


2019

A Framework to Develop Anomaly Detection/Fault Isolation Architecture Using System Engineering Principles

Thomas Clark
University of Central Florida

 Part of the [Industrial Engineering Commons](#), and the [Systems Engineering Commons](#)
Find similar works at: <https://stars.library.ucf.edu/etd>
University of Central Florida Libraries <http://library.ucf.edu>

This Doctoral Dissertation (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation

Clark, Thomas, "A Framework to Develop Anomaly Detection/Fault Isolation Architecture Using System Engineering Principles" (2019). *Electronic Theses and Dissertations*. 6834.
<https://stars.library.ucf.edu/etd/6834>

**A FRAMEWORK TO DEVELOP ANOMALY DETECTION/FAULT ISOLATION
ARCHITECTURE USING SYSTEM ENGINEERING PRINCIPLES**

by

THOMAS J. CLARK
B.S.B.A. Rollins College, 1988
M.B.A. University of Central Florida, 1991
B.S.C.S. Rollins College, 1996
M.S.I.E. University of Central Florida, 2012

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Industrial Engineering and Management Systems
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2019

Major Professors: Luis Rabelo, Gene Lee

© 2019 Thomas J. Clark

ABSTRACT

For critical systems, timely recognition of an anomalous condition immediately starts the evaluation process. For complex systems, isolating the fault to a component or subsystem results in corrective action sooner so that undesired consequences may be minimized. There are many unique anomaly detection and fault isolation capabilities available with innovative techniques to quickly discover an issue and identify the underlying problems.

This research develops a framework to aid in the selection of appropriate anomaly detection and fault isolation technology to augment a given system. To optimize this process, the framework employs a model based systems engineering approach. Specifically, a SysML model is generated that enables a system-level evaluation of alternative detection and isolation techniques, and subsequently identifies the preferable application(s) from these technologies

A case study is conducted on a cryogenic liquid hydrogen system that was used to fuel the Space Shuttles at the Kennedy Space Center, Florida (and will be used to fuel the next generation Space Launch System rocket). This system is operated remotely and supports time-critical and highly hazardous operations making it a good candidate to augment with this technology. As the process depicted by the framework down-selects to potential applications for consideration, these too are tested in their ability to achieve required goals.

ACKNOWLEDGMENTS

I would like to thank Dr. Rabelo for his continued guidance and support during this research project. His insightfulness in in the fields of System Engineering and Anomaly Detection were quite valuable. I would also like to thank my committee members, Dr Lee, Dr. Elshennawy and Dr. Yazici. Their time, contributions and patience with this effort is greatly appreciated.

TABLE OF CONTENTS

LIST OF FIGURES	x
LIST OF TABLES	xii
CHAPTER ONE: INTRODUCTION.....	1
Anomaly Detection	2
Fault Isolation	4
Problem Statement	5
Research Objectives	5
Research Contributions	6
Dissertation Organization.....	7
CHAPTER TWO: LITERATURE REVIEW	9
Purpose.....	9
Anomaly Detectors.....	10
Data-Driven Models	10
Models/Algorithms.....	12
Fault Isolators.....	20

Fault Isolation Models	22
Anomaly Detection and Fault Isolation	25
System Engineering Tools	25
Gap Analysis	29
Gap Analysis Observations	34
Literature Review Summary	35
 CHAPTER THREE: RESEARCH METHODOLOGY	 37
Methodology	37
Problem Statement	39
Research Objectives	39
Literature Review	40
Gap Analysis	40
Synthesis.....	40
Preliminary Framework Development	45
Case Study	46
Evaluation.....	48
Framework	48
Conclusion.....	49
 CHAPTER FOUR: PRELIMINARY FRAMEWORK	 50

System Scope	51
Sensor Array.....	52
Determine Potential Faults	53
Fault Reduction from Sensor Capability	55
Anomaly Detection/Fault Isolation Applications.....	57
Model the System.....	59
System Structure	59
System Behavior	61
Constraints.....	62
Requirements.....	63
Existing AD/FI Capabilities	63
Model AD/FI Applications	63
Trade Studies/Application Evaluations.....	64
Make Recommendation(s)	65
CHAPTER FIVE: CASE STUDY	66
Framework Development.....	67
System Scope	68
Identify and Categorize Fault Modes	75

Identify and Categorize Sensor Data.....	89
Identify and Categorize AD/FI Applications	94
Model the System.....	112
Model the AD/FI Applications.....	118
Perform Trade Studies.....	142
Make Recommendations	147
 CHAPTER SIX: ANALYSIS/EVALUATION	 149
Framework Analysis	149
Scope the System	150
Identify and Categorize Sensor Data.....	150
Identify and Categorize Fault Modes	151
Identify and Categorize AD/FI Applications	152
Model the System.....	154
Model the AD/FI Applications.....	156
Proposed Framework	161
Framework Validation	165
 CHAPTER SEVEN: CONCLUSION	 171
Summary	171
Framework	173

Contributions.....	174
Limitations	175
Future Work	176
REFERENCES	178

LIST OF FIGURES

Figure 1 - Anomaly Detection Process Flow Example.....	17
Figure 2 - Potential System Fault Sources	21
Figure 3 - Research Methodology Diagram.....	38
Figure 4 - Preliminary Framework Process Flow	51
Figure 5 - Valve Component Fault Tree	54
Figure 6 - BDD for Remote Operated Valve Assembly	60
Figure 7 - IBD/Remote Operated Valve Assembly	61
Figure 8 - Preliminary Framework Process Flow	68
Figure 9 - LH2 Pressurization System	71
Figure 10 - IBD/LH2StorTankPressSys	82
Figure 11 - Model Repository Example.....	83
Figure 12 – ‘FailureMode’ to Part BDD.....	84
Figure 13 - Relationship Matrix.....	85
Figure 14 - Excel/VBA Generated Partial FT.....	87
Figure 15 – BDD of Remote Valve with 1 or 2 Limit Switches.....	92
Figure 16 - BDD Containing Temperature Indicators	93
Figure 17 – Mission Statement/Requirements Diagram.....	96
Figure 18 - Mission Requirements.....	97
Figure 19 - Requirements AD-FI Diagram.....	98
Figure 20 - BDD/LH2StorTankPressSys.....	114
Figure 21 - IBD/LH2StorTankPressSys	115

Figure 22 - BDD/LH2VapCntrlVlvSys	116
Figure 23 - IBD/LH2VapCntrlVlvSys.....	117
Figure 24 - Single Indicator Test	125
Figure 25 - Multiple Indicator Test (1%).....	126
Figure 26 - Multiple Indicator test (2%)	127
Figure 27 – Baseline Data (Noisy).....	128
Figure 28 - Single Indicator Test (noisy).....	129
Figure 29 - Multiple Indicator Test (1% - Noisy).....	130
Figure 30 - Multiple Indicator Test (2% - Noisy).....	130
Figure 31 - STS-134 LH2 Loading (SF to FF)	132
Figure 32 - STS-135 LH2 Loading (SF to FF)	132
Figure 33 - STS-135 Flow and K-Distance	134
Figure 34 - Pressure and K-Distance	136
Figure 35 - Pressure and K-Distance with Failure.....	137
Figure 36 - STS-135 LH2 Loading (SF to FF) with STS-134 Temp.....	160
Figure 37 - Proposed ADFI Selection Framework	164
Figure 38 - STS-134 LOX Pump - Replenish.....	167
Figure 39 - STS-135 LOX Pump – Replenish (k-distance).....	168

LIST OF TABLES

Table 1 - Expert System Techniques for Fault Detection/Diagnosis (Angeli, 2010)	24
Table 2 - Gap Analysis Summary	33
Table 3 - Valve Fault Scenarios	44
Table 4 - Failures vs. Instrument Matrix	56
Table 5 – Requirement Summary	100
Table 6 - Requirement/AD-Class Matrix.....	107
Table 7 - Requirement/IF-Class Matrix	111
Table 8 - Training Data.....	122
Table 9 - Elements used for K-Distance	135
Table 10 - Fault Map (PT057-High).....	140
Table 11 - Fault Map (PT076 Check).....	141

CHAPTER ONE: INTRODUCTION

Space operation missions are of a critical nature. This is due to the large expense associated with such missions, many of which have interrelated costs approaching, or exceeding, billions of dollars. In addition, space exploration missions often have small windows with limited chances to recover from issues that may occur, and complete the mission successfully. Once committed to a given phase during these missions, opportunities for do-overs are rare. Adding to this criticality is that space operation systems require highly hazardous commodities to propel, power and operate the various systems. This adds a safety factor both for those participating in crewed missions as well as those involved in ground processing of the launch vehicle and spacecraft.

These spacecraft are comprised of numerous complex systems. This includes the equipment that makeup the spacecraft used for delivery, and the various payloads and science instruments used to meet the research objectives. Accompanying this complexity will be the assorted complications. As individual components do not always operate as projected, some failures are to be expected. This is further compounded as the system's complexity increases. With payload costs just recently starting to approach \$1000 per pound (a tenth of that experienced by the space shuttle) (SpaceX, 2011), designers must still strike a balance between system redundancy, sensor allocation and hardware weight. An absence of redundancy minimizes the options an operator has should a failure occur. A lack of instrumentation limits visibility into system performance.

For many systems, timely recognition of an anomalous situation means the issue gets evaluated and a course of action is decided on quickly. Taking corrective action sooner can help minimize

damage generated from off-nominal conditions, or avoid serious outcomes for those problems that can escalate rapidly. The ability to quickly detect and identify anomalies that arise is vital for critical space operations.

Anomaly Detection

One of the earlier definitions of ‘anomaly’ is that it is an observation that appears to deviate markedly from other members of the sample in which it occurs (Grubbs, 1969). More recent definitions conclude that anomalies (or outliers) are patterns in the data that do not conform to a well-defined notion of normal behavior (Chandola, Banerjee, & Kumar, 2009). Anomaly detection is a recurring term found within the academic literature and has been applied to a variety of fields. One of these fields, and the one applicable to this topic, includes health management of complex systems. For this research project, anomaly detection will be referred to as the ability to uncover abnormalities that arise. This term is more comprehensive than ‘fault detection’ (also common in the literature) as it also includes anomalies generated by environmental influences or operational circumstances.

In many cases, space operation systems are predisposed to take advantage of anomaly detection applications. This not only applies to actual space missions that rely on autonomous designs, but also includes supporting ground systems. Due to the hazardous nature of testing and launch support environments, many of the ground systems are operated remotely. As processes operating via remote command and control (C&C) rely on (limited) sensor information, anomaly

detection capabilities can supplement the flow of useful information to the operators based on various techniques used to evaluate the sensor data.

Anomaly detection routinely involves one or more people to monitor the system measurements. The primary method of anomaly detection is to bind the sensor data to predetermined limit exceptions. If an excursion from these limits occurs, an operator is alerted who then assesses the significance of the exception. This method is reliant on an operator with sufficient domain knowledge to take appropriate action, and subsequently, a large number of experts may be employed for complex systems. Operators too can monitor the data real-time or during post-operation analysis. However, with limited display space, it is impractical to have visibility for every remote sensor in a complex system. This is further hindered in that an operator can only focus on a smaller subset of displayed data. Therefore, the operator relies heavily on the automated monitoring of the entire sensor array. This reliance means an operator could benefit considerably from an enhanced anomaly detection capability.

The predominant method for developing an anomaly detection system is to select one or more applications that may fit one's needs. The selection process may be limited to what is readily available, and a scrub against the system requirements may end up diminished. After these modules are selected, development then starts on a system-level architecture that can integrate the various elements into the current design. As this is a niche application, a certain amount of tailoring will be required to render the forthcoming architecture functional. An application that

is not a good fit can amass excessive development hours in an effort to make the system operational.

Fault Isolation

Detecting an anomaly is the first half of providing the operator with the vital data necessary to develop a course of action. After an anomaly has been detected, the other half of the process (and equally important) is to isolate the fault within the system. Isolation to a component or subassembly provides the domain experts with the essential information necessary to respond and possibly remedy the situation. Fault isolation will optimally be able to point to a specific source. However, a lack of sensor information often leads to uncertainty which can result in an inadequate diagnosis. It is often the case that the initial data related to the anomaly is not sufficient to pinpoint the original problem, and involve additional troubleshooting to find the cause. This may result in an initial isolation of the problem to an upper subsystem level. In this limited-visibility scenario, it would not be unusual to have multiple suspect subassemblies and/or components identified.

In addition to issues related to limited sensor data, system complexity can make the fault isolation process arduous. The task of isolating to a fault entails assessing all possible contributors to the problem, each with varying degrees of sensor coverage. For a complex system, these contributors can number into the hundreds and possibly thousands. A fault that yields additional damage compounds the problem as one must differentiate if that damage is collateral to a single fault or originating from an entirely different fault source.

Problem Statement

There are many types of anomaly detection and fault isolation techniques described within the academic literature. Some of these applications are specific to a given system or to a type of problem, while others cover a wider spectrum of cases in general context. In addition, they have varying degrees of effectiveness. Complex systems may require several anomaly detectors and fault isolators to provide an adequate discovery capability. These diverse applications may target different areas of a system, or may focus on a specific concern and work independently to provide consensus that an anomaly is occurring or the fault source has been determined.

Although a multitude of anomaly detection and fault isolation programs can be found in the research literature, there does not appear to be any work published on architectural templates that could take advantage of multiple programs and integrate them into the desired systems. More specifically, there is an absence of a methodological process for generating anomaly detection and fault isolation designs to either embed within new system concepts, or supplement existing schemes.

Research Objectives

An architectural framework template is being considered that assists with anomaly detection and fault isolation module selection. Such a framework would consider the user requirements and then be able to model the proposed system. This will enhance the module selection process and thereby optimizing the detection/isolation suite. Such a model will assist the developers when it comes time to implement the system. The primary objectives of this research include:

- Develop an architectural framework template using system engineering principles that standardizes how users can model a system augmented with detection/isolation capabilities
- Based on architectural analysis, provide a methodology that can determine an optimal suite of detectors and isolators that best meet the user requirements.
- Generate a model that can integrate the detector components into the system and provide a basis to directly produce design implementation documentation
- Verify and validate the model by experimentation using actual space operation systems data

Research Contributions

The importance of anomaly detection and fault isolation is already valued by those operating complex and critical systems. This is consistent with the amount of work devoted to development of these methods and the abundance of techniques that currently exist. Many of these detectors and isolators are developed for specific applications with a very narrow field of focus.

The primary contribution of this research effort is realizing a conceptual model that assists users in generating anomaly detection and fault isolation schema. This research should extend the contributions of those development efforts by providing a means to organize the detectors/isolators for ingestion into the model, and subsequently, acceptance for use should the

capabilities meet the desired requirements (or rejection of the detector/isolator should they not).

The secondary contributions should then include:

- Couple anomaly detectors and/or fault isolators with unique applications for which they were never intended, but could benefit from the underlying detection/isolation techniques
- Improve accuracy in anomaly detection and fault isolation capabilities by pairing those deemed optimal for the given environment in which they will operate

Dissertation Organization

This chapter provides an introduction to the anomaly detection/fault isolation topic, and how this research effort will focus on developing an architectural framework for inclusion of these technologies for space operation systems.

Chapter 2 will survey the academic literature for relevant anomaly detection and fault isolation technologies. This review will also encompass system engineering techniques that may support development of an architectural framework. A gap analysis is then performed to determine where a need might exist to extend the prevailing level of research.

Chapter 3 will discuss the methodologies and procedures used to conduct the research effort. This includes an outline of the research design for the proposed framework, rationale of the methodologies used, type and source of the information needed and analysis of the data gathered.

Chapter 4 focuses on developing the proposed framework. This consists of examination of the detection/isolation techniques and a means to organize these applications by variables that support the architectural design. System engineering practices will provide the foundation for model development.

Chapter 5 will present a space-operations related case study that showcases an implementation of the proposed framework.

Chapter 6 centers on the analysis of the case study. This chapter will also validate the model being generated.

Chapter 7 will summarize the research results, provide concluding remarks, and offer recommendations for future work.

CHAPTER TWO: LITERATURE REVIEW

Purpose

To formulate a conceptual architecture addressing anomaly detection and fault isolation, a comprehensive literature review was conducted. The rationale for conducting the review was twofold. First, a thorough review aids in bounding the research problem and directing the path forward. This is accomplished by identifying the existing work, and from that, recognizing which areas within the field of study that can benefit from additional inquiry. These ‘gaps’ enable the narrowing of the designated field to either a new study domain or one that extends existing research, thus avoiding duplicate efforts that have already taken place. A gap analysis will further assist with differentiating those areas that could benefit from additional study.

The second reason is that a literature review expands insight into the chosen topic. A review is necessary to assess the related prevailing concepts. More specifically, the review includes discovering the various detection/isolation methodologies already developed and understanding the variables that make up the different technologies. Insight is also gained by identifying relationships among the applications and realizing different perspectives for implementing within diverse systems.

One objective of the review was to identify the current scope of anomaly detection and fault isolation applications. This includes existing technology that may already be in use, conceptual designs not yet implemented or paired with a system, or any related emerging technologies. The other review objective was to attain relevant system engineering methodologies that could be

used to build an architectural framework that forms a standard model to support future implementation. This chapter surveys the relevant academic literature related to these topics and provides a baseline from which advancement by new research can be appraised.

Anomaly Detectors

Numerous models are available in both model-based and data-driven classes. The algorithms involved tend to be ‘specialists’ in that they are most effective for selective failure modes and/or component types. Anomaly detection is typically accomplished by a rule or signal-based method. However, data-driven models have found a niche for possible better performance in complex, dynamical systems, an important factor for critical systems.

Data-Driven Models

One area of anomaly detection that is getting considerable attention involves a data-driven approach. This involves developing a knowledgebase of data depicting normal behavior which becomes a baseline set for comparison. Abnormal behavior is then described as incidents where the data behavior diverges from the baseline. Data-driven models tend to disregard the physics behind the data and instead focus on the differences behind the dataset standard and test case data. Hence, an advantage of data-driven models is that the developers do not require domain knowledge of the system under study nor do they need to model the system specifics. Such a design allows for distribution across multiple system platforms with little (if any) modification. In addition, the requisite system knowledge is captured in the training datasets. These datasets can also be expanded as nominal operational data is collected (D. Iverson et al., 2012).

Developing the model will require some system subject matter expert (SME) input to identify related subsystem sensor data. The SME can also characterize the sensors based on criticality. This information can be used to adjust sensitivity levels and establishing threshold values. This data-driven method appears to be a simple approach, but does have its challenges as stated below by (Chandola et al., 2009):

- Defining a normal region that encompasses every possible normal behavior is very difficult. In addition, the boundary between normal and anomalous behavior is often not precise. Thus an anomalous observation that lies close to the boundary can actually be normal, and vice versa.
- In many domains normal behavior keeps evolving and a current notion of normal behavior might not be sufficiently representative in the future.
- The exact notion of an anomaly is different for different application domains. For example, in the medical domain a small deviation from normal (e.g., fluctuations in body temperature) might be an anomaly, while similar deviation in the stock market domain (e.g., fluctuations in the value of a stock) might be considered as normal. Thus applying a technique developed in one domain to another, is not straightforward.
- Availability of labeled data for training/validation of models used by anomaly detection techniques is usually a major issue.
- Often the data contains noise that tends to be similar to the actual anomalies and hence is difficult to distinguish and remove. (p. 15:3)

For data driven models, the baseline dataset is often referred to as the ‘training’ data. The training data itself has different classifications based on what is known about this dataset. A *supervised* dataset is one that combines known anomalies with known normal data. Such a dataset is considered labeled accordingly (anomaly and normal). A *semi-supervised* dataset contains only normal data and an unsupervised data set does not have any labels (Omar, Ngadi, & Jebur, 2013). In many cases, obtaining labeled datasets is not at all practical for complex systems. For anomaly sets, this requires simulating the anomalies to a resolution that closely mimics real-world. Fabricating anomalies such that the issue is fully propagated throughout the system can be both a difficult and comprehensive task. The alternative to simulation is actually experiencing the anomaly numerous times. This (of course) is not the optimal approach to developing a training dataset and would only be practical if a hardware failure could be simulated without system collateral damage.

Models/Algorithms

There is an assortment of algorithms that have been developed and applied to many complex system applications. Due to the longevity of the Space Shuttle program, its unique need for anomaly detection capabilities, complexity, and NASA’s inherent goal to support scientific research, it has been the subject of numerous studies and testing related to algorithm development. The simplest type of anomaly is classified as a ‘point anomaly.’ This is “an instance of the data that has been found to be anomalous with respect to the rest of the data” (Gogoi, Bhattacharyya, Borah, & Kalita, 2011). As this includes the data found with sensor arrays under study, the anomaly detection methods will only address this type of anomaly. In a

majority of applications, this is the type of anomaly occurs most often, and a good amount of research addresses this issue. The following list summarizes the various point-type anomaly detection methodologies as described by (Chandola et al., 2009).

- Classification
 - Neural Networks
 - Bayesian Networks
 - Support Vector Machines
 - Rule-Based
- Nearest Neighbor
 - K^{th} Nearest Neighbor (K-NN)
 - Relative Density
- Clustering
- Statistical
 - Parametric Techniques
 - Gaussian Model
 - Regression Model
 - Mixture of Parametric Distributions
 - Non-Parametric Techniques
 - Histograms
 - Kernel Function
- Information Theoretic
- Spectral

All of these techniques will be given consideration for inclusion in the anomaly detection architecture. Using the classifications from the list above, the following sections will review some of the anomaly detection research that has already been applied to space operation systems.

Rule Based

The primary method currently used for anomaly detection is an exception notification methodology which could be considered a derivation of a rule-based practice. Although Chandola et al describe this method as requiring a rule-learning algorithm, due to both the criticality and reliance of this methodology, the rules are predesigned and embedded within the controlling application.

The rules are quite simple. Each analog parameter is given an upper and/or lower exception limit value that encompasses the nominal range (also called signal-based). The exception limit for a discrete variable is the opposite of its current state. If an exception to these limits occur, the operator is alerted. Exception limits can be generated to protect either the design or operational limits of the system. As the operational environment changes, limit settings can be widened or inhibited so as not to alert on nominal transient responses, and then reset to the newly desired limits for that phase of the operation. Note that transient operations often create 'blind-spots' while monitoring the system as anomaly detection works best with stable processes. For hazardous, time-critical or hardware-concern issues, exception limits are often used as trigger-

points to initiate additional rules (i.e. turning off a failed sensor or switching from a primary to a secondary system).

Nearest Neighbor

The nearest neighbor approach is based on an assumption that related data tends to group in dense neighborhoods. Anomalies are those outliers that are found some distance away from the closest neighbor (Chandola et al., 2009).

An anomaly detection method called Orca (Bay & Schwabacher, 2003) uses a nearest neighbor based algorithm to determine outliers. To minimize the computational time, it employs a pruning technique which allows it to perform in near linear time. Orca calculates a weighted average of the Euclidian distance for the numerical values and a Hanning distance for the discrete variables. The output from Orca is a distance score which represents the average distance to its k-nearest neighbors. The further away the nearest neighbors, the more anomalous the data correlating to a higher score. Orca has been used to detect anomalies in the Space Shuttle main engines (SSME) during both flight and engine test-stand runs (Abdul-Aziz, Woike, Oza, Matthews, & Iekki, 2011) (M. Schwabacher, Oza, & Matthews, 2009).

Clustering Algorithms

In a clustering-based approach to outlier detection, the “key assumption made is that large and dense clusters have normal data. The data which do not belong to any cluster or small clusters (low dense clusters) are considered outliers” (Murugavel & Punithavalli, 2011).

A data-driven application called Inductive Monitoring System (IMS) is a distance-based anomaly detection tool that uses a clustering technique. The data structure used for distance-based analysis is a vector of concurrent values from related system parameters. IMS reads real-time (or archived) data and formats it into a vector structure. It then searches the knowledgebase of nominal data (training data) and returns the distance between real-time and the nearest nominal data vectors (Matthews, Srivastava, Iverson, Beil, & Lane, 2011) (Martin, Schwabacher, & Matthews, 2010). When the real-time data is consistent with nominal, this difference is close to zero. If the data vectors start to diverge, an increase in the vector differences is noted and the real-time data is then deemed ‘out-of-family.’ This can be an indication of an anomaly that is occurring.

It should be noted that the real-time data is being compared to previous collected empirical data. Thus, an out-of-family indication can also reflect a normal condition that was not fully characterized within the nominal data sets used to ‘train’ the model. The IMS application works well with unsupervised data which is likely the only type of data available for most large complex systems. Unsupervised means there is an assumption of normalcy, but a potential exists that undetected anomalies are embedded within such data sets. In these cases, IMS could treat some anomaly precursors as nominal, requiring even greater vector disparity before getting flagged as anomalous.

IMS has been used for anomaly detection testing in the Space Shuttle (wings, main engines) and ground launch systems. In addition, it is currently used to monitor Space Station subsystems (Matthews et al., 2011). Reference Figure 1 - Anomaly Detection Process Flow Example for a process flow example using a data-driven distance-based anomaly detection model.

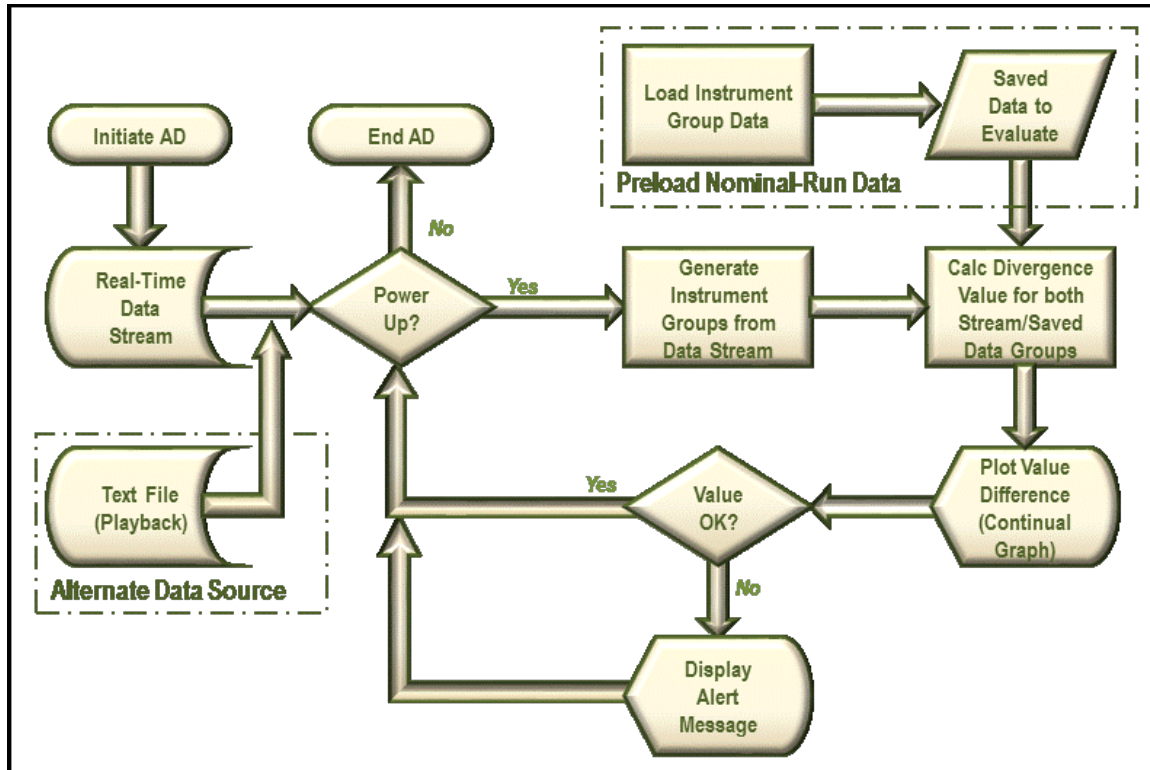


Figure 1 - Anomaly Detection Process Flow Example

Neural Network

A neural network is trained on a nominal reference data set to learn the different normal classes. Each test occurrence is then submitted as an input to the neural network. If accepted, the test

instance is deemed normal, and if rejected, anomalous (De Stefano, Sansone, & Vento, 2000) (Chandola et al., 2009).

NASA started the Methane Thruster Test-Bed Project (MTTP) as a platform for research of plume diagnostics and Integrated System Health Management ISHM. A method to validate the sensors was developed using an auto-associative neural network (AANN). Archived data was used to train and test the (AANN) for sensor validation. Sensor faults ranging from hard (loss of power or over powered which would drive the sensor off-scale low or high) to soft (indication drifts from actual) were artificially injected. The AANN was able to detect the faults from within the pressure sensor data as well as predict the values of the pressure measurement to a reasonable degree (Russell, Lecakes, Mandayam, & Jensen, 2011).

Statistical/Parametric

A regression analysis for anomaly detection requires that the individual data be fitted to the regression model. The focus is then on the residuals as these represent data that the regression model could not explain. The anomaly score is an accumulation of divergence values of the residuals from the model.

A Beacon-based Exception Analysis for Multi-missions (BEAM) tool was developed by the Jet Propulsion Laboratory (JPL) to monitor autonomous space systems. This application was then modified to support the monitoring of the Space Shuttle main engines during both flight (real-time) and post-flight analysis (or post-test for ground testing). The anomaly detection module

for this application is called Dynamical Invariant Anomaly Detector (DIAD). The DIAD element performs a parametric estimate of the residuals based on a single quantitative measurement. It is believed that the ‘dynamical invariants’ are less sensitive to operational influences and impacted more by internal changes to the system dynamics (Park et al., 2002).

A method of generating an adaptive anomaly detection threshold using interval models has been proposed by (Puig, Quevedo, Escobet, Nejari, & de las Heras, 2008). This concept was adapted to monitor a propellant ground controlled linear-actuated valve used for rocket engine testing at the Stennis Space Center (SSC). Nominal data was obtained from both the performance and simulated operations of the valve under study. A number of autoregressive moving average (ARMA) models are generated so that the valve’s behavior is satisfactorily represented based on the control data (this can be a trial-and-error process) (Russell et al., 2011). The valve’s control pressure was adjusted such that the valve could not close completely, thus simulating an obstruction which is subsequently detected as a fault by the model. It should be noted that applying this method of simulation, manipulating control parameters to achieve a desired result, negates the use of those control outputs for the nominal data set (often, these outputs are monitored to determine valve performance).

One-Class Support Vector Machines

Support vector machines (SVM) map the input vector into a higher-dimensional feature space and then separates the nominal data from anomalous in that feature space (one-class refers to the possibility that only normal data is available). A separating hyper-plane is determined by

support vectors (a subset of the training data) rather than the whole training samples and thus is extremely robust to outliers. The training and test cases are represented using a kernel function that returns the distances between pairs of examples. The anomaly score reported is the distance from the test data point to the hyper-plane as measured in feature space. One-class SVMs have been used to detect anomalies in the SSMEs during both flight and engine test-stand runs (Abdul-Aziz et al., 2011) (Omar et al., 2013) (M. Schwabacher et al., 2009).

Fault Isolators

With an overabundant number of potential fault sources for a given anomaly, it would be ideal to have a model that can automate the fault isolation process. This provides the capability to ascertain each of the possible failure scenarios, and utilize the entire sensor array to evaluate each case. In those instances when multiple fault sources or subsystems are identified, the model can rank the potential candidates and present them in order of those deemed ‘most-likely.’ For time and safety critical circumstances, the initial system-safe actions can be automated to trigger based on the type of fault identified. To accomplish this, the fault isolation algorithm must recognize the failure type, locate the failure position and detecting the extent of the failure (Wu, 2005).

Data-driven detection models are indifferent to the physics behind the sensor data as they devote their attention to abnormalities found within the data. However, fault isolation models require system knowledge to accurately pinpoint the source of the fault. There are cases where data observation alone will be able to identify the faulty component. For instance, an analog sensor

with a nominal indication (approximately midscale) goes off-scale low or high in a single sample step (an electronically high sample rate is assumed). Typically, such a rate of change would be a physical impossibility for that system. Therefore, a model could accurately conclude the sensor itself has failed. Conversely, if the sensor indication just starts drifting away from nominal, the challenge is then determining if the sensor is reporting system dynamics accurately, or if in fact, the sensor is failed. Note that a sensor is also just one component in a command and control system that leads back to an operator. This requires additional corroborating data combined with system knowledge (for both Process and C&C systems). This highlights that the fault source may occur at any point from the C&C work station to the remote system being operated (reference Figure 2 - Potential System Fault Sources). A supervised data-driven model has the capability to accomplish this task, but this requires a bank of anomaly classified datasets (a method of archiving system knowledge). As stated earlier in this paper, deriving anomaly classification datasets is likely an impractical option for complex systems.

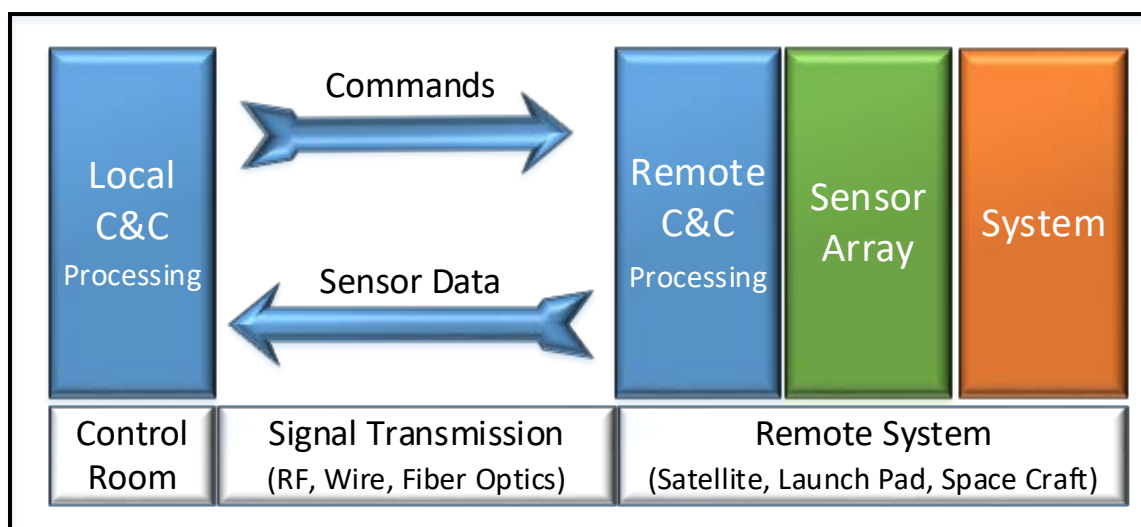


Figure 2 - Potential System Fault Sources

Fault Isolation Models

The following is a review of fault isolation research that is being applied to space operation systems. The emphasis is on work that supports large-scale complex systems (vs. isolation at the component level or smaller subsystems). All of these techniques will be given consideration for inclusion in the fault isolation portion of the proposed architecture.

Physics Model

Physics-based modeling that accurately represents the system can be adapted to perform fault isolation duties. A physics model captures the system knowledge within mathematical formulas that define the system. Therefore, such a model will ‘understand’ the system dynamics to include areas not covered by instrumentation, an advantage that overcomes limited sensor deployment. A physics model can be used to simulate a given system, and failures can be injected and subsequent outcomes recorded. The expectation is the model will fully propagate the issue throughout the system. This methodology can be used to develop anomaly cases that could support both detection and isolation. For unknown problems, one can alter the parameters of a high-resolution model to match suspect failures until an outcome comparable to the issue experienced is obtained. Physics models are complex and may not be deemed practical for a fault isolation application alone. However, these models have become the norm for assisting with the design of new complex systems. This means much of the computational effort may already be accomplished and available for modification and integration into a fault isolation environment.

NASA is developing a physics based model to simulate the launch pad's liquid hydrogen propellant ground system. They modify the nominal-run model by simulating faults. The sensor data is collected and archived for use in fault diagnosis applications (Osipov et al., 2011). A modeled-based diagnostic approach to the system is accomplished using a combined qualitative-quantitative methodology approach per (Mosterman & Biswas, 1999). As the measured values diverged from predicted values, these are compared to qualitative predictions made using the system model for fault isolation. Fault identification is performed using particle filters for joint state-parameter estimation (Daigle, Foygel, & Smelyanskiy, 2011).

Expert Systems

As expert systems are intended to mimic human-reasoning (the predominant method employed to identify fault sources), they have been widely used for fault isolation applications. Expert systems are developed using rules based on empirical associations. Fault diagnosis is a hierarchical process carried out in a step-by-step manner with the next step dependent on the results from the previous one (Kodavade, 2012). An expert will reason via a set of rules that leads to a logical chain of events. A fault is detected if a violation of these rules occurs (Marzat, Piet-Lahanier, Damongeot, & Walter, 2012). Table 1 - Expert System Techniques for Fault Detection/Diagnosis (Angeli, 2010) provides a summary of the pros/cons to the different types of expert systems.

