure 1: Use of a Shared Model for Safety Assessment and System Development, Adapted from Previous Works (Stewart et al. 2019; SAE Aerospace 2010)

Review of Existing Methodologies

This section will provide a discussion of existing methodologies that allow for a Model-Based Safety Analysis approach to be used. In order to narrow the scope of research and be relevant to aircraft systems, the focus will be on works using SysML as the modeling language. The existing work can be largely grouped into two methods, model-to-model transformation and an extension of the modeling language, both of which will be described in more detail below.

Model-to-Model Transformation

The concept behind the model-to-model transformation method is to take advantage of existing safety tools to perform safety analysis and automatically generate FTAs and FMEAs. To achieve this, the system model is transformed into a form that can be processed by such existing tools. The two methods that will be discussed in this section both propose transformations to AltaRica, which is a formal modeling language developed specifically for safety analysis that allows for the modeling and simulation of failure events (Batteux, Prosvirnova & Rauzy 2017).

SMF-FTA Methodology. Yakmets, Jaber & Lanusse (2013) presented the Safety Modelling Framework for Fault Tree Generation (SMF-FTA). SMF-FTA is a framework that introduces a model transformation tool and verification algorithm to integrate safety analysis into MBSE. The fundamental step in this process is to use a converter program embedded within the safety tool to convert the SysML system model into AltaRica language. Existing tools can then take the AltaRica models and derive minimal cut sets, thus automatically generating fault trees. A problem that arises with this approach is that even though an embedded checker, ARC (AltaRica Checker), exists within the AltaRica toolset, there can still be errors resulting from the model transformation that are not recognized by the tool. Ultimately, safety engineers must manually check the AltaRica models and make corrections regardless of the ARC verification results.

MéDISIS Methodology. David, Idasiak & Kratz (2010) proposed the MéDISIS methodology which enables the automatic creation of a preliminary FMEA report from functional behaviors modeled in a system model. To achieve this, an algorithm in XML (Extensible Markup Language) is used to analyze and organize data from the system model. The model is then transformed into the AltaRica language, and external tools are used to generate reliability indicators and identify causes of failure. Finally, a manual review from safety engineers is needed to complete the final FMEA. Once again, this methodology takes advantage of existing tools that can perform various safety analysis activities on an AltaRica model. The advantage MéDISIS holds over the SMF-FTA methodology is the addition of a Dysfunctional Behavior Database (DBD) as a repository to manage safety data and the propagation of any modifications. Nevertheless, there is still a need for domain expert completion at every iteration.

Extension of Modeling Language

In contrast, the main concept of the modeling language extension approach is to extend the SysML profile to include concepts from the safety domain. This allows safety data to be added to the system development model, hence making it possible to directly generate safety analysis artefacts, instead of the need for a model transformation as discussed above. The Systems Modeling Language, SysML, is an extension of the Unified Modeling Language (UML) which was first developed as a generic modeling language in the software field. It has a profile extension mechanism that allows users to customize the UML profile for a particular domain, and therefore SysML can also be extended to suit the users' needs. Examples of approaches which extend the modeling language are discussed below.

Phillip Helle's Methodology. Helle (2012) describes a proposed method with such an approach involving extending the SysML profile. Safety requirements are introduced as textual elements and subsequently formalized into Failure Cases which are modeled as SysML Use Cases. These Failure Cases are introduced into the SysML metamodel and connected to the functional architecture of the system model. As Helle's method specifies IBM Rhapsody as the modeling tool, this allows the use of the Rhapsody API for Java programs to be used to create a program to read the model and extract relevant data. The program then calculates and determines the failure rate of certain components, and hence whether a design would be accepted depending on the imposed allowed failure probability. With this, safety analysis is integrated within the system development process and system engineers can obtain safety related feedback on their design decisions immediately. However, as acknowledged by Helle, this approach cannot be a substitute for safety analysis for certification and should only be used to support the design of safer systems and checking against preliminary safety requirements.

SafeSysE Approach. Another approach named SafeSysE was proposed by Mhenni, Nguyen & Choley (2018). This approach is directly applicable to the development of aircraft systems as it facilitates the automatic generation of FTAs and FMEAs from the architecture model. Various new stereotypes and attributes are introduced with the purpose of storing information such as failure modes and rates, severity, causal factors and failed states, all of which are used in the creation of safety artefacts. As the SafeSysE method uses SysML, the models can be exported as XMI (XML Metadata Interchange) files, which are then given as inputs to a Python developed tool that is specifically created to extract the desired safety data from the XMI files. Finally, the tool uses this data to construct FMEAs and FTAs as needed.

A useful functionality of SafeSysE is the capability to propagate changes made by safety engineers in the outputted FMEA worksheets back into the system model via the Python tool. However, the absence of any connections to safety requirements means that this method is lacking one of the key aspects of safety analysis.

Existing Methodologies Summary

The four methodologies described in the preceding section represent proposed methods that aim to integrate safety analysis into MBSE, thus effectively implementing MBSA. The key functionalities each method addresses are summarized in Table 1.

