NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A DIGITAL TWIN MODEL-BASED SYSTEM ENGINEERING APPROACH TO FAILURE ANALYSIS FOR AN ENGINE SYSTEM

by

Xue Yong Yap

September 2022

Thesis Advisor: Douglas L. Van Bossuyt
Co-Advisors: Mark M. Rhoades
Jason Bickford (NSWC PHD)

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 2022	3. REPORT TYPE AND DATES COVERED Master's thesis

4. TITLE AND SUBTITLE A DIGITAL TWIN MODEL-BASED SYSTEM ENGINEERING APPROACH TO FAILURE ANALYSIS FOR AN ENGINE SYSTEM	5. FUNDING NUMBERS
6. AUTHOR(S) Xue Yong Yap	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	12b. DISTRIBUTION CODE A

13. ABSTRACT (maximum 200 words)

In every portion of a product's life cycle, system failures can occur. The DOD and defense contractors use failure analysis methods such as Failure Modes and Effects Analysis (FMEA) and Fault Tree Analysis (FTA), which have successfully reduced maintenance downtime and lowered life cycle costs as a result of reducing failures during system operations. However, challenges can hinder the effectiveness of failure analysis and have not been fully addressed due to limitations of current failure analysis and root cause analysis methods. This thesis addresses two of the challenges: (1) incorrectly classified system failures and (2) lack of failure traceability in the system life cycle. A methodology is proposed that incorporates some elements of both FMEA and FTA into a digital twin (DT) by employing the MBSE tool Magic Systems of Systems Architecture (MSOSA) to aid in failure analysis. An MBSE model simulation using an example of an engine is presented to demonstrate the efficacy of the methodology. The results are shown in terms of system operational availability. This research concludes that the approach has potential in emulating the real system's behavior and can offer a more accurate estimated result. However, due to semantic errors in modeling in MSOSA, the model was unable to generate accurate results as intended. The model's semantic errors must be addressed before it can be validated by comparing the results from MBSE model and the corresponding real-world system.

14. SUBJECT TERMS digital twin, model-based system engineering, failure analysis, system engineering, MBSE	15. NUMBER OF PAGES 95
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

A DIGITAL TWIN MODEL-BASED SYSTEM ENGINEERING APPROACH TO FAILURE ANALYSIS FOR AN ENGINE SYSTEM

Xue Yong Yap
Civilian, ST Engineering Land Systems Ltd
BSME, Newcastle University, 2013

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2022

Approved by: Douglas L. Van Bossuyt
Advisor

Mark M. Rhoades
Co-Advisor

Jason Bickford
Co-Advisor

Oleg A. Yakimenko
Chair, Department of Systems Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

In every portion of a product's life cycle, system failures can occur. The DOD and defense contractors use failure analysis methods such as Failure Modes and Effects Analysis (FMEA) and Fault Tree Analysis (FTA), which have successfully reduced maintenance downtime and lowered life cycle costs as a result of reducing failures during system operations. However, challenges can hinder the effectiveness of failure analysis and have not been fully addressed due to limitations of current failure analysis and root cause analysis methods. This thesis addresses two of the challenges: (1) incorrectly classified system failures and (2) lack of failure traceability in the system life cycle. A methodology is proposed that incorporates some elements of both FMEA and FTA into a digital twin (DT) by employing the MBSE tool Magic Systems of Systems Architecture (MSOSA) to aid in failure analysis. An MBSE model simulation using an example of an engine is presented to demonstrate the efficacy of the methodology. The results are shown in terms of system operational availability. This research concludes that the approach has potential in emulating the real system's behavior and can offer a more accurate estimated result. However, due to semantic errors in modeling in MSOSA, the model was unable to generate accurate results as intended. The model's semantic errors must be addressed before it can be validated by comparing the results from MBSE model and the corresponding real-world system.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

BDD	Block Definition Diagram
CBM	Condition-Based Maintenance
DBSE	Document-Based Systems Engineering
DFMEA	Design Failure Mode and Effect Analysis
DoD	Department of Defense
DT	Digital Twin
FDM	Failure-Driven Maintenance
FE	Finite Element
FMEA	Failure Modes and Effects Analysis
FTA	Fault Tree Analysis
IIoT	Industrial Internet of Things
INCOSE	International Council on Systems Engineering
IPDSS	Intelligent Predictive Decision Support System
MBSE	Model-Based Systems Engineering
ML/AI	Machine Learning / Artificial Intelligence
MSOSA	Magic System of Systems Architecture
MTBF	Mean Time Between Failure
MTTF	Mean Time to Failure
OMG	Object Management Group
PM	Preventative Maintenance
RAM	Reliability, Availability, and Maintainability
SoI	System of Interest
SoS	System of Systems
STM	State Machine Diagram
SysML	Systems Modeling Language
TBM	Time-Based Maintenance

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

At the fast pace of technological evolution and constant upgrading of systems or system of systems, the Department of Defense (DoD) or any defense contractor has likely encountered difficulties in keeping up with addressing system failures by using traditional failure analysis methods. Coupled with the lack of trained personnel in failure analysis and understanding the true value of maintenance planning, there is the potential to implement an automated failure modes and effects analysis (FMEA) model with simple computation of the data from the model to aid decision-making in maintenance support.

This thesis aims to understand how a failure occurs via analysis conducted during a corrective or preventative maintenance triggered under a typical operating scenario. In addition, this thesis seeks to aid engineering efforts in managing system failures. This thesis will study, analyze, and discuss the efficacy of using a digital twin (DT) and model-based systems engineering (MBSE) approach for failure analysis. The questions that guide this research are as follows.

Primary Questions:

1. What is an effective system engineering approach to analyze the impact of failures using a DT?
2. How can the DT be used not only in the early stages of design or system development but throughout the entire lifecycle of the system to support failure analysis?
3. How does applying the DT to failure analysis influence important decisions on Preventative Maintenance and Corrective Maintenance tasks?

Secondary Questions:

1. How can the DT model help alleviate the sparing and staffing issues in military context?
2. What are the potential areas for future research with this model development process as baseline?

Using the Dassault Systems CAMEO Magic System of Systems Architecture (MSOSA)

software, this thesis studies failure analysis from a system engineering perspective and builds a System Modeling Language (SysML) model that incorporates failure analysis in a scenario-based case study. Further, the model attempts to compute the operational availability at each level of abstraction.

The model scenario consists of five states which are derived logically for a typical system as shown in Figure 1. For this model, an established design failure mode and effect analysis (DFMEA) of an engine is used as case study and the engine architecture is built based on its system breakdown. This state machine scenario is built mainly at the lowest abstraction level; component level and triggers the operational states of its associated subsystems and system with the uses of parametric diagrams, state machine diagrams and internal body diagrams.

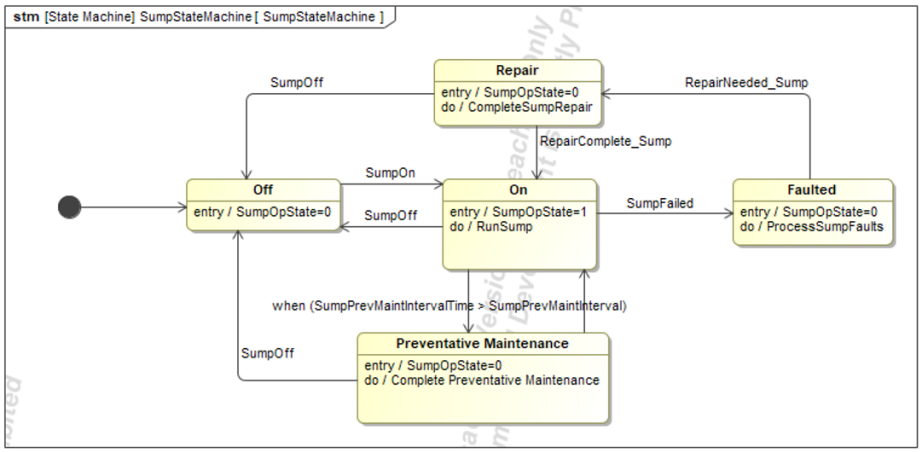

Figure 1. State Machine Diagram of Engine Oil Sump

Findings from the development of the model gave information about how failures happen in the system of interest (SOI). It was shown that the MBSE approach can be used to track, and tag known potential failures to specific parts. The MBSE simulation model can also show how states change within and from the component level to the subsystem level and the system level. This facilitated prediction of the number of failures and the amount of time the system would be able to work. Based on the results of the simulation, once the race condition and counter clock problems (issues encountered during implementation) have been resolved, the model should enable it to plan for maintenance.

Despite the problems encountered during model implementation, the work performed in this thesis enabled the author to understand more about the semantic errors that the model has presented and iterated on the model development process to refine the relations and expected behaviors between each abstraction level. The author strongly believes that once the underlying issues have been resolved, this model follows the actual system's behavior more closely than conventional reliability models and may provide a more precise estimation of system availability. This assertion must be tested by comparing the results of the MBSE digital model of a real-world system to the actual real-world system.

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

The author would like to thank Professor Douglas L. Van Bossuyt for his patient guidance on the entire thesis journey and Professor Mark Rhoades for offering his expertise and advice on MSOSA modeling.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

This thesis proposes a method of conducting reliability, availability, and maintainability (RAM) analysis within a model-based systems engineering (MBSE) environment. An implementation within the Dassault Systems CAMEO Magic System of Systems Architecture (MSOSA) software package is presented and discussed. The initial results of this work show the promise of incorporating RAM into MBSE as a means of better understanding failures and improving system availability.

System failures occur due to a variety of factors including during the design phase of a system design process where such issues as communication errors between different design artifacts in document-based systems engineering processes (DBSE) can cause failures. In DBSE-driven processes, documents are used to transmit important information such as requirements and design specifications which can be challenging to keep up-to-date between different organizations and groups. Furthermore, in a DBSE environment, different lexicons may be used in different organizations compounding the chance of error. One method of reducing the chance of error during design is the model-based system engineering (MBSE) approach which is rapidly replacing the DBSE approach among practitioners [1]. As part of the Department of Defense's (DoD's) digital engineering (DE) strategy, MBSE is being incorporated into the systems engineering process and marks a paradigm shift from the traditional DBSE approach to a MBSE approach which helps in addressing program requirements for acquisition programs [2].

The DoD emphasizes the need for reliability, safety, and risk assessment as part of acquisition program requirements [3]. Furthermore, the DoD guide for achieving reliability, availability, and maintainability (RAM) is a requirement for contractors to follow [4]. One way of ensuring that RAM requirements are being met is through incorporating elements of RAM into MBSE models.

While the shift to MBSE has helped to alleviate some of the sources of failure in the system design process, many system failures occur during system operation. These failures can have many different initiating events such as wear-out, random failure, design flaws, operating

beyond their design basis, and more. In many defense systems and other complex systems, maintenance logs are kept in part to determine the root cause of failures. However, these data can often be incorrectly classified due to the lack of skilled and experienced analysts and proper analysis tools [5]. The limitations of failure analysis via maintenance records have led to recurring failures and a waste of resources in resolving failures at their true root causes. As failure occurs throughout the system life cycle from system design to disposal, it is laborious for one or a team of engineers to track, analyze or even predict failures. Compounding the issue, some failures that occur are not tracked due to the tedious effort of documentation which leads to either improper record keeping or completely absent data, the inability to identify a failure, and not understanding the consequences of failure [5]. To improve failure analysis, the system engineer must view the system life cycle process carefully to manage system failures by reducing the errors in failure classification and enabling traceability in the process.

While a variety of methods exist to conduct failure analysis such as failure modes and effects analysis (FMEA) [6], fault tree analysis (FTA) [7], the Apollo method [8], and others, it can still be challenging to identify the true cause of a failure during maintenance activities. Ideally, methods such as FMEA should be conducted at multiple stages throughout the system development and operations process to identify failure mechanisms and potential deficiencies in the design and operation of a system to rectify the situation. However, this practice has not been feasible using existing analysis techniques because of their cost to implement and the skilled labor requirements to execute the analyses.

Efforts have been made to use MBSE as a tool in understanding system failures during the design process. However, often times syntax errors which are easy to detect using built-in validation tools within MBSE software packages are often missed simply because the built-in validation tools are not used. In some cases, a visual inspection of models would have caught the errors but often times the experienced personnel required to conduct such inspections are unavailable [9]. Hecht et al. [10] highlighted that SysML and MBSE may allow "early and constant interaction between the functional design and safety, reliability, and cybersecurity experts". Their research also stated that SysML enables automated generation of FMEAs as requested and together with other design languages, they have been used to develop automated FMEA creation tools [10].

1.1 Specific Contribution

This paper proposes a method that can aid RAM engineers and design engineers in development of a MBSE model that can analyze the RAM capabilities of any system or sub-system quantitatively throughout the system life cycle and especially during maintenance. The proposed method incorporates the concepts of failure analysis in a system perspective and is implemented in the MSOSA MBSE tool. A scenario-based model is developed to emulate the behavioral, operational, and failure patterns of a simple engine. The result of this method is to improve engineering efforts in failure analysis and RAM capabilities. The proposed method can help overcome the issues of incorrectly classified failures especially during maintenance and help trace failures throughout a system life cycle.

This research aims to develop a MBSE approach to develop a digital twin (DT) for failure analysis. The DT enables personnel to better identify, analyze, and resolve potential system failures that may degrade or disable a system. A use case of a simple engine demonstrates the proposed method by adopting a DT and MBSE-enabled FMEA.

Past research has shown that the RAM capabilities of a system are directly linked to the maintenance cycles [11]. The ability to detect and analyze system failures promptly throughout the system life cycle will aid the DoD, or any organization, to achieve improved RAM such as reduced component downtime and improved system performance.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Background and Related Research

In this chapter, background and related research necessary to understand the proposed methodology is presented. In specific, this chapter covers the following: System Engineering, from an overview to the two main approaches which are DBSE and MBSE; Failure Analysis, focusing on FMEA and FTA; Maintenance Techniques, focusing on Failure-driven Maintenance and Time-driven maintenance; and finally MBSE, focusing on the application of the three MBSE pillars: the modeling tool, the computing language, and the method.

2.1 Systems Engineering

System Engineering is a discipline that addresses stakeholders' needs especially for complex systems through the aid of software and hardware solutions. A classical representation of the systems engineering design process is the V-Model, as shown in Figure 2.1 where a system moves from left to right through the V-Model to eventual operations. System Engineering then branches out into two main approaches which are DBSE and MBSE [12], shown in Figure 2.2.

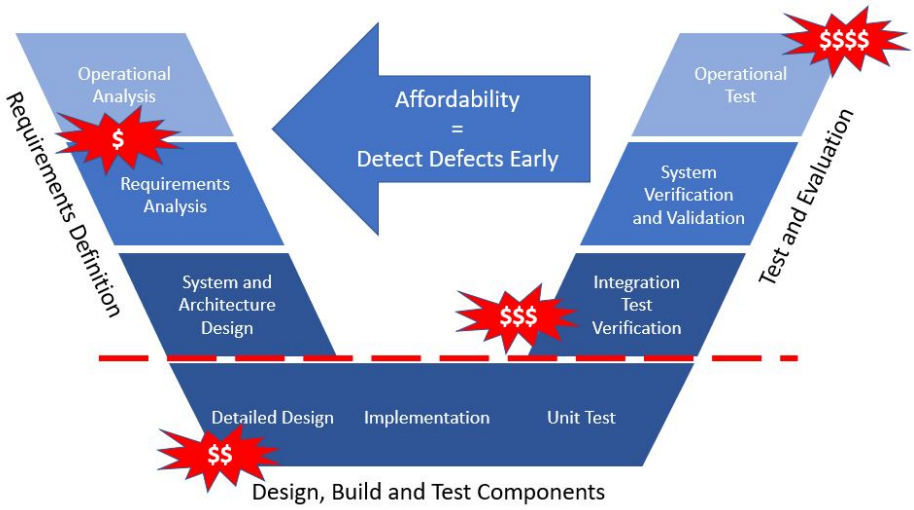

Figure 2.1. V-Model for Failure Detection. Adapted from [13].

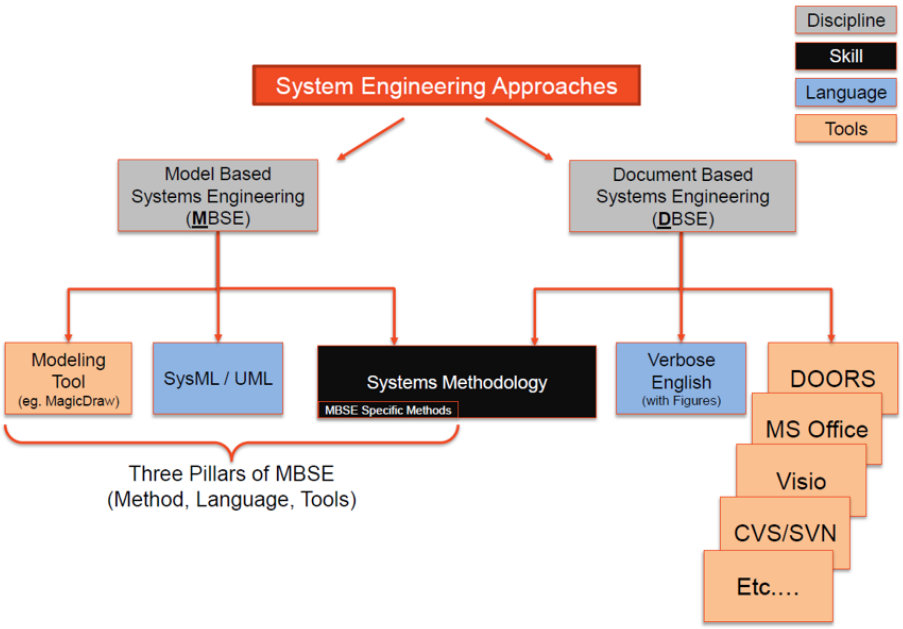

Figure 2.2. System Engineering Approaches. Adapted from [12].

There are limits to DBSE especially as bandwidth and system complexity increase exponentially. In these cases, the DBSE methodology is no longer applicable to large-scale systems. Using DBSE in such a situation quickly introduces many errors during the first half of the systems engineering process (the design phase) and can quickly break down entirely. This limitation was addressed with the implementation of MBSE to supplement DBSE.

2.2 Model-Based Systems Engineering

The International Council on System Engineering (INCOSE) defines MBSE as "the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases" [1]. Figure 2.3 shows how MBSE is applied across the entire system lifecycle. System models link to component models and operational models. In the work presented in this paper, the focus is primarily on component models.

