

**Calhoun: The NPS Institutional Archive**  
**DSpace Repository**

---

NPS Scholarship

Publications

---

2017

# Systems Architecture in Failure Analysis (Applications of Architecture Modeling to System Failure Analysis)

Rabikur, Alex; Giammarco, Kristin; O'Halloran, Bryan

IEEE

---

Rambikur, Alex, Kristin Giammarco, and Bryan O'Halloran. "Systems architecture in failure analysis (Applications of architecture modeling to system failure analysis)." 2017 12th System of Systems Engineering Conference (SoSE). IEEE, 2017. <https://hdl.handle.net/10945/63783>

---

This publication is a work of the U.S. Government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States.

*Downloaded from NPS Archive: Calhoun*



Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

**Dudley Knox Library / Naval Postgraduate School**  
**411 Dyer Road / 1 University Circle**  
**Monterey, California USA 93943**

<http://www.nps.edu/library>

# *Systems Architecture in Failure Analysis*

## *(Applications of Architecture Modeling to System Failure Analysis)*

Alex Rambikur, Kristin Giammarco and Bryan O'Halloran

Department of Systems Engineering  
Naval Postgraduate School  
Monterey, CA, USA  
sarambik@nps.edu

**Abstract**— This paper describes research into the applicability and value of architecture modeling techniques in conducting failure analysis on a system that is currently in service. Many Systems of Systems (SoS) currently in service did not include architecture modeling in their design and integration phases. In the absence of architectural models, knowledge of critical items, interfaces, and interactions, is difficult to learn, communicate, and maintain. Failure analysis, particularly for SoS, requires comprehensive system knowledge to produce accurate and timely results. This paper contributes specifically by summarizing existing model-based failure analysis research in terms of applicability to in-service systems and by focusing on Fault Tree Analysis (FTA) and Failure Modes Effects Analysis (FMEA). Much of the existing research in this area is either specific to design or development phases of system lifecycle, or presupposes that architectural models exist, and does not approach the topic from the perspective of value and practicality to an in-service system. Firstly, a general analysis of the commonality of principles, methods, and resources used for failure analysis with those of system architecture modeling is presented. The second contribution is a review of system and SoS modeling techniques and current research as applied to these failure analysis methods. The paper concludes with a summary of model applicability to FTA and FMEA, and a brief analysis of the studied methods, which will guide future research and evaluation of architecture modeling-based methods for failure analysis.

**Keywords**— *behavior modeling; failure modes; failure analysis; MBSE, FTA, FMEA*

### I. INTRODUCTION

#### A. Background

For many in-service systems, where original design did not include architecture modeling, knowledge of the system's structure, critical items and interfaces, dependencies, and context is difficult to acquire, communicate, and maintain. In the case of many navy weapon systems, this information is contained only in the drawings and specification documents. The format of the information limits its practical understandability and use. In many cases, learning about the system and its behavior through direct observation during operation is not possible and analysis of operational data and hardware after operation is neither readily available nor comprehensive, as it is limited to forensic interpretation of the evidence. As a result, identification of critical items and

interfaces in conjunction with an understanding of expected behavior is currently only available through labor intensive research, brainstorming sessions, and expert judgment limited to those with extensive experience. The current challenge to those investigating system failures is that the resulting fault trees and effects analyses require a long time to create, and are prone to inaccuracies resulting from human error.

The usefulness of architecture models, including both physical and functional hierarchy models and behavior models, to describe and communicate information about structure, critical items and interfaces, dependencies, and context is well established [1]. Architectural models, when developed using automated tools, also allow for comprehensive analysis of the system and its behavior in a manner that is not limited in the way that direct observation is. If shown to be advantageous, implementation of a model-based approach to the failure analysis processes could improve the accuracy and efficiency of investigations into system failures and their resultant corrective actions. However, where architecture models for many of these systems do not already exist, generation of models describing a complex, in-service system is not likely to be considered a priority unless it can be associated with a performance or cost benefit.