Table 1: Comparison of Various MBSA Methodologies

	Model-to-model Transformation		Extension of SysML	
Key Functionalities	SMF-FTA	MéDISIS	Helle's methodology	SafeSysE methodology
Capture of requirements	No. Not considered.	Captured as text in the system model with traceability to components.	Captured as text in the system model and formalized into failure cases.	None.
Identifying design acceptance based on failure probability	No. Not considered.	Yes. Generated via AltaRica model.	Yes. Generated via Java program.	No. However the infrastructure needed to do this is available.
Auto-generation of FTA	Yes. Generated from AltaRica model.	Not mentioned but achievable with AltaRica model.	None.	Yes. Generated via XMI files with a Python program.
Auto-generation of FMEA	No.	Yes. Generated via XML algorithm.	None.	Yes. Generated via XMI files with a Python program.
Flexibility with different modeling tools	Yes. Valid for all tools compatible with SysML.	Yes. Valid for all tools compatible with SysML.	Only applicable to Rhapsody.	Yes. Valid for all tools compatible with SysML.
Propagation of manual edits in SA artefacts back into the system model	No. Must be done manually.	Yes, using DBD model.	N/A.	Yes.

Analysis of Existing Works

Practically speaking, model-to-model transformation methodologies are simpler to implement as they make use of existing safety analysis technologies and their capabilities. AltaRica models are already used for safety assessment at many major companies within the aerospace industry and hence tools are well developed and, in some cases, even certified (Bozzano et al. 2015). As it is commonly used, it is likely that no extra training is needed for safety engineers to perform verification and modifications on the safety model.

On the other hand, the benefit of extending the modeling profile (SysML) is that it allows both system design and safety analysis to be performed in a single tool. It removes the need to carry out model-to-model transformations from a systems engineering domain to the safety domain. The avoidance of this process decreases the likelihood of errors resulting from the transformation process and can save time.

Nevertheless, as acknowledged by both papers, the auto-generated safety analysis artefacts from these models cannot be used as the final source of truth and must be reviewed by safety engineers. As defined by aerospace standards, safety certification must still be carried out independently; hence a model-based approach can only support the generation of preliminary analysis and cannot be viewed as a substitute method.

Identifying Objectives

From the above analysis, it is identified that there are two key functionalities omitted from existing methodologies. The first missing functionality is the lack of an active traceability between model elements and requirements. Safety requirements are an essential part of the safety assessment process and hence must be updated continuously as safety analysis is performed. Within the aerospace industry, the management of system requirements and validation and verification activities is typically done using requirements modules. These are often stored in a separate requirements management tool such as DOORS. Of the two methods analyzed above that do consider requirements, they are only stored as text within the system model. Due to the lack of traceability between these textual requirements in the model and the requirements modules, modifications or creation of new requirements due to safety analyses are not reflected and would require manual tracing. The second aspect missing in existing methodologies is the generation of Functional Hazard Assessment (FHA) documents. This is crucial as according to the ARP4761 standard, it is the first step in the Safety Assessment Processes that is performed on new or modified aircraft programs.

As a result of the research and analysis into related works, the proposed methodology, which will be detailed in the following section, should accomplish the following objectives:

1. Automatic generation of FHA, FMEA and FTA
2. Propagation of manual edits in the generated safety analysis artefacts back into the model
3. Traceability of model elements to requirements modules

Proposed Approach

In this section, a methodology for implementing MBSA involving the extension of the SysML profile will be introduced. This methodology is designed to provide a more complete approach to integrating safety analysis and is built on leveraging existing SysML functionalities, hence giving it the flexibility to be used with different modeling tools.

Overall Process

The key element of this approach is the extension of the SysML profile. This allows the safety model containing all relevant safety data to be stored together with the system model in what is referred to in Figure 2 as the architectural model. During the design phase, it is crucial that safety analysis is performed on the latest version of the system design. It is also equally as important that the analysis results, along with any newly generated safety requirements are incorporated back into the design. To facilitate this, it is proposed that a Safety Application is created which has the ability to access, change, add to and delete model elements in the architectural model through an API. The Safety Application is designed to be capable of automatically generating FHA, FMEA and FTA using safety data extracted from the model. Similarly, changes made by safety engineers to these documents can be propagated back into the model via the API without them having to edit the model itself. As modifications by either system or safety engineers must first pass through the application, changes can be tracked easily, thus supporting change management activities. The benefits of having the Safety Application is that even though some modeling tools have the functionality of generating tables directly, the application can format the tables in a way that is familiar to safety engineers. In addition, using the application can enhance user experience as it provides a common platform where safety engineers can obtain all the information they require, instead of going through multiple tools.