Often times, an architectural framework is implemented within an MBSE environment to help develop specific views and viewpoints that cater to specific stakeholders' and engineers'

needs. The system architectures that are developed within the architectural framework and MBSE environment can be used to support decision-making, identify potential options, elucidate requirements, and help drive the system design to meet the purpose of the system. Then the system design can be undertaken within the MBSE environment where system components are developed, the system configuration is chosen and frozen, the system is constructed, and then later during operations the system is maintained, modified, and reconfigured. Figure 2.4 shows a simplified view of how MBSE can be used in conjunction with an architectural framework to support solving a problem in a system design.

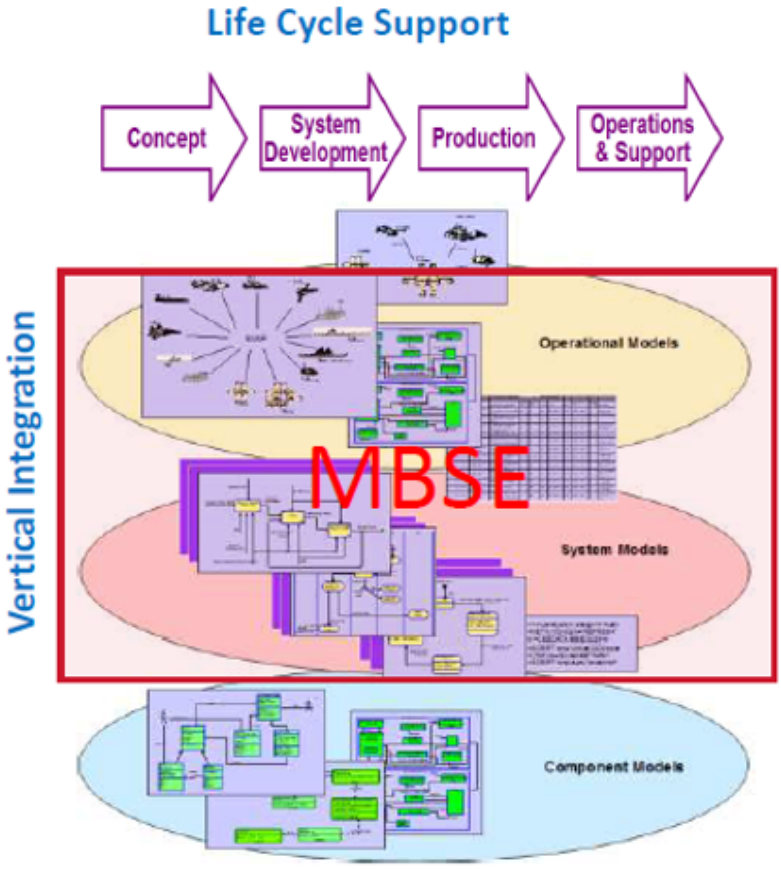

Figure 2.3. Overall Life Cycle MBSE Support. Adapted from [14].

An additional benefit of using an MBSE approach versus a DBSE approach is that traceability of requirements to solutions is significantly improved. This aids in better understanding how the system solves stakeholder needs. Furthermore, MBSE allows for more direct verification and validation of requirements throughout the system lifecycle and design process.

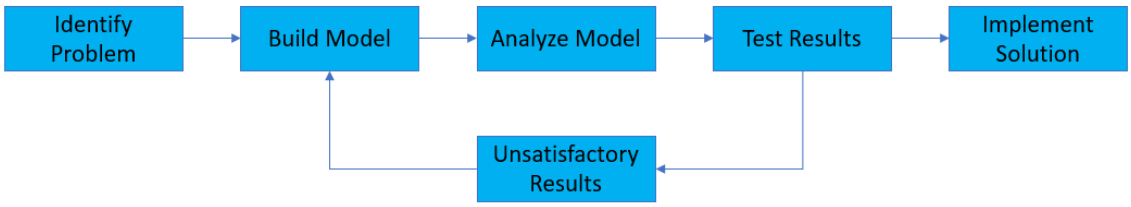

Figure 2.4. MBSE Framework

Bickford et al. [15] stated that MBSE is "the study and application of system models throughout a system's lifecycle" which is applicable to this thesis research because the potential of system modeling being discussed next. In Bickford et al.'s study, the benefits of using MBSE in DT applications are that it allows tracking of performance characteristics even in the early stages of the program, it helps to maximize total utility by linking all analyses performed throughout the lifecycle, and the DT development is not a one-time event that was put off until a systemic problem in operations and maintenance was found. The limitations are the lack of subject matter expertise in specific fields where there is a need for new or different architecture creation from prior system designs, and stakeholder requirements, objectives, and possibly risk assessments must be mature. The proposed method in this paper seek to assess the fidelity of MBSE approach in DT application and minimize the limitations, thus improving upon Bickford et al.'s approach.

Finite element (FE) and physic-based modeling is commonly used in the early system design such as in system development and test and evaluation to aid the decision making process [16]. FE and physics-based modeling is used to verify specific system performance in many design areas such as gas turbine engines [16]. The benefits of FE and physic-based modeling are that observations of specific performance parameters can be integrated directly into a DT, and can be characterized based on the system operating conditions. The limitations are the uncertainty of data which relies heavily on the design setup given the (1) specific operating conditions, (2) design parameters, and (3) measure objectives, and the need for a new model whenever there is a change in any of the three above factors. The method proposed in this paper aims to scope the three factors appropriately to minimize the influence of data uncertainty.

2.2.1 Model-Based Systems Engineering Capabilities and Limitations

In a digital-modeling world, MBSE offers some key advantages over DBSE [17]. First, the ability to build an authoritative source of truth with cross-disciplinary views for the system to address the DBSE limitations of design errors due to different design artifacts. Second, the creation of a common standards-based field for system documentation which can be programmatically verified and validated to eliminate model inconsistencies and to ensure all stakeholders conform to the use of a standard. Third, the shared modeling environment helps system analysis and reduces the amount of aforementioned errors due to DBSE. Lastly, the availability of digitized system data for cross-disciplinary studies enables the consistent transmission of adjustments as well as inclusion of new knowledge and design decisions [17]. However, there is no standard model to capture, export, or import architecturally significant properties (uptime, mean time to failure (MTTF), etc.) of elements and their relationships which allows automation of the use of analytic models to reason about quality attributes like performance. Table 2.1 categorizes different aspects of DBSE and MBSE.

V.Salehi's study [18] describes that an MBSE approach is often implemented on a large scale across the project and is time-consuming. His study stated that implementing MBSE on a relatively big scale with a traditional MBSE strategy typically requires highly skilled individuals and is time-consuming. Continuing from his study, many MBSE systems disregard the process flow from production and manufacturing phases to operations and maintenance phases. The study concluded that the more implementation and integration of MBSE makes it clear that the requirement that the whole project is based on should be tracked and validated throughout the entire system development process lifecycle [18]. Thus, it is clear that a new approach should be pursued to ameliorate these issues.

2.3 Failure Analysis

FMEA and FTA are two commonly implemented failure analysis methods which make use of top-down and bottom-up approaches respectively. These methods are used throughout the system lifecycle to aid engineers in determining potential failure risks to a system, and identifying potential methods of remediating the risks.

FMEA is a technique for analyzing probable system breakdowns [19]. To determine these failure modes and their causes and effects inside a system, as many components, assemblies,

and subsystems as possible are therefore studied. FMEA uses a qualitative and systematic approach to evaluate potential failure problems or modes in the design and manufacturing phases [19]. It starts with the formulation of all possible product and process failures during its system life cycle and then evaluation of the failure effects. The benefits of an FMEA approach are that there is traceability of each failure as their reporting and insights are being recorded and documented. It aids in the reduction and elimination of current and future product and process failures respectively through corrective action. The limitation of an FMEA approach is that the FMEA data has not been fully utilized because of poor information flow between product design and process control, "how and when to link the FMEA information with process control functions" and the FMEA process is halted by many organizations after the report is completed [6].

Table 2.1. DBSE versus MBSE. Adapted from [14]

	DBSE	MBSE
Information	Mostly Text Ad-Hoc Diagrams Loosely coupled Repeated in multiple documents	Visual and Textual Constructs Defined once and reused Shared across Domains Consistent notation in diagrams Defined relationships
Information Views	By document	Provides Viewpoints Filters By Domain, Problem, Space, etc.
Measuring Change Impact	Spans across Multiple Documents Text requirements which are isolated from Structure and Behavior	Relationships define traceability paths Natural part of the modeling process Programmatically Automated
Measuring Integrity in terms of Completeness, Quality and Accuracy	Manual inspection	Programmatically Automated Animation of Spec

FMEA employs a systematic and empirical analysis approach; the analyst has to identify the effects of each system component's potential failure modes on the system's performance

[20]. From the FMEA methodology [20], the procedure consists of the following steps: "1. Identify all failure modes; 2. Determine the effect of the failure for each failure mode, both locally and on the entire system being analyzed; 3. Classify the failure by its effects on system operation and mission; 4. Determine the failure probability; 5. Identify how the failure mode can be detected; and 6. Identify any compensating provisions or design changes to mitigate the failure effects".

FTA employs a top-down logical approach by translating the physical system into a logical diagram. Commonly used in risk assessment for the aerospace and nuclear power industries, it depicts a visual diagram of how failures and other factors such as human error lead towards a specific event using logical gates and small events showing the failure path sequence. The main benefit of an FTA approach is that the user identifies a specific interest in the top event and the root cause is identified with the development of a fault tree. The limitations are that if the system is too complex with a large variety of contributing components, it will be too time-consuming, will require more engineering efforts to complete the FTA, and may not even cover all failure possibilities. Generally, a partial failure concept is non-existent and can affect the accuracy of reliability calculations of the system [7].

Dean [21] identified failure modes that were inadequately addressed by breaking down actual mishap reports and employed a Probabilistic Risk Assessment (PRA) approach. This is supported by a human reliability assessment (HRA) for the identified mishap and the results implied that the PRA method provides a more accurate and quantifiable risk assessment than the MIL-STD-882E method for mishaps that happen during system operations.

Two significant maintenance management approaches used in industry are failure-driven and time-based maintenance (TBM) [22]. Yam et.al [23] discussed on failure-driven maintenance (FDM) which is a reactive management strategy where unanticipated occurrences generally influence corrective maintenance, which is performed only after an obvious functional system failure has happened. The study also stated that corrective maintenance can either repair or replace a failed component to restore an item of failing equipment. Unplanned equipment stoppages will cause minimal disturbance to production for non-critical components and FDM may be applicable [22]. But if equipment breaks down at random and it has a serious failure consequence, emergency maintenance is required [24]. Yam et.al's study [23] also stated that TBM reduces the degradation rates for failures by performing

periodical maintenance regularly with the assumption that for operational equipment and machinery, "the estimated failure behaviour of the equipment, i.e. the mean time between functional failures (MTBF) is statistically or experientially known". In summary, the main benefit of TBM is the prevention of functional failures through earlier regular corrective maintenance before its expected lifecycle ends. However, there are various limitations such as a possibility of random catastrophic failure occurrences, the compatibility and updates of TBM procedures to match the automated plant's actual operational requirements in modern times, maintenance planning; which has to be done by skilled planners based on the initial manufacturer's recommendations and failure statistics with the expertise of the technicians. In addition, it is vital for employees to work under pressure as the maintenance tasks requires pre-planning, and this becomes an intangible factor that may result in failure if not adequately prepared [25].

Yam et al. [23] also researched on Condition-Based Maintenance (CBM) which aims to reduce the redundancy of maintenance activities based on the system condition in TBM and FDM. Yam et al. proposed the implementation of an intelligent predictive decision support system (IPDSS) for CBM which used readily available prognostic indicators to establish potential failures that reduce the system performance. His study on CBM emphasized the importance in prediction of component deterioration for decision making in maintenance planning processes. The study has shown that the benefits of implementing IPDSS for CBM are 1) the overhaul intervals between major systems can be extended when needed, 2) maintenance planning is allowed more time with earlier failure predictors, 3) significant component damage can be reduced and 4) spare part inventory management can be more facilitated. The limitations of CBM which are its reliance on the behavior of component deterioration processes, the failure severity, the setup time requirement, the accuracy of the system condition measurements, and the degree of randomness due to the system deterioration at which failure occurs [23].

The common significant challenge in all of the above reviewed failure analysis techniques is that they require extensive documentation. Often times, the documentation is not stored, processed, or updated in a shared depository. Furthermore, it is rarely connected to a MBSE environment.

2.4 Digital Twin

A DT is defined as "an integrated multiphysics, multiscale, probabilistic simulation of an as-built vehicle or system that uses the best available physical models, sensor updates" [26]. Using simulation, machine learning, and reasoning, DT is a virtual model of a physical thing or system over its entire useful lifetime. in aiding stakeholders in decision-making processes [27].

Dave [28] highlighted the benefits of DT include simulating the system's years of operations in terms of maintenance to analyze the effect and results such as operating cost, in just hours or days. He also highlighted that the potential of DTs can be magnified by its connection to the real world with the application of sensors and smart devices forming the internet of things (IoT), enabling constant tracking of the events in the system or process. Furthermore, the data collected from the IoT devices can be used in the previously mentioned benefit which is DT simulation for self-critique to compare and analyze the differences between the predicted engineering data and the actual collected data. In this way, the DT continues to develop and improve, becoming more comprehensive and precise over time [28].

Often times, each physical asset or system has its own DT instantiation. Each DT instantiation captures the specific system's history and real-time data with the aid of sensors, maintenance data, and etc. The uses and classifications of a DT shown in Figure 2.5 vary according to the requirements derived from the stakeholders' needs. The intent behind the model used to create a DT is determined by the specific problems that the stakeholders want to address. DTs can include both data-driven models and physics-based models among others. In this thesis, a DT model structure is developed that can be suited for both data-driven and physics-based modeling, by representing the parameters (value properties in modeling context) with pseudo values that can be replaced with any vital parameters to generate insightful results. The model will illustrate the uses of value properties such as operation time, operation state, failure modes to calculate the availability for the overall system, subsystems, and components respectively.

2.4.1 Digital Twin Application in Maintenance Operations

With the utilization of DTs, failures can be predicted and action taken to prevent unexpected downtime, better manage spare component inventories, monitor and manage a fleet of

equipment, perform what-if scenarios, and optimize operations [29]. DT helps to shift from a time-based maintenance (TBM) philosophy to a condition-based maintenance (CBM) philosophy. Reducing unplanned downtime by shifting to CBM and improving inventory management are both significant factors in the push for broad DT adoption. Especially on large, complex systems where it may be difficult to implement an effective TBM and where unplanned downtime is significant, shifting to a DT-based CBM philosophy can improve maintenance operations.

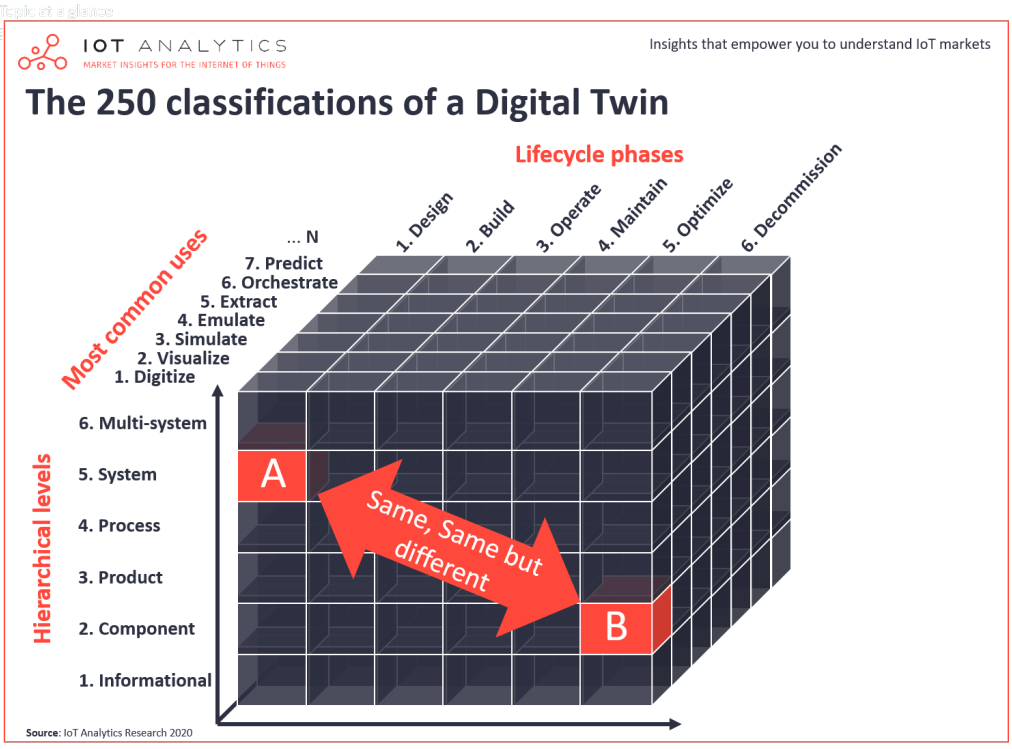

Figure 2.5. Classifications of a Digital Twin. Adapted from [30]

MATLAB [31] has developed DT models in maintenance operations using the predictive preventative maintenance algorithm as shown in Figure 2.6. The key benefit in using this application is that companies can upkeep their complex systems with expensive components through optimized maintenance planning in terms of both sparing and staffing; a shift from TBM to CBM.

Referring to the workflow in Figure 2.6, the main challenges in adopting DT models mainly revolve around data; the data collection and processing, and development of model with data.

In MATLAB's white paper [30], there are four common barriers in predictive maintenance and they are 1) insufficient sensory data to build model, 2) lack of failure data to compute accurate results, 3) inability to incorporate root cause analysis to predict failures and 4) no prior knowledge to perform predictive maintenance. The main challenges and common barriers form part of the motivation behind this thesis research.

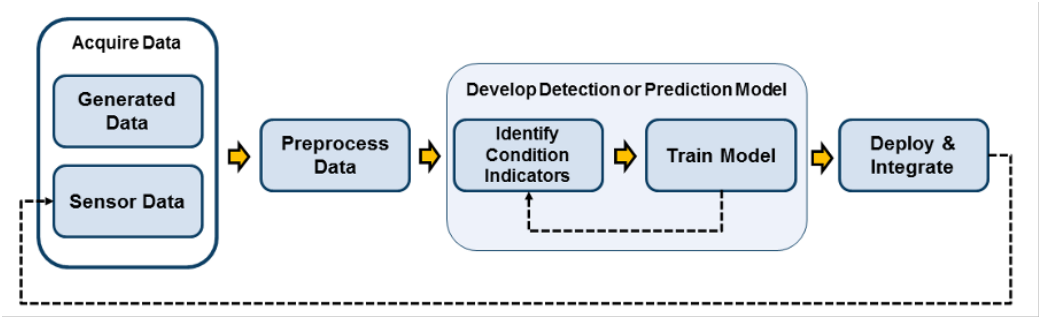

Figure 2.6. Predictive Maintenance Workflow. Adapted from [31]

With MATLAB and Simulink, DTs can be created in many ways to build a predictive maintenance algorithm [32]. Furthermore, the intended uses of each DT vary from failure classification to remaining useful life estimation. There exist some common challenges in the published methodologies which mostly revolve around data in building the model. They are the lack of sensory data to execute machine learning algorithms and the lack of actual failure data [32]. All in all, the biggest challenge in the published research on DT technology is about the accuracy of model in terms of results validated with actual output. The contributing factors will be the time. The time to develop a mature DT model, the time to train personnel in failure analysis, and the lack of experience in building predictive maintenance algorithms.