Table 1 - Expert System Techniques for Fault Detection/Diagnosis (Angeli, 2010)

ADVANTAGES	DISADVANTAGES
Rule based diagnostic expert systems	
Rules can be added or removed easily	Lack of generality
Explanation of the reasoning process	Poor handling of novel situations
Induction and Deduction process is easy	Inability to represent time-varying and spatially varying phenomena
A process model is not required	Inability to learn from their errors
Efficiency and effectiveness in fault detection	Difficulties in acquiring knowledge from experts reliably
	Development and maintenance is costly
Model based diagnostic expert systems	
Device independent diagnosis	Domain dependent
Knowledge acquisition is not needed	Difficulties in isolation of faults
Ability of diagnosing incipient faults	Knowledgebases very demanding
Deal with unexpected cases	
Flexibility in the cases of design changes	
Dynamic fault detection	
On-line diagnostic expert systems	
Real time fault diagnosis	Domain dependent
Ability to handle noise	Good models are required
Generalization	Require considerable data
Fast computation	Inability to explain the reasoning process
Ability to handle with dynamics	Computationally expensive

NASA has developed a rule-based expert system called Spacecraft Health Inference Engine (SHINE) to perform system health diagnostic functions. SHINE uses heuristics to quickly isolate possible fault causes and causal-reasoning to analyze the fault and further refine possible causes (Straub, 2011). This system has been used for ground testing of the ARES 1X rocket (M. A. Schwabacher et al., 2010a) and the Tactical Satellite-3 (TacSat-3) spacecraft (Mackey, Brownston, Castle, & Sweet, 2010).

Functional Fault Model

A functional fault model (FFM) is a term being applied to an application that maps out the system in a way that links the inputs/outputs down to specific components. A commercial product being used in several space operation systems is called TEAMS (“Qualtech Systems » TEAMS-Designer,” n.d.). An FFM will identify the Failure Effect Propagation Paths (FEPP) from a failure mode back to the sensor that detected the anomaly. It then uses the archived maps to identify the potential failure sources or modes that are consistent with the system response to the anomaly (Ferrell, Lewis, Perotti, Oostdyk, & Brown, 2010).

Anomaly Detection and Fault Isolation

This paper has reviewed the topics of anomaly detection and fault isolation separately as approaches to developing the corresponding models differ substantially. This is further necessitated as the forthcoming architectural framework will have to treat the models independently. It should be noted that the academic literature often combines these two areas of study into a single topic. There is compelling rationale to take this approach as both detection and isolation must occur before corrective action(s) take place. This complexity of complex models is the driving force behind the need for an architectural framework that can integrate many ‘modules’ that will comprise fault detection and isolation schemes.

System Engineering Tools

This research effort is focused on building a framework that enables the current anomaly detection and fault isolation technologies. Such a framework must be capable of integrating a

multitude of potential models to meet user requirements (many still in development). In addition, it shall be readily adaptable so that it can be ‘custom-fitted’ to meet specific mission requirements for the various operations it is envisioned to support. With an assortment of models and algorithms available (each with its own unique specialty) and the numerous requirements anticipated, a systems engineering approach is deemed the best method to manage the complex architecture development. After an initial survey of the available tools, System Modeling Language (SysML) is the application selected to support this research effort (“OMGSysML-v1.3-12-06-02.pdf,” n.d.). A complete specification of SysML can be found in (Friedenthal, Moore, & Steiner, 2012a).

Model-based systems engineering supports analysis, specification, design, and verification of the system by organizing activities through formalized representations of the system referred to as models. This methodology enhances the quality of the design process, supports reuse of the various output components and augments the identification of system impacts should subsequent design changes be considered (Friedenthal, Moore, & Steiner, 2012b) (Cressent, David, Idasiak, & Kratz, 2010).

SysML is a derivative of the Unified Modeling Language (UML). UML has become a very popular tool used to develop large-scale, complex software applications across multiple platforms. As UML is software-centric, SysML was developed to apply the successful UML techniques to the system engineering field in all areas (not just software engineering). To support an application base that includes both hardware (mechanical, fluids, electrical) and

software applications, SysML reuses and extends a subset of UML 2.1 constructs (Johnson, Kerzhner, Paredis, & Burkhart, 2012, para. 1.1):

- Extends UML classes into blocks
- Enables requirements modeling
- Supports parametric modeling
- Extends UML dependencies into allocations
- Reuses and modifies UML activities
- Extends UML standard ports into flow ports

Utilizing the SysML language, models can be produced that are capable of describing the system in detail. Disciplinary engineers use analytical tools to accomplish design and analysis tasks. If there are times when a study (cost, risk, tradeoff, etc.) requires both system and analytical information, this must be manually obtained from each application. There are system engineering tools that bridge this gap and integrate the corresponding information and subsequent updates (Kim, Fried, Menegay, Soremekun, & Oster, 2013). This points out that a single system engineering tool may not be sufficient to achieve research goals, and SysML may have to be augmented with a compliment of supporting tools.

The European Space Agency (ESA) has a facility called the Concurrent Design Facility (CDF) that is a state-of-the-art program in the field of concurrent engineering and system engineering research. The CDF is used to perform feasibility studies for potential future space missions. They currently build system engineering models using Excel, and decided to test a model-based

system engineering approach using SysML. The MBSE model was considered applicable to the concurrent engineering approach. They selected a case study on a project called Near Earth Exploration Minimum System. The results of their testing are mixed with complaints about the significant amount of time to build the model with too little added value. However, their final conclusion was that SysML modeling should be paired to work in conjunction with their current integrated design model as they see potential in this tool as the technology matures (de Lange, Guo, & de Koning, 2012).

There is a French program developing a ramjet powered vehicle capable of reaching speeds between Mach 4 and 8 (called LEA). A Failure Modes and Effects Analysis (FMEA) was performed on the components that make up the vehicle. They input the FMEA results into SysML identifying all the blocks and parts and establishing the hierarchy between these items. Then they mapped each component using ports and connectors. With several system architectures to choose from, the resulting model allowed the final decision to include the failure mode of the system (Cressent et al., 2010). In fault isolation modeling, a FMEA is routinely the first document assessed as much of the work in identifying the failure modes and components involved is complete. SysML's diagramming tools allow for suitable characterization of these failures and this technique could be adapted to developing a fault isolation model.

Recognizing the trend in model-based system engineering (MBSE), NASA's Langley Research Center initiated a project to test this technique. They implemented a pilot program to evaluate MBSE methodology and centered it on the early phase of the Materials International Space

Station Experiment-X (MISSE-X). MISSE-X is designed to be installed on the exterior of the international space station in which experiments reside that “advance the technology readiness of materials and devices necessary for future space exploration.” The goal was to develop a SysML model that could capture requirements, behavior, architecture and operating environment of the experiment. The results of the pilot program showed that the investment of effort in MBSE is substantial, but one that produced noteworthy returns (Vipavetz, Murphy, & Infeld, 2012).

Gap Analysis

A gap analysis on the reviewed literature is essential to determine if a research gap exists, thus identifying an area for which additional study is appropriate. The review was structured such that the literature cited would best support the research topic. However, since the goal is to find a research area that may benefit from additional study, a lack of conclusive references specific to the topic should be expected. To determine if the documents reviewed are supportive to this research effort, certain questions are asked to include:

- Anomaly Detection/Fault Isolation
 - What is the scope of existing anomaly detection and fault isolation applications?
 - Are they specific to an application or more general and used universally?
 - Can anomaly detection techniques be used for fault isolation (and vice-versa)?
 - Are multiple anomaly detection/fault isolation models presented?

- Are the models specific to anomaly detection or fault isolation, or are they cross functional?
 - Are these models integrated so that they work collectively?
 - Is there an architecture defined for the model (or multiple models)?
- System Engineering Tools
 - Which methods/tools are used for framework development?
 - Are there existing applications supporting space operations?
 - Are there existing applications focused on anomaly detection and/or fault isolation?
 - Is there a conceptual detector and/or isolator framework already in-place?

The literature review concentrated on three areas to include anomaly detection, fault isolation and system engineering tools from which to build a model. During the review, it was found that the majority of the anomaly detectors and fault isolators were “specialists.” These techniques often had narrow design functions targeting specific technologies. Even the data driven applications, those that are effective without insight into the physics behind the system, have to be fine-tuned to recognize system operational nuances. As this research effort is intended to be applicable to complex systems, the use of multiple anomaly detectors and/or fault isolators is anticipated to be the norm. Therefore, it is important that any resulting framework must be able to integrate multiple and diverse applications.

The gap analysis commenced by identifying the characteristics that support development of a standardized framework for designing anomaly detection/fault isolation systems. The research

literature was then reviewed and documents that met the selected characteristics were identified. These characteristics were divided into three separate categories. The first category simply identified the documents as being related to either anomaly detection, fault isolation or research that supports framework development, thus matching the three focus areas mentioned above. Referenced literature that did not meet one of these characteristics was excluded from the gap analysis.

It was not unusual to find research that included both anomaly detection and fault isolation as these topics are often combined to meet industrial needs. However, it should be noted that literature involving anomaly detection or fault isolation did not include framework development methods for selecting these types of applications. Nor did any of the framework development literature reviewed involve applications specific to anomaly detection or fault isolation content.

The next category centered on the anomaly detection and fault isolation literature. These characteristics first included the class of technology that these detectors/isolators fit as outlined earlier in this literature review (reference table 2). This classification allowed for identifying common techniques between the anomaly detection and fault isolation applications. Another characteristic within this group then keyed on whether these works pertained to multiple models, and if so, did the research integrate these models together. This is considered important as the eventual detection/isolation system developed will likely be comprised of multiple models. The last characteristic in this category highlighted any of the works that included an architecture depicting the models.

The third category addresses the system engineering practices for generating a framework that standardizes anomaly detection and fault isolation system development. The first characteristic within this category refers to whether the literature includes SysML and/or MBSE techniques for system development. Next it identifies those works where SysML/MBSE has been applied to anomaly detection or fault isolation applications. Finally, it determines if architectural development is already occurred these areas. Reference Table 2 - Gap Analysis Summary for a summary of the gap analysis results.

Table 2 - Gap Analysis Summary

	Anomaly Detection	Fault Isolation	Framework Development	Classification	Nearest Neighbor	Clustering	Statistical	Physics Based	Expert Systems	Functional Fault Mapping	Multiple Models	Models Integrated	Architecture Defined	SysML/MBSE	AD and/or FI Application(s)	AD/FI Architecture Defined
Researchers	Category		Anomaly Detection/Fault Isolation													Sys Eng
Abdul-Aziz, et al., 2011	X				X	X	X					X				
Angeli, 2010		X										X				
Bay & Schwabacher, 2003	X				X											
Chandola, et al., 2009	X			X	X	X	X					X				
Cressant, et al., 2010			X												X	
Daigle, et al., 2011	X	X						X								
De Lange, et al., 2012			X												X	
De Stefano, et al., 2000	X			X												
Ferrell, et al., 2010		X									X					
Gogoi, et al., 2011	X				X	X	X					X				
Friedenthal, et al., 2012			X												X	
Iverson, et al., 2012	X					X										
Johnson, et al., 2012			X												X	
Kim, et al., 2013			X												X	
Kodavade & Apte, 2012		X								X				X		
Mackey, et al., 2010		X				X				X	X	X				
Martin, et al., 2010	X					X					X	X				
Mathews, et al., 2011	X					X										
Marzat, et al., 2011		X						X	X			X				
Murugavel & Prunithavalli, 2011		X				X						X				
Omar, et al., 2013	X			X	X		X					X				
Osipov, et al., 2011		X						X								
Park, et al., 2001	X						X					X	X			
Puig, et al., 2008	X						X					X				
Russell, et al., 2011	X	X		X			X	X				X	X			
Schwabacher, et al., 2010	X	X		X		X					X	X	X	X		
Schwabacher, et al., 2009	X			X	X	X						X				
Vipavetz, et al., 2012			X												X	
Wu, 2005	X	X		X						X		X				
Clark, 2015	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Gap Analysis Observations

The various characteristics have been identified from the literature reviewed and this information has been consolidated in table 3. Inspection of this table shows that ‘gaps’ do appear to exist in relation to the research topic. The following observations summarize areas in which conclusive research is absent:

- The literature involving anomaly detection/fault isolation (AD/FI) did not include an architecture as to how these applications should be selected and used. Those works that had the ‘architecture defined’ feature selected (Kodavade & Apte, Schwabacher, et al.) only had an upper-level depiction of the architecture specific to the model(s) being presented.
- The SysML/MBSE references did present several instances of applications related to space operations processing. However, none presented methods for developing a framework specific to AD/FI applications.
- There was little research encountered related to integrating multiple models. Much of the AD/FI literature was specific to a single application. Several surveys described multiple models (Abdul-Aziz, et al., Chandola, et al., Gogoi, et al. and Omar, et al.) but these works did not attempt to integrate the models exhibited. Park provides an overview of an integrated anomaly detection scheme called ‘BEAM,’ but the emphasis of the article is on a single module within this system (Park et al., 2002). Russell and Schwabacher both present integrated AD/FI systems (with mixed results), but each uses an unique framework (Russell et al., 2011) (M. A. Schwabacher et al., 2010b). None of the literature reviewed provides the methodology for AD/FI application selection.

Literature Review Summary

A survey of the literature was performed on the topics of anomaly detection and fault isolation, as well as system engineering tools that could be used to develop a detection/isolation framework. Numerous models are available in both model-based and data-driven classes. The algorithms involved tend to be ‘specialists’ in that they are most effective for selective failure modes and/or component types. Therefore, it is anticipated that a detector/isolator system will be comprised of multiple applications so that it is effective on the complex system for which it is being designed

The primary method of anomaly detection is an exception-based method. This method notifies an operator if design or operational limits are exceeded. Data-driven models have found a role in complex, dynamical systems, and function by detecting outliers in the data which have not yet exceeded predetermined limits. Such models can disregard the physics behind the system allowing for distribution across multiple systems, though detection accuracy is dependent on the quality of training data and effectiveness of the scoring-algorithm. Fault isolation techniques tend to be model-based as system knowledge is required to isolate the fault to its source. Isolation is a difficult task as systems often lack the requisite sensor data, hence lacking the necessary insight for accurate identification. This difficulty is further compounded by large numbers of potential sources to evaluate within complex systems.

An architectural framework that combines these methods is desirable. A model based system engineering tool, SysML, shall be used to evaluate the premise that such a framework is possible.

A descriptive model that can assist with analysis, specification, design and verification of this concept is the desired outcome.

A gap analysis was performed on the literature reviewed. The analysis shows that a gap exists in the methodology for selecting anomaly detection and fault isolation applications. In addition, the review was unable to uncover a systematic approach for a selection process using model-based system engineering techniques. This dissertation will pursue this line of research.

CHAPTER THREE: RESEARCH METHODOLOGY

This chapter depicts the methodology used in this dissertation. It provides a road map towards developing a framework that can standardize the selection of anomaly detection and fault isolation applications that can best be integrated into a desired system. This design addresses the research gap identified and provides a process by which the research objectives can be realized.

Methodology

This research topic originated from an observation that anomaly detection and fault isolation applications were selected based more on availability than on ability to meet system needs. Initial research was unable to uncover a practice that could assist a user with this selection process. In addition, this preliminary research determined that a significant amount of research has been generated related to anomaly detection and fault isolation techniques. Much of this research has not been applied in commercial applications. This led to the Problem Statement described in Chapter 1.

A framework that standardizes this selection process using system engineering principles is the goal of this research. Such a model must be able to pair numerous and unique detection/isolation techniques to a variety of applications in a way that maximizes efficiency of the integrated system. Figure 3 - Research Methodology Diagram illustrates how this study will go forward to meet this objective.

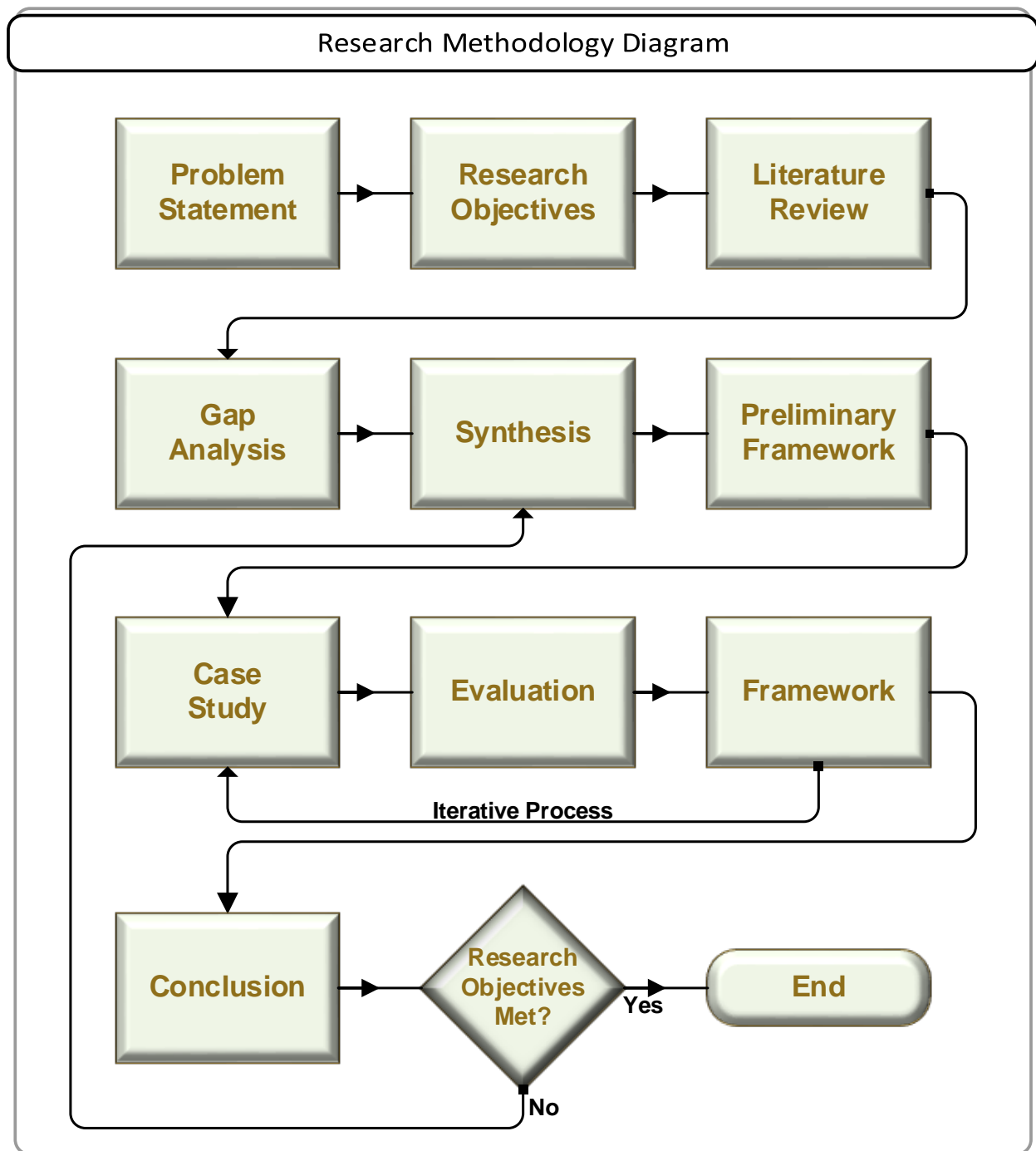


Figure 3 - Research Methodology Diagram

Problem Statement

The research process starts by first identifying a ‘problem,’ or an area that could benefit from additional academic-level research. In this case, the focus is on anomaly detection and fault isolation applications that could be used in space operation systems. Little research could be found on existing architectural templates that could integrate these applications into the designated systems. More specifically, there is an absence of a methodological process for generating anomaly detection and fault isolation designs to either embed within new system concepts, or supplement existing schemes.

Research Objectives

The next step is to generate objectives that will work towards resolving the problem area identified. Achieving these objectives is the goal of this study (and meeting this goal signals that the dissertation research effort is complete). The objectives for this research include:

- Develop a framework that standardizes how users can augment a system with detection/isolation capabilities
 - Framework to use system engineering principles
 - System can be either existing or a new design
 - Framework to provide a means to rank or optimize detectors and isolators under consideration
- Validate the model by experimentation using actual space operation systems data

Literature Review

A comprehensive literature review is performed to survey relevant works that may exist addressing the problem identified. A thorough review aids in bounding the research problem and directing the path forward. A literature review expands insight into the chosen topic, and allows for assessing related prevailing concepts. More specifically, the review includes discovering the various detection/isolation methodologies already developed and understanding the variables that make up the different technologies. These applications have been categorized into class-objects as this organization will assist with model development. Insight is also gained by identifying relationships among the system applications for which these detectors/isolators are designed to support.

Gap Analysis

After a literature review is complete and the effort summarized, a gap analysis is performed to determine where existing research efforts are lacking. Research gaps in the designated field are an indication that those areas could benefit from additional study. In this case, conclusive references specific to system engineering techniques that support development of a detector/isolator framework were not discovered, thus signifying that this topic is deserving of further pursuit.

Synthesis

Identifying all the pertinent data is the first step in generating a framework. When an anomaly occurs, it is expected to ‘disrupt’ the sensor array thus signaling an operator that the system is

diverging from nominal. A domain expert (or an algorithm) must then ‘interpret’ the deviations observed from the instrumentation, and using a logical process of elimination, isolate the problem to a specific subsystem or component fault. At this point, remedial action (if required) can be considered. Therefore, an important relationship exists between the type of anomalies that can occur and the availability/arrangement of sensors used to monitor the system. These are the primary dataset sources that will support this research.

Failure Identification

As the goal of this research is the enhancement of anomaly detection and fault isolation capabilities of complex systems, the potential failures that can occur must be quantified to encompass all that can operationally impair the system. If a ‘Failure Modes and Effects Analysis’ (FMEA) has been accomplished on the system, then potential faults may have already been identified. Fault-tree analysis is a technique that can be used to scope the potential failures for a component or system. Each of the fault-tree’s basic or intermediate events denotes a failure that can impact the functionality of the top-level item indicated.

Not all failures will impact system performance. For example, a cabinet that houses system instrumentation inside a conditioned room with a broken latch may be considered a benign failure. The same broken-latch cabinet mounted outside may have more ominous consequences while operating during inclement weather. Therefore, failure criticality must be taken into consideration when determining those problems that require inclusion. In addition, some potential problems may have an extremely low probability of occurring. A pipe support tends to

be a static structure designed to carry more than its prescribed load. This precludes having to instrument each and every pipe support even though a support failure could result in damage to a crucial pipe run. A risk analysis (criticality vs. probability) can be used to maintain the list of potential failures at a manageable level and remain focused on credible issues that threaten system performance.

As the path forward in developing this framework is guided by a systems engineering approach, defining requirements will be an essential element to this process. The inventory of failures generated by this analysis will lead to a corresponding requirement that states the failure mode shall be detected.

System Monitoring

The principal limitation in the ability to fully detect/isolate anomalies can be directly correlated to the system instrumentation. Instrumentation must be embedded within the remote hardware it is evaluating, and communicate via a C&C subsystem to provide operator feedback. This is costly and makes it impractical to include a sensor for every possible failure mode. These indicators too are susceptible to failure which results in some system degradation (for non-redundant sensors) as a reduction in visibility occurs. The first question routinely asked when a measurement alarms is, “Did the sensor fail?” In addition, unwanted actions may occur as automated processes may be invoked that are linked to (failed) instrument data.

For remote systems, the operator's 'view' is restricted to what the sensor array provides. The instrumentation encompassed within a design will have a specific purpose for its inclusion. Typically, it will meet operational requirements for monitoring the system functionality in general, as well as supporting various tasks. These operational requirements will bound the measurement to a tolerance range, and induce an alarm should the tolerance be exceeded. This is where the various anomaly detection techniques come into play. They are not limited to monitoring a specific measurement for a specific operational band. Instead, they look at the system or subsystems as a whole and extrapolate information from multiple sensors. This method uses both direct and indirect measurements to infer system health.

Using, for example, a valve that fails to indicate closed after being commanded to that state. The fact that the close switch never went on was a direct indication of that valve's state. However, this one indicator should not be taken at face value, but treated only as an alarm that something is amiss. A fault tree may show numerous faults that can lead to a valve malfunctioning. In addition to the closed indicator remaining OFF, Table 3 - Valve Fault Scenarios, describes sensor feedback that will assist in determining the valve's current position. For all three possible states, multiple sensors must be evaluated to corroborate that position. This process is anomaly detection. It is not limited to simply fielding an alarm, but using this alarm combined with other indications (both anomalous and nominal) to determine the system's current state.

Table 3 - Valve Fault Scenarios

Valve Position	Sensor Feedback
Open	Close Limit Switch Remained OFF (anomalous)
	Open Limit Switch Remained ON (anomalous)
	No changes to immediate upstream/downstream pressures or temperatures (anomalous)
Partially Open/Closed	Close Limit Switch Remained OFF (anomalous)
	Open limit switch goes OFF (valve moved) (nominal)
	Downstream pressure drops some, but not fully (anomalous)
Closed	Close Limit Switch Remained OFF (anomalous)
	Open limit switch goes OFF (valve moved) (nominal)
	Downstream pressure drops completely (nominal)

Fault isolation is the process of using the sensor array to pinpoint the source of the fault to a specific component (or base-event on a fault tree). This too is important as it assists in determining the extent of the anomaly's impact. If the example valve did in fact close, this would imply the close limit switch failed per the fault scenarios above (reference Table 3 - Valve Fault Scenarios). However, if the failure was due to a failed discrete processing card in the C&C subsystem, then it may have other implications as these cards typically contain multiple indicators. This requires that all measurements that can provide any insight into an anomaly be identified for that anomalous condition.

Datasets

Several datasets have been identified to support this research project. First, all potential anomalies must be identified. This will be accomplished using fault-tree analysis at the component level. A risk analysis will limit the collection of faults by ruling out those deemed

non-credible based on probability and criticality of the failure. Next, all system sensors must be described. These will be cataloged to the type of failure they can detect and subsystem they support.

A matrix can be generated that combines these datasets and relates this information at a component level. For each fault, any indicators that can provide awareness to that component and supporting subsystems will be listed. In addition, when multiple measurements are required to make a judgment, the matrix must be able to distinguish those sensors that must collaborate with others to make a failure determination.

The Space Shuttle program's Problem Reporting and Compliance Application (PRACA) repository contains all non-conformances reported for both the LH2 and LO2 systems. This will be a valuable source of actual issues that can support both model testing and validation. Synthetic problems may also need to be generated to account for credible problems not encountered during critical Shuttle operations.

Preliminary Framework Development

This section focuses on developing a preliminary framework. This framework begins with the data synthesis just described. In keeping with a systems engineering approach in this research effort, a Model-Based System Engineering (MBSE) application will be employed for framework development. This application will meet SysML language standards. SysML is derived from UML in that it has been extended to support both hardware and software development. An

MBSE model has several advantages that support this effort. First, the ability to make changes and analyze the subsequent impact will be beneficial when it comes time to fit the various detectors/isolators onto the designated system. Next, the capability to reuse objects created can reduce the overall effort, a process that can be quite tedious for a complex system. A SysML modeling tool enforces the language rules and also provides means for tracking requirements and validating the model which are important features for this project.

With a modeling approach selected, initial framework development involves examination of the detection/isolation techniques and a means to organize these applications by variables that support the framework design. These applications will be aligned into classes (and sub-classes) consistent with the groupings outlined within the literature review. Each anomaly detection and fault isolation class will be labeled by both their capabilities and interface. The capabilities (or behaviors) will be used to determine which requirements they can satisfy, and the interface will identify the inputs/outputs for that application. This preliminary framework will result in detection/isolation ‘modules’ that are ready for system inclusion in a model-based environment.

Case Study

A space-operations related case study will be presented that showcases the implementation of the proposed framework. In this case, a system to augment with a detector/isolator application is necessary. To meet this need, the cryogenic liquid hydrogen (LH2) and liquid oxygen (LO2) systems at the Kennedy Space Center (KSC) have been selected. These systems, located at the launch pad, were used to fill the Space Shuttle’s external tank with propellant and oxidizer for

the Shuttle's three main engines (and are slated for use again with the next NASA Space Launch System (SLS) program). Due to the hazardous aspects of these operations, the pad systems are operated remotely in a control room located approximately three miles away. In addition, the cryogenic properties of the propellant dictates that loading the Shuttle occurred within hours of launch leaving little time to resolve issues that arise in narrow launch windows. These time-critical and high-risk operations makes the designated systems good candidates to be 'outfitted' with anomaly detection and fault isolation enhancements.

This case study involves taking the LH2 system initially and replicating it in an MBSE format. A unique approach is planned that models the system not only as it operates nominally, but as a system of 'failures.' This involves capturing the component states at a given failure mode and modeling the subsequent actions (behavior) as an impact to the sensor array. It is envisioned that this method will better enable the detector/isolator selection process. Being able to match the application capabilities directly to failure modes they are designed to detect should facilitate the application-system pairing process.

This initial modeling includes identifying those components and assemblies in which detection/isolation attributes are desired as these will evolve into requirements. At this point, the anomaly detection and fault isolation modules will be integrated into the model. The goal here is to ensure all potential fault sources are covered and corresponding system requirements are being satisfied. Legacy LH2 system problem data will be available for ingestion into the model while developing and testing this case study.

Evaluation

This section centers on the analysis of the case study results. This includes verifying the progressing design to include confirmation that requirements are fulfilled and all system interfaces are identified. During evaluation, a methodology will be developed that optimizes the component selection. By optimal, it will assume a design that meets requirements while lessening complexity, and subsequently, the aggregate cost for design, implementation and procurement. This will be accomplished by minimizing the number of detection/isolator applications and enabling data sharing via common interfaces. The advantage to an MBSE approach is the capability to insert/remove various components (from both system and/or detector/isolator applications) and assess the overall impact on the design. This is expected to ease process development. Finally, the resultant data will be interpreted, synthesized, and all findings uncovered shall be reported.

Framework

Testing via the system (and problem data) provided from the case study, evaluation of the results and framework development is expected to be an iterative process. This task will focus on capturing this process and will ultimately define the framework. The initial phase will continuously modify the model until a (sufficiently) functional framework emerges. This will be followed by fine-tuning the framework to achieve some optimizing characteristics for the selection process.

Once the resultant framework is specified, it will be validated. This will be accomplished by using the framework to augment the LO2 system with anomaly detection and fault isolation capabilities. Both real system faults (legacy) and synthetic problems will be used to test the model. This section concludes when a framework can be validated that ideally meets the research objectives.

Conclusion

The conclusion will summarize the research to include analysis, interpretations, findings, results and concluding remarks. This will also comprise the various accomplishments and their relation to the research objectives. Recommendations will be suggested for future work from either related questions raised during the study in areas that may benefit from closer examination, or for the next logical path in further developing a standard that integrates anomaly detection and fault isolation technologies.

CHAPTER FOUR: PRELIMINARY FRAMEWORK

This chapter proposes a preliminary framework that forms the foundation from which this research effort is based. This framework will describe the principles and procedures used to pair anomaly detection and fault isolation (AD/FI) applications to new or existing complex systems.

This framework involves a multi-stage process as outlined below:

- Ascertain and scope the system to be augmented
- Identify and categorize the sensor data available for ingest
- Identify and categorize the potential system faults
- Identify and categorize the possible AD/FI applications for consideration
- Model the system
- Model the AD/FI techniques
- Perform MBSE-centered ‘trade studies’ of the various AD/FI techniques
 - Evaluate/analyze those tested
- Make recommendation(s)

Some of these processes may work in parallel while others have distinct predecessors and/or successors. Reference Figure 4 - Preliminary Framework Process Flow for a process flow diagram of the initial framework.

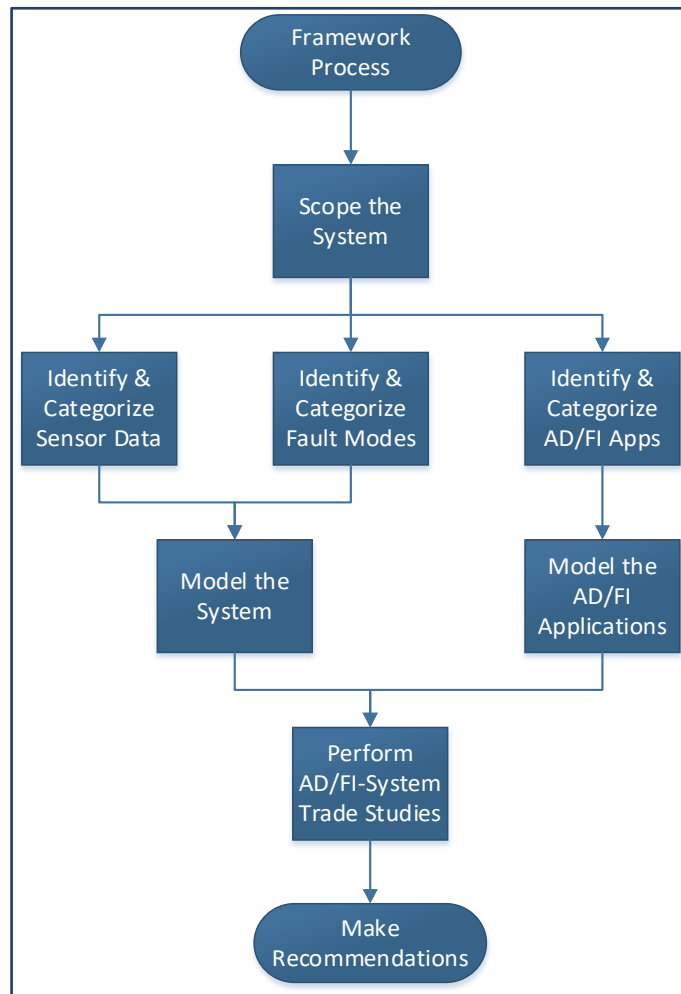


Figure 4 - Preliminary Framework Process Flow

System Scope

The first step is to define the system to be enhanced with AD/FI capabilities and determine the scope to which detection is required. This scope not only includes the breath or boundaries of the system, it is also comprised of the level of granularity to which detection abilities are applicable. These boundaries should encompass the system itself, the sensors that provide

feedback to the operators and the command-and-control subsystems (both reception and transmit locales).

The level of detail at which the detection capabilities must function must also be defined. This detail level will be dependent on the system design combined with requirements derived from the stakeholders. Typically, this detail will go to the component level at which a specific element is replaced. However, when the system includes redundant subassemblies or process legs, then detection may be required only for this level as the secondary assembly/process may be brought online should the primary subsystem fail.

Scoping the system should also identify AD/FI capabilities that already exist within the system. This can avoid unnecessary overlap in cases where existing techniques are robust. It may also identify cases where additional enhancement is required for capabilities that may be underprovided.

Sensor Array

The sensors are the principal means of providing visibility into the health and status of a remote operated system for those monitoring its performance. This is also the limiting factor in the ability to determine if an anomaly is occurring and what is the root cause for that problem. The sensors are designed into the system in positions that communicate key information for specific operational scenarios. For complex systems, it may take many such measurements to provide an

adequate status. Often, system health and status is inferred from a combination of indicators, and not necessarily as that specific measurement were originally intended to be used.

As the sensor array provides the view of the system, it is very important to identify all the sensors available within the system. These indicators will determine the detail level scope at which the system will be modeled. There is no need to provide high granularity detail if the sensor array does not provide high resolution visibility. Once the sensors are identified, they must then be categorized based on the type of data they provide. This will include both direct and indirect information that can be gathered from these indicators. This is a key step. Many of the AD/FI techniques are based on their ability to garner bits of information from multiple sensors and provide an accurate depiction of the system status.

Determine Potential Faults

All potential faults that can adversely impact system performance must be identified. The resulting list will drive requirement development stating that the system shall have the capability to detect such faults. This will initially be accomplished using a fault tree analysis approach. Fault trees are a graphical method that model component failures and also show how such failures can propagate through the system (Ruijters & Stoelinga, 2015). As the name implies, this is a tree structure that identifies basic (circle) and intermediate (rectangle) events that could possibly lead to the issue denoted in the top-level block. These events (or failures) follow a path towards this top-level anomaly, and this path is controlled by AND or OR gates. Reference

Figure 5 - Valve Component Fault Tree for a partially developed fault-tree representing a remote operated valve).