#### B. Scope

The scope of this study is limited to reviewing the application of architecture models to classic failure analysis methods, used by a Navy weapon system practitioner, such as fault tree analysis (FTA) and Failure Modes and Effects Analysis (FMEA). Architectural models to be studied for applicability are limited to Physical Architecture Models, Functional Architecture Models, and Behavior Models. Discussion is limited in relevance to in-service systems where original design did not include architecture modeling. Navy weapon systems are operating increasingly as systems of systems and special emphasis is placed on applicability to those systems. This study does not address other methods of failure analysis, nor go into detail with regard to reliability/probability calculations, as these have been covered extensively elsewhere. The general benefits of Model Based Systems Engineering (MBSE) to the system development process are likewise not broadly addressed in this study. This paper focuses specifically on a review of the existing literature

and research, evaluating methods and approaches based on their applicability to the system type, level of fidelity possible, practicality to create/maintain, and benefit to the investigation of SoS failures.

## II. COMMON PRINCIPLES, METHODS, AND RESOURCES

### A. Overview

Failure Analysis is a critical component of system sustainment, which requires application of many of the same elements used in system architecture model development. Common principles such as boundaries, modularity, decomposition, layer hierarchy, and relations; methods such as systematic deductive/inductive reasoning, logic models, and graphic representation; and Resources such as; system descriptions, knowledge of system behavior, and engineering manpower are required for each. The first subsection addresses these commonalities in the context of system failure analysis and introduces the two methods to be used for the comparison. A comparison of FTA and FMEA with system architecture bears out the common elements in subsequent sections.

### B. Failure Reporting, Analysis, and Corrective Action System (FRACAS)

A Failure Reporting, Analysis, and Corrective Action System (FRACAS) is a key element in any Reliability and Maintainability program and its existence is often a requirement for Navy systems [2]. An explanation of the FRACA process is instructive in order to understand the context of failure analysis in systems engineering through the system lifecycle. As stated in Military Standard 2155(AS), “the essence of a closed loop FRACAS process is that failures and faults of both hardware and software are formally reported, analysis is performed to the extent that the failure cause is understood, and positive corrective actions are identified, implemented, and verified to prevent further recurrence of failure [2].” In fact, the purpose of the FRACA in both the development and operations phase is primarily to improve reliability and maintainability of the system in an efficient way. It is easy to understand how the accurate and timely performance of failure analysis is important to the effectiveness of a FRACA.

Fault Tree Analysis (FTA) and Failure Modes and Effects Analysis (FMEA) are standard root cause analysis techniques employed in a FRACA process. Application of these two methods is not universal, and each has its advantages and disadvantages in the context of failure investigation and the Reliability and Maintainability (R&M) FRACAS process. The main advantage of FMEA is for understanding system behavior and quantifying criticality while a disadvantage to the method is that the comprehensive understanding is accomplished by way of a difficult and tedious process. FTA is easily applied to failure investigations where the effect is witnessed and the mode/cause is not known, however the disadvantage of this method is that it is reactive and narrowly focused on a single failure mode. The quality of the result is

also highly dependent on the system knowledge and experience of the persons conducting it. If FMEA and FTA results are incomplete or inaccurate, corrective actions could be implemented which don't address the actual root cause, with unknown consequences to system performance.

Other challenges exist in conducting an accurate and timely analysis using FMEA and FTA. Resources such as time, cost, and engineering manpower often limit the extent to which analysis can be conducted [2][5]. In addition to resource limitations, knowledge/understanding of the system being analyzed affects the accuracy and timeliness of the analysis. FMEA requires a qualified team of individuals with a thorough knowledge of the specified system design and performance [3]. The suggestion that qualified practitioners, with ample knowledge, be the ones who perform analysis of systems failures is due to the breadth of system understanding required to take on formal failure analysis. However, through application of decomposition and layer hierarchy principles, system information is often organized and described in architecture models in such a way that it is possible for those with limited understanding of system elements to leverage that information to conduct these methods effectively. As a result, there is potential to overcome the challenges of resource constraints and lack of knowledge by capitalizing on the common elements in system architecture models. A brief explanation of each of FMEA and FTA is sufficient to emphasize why knowledge of these system architecture elements is required and how architecture model descriptions may serve to inform the process. References commonly used by Navy weapon system practitioners for each these methods were reviewed, and the aspects common to system architecture models are documented here. Additional information on both FMEA and FTA methods is available in great detail in the IEC 60812 Procedure for failure mode and effects analysis (FMEA) and the NASA Fault Tree Handbook, references [3] and [4] respectively.