The proposed method in this thesis demonstrates the logical model construction for a system of systems (SOS) by following the system physical architecture, relating all potential failure modes to components and subsystems through a completed DFMEA, and assigning pseudo probability values and preventative maintenance time. The objectives of the method is to demonstrate the efficacy ability in failure traceability in a system of system and the potential of improving operational availability in a system without the use of data.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Methodology

The proposed methodology for the development process of the system DT and the maintenance support analysis is based on the five steps as shown in Figure 3.1.

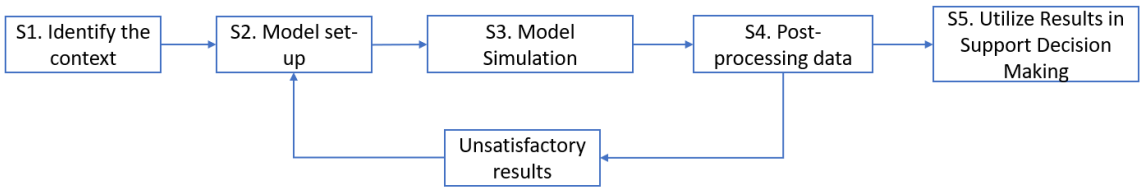

Figure 3.1. Flow Chart of the Proposed Methodology

3.1 Step 1. Identify the Context

As shown in the literature review, it is important to understand the context behind the main challenges in both current SE and failure analysis processes. Some of those challenges are the traceability of failures within the system and understanding the interactions of both corrective and preventative maintenance. First, the system of interest (SOI) must be defined properly in terms of the relations and interactions between the system, subsystems, and components. An accurate representation of the system structure will minimize the error in traceability when a failure occurs, or maintenance is done. Second, the specific parameters of interest to be collected from the DT must be identified to ensure that those parameters are included in the model so they can be captured at all levels of the SOI. Examples of these parameters include operational availability (Ao), number of failures, operation time, and number of preventative maintenance events.

3.2 Step 2. Model Set-up

Systems Modeling Language (SysML) developed by Object Management Group (OMG) is a common graphical modeling language for system engineering in a complex project [33]. By utilizing MSOSA which is a MBSE modeling tool in SysML language, a DT model of the system of interest is developed using an architecture development methodology. This methodology could be the MagicGrid grid methodology, shown in Figure 3.2 [34]. Since

this study focuses on incorporation of the failure analysis, the building of the DT model will not be discussed further. For this study, a simple model of an engine was developed that included three subsystems. This model is described in detail in the next chapter.

			Pillar			
			Requirements	**Behavior**	**Structure**	**Parametrics**
Layer of Abstraction	**Problem**	**Black Box**	Stakeholder Needs: • Requirements diagram • Requirements table	Use Cases: • Use Case diagram • Activity diagram	System Content: • Internal block diagram	Measures of Effectiveness: • Block definition diagram
		White Box	System Requirements: • Requirements diagram • Requirements table	Functional Analysis: • Activity diagram	Logical Subsystems Communication: • Block definition diagram • Internal block diagram	MoEs of Subsystems: • Block definition diagram
	Solution		Component Requirements: • Requirements diagram • Requirements table	Component Behavior: • State machine diagram • Activity diagram • Sequence diagram	Component Structure: • Block definition diagram • Internal block diagram	Component Parameters • Parametric diagram

Figure 3.2. MBSE Grid Mapping to SysML. Adapted from [34]

Once the DT model structure has been built, the failure modes and preventative maintenance events of the respective components are incorporated into the DT model. The failure modes are defined using classical reliability techniques such as FMEA or FMECA. The failure probabilities for each component failure mode must be determined from component reliability data or through testing. Again, this is beyond the scope of the study. For the purposes of this study, the failure modes and respective failure probabilities are assumed to be known. This paper explores incorporating the information from the component failure modes and failure probabilities to understand higher level effects on the system. This study uses state machines to incorporate the component level failure behavior into the DT model. A component level state machine usually has two states, On and Off. To account for the component's failure and preventative maintenance behavior, additional states are added to the component's state machine diagram. These states are: Faulted, Repair, and Preventative Maintenance. Transitions between states are governed by the condition of the component (e.g., SumpFailed) or control signals from higher level subsystems or the system of interest (e.g. SumpOn or SumpOff).

Another key feature of the model is the use of the operational state binary variable that indicates whether the component is functioning or not. In Figure 3.3, the Sump is operational

when the SumpOpState equals one, as shown in the On state, and not operational when the SumpOpState equals zero as is the case in all of the other states shown in the figure.

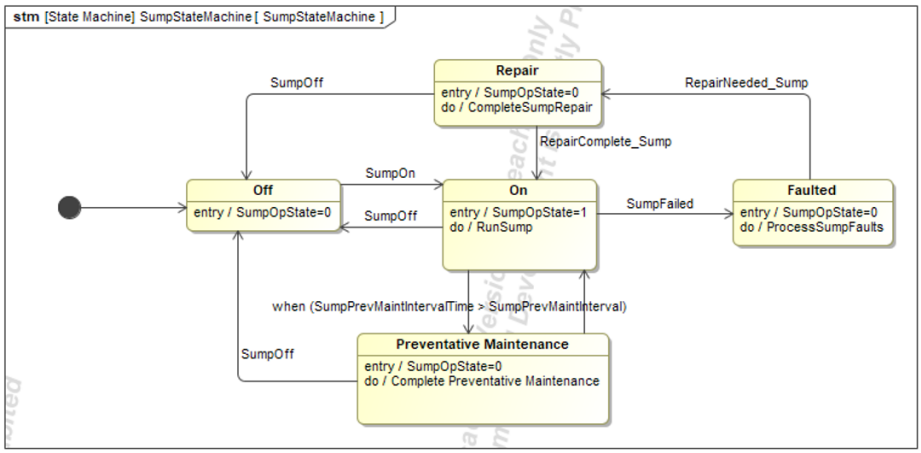

Figure 3.3. State Machine Diagram of Sump Component

System or subsystem operational states can then be determined by the structure function of the operational state variables of their components [35]. For example, if a system has three components the system's operational state vector is Comp1OpState, Comp2OpState, Comp3OpState3. If all the components must work for the system to work, then the system is a series system and the system's operational state vector is 1,1,1. One way to calculate a series system's structure function is simply to multiply the component operational state binary variables. If any of the components are failed or off-line due to preventative maintenance, then the product will also be zero indicating the system has failed, OpState=0. The model uses parametric diagrams to determine the operational states of the systems or subsystems based on the series structure function. While this study only implements series type structure functions, the technique can be extended to other types of structure functions including subsystems with parallel structure, k of n structure, or non-series parallel structures. The output from structure function parametric diagram was then used in a time counter actions such as "timecounter = timecounter + timestep*OpState" using SysML opaque actions, where the timecounter for the system or subsystem would increment only when the subsystem was operational. These counters track the amount of operational time of each component, subsystem, and the system of interest.

In the ON state's do behavior, the model uses two counters as shown in Figure 3.4. The first counter counts time while the component is operational. The second counter counts time

toward the preventative maintenance interval. This is followed by parallel actions where one branch calls for performance of the normal behavior of the component and the other branch checks for simulated faults, CheckforFaults.

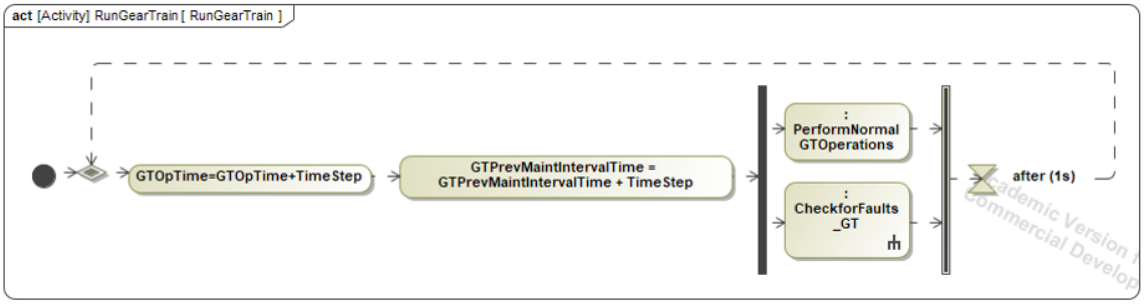

Figure 3.4. Using Individual Time Counters for Tracking

The CheckforFaults action calls an activity has a branch for each failure mode as shown in Figure 3.5. The activity starts by ensuring the value properties (FM_failuremode) that count whether a failure has occurred with binary values (0 or 1) are set to 0. Then for each failure mode a random number is generated which is then compared to the probability of failure for that failure mode. If the random number is less than the probability of failure, then a simulated failure has occurred and the value property FM_failuremode is set to one. Since these values are set independently in each branch, the simulated component can have multiple failures at one time. These failures are summed and used to feed the decision node where if the summation is greater than zero, the component sends a signal to the next higher level (subsystem or system) to indicate that the component has failed. In a real system, this would happen automatically due to the loss of power or force but in the model an artificial signal is used to alert higher level subsystems or system of the failure.

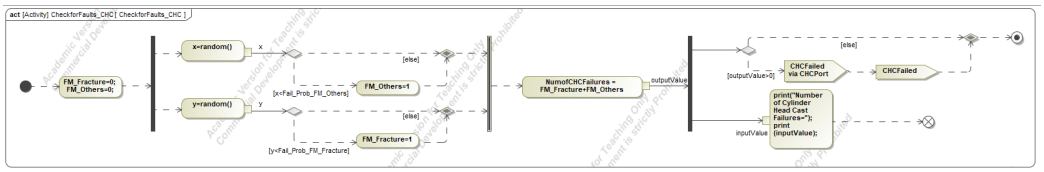

Figure 3.5. Activity Diagram for Checking Every Failure Mode

As shown in Figure 3.6, for subsystems with components, the state machine for the subsystem is fairly basic with just an Off and On state. The transition between these states is driven by

control signals from the next higher level subsystem or the top system block, depending on where the subsystem resides in the system hierarchy.

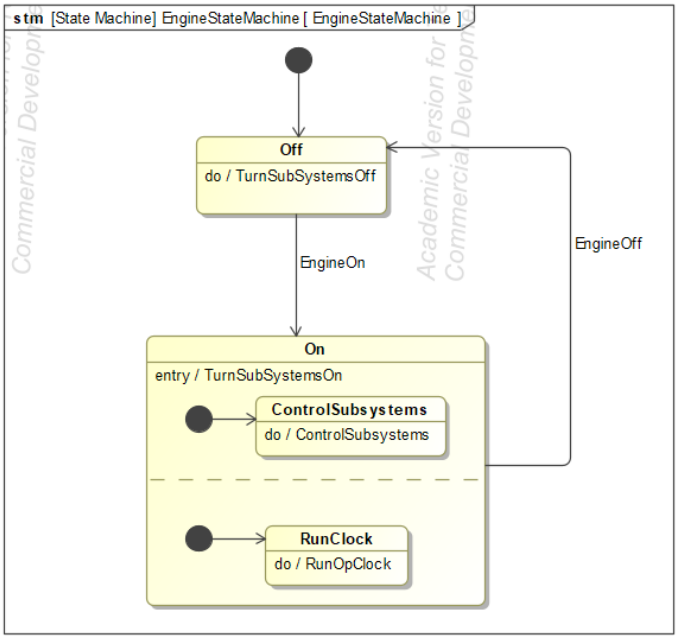

Figure 3.6. State Machine Diagram for Subsystem

When the subsystem is in the Off state, the subsystem sends out control signals to turn off all its components. When the subsystem receives the On signal, the subsystem transitions to the On state where it turns on all of its components and activates the ControlSubsystems substate where it listens for signals from the components. If it receives a signal indicating that a component is inoperative, then the subsystem will shut off all the other components in the subsystem while allow the inoperative component to continue its path through the Faulted and Repair states. Once those states complete their behavior, the component sends a signal to indicate the component has returned to an operative condition whereupon the subsystem turns all its other components back on. The model uses an orthogonal state for the On state so the clock counter can run independently of the control activity. If one desires to model subsystems without components, then the structure of the state machine for the subsystem is modeled just like a component.

One can control the desired scenario parameters by modifying the Domain block. Normally, the Domain block is only used to contain all the system and any external entities that the

system must interact with and provides a contextual view of the system. For this study, there were no external entities. Additional value properties were added to the Domain block. The value properties were CalendarTime, a counter to capture elapsed time; TimeStep, the amount by which the counter is incremented; and EndTime, a point at which the simulation will terminate. One must also create a state machine for the Domain Scenario, which will control the execution of the simulated scenario and then at the end of the scenario execute the post processing calculations to facilitate data capture.

While state machines are used to govern the overall behavior of the system, subsystem, and components, other SysML diagrams are also needed to describe the structure and behavior of the overall system and components, specifically activity diagrams, block definition diagrams, internal block diagrams, and parametric diagrams, which are used to perform various calculations. The details of these other diagrams will be discussed in the next chapter. Lastly, a simulation configuration was created in MSOSA to set simulation parameters. The execution target was set to the Domain block. The result location was set to record the data output in the form of instances to a package and the number of runs set to a desired integer value. Timing properties were set with the Start Time set to 0, Step Delay to 1, Step size to 1, Time Unit to second, and Time Variable Name set to simtime. All other values in the simulation configuration were left at their respective default values.

3.3 Step 3. Model Simulation

After the model has been developed, the simulation is run using the simulation configuration for a desired number of runs to capture data in instances for each of the parameters of interest. In a real system analysis, one would ensure that there are enough data points to accomplish statistical analysis to include distribution fitting and creation of histograms for the various parameters of interest during post-processing. Since this study is merely developing and demonstrating the methodology, a limited number of runs was done.

3.4 Step 4. Post-processing Data

The collected data captured from the model simulation is then placed in an instance table. From the instance table, the data can be exported to a CSV file which can then be processed using MSExcel or other statistical analysis software like Minitab. If one uses MSExcel,

distribution fitting could be performed on each specific parameters of interest to characterize the probability distributions by using plug-in software like Risk Simulator, Crystal Ball or Minitab. For this study, the author created empirical techniques on the raw data to capture the range of values and the cumulative probability distribution for the parameters of interest. The probability distributions can be displayed using line charts, bar charts, or box and whisker charts. In the event of unsatisfactory results (post-processed data does not make sense and has no value), the model set-up is reviewed on iteration for that parameter of interest in question and the model is revised (back to step 2. Model set-up).

3.5 Step 5. Utilize Results in Support Decision Making

One can use the probability distributions determined in the Post-processing step in various forecast models, possibly allowing a more accurate estimate on the parameters of interest for better decision-making. From the post-processed data, there may be more clarity on the trades that the decision makers can take in terms of design improvement and maintenance support. The forecasts will provide better insight into system availability, which subsystems or components are likely to fail, which failure modes contribute the most and which components may be candidates for design improvement, sparing support needed, and maintenance staffing based on the failure forecasts and number of preventative maintenance events. In addition to just raw data, visual illustration may also be done for every parameter of interest at the system, subsystem, and component levels to better inform decision makers on the interactions among the components and subsystems on the overall performance and availability of the system of interest.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Development of Case Study Model

An engine case study is used to illustrate the proposed methodology. This section provides a detailed explanation of the SOI and the equations used for the model.

Based on the system level breakdown [36], the system, subsystems, and components for the SOI (engine) are defined and the physical architecture is built from the system to the subsystems and finally to the components, shown in Figure 4.1. Under the engine (system), the subsystems comprise the Cylinder Block, Exhaust System, Valve Train, Engine Cooling System, Injection System, Cylinder Head, Cranktrain and Geartrain. Under the subsystems, there are the components that are parts of the subsystems, for ease of illustration, Figure 4.1 shows only the components for the CylinderHead in direct notation and the components of the other subsystems are hidden.

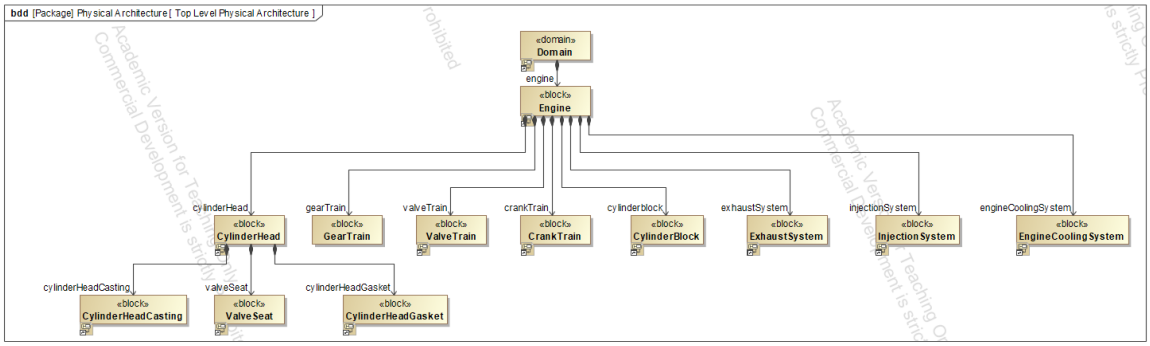

Figure 4.1. Physical Architecture of the SOI (Engine)

In this context of the proposed methodology, the components are defined as things from which the subsystems are composed, and subsystems are "components" from which the system is composed. In the SysML (Systems Modeling Language) language, interfaces are defined as the connectors between the components, subsystems, and system. Furthermore, constraints are used as requirements to define the system behaviors for each of them.

Using a block definition diagram (BDD) to build the top-level physical architecture shown in Figure 4.1, the composition of all system blocks used in the system as well as the lower-level blocks, such as subsystems and component is specified. The subsections that follow

provide a step-by-step guidance to model development using the engine as case study. To demonstrate the fidelity of the proposed methodology and understanding the model development process, only three subsystems will be built as shown in Figure 4.2.

