Fault Insertions into Hardware-in-the-Loop Simulation

Undergraduate Thesis

Presented in Partial Fulfillment of the Requirements for Graduating with Honors Research Distinction at The Ohio State University

By

Martin, Tyler R.

Undergraduate Program in Mechanical Engineering

The Ohio State University

2020

Thesis Committee

Dr. Shawn Midlam-Mohler, Advisor

Qadeer Ahmed, Committee Member

Abstract

The Ohio State EcoCAR Mobility challenge is an intercollegiate team that designs, builds, and tests a hybrid electric vehicle. One of the main goals of this team is to build a hybrid supervisory controls strategy that tests the potential failure mechanisms derived from fault analysis. Currently, Automotive companies are focused on integrating model-based designs enabling simulations for low-cost, rapid experimentation that assess a vehicle's performance. Model-based designs allow engineers to simulate specific tests within controlled environmental conditions. Through the use of model-based design, engineers can test vehicle and component faults inside a simulation model to assess how the vehicle behaves during various failures without incurring the cost of destructive testing.

This thesis, in partner with the EcoCAR Mobility Challenge, aims to incorporate modern industrial fault diagnostics into a hardware-in-the-loop (HIL) simulation and analyze the performance of the model-based design. Fault Tree Analysis (FTA) and Failure Mode and Effect Analysis (FMEA) were used to develop the necessary requirements for the vehicle system. Different faults were intended to be tested for each major component, including, but not limited to, the energy storage system (ESS), rear electric motor, belted alternator starter, DC-DC converter, and the multiplexed vehicle electrical center. The ESS was the only component demonstrated as an example for integrating the fault insertion method. The research details how a standard method was constructed for developing and inserting faults in the HIL test environment. The process is used for testing and designing the control algorithm for a hybrid supervisor controller.

Dedication

I dedicate this research to the automotive industry and to the Ohio State University for all the practical information I have obtained from them. I took on the task of doing research to grasp a better knowledge of useful simulation practices that can be used to further improve vehicles people drive day-to-day. I hope to make vehicles safer and cheaper; allowing people to continue having a fast and enjoyable transportation method.

Acknowledgments

I want to give a sincere thanks for all the people that have help me along this difficult learning process. To Dr. Shawn Midlam-Mohler for his excellent advising and support for the EcoCAR mobility challenge team. To Kristina for the terrific support as my manager and for the incredible guidance during the research experience. And finally, to both Mahaveer and Hari for working beside me and learning the different software while supporting one another. It is a true honor to say that the EcoCAR team has become a second family for me and the team will always have a special place in my heart – keep doing amazing things!

Vita

March 25, 1996	Born – Columbus, OH
Summer 2016, 2017, 2018	Internship, Sutphen Towers – Hilliard, OH
Spring, Autumn 2018	Co-op, Robert Bosch – SC, Charleston
Summer 2019	Honda R&D – Raymond, OH

Fields of Study

Major Field: Mechanical Engineering

Table of Contents

List of Tables

List of Figures

Chapter 1. Introduction

The design of a vehicle's hybrid supervisory controller is a difficult task that must be separated into multiple subsections. This thesis is developed to go into detail about testing the fault insertion subsections for designing the controller. Testing faults with the vehicle's controller is an essential way to ensure that the controller is designed to be robust and handle different fault scenarios. In order to not damage the controller and safely run the execution of inserting multiple different fault scenarios the testing was conducted inside Hardware-in-the-Loop (HIL) simulation. HIL simulation allows for the controller to act as if installed on the vehicle and transmit different electrical signals via CAN communication. The usage of different HIL software allows the user to track the communication and check that the controller is designed to pass each developed requirement. The user can then send a faulty signal through the software and track that the controller behaves according to design.

1.1. Motivation

The automotive industry is spending large sums of money on testing and destroying multiple prototype vehicles. Companies are currently looking towards simulation techniques that can reduce the number of prototypes that are being destroyed or damaged from testing different fault cases. This research aims to provide a method for

conducting fault insertion with a simulation technique that both aids in the development and design of a controller in a safe and non-damaging way.

1.2. EcoCAR Mobility Challenge

The research is conducted in partnership with the Ohio State EcoCAR Mobility Challenge Team. The team constructs a four-year build cycle of taking apart a stock General Motors vehicle – this time it's a 2019 Chevy Blazer – and modifying the vehicle to become a hybrid electric automobile. The research is specifically working with the propulsion controls and modeling sub team that is conducted at the Center for Automotive Research. During this thesis the team will be finishing year two and it is important to demonstrated proper testing with designing the team's controller, the MicroAutoBox (MABx).

1.3. Thesis Structure

The thesis is broken up into six chapters and a short description of each chapter is as follows:

- Chapter 1: Introduction, briefly discusses what the research is about, the motivation behind conducting the research, and the partnership support for guiding the research
- Chapter 2: Background, goes over literature on different research that has been conducted on the similar matters regarding fault diagnostics, HIL simulation, and model-based design.

- Chapter 3: Methodology, takes what was learned from Chapter 2 and transitions to how that information was used within the thesis.
- Chapter 4: Implementations, covers the different software used for testing and inserting a fault into the team's controller
- Chapter 5: Results, provides evidence that the research was a success and that the fault was adequately inserted into the controller and verified with the comparison of another simulation technique
- Chapter 6: Conclusion, reiterates on what was covered in the course of the thesis and provides additional future work to be continued on the topic.

Chapter 2. Background

For every professional automotive company, process is important to follow; this ensures that the development of a system is fully defined and working as intended. The industry follows a standard V-diagram, shown in Figure 1, as a guide stone for developing a sequence of events to take place for the entire design of a product [1]. Everything starts from the top left of the diagram, the high-level product requirement, and is designed downwards into more component specific. High-level requirements are intended for brainstorming and building a functional base to guide the project along. Requirements will be continually updated as the design process continues, and the development of requirements will be highlighted in the following section. Each block is separated by having a deliverable/task to complete before moving on to the next block. The V-diagram is split into two section: the left side for system/model design and the right side for verification and validation [2]. The diagram is structured to become more component specific (low level) when working downwards from top to bottom while the top of the diagram is for the high-level systems.

Since this thesis is primarily about integrating a hybrid supervisor microcontroller into a vehicle we will only focus on a few necessary aspects of the V-diagram shown. *Requirements* will be analyzed by two different fault analyses methods, *Development/Prototyping* will be touch on with the integration of software-in-the-loop,

Testing & Tuning block will be addressed thoroughly with the integration of fault within hardware-in-the-loop simulation, and the additional *system* and *field testing* blocks will be cover briefly by component-in-the-loop and vehicle-in-the-loop simulations.

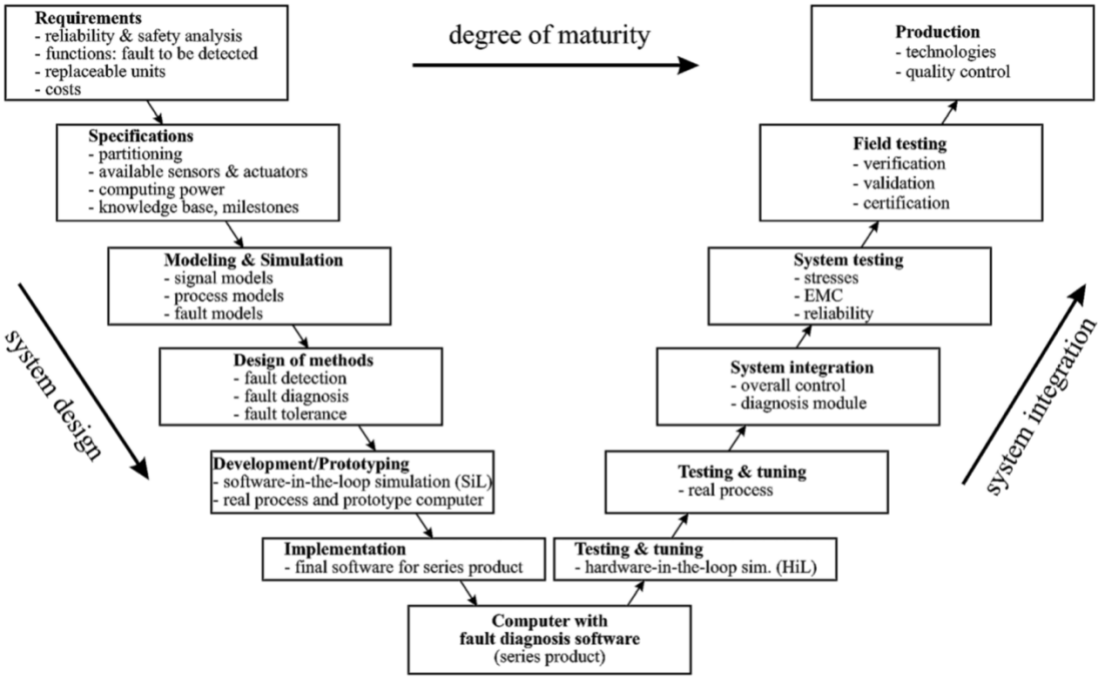

Figure 1 Industry Fault Diagnostic V-Diagram

2.1. Requirements Development/Fault Diagnostics

The immediate process for any good design is the development of proper requirements: “In industry, 40% of budget is spent on rework; of that 70-85% of the rework is due to errors in requirements” [3]. The high expense and risk with making improper requirements makes this section a vital portion of including proper requirements into the research project for fault insertion and fault diagnostics.