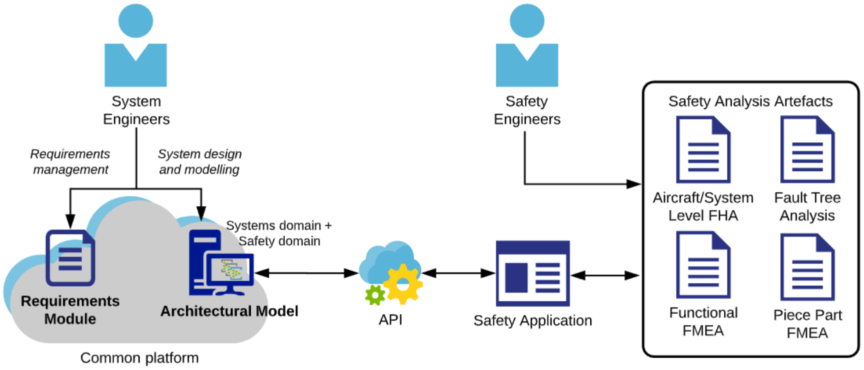


Figure 2: Overall Process of Proposed Methodology

On the other hand, in order to maintain traceability between model elements and safety requirements, the requirements module and architectural model will need to be hosted on a common platform, as shown in Figure 2. This allows requirements to be directly linked to the relevant model element, making it easier to manage and trace compliance and for modifications made in either tool to be automatically reflected. A model checker tool can also be used within the architectural model to check that all safety requirements in the module are addressed by a model element. Results from the model checker can either be exported to the common platform through the API or kept within the modeling environment, depending on the users' preferences.

Tools Selection. For any organization to deploy this methodology, an assessment of tools must first be performed to identify a tool set that best fits with the organization's needs. For illustrative purposes in this paper, the IBM Engineering Lifecycle Management (ELM) tool set is selected, however it should be acknowledged that other tools can be used.

Using the IBM ELM tool set means that architecture models are developed in IBM Engineering Systems Design Rhapsody (Rhapsody) and requirements modules are managed in IBM Engineering Requirements Management DOORS Next (DOORS Next). These IBM ELM tools are designed to work together to offer full traceability across the engineering process, hence both Rhapsody and DOORS Next are hosted on the IBM Jazz platform. This allows requirements to be automatically loaded into the architectural model, and if any requirements are added or updated as a result of safety analysis, they can be immediately synchronized into the model as well. System engineers are then notified of the changes and can then verify if the design still satisfies all the requirements. In addition, Rhapsody also comes with a Java API and an internal model checker which allows the proposed methodology to be implemented smoothly.

Metamodel Overview

As mentioned previously, SysML is used as the basis for metamodel extension and hence the notations used to illustrate the extension will also adhere to SysML definitions. SysML is chosen as the basis for the proposed methodology since it is commonly used by companies within the aircraft industry and it is "the de facto standard architecture modeling language for MBSE applications" (SysML Forum 2003). In this section, only elements that are introduced as part of the safety profile in addition to the existing standardized SysML metamodel will be discussed. All other elements and relations will remain the same and these can be found in specifications documented online (Object Management Group 2019) or in books such as *A Practical Guide to SysML* (Friedenthal, Moore & Steiner 2014).

To construct new elements, stereotypes are created by either extending existing UML/SysML meta-classes or specializing from existing elements as shown in Table 2.

Table 2: New Metamodel Elements

New metamodel elements	Relation to UML/SysML
System	Stereotype of SysML Block
Component	Stereotype of SysML Block
Function	Stereotype of SysML Operation
FailureCondition	Stereotype of SysML Property
FailureMode	Stereotype of SysML Property
FaultTreeDiagram	Specialization of UML BehaviorDiagram
NominalState, DegradedState, FailedState	Specialization of UML State

These newly introduced metamodel elements are created for the purpose of capturing safety data within the architectural model and to support the automatic generation of FHA, FMEA and FTA. This also allows for FTAs to be stored within the model and for dysfunctional behavior to be modeled in State Machine Diagrams. However, it is worth mentioning that since safety certification must still be carried out independently as per aerospace standards, the generated safety analysis artefacts would only act as a preliminary document and would still need to be verified by safety experts.

An overview of the relationships among these new elements is modeled by the class diagram in Figure 3. The key specifications represented by this diagram are:

- A system can have functions, failure conditions, failure modes and states
- A system can be composed of components, and each component can have functions, failure modes and a failure effect code
- Each failure condition or failure mode is allocated to a function
- Each failure condition or mode has an association link to a state
- There are three types of states: nominal, degraded and failed

It should also be noted that the metamodel element 'System' can be applied to either the entire aircraft itself or a system of the aircraft (e.g. landing gear system). This solely depends on the level of abstraction at which modelling is being performed.