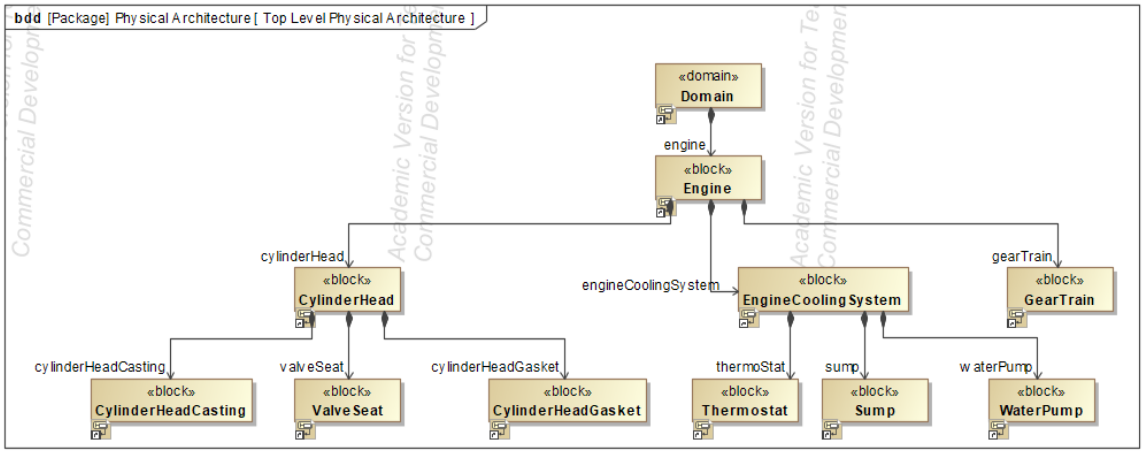

Figure 4.2. Uses of 3 Subsystems for SOI

To understand the building process of the model in gradually increasing complexity, the model description will be provided in the following order, 1) Domain Level, 2) System Level, 3) Subsystem Level without components – only 1 failure mode, 4) Subsystem Level with components and 5) Component Level.

4.1 Domain Level Description

The top-level physical architecture starts with a domain block which controls the execution of the stimulated scenario and post processing of data such as calculations of system overall operational availability through a state machine, as shown in Figure 4.3.

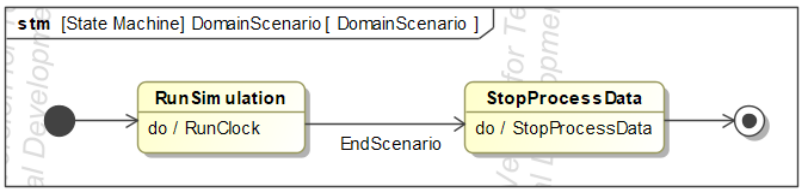

Figure 4.3. State Machine Diagram for Domain Scenario

Value properties are assigned to the domain block as shown in Figure 4.4 and the purpose of each value property is summarized in Table 4.1. These value properties create a discrete-event like simulation inside the system model. CalendarTime begins the scenario at 0 time

and then it is incremented using the TimeStep until the EndTime is reached. To facilitate time compression, each second of simulated time is assumed to represent one hour of real time. So the 1440 steps would represent 60 days. This time compression assumption is used throughout the model.

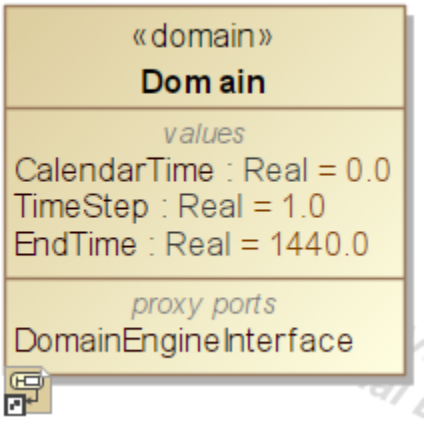

Figure 4.4. Block Definition Diagram for Domain Block

Table 4.1. Uses of Value Properties for Domain Block

Value Property	Type	Default Value	Purpose
CalendarTime	Real	0 for new systems	Accumulates time for calculation of availability and other suitability values.
TimeStep	Real	Desired time step value	Sets value for the advance of time during simulation.
EndTime	Real	Desired end time of simulation	Represents the end of the period of interest

In the DomainScenario state machine shown in Figure 4.3 and under the RunSimulation activity diagram in Figure 4.5, signals (StartScenario and EndScenario) are sent through the Domain's interface to trigger the start and end states of the simulation. In addition, time step is added to the calendar time counter as the simulation runs until it meets the desired period.

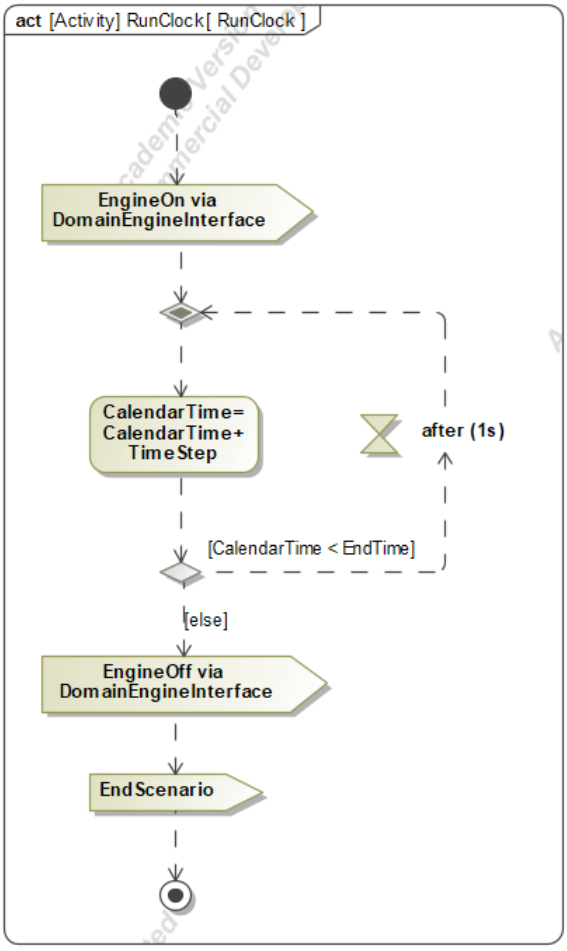

Figure 4.5. Activity Diagram for RunClock under Run Simulation State

For simplicity in designing and understanding the model, the domain communicates only with the top-level system (SOI) which is the engine as shown in Figure 4.6. The engine then communicates with the next lower level of subsystems and from the subsystems down to the components level.

4.2 System Level Description

To implement the methodology at the system level, value properties are assigned to the Engine System block as shown in Figure 4.7. The uses of each value property is summarized in Table 4.2. EngineOpTime starts the engine time counter at 0 time and then it is incremented using the TimeStep. The EngineOpState starts at 1 in the form of integer and is computed

based on the described scenario. The Engine_Ao refers to the operational availability of the engine system.

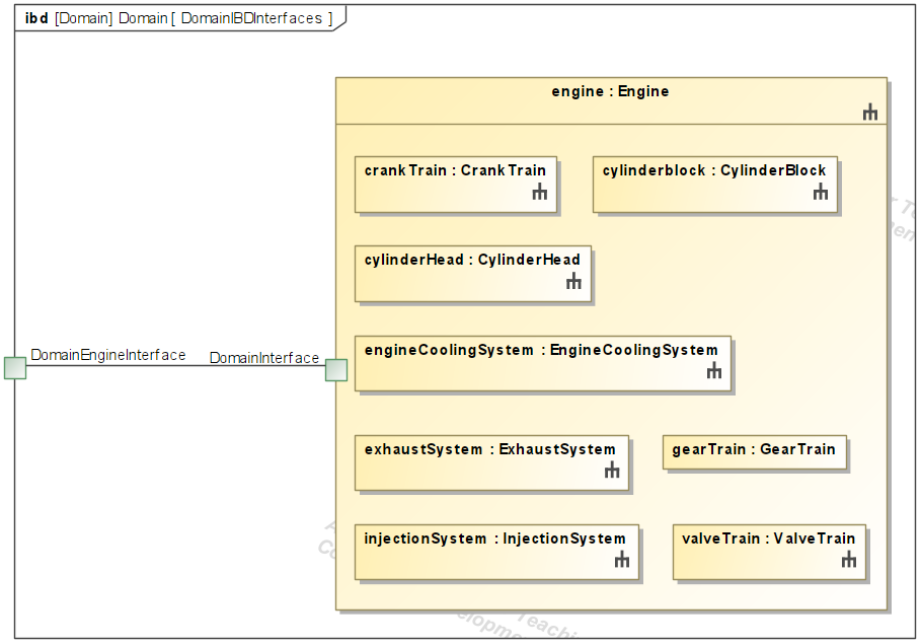

Figure 4.6. Internal Block Diagram for the Domain Interfaces

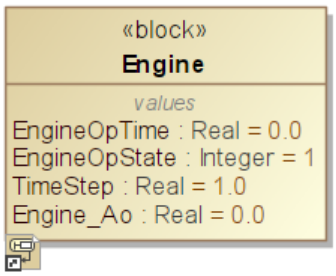

Figure 4.7. Block Definition Diagram for Engine System Block

At the system level shown in Figure 4.8, the engine operates in two simple states, “Off” and “On.” Both states have the similar behavior described using activity diagrams which turn the subsystems off and on via the interface ports, shown in Figures 4.9 and 4.10. In the “On” state, there are two internal states running concurrently whose functions are to control the subsystems and to increment the engine operation time when it is operational.

Table 4.2. Uses of Value Properties for System Block

Value Property	Type	Default Value	Purpose
“System”OpTime	Real	0	Accumulates time of system in the operating state
“System”OpState	Integer	1	Represents whether system is operative or inoperative
TimeStep	Real	0	Used to advance OpTime a set interval (could be assigned through inherited property using generalization relationship)
“System”_Ao	Real	0	Used to capture value of Ao for system

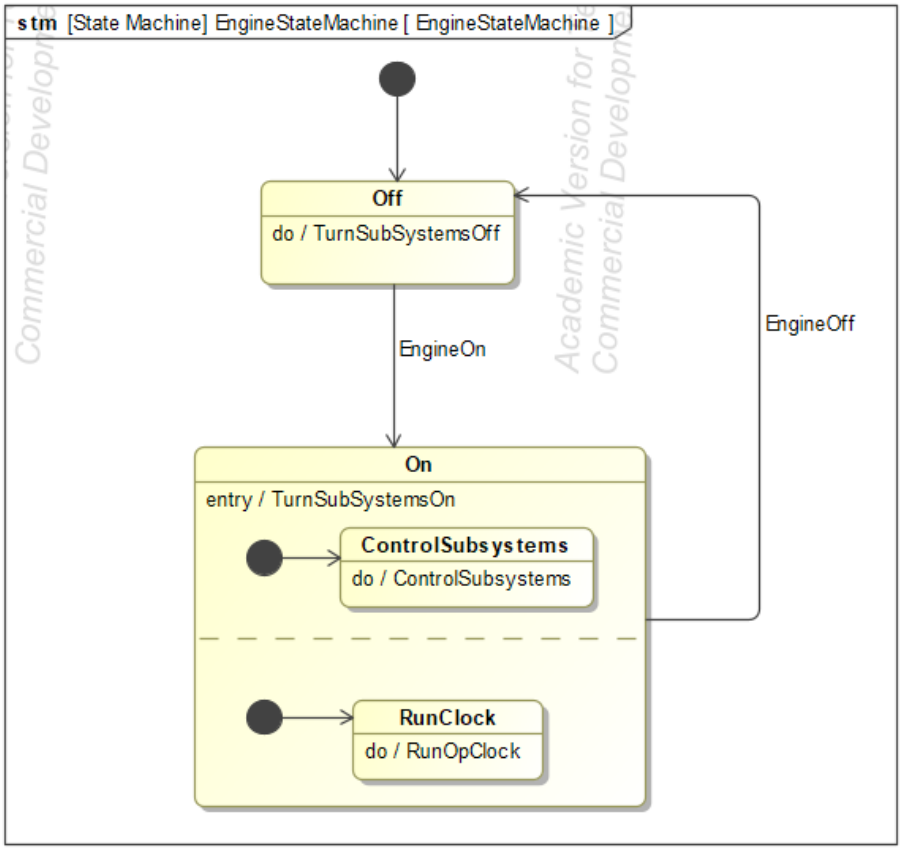

Figure 4.8. State Machine Diagram for Engine System

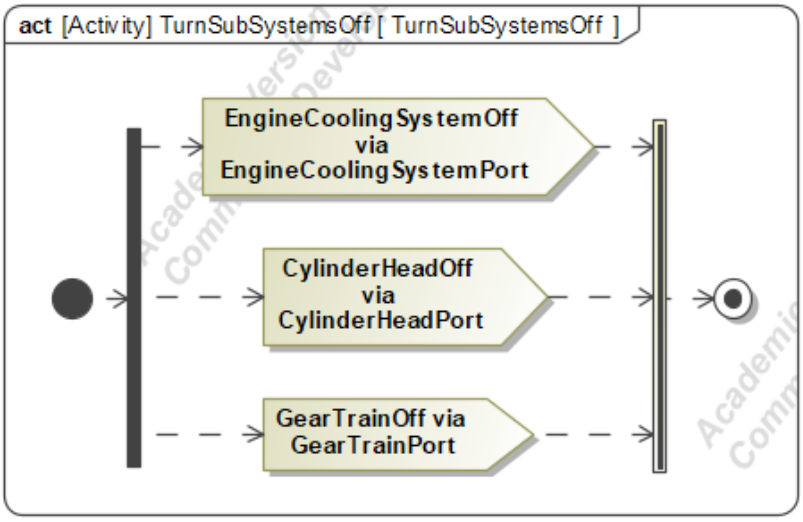

Figure 4.9. Activity Diagram for Turning Subsystems Off

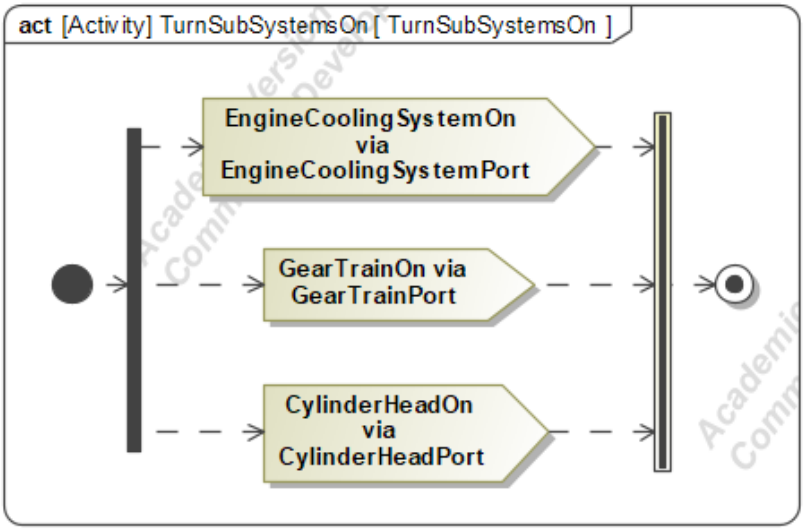

Figure 4.10. Activity Diagram for Turning Subsystems On

In the activity diagram for controlling the subsystems shown in Figure 4.11, the operational states of each subsystem is controlled by signals that will toggle the operational states of the other subsystems. For example, when the system receives a signal that geartrain is inoperative, signals will be sent to turn off the other subsystems, and when the geartrain signals that it is again operative the system will turn on the other subsystems. This sequence also occurs when a component goes into a preventative maintenance event. Several configurations of

this activity were tried with the one shown in Figure 4.11 being the most successful, but not entirely successful. There is still an issue with simultaneous failures, which causes a race condition to be created among different parts of the activity diagram. This causes systems to be shut off or turned on at the wrong times. If the failure probabilities are low, then the likelihood of simultaneous failures are very low so it occurs rarely. More likely is that preventative maintenance intervals may align causing several preventative maintenance events may occur simultaneously. In the model, the author purposely ensured that this would not occur by setting the preventative maintenance interval times at odd intervals.

In the activity diagram for running the engine system operation time counter shown in Figure 4.12, an opaque behavior that increments the engine operation time and based on the amount of time step and the operation state of the engine system. As the operational state of the engine is dependent on the operational states of the subsystems which rely on the components' operational states, when any one of the subsystems or components is off or on, the engine operational state will follow suit. There is also a time delay of one second added to the loop to limit the rate of the clock running to once per second. This limitation also helped the model from overloading the computer processor because normal simulation rates occur at million times per second or in milliseconds.

For the engine system to communicate with the subsystems, an IBD shown in Figure 4.13 is used to create the interfaces and ports to send item flows such as signals to trigger the states of each connected block.

4.3 Subsystem Level without Components only One Failure Mode Description

The model modifications can be developed at different levels of fidelity. If one wants to implement the failure model at the subsystem level and not include the components, value properties can be assigned for the failure modes, probability of failure, number of failures, etc... in the subsystem block. Value properties are assigned to the GearTrain Subsystem block as shown in Figure 4.14 and there are more value properties required to facilitate additional states for realism of the scenario simulation.