Fault Diagnostics are used worldwide for research and design. The goal for fault diagnostics is to understand faults, defined as “unpermitted deviations of a feature in a system from the acceptable, usual, or standard conditions”, in order to prevent or minimize failures from occurring. The classification for a failure is a “permanent interruptions of a system’s ability to perform a required function under specified operating conditions” [1]. Developing proper requirements aids in the analysis of a system’s failure and to create those requirement, two common industrial methods were used: Fault Tree Analysis (FTA) and Failure Mode and Effects Analysis (FMEA).

2.1.1. Fault Tree Analysis (FTA)

Fault Tree Analysis is an approach to understanding the causes of already known failures by breaking up the failure into smaller component specific faults. Fault Tree Analysis (FTA) is a top down approach that begins with the failure of a system and determines the possible causes for the components basic failures which include logic operations, Figure 2 [1]. FTA is a useful tool for decision making and has multiple purposes including: understanding the logic leading to the top event, prioritizing contributors to the top event, preventing the top event from occurring, monitoring the performance of the system, minimizing and optimizing resources, assisting in the design of a system, and diagnosing causes of the top event. [4] Only three of those conditions were utilized in the Fault Tree Analysis for this paper; understanding the logic leading to the top event, assisting in the design of a system, and diagnosing causes of the top event.

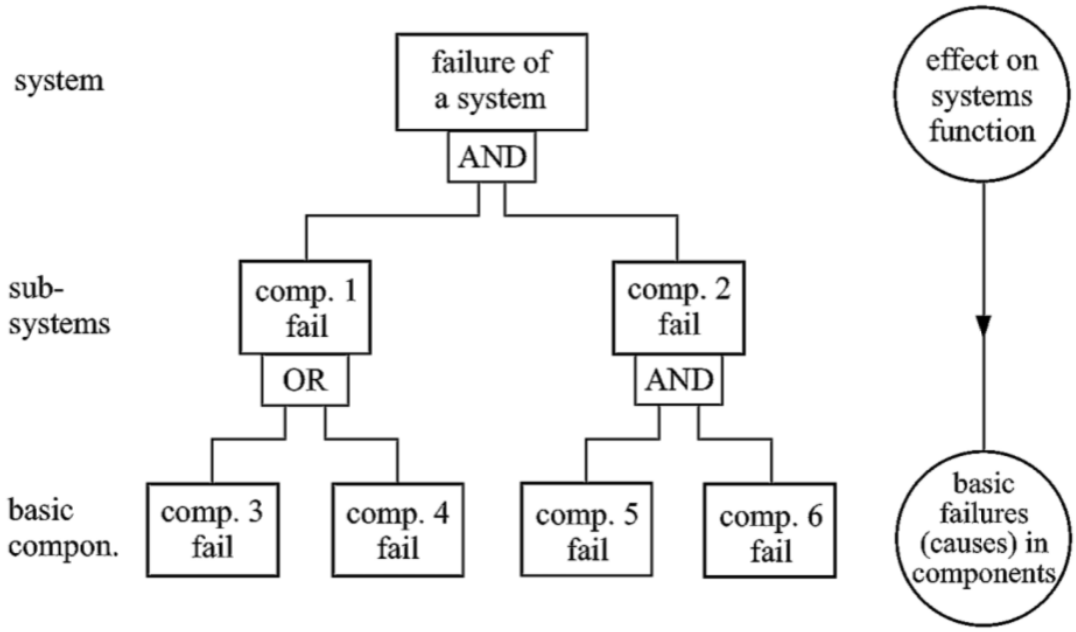

Figure 2 Fault Tree Analysis Block Diagram

In order to conduct an FTA the following list must be followed in order [4]:

1. Identify the *objective* for the FTA.
2. Define the *top event* of the FTA.
3. Define the *scope* of the FTA.
4. Define the *resolution* of the FTA.
5. Define ground rules for the FTA.
6. Construct the FTA.
7. Evaluate the FTA.
8. Interpret and present the results.

Each step is essential for properly guiding the construction of a fault tree, and its main function is to have the analysis remain useful for the intended purpose. Steps 1-5 are prior to the construction of the actual fault tree (see Figure 2 for an example) and are important for setting up the guidelines in order to efficiently produce an analysis that is both helpful and logical for constructing requirements in the future. The scope defines which failures and contributions will be involve with the analysis, and the resolution is the planned amount of detail for breaking up the fault tree [4] hence these two steps can be thought of as going together. The ground rules provide the symbolic meaning of each gate/block. Figure 29, in Appendix B, shows the rules in which these blocks were constructed for this thesis. Step 6 and 7 are the visuals and construction of the analysis itself that breakup the common whole level failure into smaller component level faults. Step 8 takes the information from conducting the analysis and uses it to produce, the next sequential method, Failure Mode and Effects Analysis.

2.1.2. Failure Mode and Effects Analysis (FMEA)

After completing the FTA, a failure mode and effects analysis (FMEA) is performed to further understand the faults derived from the previous analysis. From Fords FMEA handbook, a FMEA is intended to “recognize and evaluate potential failures and its effects, and identify actions that could eliminate/reduce the failure from occurring while documenting the process” [5]. FMEA can be typically combined with FTA because the derived failure results from the FMEA can be incorporated into the FTA and vice versa. [1]. A constant loop can occur from these two methods by taking the faults from the FTA, using them in a FMEA to further understanding the faults, and use the in-depth

FMEA results as inputs for a new FTA. Continuing the loop until satisfied with the results/or have enough knowledge to complete the objective of the analysis, Figure 3.

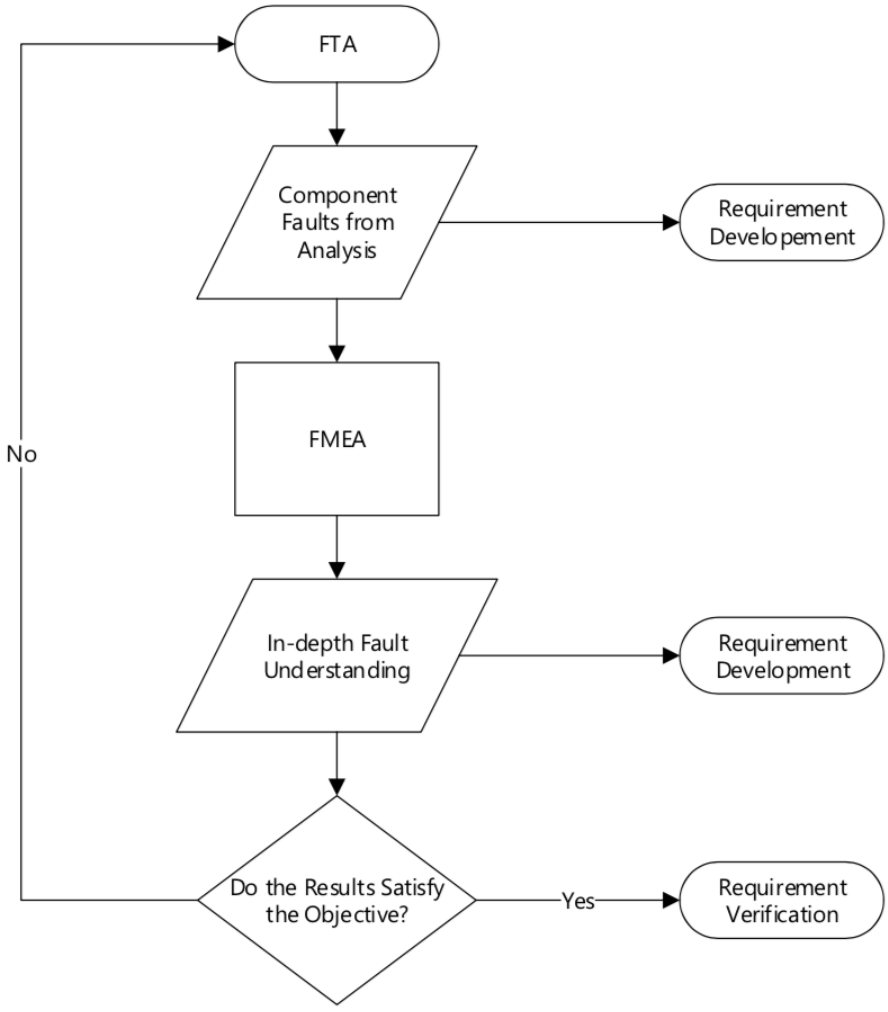

Figure 3 Requirement Development Flow chart

The FMEA involves filling out a form that gives specific details regarding the components of interest. Table 1 shows the structure of the form, where the bold words are from the form and the rest of the box is an example for the structure. There can be

multiple functions for a specific item and each function can break down further with multiple potential failure modes and multiple potential effects. The first column instructs listing a function which must be measurable, similarly to a requirement [5]. The second column consists of failure modes which can fall into four categories: No function, partial function (degradation over time), intermittent function ("loses functionality due to external factors"), and unintended function. Columns 3 and 5 specifically draw out different effects and causes related with the failure. Columns 6, 8, and 11 relate with ways to avoid/reduce the failure from occurring and a "recommended action" of improving the design with the results to "reduce risk and increase customer satisfaction" [5].