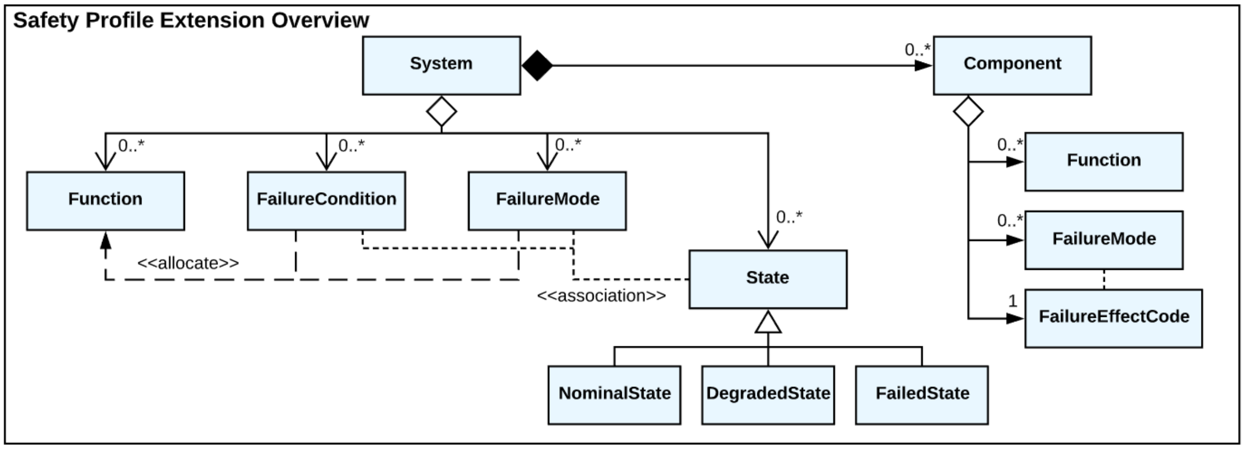
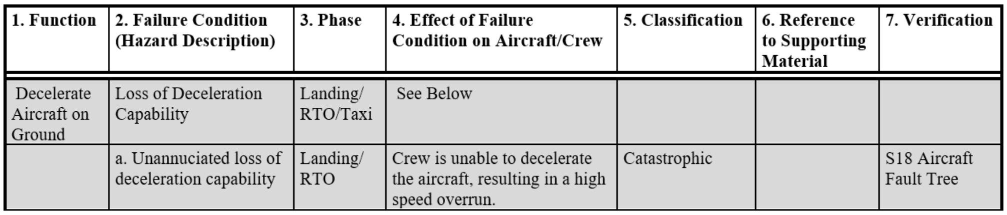


Figure 3: Overview of Relationships among Safety Profile Elements

FHA Generation

Functional Hazard Assessments (FHAs) are performed at both the aircraft and system level of integration. The aircraft level FHA assesses the basic functions of the aircraft and the associated potential failure conditions, whereas the system level FHA considers a failure or a combination of system failures that affect an aircraft function. The standard form of an FHA table as defined in ARP4761 is shown in Figure 4. The fields shown here were used as the basis for the extension of the profile. The elements in the extended profile that are thus relevant to the generation of FHAs are depicted in Figure 5.



1. Function	2. Failure Condition (Hazard Description)	3. Phase	4. Effect of Failure Condition on Aircraft/Crew	5. Classification	6. Reference to Supporting Material	7. Verification
Decelerate Aircraft on Ground	Loss of Deceleration Capability	Landing/ RTO/Taxi	See Below			
	a. Unannuciated loss of deceleration capability	Landing/ RTO	Crew is unable to decelerate the aircraft, resulting in a high speed overrun.	Catastrophic		S18 Aircraft Fault Tree

Figure 4: Partial Aircraft FHA, with Example Entry, Adapted from ARP4761 (SAE International 1996)

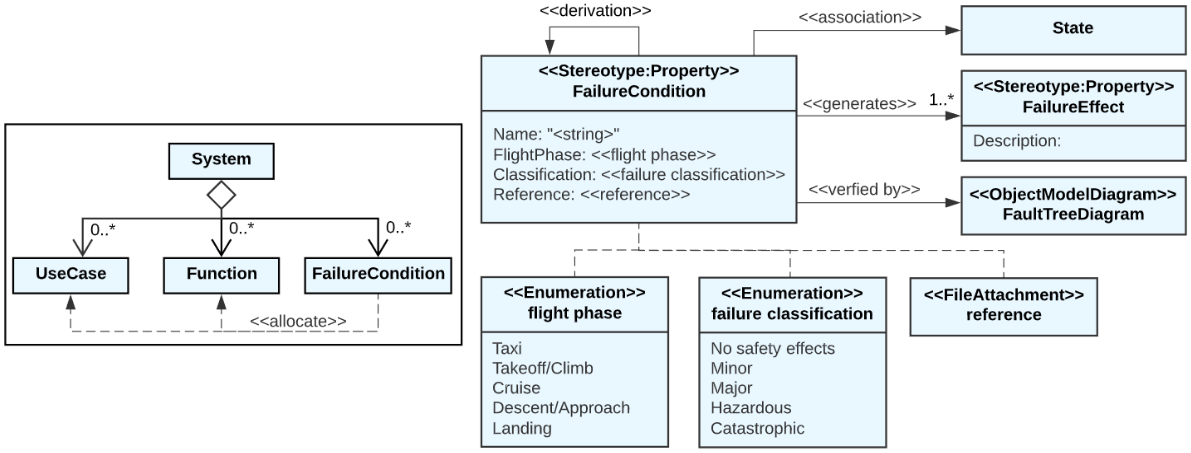
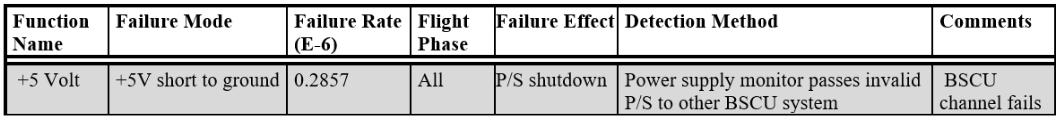