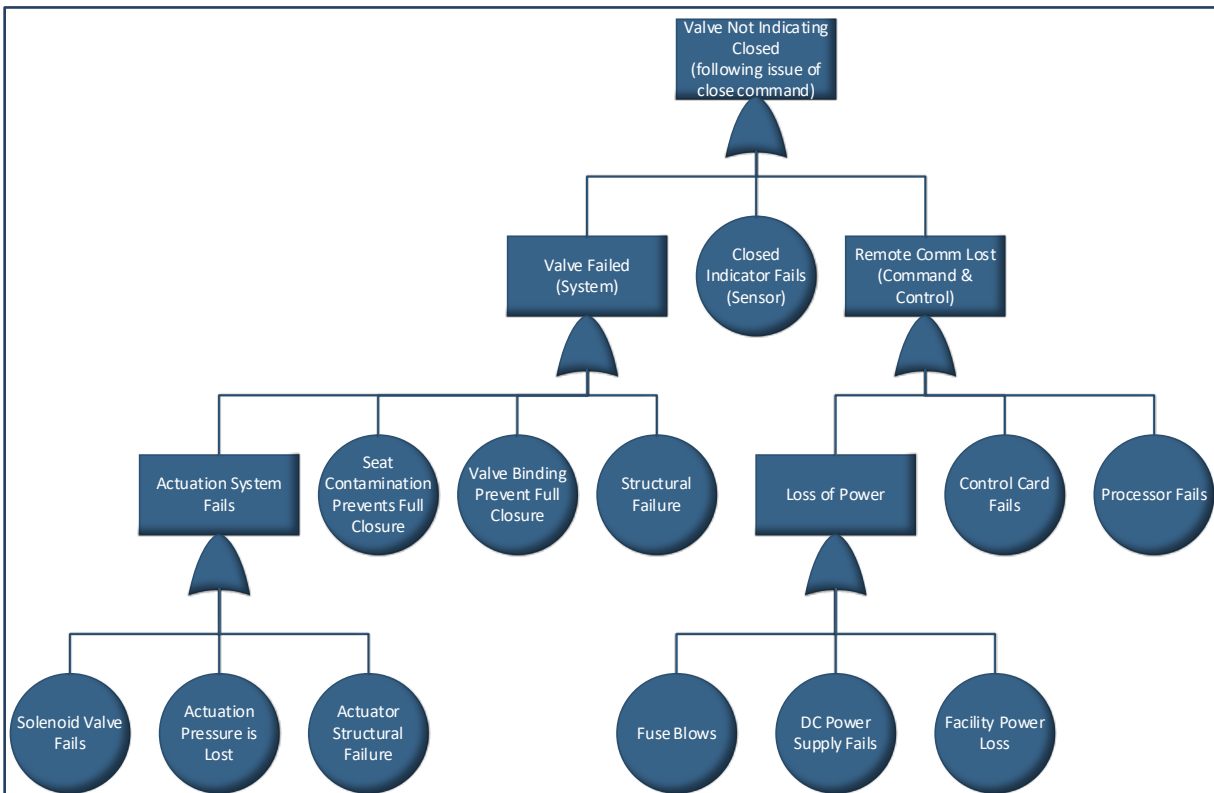


Figure 5 - Valve Component Fault Tree

A systematic approach should be applied to bind the number of potential faults. Initially, this will encompass all components at the operational level at which they are replaced should a failure occur. However, there may be circumstances when it is not practical to provide detail all the way down to the component level. This could include cases where the component is not that critical and its loss will have minor, if any, impact on the system. In addition, there may be redundant process legs that can be completely isolated from one another. The failure mode in this scenario need only be identified to one of the redundant subassemblies. Finally, the system

visibility provided by the sensor array will likely not cover 100% of the system components, thus the system design forces a reduction in failure modes for inclusion.

Fault Reduction from Sensor Capability

The sensor array will be the primary factor in resolving the failure modes identified by the fault tree to those in consideration for the framework. For those components that have some degree of sensor oversight, criticality will be assumed (and assumed non-critical if sensor visibility is lacking). If indicator granularity can only provide insight to a subassembly level, then the corresponding failure mode will only be identified to this level.

A ‘Failures vs. Measurements’ table was produced using the fault-tree failures and a hypothetical suite of corresponding measurements (reference Table 4 - Failures vs. Instrument Matrix). This table uses a “D” to denote an indicator that *directly* monitors for a particular failure. Assuming that sensor has not failed, then it is a sufficient data point to ascertain the corresponding failure mode as the problem source. An “I” represents an *indirect* measurement. These cannot exclusively determine the failure mode and require additional collaboration to reach a conclusion.

Table 4 - Failures vs. Instrument Matrix

	Seat Contamination	Valve Binds	Structural Failure	Actuation Solenoid Fails	Actuator Pressure is lost	Closed Indicator Failure	Control Card Fails	Processor Fails	Power - Fuse Blows	DC Power Supply Fails	Loss of Facility Power
Valve Fails to Close	Valve						Ind	C&C			
Open Indicator	I	I	I	I	I	I	I	I	I	I	I
Close Indicator	I	I	I	I	I	I	D	I	I	I	I
Upstream Pressure	I	I	I				I				
Downstream Pressure	I	I	I				I				
Downstream Temperature	I	I	I				I				
Actuation Pressure				I	D						
DC Voltage										D	D
Amperage				I						D	I
Comm/Health Relays								D	D		

The above matrix (Table 4 - Failures vs. Instrument Matrix) can be used to further reduce the number of failure modes. If a failure mode results in duplicate 'mode vs. sensor' allocation, then these are candidates for merging into a single problem. In this case, the instrumentation may detect a valve failure, but cannot distinguish between Seat Contamination, Valve Binding, Valve Structural Failure or Actuator Structural Failure.

In the process of scaling down potential faults due to sensor limitations, it will not be unusual to find gaps in the design that may allow critical processes to fail without detection. This can be related to a design process that focuses on operational requirements. By performing an analysis of the various fault modes, weaknesses in sensor types and distribution may be uncovered. This

is a key point, and one that emphasizes the need to complete the arduous task of identifying the majority of the potential faults. The task of selecting AD/FI should be biased heavily towards anomalous conditions and less so towards nominal operations.

Anomaly Detection/Fault Isolation Applications

A review of the available AD/FI techniques should be performed to determine which applications should be considered for system inclusion. This will be based on the requirements generated that the application is expected to satisfy. It should not be assumed that a single application will suffice. An ‘all-purpose’ algorithm may give up precision to accommodate a broad detection capability while issue-specific methods may provide the needed accuracy, but fulfill fewer requirements. Several factors may be used to prescreen which techniques will be applicable for the given circumstance. These can include:

- Budget
 - License costs to purchase an existing application
 - Setup costs to ‘customize’ the application for the given system
 - Costs to develop a non-commercially available application
 - Maintenance and data gathering to support functionality
 - Hardware platforms/system integration
- Effectiveness
 - Meets requirements
 - Specific functionality vs. general application

- Accuracy
 - Captures all (most) issues
 - Minimal ‘false’ alarms

The AD/FI applications will be addressed as classes that describe how their corresponding techniques function. The framework will make a recommendation at this class level. Therefore, it will be incumbent on the user to determine if commercial applications exist from which to select the final product, or if development of a custom application is required. The following list outlines the AD/FI classes that will be developed for this framework.

- Anomaly Detectors
 - Rule Based
 - Nearest Neighbor
 - Clustering Algorithms
 - Neural Network
 - Statistical/Parametric
 - One-Class Support Vector Machines
- Fault Isolators
 - Physics Model
 - Expert Systems
 - Fault Map Model

Model the System

Using a model-based system engineering (MBSE) approach, the system will be modeled by means of the system modeling language (SysML). SysML uses a complement of diagrams to portray the system graphically for users and stakeholders. These diagrams provide a 'view' of a portion of that system. However, there is an underlying structure that connects the various diagrams and interrelates with the model elements that are generated. The package diagrams will be used to portray the structure of the model. When modeling the system, the following list highlights fundamental elements that will be used to compose the model.

- System Structure
- System Behavior
- Constraints
- Requirements
- Include Existing AD/FI Capabilities

The SysML diagrams are designed to support model development specific for this functionality. These elements are explained in detail in the following sections.

System Structure

The system will be modeled by first focusing on the system structure. The SysML block definition diagram (BDD) and internal block diagram (IBD) are used to define the system structure. The fundamental element of structure in SysML is called a block which is used to represent systems, subassemblies and components (among other abstractions). A BDD is used to describe the structural schema of a system, and is composed of blocks that show their

relationship with other blocks. A BDD was generated for a 'remote operated valve' assembly as an example (reference Figure 6 - BDD for Remote Operated Valve Assembly). This BDD shows that the Remote Operated Valve is composed of an Actuator, Valve and open/close Solenoid Valves. The valve is also composed of 1 or 2 limit switches.

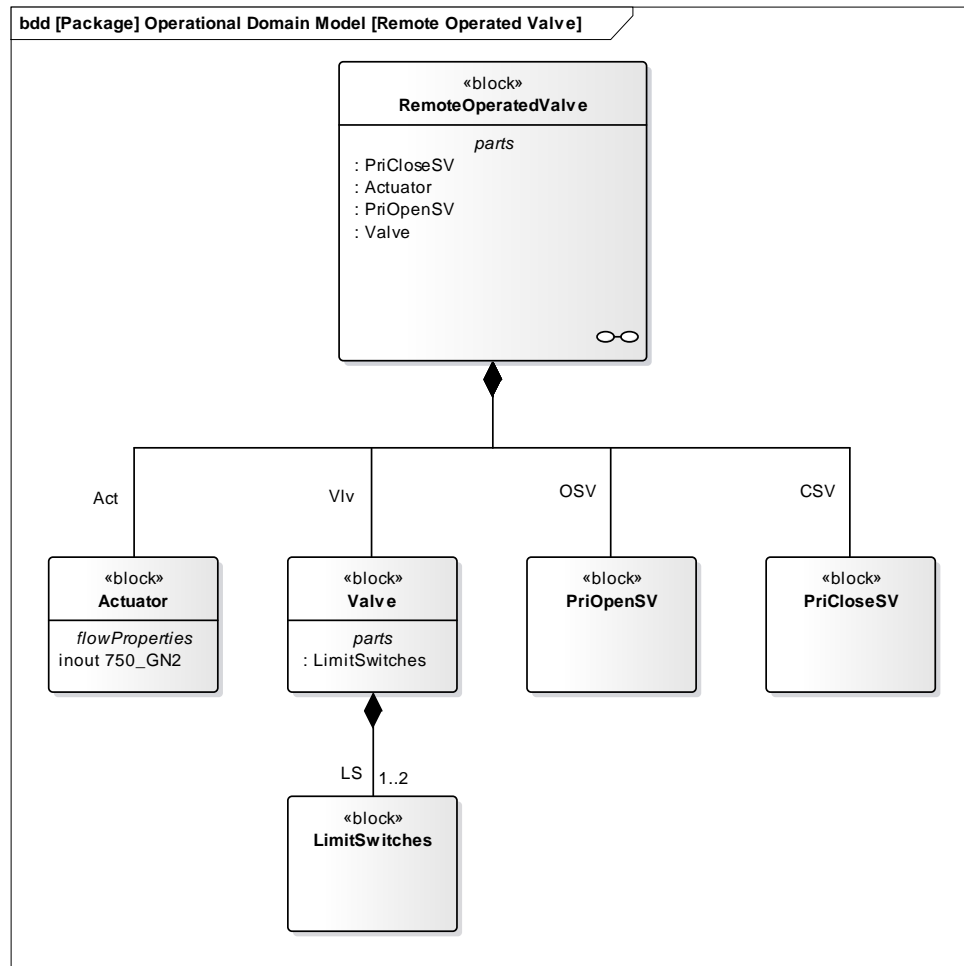


Figure 6 - BDD for Remote Operated Valve Assembly

An IBD is used to show the internal connections of the parts within a block. This is used when there is a desire to add resolution to the composition of a block. An IBD was produced (reference Figure 7 - IBD/Remote Operated Valve Assembly) that uses the parts that make up

the Remote Operated Valve block (per figure 6). This diagram shows how the various parts interface with one another. The parts include ‘ports’ that reveal some type of media is passed between those parts. In this case, if the open and closed solenoid valves are energized, 750 pounds per square inch (PSIG) of gaseous nitrogen (GN2) is applied to the actuator’s open side while the closed side is vented. This forces the actuator to move upward which opens the valve (connected by valve stem).

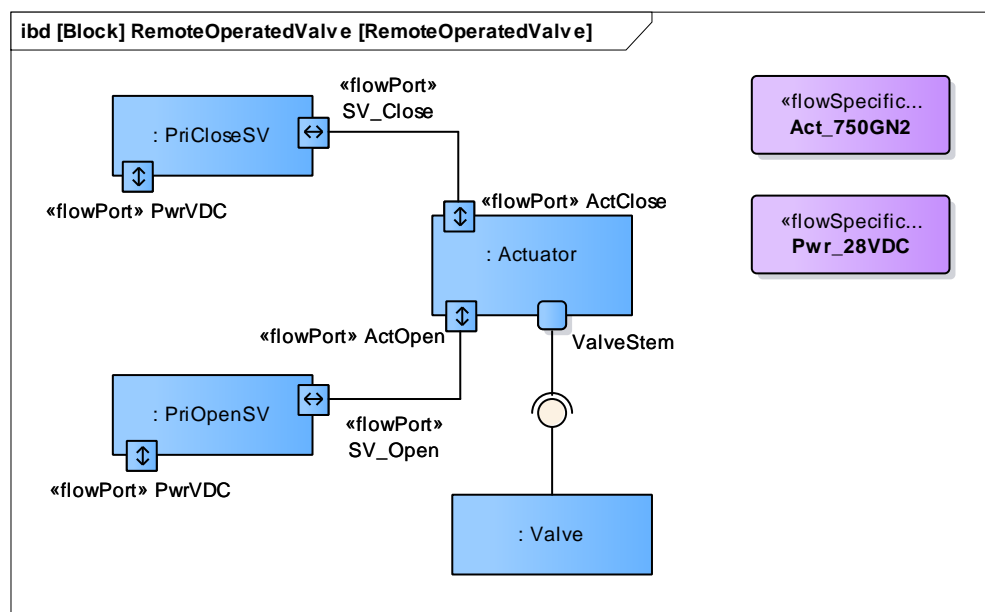


Figure 7 - IBD/Remote Operated Valve Assembly

System Behavior

SysML also provides diagrams that depict system behavior to include Activity, Sequence, State Machines and Use Cases. A use case diagram is simply used to show (typically) high level use cases that the system may perform. A sequence diagram shows the interactions among the

various system elements (or environment) based on ‘messages’ between these elements. These two diagrams will be used in this framework to a lesser degree (if at all).

To model system behavior, this effort will concentrate on capturing that behavior using activity and state machine diagrams. An activity diagram is used to portray behavior over time with an emphasis on the flow of matter, energy and data among a set of actions (Delligatti, 2013). State machines focus on event based behavior, and show how the system reacts to an event via state changes of the model elements. These events are often asynchronous which is consistent with anomaly occurrence within a system. State machine diagrams will be used to represent anomaly events and the subsequent impact these events have on the system in the form of state changes. This is a very important aspect of the modeling effort. Modeling the failure modes will enable the ability to adapt model segments of the AD/FI applications to the system model, and subsequently, the ability to test performance of those model sections.

Constraints

A parametric diagram is used to express constraints in the form of equations, expressions or rules (Holt & Perry, 2013). This will provide support for analysis in the performance of the AD/FI applications being tested. Violating a constraint is what signals the model that an anomalous condition is occurring, thus constraints will be tied closely to the system sensors.

Requirements

A requirements diagram is also provided by the SysML modeling language. As stated earlier, requirements will be generated for those anomalies that the system *shall* require the capability to detect. The requirements diagram is text based, though it allows one to link requirements to both structural and behavioral model elements. This enhances the traceability between the requirement, its implementation and satisfaction.

Existing AD/FI Capabilities

When modeling the system, it will be important to identify existing AD/FI capabilities embedded within the system. This will minimize the duplication of capabilities when selecting from the various applications, though some overlap will be expected. Often, these existing capabilities will fall short of the desired detection level. Hence, the need to augment those capabilities with additional coverage.

Model AD/FI Applications

Similar to modeling the system, the AD/FI applications too must be modeled. As previously stated, a constraint violation will flag the system that an anomaly is occurring. Therefore, the goal in modeling these techniques is to further bind the constraints which results in a higher expectation that an exception will occur. These 'easier' exceptions should correlate to a quicker detection of a problem from a wider range of potential anomalies. This will be accomplished by using the parametric diagrams to capture the application techniques, and then describing these techniques in the terms of a system constraint.

Trade Studies/Application Evaluations

Within the SysML literature, it is at times stated that an MBSE approach enables the ability to perform trade studies. However, there is little written that formalizes this process. This is not unusual as SysML does not dictate model methodology, it only specifies the language in the form of rules. The actual model implementation is left to be formed by the user.

As part of the trade study, it is important to evaluate each of the alternatives and quantify the value that application can add to the system. For a complex system, it is unlikely that a ‘one-size-fits-all’ application will suffice, thus several of the alternatives may be required. Analysis will be required to rank the options. The goal will be maximizing the effectiveness while minimizing the cost (assumed to correlate to the number of applications).

The Object Management Group (OMG), the organization that governs the SysML standard, has recognized the necessity of trade studies. In the current specification for SysML 1.3, the OMG includes an Annex for “non-normative extensions” that it may consider for inclusion into the language in future versions. This Annex (D.3) describes the extension of a parametric diagram to support trade studies and analysis. A trade study will be used to evaluate a set of alternative AD/FI based on predefined criteria. An objective function can be used to represent the criteria and determine the value of each alternative. A measure of effectiveness (MOE) will represent a parameter with a value that is essential for determining the performance level of the alternative applications (“OMGSysML-v1.3-12-06-02.pdf,” 2012).

By extending the SysML language as outlined above, the process of performing the trade study and evaluating the alternatives can be accomplished using an MBSE approach. This effort then becomes embedded within the SysML model hierarchy, and subsequently, is available for recall if system design artifacts are requested.

Make Recommendation(s)

Following the trade study, a list of recommendations should be produced. These recommendations should be consistent with the analysis completed, however they should also take into consideration the deficiencies that were observed during this process. For instance, not all of the requirements may have been fully met with the available suite of candidate AD/FI applications. This may drive a modification to the requirement or to the system itself. It may also identify the need for a custom detector/isolator to meet the requirement. In addition, this type of analysis will typically uncover inadequacies in the system design. This may highlight the need for additional sensors to provide added feedback, or it may uncover critical components without adequate redundancy. All such findings should be included in the recommendations.

CHAPTER FIVE: CASE STUDY

This chapter will present a case study to describe the implementation of the anomaly detection (AD) and fault isolation (FI) selection framework. As stated previously, the system under study is the liquid hydrogen (LH2) system at the Kennedy Space Center (KSC). This system is located at the launch pads and was used to load both the Apollo and Space Shuttle launch vehicles. It is currently going through a redesign process to support NASA's next generation Space Launch System (SLS) program.

The LH2 system provides the fuel for the launch vehicle's oxygen/hydrogen engines. For Shuttle, nearly 400,000 gallons of this fuel was loaded into the external tank (ET). Working with LH2 poses many technical challenges. First, LH2 is a cryogenic fluid at -423 degrees Fahrenheit. This extremely low temperature drives a system design that must be highly insulated to minimize the commodity boil-off, and the hardware itself must be able to operate while withstanding thermal cycles from ambient to cryogenic temperatures. As hydrogen is the smallest molecule known, it is can prove difficult to keep leak-free within the system, a highly desirable feature given that hydrogen is extremely flammable. Liquid hydrogen also poses risks to personnel in that direct exposure will cause severe cryogenic burns, induces asphyxiation if released in confined spaces and has a propensity to ignite and/or detonate if large quantities are released in air.

These safety concerns, combined with the technical challenges, result in the cryogenic tank loading operations being performed remotely with the Pad cleared of all personnel. The

astronauts and support crews do not enter the pad until the initial filling is complete. At this point, only ‘replenish’ loading operations are underway to make up for boil-off losses (over 100 gallons per minute). To minimize the boil-off losses, loading operations commence as late as possible resulting in a time-critical process. It is these operationally complex, highly hazardous and time-critical characteristics that make this system an ideal candidate to augment with AD/FI technology.

Framework Development

This case study will follow the proposed framework identified in Chapter 4 for initial system model development (reference Figure 8 - Preliminary Framework Process Flow). As the course of developing this case study model is anticipated to be an iterative process, this framework will be improved and the implementing details refined as the model matures. The final framework will be presented following analysis in the next chapter.

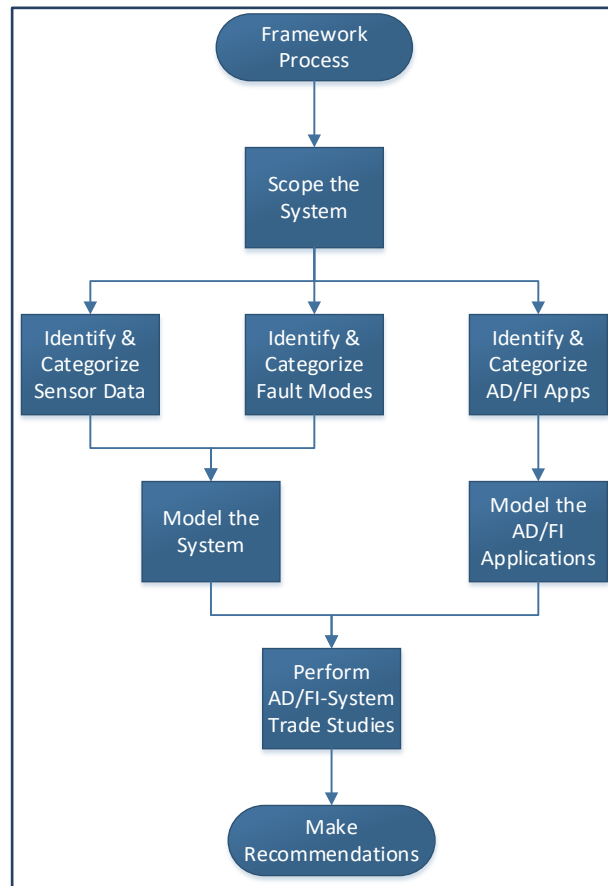


Figure 8 - Preliminary Framework Process Flow

System Scope

The first step delineated in the proposed framework is to scope the system to be augmented with enhanced AD/FI capabilities. This scope not only includes the breath or boundaries of the system, it is also comprised of the level of granularity to which detection abilities are applicable. These boundaries should encompass the system itself, the sensors that provide feedback to the operators and the command-and-control subsystems (at both reception and transmit locales).

The level of detail at which the detection capabilities must function need also be defined. This detail level will be dependent on the system design combined with requirements derived from the stakeholders. Typically, this detail will go to the component level at which a specific element is replaced. However, when the system includes redundant subassemblies or process legs, then detection may be required only for this level as the secondary assembly/process may be brought online should the primary subsystem fail.

Scoping the system should also identify AD/FI capabilities that already exist within the system. This can avoid unnecessary overlap in cases where existing techniques are robust. It may also identify cases where additional enhancement is required for capabilities that may be underprovided.

Liquid Hydrogen System

The LH2 system within the launch pads at KSC was used to fill the fuel portion of the space shuttle's external tank (ET) with nearly 400,000 gallons. The LH2 was used as the fuel for the shuttle's three main engines. This same system is planned to support the next generation Space Launch System (SLS). Part of the system resides on the mobile launcher platform (MLP). The shuttle vehicle is mounted on the MLP in the vehicle assembly building (VAB). The MLP then rolls to the pad the LH2 lines are mated at the Pad/MLP interface to 'complete' the system. The LH2 system hardware for both the pad and MLP are included in this scope. The LH2 is stored in a vacuum jacketed storage tank with a total capacity of 900,000 gallons.

There are three primary hardware subsystems built around this storage tank that enable the transfer of LH2 to the flight vehicle. These include pressurization, transfer and vent systems. In addition, a command and control system (C&C) is used to enable remote operations from a safe distance. These subsystems are further detailed as follows.

Pressurization Subsystem

As LH2 is a very light liquid (0.591 lbs/gal), and the ET operates at lower pressures, the use of pumps to flow the liquid is not necessary. Instead, the storage tank is pressurized to a nominal pressure of 66 PSIG for initial higher-flow operations, and subsequently lowered as the flow rate is decreased. The primary components that comprise this subsystem include a main and auxiliary vaporizer (heat exchangers), a variable flow control valve (main) and control valves (main & aux). The vaporizers are supplied LH2 from the tank separate from the cross-country transfer lines. The vaporizer outlets return the gaseous hydrogen (GH2) to the top of the tank (reference Figure 9 - LH2 Pressurization System). LH2 has an expansion ratio of 833:1, so vaporization of a relatively small amount of liquid provides an adequate gas volume that is compressed to pressurize the tank. As LH2 has a boiling point below -400 degrees Fahrenheit, exposing the liquid to near ambient temperatures will force the evaporation necessary to generate tank pressure.

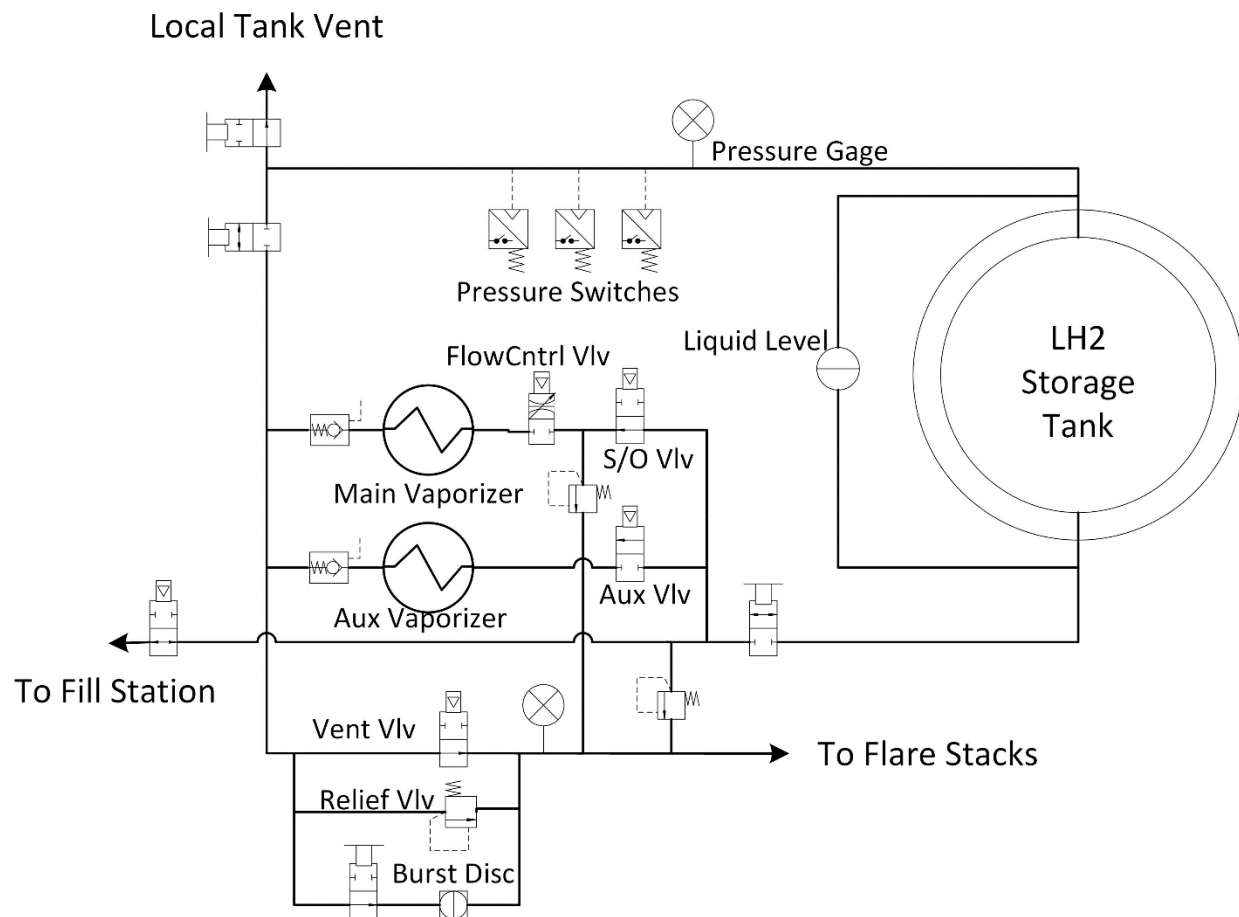


Figure 9 - LH2 Pressurization System

Transfer Subsystem

The transfer subsystem consists of piping that traverses the pad and MLP and connects to the flight vehicle via an umbilical. This piping is dual-walled with a vacuum maintained between the piping's annular-space. These vacuum-jacketed lines provide the insulation necessary to minimize the boil-off of the cryogen LH2.

These cross-country transfer lines also include two valve control assemblies. One valve complex is located at the base of the storage tank, and is comprised of several valves that allow both high and low flow rates, as well as venting capability. The other valve complex is located on the ML just upstream (when loading) of the vehicle umbilical. This valve assembly is used in conjunction with the pad control system to provide various flow rates during vehicle loading, and also supports drain operations should the launch get scrubbed for that day.

Vent Subsystem

As hydrogen is highly flammable, the GH2 is not allowed to be vented directly overboard from the vehicle during loading operations. Instead, this vented gas is captured and routed to flare stacks for safe disposal. There are four primary sources of vented GH2 as follows:

1. The boil-off gas generated from the ET during fill operations.
2. Within the flight vehicle, a small volume of LH2 is diverted to the engines to provide thermal conditioning during the loading operations. The LH2/GH2 from this 'bleed' flow is captured by the vent system.
3. The LH2 storage tank (following pressurization).
4. Each section of the cross-country transfer lines that can be independently isolated.

The vent system includes the isolation valves for all vent sources. The vehicle sources include both vehicle and ground isolation valves (vehicle valves are out-of-scope for this case study).

The vent system also includes two flare stacks, and corresponding subsystems, that support burning the exhaust GH2.

Command and Control Subsystem

The C&C architecture is composed of computer servers (and supporting peripheral equipment) within the control rooms that communicate with programmable logic controllers (PLCs) in the vicinity of the hardware for which they control. The operators interface with their system via keyboard and display(s). The PLCs located in the field directly energize/de-energize the equipment based on the operator's (or automated software) commanding. These PLCs also provide instrumentation feedback for monitoring system performance.

As the operation and maintenance of C&C hardware requires a different skill set than those performing launch vehicle loading operations, it is classified as a different subsystem from the LH2 subsystem under study. Much of the C&C hardware has health and diagnostic functionality built into the architecture, so there are limited opportunities to augment this with additional value-added AD/FI technology. This health is monitored by that subsystem when the control room is active. The C&C system also impacts multiple subsystems making it difficult to limit the scope for this analysis. However, it is imperative that the operators know if they are dealing with an issue related to their subsystem hardware as this directly influences the course of action going forward. As the PLCs include numerous command or measurement cards (with less 'health' capability), the PLC control system will be included in this scope.

Existing AD/FI Capabilities

Scoping the system includes identifying the current AD/FI capabilities already employed. This information is used to disqualify potential redundant AD applications from consideration, or

select AD technologies intended to enhance those existing capabilities. For the LH2 system under review, the primary method of anomaly detection is to bind tolerances to the instrumentation. Should an exceedance occur on one of these sensors, an alarm is generated that alerts the operators. This includes discrete measurements (i.e. position indicators or pressure switches) in which case the alarm-state is set to the opposing nominal state. It also pertains to analog measurements in which specific tolerances can be set both above and below a nominal range. There are no fault isolation applications used within the LH2 system.

There are external AD provided by other subsystems that monitor LH2 operations. As stated earlier, much of the C&C subsystem has health detection embedded within the architecture. Any exceptions observed are annunciated over an audio communication network as the ability to field an alarm on console may be suspect. There is also a Hazardous Gas Leak and Fire Detection system. This subsystem monitors all vehicle and ground subsystems operating with hazardous commodities.

Scope Overview

The LH2 system scope for this case study can be defined as follows:

- Pressurization subsystem to the component level
- Transfer subsystem to the component level
- Vent subsystem to the component level
- C&C subsystem to PLC end items only
 - Need ability to differentiate between system vs. C&C failure

Existing AD/FI capabilities in which duplication is not desired include:

- Alarm setting on system sensors
- Health status for C&C processing components
 - Monitored by C&C subsystem
 - Limited health status for C&C end-item components (directly interfaces with hardware)
- Hazardous gas leak and fire detection
 - Monitored and operated by HGLFD subsystem

Identify and Categorize Fault Modes

To adequately apply AD/FI techniques on the given system, the potential faults that can have detrimental consequences will need to be identified. There is an assortment of methods available to include Failure Modes and Effects Analysis (FMEA) and its extension Failure Mode, Effects, and Criticality Analysis (FMECA), Design Review by Failure Mode (DRBFM), Fault Tree Analysis (FTA) and its extension Event Tree Analysis (ETA), Hazard & Operability Studies (HAZOP), Hazard Analysis and Critical Control Points (HACCP) among others. The most predominant techniques used in industry are FMEA and FTA (Cristea & Constantinescu, 2017).

Fault Tree Analysis

Fault tree analysis will be used to identify potential faults for this case study. Fault trees (FTs) offer a graphical breakdown with regard to the hierarchy of failure modes. The base of the tree is called the top event and the leaves are called basic events (Adler et al., 2011). Fault trees use a

top-down methodology that depicts, via a graphical representation, of how an anomaly can propagate through the system. It is this propagation that may stimulate the AD/FI techniques employed to detect system anomalies when direct signals may not be available/adequate to alert the users, hence the selection of FTA.

Fault Tree Development

For complex systems, fault analysis is often accomplished in parallel during the design's development phase. This aids the designers with identifying critical areas that may require fortification, redundancy and/or additional instrumentation for visibility. For added efficiency, a system engineering best-practice would then be to leverage off existing analysis. To facilitate this process, such analysis would optimally be embedded within the SysML model. The following depicts a method to auto-generate FTs based on the SysML model generated (Clark, Rabelo, & Yazici, 2017)

SysML uses diagrams to portray the system. The system's structure is represented by Block Definition Diagrams (BDDs) which are intended to describe the hierarchy of the structural elements, or blocks. A block can represent a single component or an entire system. Structure is further defined by Internal Block Diagrams (IBDs) used to depict the how the elements within a block are connected and the type of matter, energy or data that flows between them. The IBD provides an alternate view that can show the 'usages' of these blocks. Specifically, how the parts are connected (and the flow that occurs between these parts) that involve the portion of the system within the IBD view. For instance, a valve can be configured to an open or closed

position. For a closed valve, both internal and external leakage may be a concern. For a valve that is open for critical operations, internal leakage would not be a concern. This detail is depicted with IBD views, so concentrating on these diagrams should result in fault associations based on the component functions applicable to that subsystem. This combination of connectivity and flow can illustrate how a failure is distributed through a system. Thus, it is conducive to providing the necessary information in developing an FT.

The development of FTs for a complex system is not a trivial task. Many of the commercial FT software packages will support import sheets with component data (beneficial to those with existing component lists within their design documentation). However, identification of the relevant failure modes, linkage of components to applicable subsystems and assignment of the appropriate ‘gate’ is a manual process. As most of a system’s individual components will likely have multiple failure modes, the number of basic events generated can far exceed the number of components. As with most largely manual efforts, the input may be prone to errors, and omission of critical data likely to occur. Since FTs are a graphical representation of the system, they are difficult to condense without undermining the readability advantage from which they are based. Subsequently, traversing large FTs also presents a challenge and can hamper the reviews intended to find/remove such errors.

The system design information embedded within an IBD (with minimal model augmentation) is used to auto-generate FTs. The intent is to provide an initial FT that is all-inclusive of the components contained within the design, and therefore, minimize the errors and omissions that

may occur from manual generation. This also reduces the effort required by the safety engineer(s) as it is easier to modify or prune an existing tree vs. generating one from scratch. It should be noted that this method does not preclude the subject matter expert's (SME) involvement. SME reviews will still be required to identify unique failures, multi-failure modes, system-level (non-component) and external failures.

SysML/FT Abstraction

As stated earlier, much of the FT development is a manual process. Although there are commercial applications available that can assist with this process, the structure of an FT renders it unique and with minimal commonality among the numerous design models supplementing development. Subsequently, there is little overlap of information to be garnered in support of generating FTs. The process for constructing an FT can be summarized as follows:

- Identify a top event and corresponding intermediate events
- Scope the system to include all components that can contribute to the failure of these events
- Generate 'Failure Modes' at the component level (basic events)
- Link the events via Boolean gates to form the tree structure

Although a SysML IBD too lacks all the information needed to accomplish the process above, much of it can be found embedded within the IBD's design. The remaining gaps can be filled by extending the SysML model. An IBD represents a predefined block and graphically shows how the parts within that block interconnect (may include the flow of matter, energy or data among

these parts). These blocks can characterize the entire system, the various subsystems, component assemblies, components and even the component makeup if that is the level of detail desired by the stakeholders. As is typical for most MBSE methods, the system is first defined at a high-level. This broadly defined system is then decomposed into subsystems, an iterative process that continues until the desired level of detail is achieved. Therefore, a developed SysML model will contain IBDs that denote the system structure at all levels of the project.

The 1st step in FT auto-generation is identifying the top event, followed by the applicable intermediate events. This is accomplished by simply using the IBD frame title as this should accurately reflect the subsystem's functionality (assuming modeling best-practices employed).

The next step is to scope the system to ensure all applicable components are included. As the IBDs illustrate the system's design structure, they also define the system scope. All parts within the IBD that have failure modes identified will be included in the FT.

Creditable failure modes must be determined for each component within the IBD. These modes will be used to identify the basic events for the FT. This is information that is not readily available within a SysML project, and therefore must be added to the model. A block can be used to define the various failure modes, though for large projects, the user may want to create a specific stereotyped element to represent these modes. To create the failure modes for the initial FT, first categorize the components into common classifications. For instance, an 'indicator' class may include pressure transducers, temperature transducers, flowmeters, etc. For each

component class, list the generic failure modes that applicable to that class. Generic failure modes for a valve-class may include:

- Valve fails open
- Valve fails closed
- Valve position unknown
- Valve leaks externally
- Valve leaks internally

Note too that failure modes may be applicable to multiple component classes (i.e. ‘leaks externally’). Generating component classes results in a much smaller subset of failure modes compared to the overall component base. Subsequently, the SysML model updates to accomplish this step are minimal, compared to embedding this information within all the component blocks.