Table 1: FMEA Form Structure

Item / Func	**Pot. Failure Mode**	**Pot. Effects**	**Sev**	**Pot. Causes**	**Prevention Controls**	**Occ**	**Detection Controls**	**Det**	**RPN**	**Rec. Action**
Item1/ Fnc1	Fail1	…	#	Cause	…	#	…	#	#	…
	Fail2	…	#	Cause	…	#	…	#	#	…
Item1/ Fnc2	Fail1	…	#	Cause	…	#	…	#	#	…
		…	#	Cause	…	#			#	…
	Fail2	…	#	Cause	…	#	…	#	#	…
		…	#		…	#			#	…
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*

Every failure mode has an associated Risk Priority Number (RPN) to quickly show which item needs to be prioritize during the design and validation phase. The RPN is a number from 0 to 1000 that showcases the importance of the potential effects by multiplying the severity rank, occurrence rank, and detection rank together. Each effect

has a specific severity rank and occurrence rank and each detection control has a specific detection rank (all ranks are from zero to ten). There are common industry rankings for each section that can be found in Appendix C: Table 2, Table 3, and Table 4, but it is recommended to develop and produce a separate internal ranking system to clearly prioritize the failures associated with the system.

2.2. Model Based Design

Model based design is commonly used amongst automotive industries and automotive related design projects. Past research articles have involved integrating and testing faults within components from cyber-physical systems such as electronic control units (ECUs) to investigating faults within a component specific system such as a hybrid electric vehicle inverter [6] [7] [8]. All model-based designs have the process of using X-in-the-Loop, where 'X' refers to any testing environment which can be model, software, or hardware [9]. X-in-the-Loop (XIL) systems are useful for developing the testing environment for a system in initial/early design phases, represented as high-fidelity models, to inserting electrical faults within the specific physical hardware. The three primary systems are Model-in-the-Loop (MIL), Software-in-the-Loop (SIL), and Hardware-in-the-Loop (HIL). XIL allows for the slow and easy transition of testing a component to be incorporated into the entire system. Each XIL has different benefits and shortcoming that are addressed in the following sections. Chapter 3, Section 3.3 will showcase an example for using the different XIL systems in order to slowly incorporate the design of the vehicle's controller.

2.2.1. Model-in-the-Loop (MIL)

Model-in-the-Loop (MIL) is a simulation environment, solely on a computer's software, that represents the component and/or system behavior. These models consist of no wires or physical connection, but are all internal to the software on the computer, which can be MATLAB/Simulink, ASCET [9], or ASM [6]. A MIL environment is decided with a use case in mind. A use case could be to quickly design the internal ECU of the physical component and test its interaction with a modeled plant environment, all without physically hooking up any hardware. Another use case example would be to represent high fidelity dynamics of an engine. The plant is typically developed separately from the controller and validated from previous physical testing. The shortcomings of a MIL environment are that it's not a perfect representation of the real-world environment. Often, as there is an increased model fidelity, simulation time of the model must be taken into consideration. Simulation time can be a benefit if it's faster than real time or a hinderance if the simulation requires heavy computation. For model-based design, MIL is often used as a baseline for calibrating parameters and quickly testing conceptional algorithms in appropriate fidelity environment.

2.2.2. Software-in-the-Loop (SIL)

SIL is the next logical step in the XIL process. SIL is the in-between stage from MIL and HIL simulation and is used to simulate the combination of internal software tools (such as Simulink [9]) and the c code (compiled code) for the designed hardware. C code is the modeled data of the component, written in the coding language C, that is used

to be flashed into the physical hardware being designed and developed. The primary purpose of SIL is to verify that the c code is constructed correctly and that the model runs properly before flashing the code into the hardware. If the c code is not being personally written and is instead compiled from built in software, like Simulink, then SIL can be skipped [8, 10].

2.2.3. Hardware-in-the-Loop (HIL)

HIL is the final step of the XIL process. HIL is running the physical hardware, hence the name, in conjunction with a HIL simulator, such as dSPACE [10] or TTEthernet [8]. A HIL simulator acts as the plant and communicates with the component's ECU via electrical communications. The simulator can be thought as tricking the ECU to believe it is in an actual vehicle. The user is allowed to adjust what messages and signals are being communicated from the simulator and test whether the ECU is properly responding to those signals. HIL allows for design engineers to test and send faults within the designed hardware without having to produce the fault case in the actual system. This produces a safe and efficient testing method, that could otherwise be harmful to the component and/or operator.

Chapter 3. Methodology

Before beginning to insert a fault into the model, it is important to first build a strong foundation and understand what faults are to be delivered and tested. For guidance on how the work was conducted for this thesis, Figure 4 shows a V-diagram specifically for the fault diagnostic development. The V-diagram is conducted identically to a standard V-diagram, starting at the top left (high level) and working to the right and downwards (more component specific, low level) and back up again. Following the current industry test procedure, it is important to first start with the development of requirements, as highlighted in Chapter 2. Requirements are developed from the first three blocks using FTA and FMEA, and then work up the V-diagram by verifying each level of requirements.

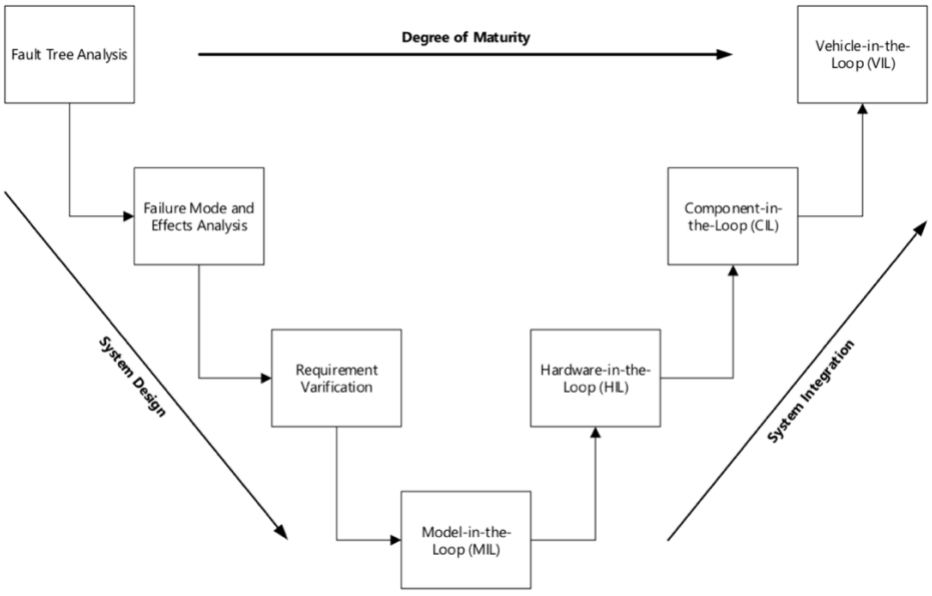

Figure 4 Research Specific V-Diagram

Going back up the V-diagram are four specific In-the-loop systems; Model-in-the-Loop (MIL), Hardware-in-the-Loop (HIL), Component-in-the-Loop (CIL), and Vehicle-in-the-Loop (VIL). The development and design of the MIL environment is done in parallel with the development and incorporation of requirements. The MIL model is structured to incorporate components and their interaction with the hybrid supervisor controller inside a simulation. HIL will be the primary focus of this thesis and takes the developed faults and physically tests those faults with the controller via electrical connections. CIL and VIL are an additional check for validating the design of the controller by testing the controller's interaction with other physical components in a safe environment. All In-the-Loop systems will be discussed in more detail further in this Chapter, Section 3.3.

3.1. Vehicle Architecture

Before building requirements for the design of the vehicle's main controller, dSPACE MicroAutoBox (MABx), it is important to lay out the components the controller will be interacting with in the vehicle. Laying out and understanding the structure of the system, and its associated components, allows visualization of the system to help with the development of potential failures or faults that could occur. Figure 5 shows OSU EcoCAR vehicle architecture, while Figure 6 highlights the controller/component interaction within the vehicle.