### C. Failure Modes and Effects Analysis (FMEA)

FMEA is referred to as an inductive method. Inductive methods involve reasoning from a specific individual case to a general conclusion [4], or, in the case of system failure analysis, reasoning from a single component failure to its effect on the system. This reasoning method is the same one underlying the aggregation approach in system architecture modeling. The purpose of this method is generally to identify potential failure modes and their causes and effects on system performance [3]. This allows for failure modes to be assigned severities. A thorough FMEA requires a hierarchal decomposition of the system to identify each of its components in their proper context as starting points. In general, decomposition is not encouraged past the component level for these analyses. Figure 1 demonstrates such a hierarchy, where failure of the transmitter would have subsequent effect on the communication subsystem and ultimately the UAV.

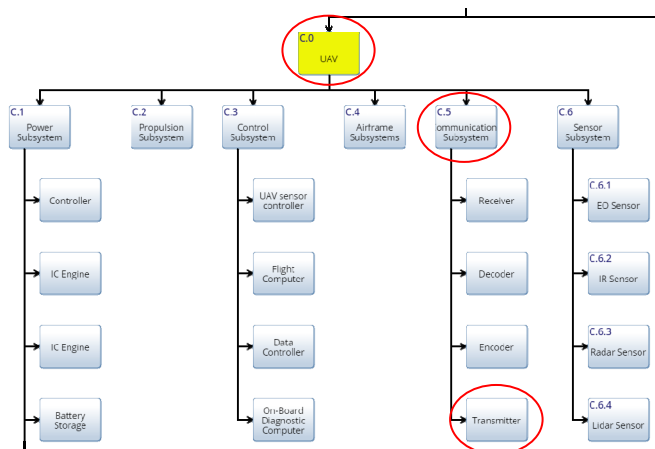


Figure 1 Physical Hierarchy

Environmental operating conditions, including those introduced in SoS, must be specified for proper conduct of FMEA to fully capture component relationships and dependencies [3]. In addition to analysis of the systems, subsystems, and components via a physical hierarchy, functional hierarchy can be analyzed to deduce failure starting points. This hierarchy, or layering, is not just important in identification of starting points, but in the effects analysis that follows. Component failure modes have immediate effects on those elements one level up on the physical and functional hierarchies. When that mode's effect on the higher level is determined to be a failure, the assessment continues to the next higher level using that failure as a mode [3]. As the effect on the system of component failure is realized, knowledge of mission parameters, user need, and stakeholder expectations for system behavior are also required to truly understand the functional impact of the effect to the system or SoS.

#### D. Fault Tree Analysis

FTA is a systematic approach to system failure analysis that is referred to as 'backward stepping,' as opposed to FMEA which is considered 'forward stepping.' As a result, FTA is a deductive approach which means that it starts with a general known or supposed failure, and then reasons (backwards in time) to the specific cause thereof (deducing the mode or mechanism of the failure from the effect) [4]. The concept of decomposition in systems architecting also follows the general to specific method of reasoning. While the product of FMEA can often be purely textual, FTA produces its own graphic model of system behavior, similar to behavior model descriptions used in system architecture. As with FMEA, definition of the system structure is important to the analyst's ability to deduce modes/causes. Definition of system boundaries to include interfaces and their respective input/output relationships allows the analyst to bound the scope. In fact, it is good practice to document the system interfaces and their states on the fault tree [4]. Identification of redundancy in the system architecture is important to FTA as

it affects the logic of the method and its quantitative analysis capability.

NASA uses FTA in its aerospace probabilistic risk assessment (PRA) applications. NASA conducts such activities as System Familiarization, Initiating Event Identification, and Structuring of Scenarios, which all contribute to a compilation of system information (primarily architectural). As seen in Figure. 2, these activities are required before attempting FTA logical modeling at NASA.