The last step for FT development is to link events via Boolean gates to form the tree’s structure. Linkage is already established between the top level of the localized FT (IBD title) and the parts contained within the IBD. However, a connection must be made between the components and failure modes added to the model. This can be accomplished by allocating failure modes to their corresponding component classes. SysML specifies the use of ‘allocations’ as a means of crosscutting the model and linking (integrating) the various model elements. An allocation simply reflects that if a change occurs to the ‘supplier’ side, a change *may* be needed on the ‘client’ side, thus it represents a dependency of the clients to their supplier (“OMGSysML-v1.4-

15-06-03.pdf,” n.d.). Allocation also allows for easy selection/deselection of the generic failure modes as not all will be applicable in every component instance.

A Boolean gate must also be inserted between each level of events. NASA’s Fault Tree Handbook with Aerospace Applications defines a “state of component” failure as one that is localized to a component (all other failures are deemed “state of system”) and that state-of-component failures should always utilize OR gates (Stamatelatos et al., 2002). This simplifies the gate selection between the components and the failure modes as all will be OR gates. However, the gates between the localized top event (likely a subsystem event) and corresponding component levels may utilize either an AND or an OR gate. For instance, both primary and secondary valves (redundant flow path) must fail for the system to fail. This relationship is FT modeled with an AND gate. This needs to be noted as that information is not readily embedded within the IBD or within the failure modes added to the model. Therefore, it needs to be addressed to maximize the integrity of the initial FT. The example that follows provides one method to accomplish this before FT auto-generation takes place.

Example

This section will provide an example of FT auto-generation. It uses an IBD that depicts the liquid hydrogen (LH2) storage tank pressurization subsystem (reference Figure 10 - IBD/LH2StorTankPressSys. This system pressurizes an LH2 storage tank which enabled LH2 flow to the Space Shuttle’s external tank. The last bracketed term in the IBD frame is ‘LH2StorTankPressSys.’ This is the top-level (subsystem) event for this IBD, and will later be

appended with “_Fails” as events should reflect the issue under analysis. All the parts within this IBD will make up the subsequent intermediate levels.

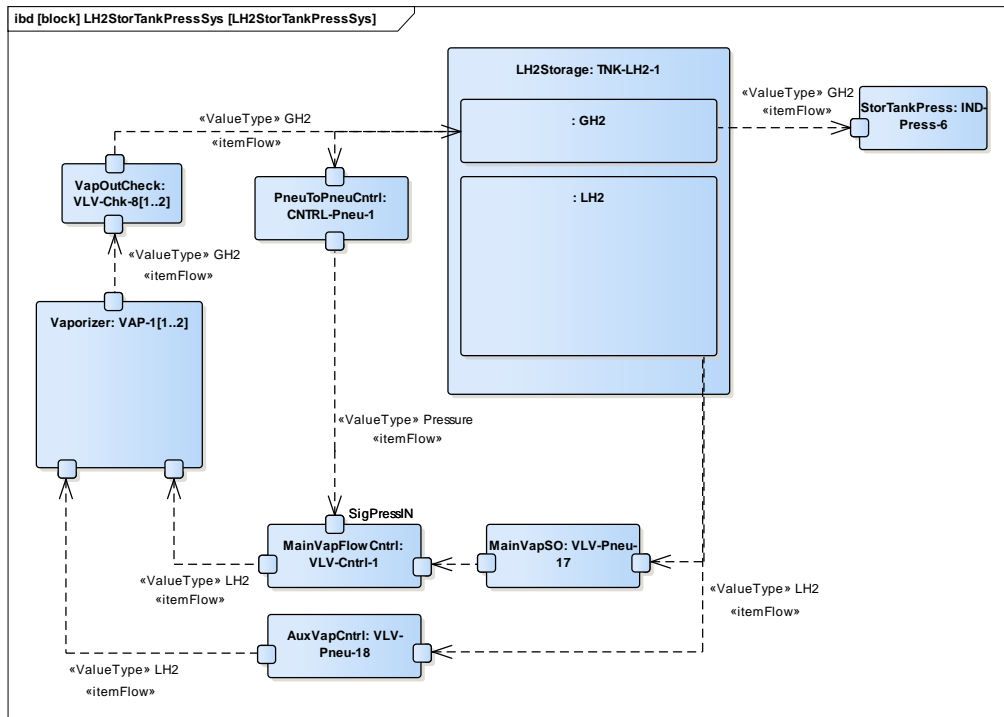


Figure 10 - IBD/LH2StorTankPressSys

The 1st step in extending the SysML model is to add the generic failure modes. For this example, failure modes have been produced for the component classes applicable to this IBD only. They

have been created as blocks and added to a package titled 'FailureModes' where they can be accessed from the model repository (reference Figure 11 - Model Repository Example).

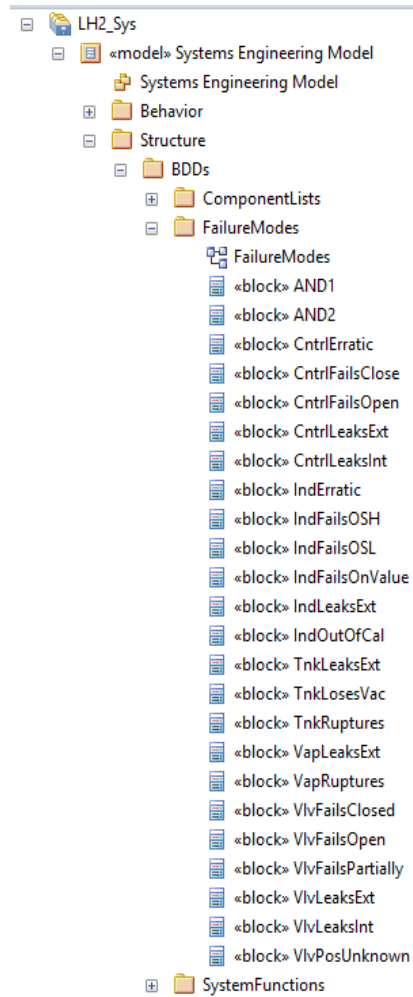


Figure 11 - Model Repository Example

With inclusion of the failure modes, they can now be allocated to applicable components. The parts within the system were initially generated as 'blocks.' However, these blocks can have multiple instances within a system, and each of those instances may have different functions. When an instance of a component block is used within an IBD, it is categorized as a part-

property. As the applicable failure modes may differ based on how that part is used, the failure modes should be linked to the individual part properties. This can be done via SysML diagrams and the corresponding tool's drawing features (reference Figure 12 – 'FailureMode' to Part BDD).

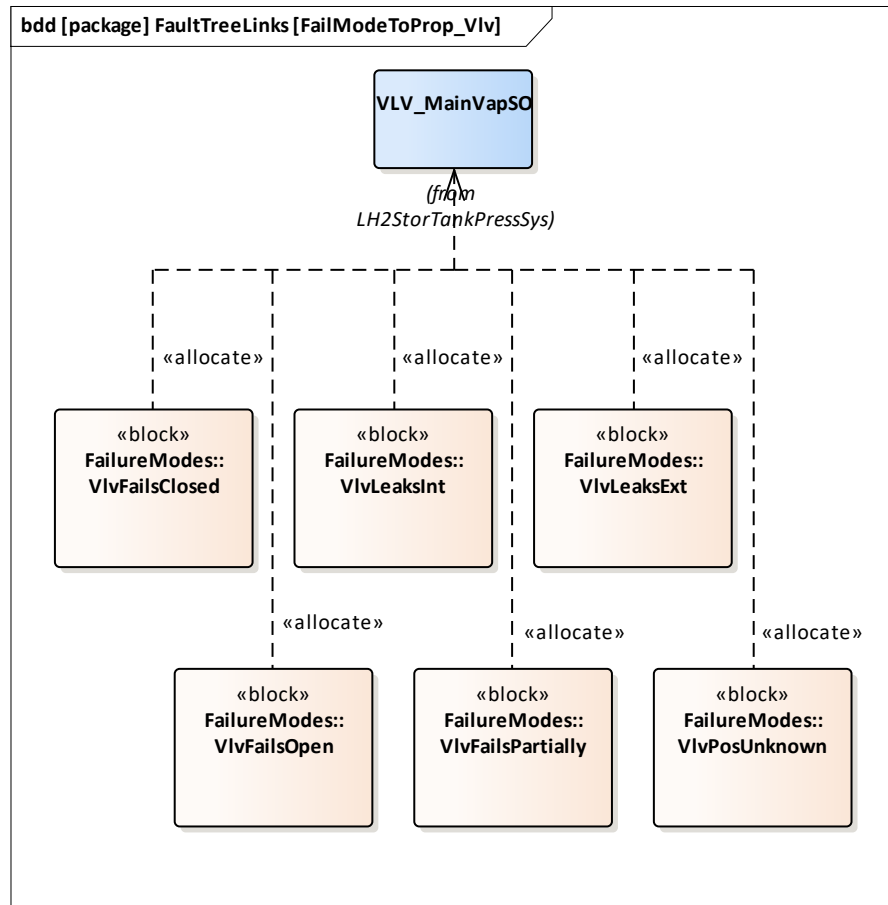


Figure 12 – 'FailureMode' to Part BDD

If a stakeholder need for a diagram view of these allocations is not required, then it is recommended that a Relationship Matrix be used. The SysML standard does promote the use of matrices, but does not standardized their use. Therefore, the functionality of matrices can differ

between the tools. A matrix enables easy selection/deselection for both individual and multiple blocks, and it also captures the IBD structure that is used to generate the FT (reference Figure 13 - Relationship Matrix).

Target \ Source		AuxVapCntrl	LH2Storage	MainVapFlowCntrl	MainVapSO	PneuToPneuCntrl	StorTankPress	Vaporizer	VapOutCheck
AND1		↑		↑					
AND2									
CntrlErratic						↑			
CntrlFailsClose						↑			
CntrlFailsOpen						↑			
CntrlLeaksExt						↑			
CntrlLeaksInt						↑			
IndErratic							↑		
IndFailsOnValue							↑		
IndFailsOSH							↑		
IndFailsOSL							↑		
IndLeaksExt							↑		
IndOutOfCal							↑		
TnkLeaksExt			↑						
TnkLosesVac			↑						
TnkRuptures			↑						
VapLeaksExt								↑	
VapRuptures								↑	
VlvFailsClosed		↑		↑	↑				↑
VlvFailsOpen		↑		↑	↑				↑
VlvFailsPartially		↑		↑	↑				
VlvLeaksExt		↑		↑	↑				↑
VlvLeaksInt		↑		↑	↑				↑
VlvPosUnknown		↑		↑	↑				

Figure 13 - Relationship Matrix

As noted earlier, the IBD does not readily signal which Boolean gate should be used at the component level. This is typically determined by analysis (and SysML is accommodating with inclusion of such data). However, using a matrix from which to build an FT does limit the information that can be embedded. To diminish the need to make such modifications after the

FT is generated, the following method was employed. Two ‘AND’ blocks were included in the model repository for failure modes. These blocks are intended to associate any AND conditions for the components within that IBD. It needs to be stated that this usage of blocks does not enhance the SysML model (and the element descriptions should be annotated accordingly). Lacking a standard that aligns SysML with FT generation, this simply provides a means to transfer information for external use. The IBD contains ‘main’ and ‘auxiliary’ vaporizer control valves. These should be reflected with an AND gate, and the Relationship matrix has been annotated accordingly.

The tool used for this example (Sparx’s Enterprise Architect) has the capability to export a relationship matrix in a ‘.csv’ format. The following steps outline the process for building an FT from the matrix. For this example, the ‘.csv’ file was imported into Excel. A VBA macro was written that performs the following steps to auto-generate the FT (OR and AND gates are shown textually) (reference Figure 16 - Excel/VBA Generated Partial FT):

- 1) Save the .csv file using the IBD name. Append “_Fails” to the file name and use this as the top event.
 - a. Place an OR gate below this event
- 2) For the next level, add all the components that are not allocated in the matrix to an AND block (target). Append each component name with “_Fails.”
 - a. Connect these events to the OR gate from the top level.
 - b. Place an OR gate below the component intermediate events.

- 3) If applicable: For those components intended for AND gates (target), create a new intermediate event (remains at the same level as the component OR gates). Event title to be composed of component names, and appended with ‘_Fail.’
 - a. Connect the events to the OR gate from the top level.
 - b. Place an AND gate below these component events.
 - c. Add a new level for those components linked to AND gates. Append each component name with “_Fails.”
 - i. Connect these events to the corresponding AND gate(s).
 - ii. Place an OR gate below these component events.
- 4) For all component events, add a new lower level with the corresponding failure modes (source).
 - a. Connect these events to the OR gates at the inetermediate level(s).
 - b. Place a circle below the failure mode events (denotes a basic event).

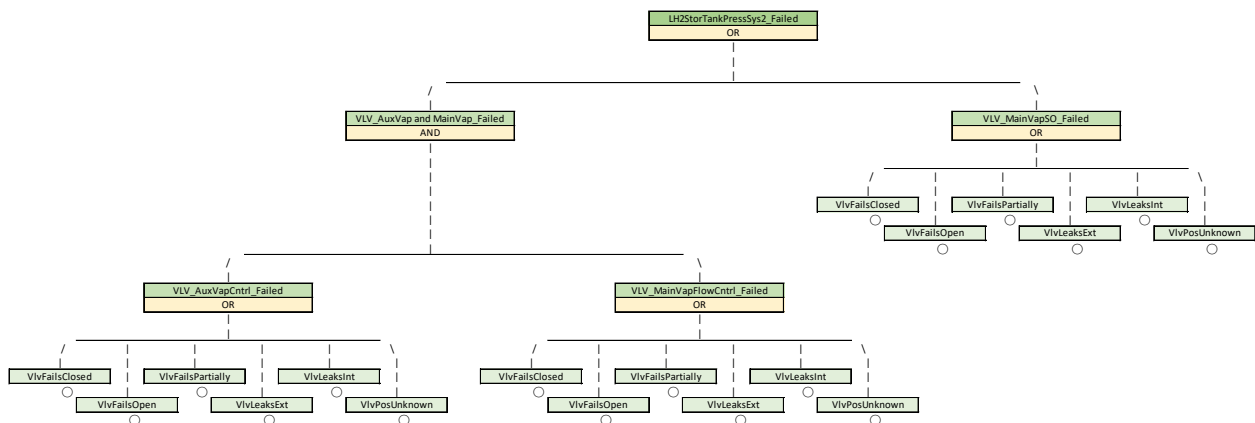


Figure 14 - Excel/VBA Generated Partial FT

As shown in Figure 14 - Excel/VBA Generated Partial FT), an FT can be generated with minimal extension of SysML to include the basis events (failure modes) and the structure internal to an IBD. Although there are limitations with the amount of information that can be transferred in a matrix, the following lists the advantages of generating FTs directly from the SysML model:

- Minimizes SME assistance for initial FT construction
 - SMEs develop the IBDs so this expertise is propagated to initial FTs
 - Stakeholder's IBD design review updates also transmitted to FTs
- Initial FT all-encompassing (component level) with inclusion of components identified within the SysML design
 - It is easier to prune or modify an existing FT than build from
- Relationship matrix provides an easy method to add/delete prior to FT generation
- FT organization consistent with SysML model (IBDs)
 - SysML updates transmitted to FTs
- Linkage between systems and potential failure modes collected in SysML model to support other analysis

Fault Tree Auto-Generation Limitations

The primary limitation is the depth or layering of the components within the IBD. With a two-dimensional matrix from which the data is transferred, capturing multiple levels of sub tiered systems and components within the IBD cannot be accomplished for FT development. Inserting multiple levels into an IBD can increase the complexity of that view, and subsequently, be

counter-productive to its readability. Modelers may minimize this practice, but it should not be restricted. For this method to be successful, multi-level IBDs must be further decomposed when encountered. Therefore, implementing this technique has the potential to drive IBD design.

Identify and Categorize Sensor Data

The sensors convey the operational status of a remotely controlled system. As such, a sensor can be defined as a component that provides feedback to the operator on the status of the system via the command and control architecture. They are the only means of providing an operator the visibility to determine the state of a monitored system. Therefore, the sensor array is the primary mechanism that can restrict and/or enhance the insight into system performance. System health and status is typically derived from sensors directly measuring a specific function, as well as a combination of indirect measurements that may have influence over that portion of the subsystem.

As the sensor array allows the system to be observed, it is very important to identify all the sensors available within the system. This information will be used to refine the fault tree developed for this system as the availability of measurements will influence the granularity of the failure modes detected. If the fault tree has identified failure modes that the existing sensors provide limited visibility and cannot reasonably detect, then it is not practical to expect an AD system to overcome this deficiency. However, a couple of insights should be noted. First, supplement AD systems are expected to be ‘smart,’ and perhaps capable of detecting issues with limited indications in ways that may not appear obvious to an observer. Second, if the (limited-

visibility) failure mode is credible and has potentially serious consequences, then the correct course of action may be a sensor modification to the system that facilitates detection of that failure mode.

The sensor array will be the primary factor in resolving the failure modes identified by the fault tree to those in consideration for the framework. For those components that have some degree of sensor oversight, criticality will be assumed (and assumed non-critical if sensor visibility is lacking). If indicator granularity can only provide insight to a subassembly level, then the corresponding failure mode will only be identified to this level.

In the preliminary framework described in chapter 4, a matrix was proposed to identify and categorize the sensor array (reference Table 4: Failures vs. Instrument Matrix). This matrix would relate the potential faults to the system's available measurements. In addition, the method applied would annotate if that measurement was a direct or indirect means of detecting the fault. This implementation would initially aid in reducing the potential faults being considered if it could be determined adequate instrumentation was not available to uncover those issues.

For this case study, the means to identify and categorize the sensors will instead build upon the current model (a systems engineering best practice). Relationship matrixes are fully available within SysML, and can be generated and modified using a matrix format, or graphically within the various SysML diagrams. Furthermore, the auto FT generator technique embedded the applicable faults within the model structure making them readily available to associate with other

model elements. The following will leverage off these existing faults to illustrate this development task supporting the framework.

When developing a SysML model, a ‘parts list’ is generated in the process of identifying the system structure. This is done by first creating Block Definition Diagrams (BDDs) depicting higher level structure, and then decomposing this system structure until the individual components are identified. Building off an accessible list within the model is not only efficient, but has the ability to capture additional associations which further develops the underlying structure of the model.

In addition to stand-alone instruments added to measure a given part of a system, indicators are also embedded within components to provide status for that component. The most common application for this case study are valves that are ‘switched’ to provide feedback to that valve’s open, closed or intermediate position. These indicators can provide discrete data such as a limit-switches that are placed so that they get ‘depressed’ when a valve reaches a given position (i.e. open/closed). A component may have 1 or more indicators to determine its position (reference Figure 15 – BDD of Remote Valve with 1 or 2 Limit Switches). They can also be potentiometer type indicators that provide an analog signal for variable position valves.

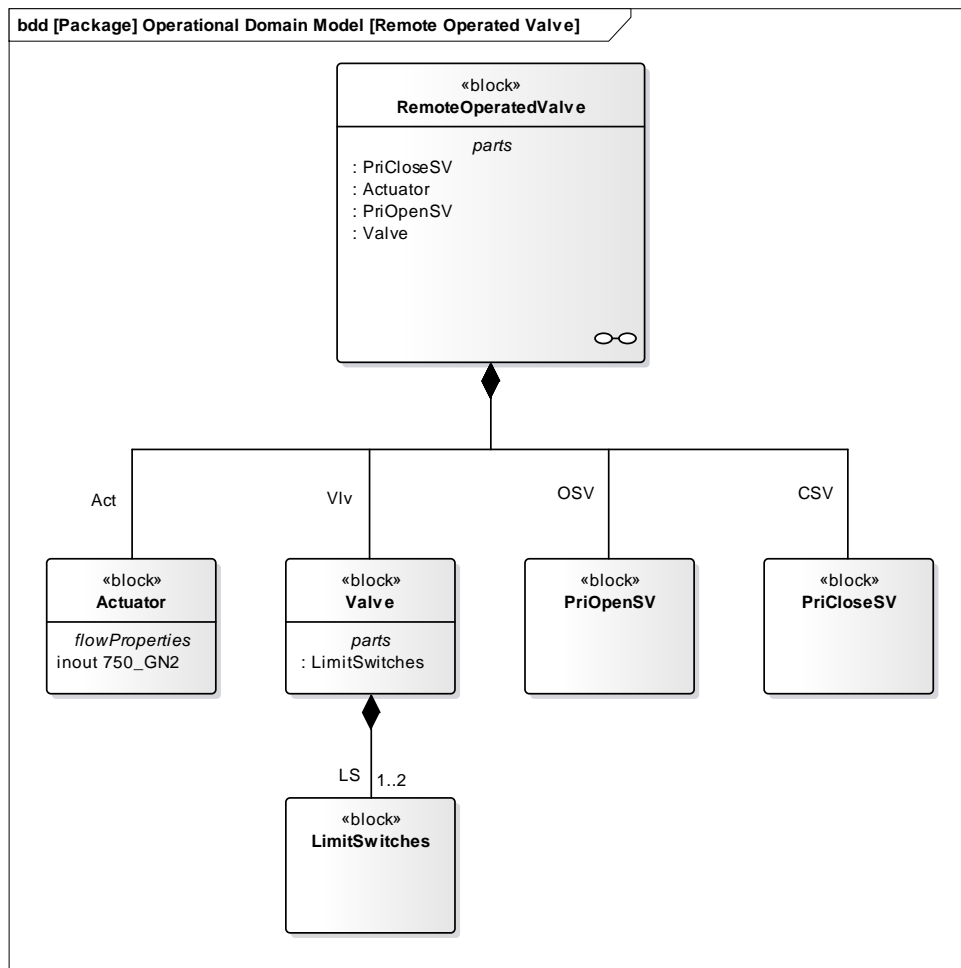


Figure 15 – BDD of Remote Valve with 1 or 2 Limit Switches

An association between indicators and faults they could potentially detect could be made.

However, the majority of the indicators will have some degree of detection for most fault modes which would result in a matrix that is mostly ‘filled-in.’ In addition, the process of assigning indicators to faults could result in leaving out measurements that may have some unique way of uncovering an issue. As this framework is intended to apply new technologies to many systems, implementing a method that predetermines which sensors are applicable to which faults can be

counterproductive. Subsequently, a method that readily identifies and categorizes all types of measurements will suffice.

For this case study, all stand-alone indicator components are assigned part numbers prefixed with “IND_,” and located in an Indicator package (embedded within a Component List package). A BDD was created to organize and show temperature indicators (reference Figure 16 - BDD Containing Temperature Indicators). To categorize the sensors, a package can be created for each type of indicator inclusive to that system. This method provides both a parts list of all indicators inclusive to the system as well as a means to quickly find that item within the diagrams so that the instrument’s functionality can also be determined.

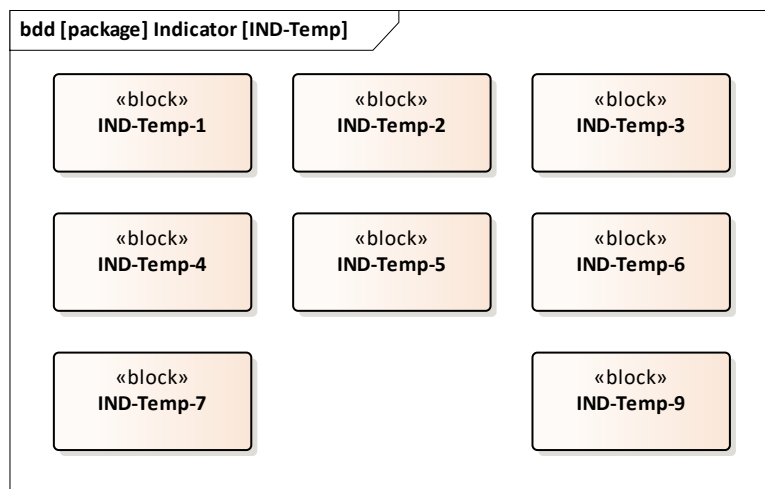


Figure 16 - BDD Containing Temperature Indicators

The parts list comprised of BDDs will show all the components used within a system, and if desired, how many of each of those components are used. The fault tree development process addresses indicator failures. This section considers the need for associating indicator

components to faults that they may detect as proposed within the initial framework. The goal here was to use this information to help bound the number of credible faults to address only those for which there is visibility. As stated above, linking indicators within an IBD to potential faults they may detect will likely result in an association for nearly all (if not all) faults-to-indicators. As an alternative, the association can be made at the BDD level, but this too could produce an outcome of the same result in using the IBDs. These indicator BDDs will be the means for identifying the system sensors and categorizing their attributes.

Identify and Categorize AD/FI Applications

The next step in this case study is to identify the AD/FI applications that should be considered for implementation. This pre-selection process should be driven primarily by the project's requirements. As there are numerous applications and techniques available, a significant amount of time can be spent researching all the possibilities which is not necessarily a practical approach. To reduce this effort, AD/FI classes are identified in which the technologies are similar. The onus for testing the various apps belongs to the users/stakeholders, though this framework assists with the selection process.

Requirements

One of the primary objectives of building a system model is having the ability to generate requirements, and then readily track implementation of those requirements through the system's life cycle. This capability is requisite within the systems engineering discipline, and SysML provides this functionality by providing a requirements diagram. This diagram works with text-

based requirements in that it can show relationships among the assorted requirements, other model elements and external objects (analysis, drawings, etc.) identified within the model. There are various notations available that provide traceability of these relationships. A requirement block (with associations) can also be dropped on other diagrams when it benefits the stakeholders to have this visibility and corresponding relationships.

Mission Statement

Requirements are typically generated at a higher level, and then broken down into lower level requirements as the design takes shape. For the LH2 system, a Mission Statement is created that describes what the system will accomplish. This statement is parsed to derive the initial upper level requirements for the system design. A requirements diagram is developed titled Mission Statement Requirements (reference Figure 17 – Mission Statement/Requirements Diagram). This diagram includes requirements ‘contained’ within the mission statement (those linked with a plus within a circle in the figure). These are the higher-level requirements that will be decomposed to system level ones that drive the design. In figure xx, a derived requirement necessitating AD/FD capabilities was added subordinate to the ‘Safe Operations’ requirement.

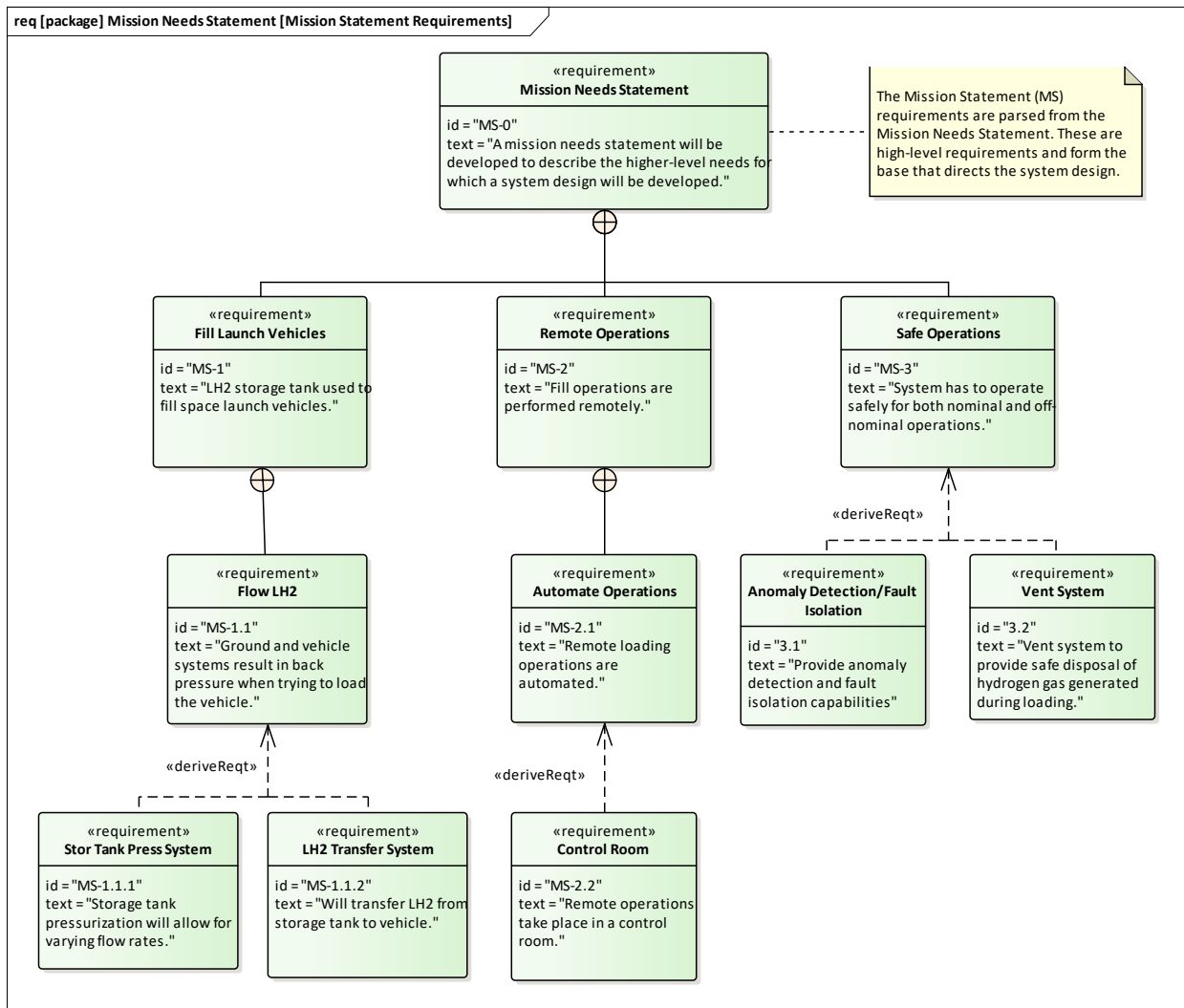


Figure 17 – Mission Statement/Requirements Diagram

Mission Requirements

With a requirement embedded in the Mission Needs Statement that dictates AD/FI capabilities, the mission requirements can now be developed. These are, for the most part, functional requirements in that they qualitatively define what is expected of the AD/FI applications. These requirements will drive the AD/FI technology to apply to the system. At this point, requirements

can be decomposed that are specific towards covering the desired subsystems. Note that requirements that focus on an explicit scenario may limit the available applications to consider. Since there are many technologies available including some that work indirectly with available data, the initial mission requirements should address overall AD/FI desires along with the operator's interface. A Mission Requirements (AD-FI) diagram was generated to collect and organize the AD/FI requirements (reference Figure 18 - Mission Requirements). It is at this level that the requirements will be used for selecting the potential AD/FI application(s) for consideration. After the selection process is completed, these requirements can be further decomposed to start showing implementation details for the system design.

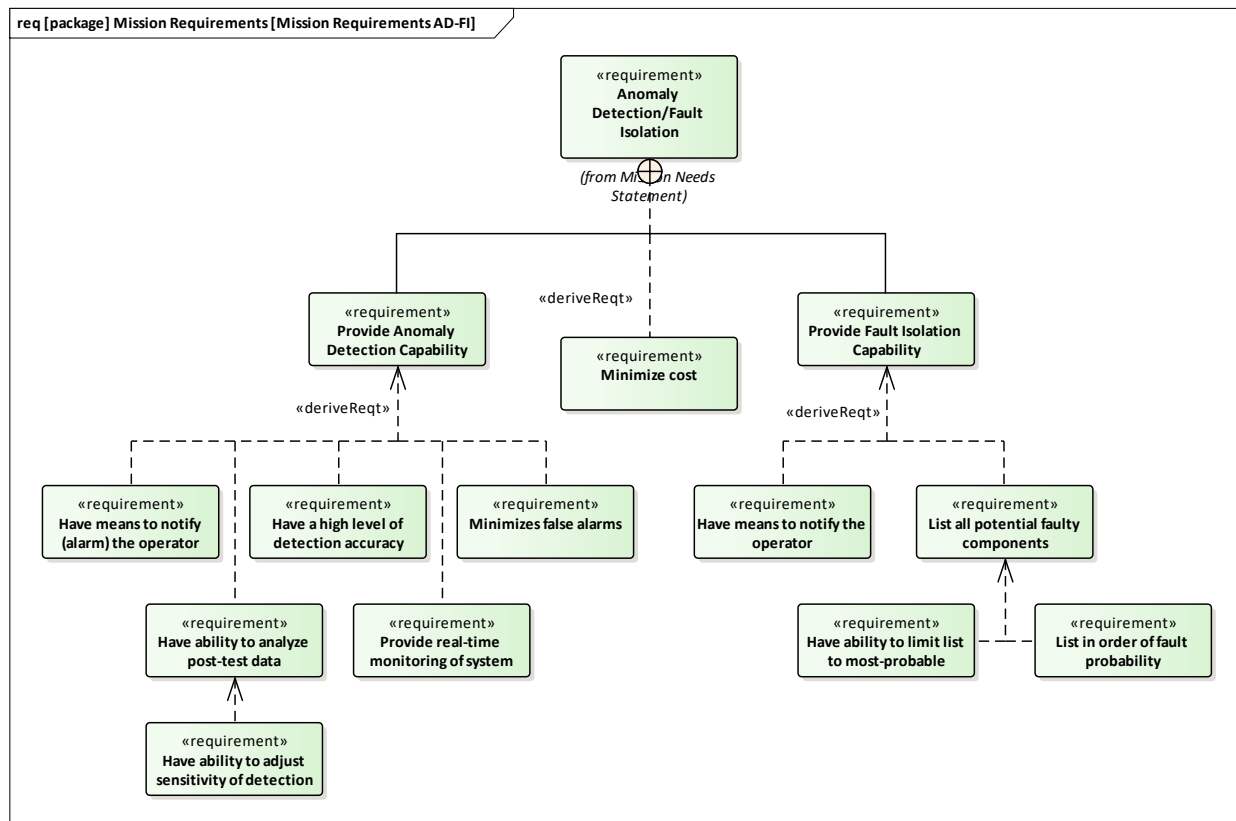


Figure 18 - Mission Requirements

A ‘minimize cost’ requirement was included with the mission requirements. As cost is always a major factor in system design, it is important to include cost factors in the prescreening process. A diagram is included that decomposes the cost requirement and provides additional detail as to how costs will be controlled (reference Figure 19 - Requirements AD-FI Diagram).

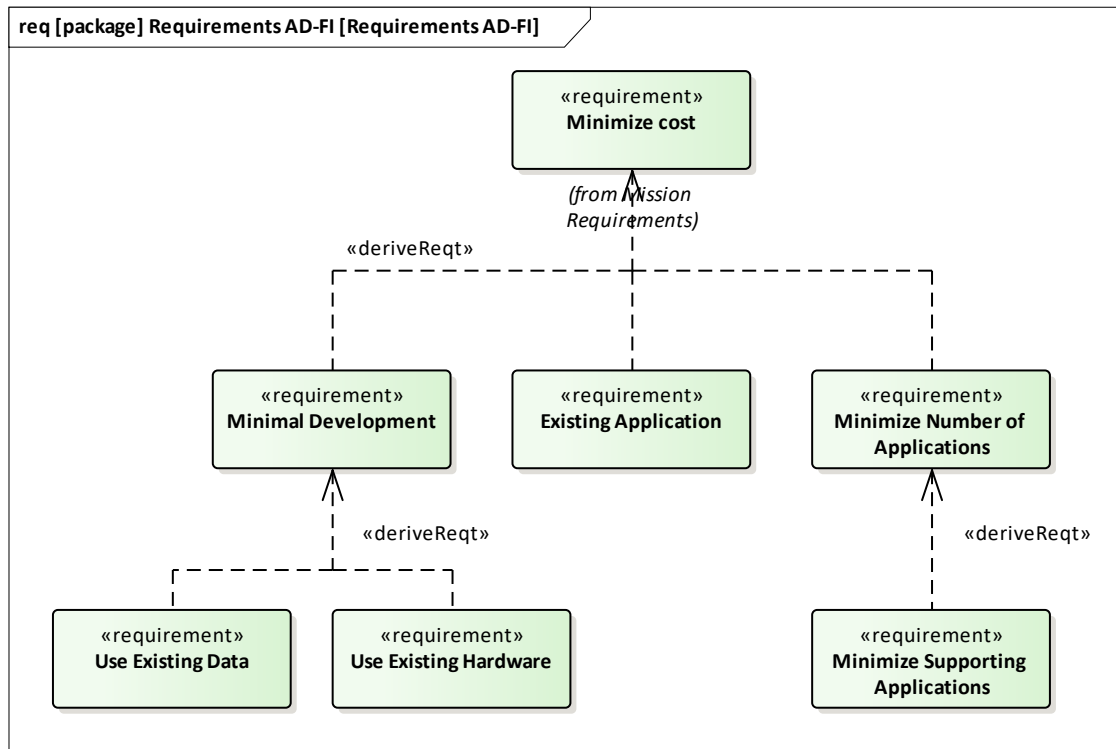


Figure 19 - Requirements AD-FI Diagram

Effectiveness Requirements

Effectiveness requirements provide a means to measure the system’s functional capabilities against the expectations of the design. Developing effectiveness AD/FI requirements can be quite challenging. Typically, a requirement is developed to meet a desired objective. As the design takes into consideration the operating environment, an expectation of how the system will

perform is rendered. Effectiveness requirements bring about ways to measure this performance. The difficulty posed by dealing with anomalies is that systems are not (purposely) designed to fail. Failures can occur for a variety of reasons to include component/material failure, operation/environment excursions, design flaws, operator error, etc. Some failures will have an immediate impact on system performance while others may slowly degrade before they reach a point that is detrimental to operations. How a failure propagates thru the system can differ based on the severity of the failure and the system configuration supporting the current process. Fault trees are used to identify various failure modes, but the same failure may present itself entirely different to the operators. For a complex system, it is unreasonable to wholly catalog the number of abnormal conditions that may arise and how the instrumentation will respond with certainty for all phases of operations. This means there are many ‘unknowns’ (which is, of course, the rationale for applying AD/FI technology). Subsequently, trying to measure the effectiveness in detecting this unknown can be challenging.