Figure 5: Metamodel of Extended Profile for FHA Generation

At the aircraft level, failure conditions are allocated to UseCases. All high-level functions at the aircraft level would involve external actors, and thus in accordance with SysML definitions, it is a UseCase and not a Function. On the other hand, at the system level, failure conditions belonging to the system of interest would be allocated to Functions. For each use case or function of interest, the information needed to populate each row of the FHA table is extracted from the FailureCondition(s) allocated to it. Sub-conditions can also be modeled and extracted with the derivation link. The attributes within the FailureCondition property can then be used to populate the rest of the table. The Flight Phase and Failure Classification entries are typically limited to a certain number of options, and thus enumerations are used for these attributes. This helps to enforce consistency in how the table is populated, and depending on the system of interest being modeled, more specific phases such as 'Rejected Take-off' or 'Take-off to Rotation' can be included in addition to what is specified in Figure 5. As the failure effect applies to and is generated from a specific failure condition, the FailureEffect property is connected with a 'generates' link. Finally, to provide verification evidence, the failure condition is verified by a Fault Tree Diagram.

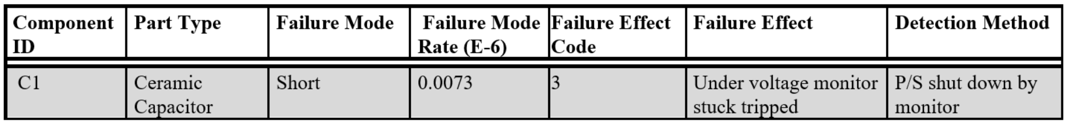
FMEA Generation

As described in the ARP4761 standard, there are two types of FMEAs: Functional FMEA and Piece Part FMEA. The layouts for these are displayed in Figure 6 and Figure 7 respectively. The information required for both FMEA types are very similar as the only difference is that the Functional FMEA indicates the flight phase at which the function of interest is operating in, while the Piece Part FMEA has a ‘Part Type’ column to indicate the component part type and a ‘Failure Effect Code’ that is specific to the component of interest.



Function Name	Failure Mode	Failure Rate (E-6)	Flight Phase	Failure Effect	Detection Method	Comments
+5 Volt	+5V short to ground	0.2857	All	P/S shutdown	Power supply monitor passes invalid P/S to other BSCU system	BSCU channel fails

Figure 6: Partial Functional FMEA for BSCU Power Supply, with Example Entry, Adapted from ARP4761



Component ID	Part Type	Failure Mode	Failure Mode Rate (E-6)	Failure Effect Code	Failure Effect	Detection Method
C1	Ceramic Capacitor	Short	0.0073	3	Under voltage monitor stuck tripped	P/S shut down by monitor

Figure 7: Partial Piece Part FMEA for BSCU Power Supply Monitor, with Example Entry, Adapted from ARP4761

As in the case of the FHA, the fields required for both FMEAs were used as the baseline for what additional elements were needed, and the corresponding metamodel is specified in more detail in Figure 8.

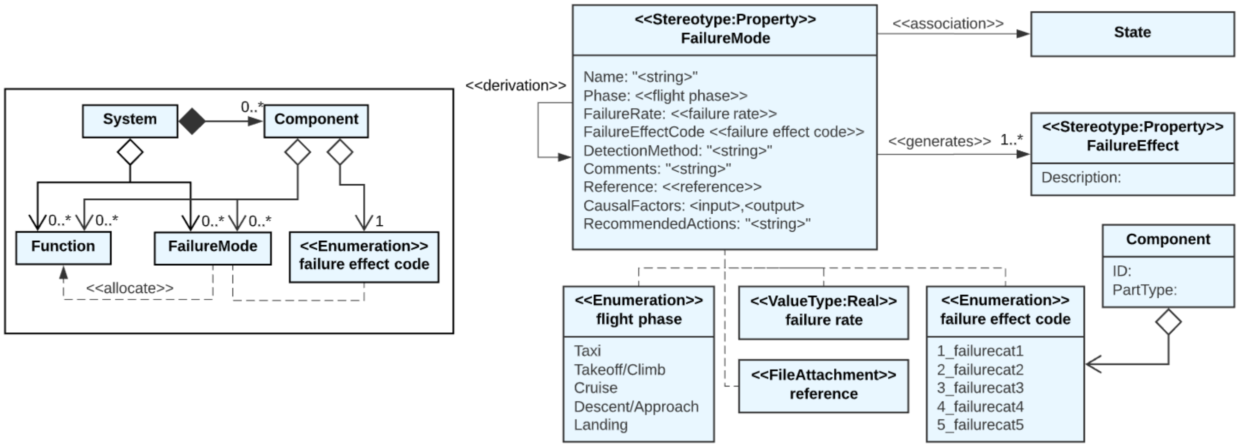

System
0..*
Component
0..*
0..*
0..*
0..*
1
Function
FailureMode
<<Enumeration>> failure effect code
<<allocate>>
<<derivation>>
<<Stereotype:Property>> FailureMode
Name: "<string>"
Phase: <<flight phase>>
FailureRate: <<failure rate>>
FailureEffectCode <<failure effect code>>
DetectionMethod: "<string>"
Comments: "<string>"
Reference: <<reference>>
CausalFactors: <input>,<output>
RecommendedActions: "<string>"
<<association>>
State
<<generates>> 1..*
<<Stereotype:Property>> FailureEffect
Description:
Component
ID:
PartType:
<<Enumeration>> flight phase
Taxi
Takeoff/Climb
Cruise
Descent/Approach
Landing
<<ValueType:Real>> failure rate
<<FileAttachment>> reference
<<Enumeration>> failure effect code
1_failurecat1
2_failurecat2
3_failurecat3
4_failurecat4
5_failurecat5