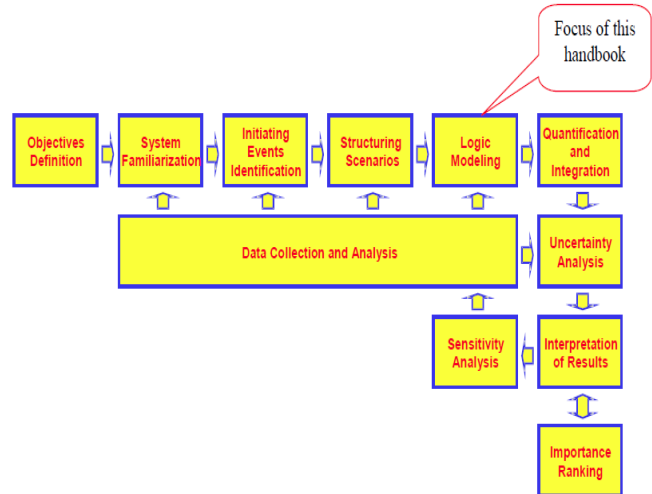


Figure 2 NASA PRA Flow - Adapted from NASA Fault Tree Handbook with Aerospace Applications, by M Stamatelatos and W. Vesely, 2002

#### E. Summary of failure analysis principles, methods, and resources

For each of the discussed failure analysis methods, elements of system architecture such as system elements, logical connections between elements, system redundancies, system context, inputs and outputs of the system and its components, system structure through different operation modes, system boundaries, systems environment and operation, are necessary to understand failures and their effects on the system. For SoS, knowledge of the operating environment is increasingly important, as fault trees often cross system boundaries [4]. Resources are limited, and those common to both, such as engineering manpower and system expertise, should be utilized in a way that avoid duplication of effort. Common principles and methods of representation such as the hierarchy or modularity facilitate simple adaptation from one field to another. Inductive and deductive reasoning methods are likewise applied in similar ways. As a result, hierarchical physical and functional representations of the system would be particularly useful for the FMEA method, while descriptions of relationships in terms of inputs/outputs, dependencies, and logical progression are especially important for FTA.

### III. REVIEW OF SYSTEM (AND SOS) MODELING TECHNIQUES AND CURRENT RESEARCH AS APPLIED TO FMEA AND FTA

#### A. Overview

Model Based Systems Engineering (MBSE), and specifically architecture modeling (physical hierarchies, functional hierarchies, and behavior models), is the formalized application of modeling to system design and analysis. Formality in the modeling process allows for precise definitions, traceability, and more effective analysis. A review of scholarly articles pertaining to the application of MBSE to failure analysis was conducted and is documented herein. This review reveals that the usefulness of the information embodied in system architectural models, specifically physical/function hierarchies and behavior models, for execution of FMEA and FTA, has been well studied by many familiar with the subject. Much of the research shares common references and initial concepts. Specifically, research conducted to date varies regarding the particular use of computer aided architecture modeling programs and languages for identification of failures. These computer aided modeling techniques add the ability to create architecture models in a way that stores the elements and their relationships, such that the system can then be viewed from a different perspective with little to no additional work [6]. Much of the work to date, as described below, involves modeling using specific parameters that better serve the purpose of failure analysis. Applications such as FMEA-like approaches using physical and functional descriptions and automatic generation of fault trees using executable behavior models will be discussed in light of their practicality and advantages to investigation into in-service weapon system failure.

#### B. Counter-Requirement Method

William Schindel presents a methodology by proposing that the existence of functional requirements implies that not meeting those requirements constitutes a failure [7]. This application of architectural modeling to failure analysis was derived from the usefulness of MBSE to the requirements definition process. As part of the systems engineering process, impacts of failure to meet top level systems requirements should have been captured, and the effects ultimately sought in FMEA are easily realized. Architecture modeling enables automatic generation of requirements, and conceivably failures in as much as they are counter-requirements, easily from functional model descriptions of the system. Not only are these failures available, but they are organized according to the requirements hierarchy, allowing the practitioner to understand the direct and indirect effects without additional analysis[7]. Automatic generation would accomplish a great deal of the tedious work involved in FMEA. Instead of manual tracing each failure through the functional hierarchy as with traditional FMEA methods, this computer analysis of functional hierarchy models would quickly generate a presentable output of each of failures and their affected systems for review and analysis by the practitioner[7]. While