By identifying both functional and cost-related requirements, enough information is provided to select which classes of AD/FI applications should be pursued. These requirements remain high-level so as not to constrain the initial selection process too tightly. They are summarized below (reference Table 5 – Requirement Summary).

Table 5 – Requirement Summary

Req ID	Title	Description
AD-1	Provide Anomaly Detection Capability	Overall requirement to provide AD capability
AD-1.1	Have means to notify (alarm) the operator	Provide a means to notify operator of exceptions (via existing C&C architecture or stand-alone system available to operators)
AD-1.2	Have a high level of detection accuracy	Have ability to accurately detect issues above existing AD capabilities
AD-1.3	Minimizes false alarms	Minimizes nuisance-alarms that detract operators from system monitoring
AD-1.4	Have ability to analyze post-test data	Have ability to 'playback' data for post-test data reviews
AD-1.4.1	Have ability to adjust sensitivity of detection	Have ability to increase the sensitivity of post-test runs so all exceptions are addressed (both nominal/off-nominal)
AD-1.5	Provide real-time monitoring of system	Provide real-time AD monitoring during operations
FI-1	Provide Fault Isolation Capability	Overall requirement to provide FI capability
FI-1.1	Have means to notify the operator	Provide a means to notify operator of exceptions (via existing C&C architecture or stand-alone system available to operators)
FI-1.2	List all potential faulty components	When more than 1 potential source of an issue is possible, list all possibilities
FI-1.2.1	Have ability to limit list to most-probable	When listing all possibilities, have the ability to limit what is displayed (no scrolling pages)
FI-1.2.2	List in order of fault probability	When listing multiple issues, rank in order of probability and display in this order
ADFI-1	Minimize cost (AD/FI)	Overall requirement to minimize ADFI costs
ADFI-1.1	Minimal Development	Minimize overall development related to adding ADFI technology
ADFI-1.1.1	Use Existing Data	Use existing data already available to support applications (i.e. system empirical data, existing models, etc).
ADFI-1.1.2	Use Existing Hardware	Use existing system instrumentation and C&C architecture (mimimize system modifications)
ADFI-1.2	Existing Application	Use existing AD and/or FI applications either commercially available or mature in development process
ADFI-1.3	Minimize Number of Applications	Minimize the number of augmented applications to meet requirements
ADFI-1.3.1	Minimize Supporting Applications	Use only existing models or applications necessary to supplement ADFI applications

AD-FI Initial Selection

Classes of both AD and FI applications/technology have been previously identified. These classes attempt to group individual techniques utilizing common methodology. This framework

will first distinguish which classes have the potential to meet requirements, and then focus on advancing these. These classes include:

- Anomaly Detectors
 - Nearest Neighbor/Clustering Algorithms
 - Neural Network
 - Statistical/Parametric
 - One-Class Support Vector Machines
- Fault Isolators
 - Physics Model
 - Expert Systems
 - Fault Map Model

Anomaly Detection

For AD, there are several existing techniques used for detecting LH2 system problems. These include limit-setting, data plotting, software notifications and system video views. The use of cameras to scan and monitor the physical system provides very limited AD capability. However, they have occasionally detected vapor clouds generated by cryogenic leaks in areas that do not have leak detection instrumentation.

Limit-setting allows the operators to set alarms above and/or below analog measurements, as well as opposing states for discrete measurements. Limits can be set on all measurements and modified as required when the system transitions to different phases. This provides an overall

level of protection that goes well beyond a small subset of measurements that can be viewed at a given time by the operator. As the alarms are set just inside the operational (or design) specifications, they can provide notification prior to an exceedance for issues that do not immediately initiate/propagate to unacceptable levels. This remains the predominant method for detecting system anomalies. The application software for controlling the system has additional AD functionality. Most valves have a design specification that identifies a maximum amount of time a valve should cycle to its opposite state when operating nominally. The software monitors all valve cycles, and displays the valve timing. If this exceeds specified values, an alarm is generated.

Another existing AD method is the ability to plot system data real-time (this capability was not available in Shuttle until approximately half-way through the program). Operators could select related data and plot these indicators on a single display. This provided a graphical view of performance over time vs. a 'snapshot' view provided by system displays. These plots made it easy to spot data trends that started to diverge from 'nominal.' This proved to be valuable at times as some anomalies could be seen developing long before an alarm was triggered. This provides additional time for an operator to respond which is highly desirable in time-critical and hazardous operations.

AD Application Rankings

The AD applications under consideration are all data-driven detection techniques. This can be expected for a couple of reasons. First, this is an existing system with a mature design. So anomaly detection hardware (additional sensors) should already be embedded within the system. There is a cost component to consider as modifying the system can be costly. Since the existing sensor array met the original design requirements, adding additional sensors or detection hardware may not be considered unless a potential (and credible) failure mode is uncovered with severe consequences. For this case study, this is also reflected in a cost-related requirement that dictates no additional hardware. Since modifying the system is not typically a viable option, then the other option is to focus on the data that the system produces. This data is relatively cheap and readily available. Based on the literature, researches are finding unique ways to yield additional information from this data.

For remote systems, operators monitor displays to ensure indications remain within specified parameters. Not every measurement can be found on a display, and a very limited number of displays can be viewed concurrently. Subsequently, the operator's overall visibility of the system is very restricted. A limit-setting application compensates for this handicap by monitoring all measurements and notifying the operator of exceedances by means of an alarm. When data plotting (real-time) became available, operator recognition of deviations from nominal trends became more discernible. This provided a way to detect failures that were developing, but had not yet triggered an alarm. Again, the operator is constrained by a lack of visibility into the overall system. The data-driven models all work to distinguish between

nominal and off-nominal data. Having an application that can monitor a system and detect when data points transition from normal-to-anomaly extends this ability beyond the visible plots, just as the limit-setting application did for tolerance violations.

The AD functional requirements are intended to provide a higher-level detail as to what is desired without overly restricting the potential AD candidate field. There are a couple of requirements (can alarm, adds value over existing AD) that direct general necessities. There are also a couple of performance ones written qualitatively (detection accuracy, minimize false alarms). These may be later be decomposed to provide specific values that the application(s) will need to meet. The remainder of the requirements address specific functions that are sought after to support the LH2 system. These include:

- Provide real-time monitoring of system
 - The application must be able to function in a real-time environment. This is a tool used to augment AD and should be available when time-critical decisions are required. This can also be difficult to implement if the underlying algorithm has a high level of computational complexity. The LH2 system will have hundreds of indicators reporting at a sample rate of 10 per second
- Have ability to analyze post-test data
 - The application should not only work real-time, but also be able to playback test data. All vehicle loadings require a post-test data review looking for irregular events. This is accomplished while not time-critical. It should be noted that if the

post test data reviews show nominal operations, that data set then gets added to the training data used for subsequent operations

- Have ability to adjust sensitivity of detection
 - As the application is intended to support both real-time and post-test analysis, it should allow for adjusting the sensitivity. Being able to desensitize the application during real-time operations should limit the concern that a high number of false alarms will just distract the operators. Post-test reviews are very thorough as they account for every unexpected data point. A highly sensitive tool will better support this effort.
- Functions in multi-phasing and transient operations
 - Much of the literature uses examples of where a system is operating, perhaps with numerous dynamics involved (i.e. rocket engine runs), but it does so in a relatively stable operational and/or configuration setting. This is not the case for loading the vehicle with cryogenic fluids. There are multiple phases to load a vehicle and each requires a change in configuration. When using limit-settings and a transition is required, the applicable limits are inhibited. Once the transition is complete, the limits are again activated to the changed levels that support the new phase. The AD application needs to be versatile enough to accommodate the various phases. It is also desirable to monitor the system transients as limit-setting is inhibited at this time leaving only visual display monitoring for issue detection.

As budget is often a major factor in determining if a project goes forward, cost requirements are needed to ensure investment does not exceed the value-added. As this technology tends to be customized towards user with unique purposes, there are very few commercial applications readily available. Therefore, developing an application to meet requirements is a possibility. This also means estimating the project cost is much more difficult as there many unknown variables at this point in the life-cycle. Subsequently, the cost requirements are written such they minimize development, implementation and maintenance costs associated with a new application. This is an indirect way of controlling costs associated with the ambiguity surrounding new development.

A matrix was developed showing both requirements and AD classes (reference Table 6 - Requirement/AD-Class Matrix). This uses a simple scoring system to rank the AD classes. It does give partial credit if it is known that at least some of the requirement can be satisfied or if it is unknown if it can be satisfied at all. This is intended to lower the scores for those classes that require additional research without ruling them out completely by scoring 'does not meet.' The requirement/AD-class were scored as follows:

- 0=Does Not Meet Requirement
- 1=Partially Meets Requirement or Unknown
- 2=Meets Requirement

Table 6 - Requirement/AD-Class Matrix

Title				
	Nearest Neighbor/Clustering	Neural Network	Statistical/Parametric	One-Class SVMs
Provide Anomaly Detection Capability				
Have means to notify (alarm) the operator	2	2	2	2
Have a high level of detection accuracy	1	1	1	1
Minimizes false alarms	1	1	1	1
Have ability to analyze post-test data	2	2	2	2
Provide real-time monitoring of system	2	1	1	1
Have ability to adjust sensitivity of detection	2	2	2	2
Functions in multi-phasing and transient ops	1	1	1	1
Adds value above existing AD capabilities	1	1	1	1
Totals:	12	11	11	11
Minimize Cost (AD/FI)				
Minimal Development	2	1	1	1
Uses Existing Data	2	2	2	2
Uses Existing Hardware	2	2	2	2
Existing Application	2	1	1	1
Minimize Number of Applications	1	1	1	1
Minimize Supporting Applications	1	1	1	1
Totals:	10	8	8	8
Grand Totals:	22	19	19	19

Within the functional requirements, the applications scored very closely. As these are all data-driven techniques, this is not unexpected. The basic difference between the classes is how they determine data is normal or anomalous. Therefore, the ability to meet functional requirements should be similar. For the performance related requirements, it is unknown if they can be met so

these too are comparably scored. However, there is an exception within the Nearest-Neighbor/Clustering class. NASA has developed an AD application called Inductive Monitoring System (IMS) (“Inductive Monitoring System,” n.d.). They have licensed this technology on a non-exclusive basis in that it will be made available for use in NASA applications. As this application is being used in industry and supporting real-time applications, it will be assumed it can meet the real-time requirement. Although the literature is generally favorable in the detection capabilities for all the AD classes, these were specific cases that are not necessarily common to the LH2 system. Subsequently, requirements related to detection skills are rated ‘unknown.’

As IMS is an existing application ready to adapt to a given system, it will require less development as the core algorithm is already functional. This advantage gives the Nearest-Neighbor/Clustering class a favorable ranking for the cost related requirements. Therefore, a down-selection will be made at this time to pursue a Clustering application. However, as the ability of this application to function well within the LH2 system remains unknown, further analysis will be required.

FI Application Rankings

There are three classes identified related to FI. The first is a physics model. It is presumed that the model can mimic system operations, and if the system goes off-nominal, the model can be adapted to determine why. The second method involves an expert system. Such a system involves developing a knowledgebase, and then uses this information to follow a path that leads

to the root cause of the issue. It does this by incrementally performing a series of tests. The result of each test determines which test is next applied. The third class is termed a fault map model. This technique also develops a knowledgebase that maps the individual failures to the sensors used to detect that failure.

A complex system will have a very large number of potential failures. Often, a failure will propagate through the system further disrupting performance and possibly creating additional failures. As this propagation is dependent on system configuration, and this often changes for the LH2 system, it is not easily predictable. With a large number of potential failures, propagation unpredictability and a sensor array that provides a limited view of the system, isolating the issue to a specific component cannot always be accomplished during remote operations. Therefore, an FI application should be able to list multiple possible failure modes.

The LH2 system is composed of approximately 2,500 labeled components (this does not include piping, wiring, fittings structural components that also comprise the system). As shown during fault tree development, each component can have multiple failures. As each of the FI classes under consideration need this information to function, generating the applicable knowledgebase will be labor intense. In addition, relating these failures (and propagation) to the instrumentation that detects them requires system expert knowledge.

The FI functional requirements are intended to provide a higher-level detail as to what is desired without restricting options as with the AD functional requirements. These requirements include:

- Have means to notify the operator
 - The application needs to display results to the operator. This can be an automated report (triggered from an alarm) or a manual request by the operator. This application can run embedded within the existing C&C architecture, or from a standalone platform.
- List all potential faulty components
 - As it is unlikely that an FI application will be able to settle on a definitive problem, all suspect issues should be displayed to the operators.
- Have ability to limit list to most-probable
 - When displaying multiple issues, the operator should have the means to limit the displayed items. This protects against a failure (i.e. major C&C component) that can generate hundreds of alarms, and subsequently, hundreds of potential faults (limits scrolling pages).
- List in order of fault probability
 - When displaying multiple issues, order the list so that ‘most-likely’ items are shown first. This requires a means to assign a probability score to the possible fault causes.

The cost-related requirements used for AD ranking were developed to minimize development costs, and these same requirements apply to the FI applications. A matrix was generated to rank the FI classes using the same 0-2 scoring method (reference Table 7 - Requirement/IF-Class Matrix).

Table 7 - Requirement/IF-Class Matrix

Title	Phisic Model	Expert System	Fault Map Model
Provide Fault Isolation Capability			
Have means to notify the operator	2	2	2
List all potential faulty components	1	1	1
Have ability to limit list to most-probable	2	2	2
List in order of fault probability	1	1	2
Totals:	6	6	7
Minimize Cost (AD/FI)			
Minimal Development	1	1	1
Uses Existing Data	2	2	2
Uses Existing Hardware	2	2	2
Existing Application	1	1	2
Minimize Number of Applications	1	1	1
Minimize Supporting Applications	0	1	1
Totals:	7	8	9
Grand Totals:	13	14	16

The FI classes scored closely for the functional requirements with the only advantage given to Fault Map Models. There is an existing application called TEAMS-RT that has been tested in other NASA applications (“TEAMS-RT,” n.d.). This model does have the ability to rank the suspect faults detected. It is unknown if the other classes can include this feature.

For the cost-related requirements, a physics model by itself cannot provide fault isolation. As the LH2 system under study does not have an existing physics model, a supporting model must be generated to provide this capability. Therefore, this class did not meet the ‘minimize supporting applications’ requirement. The Fault Map Model class had a slight advantage over Expert Systems in that a commercial application is available (and tested in other NASA projects). As Fault Map Models ranked slightly higher in both functional and cost requirements, this class will be pursued further. It should be noted that there are still several unknowns associated with this class and additional analysis will be required.

Model the System

By selecting a model-based system engineering (MBSE) approach, the system will be modeled as part of the design process. SysML uses diagrams to convey graphically the various subsystems to users. The power of an MBSE approach is the underlying structure generated that links the various diagrams and model elements. Since diagram development with element associations are part of the MBSE model construction, no additional effort is required to provide information related to AD/FI applications. If augmenting an existing system that lacks the requisite documentation in a readily usable format, then at a minimum, the system structure should be modeled in SysML. For this case study, BDDs and IBDs will be developed for the three major LH2 subsystems identified in the Scope Overview section to include:

- Pressurization subsystem
- Transfer subsystem

- Vent subsystem

A BDD is used to describe physical characteristics of a system and uses a ‘block’ to represent those items. The system itself is a block which can be decomposed to subsystems, components and parts as deemed necessary by the stakeholders. The BDD shows the relationships in a hierarchical format. Since the BDDs identify all the components used within a system, this information is used to define system scope. While a BDD shows the composition of a block, an IBD differs in that it shows how the internal parts within a block are connected. This includes the flow of matter, energy and data among these parts. This information will assist with the generation of potential faults for the system.

LH2 Pressurization Subsystem

The LH2 pressurization system is used to pressurize the LH2 storage tank. This pressure, combined with different sized valves or variable position valves, controls the flow rate to the vehicle for the various loading phases. This system utilizes a main and auxiliary vaporizer (heat exchangers), a variable flow control valve (main) and control valves (main & auxiliary) to control the tank pressure. A BDD titled ‘LH2StorTankPressSys’ identifies the main components that comprise this subsystem (reference Figure 20 - BDD/LH2StorTankPressSys).

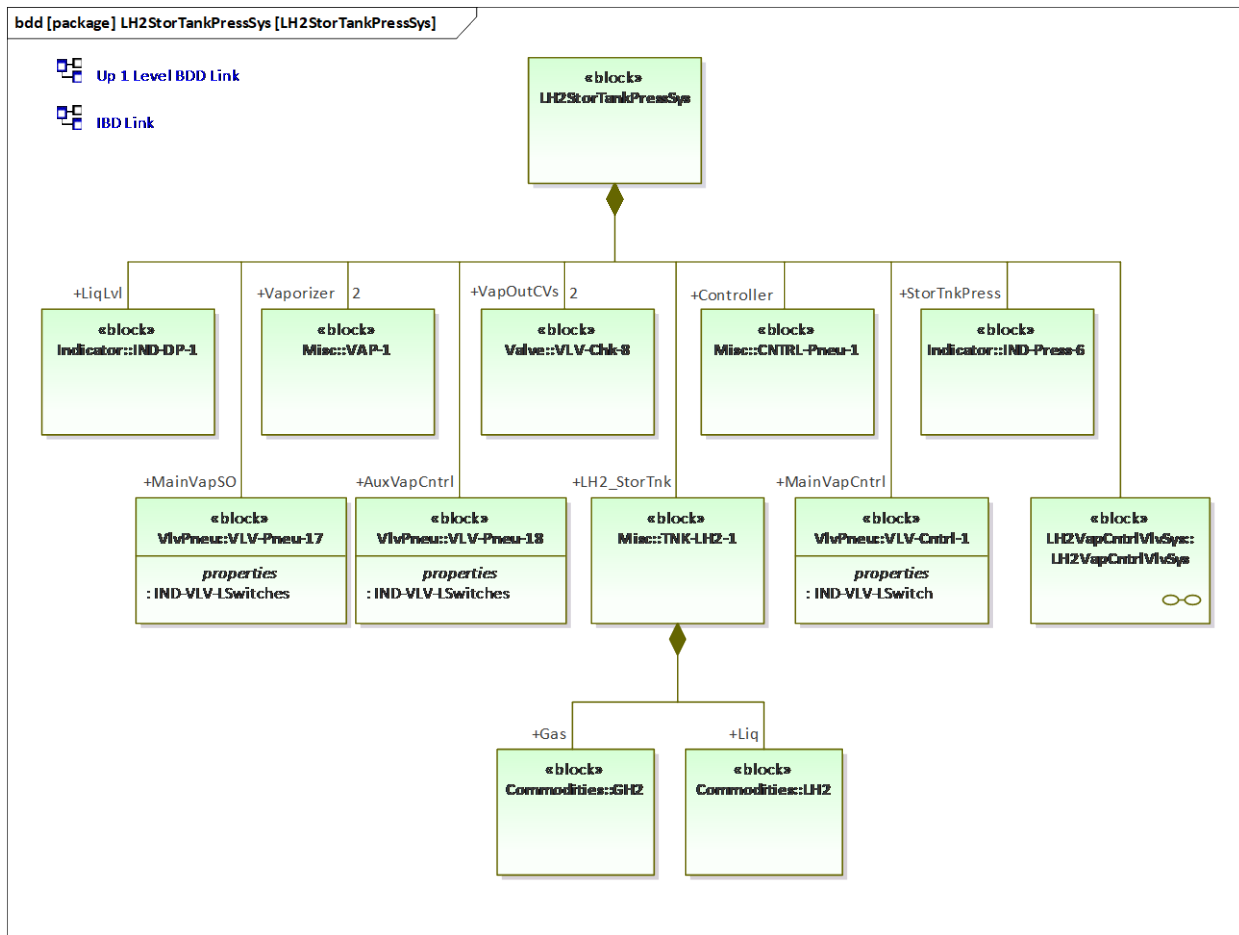


Figure 20 - BDD/LH2StorTankPressSys

As stated earlier, the BDD gives a view of the structure comprised of blocks in a hierarchy format. An IBD shows the internal connections of the block to include any media that is passed among those parts. The ‘LH2StorTankPressSys’ IBD shows how LH2 flows from the storage tank to the vaporizer, is converted to gaseous hydrogen (GH2) and returned to the tank (reference Figure 21 - IBD/LH2StorTankPressSys). As LH2 has an expansion ratio of approximately 833:1, a much larger volume is returned to the tank which drives the pressure upwards.

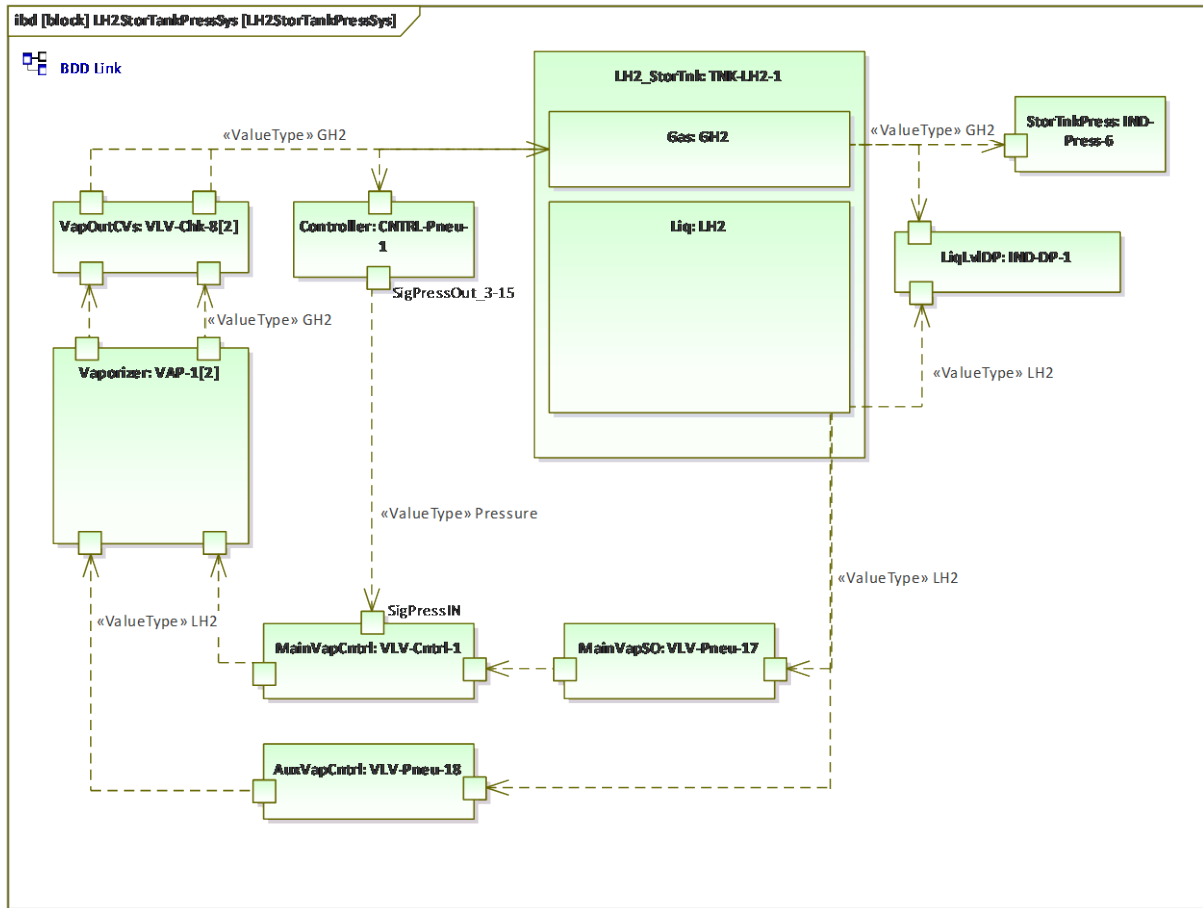


Figure 21 - IBD/LH2StorTankPressSys

The 'LH2StorTankPressSys' is a higher level view of this subsystem. During the modeling process, these higher-level diagrams will be further refined until adequate detail is represented that meets the design, user, and stakeholder groups. As an example, in the 'LH2StorTankPressSys' BDD, there is a block titled 'LH2VapCntrlVlvSys' which is decomposed further in a BDD of the same name (reference Figure 22 - BDD/LH2VapCntrlVlvSys).

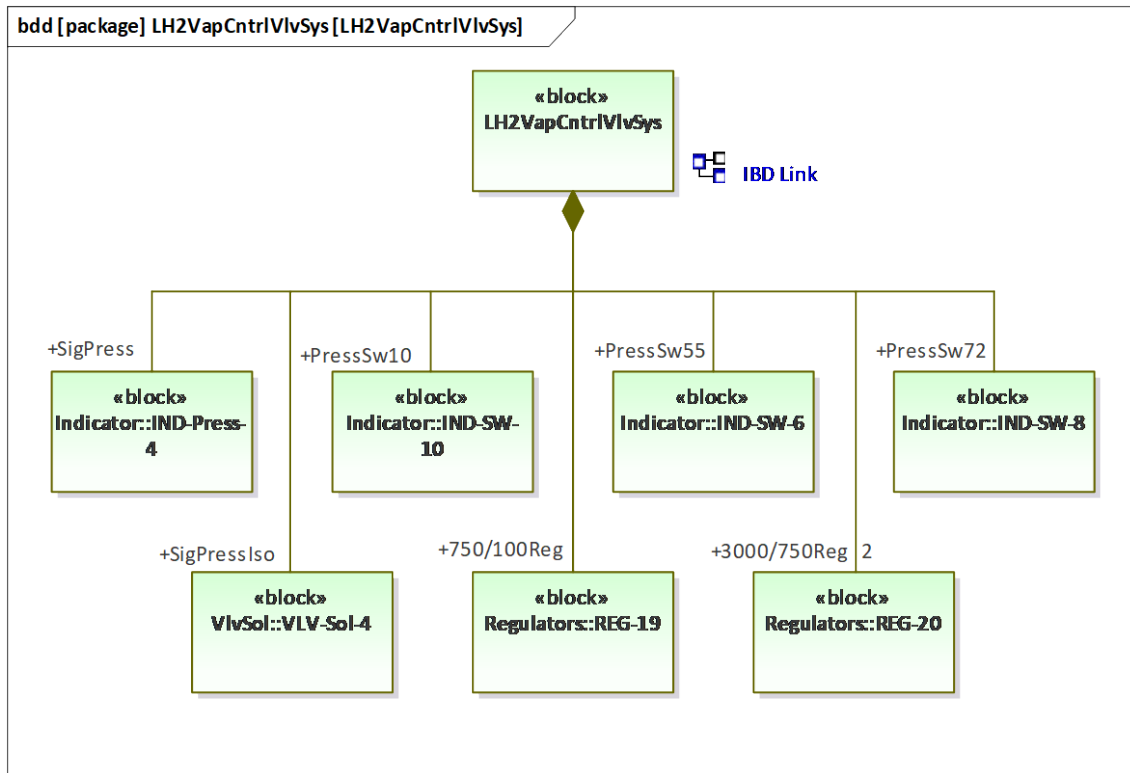


Figure 22 - BDD/LH2VapCntrlVlvSys

The corresponding IBD shows the interaction among these components. Note that parts that are not part of the corresponding BDD can still be included within the IBD when necessary to enhance the diagram's view (annotated "from *IBD*"). In addition, when the flow inputs and outputs (matter, energy and data) do not originate or terminate within the diagram, these are shown as 'ported' to the diagram's frame (reference Figure 23 - IBD/LH2VapCntrlVlvSys).

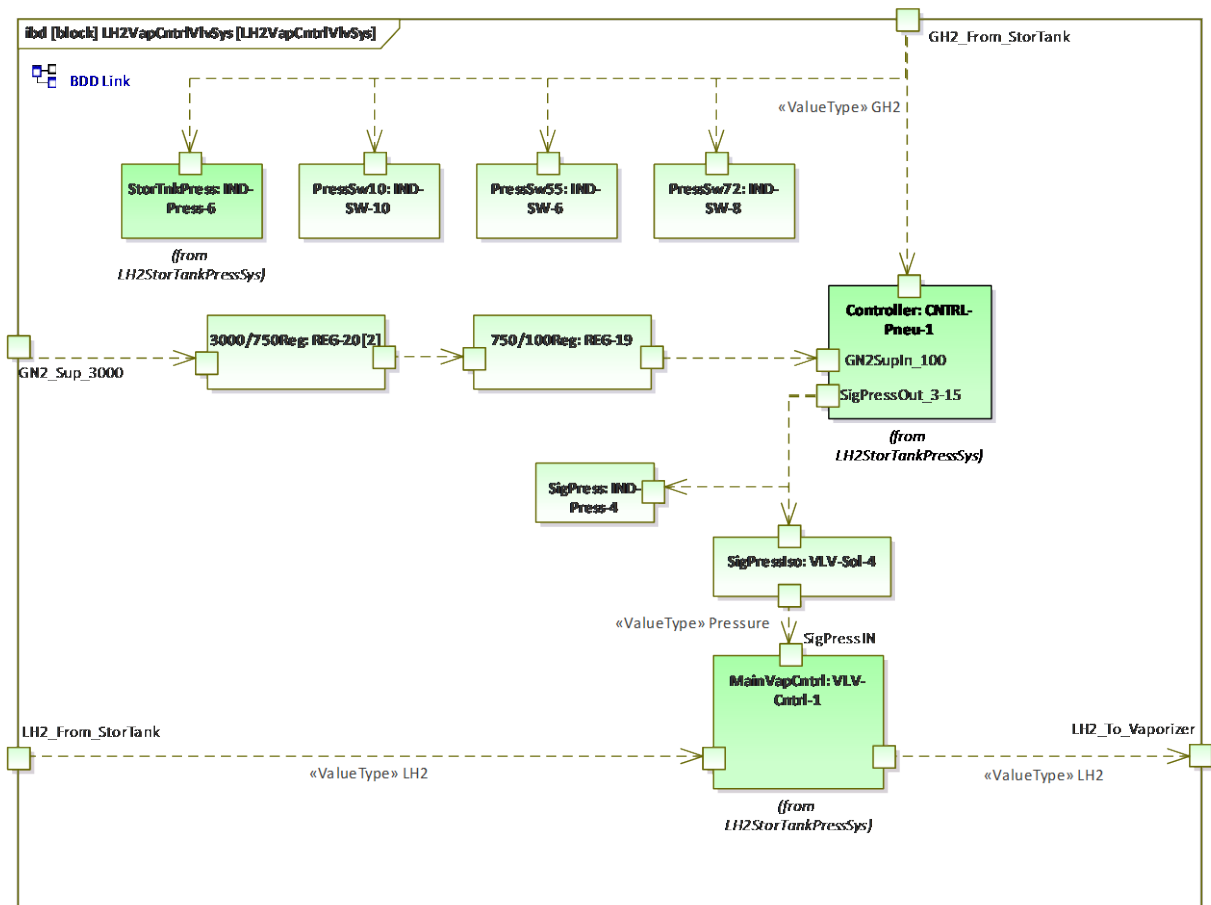


Figure 23 - IBD/LH2VapCntrlVlvSys

LH2 Transfer Subsystem

There are two valve control assemblies (ground systems) that control the LH2 flow to the launch vehicle. One of these valve skids is located at the Pad in the vicinity of the storage tank, while the other is on the mobile launcher platform just upstream of the Shuttle vehicle. A BDD showing both valve complexes was developed. Note that all the valves on these skids are included, though not all support LH2 transfer operations.

The focus of this IBD is to depict the transfer of LH2 from the storage tank to the launch vehicle. Therefore, some of the components listed in the BDD that do not directly support LH2 flow are not included in this IBD.

LH2 Vent Subsystem

As LH2 is a cryogenic liquid that is constantly boiling, anywhere you have the potential to trap liquid (i.e. storage/flight tanks, transfer line piping) must also have the ability to vent that part of the system to prevent over-pressurization. For the system under study, there are four primary generators of GH2 that require active control during loading operations to include:

1. The GH2 generated from the external tank while being filled.
2. A liquid bleed flow used to thermally condition the Shuttles' engines.
3. The LH2 storage tank (following pressurization).
4. The cross-country transfer lines that can be independently isolated.

The Pad LH2 valve skid has the ability to vent the storage tank as well as the transfer line piping between the Pad and MLP valve skids. The MLP LH2 valve skid controls venting of the transfer line between this valve skid and the vehicle. It also controls the bleed flow from the vehicle.

Model the AD/FI Applications

When the down-selection was made to determine which AD and FI classes would be pursued, it was noted that enough information was not available to determine if these classes could fully

meet the requirements. This section will model the selected AD and FI classes so this determination can be made.

Anomaly Detection Model

The AD class chosen was the Nearest Neighbor/Clustering method. This is a data driven application. Data is ingested into the model (referred to as training data) which provides a reference when later compared to test data. Data sources can be classified as supervised, semi-supervised or unsupervised. Supervised data means that the data set includes both normal and anomalous data that is ‘labelled.’ Labelled data is known to be normal or otherwise. Semi-supervised data can include both normal and anomalous, but only the normal data is labelled. Unsupervised data is not labelled (Goldstein & Uchida, 2016). For large datasets, it is typically impractical to label the data. As this is the case for the LH2 system, only unsupervised data will be utilized. This data comes from previous LH2 loadings that were deemed nominal following a post-test review. This is not sufficient to label the data, but it does increase the level of confidence that the data represents only nominal operations. When using unsupervised data, it is *assumed* to be normal and all exceptions anomalous. However, it needs to be noted that such datasets may include (undetected) abnormal data. In addition, divergence from nominal does not always reflect an anomalous condition. It could mean the training data does not include all variations that represent nominal operations.

The Nearest Neighbor and Clustering methods are two different techniques. These were combined into a single class as the most common approach to implementing both involves

determining the Euclidian distance of a test point to either its nearest neighbor(s), or to a central point within a cluster. This class ended up ranked the highest because there is an existing application that was originally developed by NASA and tested in various applications. This application utilizes the Clustering model. The main advantage to this is that the center of the clusters is calculated during the training phase, and that value is provided as a constant during testing. The Nearest-Neighbor needs to know the test point so it can seek out the neighbors. By shifting as much of the calculation process to the training-side of the model as possible, the computational complexity during testing is reduced enabling the application to run real-time.

During the down-select process, there were four requirements that were flagged ‘unknown’ requiring further analysis. The Clustering AD process will be modeled to determine if these requirements can be met. These requirements include:

- Have a high level of detection accuracy
- Minimizes false alarms
- Functions in multi-phasing and transient ops
- Adds value above existing AD capabilities

The K-Means model follows the methodology as described by the IMS developers (D. L. Iverson & Field, n.d.). The initial testing will focus on a single cluster as the objective at this point is to test the sensitivity of changes in test data. The training data is synthetic, but based on actual parameters used within the LH2 system. The data types are also varied as this would be common when developing the desired vectors for system monitoring. A vector would be composed of

measurements that are in some way related to one another (user defined). This is intended to capture system performance by monitoring sensor groups that are influenced by associated operations.

Sensitivity Analysis

The K-means algorithm makes the nominal/off-nominal determination of a data point based on its distance from the cluster's center. This application defines that center as the average between the high and low values for each vector element. Subsequently, the only information needed to support testing is the high and low values for each range within the vector. The training data (single cluster) uses only ten inputs. This data is centered on the nominal value it represents, though it was varied by $\pm 1\%$ via a random number function. So the total range of the data within this cluster does not exceed 2% (reference Table 8 - Training Data).