Figure 8: Metamodel of Extended Profile for FMEA Generation

With the introduction of the FailureMode stereotype, the system model can thus store all the relevant safety information needed to produce Functional and Piece Part FMEAs. For the Functional FMEA, the focus is on the failure modes of functions, thus the table is populated by extracting all the FailureModes allocated to the function of interest. To access sub-functions, a traversal through the relevant activity diagrams, which effectively gives a functional breakdown of the high-level system function, can be performed. The attributes stored in each FailureMode can then be used to populate the rest of the row entry. Once again, FailureEffects are connected to FailureModes with a ‘generates’ link. There is also an ‘association’ link between a FailureMode and State. As shown in Figure 3, there are three types of states: nominal, degraded and failed. By introducing this specification, the state that the system enters when the failure mode occurs can be traced and represented within a state machine diagram for dysfunctional analysis.

Piece Part FMEAs focus on the failure modes of specific parts, hence all FailureModes belonging to the component of interest are extracted from the model to populate the table. The ‘Component ID’ and ‘Part Type’ are stored as an attribute of the Component stereotype. The FailureEffectCode attribute is an enumeration with literals that are specific to the component and can be pre-specified, hence it is aggregated to and contained within the component itself and not the failure mode. In addition to the fields listed in the standardized FMEA tables, two more attributes are introduced to the FailureMode stereotype: RecommendedActions and CausalFactors. The former allows both system and safety engineers to suggest measures to apply corrective action to the associated failure mode and identify who is responsible, while the latter lists what flows through the input and output ports of the component. The idea here is to provide safety engineers a starting point when investigating possible causes for failure.

FTA Generation

To construct FTAs, a Safety Application can be written that uses the API to access, extract and manipulate model elements in the architectural model. With this capability, a fault tree diagram can be populated using extracted information from the internal block diagram of the system or component to which the top-level event/function belongs.

To begin with, the two simplest failure cases are taken into consideration: an internal part error and an input error. Translating this to a fault tree graph means that for a certain output error, the fault can be caused by either an internal failure of the component OR an input error to the component. For an internal block diagram that has redundant components, an AND logic gate can be used to represent that all redundant components must fail for the next component to fail. These rules are best represented in graphical form and are hence shown in Figure 9. A simple internal block diagram with entry, exit and redundant components is shown on the left, while the corresponding fault tree is shown on the right.

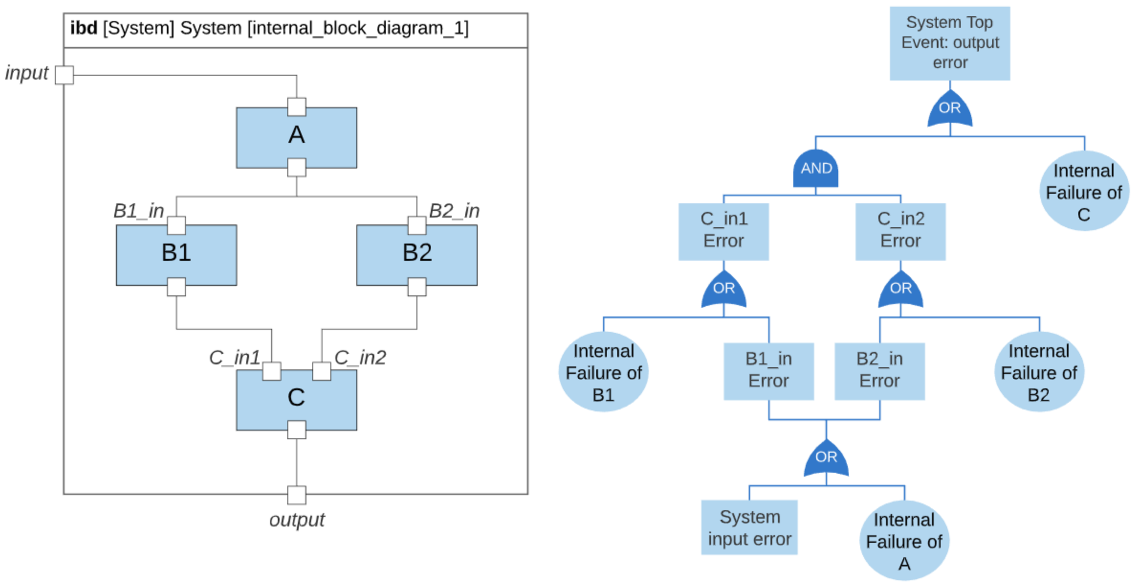


Figure 9: Internal Block Diagram with Corresponding Fault Tree

For a given failure mode or condition (which serves as the top event of the FTA), the Safety Application can identify the system/component it belongs to and traverse into the internal block diagram of the system/component. Starting from the output port of the diagram, the application can be coded to navigate through the diagram towards the input port and populate the fault tree diagram with the possible failure events using AND and OR logic gates.