this analysis can be accomplished independently of the physical architecture description, the ability to allocate functions and their requirements to the physical assets which perform them sometimes simplifies the analysis indicated for those failures identified (where components chosen have known failure modes and predictable behavior). Rather than a detailed process or tool, Schindel's method is conceptual, based on patterns inherent in system architecture and failure analysis. Schindel argues that this conceptual approach is more powerful than data links between systems engineering architecture modeling tools and Reliability Modeling tools because of its universal nature [7].

#### C. Behavior Model Method

Rao et al., present a methodology which utilizes a simulation based model defined using VHSIC Hardware Description Language (VHDL). The advantage to this methodology is that a single model can be used to study performance, reliability, and behavior of the system hardware it is modeling. VHDL is a very low level description language often used for circuit card assembly descriptions and other similar hardware. While it is indeed a behavior model, the functional and structural hierarchy is not easily communicated by the model, which is mostly intended for simulation. The reliability models synthesized from the VHDL description are Markov models for determining failure probabilities [8]. While such a synthesis was among the first to contribute proof that information can be modeled to both represent architecture and serve FRACAS purposes, the language used for description is not as broadly applicable to either systems engineering or failure analysis.

#### D. UML Dynamic Fault Tree synthesis method

G. J. Pai and J. B. Dugan present a methodology that utilizes Unified Modeling Language (UML) systems models to automatically synthesize fault trees (and Dynamic Fault Trees). The UML is "a standard graphical language used to visualize, specify, construct, and document the artifacts of a software-intensive system" [9]. This methodology was developed based on UML but required specification of additional criteria in order to model redundant elements or capture error propagation potential. This research is considered an improvement on the previous method because the fault trees can be synthesized from functional diagrams and don't rely on generation of the more complex behavior models, although they may be synthesized from behavior models if desired. A customized parser can be used to extract the logical model from the UML description, which the synthesis algorithm uses to generate the fault tree [10]. Advantages to this methodology include the utilization of a widely used and robust modeling language, and exportability to allow analysis of the resultant fault trees in proven reliability software programs [10]. This would streamline failure analysis once the capability was put in place. Disadvantages to this methodology are the degree to which the description language must be specified within the larger

vocabulary of UML and the tendency of its use to be software system centric. An effort to model an existing weapon system within a SoS would certainly be possible, but one might not find the tools to do so intuitive for that application nor the majority of the knowledge about the language geared toward such a system [1]. However, because the key to the model synthesis is largely the customized parser and algorithm, the theory behind the methodology should be applicable to most modeling languages capable of creating functional architecture models, assuming reliability/probability information can be embedded in the element description (metadata) [10]. While automatic generation of FMEA is not addressed here, the methodology would seem to support it.

#### E. Error Libraries Method using AADL

Building on the UML Dynamic Fault Tree synthesis method J. Dehlinger and J.B Dugan present a methodology that was created to derive dynamic fault trees from system models described using Architecture Analysis & Design Language (AADL) (formerly Avionic Architecture Description Language). This contribution is uniquely capable in that the AADL includes software, hardware, and system component abstractions to specify and analyze real-time embedded systems, complex SoS, and specialized performance capability systems, as well as map software onto computational hardware elements [11]. The advantages of AADL are the formality of the syntax used in describing the system elements and their interactions, allowing for consistent and accurate analysis. The graphical view of component or architectural modeling is minimal with this language which could be a disadvantage in communicating certain system aspects. The textual focus can be easily extended through adding vocabulary libraries known as Annexes. This extension allows for flexibility of the language, making it easily adaptable to analysis capabilities such as fault tree analysis and FMEA[11]. This flexibility is advantageous to failure analysis, particularly for those real-time systems that other modeling languages struggle to describe. Another advantage is that very little processing is required to derive the fault tree information in a way that can be used by reliability software programs. Disadvantages include the level of effort required in building the failure annex/library for use with each system element in a complex system, which requires domain specific expertise [11]. Depending on the type and complexity of the system, the apparent impracticality may discourage adoption of this modeling methodology.