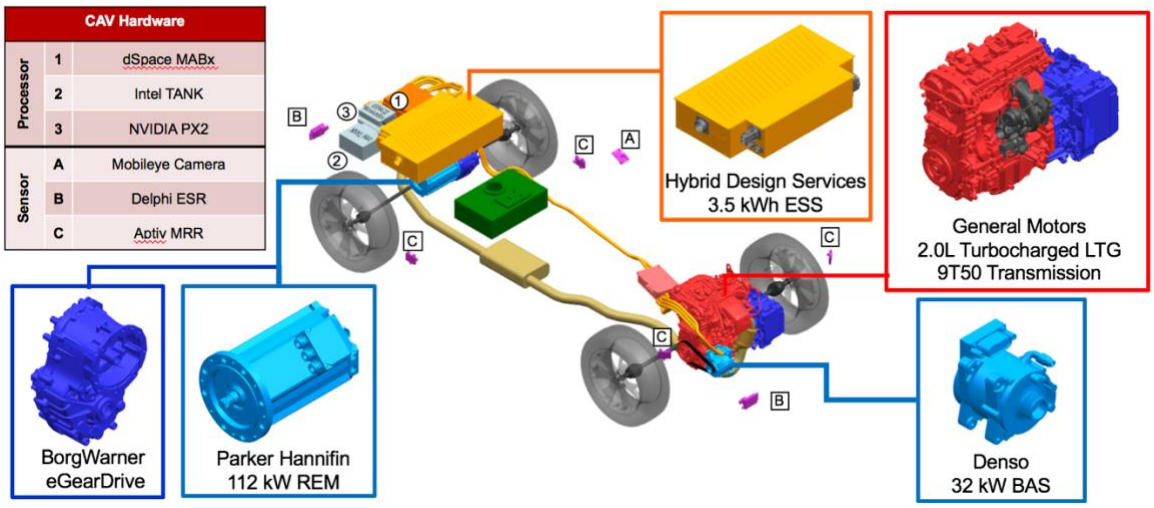

Figure 5 OSU EcoCAR Vehicle Architecture

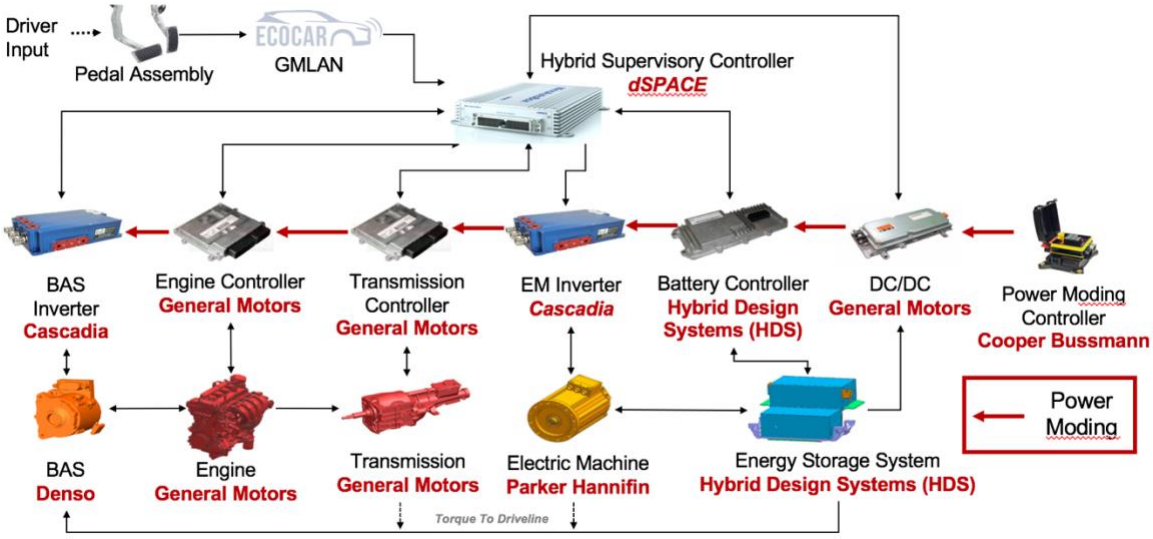

Figure 6 Vehicle Component Interaction Diagram

It is not only important to understand issues with communication from the controller to the vehicle's subcomponent ECUs, but to also address issues with how the ECUs interact with their physical component. This may seem as an obvious statement to

make but it is still addressed in this thesis because it can be a simple concept to overlook. Understanding the interaction with the subcomponent ECUs and their physical subsystem, ensures that during the requirement development phase that physical implementations of the controller are not overlooked.

Take for example the requirement developed from the interface between the MABx and the Electric Machine Inverter, Figure 6. If the physical Electrical Machine, the motor, was not considered while developing the requirements then this would leave for a flawed design and potentially cause the motor to do harm to the vehicle, the person driving, or damage the component. The process can continually break down further, going as far as to incorporate the bolts inside the motor, but clearly this would be outside the scope of the development for the MABx controller, hence it is ignored.

3.2. Requirement Development

The next step after developing the vehicle's architecture and subsystem interaction is to use this information to develop requirements. Requirements are an ongoing process that are continually being updated and modified as engineers learn more about their system. There are multiple tools to aid in the process of developing requirements but for this thesis two primarily tools were used: Fault Tree Analysis (FTA) and Failure Mode and Effect Analysis (FMEA). Additionally, requirements were produced and stored inside a requirements trackability matrix (RTM), which is continuously updated from the entire EcoCAR team. This central location allows all team engineers to know the software, hardware, and system requirements.

3.2.1. Fault Tree Analysis (FTA)

Fault Tree Analysis is an industry wide method for constructing a visual representation that aids in the development of requirements. FTA can be thought of going one step further from the vehicle architecture, since an FTA shows what component are the cause of a higher-level system failure. As described in Chapter 2, the FTA starts from the top level and works downward into more specific circumstances or lower level subsystems/components. Before constructing an FTA analysis, it is important to first identify the objective for the analysis. The objective helps keep the analysis on track and prevents unnecessary bombardment of unrelated information, allowing the FTA to be an efficient use of time. The objective for every FTA conducted for this thesis was to gain a more general understanding of components and their interactions with the vehicle controllers, component controllers, and hybrid supervisory controller.

The second step, defining the top event/failure, typically comes from experience. Fortunately, since EcoCAR has a partnership with General Motors (GM), GM provided a list of common top-level automotive industry faults. The list was the following:

- Inadequate/delayed loss of vehicle deceleration including malfunction within the regen braking system.
- Unintended acceleration
- Unintended longitudinal motion; unintended vehicle motion (rollaway)
- Unintended travel in the wrong direction, unintended propulsion flow
- Unintended or loss of lateral motion (includes locked steering)
- Unintended deceleration

- Loss or degradation of acceleration; loss or degradation of propulsion (e.g., stall)
- Unintended release of thermal energy causing burns or fire
- Unintended exposure to high voltage energy system (shock)
- Unintended exposure to toxic / flammable chemicals (gas/liquid)
- Unintended access to rotating or moving components (e.g., engine start)

The final steps to be conducted before actually constructing the fault tree include defining the scope, resolution, and ground rules. The scope and resolution limited the fault tree to the components shown in Figure 6. The ground rules included using only the standard block schematics found in Appendix B: Figure 29.

The next step in the process is to physically construct the fault tree diagram. The diagram in constructed by taking a common fault and using that as the starting point for the fault tree. Figure 7 shows an example of an FTA for a vehicle deceleration failure. This is a high level "common" fault from the GM provided categorized failure list.

The common fault, in red, is always addressed first and broken down to more specific components, in orange. The FTA has three color coding: red, orange, and blue. Red represents the highest-level event, orange indicates additional lower lever events will follow, and blue represents the lowest level event for that analysis. Each block can typically be considered as either an AND gate or an OR gate. An AND gate means that **all** the following events must occur for the higher event to take place, and an OR gate means that if **any** of the following events occur then the higher event will take place. The event is shown to be a OR gate by a circular arc underneath the event block; an AND gate would have a straight line. Since, the blue events are the lowest level, they do not

associate with being an AND or OR gate. For a better understanding of the block and color identification see Appendix B.

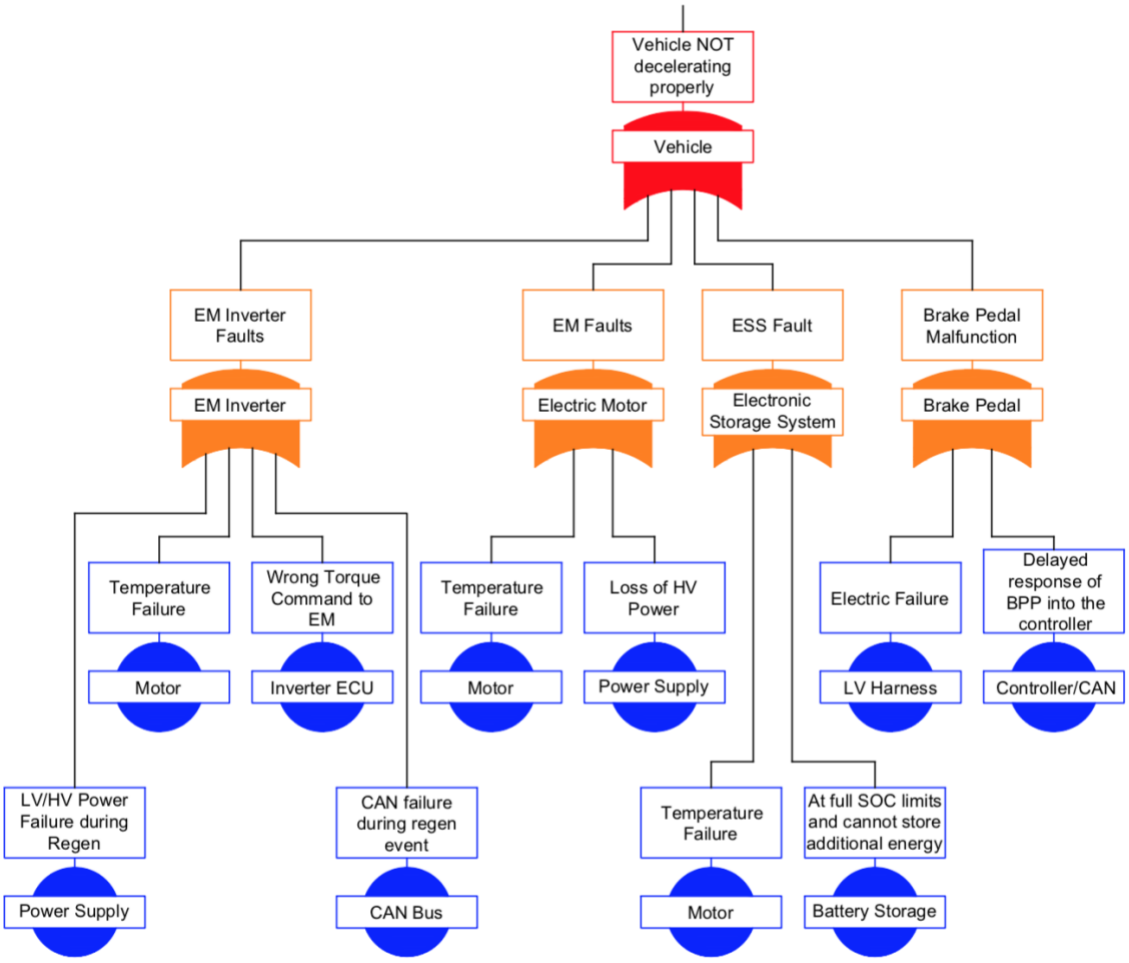

Figure 7: Fault Tree Analysis (FTA) for a Deceleration Failure

The FTA shows the different component level issues that are associated with the full vehicle system failure. The FTA example showed that when designing the controller and considering faults with improper vehicle deceleration, it is important to test electrical

faults from the 12V battery supply, motor, inverter, CAN bus, LV Harness, and Energy Storage System (ESS). Hence the FTA provided a list of components for developing the fault requirements associated with a full-level vehicle braking failure.