A complete FTA typically contains logic gates and relations of a higher complexity, and hence the output produced by the application can only be used as a preliminary diagram. Nevertheless, this diagram can then be stored as a FaultTreeDiagram in the architectural model and be viewed and edited by both system and safety engineers.

Requirements Management Integration

In order to integrate requirements management into this methodology, the proposed profile extension introduces a direct connection between safety requirements and model elements. As depicted in Figure 10, 'satisfy' links are introduced as there are many ways in which a requirement can be satisfied. For example, it could be by simply adding a component to the design, proving that a failure mode has a certain failure effect or specifying if a failure condition has a certain classification. Furthermore, to account for possibilities where safety requirements are developed or updated as a result of either safety or model analysis, the profile also allows for 'derivation' relationships from a requirement to any model element. A model checker can then be set up to verify that each and every requirement has a 'satisfy' link to verify that all requirements within a module have been considered and accounted for.

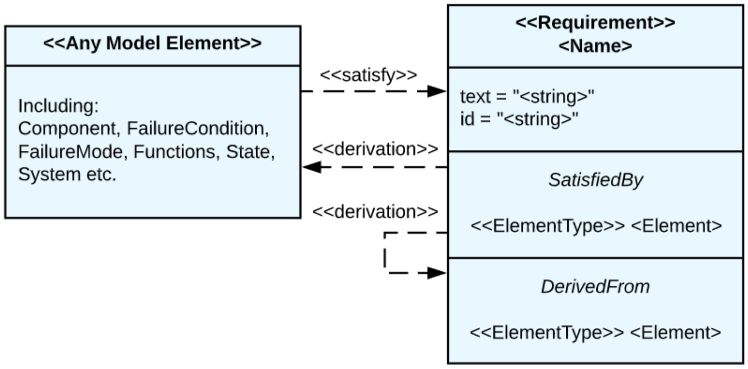


Figure 10: Requirements Traceability Links

As discussed previously, the IBM ELM tool set is well suited to the proposed methodology. By storing requirements in DOORS Next, which is a requirements management tool hosted on the IBM Jazz platform, requirements can be reused in more than one module. This is particularly useful for aircraft systems as requirements are often relevant to multiple modules. The tool also has built-in automation that updates all relevant modules when a requirement is changed, enables engineers to understand how requirements are interconnected through a dependency tree and see how changing one requirement can affect other existing ones. Via the Rhapsody Model Manager application, DOORS Next can be linked with Rhapsody, allowing modules to be loaded into the Rhapsody project (IBM Corporation 2014). The connection can be configured such that requirements within the Rhapsody model are automatically updated if requirements are modified.

Most importantly, active traceability links can be generated between model elements in Rhapsody and requirements in DOORS Next, a capability missing in existing methodologies.

Summary of Proposed Approach

In summary, the proposed methodology consists of two key parts: an extension of the SysML profile to include safety data; and, the introduction of a Safety Application that extracts data from the system model to generate safety analysis artefacts. The extended profile is introduced such that safety-relevant properties can be stored within the architectural model in order to facilitate the automatic generation of safety artefacts. The Safety Application then pulls data from the system model in real-time and automatically generates FHAs, FMEAs and FTAs. Manual edits to these safety analysis artefacts can be easily propagated back into the architectural model, which helps to maintain consistency between the two domains. Requirements can also be loaded directly into the architectural model, and by doing so, satisfaction and derivation relationships can be created between requirements and model elements allowing for increased traceability.

Conclusion and Next Steps

With the increasing complexity of aircraft systems in recent years, inconsistency between system and safety domains is a common challenge. This paper proposes a methodology that addresses this problem by integrating safety analysis into Model-Based Systems Engineering activities. By using the SysML language as a basis for development, the extended profile that has been introduced in this paper can be readily integrated into any existing modeling framework and software.

This paper provides an outline of how this methodology should be implemented; thus further work will be conducted to develop this idea such that it reaches a level of maturity that can be deployed in an industry setting. This includes conducting a review with industry experts to obtain feedback and carrying out a case study to demonstrate the effectiveness of this methodology and the added value it brings. Additionally, in order to apply this method within an organization, a more detailed implementation plan will need to be produced once a software tool is specified, along with any minor modifications needed to comply with the organization's existing framework.

It is also important to reiterate that the proposed methodology only acts to support the generation of preliminary safety analysis, and hence cannot be used as a substitute for aircraft safety certification.

Ultimately, by introducing the proposed methodology, safety considerations will be incorporated in earlier phases of development and hence late design changes will be avoided. System and safety development processes will also become more efficient and reliable, reducing both development times and error proneness when performing safety assessment. This will empower companies to become more competitive and be able to deliver safer systems, and thus resulting in the production of safer aircraft.

References

Baklouti, A, Nguyen, N, Mhenni, F, Choley, JY & Mlika, A 2019, ‘Improved safety analysis integration in a systems engineering approach’, Applied Sciences, vol. 9, no. 6.