#### F. Risk Informed Design using Innoslate

In contrast to the methods using VHDL, UML or AADL, the final approach reviewed was one in which the capability was integrated into the architectural modeling tool. Rafael Perez demonstrated a methodology using the web-based computer aided architectural modeling tool Innoslate. During the course of the demonstration, the physical and functional system architectures are created, from which an FMEA style analysis continues systematically, identifying all possible

failures by analyzing each element, and in turn describing them in their own sequential architecture. The resultant model is a fault tree diagram in which reliability data can be embedded for reliability calculations [12]. Perez also demonstrates that Innoslate can display the failures identified in a hierarchy or in table format, as required for the FMEA method. As with other computer aided modeling tools, Innoslate stores element data in a way that is available to each model created, so less effort is required once physical and function system architectures are created [12]. This approach is uniquely practical because of its integration into one tool or interface. Another advantage to this approach is the use of Lifecycle Modeling Language, which uses common language descriptions to make it intuitive and adaptable [13].

#### G. Ongoing research combining Monterey Phoenix Behavior Modeling with Innoslate Fault Tree Analysis

T. Moulds and K. Giammarco have conducted research into extension of the methodology of Rafael Perez by using Innoslate in combination with Monterey Phoenix. The use of the Monterey Phoenix simulation tool contributes by further exploiting unexpected or emergent interaction behaviors which may have been missed when identifying failures in the Innoslate model from the physical and functional architecture descriptions [14].

#### H. Comparison of Methods

Table 1 summarizes a comparison of the discussed methods based on their applicability to the system type, level of fidelity possible, practicality to create/maintain, and benefit to the investigation of SoS failures. Applicability to SoS addresses the suitability of the method and/or tool to description of SoS architecture, using factors such as the system language and description versatility. Level of fidelity captures the accuracy of the system structure and behavior representation/description for SoS, primarily measured based on description detail required. Practicality is qualitative judgment of how easily the method can be adopted in terms of time investment. Benefit to Failure Analysis is judged on the basis of its contribution to FTA and FMEA, using factors such as level and structure of information communicated.

Table 1 Comparison of reviewed methods

<b>Method</b>	<b>Applicability to SoS</b>	<b>Level of Fidelity</b>	<b>Practicality</b>	<b>Benefit to Failure Analysis</b>
CRM	High	Medium	High	High
VHDL	Low	High	Low	Medium
UML	Medium	High	Medium	High
AADL	High	High	Low	High
LML	High	Medium	High	High

## IV. SUMMARY AND WAY AHEAD

### A. Summary

FMEA and FTA are effective methods for use in failure investigation and the FRACAS process that it belongs to. However, discussion of each method reveals that they each require a high level of system knowledge. These methods have been described as labor intensive and tedious, dependent on the skills and background of those performing them, and difficult to execute as systematically as required. System information available to those involved in investigations is limited to the technical data package (drawings, specifications, and documents).

Discussion into the knowledge requirements of the methods revealed that many of them are known elements of system architecture; system elements, logical connections between elements, system redundancies, system context, inputs and outputs of system and its components, system structure through different operation modes, boundary definitions, interface definitions, triggers, etc. These elements could be described comprehensively for utilization in FTA and FMEA using physical and function hierarchy diagrams and behavior models. Further review of available literature reveals that the advantageous use of architecture models for failure analysis has been recognized by others. Demonstrated applications include FMEA made possible using requirements models, Markov models synthesized from VHDL behavior models, Fault Trees automatically synthesized from UML models, FMEA and fault trees synthesized from AADL models, and finally FTA and FMEA as an architectural model view in Innoslate. A comparison of these methods revealed that while the fidelity of some methods is high, they cannot easily be used to describe SoS, their models are time consuming to create, or the information delivered is not suited to FTA or FMEA on SoS (VHDL, AADL, and UML). In other cases, the method was easy to apply to SoS, requiring less time, but did not represent the system in as high of a level of detail as the others (CRM, LML). In each case, however, system architecture information that is available for other systems engineering purposes is leveraged to provide a more accurate analysis of failures and their effects in a more timely manner than possible otherwise.