Table 8 - Training Data

	Press1	Press2	Press3	Temp1	Temp2	VlvPos(%)
Nom	100	750	3000	-423	85	100
Training Data	99.04	744.03	3004.98	-420.06	85.78	99.55
	100.48	756.60	2980.02	-425.34	84.23	100.70
	100.43	754.45	2985.01	-421.87	85.16	100.60
	100.06	755.54	2991.16	-421.39	85.75	100.46
	99.97	743.15	3013.36	-424.95	85.33	100.66
	100.41	755.18	3003.67	-422.32	85.33	99.45
	100.70	746.93	2971.44	-418.85	84.68	99.30
	99.21	743.52	3008.02	-425.55	85.82	99.81
Min	99.0	743.1	2971.4	-425.5	84.2	99.3
Max	100.7	756.6	3013.4	-418.9	85.8	100.7

The measurements chosen have quite a bit of disparity in the numerical values. To determine the distance a test point falls away from the cluster's center, a Euclidian metric is used. As this method determines a vector length, larger values will have a disproportional impact on this distance. Therefore, the data should be normalized before entered into the vector. There may be cases when it is desirable for the model to be more sensitive to critical indicators. These parameters should still be normalized, but can then be weighted to obtain an elevated (or muted) response. All data will be normalized to a 0-1 scale per equation 1.

$$X_{NORM} = \frac{X - X_{MIN}}{X_{MAX} - X_{MIN}} \quad (1)$$

The model works by finding the distance between two points: the cluster center and individual test data. If $a = (a_1, a_2, \dots, a_n)$ and $b = (b_1, b_2, \dots, b_n)$, the Euclidian distance (d) between points a and b is shown below (equation 2).

$$d(a, b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \dots + (a_n - b_n)^2} \quad (2)$$

The training data tracks the high and low values for the entire range of each parameter for a given cluster. As it uses the average of these two values, the normalized value for each element results in '0.5'. Equation 2 can be summarized as follows:

$$d(a, b) = \sqrt{\sum_{i=1}^n (0.5 - b_i)^2} \quad (3)$$

With the training data established for a single cluster, the test data can be formulated. This data will start nominally, and then be manipulated to simulate various anomalies. This initial testing will assist in determining the sensitivity of the model to detect deviations from normal. For demonstration purposes, the test data will be limited to a count of fifty. Each vector length is calculated and then plotted. When data is nominal, a baseline is formed. An upward deflection from this baseline is an indication that one or more of the monitored sensors is deviating from nominal. It is this visual cue that alerts the operator that the system has changed. An alarm can be established by setting a threshold above the baseline.

The initial data selected does not represent a related subsystem. This data was chosen to model the responsiveness of a sensor grouping that includes both extremes (nominal pressures from 100 to 3000 PSIG; temperatures from 85 to -423°F) and common values (pressure and valve position with a nominal value of 100). Initial values are from actual data and all fall within the training

data's min/max values, or are very close. This ensures the initial vector generated would be assigned to this cluster.

The first test will increment a single parameter (100 PSIG) in the test data. Typically, a 100 PSIG system would have an operating range of +/- 10% (90 – 110 PSIG). The low and high alarm limits would be set to those operational limits. A plot of this test is presented below (reference Figure 28 - Single Indicator Test). The first 10 samples represent a nominal baseline. This measurement is incremented by 1 PSIG at every 10th sample (to 5%). The min/max values for this indicator are 99.04 and 100.70 respectively (reference Figure 24 - Single Indicator Test). The first increment to 101 PSIG moved the parameter just outside its max-value, and the plot registered a slight increase from baseline. As data toggles around a value, the baseline will oscillate to reflect these slight variations. With each subsequent increment, the test data moved further from the cluster as exhibited with the upward trend from baseline. A 5% increase on a single variable resulted in a distance value that is more than triple the baseline.

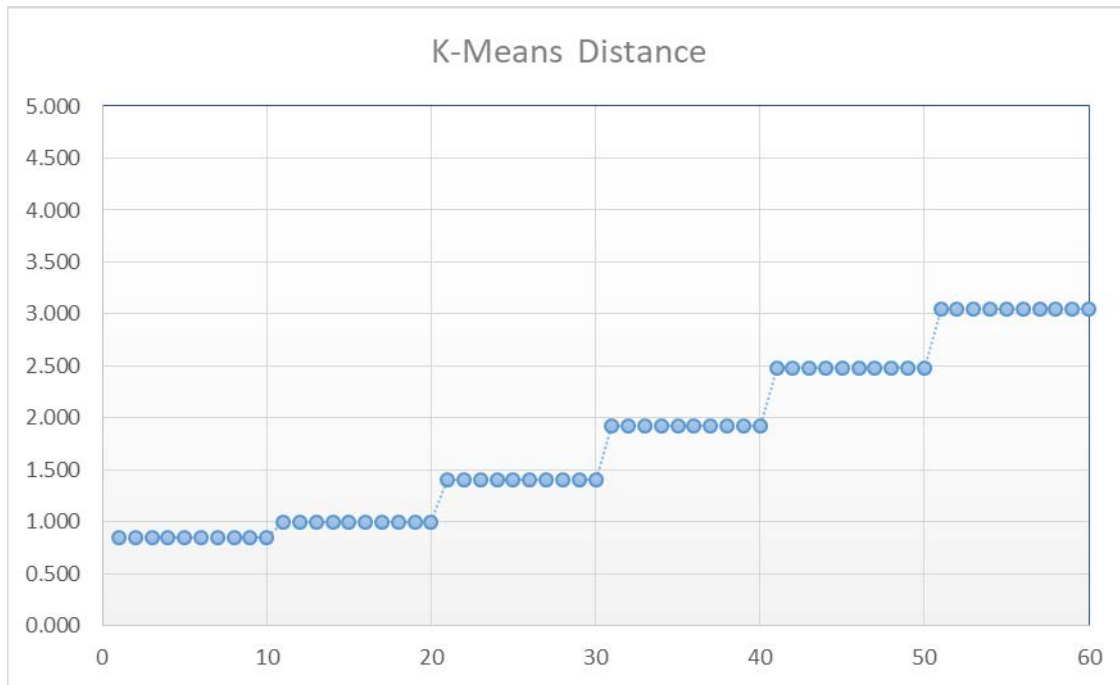


Figure 24 - Single Indicator Test

The next test involves incrementing multiple variables within the vector. These values were adjusted both upward and downward. The first 20 data points remain the same as in the previous test (baseline and 1% increase on 1 indicator). The subsequent samples represent a 1% increase on a different measurement every 10th sample (reference Figure 25 - Multiple Indicator Test). Similar to the previous test, a 1% increment has a slight impact to the baseline. In addition, the last variable incremented brings the baseline back down (Temp-2). This measurement's initial value is set just under the min-value for this parameter. So a 1% increase brings the distance closer to the cluster as depicted by the plot. An upward trend can be noted, but not necessarily significant enough to warrant additional evaluation.

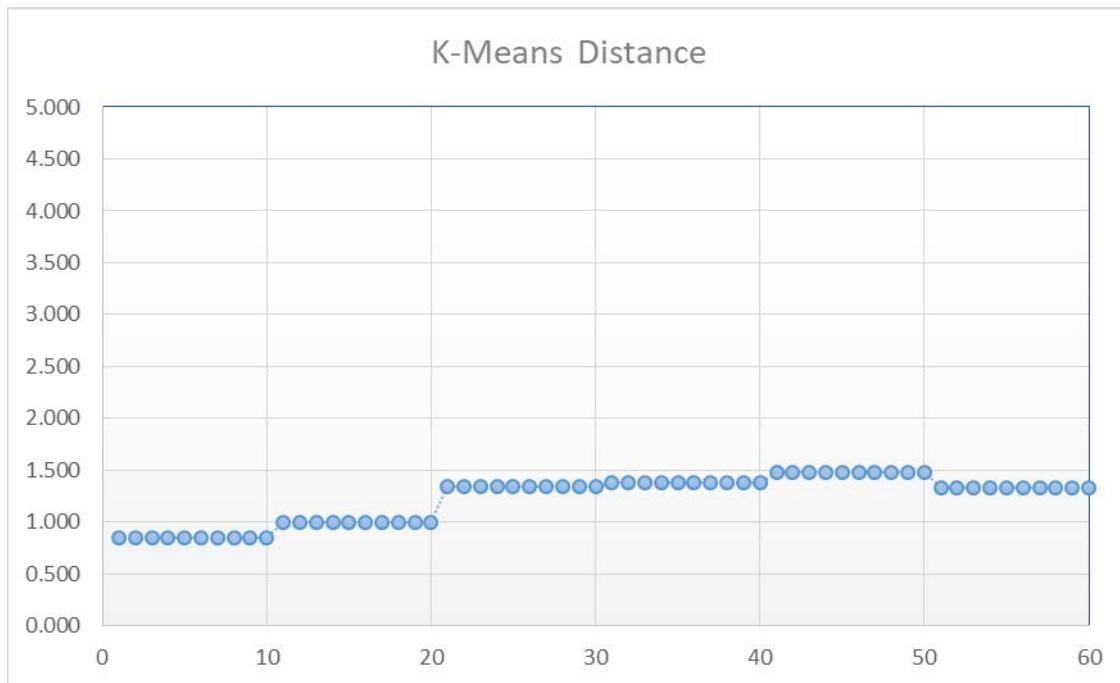


Figure 25 - Multiple Indicator Test (1%)

The next plot is the same five indicator test, but with 2% increments (reference Figure 26 - Multiple Indicator test (2%)). The upward trend is now more noteworthy. Had this been an interrelated grouping of measurements, this would have flagged the operator that the system was changing in a way not captured by the training data. It should also be noted that the ten-sample increments are intended to test sensitivity. If the change in the system did influence five of the six vector elements, the plot would have captured this over the transient range. For short duration transients, this could make the plotted shift more apparent. A longer duration trend may not be as obvious. This is rationale for including a threshold limit as subtle trend changes may not be easy to observe, but a declining gap between trend and threshold should be noticed.

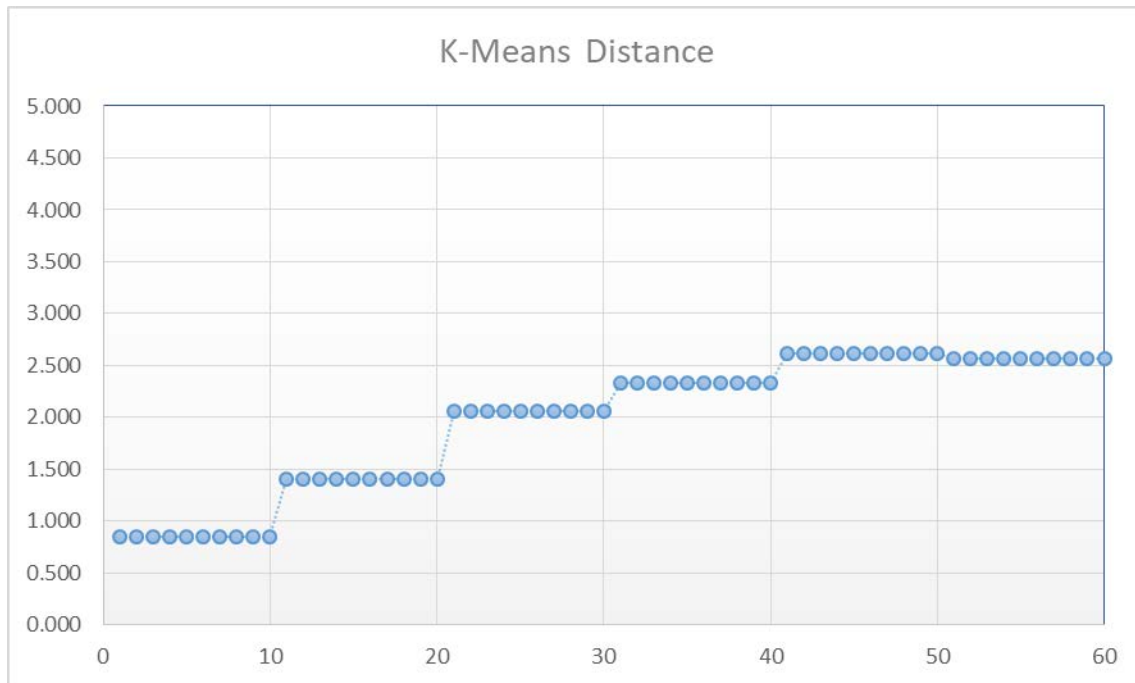


Figure 26 - Multiple Indicator test (2%)

The previous tests held the test data fixed so that the actual impact from a change in state was easily observed. However, data from dynamic systems often have noise associated with the measurement (frequent oscillations above/below a value). The previous tests will be repeated, but this time with noise being introduced into all the test variables. This is accomplished by generating a random number that is $\pm 1\%$ of the nominal value, much like what was done with the training data. The noise will be maintained while the data is being manipulated to simulate a change in the system. This 2% total range is aggressive for simulating noise as it is not often observed to this extent. It will also be applied to all the test variables, though many measurements are often stable indications. Consequently, this test case should represent a worse-case scenario. As the baseline is not so easily distinguished with noisy data, a full-run plot is displayed showing only the baseline from which subsequent test plots build upon (reference Figure 27 – Baseline Data (Noisy)).

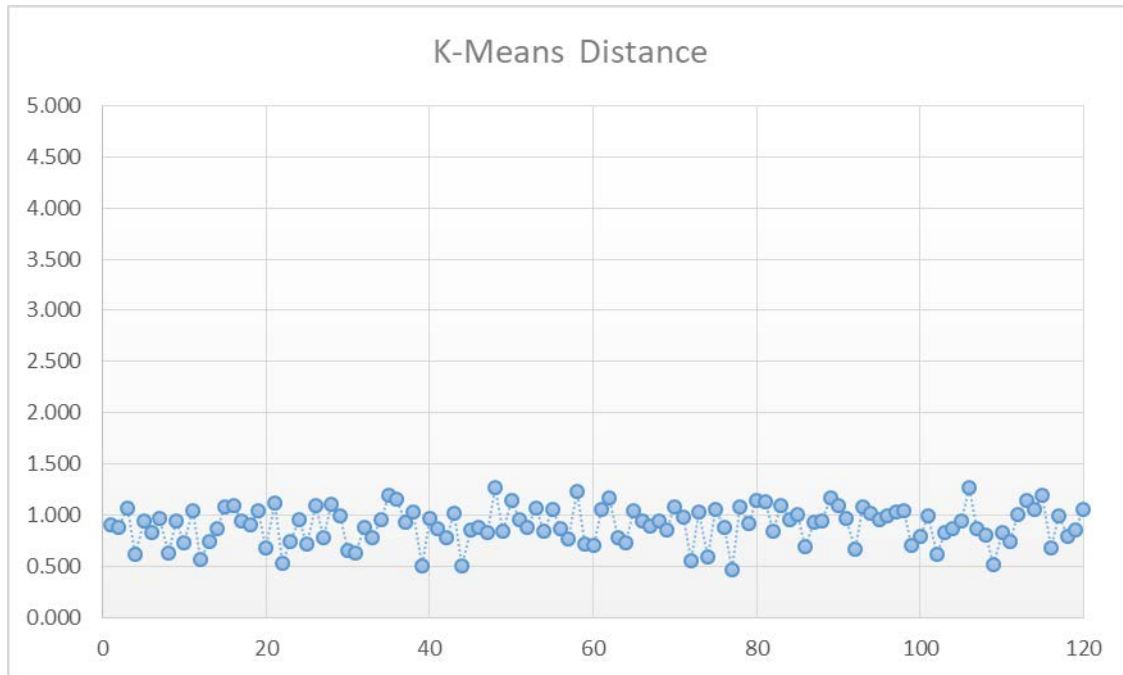


Figure 27 – Baseline Data (Noisy)

The first 20 samples now represent a nominal baseline and the measurement is incremented by 1 PSIG at every 20th sample to 5% (reference Figure 28 - Single Indicator Test (noisy)). There is no longer a discernable 'step' formation with the plot. However, there is an obvious upward trend that is approximately three times that of the baseline. This is consistent with what was observed with the non-noisy test.

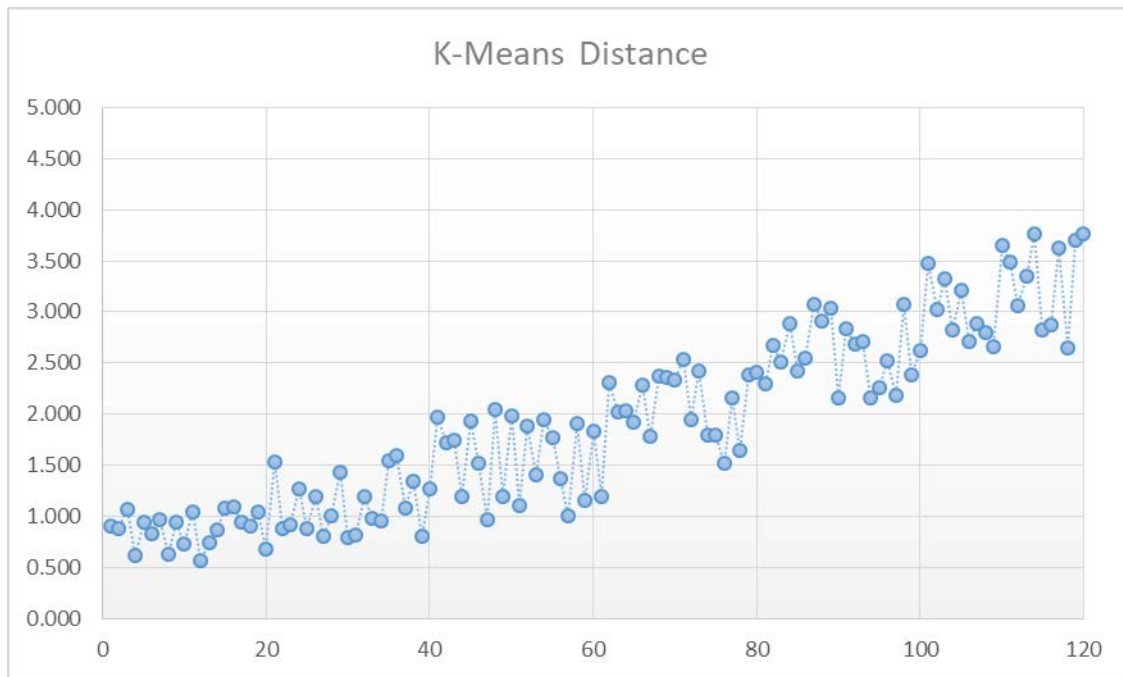


Figure 28 - Single Indicator Test (noisy)

The next two tests involve incrementing multiple variables within the vector. A new indicator was added to the plot every 20th sample. Each new value was increased by either 1% (reference Figure 29 - Multiple Indicator Test (1% - Noisy)), or 2% (reference Figure 30 - Multiple Indicator Test (2% - Noisy)) based on the nominal target, which is then modified to continue mimicking noise. An upward trend can be noted in both plots, though it is much more predominant with 2% adjustments.

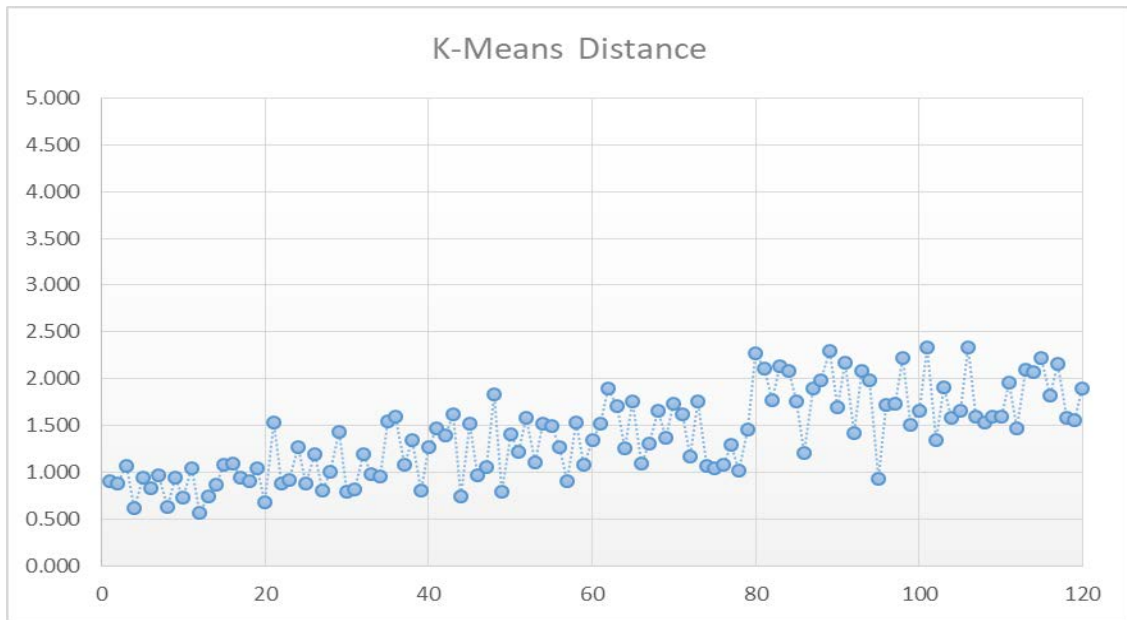


Figure 29 - Multiple Indicator Test (1% - Noisy)

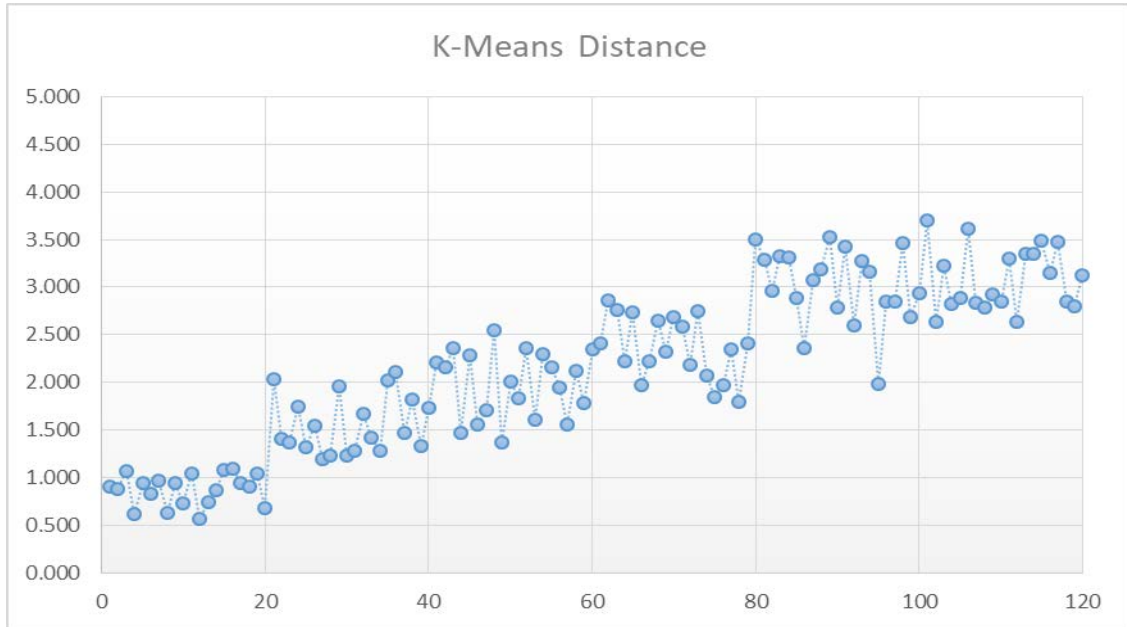


Figure 30 - Multiple Indicator Test (2% - Noisy)

Transition Analysis

One of the requirements levied is that the AD application function during the different operational phases and be able to capture transient conditions. As the limit settings for the indicators monitoring these transitions are inhibited between phases, much of the alarms are not available for these short durations. Hence, an AD application that can determine if the transition was nominal or not was deemed desirable.

For this testing, data was used from both STS-134 and STS-135 Shuttle missions (reference Figure 31 - STS-134 LH2 Loading (SF to FF) and Figure 32 - STS-135 LH2 Loading (SF to FF)). This data includes several pressure sensors and one temperature measurement. These indicators are related in that changes in the flow of LH2 will influence all four indicators. The data from the STS-134 mission will be used as the training data for the model. Both missions took place on Pad A. STS-134 used MLP-2 while STS-135 took place on MLP-3. The timeframe within the loading operations targets the transition from Slow Fill (approximately 900 GPM) to Fast Fill (approximately 8000 GPM).

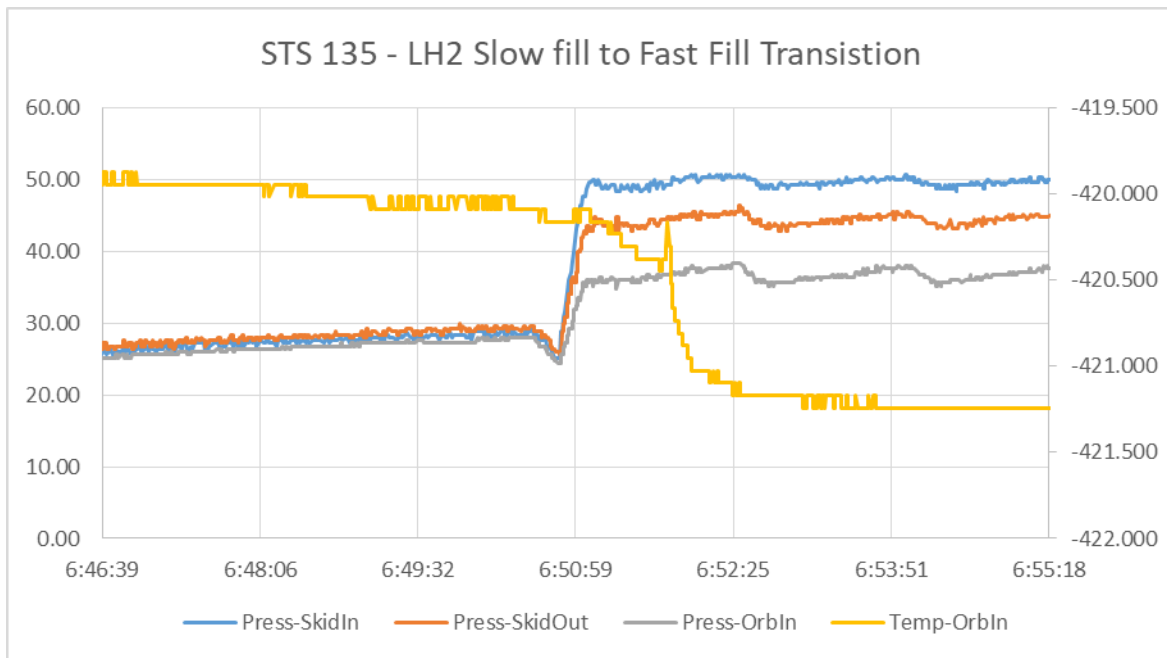


Figure 32 - STS-135 LH2 Loading (SF to FF)

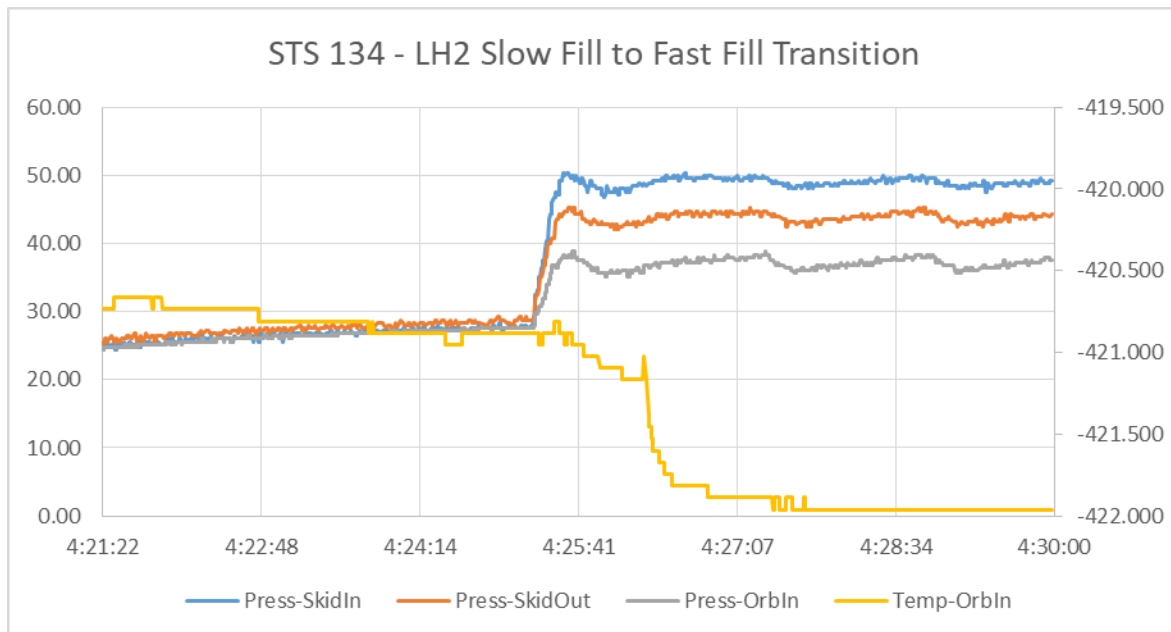


Figure 31 - STS-134 LH2 Loading (SF to FF)

LH2 Slow Fill involves a low flow rate, and all three pressures can be seen to be near-equal. As the LH2 external tank (ET) is loaded under pressure, the line pressures are gradually increasing. For STS-135, an ET vent cycle is observed just before transition as noted by the transfer line pressure drop that precedes an increase. Fast Fill is initiated by opening a larger fill valve increasing the flow rate from approximately 900 GPM to 8,000 GPM. The transfer line pressures all increase, but the impact of the various flow restrictions becomes obvious at the higher flow rate based on the disparity among the pressure measurements. The ET vent valve cycling becomes more frequent as depicted by the oscillations seen in the pressures.

The temperature is slowly decreasing towards that of LH2, and is within 2°F at the latter part of Slow Fill. When Fast Fill is initiated, the temperature starts to decline quicker as the transfer line pressure is elevated (flow rate increased). During this transition, it is typical to see a slight upward spike in temperature caused by liquid hitting the un-wetted surfaces which generates additional boil off of liquid. As the line completely fills with liquid, a rapid chill down to LH2 temperatures occurs.

Although the temperature decline profiles shown in figures 35 and 36 are similar, there is a shift between the two. These values are scaled the same, and the value ranges shown in the plots equate to approximately 1.5°F for each indicator. Therefore, this is high-granularity data (instrument range is -409°F to 427°F). Assuming the temperature of LH2 remains constant (at

pressures shown), a bias can be determined when the temperatures bottom-out. In this case, it is approximately 0.8°F. Although this is a low value, and an acceptable bias, it is significant when compared to the 1.5°F. The following plot includes the K-Distance with the STS-135 data (Figure 33 - STS-135 Flow and K-Distance).

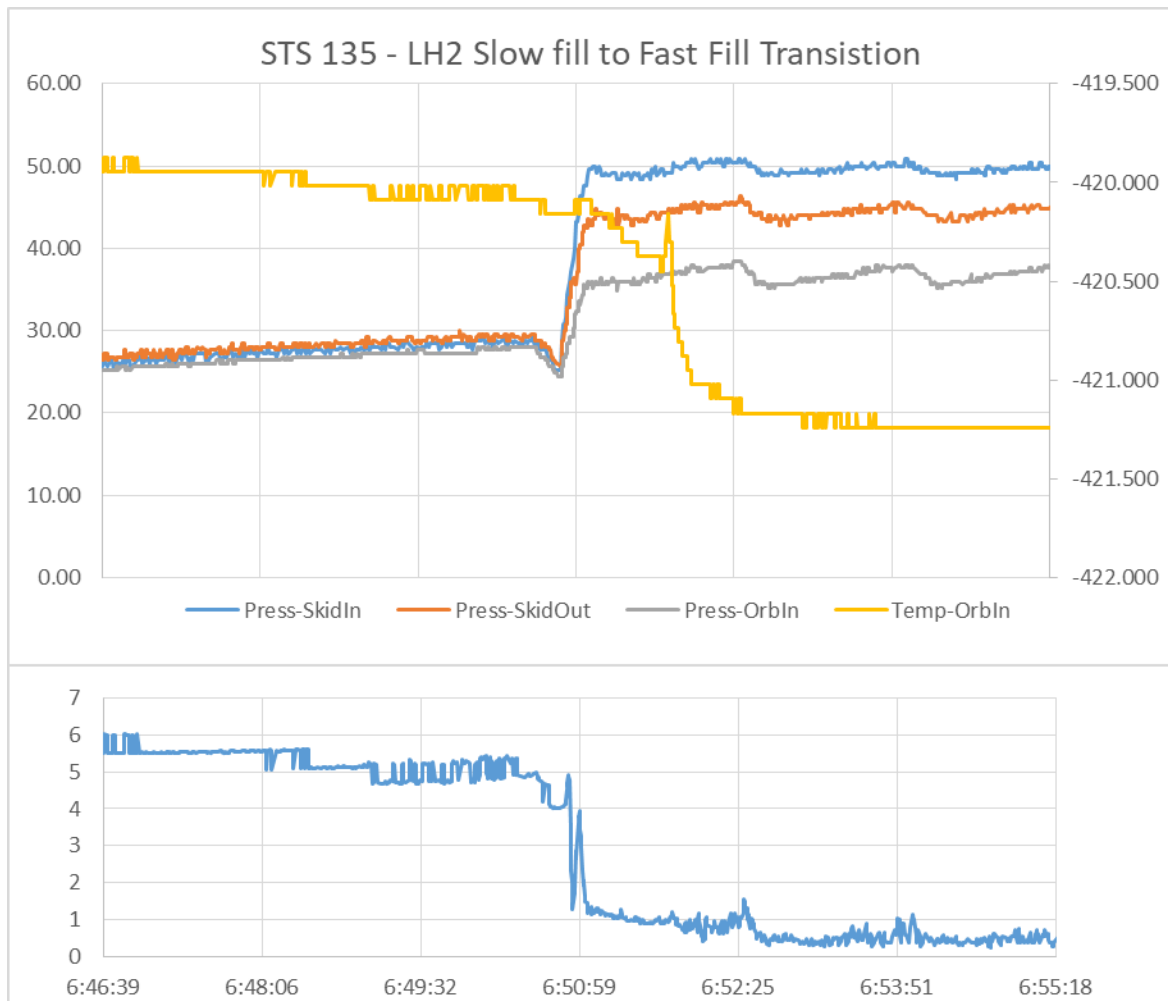


Figure 33 - STS-135 Flow and K-Distance

The K-Distance value shows the system is off-nominal (compared to the STS-134 data) during Slow Fill. One of the advantages of a K-Means methodology is it is easy to determine which variable(s) are responsible for the deviation. In the following table (reference Table 9 - Elements used for K-Distance), K-Distance is the variable plotted (against GMT). Also shown are the normalized values of the four variables that make up this distance calculation. Only the Orbiter Inlet temperature is (significantly) over one making this temperature the only outlier.

Table 9 - Elements used for K-Distance

GMT	K-Dist	P-SkidIn	P-SkidOut	P-OrbIn	T-OrbIn
6:46:39	5.504	0.60	0.67	0.40	6.00
6:46:39	5.502	0.60	0.50	0.40	6.00
6:46:40	5.509	0.20	0.50	0.40	6.00
6:46:40	6.008	0.20	0.50	0.40	6.50

The next test pulls the temperature out of the plot to see how K-Distance works with the three pressure values during transition (reference Figure 34 - Pressure and K-Distance). An increasing trend during Slow Fill shows some disparity compared to the training data, but at just over a value of 1.0, this is not considered problematic. A spike to just over 3.0 during transition is noted. K-Distance then returns to nominal, though it does track with the ET vent valve cycles.