Batteux, M, Prosvirnova, T, Rauzy, A, 2017, AltaRica 3.0 Language Specification, AltaRica Association, viewed 25 March 2021, <https://www.openaltarica.fr/docs/AltaRica%203.0%20Language%20Specification%20-%20v1.1.pdf>.

Bozzano, M, Cimatti, A, Lisagor, O, Mattarei, C, Mover, S, Roveri, M & Tonetta, S 2015, ‘Safety assessment of AltaRica models via symbolic model checking’, Science of Computer Programming.

David, P, Idasiak, V & Kratz, F 2010, ‘Reliability study of complex physical systems using SysML’, Reliability Engineering and System Safety, vol. 95, no. 4, pp. 431–450.

Friedenthal, S, Moore, A & Steiner, R 2014, A Practical Guide to SysML, Morgan Kaufmann, 3rd edn., Waltham, MA.

Gabbai, JME 2005, Complexity and the Aerospace Industry: Understanding Emergence by Relating Structure to Performance using MultiAgent Systems, The University of Manchester.

Helle, P 2012, ‘Automatic SysML-based safety analysis’, ACES-MB ’12: Proceedings of the 5th International Workshop on Model Based Architecting and Construction of Embedded Systems.

Howard, CE 2018, MBSE is transforming aerospace engineering, systems integration, SAE International, viewed 23 January, 2020, <https://www.sae.org/news/2018/10/mbse-is-transforming-aerospace-engineering-systems-integration>.

IBM Corporation 2014, Rhapsody Model Manager, IBM Corporation, viewed 13 March, 2020, <www.ibm.com/support/knowledgecenter/ssymrc_6.0.6/com.ibm.rational.rmm.overview.doc/com.ibm.rational.rmm.overview.doc_eclipse-gentopic1.html>.

INCOSE 2015, Systems Engineering Handbook - A Guide for System Life Cycle Processes and Activities, 4th Ed., Wiley, San Diego, CA.

Leveson, NG 2018, ‘Safety Analysis in Early Concept Development and Requirements Generation’, INCOSE International Symposium.

Li, Y, Gong, Q & Su, D 2014, ‘Model-based system safety assessment of aircraft power plant’, Procedia Engineering, vol. 80, Elsevier B.V., pp. 85–92.

Mhenni, F 2014, 'Safety analysis integration in a systems engineering approach for mechatronic systems design', PhD thesis, Ecole Centrale Paris, Paris.

Mhenni, F, Nguyen, N & Choley, JY 2018, ‘SafeSysE: A Safety Analysis Integration in Systems Engineering Approach’, IEEE Systems Journal, vol. 12, no. 1, IEEE, pp. 161–172.

——— 2013, ‘Towards the integration of safety analysis in a model-based system engineering approach with SysML’, Fifth International Conference Design and Modeling of Mechanical Systems, vol. 1, pp. 61–68.

Object Management Group 2019, OMG Systems Modeling Language Version 1.6.

SAE 2010, Guidelines for Development of Civil Aircraft Systems, (ARP4754A), SAE Aerospace.

——— 1996, Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airbone Systems and Equipment, (ARP4761), SAE International.

Stewart, D, Liu, J, Cofer, D, Heimdahl, M, Whalen, MW & Peterson, M 2019, Architectural Modeling and Analysis for Safety Engineering (AMASE).

SysML Forum 2003, What is SysML? - What You Need to Know, SysML Forum, viewed 27 March, 2020, <https://sysmlforum.com/>.

Yakymets, N, Jaber, H & Lanusse, A 2013, ‘Model-based system engineering for fault tree generation and analysis’, International Conference on Model-Driven Engineering and Software Development, pp. 210–214.

Biography

Kimberly Lai is a Master’s student at the University of Toronto and is currently conducting research in Model-Based Systems Engineering and Safety Analysis. She holds a Bachelor degree in Engineering Science, majoring in Aerospace Engineering from the University of Toronto and has experience with implementing MBSE at Safran Landing Systems. The first iteration of this paper was written as part of her undergraduate thesis under the supervision of Dr. Olechowski.

Thomas Robert is a Systems Architect and MBSE Deployment Lead at Safran Landing Systems, developing integrated landing gear systems for business, commercial and military aircraft. He has more than 10 years of experience in systems engineering on highly integrated and complex systems, for aerospace and railway projects, with a specialization in operational deployment of MBSE process, methods and tools. He holds Bachelor and Master degrees in Mechanical Engineering from the University of Poitiers and INSA Rouen Normandie.

David Shindman is a Systems Architect and Senior Expert at Safran Landing Systems, developing integrated landing gear systems for business, commercial and military aircraft. He has over 30 years of experience in aviation including management of systems engineering and airworthiness. He has also developed automatic train control systems for rapid transit. He holds Bachelor and Master degrees in Aerospace Engineering from the University of Toronto and a Diploma in Fluid Dynamics from the von Karman Institute for Fluid Dynamics.

Dr. Alison Olechowski is an Assistant Professor in the Department of Mechanical & Industrial Engineering and the Troost Institute for Leadership Education in Engineering (ILead) at the University of Toronto. Dr. Olechowski and her team study the processes and tools that teams of engineers use in industry as they design innovative new products. She has studied engineering products and projects in the automotive, electronics, aerospace, medical device and oil & gas industries.