3.2.2. Failure Mode and Effects Analysis

The Failure Mode and Effects Analysis (FMEA) takes what was learned from the FTA and breaks it down further. Continuing with the deceleration failure example, an FMEA guides the user to incorporate these specific component failures from the FTA into the FMEA document for further, more detailed, review. Figure 8 shows the filled out FMEA document from the conducted FTA. The first column is filled out with each component, derived from the FTA, and the function for that component is broken up as a high-level function since the analysis was conducted for a high-level system. Since the functions are high-level, the standard for making measurable functions had to be overlook for this analysis. Each column was filled out according to the potential failure (typically shown from FTA), potential causes of failure, and prevention and detection controls. Each situation was raked accordingly from the EMC rankings, found in Appendix C: Table 5-Table 7, that were specifically created for this FMEA.

Item/Function	Potential Failure Mode	Potential Effect(s) of Failure	Sev	Potential Cause/ Mechanism of Failure	Prevention Controls	Occ	Detection Controls	Det	RPN	Recommended Action
CAN Bus: Transmits Communication from BCM to Controller	No Communication Occurs - Vehicle does NOT deaccelerate	Potential Servere Crash and Death or Injury	10	Large Electrical Noise in CAN bus	Terminating Resistors	3	Controller checks Communication from BCM	3	90	Driving is alerted of the situation and is highly requested to have the vehicle repaired
CAN Bus: Transmits Communication from Controller to Inverter	CAN fails to transmit information	Motor remains in Neutral	3	Large Electrical Noise in CAN bus	Terminating Resistors	1	Controller checks if CAN bus is active; validate Motor Speed with signal requet	5	15	Light indicator or warning for CAN bus failure
ESS: Powers the EM Inverter	Stops supplying Power to the EM Inverter	Rear Wheels Do NOT Provide Forward Motion	3	Disconection or shortage of the Electrical Harness	Design Harness to remove pinch points	3	Controller checks Communication from EM Inverter	5	45	Driving is alerted of the situation and is highly requested to have the vehicle repaired
		Rear Wheels Do NOT Regen or Help Slow the Vehicle	3		Harness built to be robust				45	
ESS: Provide HV Power supply for Vehicle	HV Power becomes dangerous and thermal runaway occurs	Potential Death and Complete Destruction of Vehicle	10	Temperature rises above battery rating	Disconection of ESS from system	3	BMS tracks ESS Temperature	7	210	Controller requests to disconect
				Current Continues to be drawn durring an ESS fault	Disconection of ESS from system	3	Controller checks that current is removed from ESS	7	210	Controller designed to remove current draw from ESS during an ESS fault
Inverter: Commands the Motor Torque	Wronge torque command sent to EM	Motor Accelerates instead of slowing down	9	Inadiquate Testing	Properly Test	1	Physical Interaction with Vehicle	1	9	Controller torque requests are tested to properly function for the lifetime of the vehicle
		Motor Deaccelerates instead of speed up	7						7	
LV Harness: Provides Communication from BPP to BCM	No Communication Occurs - Vehicle does NOT deaccelerate	Potential Servere Crash and Death or Injury	10	Disconection or shortage of the Electrical Harness	Harness built to be robust	1	Physical Inspection	2	20	Driving is alerted of the situation and is highly requested to have the vehicle repaired
Motor: Drives the rear wheels	Motor Over Heats	Motor Stops Providing Rear Wheel Forward Motion	3	Excessive amount of Current being supplied to Motor	Read Motor Temperature and Moderate Current Supply	3	Controller checks Temperature Rating from EM Inverter	7	63	Controller removes current draw when receiving a Motor fault from the Inverter
		Rear Wheels Do NOT Regen or Help Slow the Vehicle	3						63	

Figure 8 FMEA from FTA Deceleration Failure

The RPN provides which component and failure needs to be prioritize during testing. The ESS came on top with an RPN of 210 and was further looking into detail for deriving a test case. The recommendation for preventing the main ESS failure was for the controller to remove any current being sent to the ESS when receiving an ESS fault. The recommendation guided the team for developing the following requirement:

"The absolute value of the ESS current shall not be less than 5 Amps within 5 seconds of an ESS fault detection".

The requirement will be used to create a test case for when the controller is tested inside the HIL simulation rack, the full example of this test case will cover in Chapter 4. The additional recommendations from the FMEA allowed for the development of additional requirements to be stored inside the RTM.

3.3. In-the-Loop System

In-the-Loop systems are commonly used when conducting a model-based design. As discussed in Chapter 2, these In-the-Loop systems are known as Model-in-the-Loop (MIL), Software-in-the-Loop (SIL), and Hardware-in-the-Loop (HIL). EcoCAR additionally adds two In-the-Loop systems; Component-in-the-Loop (CIL) and Vehicle-in-the-Loop (VIL). Though, all these In-the-Loop systems will be touched on in this section, it is important to point out that HIL will be highly covered since HIL is the primary method incorporated with this research.

3.3.1. Model-in-the-Loop (MIL)

Model-in-the-Loop is a way to model the entire vehicle via a simulation. The model is developed from MathWorks's software, Simulink, to incorporates the full vehicle model. This full vehicle model simulates all major powertrain components for various drive cycles. MIL is used to develop the logic for tested failures by sending error/warning signals to the controller model and verifying its controller response. The MIL model is shown in Figure 9 where it clearly shows the Plant (the simulated vehicle), the driver model, and the vehicle's controller. SIL is overlooked for the transition from MIL to HIL because Simulink provides a built-in c-code compiler, which makes SIL unnecessary.

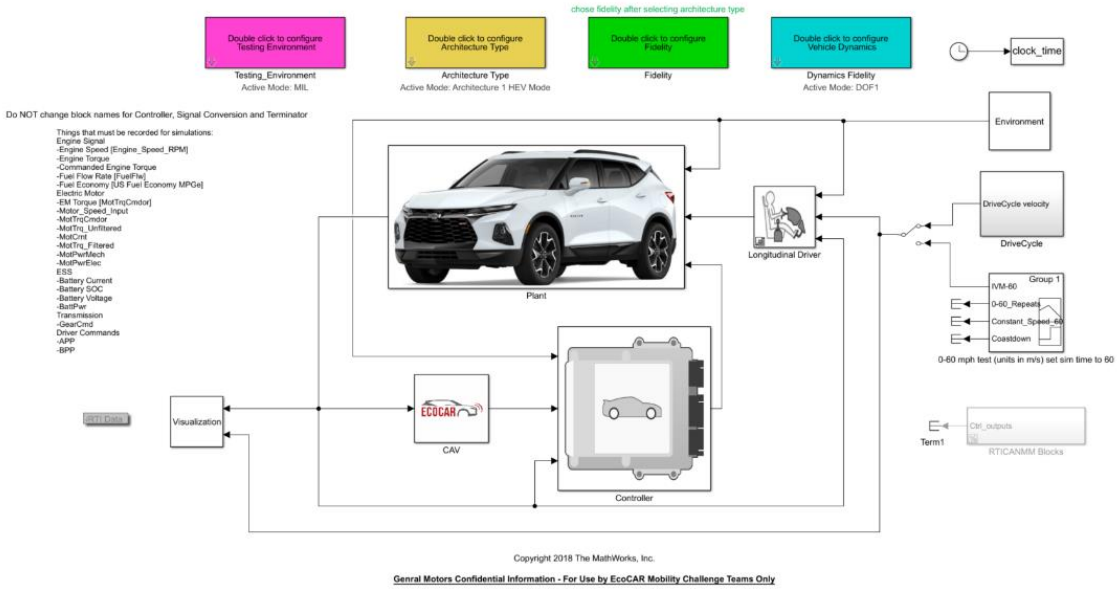

Figure 9: EcoCAR's Model-in-the-Loop Simulink Model

3.3.2. Hardware-in-the-Loop (HIL)

Unlike MIL, HIL goes one step further by physically testing and checking the communication from the plant to the controller. The design of the hybrid supervisory controller known as the dSPACE MABx can then be tested to ensure in real time that the signals being transmitted are as expected. The MABx is connected to a dSPACE mid-size HIL simulator by physical connections. The dSPACE mid-size HIL simulator incorporates a licensed program, ControlDesk, which enables tracking serial communication in real time. An example and more information about ControlDesk will be covered in Chapter 4.

The Simulink model, Figure 9, allows for a quick transition from MIL to HIL by double clicking and changing the testing environment mask. The transition keeps the MIL based model algorithm for both the plant and the controller, and only changes the input and output layers. Quick validation for MIL and HIL is able to be conducted since both the MIL and HIL models have the same base algorithm. The input and output layers change the communication from virtual to electrical, when transitioning over to HIL. The HIL input and output layers incorporate CAN communication, enabling multiple signals and messages to be sent through the bus.