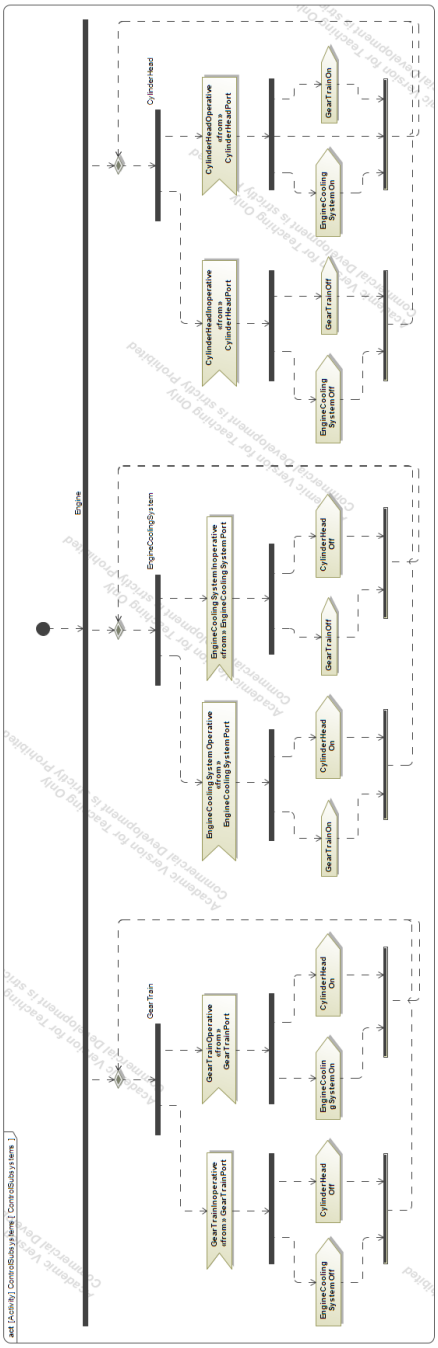

Figure 4.11. Activity Diagram for Controlling Subsystems

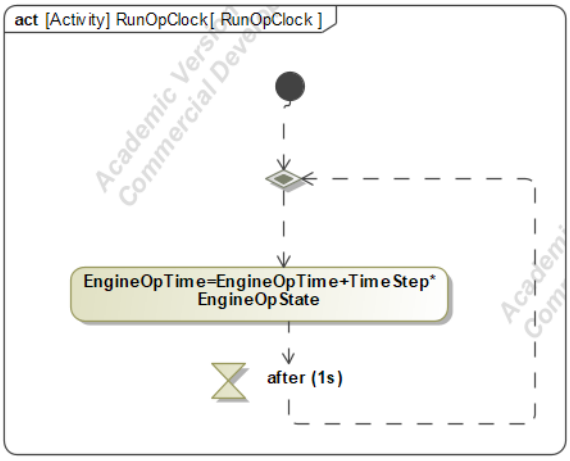

Figure 4.12. Activity Diagram for Running Engine System Operation Time Counter

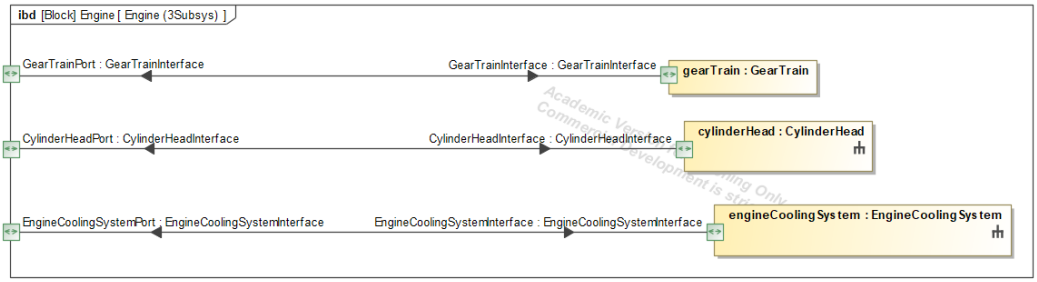

Figure 4.13. Internal Block Diagram for the Engine System Interfaces with Subsystems

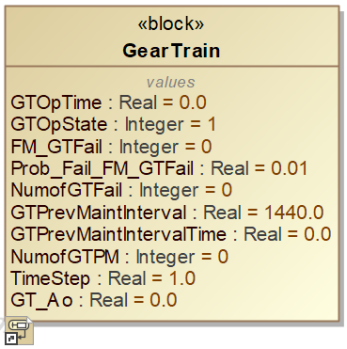

Figure 4.14. Block Definition Diagram for GearTrain Subsystem Block

The uses of each value property is summarized in Table 4.3. For the operation time, time step, operational state and operational availability of the subsystem, same logic applies as at the system level description. There are additional value properties required as tracking parameters for each state in the failure model. The subsystem starts with no occurrence of failure; failure mode FM_"Failure Mode" begins at 0 in the form of an integer. When a small probability of failure mode (Prob_Fail_FM_"FailureMode" in the form of real value) triggers a failure occurrence, the subsystem operational state will change from 1 (On) to 0 (Off) and the number of simulated failures (Numof"FailureMode") is summed during the time step. There is a preventative maintenance time counter ("Subsystem"PrevMaintIntervalTime) that increments as the subsystem is operating until it reaches the stipulated preventative maintenance interval ("Subsystem"PrevMaintInterval). Each preventative maintenance performed then clocks to the number of preventative maintenance done (Numof"Subsystem"PM).

Table 4.3. Uses of Value Properties for Subsystem Block

Value Property	Type	Default Value	Purpose
"Subsystem"OpTime	Real	0	Accumulates time of subsystem in the operating state
"Subsystem"OpState	Integer	1	Represents whether subsystem is operative or inoperative
TimeStep	Real	0	Used to advance Optime a set interval (could be assigned through inherited property using generalization relationship)
"Subsystem"_Ao	Real	0	Used to capture value of Ao for subsystem
FM_"FailureMode"	Integer	0	Binary number to indicate whether failure mode has occurred. (1=FM occurred; 0 FM has not occurred)

Continued on next page

Table 4.3 – continued from previous page

Value Property	Type	Default Value	Purpose
Prob_Fail_FM_“FailureMode”	Real	varies	Provides failure probability for comparison to random number generator during simulation.
Numof“FailureMode”	Integer	0	Sum of simulated failures that have occurred during a time step.
“Subsystem”PrevMaintInterval	Real	varies	Sets time limit for preventative maintenance interval
“Subsystem”PrevMaintIntervalTime	Integer	0	Accumulates operation time to trigger transition to PM state
“Numof”Subsystem”PM	Integer	0	Accumulates the number of PM events

For simple illustration on the transition of states at the subsystem level, the GearTrain subsystem has no component and there is only one failure mode on the subsystem. Figure 4.15 shows the uses of state machine diagrams and activity diagrams associated to the subsystem state machines. State machine diagram is created to represent the 5 different states: 1) Off, 2) On, 3) Faulted, 4) Repair and 5) Preventative Maintenance. The initial state is Off state and the only transition state from Off state is On state. The other transition states revolve around the On state and are explained in details in following sections.

Within each state, an activity diagram is used to define the detailed behaviors of the blocks and propagation from one state to another or one block to another, in terms of 1) entry, 2) do and 3) exit behaviors.

For the entry behaviors, opaque behaviors are used to trigger the operation state from 0 (off or inoperative) to 1 (on or operative). Activity diagrams are created to define the do behavior inside each state to perform functions such as running of components, processing fault, completing repair and preventative maintenance, shown in Figures 4.16, 4.18, 19 and 20.

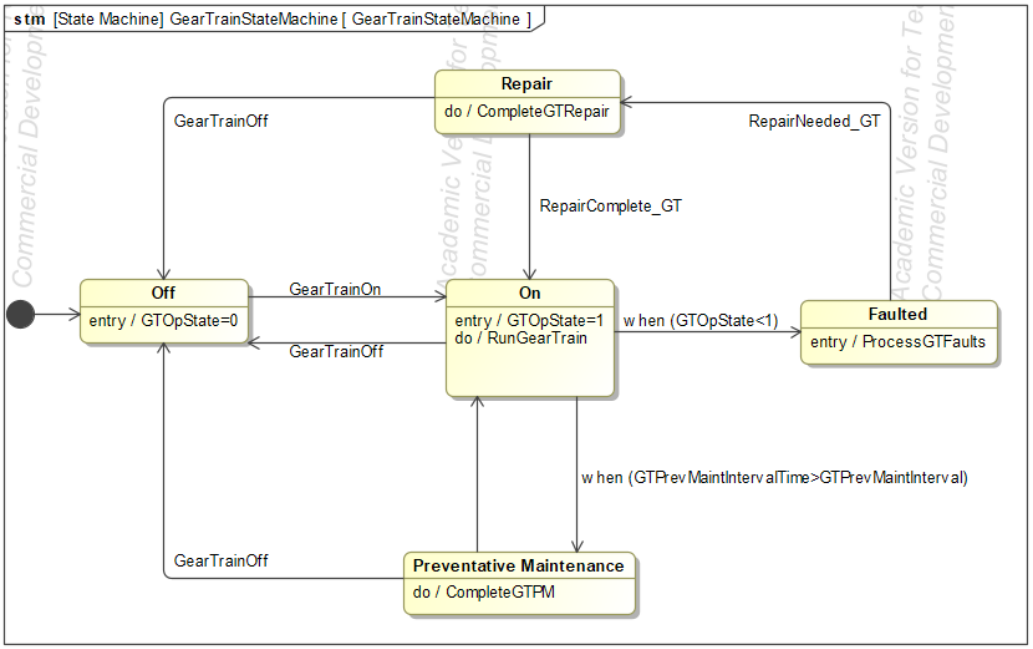

Figure 4.15. State Machine Diagram for GearTrain (System without Components)

When the GearTrain subsystem is turned on via the signal GearTrainOn, the state changes to On state triggering both the entry and do behaviors. The do behavior is represented by an activity diagram shown in Figure 4.16. Opaque actions in this do behavior function to add time step to both the subsystem operation time counter and preventative maintenance time interval counter.

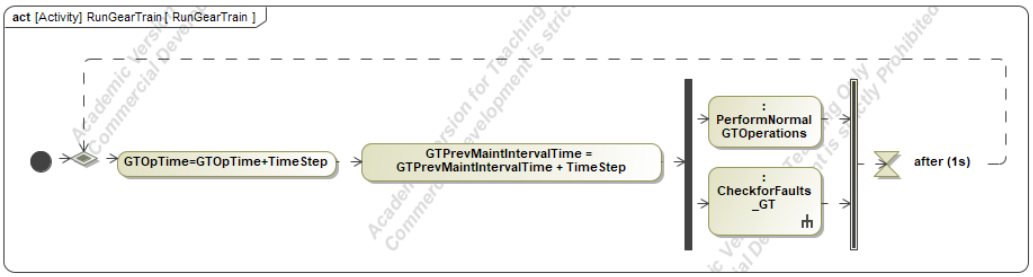

Figure 4.16. Activity Diagram for Running GearTrain Subsystem

In addition, sub-activity such as checking for fault is used to represent part of the activity under the Do behavior, shown in Figure 4.17. Under the do behavior activity diagram, Figure

4.17 depicts the sub-activity diagram called “CheckforFaults_GT” which runs checks on any failure occurrence. This simulated failure occurrence is triggered by utilizing a random number generator based on comparison of the random number with a failure probability for the respective failure mode. When the specified failure mode(s) is or are activated (failure mode = 1), a signal “GearTrainInoperative” is sent through its interface to communicate with the engine system to trigger the next state of Faulted and an opaque behavior changes the geartrain operation state to 0.

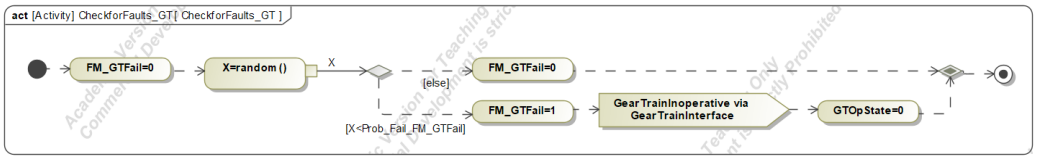

Figure 4.17. Activity Diagram for Sub-activity; Check for Faults

When a failure occurrence takes place the GTOpState is set equal to zero and the On state will transition to the Faulted state through the guard condition (when GTOpState is less than 1). Under the Faulted state, an activity diagram called “ProcessGTFault” is used for the do behavior whose opaque action functions to add counter to the number of geartrain failure mode and send the signal “RepairNeeded_GT” to trigger the transition to the Repair state, as shown in Figure 4.18.

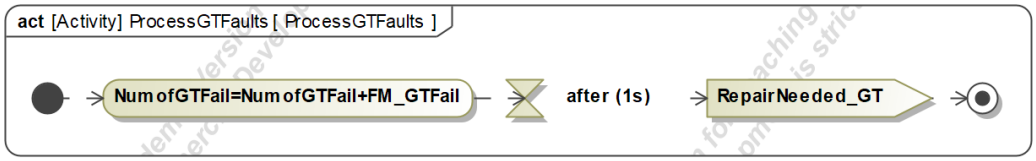

Figure 4.18. Activity Diagram for Processing Faults

Figure 4.19 depicts the Repair state where a time event is used to represent the time needed to accomplish failure resolution. After the failure resolution is completed, the failure occurrence will be reset to zero and a signal “RepairComplete_GT” is sent and transition to the On state.

When the preventative maintenance time interval counter in the On state exceeds the preset preventative maintenance time interval, a guard condition will trigger the transition from On state to the Preventative state. Figure 4.20 depicts the activity diagram under the do behavior for the Preventative Maintenance state. Under the preventative maintenance, the

do behavior activity diagram execute 5 functions in sequence, 1) sends a signal "GearTrain-Inoperative" through the interface to communicate with the engine system in shutting off other subsystems, 2) executes a time delay to represent the completion of the preventative maintenance, 3) resets the preventative maintenance time interval counter, 4) increments the counter of the number of preventative maintenance completed and 5) sends a signal "GearTrainOperative" through the same interface to the system, which will turn on other subsystems.

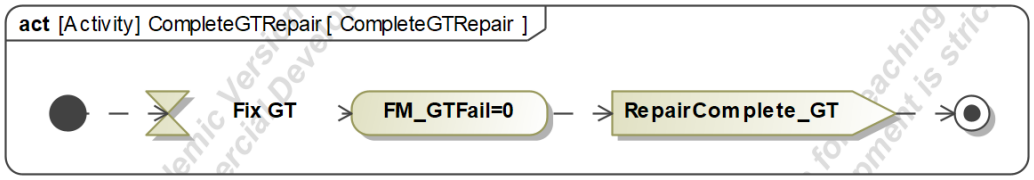

Figure 4.19. Activity Diagram for Completing Repair

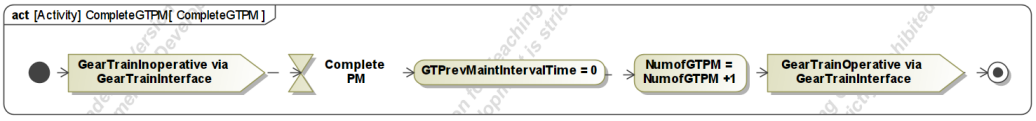

Figure 4.20. Activity Diagram for Completing Preventative Maintenance

4.4 Subsystem Level with Components Description

If one wants to model the subsystem with more fidelity, where all of the subsystem functions occur in the subsystem's components, then some of the value properties are moved from the subsystem level to the component level. The model building process for subsystem level with components is the same as step 2 for system level but with the additional same types of value properties used in the previous step: subsystem level without components.

Referring to a published DFMEA research [36] for the SOI (engine), the various failure modes are tagged to the respective components using value properties which facilitate the simulation and analysis. For each of the components, value properties for the failure modes, probability for each failure mode, operation time, component operation state, component preventative maintenance interval, component preventative maintenance interval time, counter for number of failures per failure mode and total number of failures are as-

signed. Figure 4.21 shows the value properties assigned to the component level blocks under the subsystem block for CylinderHead.

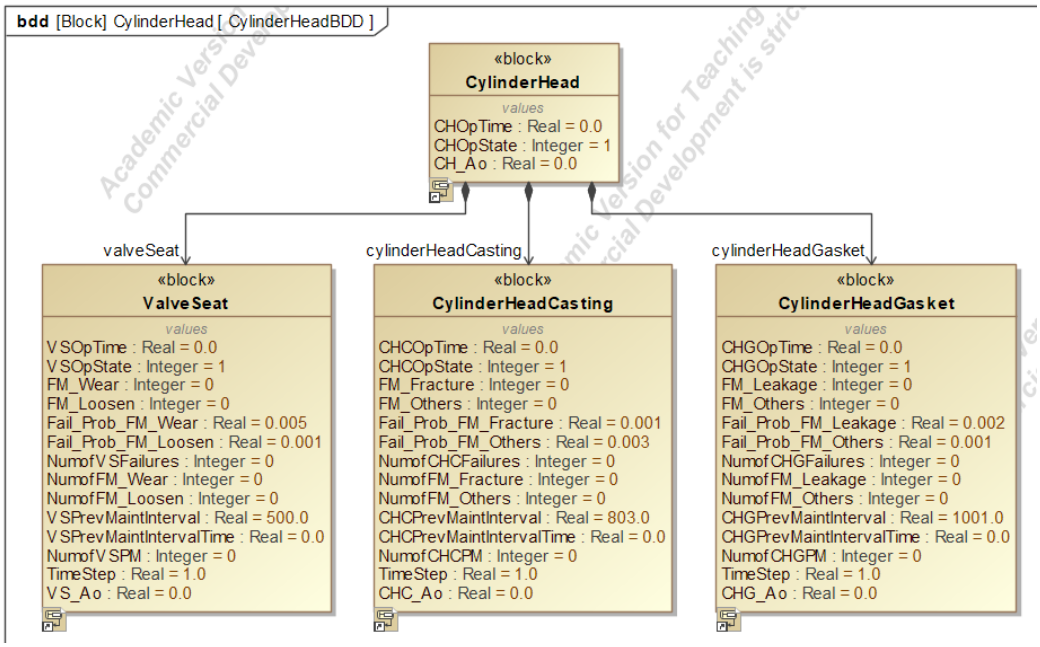

Figure 4.21. Assigning Value Properties to Component Blocks

Following the same way in constructing the state machine for the system level, Figure 4.22 depicts the states for the CylinderHead subsystem. In addition, the behaviors are the same in both turning on and off components. There is a difference when adding time step to the operation time counter, however, because in the Run Op Clock activity diagram the time step value is multiplied by the subsystem operational state value, which is a binary (0 or 1) value determined by a parametric diagram based on the structure function. This causes the operation time (OpTime) to increment only when the subsystem is operational. These activity diagrams are shown in Figures 4.23, 4.24 and 4.25, respectively.

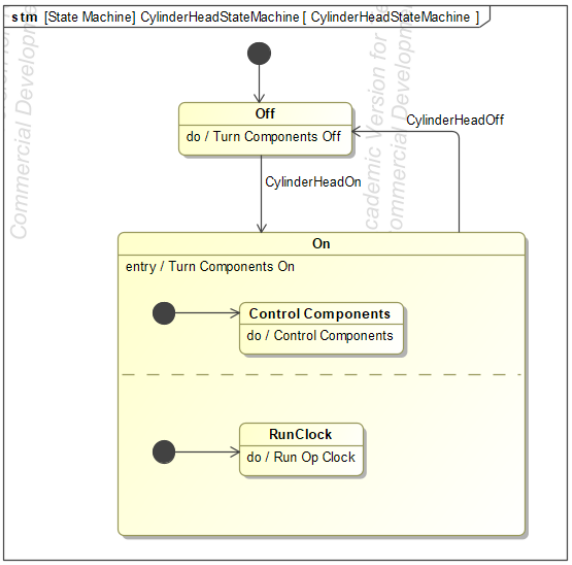

Figure 4.22. State Machine Diagram for CylinderHead Subsystem

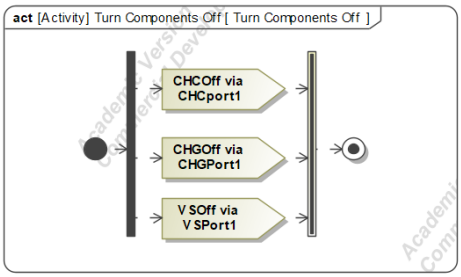

Figure 4.23. Activity Diagram for Turning Off Components under Cylinder-Head Subsystem

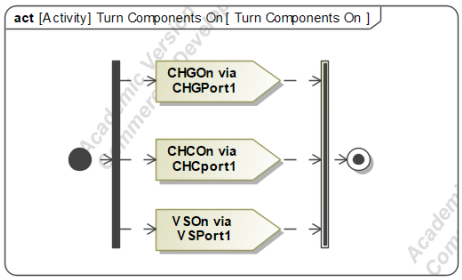

Figure 4.24. Activity Diagram for Turning On Components under Cylinder-Head Subsystem

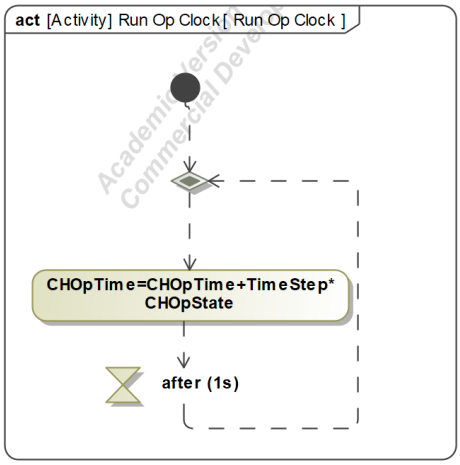

Figure 4.25. Activity Diagram for Running Operation Clock for CylinderHead Subsystem

In the activity diagram for the controlling of components under the CylinderHead subsystem shown in Figure 4.26, the subsystem reads signals from the components and turn off or on the other components accordingly. For example, when the CylinderHead subsystem receives the signal "VSPMEnd" or "RepairComplete_VS" from the valveseat component which implies that the valveseat is ready to operate or turns on, it will send signals to turn on the other two components. In addition, once all components are turned on, a signal "CylinderHeadOperative" will be sent through the CylinderHead interface to trigger the operating state at the CylinderHead subsystem level.