### B. Way Ahead

For failure analysis, the primary advantage to leveraging system architecture models, particularly those created with computer aided tools, is the accuracy and efficiency with which the analysis can be accomplished in contrast to manual methods. Other advantages present themselves as well, such as the potential for integration of architecture with reliability, requirements, risk and cost data in one analysis framework. Amongst the methods reviewed, the counter requirement method proposed by Schindel and the use of LML-based

diagrams demonstrated by Perez suggest it is possible that the benefits of FMEA and FTA can be realized within one tool. In the context of an existing SoS, the required up-front investment in the modeling tool and the development of the system models increases the weight of the practicality aspect in application of a method and tool. An intuitive and easily integrated product that provides the needed capability is the most advantageous. Where the examples given in the demonstrated research have used simplified systems and failures to prove the capability, the way ahead must prove its usefulness in a real-world investigation ongoing into weapon system launch failures. For such a model to be advantageous it must be an accurate representation of the technical data package (drawings and specifications), be practical to create, communicate the system structure, function, and behavior to failure analysts, aggregate failures into fault tree formats using common syntax, and identify critical elements and interactions (FMEA).

## REFERENCES

- [1] Maier, Mark W. and Eberhardt Rechtin, 2015. *The Art of System Architecting* (3rd edition)
- [2] Military Standard 2155(AS), *Failure Reporting, Analysis, and Corrective Action System*, (DOD, 24 July 1985)
- [3] IEC 60812 2ed, *Analysis Techniques for System Reliability – Procedure for Failure Modes and Effects Analysis (FMEA)*, (IS 2006), pp. 15-72
- [4] M. Stametelatos, W. Vesely, *Fault Tree Handbook with Aerospace Applications*, NASA, 2002, pp. 1-109
- [5] Reliability Analysis Center, *Failure Reporting, Analysis and Corrective Action System (FRACAS) Assessment – 231-0-6900-146-001*, ITT Research Institute, 2002
- [6] Vaneman, Warren, *Model-Based Systems Engineering De-Mystified, Presentation for Strategic Systems Program*, (NPS 11 August 2016)
- [7] Schindel, W. D. (2010), 10.1.2 Failure Analysis: Insights from Model-Based Systems Engineering. *INCOSE International Symposium*, 20: 1227–1239.
- [8] R. Rao, G. Swaminathan, B. W. Johnson and J. H. Aylor, "Synthesis of reliability models from behavioral-performance models," *Proceedings of Annual Reliability and Maintainability Symposium (RAMS)*, Anaheim, CA, 1994, pp. 292-297.
- [9] G. Booch, J. Rumbaugh, I. Jacobson, *The Unified Modeling Language User Guide*, Addison Wesley, 1999.
- [10] G. J. Pai and J. B. Dugan, "Automatic synthesis of dynamic fault trees from UML system models," *13th International Symposium on Software Reliability Engineering*, 2002. *Proceedings.*, 2002, pp. 243-254.
- [11] Dehlinger, Josh, & Dugan, Joanne Bechta (2008). Analyzing dynamic fault trees derived from model-based system architectures. *Nuclear Engineering and Technology*, 40(5), 365-374.
- [12] Rafael M. Perez. "Application of MBSE to Risk-Informed Design Methods for Space Mission Applications", *AIAA SPACE 2014 Conference and Exposition, AIAA SPACE Forum*, (AIAA 2014-4411)
- [13] How to Perform Reliability, Maintainability, and Availability Analysis with Innoslate. (n.d.). Retrieved September 27, 2016, from <https://www.specinnovations.com/how-to-perform-reliability-maintainability-and-availability-analysis-with-innoslate/>
- [14] Moulds, Tom, & Giammarco, Kristin (2017), "Understanding Faults in System Models: Comparing Classical Methods Versus System Architecture Tools," unpublished