The HIL structure allows for sending a failure/error message to the controller from the modelled plant and checks that during a simulated drive cycle the controller acts as expected. The developed requirements, stored in the RTM, are used to check and validate that the controller passed the requirement from receiving the fault message. The process is continued to check the controller satisfies each requirement.

3.3.3. Component-in-the-Loop (CIL) and Vehicle-in-the-Loop (VIL)

CIL and VIL are additional X-in-the-Loop systems that are essential to conduct prior to the release of the designed vehicle. Both systems ensure that all components interact with one another as intended by integrating the physical components together. The components are typically integrated into the vehicle and tested for functionality and the vehicle's performance. Since both of these systems consist of physical interaction and not solely electrical, like HIL, a replication of testing component failure would cause hardware damage, hence fault insertion would not be wise for these methods. These systems are addressed in this thesis for completion and to provide further information regarding additional X-in-the-Loop systems that are to be conducted before the release of the designed controller.

CIL takes either a specific component or a portion of the entire full-vehicle system and tests for functionality as well as how the subsystem components interact with one another. The subsystem can be tested on a dynamometer by disconnecting part of the full-vehicle system and only using the installed section. Figure 10 showed testing the engine/ transmission subsystem in the vehicle without having the rear electric motor incorporated.

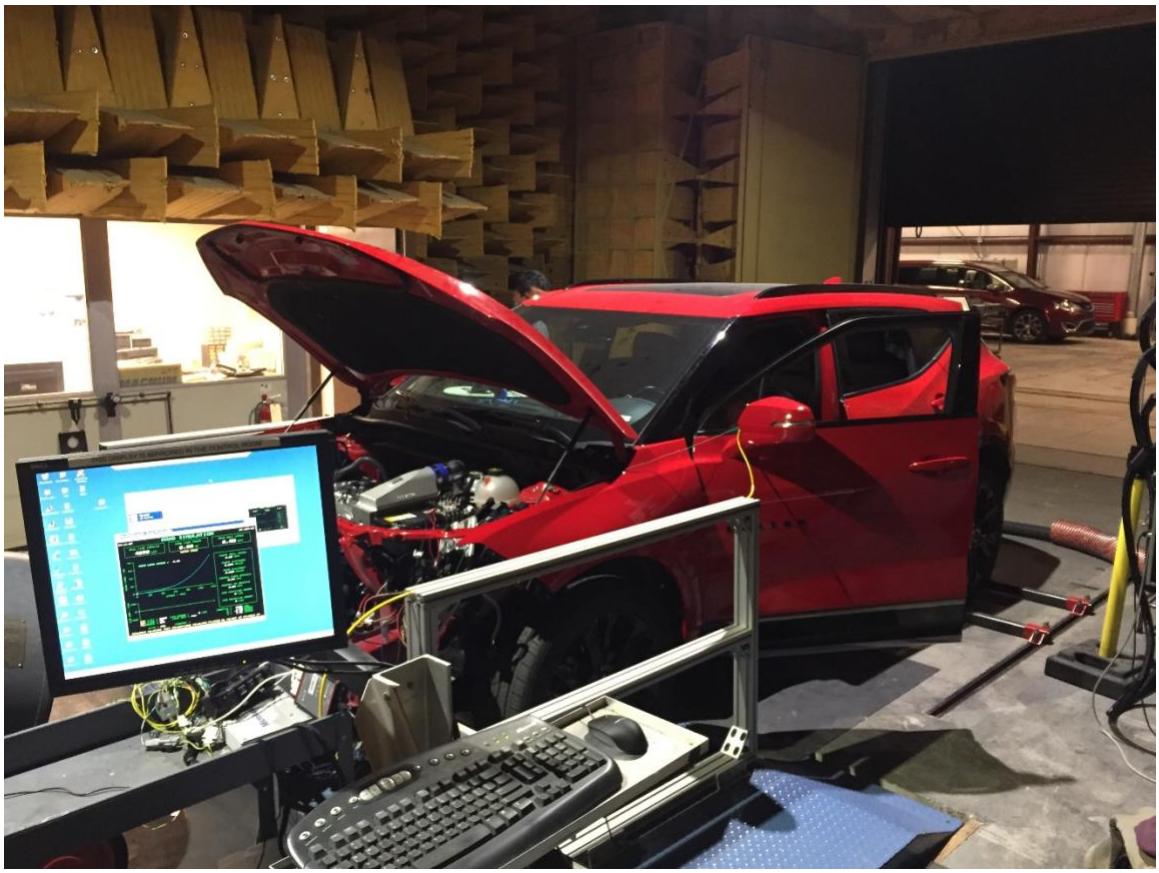

Figure 10: OSU EcoCAR Engine Component in the Loop Testing

VIL, one step further than CIL, consists of testing the entire vehicle with all the physical components installed on the vehicle. The vehicle can be tested either at a testing facility like TRC or on an AWD dynamometer. This is the final step of ensuring the components function safely and properly before releasing to the public.

Chapter 4. Implementations

In order to test that a fault is being transmitted, it is necessary to ensure that the controller both reads either a warning/error or fault message and responds with an appropriate control action. This process needs to be adequately documented and done in real time to track that the response rate set from requirements are satisfied. There are primarily three software interfaces that are used for this research, MATLAB/Simulink, dSPACE ControlDesk and dSPACE AutomationDesk. dSPACE Synect, a test management software, was originally planned to be added to these software toolboxes, but due to unexpected circumstances this software wasn't able to be incorporated.

This Chapter's main focus will be on the different software that were incorporated into this thesis. To aid in the understanding of the different software, a fault pertaining to the ESS will be inserted and tested. The ESS was chosen first because it is rated as a high-risk component that additionally lacks proper testing and was rated high from the FMEA conducted in Chapter 3. The requirement to be tested for the ESS was framed as follows:

"The absolute value of the ESS current shall not be less than 5 Amps within 5 seconds of an ESS fault detection".

All other requirements are stored inside the requirement traceability matrix (RTM) and the RTM is used to check that all requirements related with controller communication are satisfied.

4.1. Simulink

Simulink, developed and produced by MathWorks, is a simulation software toolbox that allows for the construction of a model that can be ran and tested. Simulink is used for the construction of the full vehicle model MIL and HIL model. Chapter 3 goes into greater detail about MIL and HIL and how MIL is used specifically with Simulink. The HIL model, as described in Chapter 3, takes the base algorithms from the MIL plant and controller and only changes the input and output (communication) algorithms. Figure 11 helps to demonstrate this concept by highlighting and showing the HIL model layout from a portion of the Simulink Model. Boxed in red are the communication layers/algorithms that are internally changed when transitioning between the MIL & HIL environment, while the main internal controller, boxed in black, remains constant.

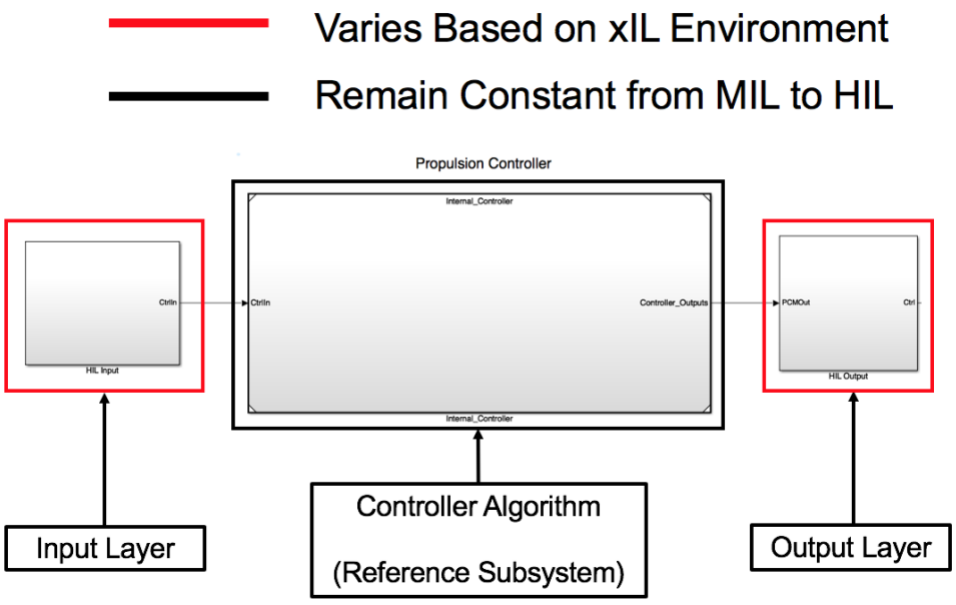

Figure 11 Controller Layout from HIL Simulink Model

Simulink is then used to prepare the HIL model for testing and inserting faults. The HIL model signals are prepped by enabling global data export for the signals associated with either the plant's warning or error CAN signals. Exporting data globally on these signals allows for ControlDesk to manipulate these signals, hence test that the controller both receives and responds according to the modified fault message.

Continuing with the ESS fault case, the ESS error and warning Boolean messages are exported to enable modification in ControlDesk. This gives the user the ability to track in ControlDesk whether the ESS requirement is satisfied. Figure 12 shows the physical layout of various plant CAN signals that are being exported for ControlDesk manipulation. The signals that are enabled are indicated by being underlined in red and from the blue wireless output icon shown above the V_ESS_Warning_Bool_NA and V_ESS_Error_Bool_NA signals. The signals will additionally need to be exported globally for ControlDesk to manipulate and read these signals.