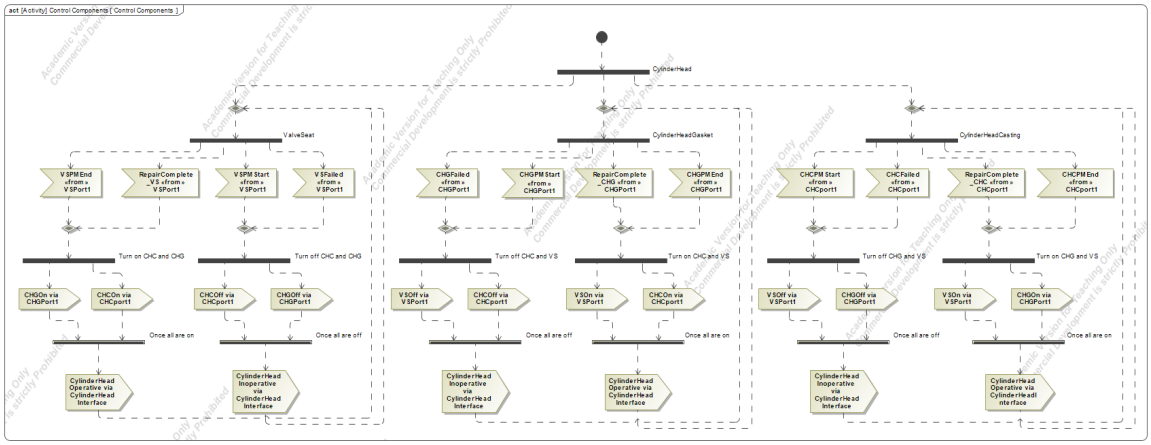

Figure 4.26. Activity Diagram for Controlling Components under Cylinder-Head Subsystem

Following the same way in constructing the interfaces between the engine system and the subsystems to trigger state transition, the IBD in Figure 4.27 depicts the interfaces between the CylinderHead subsystem and the components.

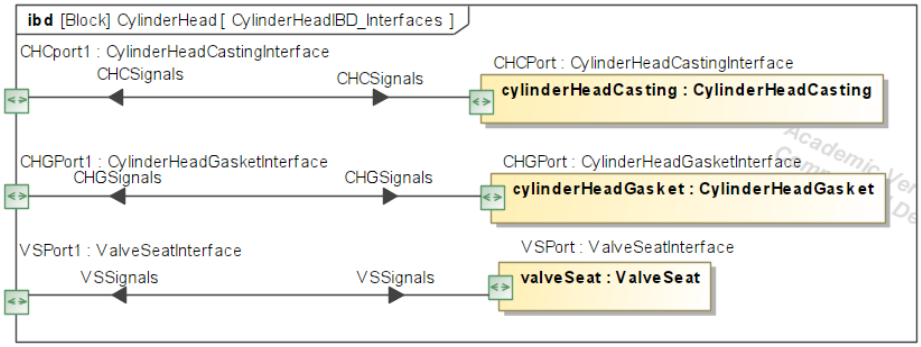

Figure 4.27. Internal Block Diagram for the CylinderHead Subsystem Interfaces with Components

4.5 Component Level Description

Figure 4.28 shows the uses of state machine diagrams and activity diagrams associated to the state machines. For the component level block, state machine diagram is created to represent the 5 different states: 1) Off, 2) On, 3) Faulted, 4) Repair and 5) Preventative Maintenance. The component initial state is inoperative or Off mode. Within each state, an activity diagram is used to define the detailed behaviors of the blocks and transition from one state to another or one block to another, in terms of 1) entry, 2) do and 3) exit behaviors.

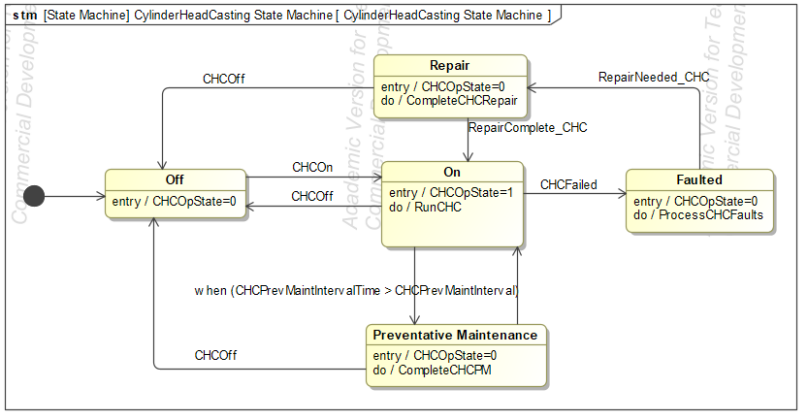

Figure 4.28. State Machine Diagram for Components

For the entry behaviors, opaque behaviors are used to trigger the operation state from 0 (off or inoperative) to 1 (on or operative). Activity diagrams are created to define the do behavior inside each state to perform functions such as running of components, processing fault, completing repair and preventative maintenance, shown in Figures 4.29, 4.31, 4.32 and 4.33. In addition, activity diagram is created to invoke other sub-activity such as checking for fault, shown in Figure 4.30.

When the component (CylinderHeadCasting) is turned on via the signal CHCOn, the state changes to On state triggering both the entry and do behaviors. The do behavior is represented by an activity diagram shown in Figure 4.30. Opaque actions in this do behavior function to add time step to both the component operation time counter and preventative maintenance time interval counter.

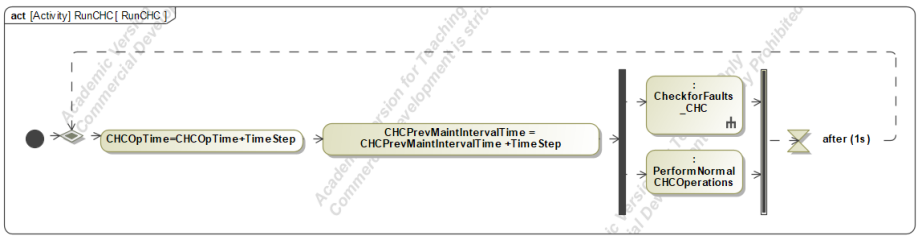

Figure 4.29. Activity Diagram for Running CylinderHeadCasting Component

Under the do behavior AD, Figure 4.30 depicts the sub-activity diagram called "Checkfor-Faults_CHC" which checks on which failure mode(s) has or have occurred. This failure occurrence is triggered by utilizing a random number generator based on comparison of the random number generated with constant failure rate. When the specified failure mode(s) is or are activated (failure mode = 1), the number of failure modes for that component adds up and sends the signal of component failure ("CHCFailed") to trigger the next state of Faulted and signal the subsystem.

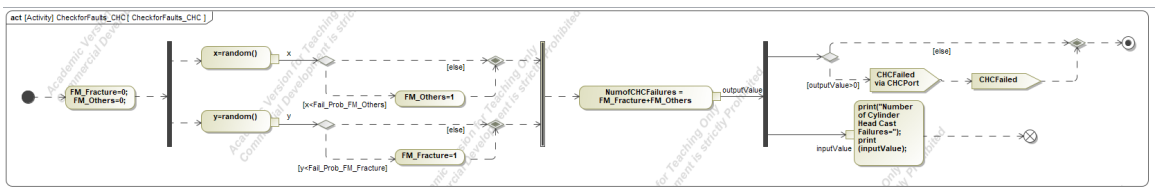

Figure 4.30. Activity Diagram for Sub-activity; Check for Faults

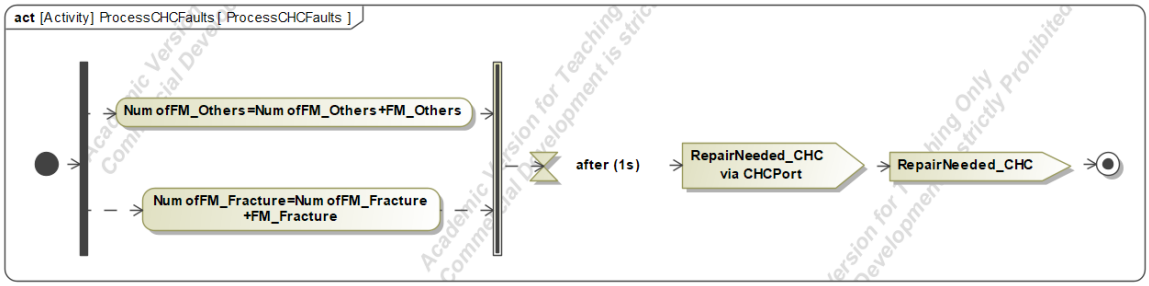

Figure 4.31. Activity Diagram for Processing Fault

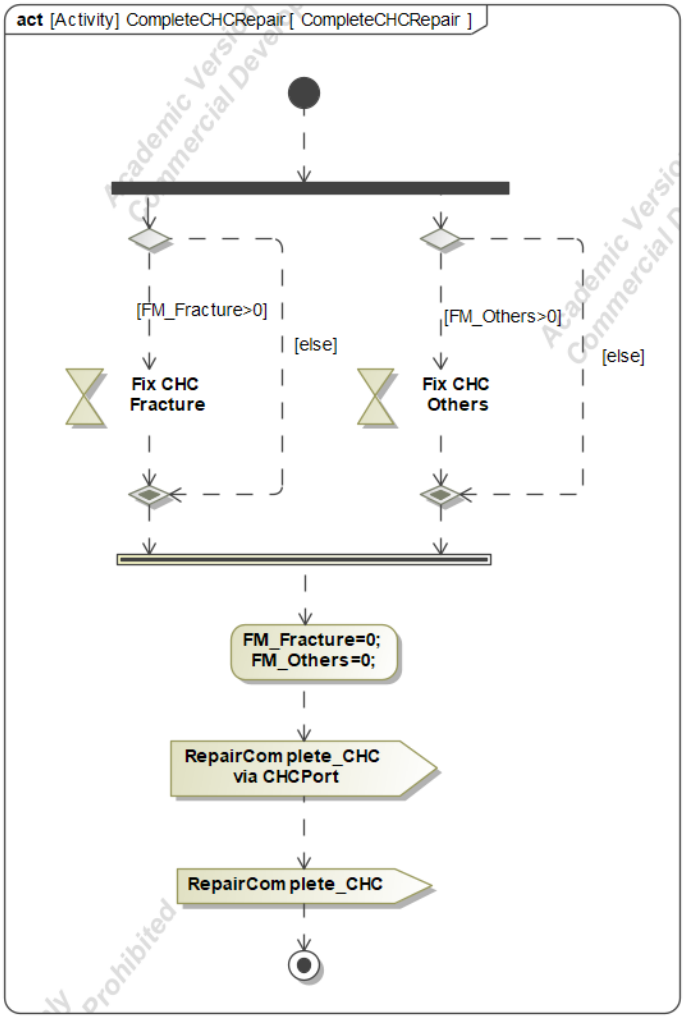

Figure 4.32. Activity Diagram for Completing Repair

Figure 4.33. Activity Diagram for Completing Preventative Maintenance

4.6 Calculating Equations

Having assigned the parameters of interest as value properties at the system, subsystem and component levels, parametric diagrams are used to bind these value properties as constraint parameters and insert calculating equations to compute the desired result parameters. Parametric diagrams are used to calculate the system and subsystem operational states, and availability metrics.

4.6.1 Operational State

Starting at the component level, the "OpState" is defined using opaque behavior. The subsystem "OpState" is then constrained by the "OpStates" of its components as shown in Figure 4.34. For simplicity, the structure function of the subsystems is defined to be in series connection as given by the equation, where each component OpState value is either zero representing that the component has failed or one representing the component is operational.

$$\begin{aligned} \text{“}Subsystem''OpState = \text{“}Component''OpState\times \\ \text{“}Component''OpState \times \text{“}Component''OpState... \end{aligned} \tag{4.1}$$

With the subsystems' "OpStates" derived from its associated components', the "OpState" of higher level which is the system level is then constrained in the same manner as computed and shown in the equation and Figure 4.35.

$$\begin{aligned} \text{“}System''OpState = \text{”}Subsystem''OpState \\ \times\text{“}Subsystem''OpState \times \text{“}Subsystem''OpState... \end{aligned} \tag{4.2}$$

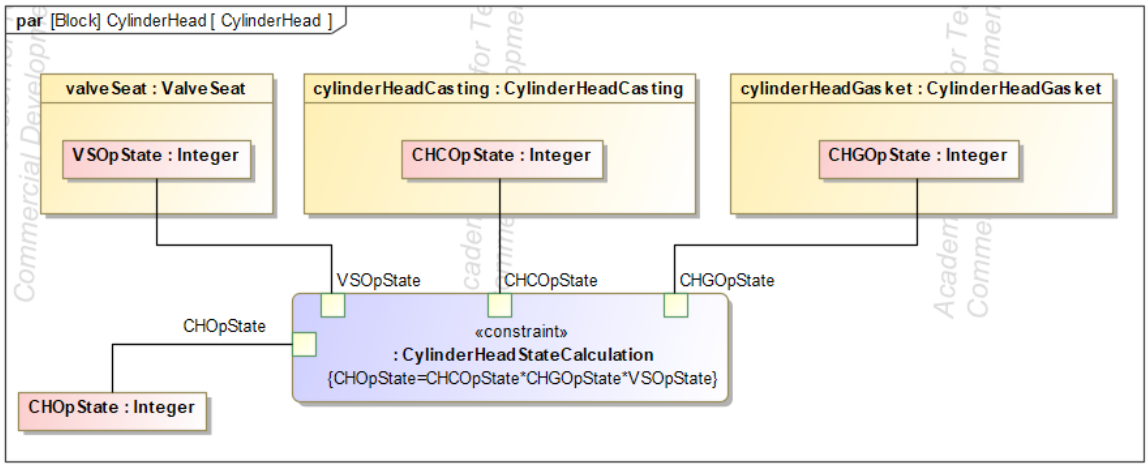

Figure 4.34. Parametric Diagram for Subsystem; CylinderHead Operation State Calculation

4.7 Measures of Effectiveness – Operational Availability

The operational availability is defined as the total uptime divided by the sum of uptime and downtime. In the model development process, when one component transitions from "On" state to any other state, a signal is sent to the subsystem level to turn off other associated components. This logic also applies at the subsystem level. Therefore, uptime is further defined as the time that the asset has been in "On" state (OpState = 1) and the sum of uptime and downtime is determined as the calendar time; total time that the parent asset has been in both "Off" and "On" states. The following subsections summarize the uses of parametric diagrams with the equations at each level. Subsection 4.7.1 and the associated Figure 4.36 and Equation 4.3 show the component level. Subsection 4.7.2 and the associated Figure 4.37 and Equation 4.4 show the subsystem level. Subsection 4.7.3 and the associated Figure 4.38 and Equation 4.5 show the system level.