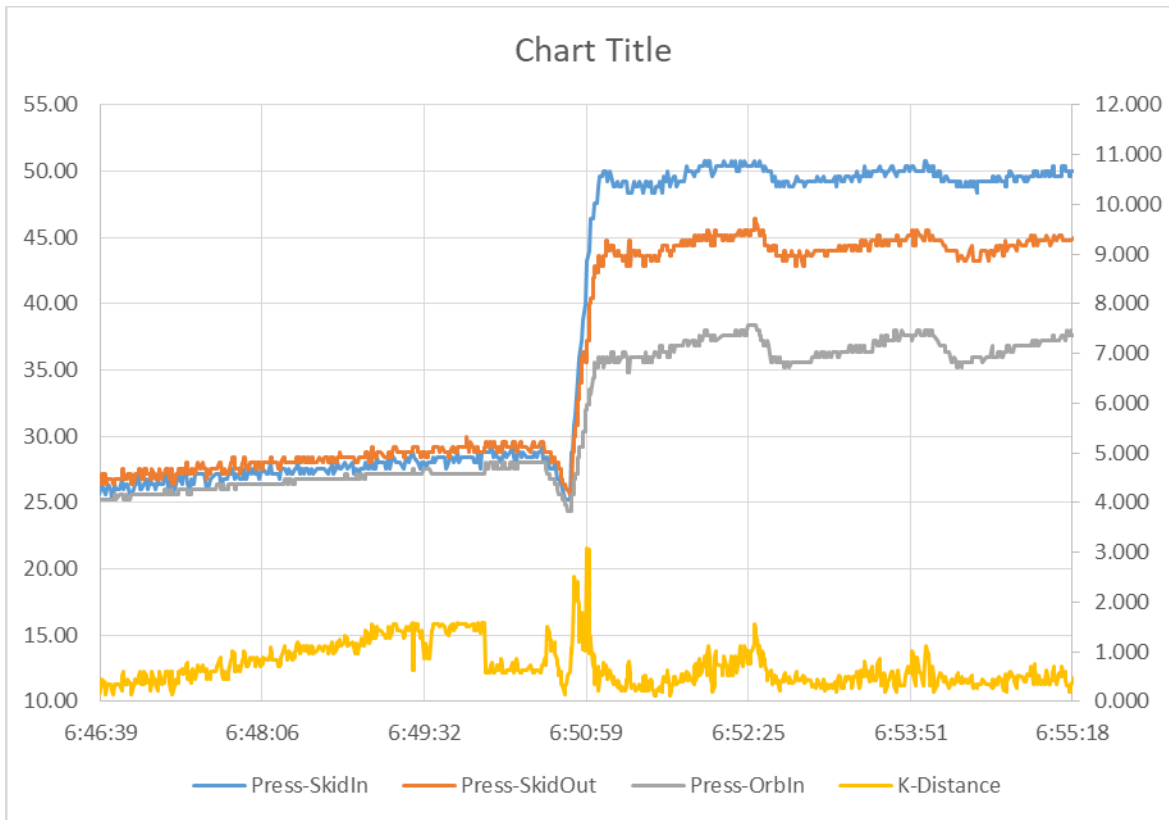


Figure 34 - Pressure and K-Distance

Finally, the same data is used but with a failure inserted. The Skid Inlet pressure was held at a fixed value during the ramp up to Fast Fill pressure (reference Figure 35 - Pressure and K-Distance with Failure). This failure mimics a loss of communication to the sensor, and subsequently, the value in the buffer does not change. When this type of failure occurs within the measurement's limit settings, an alarm is not produced. If it is close to nominal system values, it is not easily recognized on a system display view. This failure type can be distinguished on a plot, with dynamic data, as it is characterized by a 'flat line.' The K-Distance plots the same during Slow Fill and transition, but once the failure is inserted, it climbs to

approximately 4.0. In this case, the K-Means AD method did well to identify a failure occurring during transition.

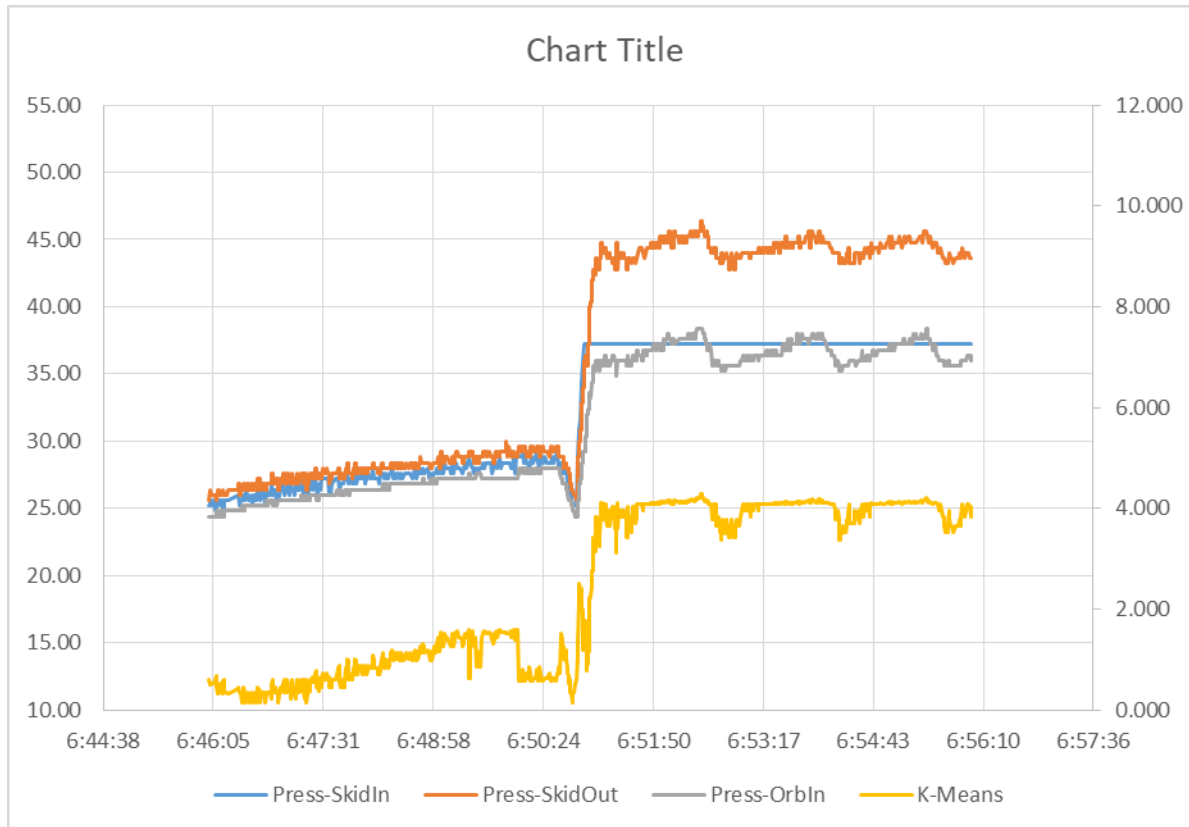


Figure 35 - Pressure and K-Distance with Failure

Fault Isolation Model

All three of the original FI classes require extensive support to implement this capability into a large system. A single-fault method would look at all components and the various failure modes that they can experience. If there is a need to consider multiple faults, then this effort can grow quite quickly. To start this mapping exercise, the focus will be on individual components. In addition, if a components' failure mode cannot be traced to a detecting indicator, then it is not

included. For example, filters have a failure mode in which the filter media fails allowing possible contaminants to pass. Such a failure would likely be undetectable by monitoring system pressure. However, debris may now pass the element and block flow thru a downstream orifice. A plugged orifice can be detected by a drop in downstream pressure. Subsequently, the model will (possibly) fault the orifice, which is behaving anomalously, though the root cause of the problem is the failed filter element.

Failure modes for components that cannot occur due to configuration are also omitted. This drops modes such as ‘internal leakage’ or ‘failure to close’ for valves that remain open during the entire operation. The initial pass at mapping the system will concentrate on single fault scenarios. It is projected that the model will be expanded to include multiple faults and all modes identified on the system fault tree. With this initial pass, modes such as ‘relief valve fails to open’ are excluded as it first requires a failure to over-pressurize the system.

As a fault map model was the selected class, the focus will now be on developing one that encompasses the LH2 system. The literature review was unable to uncover detail methods utilized by the (very few) vendors. So Excel will be employed to capture and organize the data needed to generate the mapping. It is anticipated that any application selected to implement this technology will have the capability to either work with Excel data (either by linking to the file or importing the desired data). The following outlines the fields to be populated:

- Component - As labeled by its unique identifier
- Description – Components nomenclature

- Failure Mode - As identified by the SysML model, and as required based on component configuration
- Sensor – The primary indicator used to detect the failure identified.
- State – The indicators' state that flagged the issue (high, low, erratic, nominal)
- Additional Sensor and corresponding State fields as required to characterize multiple sensors used in detection

With a fault map knowledgebase being created within Excel (using the fields defined above), the FI inputs can be tested while the knowledgebase is being produced. Simply adding a filter to each of the fields enables the operator to select an offending measurement to get a list of potential component faults. If there are multiple sensors available with applicable states, these too can be selected to refine the list, thus allowing the SME to test inputs in parallel with development. To model this effort, a knowledgebase for a purge panel feeding the LH2 transfer lines was generated. A high limit setting is exceeded for one of the indicators (PT057), which is then selected (reference Table 10 - Fault Map (PT057-High)).

Table 10 - Fault Map (PT057-High)

Comp	Description	Failure Mode	Sensor-1	State-1	Sensor-2	State-2	Sensor-3	State-3
HR051	3000/750 Hand Reg (Dome)	RegCreepsHigh	PT057	High	PT076	High		
HR051	3000/750 Hand Reg (Dome)	RegCreepsLow	PT057	Low	PT076	Low		
HR051	3000/750 Hand Reg (Dome)	RegFailsClosed	PT057	Low	PT076	Low		
HR051	3000/750 Hand Reg (Dome)	RegFailsOpen	PT057	High	PT076	High	PT047	Low
HR051	3000/750 Hand Reg (Dome)	RegLeaksExt	PT057	Low	PT076	Low		
HR051	3000/750 Hand Reg (Dome)	RegLeaksInt	PT057	High	PT076	High		
HR051	3000/750 Hand Reg (Dome)	RegPressUnstable	PT057	Erratic	PT076	Erratic		
DR054	3000/750 Dome Reg	RegCreepsHigh	PT057	High	PT076	High		
DR054	3000/750 Dome Reg	RegCreepsLow	PT057	Low	PT076	Low		
DR054	3000/750 Dome Reg	RegFailsClosed	PT057	Low	PT076	Low		
DR054	3000/750 Dome Reg	RegFailsOpen	PT057	High	PT076	High	PT047	Low
DR054	3000/750 Dome Reg	RegLeaksExt	PT057	Low	PT076	Low		
DR054	3000/750 Dome Reg	RegLeaksInt	PT057	High	PT076	High		
DR054	3000/750 Dome Reg	RegPressUnstable	PT057	Erratic	PT076	Erratic		
RV059	750 Relief Valve	VlvFailsOpen	PT057	Low	PT076	Low		
RV059	750 Relief Valve	VlvLeaksExt	PT057	Low	PT076	Low		
RV059	750 Relief Valve	VlvLeaksInt	PT057	Low	PT076	Low		
FL063	750 Filter	FilterPlugged	PT057	Low	PT076	Low		
FL063	750 Filter	FilterLeaksExt	PT057	Low	PT076	Low		
MV105	750 Purge-Leg Iso Valve	VlvLeaksExt	PT057	Low				
SV060	Vent Line Purge Sol Valve	VlvLeaksExt	PT057	Low				
SV065	Fill Line Purge S/O Sol Valve	VlvLeaksExt	PT057	Low				
SV038	Fill Line Purge Sol Valve	VlvLeaksExt	PT057	Low				
PT057	750 GHe Press	IndHigh	PT057	High	PT076	Nominal	PT5977	Nominal
PT057	750 GHe Press	IndLow	PT057	Low	PT076	Nominal	PT5977	Nominal
PT057	750 GHe Press	IndErratic	PT057	Erratic	PT076	Nominal	PT5977	Nominal

This results in a list of 26 component/failure combinations. Since a high limit was exceeded, the state of the corresponding sensor is then filtered on ‘high,’ reducing the potential failures to seven. A secondary indication (PT076) is also available for the remaining faults (this could be multiple indicators). The field filtering can continue, but with only 7 items listed, the logic can be carried out by observation (reference Table 11 - Fault Map (PT076 Check)). If a review of PT076 in the timeframe that the alarm was received showed no change in status, then a failure of PT076 is suspected, which can be further confirmed by looking to see if PT5977 remained nominal. For this example, we will assume PT076 also diverged high (it does not have to set off an alarm), leaving one additional indicator (PT047) that could have been influenced by the

change in system status. If this trended downwards, then the FI leads to two possible faults (vs. six if it remained nominal). These include the hand or dome regulators to have failed open.

Table 11 - Fault Map (PT076 Check)

Comp	Description	Failure Mode	Sensor-1	State-1	Sensor-2	State-2	Sensor-3	State-3
HR051	3000/750 Hand Reg (Dome)	RegCreepsHigh	PT057	High	PT076	High		
HR051	3000/750 Hand Reg (Dome)	RegFailsOpen	PT057	High	PT076	High	PT047	Low
HR051	3000/750 Hand Reg (Dome)	RegLeaksInt	PT057	High	PT076	High		
DR054	3000/750 Dome Reg	RegCreepsHigh	PT057	High	PT076	High		
DR054	3000/750 Dome Reg	RegFailsOpen	PT057	High	PT076	High	PT047	Low
DR054	3000/750 Dome Reg	RegLeaksInt	PT057	High	PT076	High		
PT057	750 GHe Press	IndHigh	PT057	High	PT076	Nominal	PT5977	Nominal

Assuming PT047 did drop, this implies either the hand or dome regulator failed open. Hand and dome regulators work in parallel. Hand regulators provide finite control in manually setting an operational pressure. The trade-off is that this manual control capability results in a component used in low flow applications. Dome regulators have the capacity for high-flow, but cannot be manually adjusted. Pressure is set by applying the desired pressure into the top (dome) of the regulator using a hand regulator.

The failure logic for this scenario is described as follows. When a regulator fails open, it is no longer able to control pressure, and the downstream side of the regulator is exposed to the upstream pressures (3,000 vs. 750 PSIG). If the hand regulator fails, it applies 3,000 PSIG to the dome regulator which then opens fully. This is the same outcome if the dome regulator fails open. In both cases, the 750 PSIG system is now exposed to 3,000 PSIG, though this leg includes a relief valve to protect against over-pressurization. There is another pressure reduction in this leg (750 to 80 PSIG) via a hand regulator only. If the upstream pressure to this regulator

spikes to relief pressure (880 PSIG), then an upward deflection would be expected on the 80 PSIG leg, monitored by PT076. Since the dome regulator is capable of high flow, combined with the high upstream pressure, it is likely flowing through the relief valve (which is sized to handle the maximum flow). Therefore, a drop in upstream pressure is expected (PT047) as the system struggles to maintain 3,000 PSIG with off-nominally high flow.

Perform Trade Studies

When the preliminary framework was developed, it was envisioned that after down-selecting to classes that best met requirements, there would be many options to choose from. This did not turn out to be the case. The existing limit-setting, plot capabilities and software controls employed in industry (and current KSC launch systems) are sufficient for most applications. In addition, the Programmable Logic Controllers (PLC's) that have become common in remote-operated control systems are embedded with health and status capabilities specific to C&C functions. Subsequently, proficient AD/FI expertise supporting control systems is included with the purchase of this hardware. There is an abundance of techniques found within the literature, but most have not advance beyond the conceptual phase. This does not diminish the need for this technology, but it does limit the commercial options. This trade study will assess using an available application or developing one internally. Both are viable options.

Anomaly Detection Application

A commercial application (IMS) exists that implements a K-Means method of AD. This application can run both real-time and playback historical data. It allows the user to select the

vector elements (variables) for inclusion and cluster size when inputting the training data. When running, the user can assign a threshold over the k-distance to be used as an alarm. It also has the ability to show how much each measurement contributes to the k-distance to readily determine which indicators are off-nominal.

The k-means methodology was modeled using Excel. However, this application cannot provide real-time system monitoring. It can provide playback of test data, but dataset size may overcome Excel's limitations requiring both training and test data be segmented. The clusters for the model were developed manually, but this could be automated using an Excel macro. The other functions provided by IMS (element selection, cluster radius, thresholds, variable contribution) can all be mirrored in Excel.

The AD model testing also resolved the 'unknown' requirements as follows:

- Have a high level of detection accuracy
 - The model illustrated a sensitivity to change for both static and dynamic data when compared to nominal training data. The training data is unsupervised, so there could be anomalous information contained within. A diverging k-distance shows a deviation from the training data which does not always reflect an off-nominal condition.
- Minimizes false alarms
 - When 'zoomed in' on a high-resolution temperature indicator, testing did show a significant deviation in k-distance. After a reviewing the plot, it was determined

that an acceptable bias existed between the instruments on from two different MLPs.

- These types of issues can be uncovered when testing historical data, and there are several actions that can be taken to mitigate the issue as follows:
 - Training data can be matched to the launch elements used (i.e. use Pad A and MLP-2 training data for launches using this combination). This ensures the same sensors are being compared.
 - Add multiple launches to the training dataset. This will further minimize the false alarms by capturing system variations within the training dataset.
 - Each element within the vector can be weighted to amplify or mute the impact on k-distance. The measurement tested is critical, so it would likely not be muted. However, this is an option for other non-critical sensors.
- Functions in a multi-phasing/transient ops
 - Testing (Slow Fill to Fast Fill) demonstrated that transient operations can be captured.
- Adds value above existing AD capabilities
 - The sensitivity analysis showed that both single and multiple indicator divergences can be detected well before a limit exceedance is triggered. This is a very strong capability.
- Minimizes number of applications and/or supporting applications

- This adds only one new application and requires no supporting applications.

Fault Isolation Application

It should be noted the LH2 system has local gages for viewing the system status which are located throughout the system. The fault mapping concentrates on a system configuration that supports launch vehicle loadings as these are the time-critical and hazardous operations that take place in a control room. The remote sensors available are a smaller subset of what can be accessed locally. In one case, a storage area panel that provides pneumatic pressure to all the remote operated valves is set up manually to support these operations. There are local gages used to reflect the supply pressure and for setting up two pressure reductions, including primary and secondary legs for the actuation pressure. However, the only remote sensor on this panel is a pressure switch that is sensing only the actuation pressure leg. In this case, all the components that can impact this pressure, and have a failure mode that results in a loss of pressure, will be linked to the switch. Failures that increase pressure above nominal will go undetected. Consequently, should this switch unexpectedly drop out, a large list of potential faults will be generated. With only a discrete measurement providing notification, the information needed to refine this list is just not available. Should the pressure dropout be real (and not a switch failure), and pressure continues to drop, any actuated valves will soon change state confirming the loss of pneumatic control.

This example is being presented to highlight that an FI application will be limited to what data is provided by the sensor array (applicable for the physics, expert systems and fault map FI

classes). Often, the fault cannot be isolated until the local hardware can be accessed to view the local instruments and/or take additional troubleshooting steps. Therefore, expectations need to be tempered for FI performance. During the analysis and development of an FI application, system design shortcomings may be uncovered. These can be addressed to determine if a system modification is warranted. Therefore, developing an FI knowledgebase also provides an indirect system review which is also beneficial.

A commercial application known as TEAMS (described in literature review) is available for fault mapping, and has been previously tested on a cryogenic test-bed at KSC. This application can support real-time monitoring and includes a means to encompass system configuration (via switches). There was only over-view information discovered as to how this application works, so it was not modeled.

A component-sensor-fault related knowledgebase was developed using Excel. The fields selected were those that are projected to be included should TEAMS (or another fault map application) be selected. The reason for developing an internal knowledgebase is this need amounts to an organization of data issue, and a means to present data in a timely manner with minimal input. It was presumed that Excel could accomplish this task, so a model was developed to test this functionality. Should an existing application be chosen, the effort to create the knowledgebase within Excel will likely not be unproductive as it is anticipated this data can be ported to other applications.

The FI model testing also resolved the ‘unknown’ requirements as follows:

- Lists all potential faulty components
 - This requirement is partially met (by design). Per the filter/orifice example described previously, FI will list all components it can link to the indicator(s) in question. This may not include the root component responsible for the failure as that failure mode may not be detectable with the current sensor array
- Minimal development
 - This requirement is partially met. There is minimal development related to getting an application active. However, populating the application with all the possible component/failure-modes will be an extensive and challenging exercise.
- Minimizes number of applications and/or supporting applications
 - This adds only one new application and requires no supporting applications.

Make Recommendations

For AD, the K-Means method was shown to be sensitive to changes when compared to previous operations that were deemed nominal. The model testing resolved the unknown requirements, and this application has the flexibility to be ‘dialed-in’ to meet various system peculiarities. However, it needs to be reiterated that the training data is unsupervised, so it is possible that abnormal data may be embedded within this dataset. The recommendation is to use the IMS application. It was developed by NASA and they retained a licensing exemption that allows its use for NASA programs.

When the Shuttle program gained real-time plotting capability, this provided the ability to monitor the system over time. As there is very limited monitor space, only one plot could be viewed at a time. For implementation, it is envisioned that K-Means vectors would be built that are common to a saved plot of related data. A display would be set up that showed multiple k-distance graphs. Should one of these start to diverge high, the corresponding plot would be brought up so the operator can see the plotted data of the indicators of concern. This methodology provides a 'health' indicator allowing the operator to indirectly monitor multiple plots on a single display. In addition, a threshold value can be set that will alarm should a k-distance value exceed it (whether it is visible or not).

During the FI modeling, a knowledgebase was developed using Excel. The intent was to see if the desired information could be presented with minimal user input. As the model for this functionality turned out to be relatively easy, creating a comprehensive knowledgebase with Excel is the recommendation going forward. If it later turns out that this is not sufficient, then a fault-map application could be considered at that time. Both methods are level in their ability to meet requirements, though minimal development could only be partially met. As this is an extensive effort, Excel has an advantage in that it requires only SME support where those inputting the model require both SME and FI application know-how.

CHAPTER SIX: ANALYSIS/EVALUATION

A preliminary framework was proposed in Chapter 4 that provided a methodological approach to selecting anomaly detection (AD) and fault isolation (FI) technologies that can be adapted to complex systems. A case study was presented in Chapter 5 that followed this process to augment the liquid hydrogen (LH2) system at the Kennedy Space Center (KSC) launch pads. This system supported loading the external tanks (ET) to support Space Shuttle launches, and will be used again for NASA's next-generation Space Launch System (SLS) rockets. As LH2 is the fuel for the Shuttle main engines, it is very flammable. LH2 is also a cryogenic fluid (-423 °F). This involves insulation challenges as the liquid is constantly boiling. To minimize losses, the ET is not loaded until very late in the launch countdown (within ten hours of planned launch). Loading the Shuttle is performed from a control room located several miles from the launch pads. Therefore, these remote, time-critical and hazardous operations make the LH2 system a good candidate to supplement with AD/FI capabilities. This chapter will analyze the case study and its implementation of the framework process, and then finalize the proposed framework. This framework will then be validated by applying the process to the liquid oxygen (LOX) system at KSC that is also used to load the ET.

Framework Analysis

The framework process follows the chart shown previously (reference Figure 8 - Preliminary Framework Process Flow). This analysis will step through each process block and review the details necessary to accomplish these steps. It will also note any changes that will be reflected in the final version.

Scope the System

Establishing the system scope not only identified the system boundaries under study, it also identified existing AD/FI capabilities. The LH2 system was broken to three major subsystems (Pressurization, transfer and vent). The existing AD capabilities were described (no existing FI functionality) so that new applications with common functionality are ground-ruled out.

This task did highlight a couple of items worth noting as this impacted the scope of this case study. First, the command and control (C&C) system at KSC went through an upgrade utilizing programmable logic controllers (PLCs). As this new C&C architecture has significant health and status capability embedded within, further AD/FI augmentation was considered unneeded. Second, as hydrogen is very flammable, and potentially explosive, the ability to monitor for leaks and/or fires is considered crucial. This is already accomplished via a ‘HazGas’ subsystem that has installed leak and fire detectors throughout the LH2 system.

Identify and Categorize Sensor Data

Identifying all the sensors supporting both AD and FI applications is a way to ensure inclusion of all possible measurements. Sensors are monitored for nominal operations with AD, and used to authenticate fault-modes with FI. Most system designs include a parts list (as does the LH2 system). However, when using SysML to model the system, it becomes intuitive to organize the component blocks in ‘packages’ that reflect the component type for later retrievals. This was helpful for itemizing the types of indicators. When these instruments are later depicted in IBDs

(as instances of ‘part properties’), they are readily found within those diagrams, and IBDs also reflect how they are used.

Identify and Categorize Fault Modes

For this case study, knowing the fault modes was not required to apply the selected AD utility. AD focused on the ability to detect instruments deviating from the norm, and therefore, it did not consider the faults that drove the disparity. The FI application requires that all faults be identified, and those that can be detected by the system instrument array be catalogued within a knowledgebase. As stated previously, this can be a comprehensive task for a complex system.

This framework development is guided by system engineering principles of which one best practice is to leverage off of existing artifacts. If a SysML model is being developed, then the corresponding IBDs will depict which components are within the system, and how they are being used. This information might make it practical to auto-generate a fault tree (FT) using the information generated by the system design. The model was extended by adding generic failure modes (blocks) and allocating them to the components shown on the IBD (via a relationship matrix). The matrix was exported to Excel, and a macro written to draw a basic fault tree as described in Chapter 5. This task was successful in highlighting the potential of extending SysML so that existing design constructs can assist in generating other design products. Fault trees are typically generated as a separate project to the design effort. Safety engineering oversees the development, though it is supplemented by the SME’s who have the required technical expertise. Having the ability to auto-generate an initial FT that is directly related to the

design documents is efficient in the use of engineering labor, less prone to mistakes and likely to identify deficiencies in the design product(s) earlier in their life-cycle. This proof-of-concept exercise focused on FT's, but it is foreseen that other design or operational products could benefit from SysML's flexibility in working with its core data.

Identify and Categorize AD/FI Applications

This section of the test case started with the discussion on requirements. One of the changes to the framework process chart is to pull the requirements development out into its own block following the system scope effort. The functional requirements development does need to follow the system scope. While defining system scope, having to apply AD/FI was ground-ruled out for C&C functions and Leak & Fire detection. In addition, the existing AD/FI methods were appraised. It is expected that the detection results will overlap, but the detection methods should not be common. This type of information is helpful before functional requirements are produced.

Requirement generation was planned to precede the AD/FI selection process, so the preliminary framework scheduled this activity accordingly. However, since the sensor and fault (more so the faults) organization can be a comprehensive undertaking, the requirements should be defined prior to committing resources to this effort. These actions are shown in parallel with the system categorization items, but the process flow will be changed to drive the requirements generation prior to additional work following system scope.

The requirements generated were categorized as functional (qualitative) and cost related. They were purposely kept at a higher level for several reasons. For the functional requirements, a concern exists that requirements that are too restrictive may exclude technologies that provide considerable benefit to the system. Also, as this system is being augmented with additional AD/FI, the current operational needs are satisfied by the implementation of the original design requirements. On the cost side, the requirements are written generically to keep potential project expenses minimized. The case study had to decide between existing applications or internal development. These options have different means of costing that impact different groups.

Modeling the AD/FI technologies is time consuming, and therefore, modeling all the potential applications is not practical. In addition, many methods found in the literature did not provide adequate details to build a test model. To lessen the potential effort, both the AD and FI applications were divided into classes based on common underlying techniques used to realize their objective. These classes were then ranked based on their ability to meet requirements. This is an intermediate step intended to reduce the potential candidates to a class that is most likely to meet system needs.

There is a cost requirement that gives priority to an existing application (or mature development). This requirement is based on an assumption that it is cheaper to purchase a software license than it is to internally develop and sustain this functionality. In addition, commercially available packages publish the capabilities of the product making it easier to determine which functional requirements they could satisfy. Thus, the requirements derived for

this case study favor such applications (both directly and indirectly) as reflected in the down-select decisions. The ability of the chosen classes to meet the functionality requirements, related to performance, remained mostly ‘unknown.’ Resolving these unknowns is what drives model development and subsequent testing.

Model the System

This framework is designed to work with both new projects and legacy systems. As the SysML language was selected for modeling the system, it is assumed any new design would also be implemented with this standard. Therefore, a suite of SysML diagrams would be available (or quickly generated) to the applicable stakeholders. This assumption is based on the remote likelihood of selecting multiple MBSE standards. Should another language/method be selected, the case study still provides an outline of system engineering practices that apply, and may be implemented in a similar way with the tool(s) supporting those standards.

This test case is enhancing a legacy system used for Shuttle (and planned to support future launch programs). The ground systems were modified from the Apollo program in the late-1970’s/early-1980’s predating any formal MBSE standards. Subsequently, a mature design exists, though it epitomizes a document-centric methodology that was predominant at that time. For operational systems, the goal is not to model the entire system as this is not an efficient use of resources. However, if the product life-cycle is to continue into the future, then incorporating SysML elements to support system design changes may be appropriate. In this case, four of the nine available SysML diagrams supported this effort to include:

- Package Diagram
 - A Package Diagram is required to contain all other SysML diagrams generated, so it is required for any SysML model. It is used simply to organize the model, and works very similar to folders used within Windows. The package name is included in the frame-label of all other diagrams.
- Block Definition Diagram (BDD)
 - BDD's were developed to show the composition of the various subsystems (LH2StorTankPressSys) and also as a way to graphically display elements stored as blocks (components, failure modes). This supported scoping the system.
- Internal Block Diagram (IBD)
 - An IBD shows the internal workings of a single block to include the inner connections of the parts and flow between them. The block 'LH2StorTankPress' was created and showed via additional blocks of all the components that comprised this subsystem. An IBD was then created from the 'LH2StorTankPress' block. However, all the composite blocks included in the BDD are changed to a property of 'parts.' A block shows a given component used within the system, where as an IBD shows an instance of that part (block equates to a part number – of which there may be many used - where an IBD part refers to the unique identifier for that component). IBD's were used to support system scoping, FT development, failure-mode development and fault mapping.
- Requirements Diagram

- SysML has a very strong capability to track requirements from initiation to closure. Requirements diagrams were used to develop the AD/FI application requirements.

Model the AD/FI Applications

It could not be determined if both the AD and FI classes selected could fully meet the requirements. This drove a need to model and test these methods to determine if they incorporating them within the LH2 system was feasible.

Anomaly Detection Analysis

The first tests performed a sensitivity analysis of the K-Means method of detection. A single cluster was simulated, and training data was selected. The initial variables represented:

- Multiple data types (pressure, temperature, valve position)
- Common value ranges (0-100%; 0-150 PSIG, 0-150 °F)
- Dissimilar value ranges (0-150 and (-409)-(-423) °F; 0-150, 0-750 and 0-3000 PSIG)

These variables replicate actual indicators within the LH2 system, and the initial training data values come from real data. However, they are not related to one another. They were selected as the initial testing is looking for responses from a diverse dataset.

When a cluster is formed, only the high and low values from each element within the vector are needed to determine a k-distance. The initial training data values were randomized to +/- 1%, and the resulting high/low values retained. This simulates the influence of training data as each

variable will have differing impacts to that baseline as corresponding value trends away from nominal, and by raising the k-distance baseline above zero.

With training data simulated, a single indicator was incremented by 1% up to 5%. As the initial 1% increment will keep it within or near the training data high/low values, only a slight shift is observed. As the increments continue, the baseline shifts grow. At 3%, the baseline has roughly doubled, and at 5%, more than tripled. This test is repeated, but with five different indicators being incremented (both high and low) as this mimics the data being related to one another. A 1% increment keeps all the test values within or near the training data high/low values, and a slight increase is noted with four of the five increments. The initial starting value on the last indicator was set outside the training data low value, so an upward increment brought it closer to its norm. Overall, a slight upward trend is noted. Each is then increased by 2%, and the baseline nearly triples.

Often, the data being tested is not as stable as shown previously. The same series of tests were repeated, but this time with +/-1% of randomized noise simulated with the test data. The results were similar, though the stable data plots stepped up while the noisy ones trended. Selecting a threshold value is a subjective task and will likely be based on how the baseline plot is acting. As a general rule for this case study, a doubling of the baseline should flag the operator of a possible trend away from nominal. A tripling should indicate divergence from nominal (as defined by the training data). A 100 PSIG system would have limits set at +/- 10%. Therefore, the K-Means method would have flagged the operator well before an alarm is issued.

The next test was specific to a requirement that was specifically developed to address an existing limitation. During the requisite configuration changes for the various loading phases, the limit settings are inhibited at the end of one phase, and activated (possibly to new levels) at the start of the next phase. This essentially turns off the alarms during these transitions. To test the K-Means ability to monitor these transient conditions, data from STS-134 (training) and STS-135 (test) during an LH2 transition from Slow Fill to Fast Fill is used. When presented during the case study, it was noted that there is an approximate 0.8 °F temperature bias between the two mobile launch platforms used for these launches (an acceptable tolerance). This bias further impacted the k-distance value due to the narrow range of the indicator in this timeframe. However, as the bias exists during the entire plot, further examination is necessary.

The STS-134 & 135 LH2 Slow Fill to Fast Fill transitions were previously plotted (reference figures 31 and 32). Both the pressure and temperature profiles, and the pressure values were much alike. Only the temperature stood out due to the noted bias. The STS-135 transition plot was changed to include the STS134 temperature (reference Figure 36 - STS-135 LH2 Loading (SF to FF) with STS-134 Temp. This plot starts out with a k-distance between 5-6 (very high) and remains above four throughout Slow Fill. During this time, a new cluster is being defined with approximately a 2 PSIG increase on the three pressure indicators. As the temperature is bit-toggling between 2 or 3 values, and a bit equates to 0.072 °F, the difference of the high/low values for this cluster equate to a small number. So normalizing a value with 0.8 bias results in a high k-distance value. During transition, a couple of spikes are noted, but these were also

observed in the graph with only pressures plotted. The spike is elevated some from the pressure-only calculations, but a view of the vector element's normalized values indicates all four measurements contributed roughly the same to this spike.

Following the transition spikes, the k-distance drops to just over one. As the bias is still present, this is unexpected. At this point, the three transfer line pressures have diverged to their Fast Fill values, and a cyclic pattern can be observed as they track the ET vent valve cycling that maintains back pressure on the tank. During this period, new clusters are being defined, but the temperature measurement is more active. This larger gap (approximately 1.0 °F) between the temperature's high and low element values lowers the normalized value, and subsequently, the k-distance.

The final observation is related to the latter part of Fast Fill. Again, the temperature is bit-toggling which drove the k-distance value significantly higher during Slow Fill. For Fast Fill, k-distance drops to approximately 0.5. What is also observed is that the STS-135 temperature has dropped into a region where the STS-134 temperature had been active (training data), and the three pressures had already started their Fast Fill profile. Therefore, the low k-distance value is correlating to the activities immediately following the start of Fast Fill for STS-134, and not the latter part of Fast Fill when it decreases to LH2 temperatures.

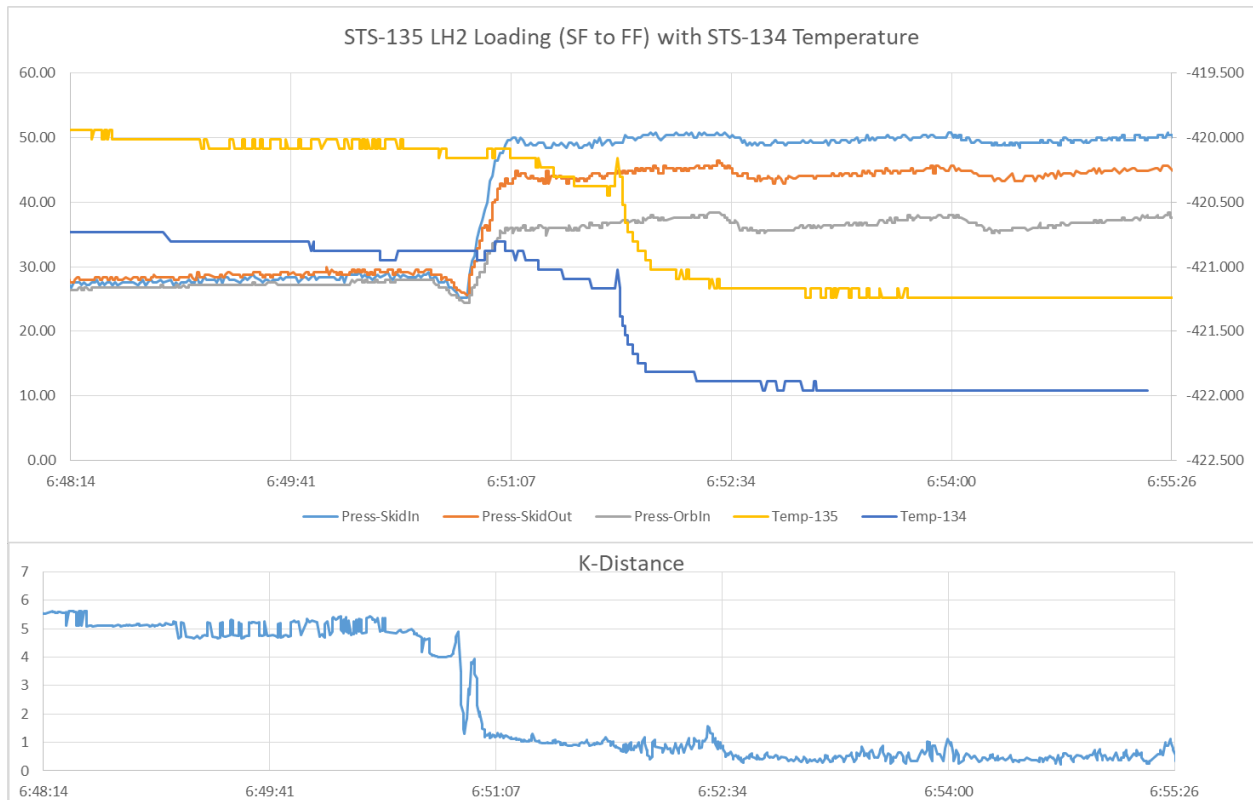


Figure 36 - STS-135 LH2 Loading (SF to FF) with STS-134 Temp

This test highlights that when working with dynamic data, there may be times this data better fits a cluster that does not reflect the same activity from which generated the test data. An observation such as this is desired when doing post-test data reviews as the time is available to resolve the issue. However, this was a non-issue that was initially depicted as off-nominal, and with the bias remaining constant, later displayed as nominal. These are not the type of events wanted for supporting real-time operations as they can distract the operators from their primary goal of monitoring system operations.