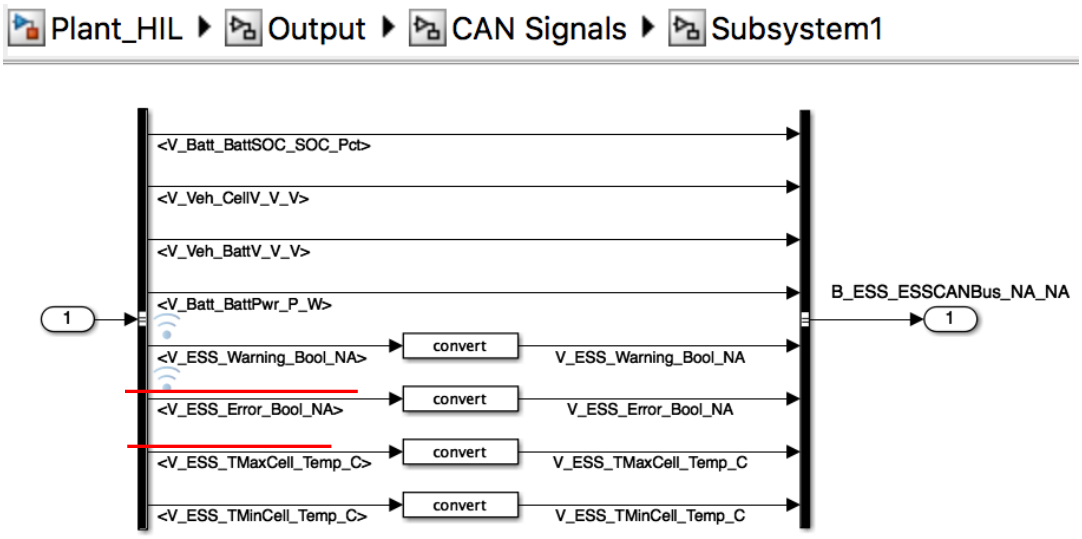

Figure 12 Simulink Plant Output CAN signals – ESS Fault Enabling

The final step for Simulink is to compile the full vehicle model into compiled code (c-code). The plant and controller being two different reference models are compiled separately. The compiled controller model and the plant model c-codes are flashed into the controller and the HIL Simulator respectively. The HIL simulator acts as the plant for the controller and both systems are connected by electrical signals that communicate via CAN. CAN communication is created in the Simulink input and output algorithm blocks using the RTI CAN Multimessage block sets. The blocks sets are specifically developed to enable interface of dSPACE hardware products such as the MABx & HIL Simulator using MATLAB/Simulink. The RTI block set allows for quick CAN modifications and works in conjunction with the Simulink code compiler.

4.2. ControlDesk

ControlDesk is a software, made by dSPACE, that contains a user-friendly interface for tracking communication and signal values in real time. ControlDesk takes the compiled code generated by Simulink and flashes the C-codes for the controller and the plant onto their respective hardware. All specified signals can be tracked and modified during a run cycle, enabling ControlDesk to be a vital tool for checking communication, inserting a fault and diagnosing the system response. The triggered error or warning signal is sent from the HIL plant to the MABx and checks if the controller receives the signal and correctly responds, passing the requirement criteria. For organizing and simplifying the test procedure, three layouts were created: the dashboard, calibration window, and diagnostic layout.

4.2.1. The Dashboard Layout

The dashboard layout is exactly what it sounds like: a representation of a vehicle's dashboard during simulation. Figure 13 shows the active dashboard layout running a drive cycle. It was initially created to test a failure with the electric storage system (ESS). The far-left side, the SimState block, gives the tester the ability to stop, pause, and start the simulated drive cycle. The middle section shows signals, represented as a dashboard, being transmitted during the cycle and is used for a visual check that the signal values are updating correctly. The right side provides a user interface with togglable switches, for triggering different vehicles functions while additionally giving the user the ability to insert and test an ESS fault. Underneath the ESS fault block is a display showing the drive trace that visibly updates in real time.

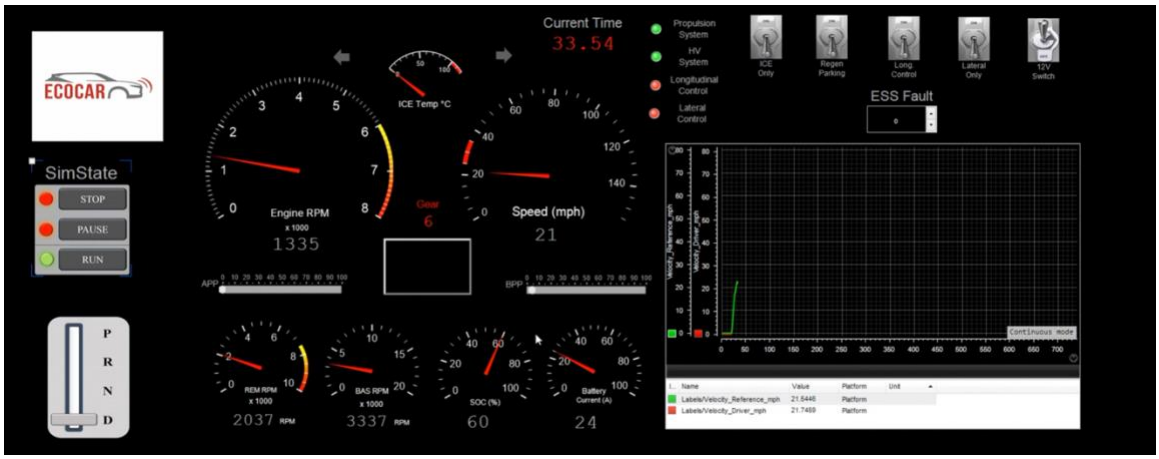

Figure 13 Dashboard Layout Running Drive Cycle

The ESS fault is togglable in this layout and gives the user the ability to see how the vehicle response when the fault is active. Here the user can validate that the vehicle

performs as expected for as quick visual check. ControlDesk additionally records the different ESS current value and time stamp so the user can ensure that the requirement is being passed.

4.2.2. Calibration Layout

The calibration layout allows for physical alterations and calibrations to be made to the controller in real time. All calibration changes will not be permanently stored inside the controller but still aids in the design and development process. Figure 14 shows the calibration layout that was constructed for the MABx controller. The Calibration layout is not intended for physically testing faults, but still remains a useful tab for designing the controller and setting/changing different controller attributes.

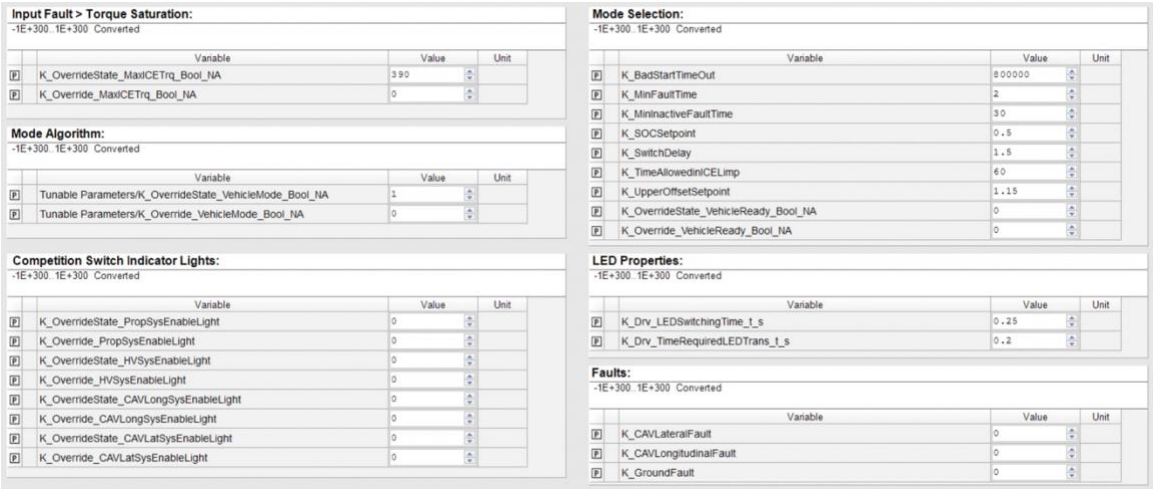

Figure 14 Calibration Layout for MABx Controller

4.2.3. Diagnostic Layout

The diagnostic layout is used for troubleshooting issues and changing signal/variable values. The CAN bus is verified to communicating properly from this layout and additionally shows important specified messages according to their corresponding CAN bus. CAN communication is verified by visually checking that the RX/TX time is updating. The CAN communication fault can also be triggered by unchecking the Global Enable checkbox. Additional messages and signals can be added to this layout for troubleshooting. Figure 15 shows a general diagnostic layout with every CAN bus being updated and validated for that window.