4.7.1 Component Level

$$\text{“}Component\text{”}\_Ao = \frac{\text{“}Component\text{”}OpTime}{\text{“}Calendar\text{”}OpTime} \tag{4.3}$$

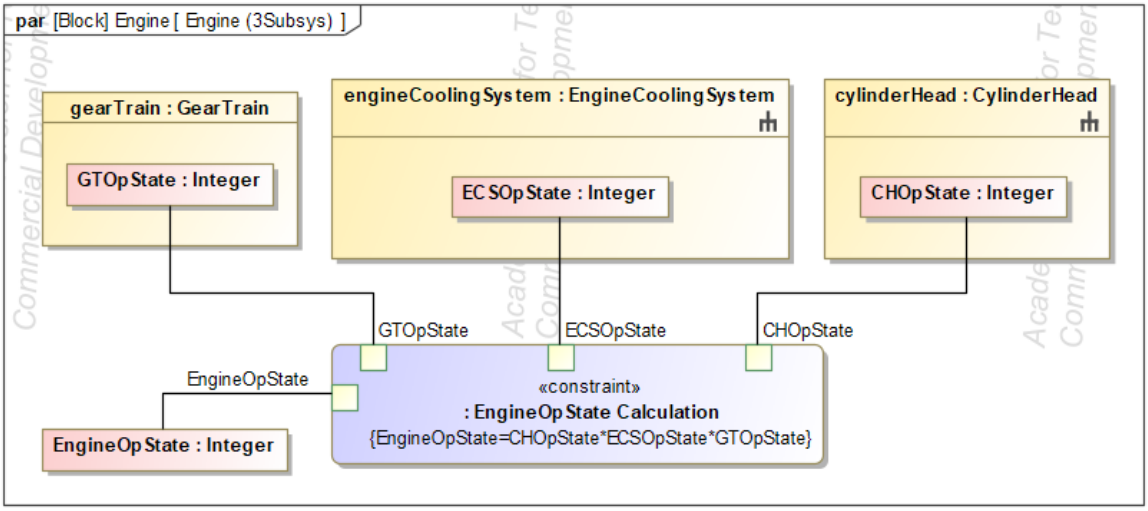

Figure 4.35. Parametric Diagram for Engine System Operation State Calculation

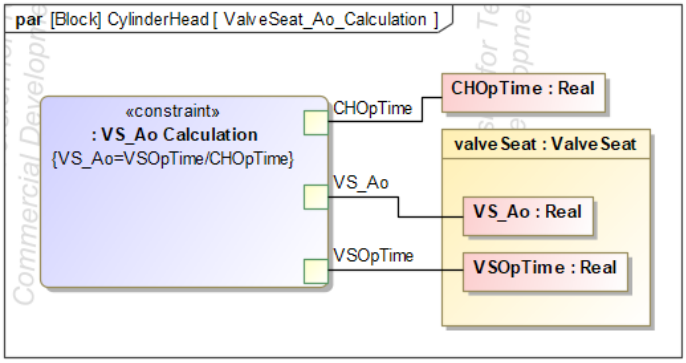

Figure 4.36. Parametric Diagram for ValveSeat Component Ao Calculation

4.7.2 Subsystem Level

$$\text{“}Subsystem\text{”}\_Ao = \frac{\text{“}Subsystem\text{”}OpTime}{\text{“}Calendar\text{”}OpTime} \tag{4.4}$$

4.7.3 System Level

$$\text{“}System\text{”}\_Ao = \frac{\text{“}System\text{”}OpTime}{\text{“}Calendar\text{”}Time} \tag{4.5}$$

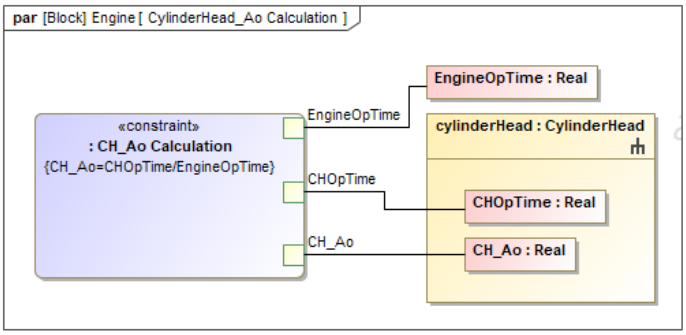

Figure 4.37. Parametric Diagram for CylinderHead Subsystem Ao Calculation

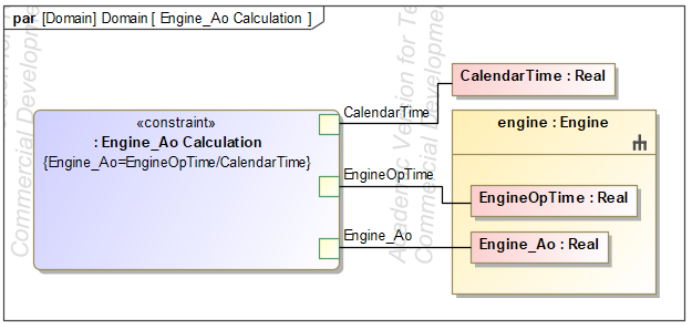

Figure 4.38. Parametric Diagram for Engine System Ao Calculation

Now that the model has been developed based on an engine case study to emulate the failure behaviors with value properties capturing the various failure metrics. The next step is to execute the model simulation and generate results.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Results

In this chapter, a summary of the findings from each step of the proposed methodology in Chapter 3 is presented. The primary focus of this section is on the computed forecast results from the model stimulation, as well as the solutions that are available for implementation based on these findings.

The proposed methodology shows that MBSE can: 1) allow for tracing of failure modes; 2) shows how the “Preventative Maintenance" and "Corrective Maintenance" states of the various components and subsystems interact with each other, 3) predict the number of failures and provide forecasts for operational availability at each level from the whole system to each individual component. Lastly from the aforementioned steps, 4) to analyze the simulation results in support of decision making for maintenance planning.

5.1 Automatic Traceability of Failure Modes

The intent identified is to enable traceability of failure modes down to the component level and improve maintenance support in both corrective and preventative actions. Using an engine DMFEA [36] to represent the SOI and create the model, it reduces the efforts in understanding the entire SOI from scratch on the architecture relations and coming up with credible failure modes and the resulting consequences. The model incorporates the consequences of the failures through the signal patterns allowing the consequences to cascade through the system automatically. For example, the consequence of a Cylinder Head Casting Failure from fracture is automatically transmitted up through the Cylinder Head and results in an Engine stoppage. While this system is failure simple and the result on the engine easily foreseen, in a more complex system some of the consequences of a particular failure may not be obvious. The model also captures specific parameters of interest of 1) operation state (OpState), 2) operation time (OpTime), 3) number of failure modes (NumofFM) and number of preventative maintenance (NumofPM). From these parameters, the intent can then be met through the following steps of the proposed methodology.

5.2 Interactions of Preventative and Corrective Maintenance States

The model was set up to emulate the failure scenarios of the SOI at all levels. First, the user turns on the SOI, which serves as the engine, followed by the transmission of signals from the system level to the subsystems, and then from the subsystem level to the components. The operations will continue until 1) a component failure triggers a shutdown and calls for repair or 2) preventative maintenance occurs. When a component failure or preventative maintenance occurs, signals are transmitted to the subsystem level to disable additional components and declare the subsystem inoperative. Then, signals are transmitted from the inoperative subsystem to communicate with the system and turn off other subsystems. Through these interactions, the number of failure mode occurrence, failed components, repairs performed, and preventative maintenance performed are tracked and recorded. In addition, to compute the intended MOE, which is operational availability, the time between state changes is obtained from these interactions.

The model configuration shown in Figure 5.1 was constructed with only three subsystems to demonstrate the proposed methodology. The structure functions of the components and subsystems are connected in series and small constant failure probabilities are used. As the failure scenario is designed to be simple, it is limited to the single-failure condition; meaning that when there are two failures happening at the same time, the model simulation will halt.

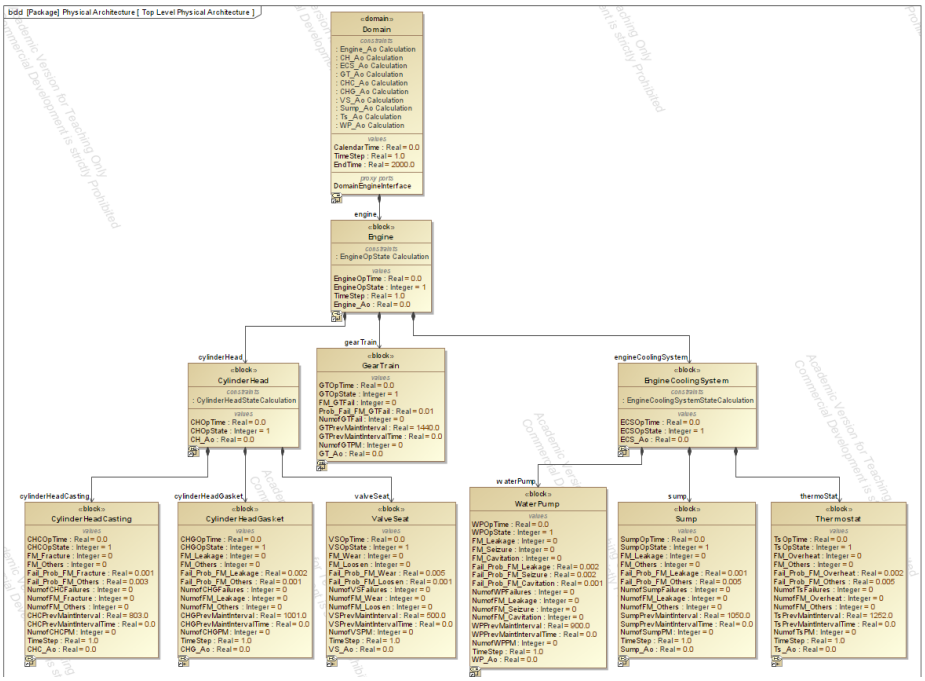

Figure 5.1. Overall Model Architecture

5.3 Using Model Simulation to Predict Number of Failures and Forecast Availability

Using a simulation configuration shown in Figure 5.2, MSOSA's simulation function enables the execution of distinct sets of defined instances. The domain block is set as the execution target of the simulation configuration.

For this study, a total of sixty instances (twenty instances each) of EndTime values of 500 steps, 1,000 steps, and 2,000 steps were executed. This allowed the author to evaluate the model at several intervals, specifically to: 1) introduce preventative maintenance at 1,000 steps and 2,000 steps, 2) induce more failure occurrences over time, and 3) evaluate, following the execution of the aforementioned steps and determine whether there were any significant differences in the results, specifically regarding operational availability, at each level of the SOI.

Figure 5.2. Simulation Configuration

Figure 5.3 shows a partial extract of the raw data generated from an instance table of the model elements. Note, boxes placed in the instance table indicate where the item falls in the system, subsystem, and component physical hierarchy. The raw data were then post-processed, and the results are discussed in the following subsections.

				System OpTime	System Ao	Subsystem OpTime		Subsystem Ao	Component OpTime & Ao		Component PM & NumofFailures		
#	Name	CalendarTime : Real	EndTime : Real	engine.EngineOpTime : Real	engine.Engine_Ao : Real	engine.cylinderHead.CHOpTime : Real		engine.cylinderHead.CH_Ao : Real	engin	engine.	eng	engin	engin
1	domain	2000	2000	714	0.357	988		0.494	1002	0.501	1	2	1
2	domain1	2000	2000	629	0.3145	985		0.4925	1002	0.501	1	0	2
3	domain2	2000	2000	693	0.3465	992		0.496	1002	0.501	1	2	2
4	domain3	2000	2000	771	0.3855	982		0.491	1002	0.501	1	2	4
5	domain4	2000	2000	799	0.3995	992		0.496	1002	0.501	1	0	3
6	domain5	2000	2000	560	0.28	985		0.4925	1002	0.501	1	2	1
7	domain6	2000	2000	667	0.3335	984		0.492	1002	0.501	1	2	4
8	domain7	2000	2000	754	0.377	991		0.4955	1002	0.501	1	1	2
9	domain8	2000	2000	453	0.2265	986		0.493	1002	0.501	1	3	3
10	domain9	2000	2000	733	0.3665	984		0.492	1002	0.501	1	0	5
11	domain10	2000	2000	811	0.4055	988		0.494	1002	0.501	1	2	4
12	domain11	2000	2000	685	0.3425	993		0.4965	1002	0.501	1	1	1
13	domain12	2000	2000	576	0.288	981		0.4905	1002	0.501	1	0	2
14	domain13	2000	2000	803	0.4015	987		0.4935	1002	0.501	1	3	1
15	domain14	2000	2000	775	0.3875	990		0.495	1002	0.501	1	2	3
16	domain15	2000	2000	683	0.3415	982		0.491	1002	0.501	1	1	5
17	domain16	2000	2000	557	0.2785	980		0.49	1002	0.501	1	2	1
18	domain17	2000	2000	694	0.347	993		0.4965	1002	0.501	1	0	2
19	domain18	2000	2000	478	0.239	981		0.4905	1002	0.501	1	3	4
20	domain19	2000	2000	689	0.3445	983		0.4915	1002	0.501	1	1	3

Figure 5.3. Partial Extract of Raw Data Generated

5.3.1 Problems Discovered During Simulation

The data from the various runs was exported into MSExcel and examined several times. Upon review, two issues were discovered. The first issue was a race condition, and the second issue was that various clock counters did not run at the same speed. These are described in detail in following subsection.

Race Conditions

It was found that the model as currently built is vulnerable to simultaneous failures, which sets up a race condition [37], where multiple Off and On signals get sent to the components or subsystems, which often lead to a component being locked in an incorrect state. It was anticipated that this condition would be the same for multiple preventative maintenance events if the timing aligned. Therefore, the preventative maintenance intervals were purposefully set so alignment could not occur but still close to the original values. It was found that that was not enough due to the operation of MSOSA simulation.

As shown partially in Figure 5.4, it was noted that there were some data columns (CHCOpTime, CHC_Ao, CHGOpTime, CHG_Ao, VSOpTime, VS_Ao) at the component level generating the same values for all twenty instances at EndTime value of 2000 steps. This led to another review of the model-setup and the cause was due to the common multiple values with too close tolerance for the ValveSeat's preventative maintenance interval (500steps) and the CylinderHeadGasket's preventative maintenance interval (1001steps). This race condition caused all the OpTime and Ao of the three components to be the same.

#	Name	CalendarTime : Real	EndTime : Real	engine.Engine OpTime : Real	engine.Engine _Ao : Real	engine.cylinderHead.CHOpTime : Real	engine.cylinderHead.CHC_Ao : Real	engine.cylinderHead.cylinderHeadCasting.CHCOpTime : Real	engine.cylinderHead.cylinderHeadCasting.CHC_Ao : Real	engine.cylinderHead.cylinderHeadCasting.NumofCHCPM : Integer	engine.cylinderHead.cylinderHeadCasting.NumofFM_Fracture : Integer	engine.cylinderHead.cylinderHeadCasting.NumofFM_Others : Integer	engine.cylinderHead.cylinderHeadGasket.CHGOpTime : Real	engine.cylinderHead.cylinderHeadGasket.CHG_Ao : Real
1	domain	2000	2000	714	0.357	988	0.494	1002	0.501	1	2	1	1002	0.501
2	domain1	2000	2000	629	0.3145	985	0.4925	1002	0.501	1	0	2	1002	0.501
3	domain2	2000	2000	693	0.3465	992	0.496	1002	0.501	1	2	2	1002	0.501
4	domain3	2000	2000	771	0.3855	982	0.491	1002	0.501	1	2	4	1002	0.501
5	domain4	2000	2000	799	0.3995	992	0.496	1002	0.501	1	0	3	1002	0.501
6	domain5	2000	2000	560	0.28	985	0.4925	1002	0.501	1	2	1	1002	0.501
7	domain6	2000	2000	667	0.3335	984	0.492	1002	0.501	1	2	4	1002	0.501
8	domain7	2000	2000	754	0.377	991	0.4955	1002	0.501	1	1	2	1002	0.501
9	domain8	2000	2000	453	0.2265	986	0.493	1002	0.501	1	3	3	1002	0.501
10	domain9	2000	2000	733	0.3665	984	0.492	1002	0.501	1	0	5	1002	0.501
11	domain10	2000	2000	811	0.4055	988	0.494	1002	0.501	1	2	4	1002	0.501
12	domain11	2000	2000	685	0.3425	993	0.4965	1002	0.501	1	1	1	1002	0.501
13	domain12	2000	2000	576	0.288	981	0.4905	1002	0.501	1	0	2	1002	0.501
14	domain13	2000	2000	803	0.4015	987	0.4935	1002	0.501	1	3	1	1002	0.501
15	domain14	2000	2000	775	0.3875	990	0.495	1002	0.501	1	2	3	1002	0.501
16	domain15	2000	2000	683	0.3415	982	0.491	1002	0.501	1	1	5	1002	0.501
17	domain16	2000	2000	557	0.2785	980	0.49	1002	0.501	1	2	1	1002	0.501
18	domain17	2000	2000	694	0.347	993	0.4965	1002	0.501	1	0	2	1002	0.501
19	domain18	2000	2000	478	0.239	981	0.4905	1002	0.501	1	3	4	1002	0.501
20	domain19	2000	2000	689	0.3445	983	0.4915	1002	0.501	1	1	3	1002	0.501

Figure 5.4. Exported Raw Data (Partially Shown) in MSExcel

Due to time constraints in the model development, an interim solution was implemented to resolve the racing condition by changing the ValveSeat's preventative maintenance interval to a non-common multiple value with bigger tolerance; i.e. 835. It is recommended that the activity that controls the components be changed to eliminate the race condition.

All Counters Do Not Run at the Same Speed

A one-second delay was incorporated in each of the activity diagrams that had a counter, assuming that would result in the counters counting at the same rate. This assumption was proven incorrect. It was discovered both in post-processed data and during active simulations that the activities which executed counter functions did not run at the same speed, with some counters running up to 50% faster than the "calendar time" counter executed in the Domain block. A resolution of moving all counter/timing actions into the same activity with the calendar time counter was explored to align the counter functions, which was successful but other problems were encountered. There was not time to fully revise the model and further de-bug the model. This issue is left for future work.

Once these two issues are resolved, the author is confident that the model will produce forecasts to support more informed decision making.

5.4 Results that Support Decision Making for Maintenance Planning

Some of the goals for digital twin models are to produce forecasts for failures, predict needed logistic support, forecast downtime, and maintenance staffing [15]. From the post-processed data, decision makers can easily see which components or subsystems will likely fail within the forecast period as shown in Figure 5.5.

Beyond the simple average number of failures, one can also develop cumulative probability curves for the range of number of failures. One can generate either the probability distribution function (PDF) or cumulative distribution function (CDF) based on the frequency of failure occurrences from the post processed data for each failure mode shown in Figure 5.6.

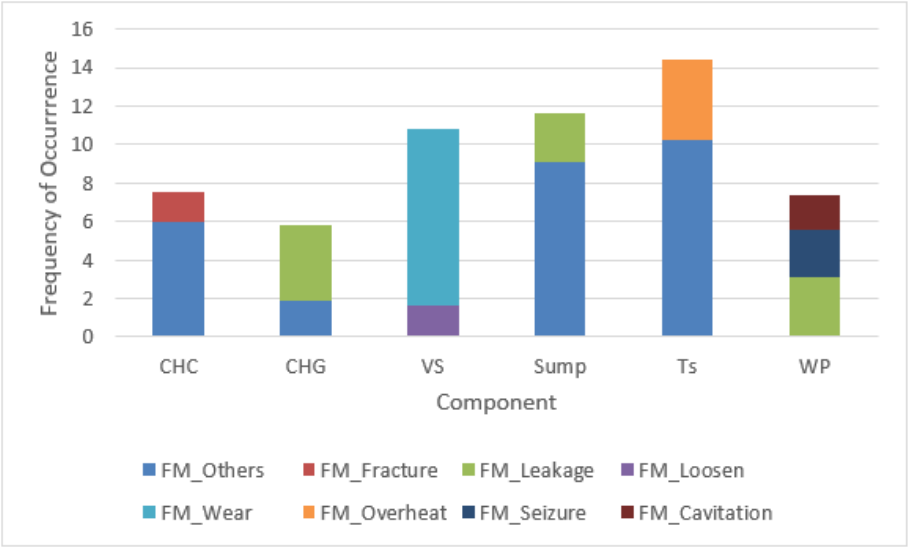

Figure 5.5. Stacked Barchart for Failure Modes under Components

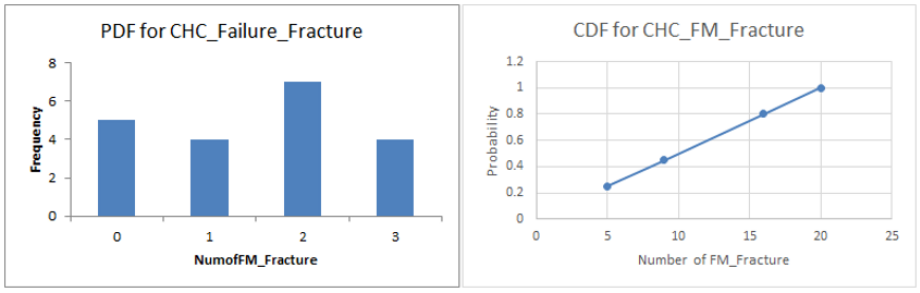

Figure 5.6. PDF and CDF for CylinderHeadCasting's Failure Mode_Fracture

From these cumulative probability curves, a decision maker can set a level of risk that he/she wishes to accept. For example, if the decision maker decides that he wants to set the risk level at 80% for sparing a particular component, then he can examine the cumulative probability curve produced from the post-processed data for that component failure and look up value associated with the 80% likelihood to determine how many spares he should provision. From Figure 5.6 , one could determine that 16 or fewer fractures would be forecast to take place. So, if one provisioned 16 spares, assuming the repair is simply to replace the part, there would be a 20% chance of not having enough spares, and four more spares would cover all of the forecasted failures.