Fault Isolation Analysis

The implementation selected involves generating a knowledgebase which is a straightforward task. However, there is an open issue regarding inclusion of single vs. multiple fault modes as follows:

- If the knowledgebase is to include multiple-fault failures:
 - Should it be limited to auxiliary and/or secondary subsystems that are activated due to a failure on the primary subsystem?
 - Should it include the possibility of dual failures within a related subsystem (i.e. a regulator fails open and the corresponding relief valve fails to open)?
 - Should it include components that may be impacted due to failure propagation thru the system (i.e. an over-pressurization due to relief valve failure)?

As this task is comprehensive, the initial focus should be including all single-fault modes that can be detected by instrumentation. It can also include auxiliary or secondary subsystems as these will be common to the primary side (and are often active and being monitored). As the goal is to present likely candidates, including multiple failures will likely generate lengthy lists with many unrealistic scenarios. This is a knowledgebase, so the user always has the option to include multi-failures they deem credible or have experienced in the past.

Proposed Framework

With the case study concluded and an analysis of the applicable elements performed, a proposed framework will be summarized. This updates the preliminary version and incorporates additional detail gathered during the study process.

- Scope the system
 - Determine the extent of the system to be augmented
 - Identify existing AD/FI capabilities
 - Avoid duplication of existing techniques
 - Rule out subsystem inclusion or failure types if already supplemented
- Generate Requirements
 - Include both cost and functional related requirements
 - Generate initial requirements at a higher level
 - Do not want to restrict initial AD/FI classes to consider
- Identify/categorize sensors
- Identify/categorize fault modes
 - Initially done at the component level, and then applied generically to like components
- Research available AD/FI technologies
 - Consider:
 - Commercially available applications
 - Mature development (algorithm(s) constructed and tested)
 - Conceptual techniques with supporting test results from multiple origins
 - Categorized technologies into classes with common underlying methodologies
- Down select to an AD and FI class that best meets requirements
- Model the system
 - For existing systems, model elements of system that will support AD/FI testing

- Model the AD/FI techniques
 - If testing proves application is unacceptable, consider next AD or FI class
- Perform trade studies of available options within the class
 - Can include existing applications or new development
- Make recommendation

Some of these tasks outlined above may also be performed in parallel. The process flow diagram has also been updated to reflect the proposed framework (reference Figure 37 - Proposed ADFI Selection Framework). The following highlights the changes to the final version:

- Generating requirements was added as its own step prior to identifying sensors and fault modes
 - This task was originally embedded within the Identify AD/FI applications block
 - Knowing the requirements may provide insight as whether listing all sensors and/or fault modes is required
- Down-select to AD and FI classes was added following steps to gather system and AD/FI information.
 - As the research uncovered numerous potential methods, a means to limit the modeling and testing was a required intermediate step

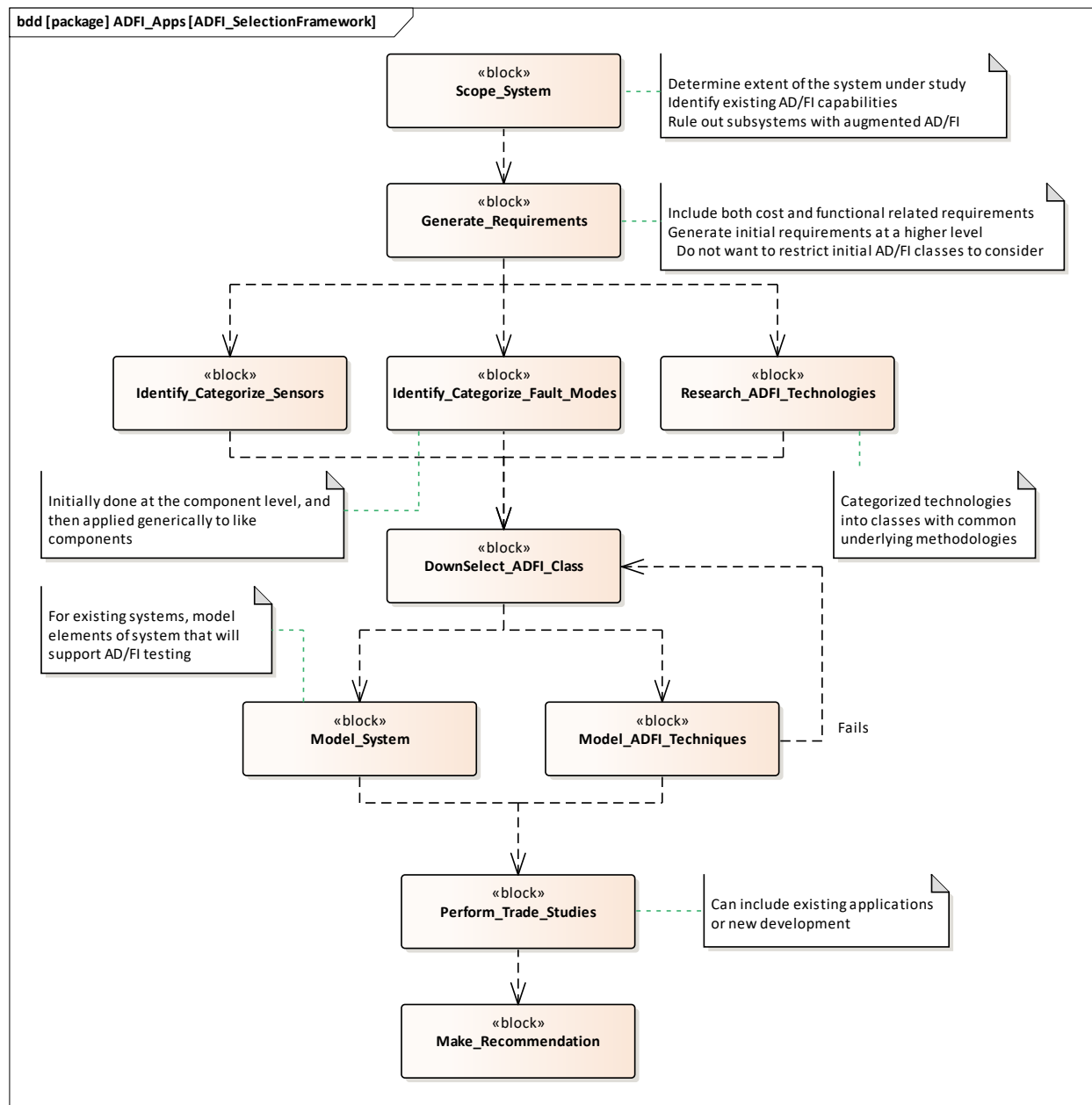


Figure 37 - Proposed ADFI Selection Framework

Framework Validation

To validate the framework, this process will be implemented on the liquid oxygen (LOX) system that supported ET loadings for launch. This system is also planned to be used with the next-generation of NASA rockets. Liquid oxygen is also a cryogenic fluid and provides the oxidizer used by the Shuttle's main engines.

The various KSC systems that operate out of the control room share the same C&C architecture, and subsequently, there is commonality as to how the existing AD and FI capabilities currently function. Therefore, many of the framework processes implemented during the case study are also applicable to these other varied systems. A system engineering best practice is to reuse any applicable artifacts as this both reduces effort by not recreating them and keeps the content consistent when used across a spectrum of disciplines. When possible,

The LOX system loading operations also take place late in the launch countdown. They are both hazardous and time-critical, so this system could also benefit from AD/FI augmentation. As LOX is much heavier than LH2, one of the main differences between these two systems is LOX requires the use of large cryogenic pumps (primary and secondary) to load the vehicle. LH2 is loaded by pressure only, so lacks any comparable hardware. Therefore, the scope for this effort will encompass the LOX pump subsystem.

The C&C subsystem has its own health and status capabilities, so this will be excluded from consideration. If a leak occurs, LOX does not pose the same threat as LH2, so it does not have a

supporting leak and fire subsystem. As cryogenic fluids produce large vapor clouds when exposed to ambient temperatures, the pad camera system will be used to identify leaks. Limit setting is the primary method of AD for this system.

With the scope defined, the requirements can be generated. The requirements developed for LH2 were reviewed, and as these are generally high-level (and not system specific), they were found to be applicable to LOX. Subsequently, the research and ranking of the AD and FI applications is also applicable. The LOX sensor and fault modes will be needed for the FI application, so they are added to the SysML model. Some of the LOX components within the pump subsystem are of the same type as LH2, so the generic fault modes for these items can be ported over to the LOX model.

As the down-select classes are suitable for the LOX system, the K-Means method will be used for AD. Training data was pulled from the STS-134 mission during the Replenish loading phase. Replenish operations follow the initial tank loading, and keep the ET liquid level at flight mass to compensate for the boil-off of cryogenic fluids. It is during Replenish when the astronauts can board the Shuttle, and this phase can last 4 to 10 hours. Four measurements related to pump performance include thrust bearing temperature, bearing oil temperature, current applied to the variable frequency drive and pump outlet flow (GPM). The data is plotted over an approximate 4-hour window, and no obvious anomalies are observed (reference Figure 38 - STS-134 LOX Pump - Replenish).

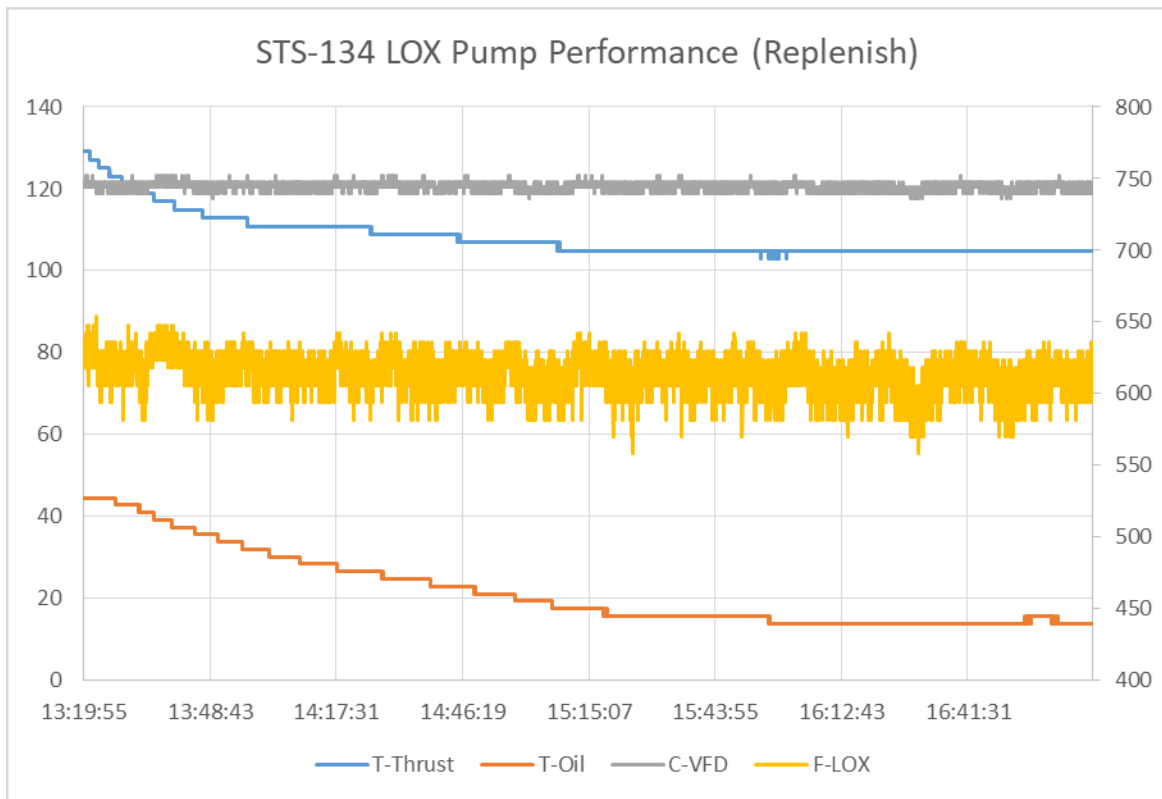


Figure 38 - STS-134 LOX Pump - Replenish

During Replenish operations for STS-135, a leak occurred on the pump after approximately three hours of Replenish. It started slowly, and very gradually worsened (reference Figure 39 - STS-135 LOX Pump – Replenish (k-distance)). After nearly an hour, the pump was secured and the secondary pump brought online to support a successful launch. The STS-135 pump plot includes the corresponding k-distance plot. Even with a couple of noisy indicators, the k-distance forms a stable baseline. When the pump temperatures start to drop, the upward deflection on k-distance is quite apparent signaling an off-nominal trend.

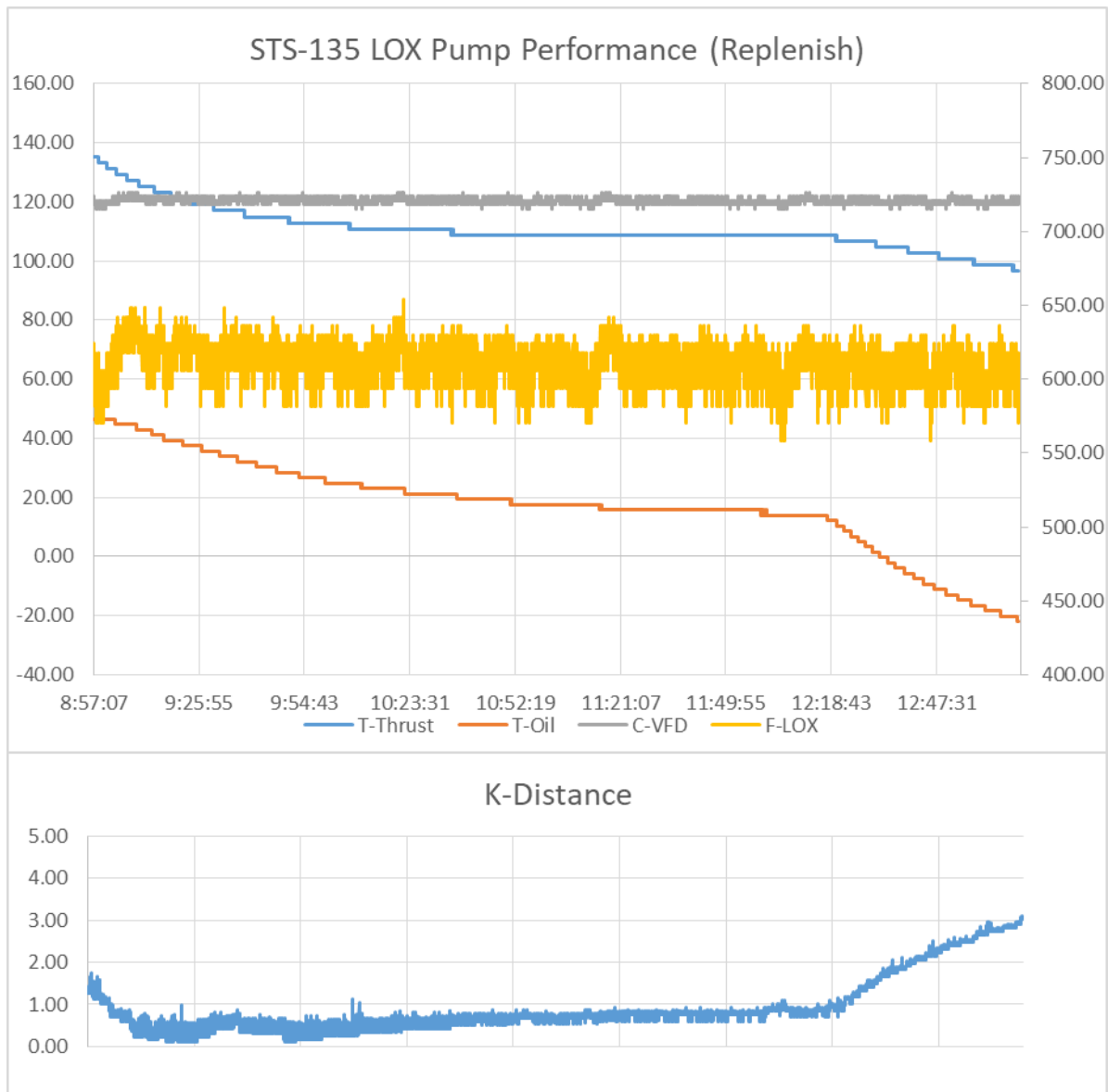


Figure 39 - STS-135 LOX Pump – Replenish (k-distance)

For a leak failure, camera views are the primary method for both detection and isolation. Cold vapors are the norm in the storage area as there are uninsulated pipes (in addition to the pumps) that experience cryogenic temperatures. When a leak occurs, these vapors will tend to envelop

the area of origin which is an indicator for the operator. Leaks can occur anywhere in the cryogenic systems, though fastened joints are the primary source. Rarely are temperature sensors ideally located to capture external leaks, so having these measurements corroborate a leak is unique. These indicators are installed to monitor the pump's bearings in which the failure mode is high temperature. The two LOX pumps are swapped operationally with each loading so they are both exposed to run time. The secondary pump's thrust bearing temp typically approaches its limit when the loads are at their highest (Fast Fill), and the pump speed is reduced slightly so it stays within specification (temperature and operator actions create entirely different pump profiles which must be considered when selecting training data).

For the FI knowledgebase, perhaps only the high limit was considered a valid failure mode for these indicators. This could then be remedied by simply adding the low value faults. A model update would be more complicated requiring one with adequate skills to accomplish the task. If the model is configuration controlled, then there are additional reviews and approvals required. This highlights how an internal knowledgebase provides more flexibility in being sustained. If it is adequately providing FI capabilities, then this ease of maintenance should be a factor involved if a decision to choose an FI model is needed.

The framework provided a process that was used to select AD/FI applications for the LOX system. The hardware was scoped to the pump subsystem as there are no pumps on the LH2 system that was used in the case study. However, when possible, actions accomplished during the case study that were applicable to the LOX system were not repeated if the results were not

expected to change. This was primarily related to requirements generation. Although the hardware differed significantly, the existing AD/FI capabilities were common (as they are to most systems in the launch control room). This validates the decision to go forward with K-Means AD (via the IMS application) and generating an internal knowledgebase to incorporate FI capabilities, and therefore, the AD/FI selection framework. It is noted that K-Means could potentially have issues related to real-time operations. However, as these issues are understood, they can also be overcome with methods to control the training data utilized combined with the flexibility IMS provides in tuning the model.

CHAPTER SEVEN: CONCLUSION

This research proposes a framework to be used by organizations with a need to enhance system anomaly detection and fault isolation capabilities. This chapter will summarize the effort, highlight resulting contributions, describe a limitation encountered and recommend future work to further expand this subject matter.

Summary

Chapter 1 introduces a need to improve upon existing anomaly detection and fault isolation capabilities for critical systems. It points out that there are other methods available, but the applications selected for implementation do not always provide the anticipated benefit. A problem statement is formulated and potential research objectives defined.

Chapter 2 performs a literature review and confirms many new AD/FI techniques have been reported. The review also focuses on systems engineering approaches to select and implement this technology. Minimal research was uncovered that addresses the implementation of these new AD/FI technologies. Furthermore, literature describing current systems engineering practices did not deal with inclusion of AD/FI technologies. A gap analysis is performed indicating additional research is warranted.

Chapter 3 organizes and defines the methodology that will be used for this research project. It depicts the development of an initial framework followed by a case study for time-critical,

highly-hazardous system. The resulting (finalized) framework would be validated and the research effort summarized.

Chapter 4 produces an initial framework to provide guidance on AD/FI selection incorporating system engineering practices. It settles on a model-based system engineering approach and selects SysML as the standard to support the MBSE modeling. It describes a detailed process flow for identifying requirements, the system under consideration, and the AD/FI technologies candidates. Determining if requirements can be satisfied drives modeling of the techniques for testing. Most of the system modeling can be accomplished within the SysML tool. Testing the candidate applications required functionality beyond SysML capabilities. The framework process ends with an evaluation of the applications followed by a recommendation.

Chapter 5 conducts the case study with a focus on the liquid hydrogen system at KSC. Liquid hydrogen is used to fuel the Space Shuttle's main engines (as well as the next generation NASA rockets). These are highly-hazardous and time-critical operations executed from a remotely located control room. Thus supplemental AD/FI technologies could be valuable additions. This study follows the initial framework while scrutinizing the individual steps in an effort to enhance the process steps. As modeling the new technologies proves time consuming (many lack adequate detail and/or contain proprietary information), an intermediate step was added to the process flow to down-select to a classes of common techniques. The modeling effort, and subsequent testing, then focuses on those methods. For this case study, the choice came down to a single application (both AD and FI), or internal development. A recommendation is made to

go forward with an AD application, and internally develop a means to provide fault mapping to accommodate the FI requirements.

Chapter 6 analyzes both the framework methodology leading to the selection process and the chosen applications ability to meet the system requirements. The proposed framework incorporates the refinements noted during the case study and those generated from the analysis to establish a ‘finalized’ product. This framework is then validated against the liquid oxygen system also located at KSC. The AD application successfully signaled a downward trend in two indicators used to monitor pump bearing performance. The failure mode was attributed to a failed seal resulting in a liquid oxygen leak at the pump. The AD plotted (k-distance) value had flagged the off-nominal trend well before an alarm was triggered.

Chapter 7 concludes the research effort by summarizing the overall project. It also describes the research contributions realized, limitations encountered during the study, and a recommendation for future work to continue the study of related topics.

Framework

The detection/isolation technologies described herein go above what is readily available, or currently implemented, in industry (and providing adequate coverage). Therefore, the customer for this technology will have critical need(s) to offset the costs and/or effort. These can include systems that are: (a) large, complex, costly; (b) highly-hazardous; (c) time-critical; (d) remote

operated (e) expensive when idle. The framework developed recognizes this customer profile, as well as the necessity to provide a value-added result. Advantages realized by this framework:

- It adheres to systems engineering practices.
 - Organizations with systems requiring such applications are likely practicing system engineering, so will be consistent with their current policies.
- Uses MBSE methodology implemented with SysML
 - Organizations using same methods may already have much of their systems adequately modeled
- Selection process is requirements driven
 - If it cannot be determined that technologies under consideration fully meet requirements, drives additional modeling/testing to test capabilities
 - Validates the model prior to making recommendations
- Objective process, so minimizes impact from biased stakeholders

Contributions

This dissertation is the first attempt to develop a framework with strict adherence to system engineering practices to improve and optimize system fault detection and isolation. The primary contribution is the framework itself as it provides a novel strategy to implementing new technology that can enhance system performance. It lays out a systematic approach to assist users in generating anomaly detection and fault isolation schema supporting existing or new designs. This directly addresses the original problem statement that initiated this research project.

Additional contributions include:

- Extending SysML to include generic component failures. This data was combined with existing Internal Block Diagrams components to auto-generate a fault tree (proof-of-concept demonstrated).
- Extend the contributions of those developing AD/FI technologies by providing a means to organize the detectors/isolators for consideration, and subsequently, acceptance for implementation should the capabilities meet the desired requirements
- Couple AD and/or FI with unique applications for which they were never intended. Path to generating AD/FI classes may uncover needs that could benefit from the underlying detection/isolation techniques
- Improve accuracy in anomaly detection and fault isolation capabilities by pairing those deemed optimal for the given environment in which they will operate

Limitations

A limitation encountered was that the systems under study, for the most part, shared the same requirements. These two systems were identified early in this project (LH2 and LOX), and the actual hardware selected for modeling/testing was (purposely) dissimilar. However, as the command and control system is common among all the systems operating out of the control room, the requirements did not change enough to drive the down-select to another class of AD or FI candidates. Ultimately, the model testing did show the selected classes were effective for both systems, even with the disparity in hardware tested. On the flip-side, the LH2/LOX testing

may be a test-case indicating these applications can be implemented for all systems operating out of the control room.

Future Work

The ability to automatically create an initial fault tree from a SysML IBD was illustrated. This relied on exporting a relationship matrix linking the IBD components to fault modes added to the SysML. The next logical step is to create new ‘stereotypes’ of the applicable model elements (further extension of the model) that can capture multiple component layering within an IBD. This information can then be passed to external applications via the SysML export standard (vs. a 2-dimensional matrix) for auto-generation use.

The construction of a system fault map was identified as being a comprehensive task. If SysML is extended to assign faults to components (as per the fault tree example above), then it can be further extended to associate indicator responses to a given component/fault combination. This too can be exported to an application for auto-generation of system fault maps.

The K-Means method of anomaly detection requires ingestion of training data where the user determines the sensitivity level in which the clusters are generated. It also allows for adjusting the sensitivity of the individual elements within a vector. Additional research should be pursued to include:

- Optimization of cluster sizing
 - To include high sensitivity for post-test reviews

- To include low sensitivity for real-time operations
- Optimization of individual test parameter settings
- Optimization for threshold (alarm) setting
- Testing for biases between datasets
- Testing for anomalies within the (unsupervised) training data
- Testing (and handling) of very narrow high/low ranges for data that will be normalized.

With limited data streams and display space to monitor the system, research that can integrate AD/FI technologies into a single application would provide value. With two applications running standalone, they are not going to be designed to communicate with one-another.

Furthermore, it is unlikely that either will be directly tied into the C&C network. An integrated application can react to its self-generated alarms driving it to fetch the corresponding faults.

Further testing of the framework outside of a space operations environment. This addresses the ‘narrow-testing’ limitation identified earlier, and would provide further confidence that this framework is appropriate for broad-industry use.

REFERENCES

- Abdul-Aziz, A., Woike, M. R., Oza, N. C., Matthews, B. L., & Iekki, J. D. (2011). Rotor health monitoring combining spin tests and data-driven anomaly detection methods. *Structural Health Monitoring*, 11(1), 3–12. <https://doi.org/10.1177/1475921710395811>
- Adler, R., Domis, D., Höfig, K., Kemmann, S., Kuhn, T., Schwinn, J.-P., & Trapp, M. (2011). Integration of Component Fault Trees into the UML. In J. Dingel & A. Solberg (Eds.), *Models in Software Engineering* (Vol. 6627, pp. 312–327). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from http://link.springer.com/10.1007/978-3-642-21210-9_30
- Angeli, C. (2010). Diagnostic Expert Systems: From Expert’s Knowledge to Real—Time Systems. *Advanced Knowledge Based Systems (Model, Applications & Search)*, 1, 50–73.
- Bay, S. D., & Schwabacher, M. (2003). Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 29–38). Retrieved from <http://dl.acm.org/citation.cfm?id=956758>
- Boeing: X-37B Orbital Test Vehicle. (n.d.). Retrieved November 22, 2013, from http://www.boeing.com/boeing/defense-space/ic/sis/x37b_otv/x37b_otv.page
- Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly Detection: A Survey. *ACM Computing Surveys*, 41(3), 15.1-15.58.
- Clark, T., Rabelo, L., & Yazici, H. (2017). Extending SysML Models to Enable Automatic Generation of Fault Trees. *IIE Annual Conference. Proceedings; Norcross*, 1085–1090.

- Cressent, R., David, P., Idasiak, V., & Kratz, F. (2010). Increasing reliability of embedded systems in a SysML centered MBSE process: Application to LEA project. *M-BED 2010 Proceedings*. Retrieved from <http://hal.archives-ouvertes.fr/hal-00630821/>
- Cristea, G., & Constantinescu, D. M. (2017). A comparative critical study between FMEA and FTA risk analysis methods. *IOP Conference Series: Materials Science and Engineering*, 252(1), 012046. <https://doi.org/10.1088/1757-899X/252/1/012046>
- Daigle, M., Foygel, M., & Smelyanskiy, V. (2011). Model-based diagnostics for propellant loading systems. In *Aerospace Conference, 2011 IEEE* (pp. 1–11). Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5747596
- de Lange, D., Guo, J., & de Koning, H.-P. (2012). Applicability of SysML to the Early Definition Phase of Space Missions in a Concurrent Environment. In *Complex Systems Design & Management* (pp. 173–185). Springer. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-25203-7_12
- De Stefano, C., Sansone, C., & Vento, M. (2000). To reject or not to reject: that is the question—an answer in case of neural classifiers. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions On*, 30(1), 84–94.
- Delligatti, L. (2013). *SysML Distilled: A Brief Guide to the Systems Modeling Language* (1 edition). Upper Saddle River, NJ: Addison-Wesley Professional.
- Ferrell, B., Lewis, M., Perotti, J., Oostdyk, R., & Brown, B. (2010). Functional Fault Modeling of a Cryogenic System for Real-Time Fault Detection and Isolation. In *Proceedings of the AIAA Infotech@ Aerospace 2010 Conference, AIAA, Atlanta, GA*. Retrieved from <http://arc.aiaa.org/doi/pdf/10.2514/6.2010-3548>

- Friedenthal, S., Moore, A., & Steiner, R. (2012a). *A practical guide to SysML: the systems modeling language*. Waltham, MA: Morgan Kaufmann.
- Friedenthal, S., Moore, A., & Steiner, R. (2012b). Model-Based Systems Engineering. In *A practical guide to SysML: the systems modeling language* (Second, pp. 15–27). Waltham, MA: Morgan Kaufmann.
- Gogoi, P., Bhattacharyya, D. K., Borah, B., & Kalita, J. K. (2011). A survey of outlier detection methods in network anomaly identification. *The Computer Journal*, 54(4), 570–588.
- Goldstein, M., & Uchida, S. (2016). A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. *PLOS ONE*, 11(4), e0152173.
<https://doi.org/10.1371/journal.pone.0152173>
- Grubbs, F. E. (1969). Procedures for Detecting Outlying Observations in Samples. *Technometrics*, 11(1), 1.
- Holt, J., & Perry, S. (2013). *SysML for Systems Engineering: A Model-Based Approach* (2 edition). London: The Institution of Engineering and Technology.
- Inductive Monitoring System. (n.d.). Retrieved March 2, 2019, from
<https://technology.nasa.gov/patent/TOP2-175>
- Iverson, D. L., & Field, M. (n.d.). Inductive System Health Monitoring, 7.
- Iverson, D., Martin, R., Schwabacher, M., Spirkovska, L., Taylor, W., Mackey, R., ... Baskaran, V. (2012). General Purpose Data-Driven System Monitoring for Space Operations. *JOURNAL OF AEROSPACE COMPUTING INFORMATION AND COMMUNICATION*, 9(2), 26–44.

- Johnson, T., Kerzhner, A., Paredis, C. J., & Burkhart, R. (2012). Integrating models and simulations of continuous dynamics into SysML. *Transactions of the ASME-S-Computing AndInfor Science in Engin*, 12(1), 011002.
- Kim, H., Fried, D., Menegay, P., Soremekun, G., & Oster, C. (2013). Application of Integrated Modeling and Analysis to Development of Complex Systems. *Procedia Computer Science*, 16, 98–107. <https://doi.org/10.1016/j.procs.2013.01.011>
- Kodavade, D. V. (2012). A Universal Object Oriented Expert System Frame Work for Fault Diagnosis. *International Journal of Intelligence Science*, 02(03), 63–70.
<https://doi.org/10.4236/ijis.2012.23009>
- Mackey, R., Brownston, L., Castle, J. P., & Sweet, A. (2010). Getting diagnostic reasoning off the ground: maturing technology with TacSat-3. *Intelligent Systems, IEEE*, 25(5), 27–35.
- Martin, R. A., Schwabacher, M. A., & Matthews, B. L. (2010). *Data-Driven Anomaly Detection Performance for the Ares I-X Ground Diagnostic Prototype*.
- Marzat, J., Piet-Lahanier, H., Damongeot, F., & Walter, E. (2012). Model-based fault diagnosis for aerospace systems: a survey. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 226(10), 1329–1360.
<https://doi.org/10.1177/0954410011421717>
- Matthews, B. L., Srivastava, A. N., Iverson, D., Beil, B., & Lane, B. (2011). Space shuttle main propulsion system anomaly detection: A case study. *Aerospace and Electronic Systems Magazine, IEEE*, 26(9), 4–13.

- Mosterman, P. j., & Biswas, G. (1999). Diagnosis of continuous valued systems in transient operating regions. *IEEE Transactions on Systems, Man & Cybernetics: Part A*, 29(6), 554–565. <https://doi.org/10.1109/3468.798059>
- Murugavel, P., & Punithavalli, M. (2011). Improved Hybrid Clustering and Distance-based Technique for Outlier Removal. *International Journal*, 3. Retrieved from <http://www.doaj.org/doaj?func=fulltext&aId=699249>
- Omar, S., Ngadi, A., & Jebur, H. H. (2013). Machine Learning Techniques for Anomaly Detection: An Overview. *International Journal of Computer Applications*, 79, 33.
- OMG SysML. (n.d.). Retrieved July 27, 2016, from http://www.omgsysml.org/#What-Is_SysML
- OMGSysML-v1.3-12-06-02.pdf. (n.d.). Retrieved from <http://sysml.org/docs/specs/OMGSysML-v1.3-12-06-02.pdf>
- OMGSysML-v1.4-15-06-03.pdf. (n.d.). Retrieved from <http://sysml.org/docs/specs/OMGSysML-v1.4-15-06-03.pdf>
- Osipov, V. V., Daigle, M. J., Muratov, C. B., Foygel, M., Smelyanskiy, V., & Watson, M. D. (2011). Dynamical Model of Rocket Propellant Loading with Liquid Hydrogen. *Journal of Spacecraft and Rockets*, 48(6), 987–998. <https://doi.org/10.2514/1.52587>
- Park, H., Mackey, R., James, M., Zak, M., Kynard, M., Sebghati, J., & Greene, W. (2002). Analysis of space shuttle main engine data using Beacon-based exception analysis for multi-missions. In *Aerospace Conference Proceedings, 2002. IEEE* (Vol. 6, pp. 6–2835). Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1036123

- Puig, V., Quevedo, J., Escobet, T., Nejari, F., & de las Heras, S. (2008). Passive Robust Fault Detection of Dynamic Processes Using Interval Models. *IEEE Transactions on Control Systems Technology*, 16(5), 1083–1089. <https://doi.org/10.1109/TCST.2007.906339>
- Qualtech Systems » TEAMS-Designer. (n.d.). Retrieved November 23, 2013, from <http://www.teamqsi.com/products/teams-designer/>
- Ruijters, E., & Stoelinga, M. (2015). Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer Science Review*, 15–16, 29–62. <https://doi.org/10.1016/j.cosrev.2015.03.001>
- Russell, M. J., Lecakes, G. D., Mandayam, S., & Jensen, S. (2011). The Intelligent Valve: A Diagnostic Framework for Integrated System-Health Management of a Rocket-Engine Test Stand. *IEEE Transactions on Instrumentation and Measurement*, 60(4), 1489–1497. <https://doi.org/10.1109/TIM.2010.2101350>
- Schwabacher, M. A., Martin, R. A., Waterman, R. D., Oostdyk, R. L., Ossenfort, J. P., & Matthews, B. (2010a). Ares IX ground diagnostic prototype. Retrieved from <http://ntrs.nasa.gov/search.jsp?R=20100027332>
- Schwabacher, M. A., Martin, R. A., Waterman, R. D., Oostdyk, R. L., Ossenfort, J. P., & Matthews, B. (2010b). Ares IX ground diagnostic prototype. Retrieved from <http://ntrs.nasa.gov/search.jsp?R=20100027332>
- Schwabacher, M., Oza, N., & Matthews, B. (2009). Unsupervised Anomaly Detection for Liquid-Fueled Rocket Propulsion Health Monitoring. *Journal of Aerospace Computing, Information, and Communication*, 6(7), 464–482. <https://doi.org/10.2514/1.42783>

Space Exploration Systems. (n.d.). Retrieved November 22, 2013, from

http://www.sncspace.com/ss_space_exploration.php

SpaceX. (2011). SpaceX Breaks Ground on Vandenberg Launch Site for Falcon Heavy - The

World's Most Powerful Rocket. *Business Wire (English)*. Retrieved from

<https://login.ezproxy.net.ucf.edu/login?auth=shibb&url=http://search.ebscohost.com/login.aspx?direct=true&db=bwh&AN=bizwire.c35402034&site=eds-live&scope=site>

Stamatelatos, M., Vesely, W., Dugan, J., Fragola, J., Minarick III, J., & Railsback, J. (2002).

Fault Tree Handbook with Aerospace Applications (Version 1.1). NASA Headquarters

Washington, DC 20546: NASA Office of Safety and Mission Assurance.

Straub, J. (2011). A Review of Spacecraft AI Control Systems. In *Proc. 15th World Multi-*

Conference on Systemics, Cybernetics and Informatics. Retrieved from

http://works.bepress.com/cgi/viewcontent.cgi?article=1042&context=jeremy_straub

TEAMS-RT. (n.d.). Retrieved March 3, 2019, from <https://www.teamqsi.com/products/teams-rt/>

Vipavetz, K. G., Murphy, D. G., & Infeld, S. I. (2012). Model-Based Systems Engineering Pilot

Program at NASA Langley. Retrieved from

<http://ntrs.nasa.gov/search.jsp?R=20120014575>

Wu, J. (2005). Liquid-propellant rocket engines health-monitoring—a survey. *Acta Astronautica*,

56(3), 347–356. <https://doi.org/10.1016/j.actaastro.2004.05.070>