In addition, one can get similar cumulative probability curves for the operation time (or Ao).

These curves (data) can be entered into more sophisticated decision models to determine whether the system could contribute to the warfighting capabilities needed. Examples are shown in Figure 5.7.

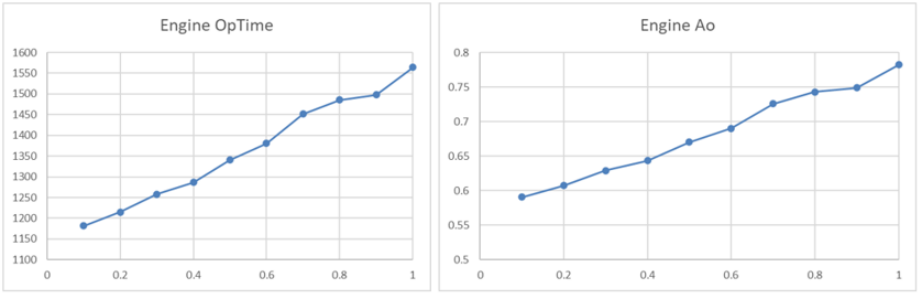

Figure 5.7. Cumulative Probability Charts for EngineOpTime and Ao

If a single number is needed in lieu of the cumulative probability curve, then one could use the median or average values developed in the forecast, though with the cumulative probability curves one could provide box and whisker chart showing the range of values obtained as well as the median and 25th and 75th percentiles as shown in Figure 5.8. This additional uncertainty information would better inform the decision makers.

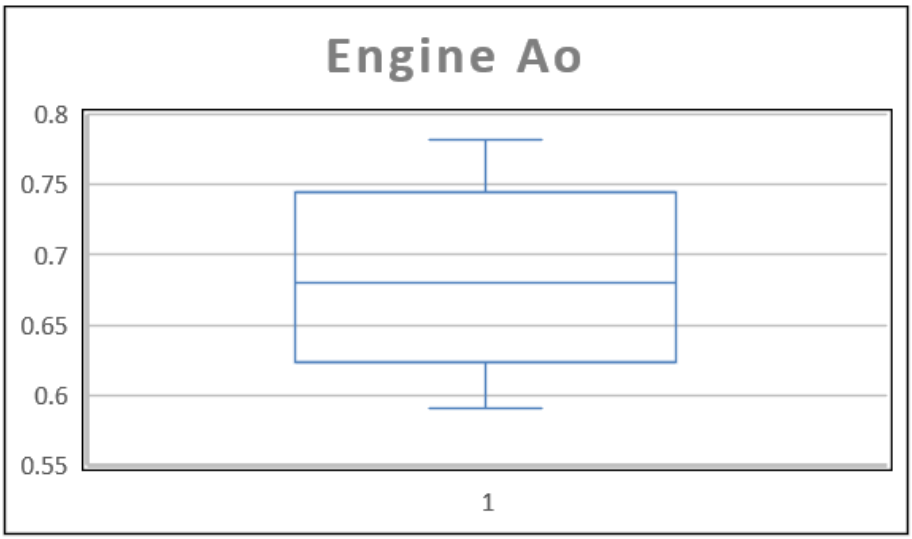

Figure 5.8. Boxplot for Engine Ao

With further development of the model, one could sum maintenance time for the various failures to help with forecasting maintenance staffing and maintenance supplies, coming up

with better maintenance plans to make sure there are enough spare parts and people to do corrective and preventative maintenance.

Longer term decisions could also be made about those components with a high rate of failure, especially those with a high-cost value. After further failure analysis by other techniques, possible solutions include making changes to the design of the component to reduce or eliminate the chance of failure.

5.5 Chapter Summary

The findings from the development of model provided insights on the failure occurrences within the SOI. It has shown that by using MBSE approach, known potential failures can be traced and tagged to individual components. Additionally, the transition of states within and from component level to subsystem and system level can be illustrated in the MBSE simulation model. This enabled the simple form of prediction in number of failures and forecasts for operational availability. From the simulation results, once the simulation problems; race condition and counter clock issues have been resolved, the model is expected to enhance the support for maintenance planning.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 6:
Conclusion and Recommendation

6.1 Research Findings and Summary

This thesis research has demonstrated the uses of a digital twin that can emulate the virtual model of a system (in this thesis; an engine) and be applied in the process of failure analysis to improve failure and availability predictions. The uses of model development in gradually increasing difficulty from the domain level down to the component level and the simulation testing in step intervals from 500 steps, 1000 steps and lastly 2000 steps allowed practitioners to build similar model concept cumulatively. In conclusion of the research, the research questions were revisited and presented in Table 6.1 below.

Table 6.1. Research Questions

S/N	Research Questions
	Primary Questions
1	What is an effective system engineering approach to analyze the impact of failures using a DT?
	By using MSOSA, a MBSE software that communicates in SysML, the impact of failures can be computed in terms of number of failures and operation time before failure at any abstraction level of the SOI.
2	How can the DT be used not only in the early stages of design or system development but throughout the entire lifecycle of the system to support failure analysis?
	The development of the DT model showed a typical scenario on how the engine system transit from operational state to faulted, preventative and corrective maintenance states. The value properties in the simulation model can be changed accordingly to suit parameters specified in the design requirement during design development phase or tagged to the parameters measured and collected by sensors in real time data during product testing or fielding.
	Continued on next page

Table 6.1 – continued from previous page

S/N	Research Questions
3	How does applying the DT to failure analysis influence important decisions on Preventative Maintenance and Corrective Maintenance tasks?
	The primary data from the model simulation allows simple understanding of the failure occurrences within the levels of abstraction. The CSV export function of MSOSA then allows the use of external analytical statistics tools like MSExcel to perform data distributions such as CDF and PDF for the operation time (or Ao) of any subsystem or component which can potentially contribute to the decision-making process on PM and CM tasking.
Secondary Questions	
4	How can the DT model help alleviate the sparing and staffing issues in military context?
	With further improvement of the model, one may sum maintenance time for various failures to anticipate maintenance personnel and supplies, coming up with improved maintenance plans to ensure there are adequate spare parts and workers to execute corrective and preventative maintenance. Long-term decisions could be made about high-cost, high-failure-rate components. After failure analysis using other methods, viable fixes include improving the component's design.
5	What are the potential areas for future research with this model development process as baseline?
	The potential areas include expanding the model architecture in performing other computation of other parameters of interest such as likelihood detection, connecting the component and subsystem blocks in various structure functions for realism of actual scenario, and replacing the small constant failure probabilities assigned with probabilities based on hazard functions over time.

The study for the thesis began with a review of the key challenges shared by the fields of failure analysis and system engineering. Due to the limitations of DBSE such as disconnection between design artifacts and a lack of competence level in failure analysis to detect the source of the failure, there were lapses in the system lifecycle where system failures may occur. The identified challenges served as the impetus for the development of the methodology.

The methodology added the failure modes and preventative maintenance events into the digital twin MBSE model. Failure modes were defined using standard reliability techniques such as Failure Mode Effect Analysis (FMEA). This study employed state machines to model component level failure. To account for the failure and preventative maintenance behavior of the component, new states are added to the state machine diagram of the component. These states include Faulted, Repair, and preventative Maintenance. This work investigated the incorporation of information from component failure modes and failure probability to comprehend system-level consequences.

For subsystems with components, the subsystem has a state machine comprised Off and On states. Depending on the subsystem's position in the system hierarchy, the transition between these states is governed by control signals from the next higher level subsystem or the top system block. The subsystem then regulates its components using control signals, which cause the state machines of the components to transit between the Off and On states. The lower level components convey their status, including failures or preventative maintenance events, to the next higher level subsystem or system via status signals.

Modifications were made to the general Domain block to control the desired scenario settings. Additional value characteristics were added to count CalendarTime with the TimeStep, and the scenario would conclude at the EndTime, the moment at which the simulation will stop. A state machine for the Domain block was constructed to govern the execution of the simulated scenario and, at the conclusion of the scenario, to support data capture through post-processing computations.

While state machines are used to manage the general behavior of the system, subsystem, and components, additional SysML diagrams were necessary to define the structure and behavior of the entire system and its components. Diagrams such as activity diagrams, block definition diagrams, internal block diagrams, and parametric diagrams were used to conduct various availability calculations.

Models were run in simulations starting at zero time to emulate new equipment. The various component OpTime properties could be set at later times to represent equipment that had already been used. Therefore, this methodology could be extended to be used throughout the system lifecycle to support failure and support analysis.

Despite the underlying issues discovered in the model development, the methodology has fulfilled the primary intent of the thesis work in providing traceability of all known failures in the system and a virtual understanding of how a typical system will transition between states in the context of failure analysis and maintenance. Furthermore, it has proven to enable computing capabilities in running multiple instances of simulation to generate parameters of interest, providing good estimates of the results from post-processing if the underlying issues are resolved.

6.2 Recommendations and Future Work

Before touching on the potential areas for future work, it is recommended that the model's underlying issues be resolved first. Described in sub-section 5.3.1, the issues are eliminating race conditions and correcting the step counter issue by modifying the model.

This model emulates the actual system's behavior more closely than traditional reliability models and may offer a more accurate estimate of system availability. This claim must be validated by doing a comparison of the results from MBSE digital model of a real-system and the corresponding real-world system.

MSOSA was chosen for model development because it uses SysML, is compatible with MSExcel, and promotes architecture development in simulation modeling. Unexpectedly, simulating just one instance of a model architecture in MSOSA with just four layers of complexity would have taken an extremely longer period. Other applications, such as Simulink or CORE, offer comparable MBSE approaches for model architecture and may be considered for this model development process.

The proposed methodology only illustrates the uses of a subsystems with components to the system and the traceability of failure modes to the components from a DFMEA study. It is possible to create a DT representing the entire SOI and utilize other parameters of interest such as the causes, likelihood of detection and severity of failure to perform a full-scale DFMEA.

For the purposes of understanding the model development process, series structure function and simple calculation equations are used to define the OpState and compute the system operational availability. One may build the relations between each abstraction level to

illustrate a more complex connection. Furthermore, the model currently uses a constant failure rate to trigger failure occurrence, and this can be extended to failure probabilities that accords to wear and tear conditions or hazard functions that vary with time.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] INCOSE, "INCOSE Systems Engineering Vision 2035," 2022 [Online]. Available: https://www.incose.org/about-systems-engineering/se-vision-2035

[2] Office of the Under Secretary of Defense for Research and Engineering, "Systems engineering guidebook," Department of Defense, Washington, DC USA, Tech. Rep., 2022.

[3] Department of Defense, "DoD Guide for Achieving Reliability, Availability, and Maintainability," Department of Defense, Tech. Rep., 2005, Washington, DC USA.

[4] Y. Jackson, P. Tabbagh, P. Gibson, and E. Seglie, "The new Department of Defense (DoD) guide for achieving and assessing RAM," in *Annual Reliability and Maintainability Symposium, 2005. Proceedings.* IEEE, 2005, pp. 1–7.

[5] O. Lynch, "Reducing logistics delays using the supply chain criticality index: A diagnostic approach," M.S. thesis, Dept. of Sys. Engr., NPS, Monterey, CA, USA, 2020 [Online]. Available: http://hdl.handle.net/10945/65395

[6] S.H.G Teng and S.Y.M. Ho, "Failure mode and effects analysis: an integrated approach for product design and process control," *International journal of quality & reliability management*, 1996, doi: 10.1108/02656719610118151.

[7] A. A. Baig, R. Ruzli, and A. B. Buang, "Reliability analysis using fault tree analysis: a review," *International Journal of Chemical Engineering and Applications*, vol. 4, no. 3, p. 169, 2013.

[8] M. Vinarcik and P. Cadzynski, "Systems architecture: The connective tissue of your digital thread," 2020, presentation at Product Innovation DX. [Online]. Available: https://events.pi.tv/2020/dx/agenda.

[9] R. A. Noguchi, "Recommended best practices based on MBSE pilot projects," in *INCOSE International Symposium*, no. 1. Wiley, 2019, vol. 29, pp. 753–770.

[10] M. Hecht, A. Chuidian, T. Tanaka, and R. Raymond, "Automated generation of FMEAs using SysML for reliability, safety, and cybersecurity," in *2020 Annual Reliability and Maintainability Symposium (RAMS)*, 2020, pp. 1–7.

[11] K. Diatte, B. O'Halloran, and D. L. Van Bossuyt, "The integration of reliability, availability, and maintainability into model-based systems engineering," *Systems*, vol. 10, no. 4, p. 101, 2022.

[12] S. Monzon, M. Pittard, D. Boudreau, B. Nall, C. Peter, and B. McCarragher, "Efficacy of model based systems engineering (MBSE) for test and evaluation," in *National Defense Industrial Association 19th Annual Systems Engineering Conference*. NIDA, 2016.

[13] J. D'Ambrosio and G. Soremekun, "Systems engineering challenges and MBSE opportunities for automotive system design," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2017, pp. 2075–2080.

[14] L. E. Hart, "Introduction to model-based system engineering (MBSE) and SysML," 2015, Presentation at Delaware Valley INCOSE Chapter Meeting.

[15] J. Bickford, D. L. Van Bossuyt, P. Beery, and A. Pollman, "Operationalizing digital twins through model-based systems engineering methods," *Systems Engineering*, vol. 23, no. 6, pp. 724–750, 2020.

[16] D. L. Allaire, "A physics-based emissions model for aircraft gas turbine combustors," Ph.D. dissertation, Dept. of Aero. and Astro., Massachusetts Institute of Technology, Cambridge, MA, USA, 2006.

[17] N. Shevchenko, "An introduction to model-based systems engineering (MBSE)," Software Engineering Institute, blog [Online], 2020. Available: https://insights.sei.cmu.edu/blog/introduction-model-based-systems-engineering-mbse/

[18] V. Salehi, "Development of an agile concept for MBSE for future digital products through the entire life cycle management called Munich Agile MBSE Concept (MAGIC)," *Computer-Aided Design and Applications*, vol. 17, no. 1, pp. 147–166, 2020.

[19] M. Rausand, A. Barros, and A. Hoyland, *System reliability theory: models, statistical methods, and applications*, 3rd ed. Hoboken, NJ, USA: Wiley, 2020.

[20] J. B. Bowles, "Failure modes and effects analysis," in *Failure Analysis and Prevention*, vol. 11. Materials Park, OH, USA: ASM International, 2002.

[21] J. A. Dean, "A probability risk assessment to support a defendable and quantitative safety assessment of the assault amphibious vehicle," M.S. thesis, Dept. of Sys. Engr., NPS, Monterey, CA, USA, 2018 [Online]. Available: http://hdl.handle.net/10945/60391

[22] J. Moubray, *Reliability-centered maintenance*, 1st ed. New York, NY, USA: Industrial Press Inc., 2001.

[23] R. Yam, P. Tse, L. Li, and P. Tu, "Intelligent predictive decision support system for condition-based maintenance," *The International Journal of Advanced Manufacturing Technology*, vol. 17, no. 5, pp. 383–391, 2001.

[24] B. W. Niebel, *Engineering maintenance management*, 1st ed. Boca Raton, FL USA: CRC Press, 1994.

[25] Y. Tu, "Decision support system for equipment diagnosis and maintenance management: An artificial intelligent approach," 1995, Research Proposal, City University of Hong Kong, HK.

[26] E. Glaessgen and D. Stargel, "The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles," in *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference
20th AIAA/ASME/AHS Adaptive Structures Conference
14th AIAA*. Honolulu, Hawaii: American Institute of Aeronautics and Astronautics, Apr. 2012.

[27] M. Grieves and J. Vickers, "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems," in *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches*, F.-J. Kahlen, S. Flumerfelt, and A. Alves, Eds. Cham, Switzerland: Springer International Publishing, 2017, pp. 85–113.

[28] D. Turbide, "Navigate the future: Digital twin – what's the big deal," 3DS Blogs, blog, 2022 [Online]. Available: https://blogs.3ds.com/northamerica/what-is-the-digital-twin/

[29] MATLAB Simscape, "Predictive maintenance, part 5: Digital twin using MATLAB - MATLAB programming," Apr 2019 [Online]. Available: https://www.matlabcoding.com/2019/04/predictive-maintenance-part-5-digital.html?

[30] K. L. Lueth, "How the world's 250 digital twins compare? Same, same but different." IoT Analytics, blog, Apr. 29, 2020 [Online]. Available: https://iot-analytics.com/how-the-worlds-250-digital-twins-compare/

[31] MATLAB & Simulink, "Models for predicting remaining useful life," [Online]. Available: https://www.mathworks.com/help/predmaint/ug/models-for-predicting-remaining-useful-life.html.

[32] MATLAB & Simulink, "Overcoming four common obstacles to predictive maintenance with MATLAB and Simulink," [Online]. Available: https://www.mathworks.com/content/dam/mathworks/white-paper/gated/predictive-maintenance-challenges-whitepaper.pdf.

[33] O. Casse, *SysML in Action with Cameo Systems Modeler*, 1st ed. Amsterdam, Netherlands: Elsevier, 2017.

[34] A. Morkevicius, A. Aleksandraviciene, D. Mazeika, L. Bisikirskiene, and Z. Strolia, "MBSE grid: A simplified SysML-based approach for modeling complex systems," in *INCOSE International Symposium*, no. 1. Wiley, 2017, vol. 27, pp. 136–150.

[35] M. Rhoades, "System suitability module 2 - reliability analysis," Naval Postgraduate School lecture, Monterey, CA USA, 2022.

[36] P. P. Graceraj, "A novel DFMEA model for high power diesel engine design," *International Journal of Engineering Research & Technology (IJERT)*, vol. 4, no. 05, 2015.

[37] B. Lutkevich, "What is a race condition?" Tech Target, blog, 2021 [Online]. Available: https://www.techtarget.com/searchstorage/definition/race-condition

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California