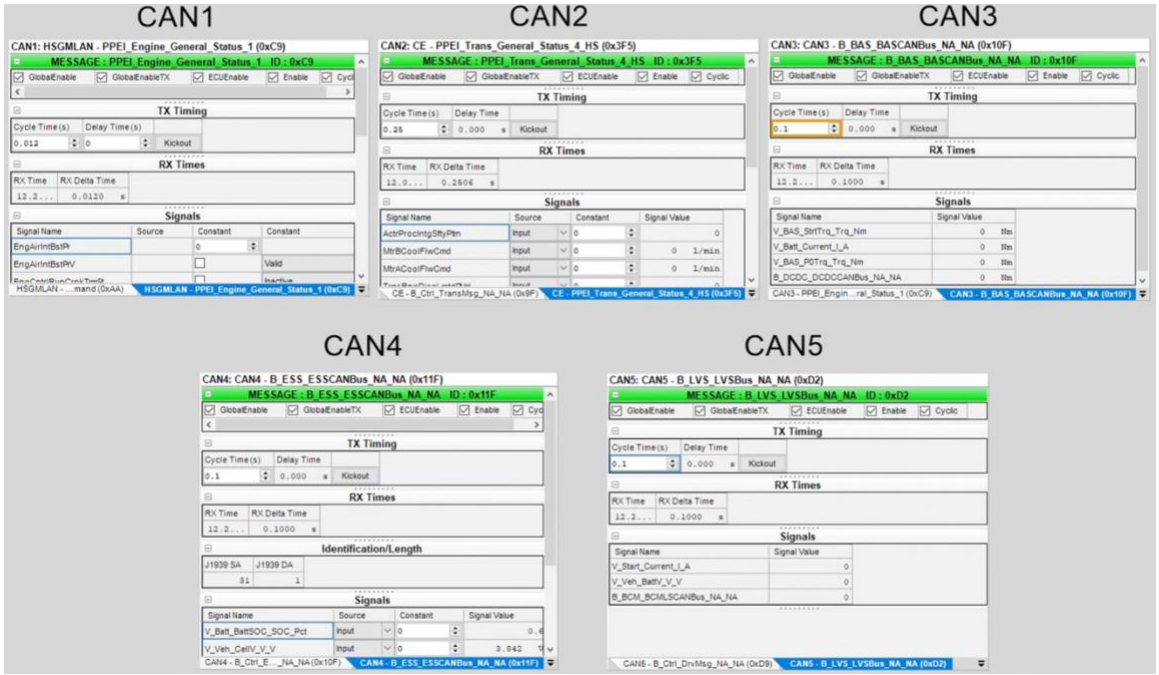

Figure 15 Generic Diagnostic Layout with CAN Communication Channels

The diagnostic tab can additionally be used for testing the ESS fault, similarly to the dashboard layout. The set up for the diagnostic tab is typically simpler than the dashboard layout hence making the diagnostic tab preferred when conducting quick requirement tests.

4.3. Automation Desk

AutomationDesk takes a list of conducted tests from ControlDesk and both collects and stores the results into a report format. The AutomationDesk code is designed and built to automate execution of test cases. These test cases are arranged in a hierarchy, with each test case being run through a sequence of blocks from top to bottom . The process is simply conducted by the click of a button and removes the need to manually initiate the execution of every test case. Since multiple components will have multiple requirements to be tested, AutomationDesk is essential for testing that all previous requirements are not affected while the controller is being designed to meet additional requirements. The automated test process can then be used to ensure that all past test cases remain valid during the development of the controller algorithm.

The ESS fault example is continued to be used here to walkthrough the benefits of using AutomationDesk. The ESS fault was tested for three different cases to ensure that the requirement continued to be satisfied during all three cases. The three cases involved inserting the ESS fault during a braking, accelerating, and coasting scenarios. The EcoCAR mobility challenge (EMC) drive trace was used for the diagnosis of a fault in the three scenarios considered. Figure 16 highlights the different portions of the drive trace where the faults were inserted. After the faults were inserted, AutomationDesk

checked that the current value was below 5 Amps after 5 seconds and reported whether the requirement was a pass or failure. The current and time values were set by the requirement stated in the beginning of the Chapter.

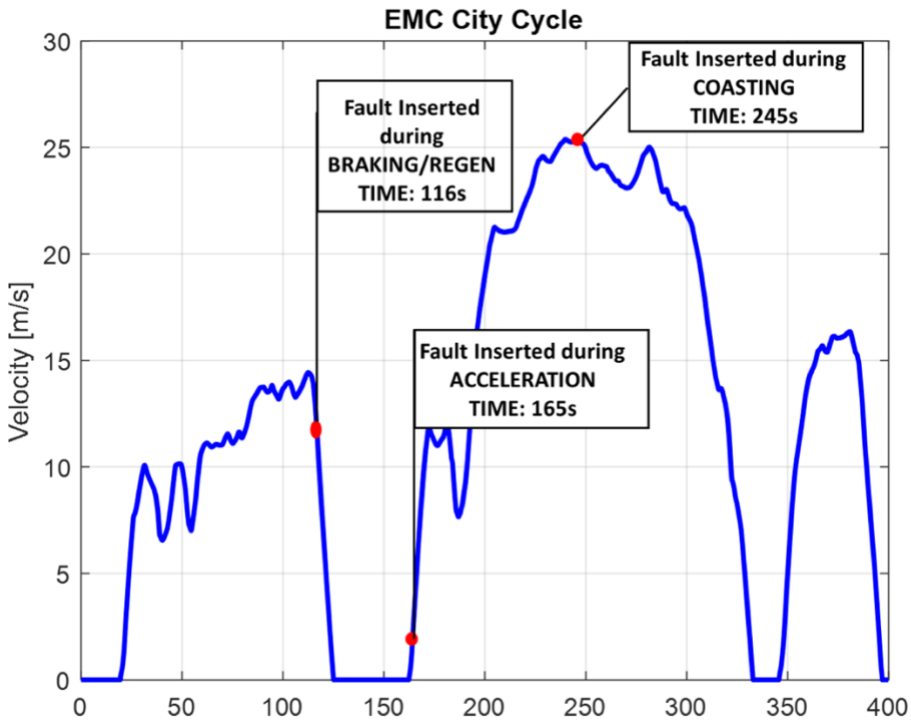

Figure 16 EMC Drive Trace w/ Highlighted ESS Fault Portions

AutomationDesk uses a block format for developing the sequential code that is ran for the process. The higher-level blocks consisted of starting and opening ControlDesk, running the Data Acquisition, and performing the Data Analysis to be stored inside the report. Figure 17 provides the AutomationDesk interface that has the ESS fault cases as described earlier. The left side shows a list of variables and folders that are stored inside AutomationDesk for organization and for reporting and checking the

different variable values. The right side is the programable interface that is organized by three compressed blocks for running the code.

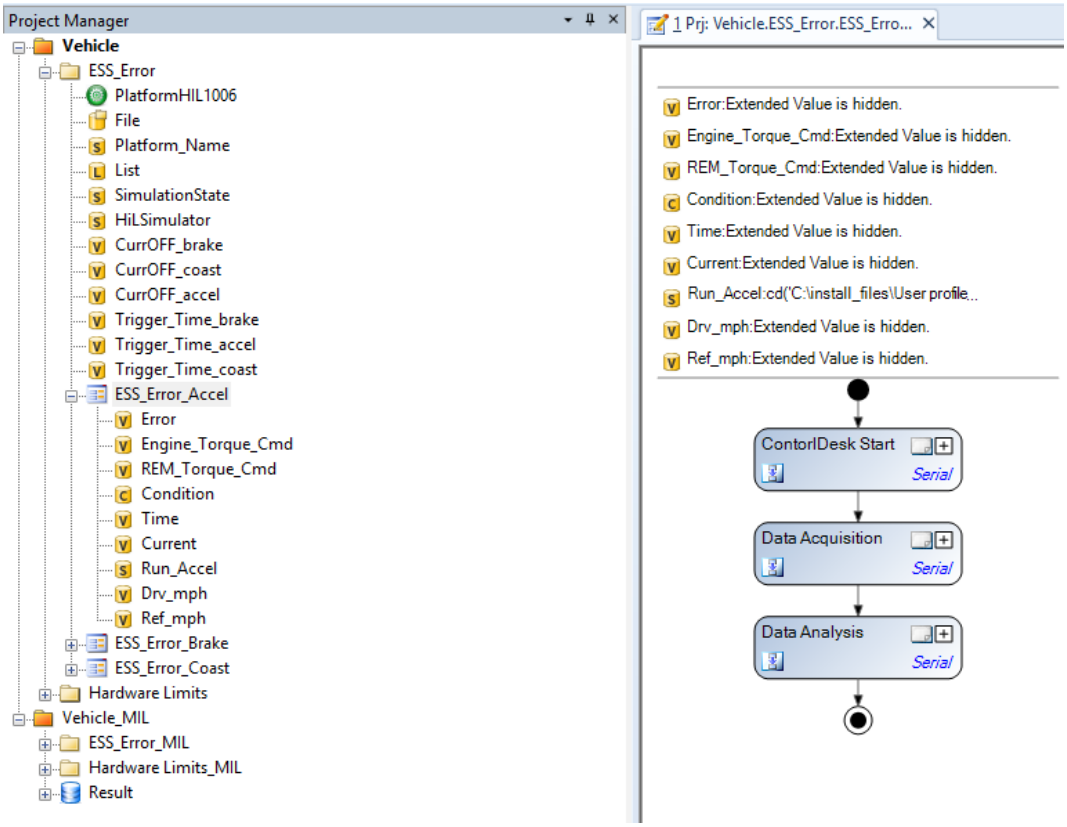

Figure 17 AutomationDesk Layout w/ ESS Fault Scenarios

Each block from the constructed code is separated primarily by its main task. The ControlDesk Start block simply opens and runs the ControlDesk model. The DataAcquisition block obtains the requested ControlDesk values and stores them in the variables. The DataAnalysis block checks that the requirement is satisfied with an if-else block for demonstrating if the criteria was a pass or fail, Figure 18.

The three test cases (fault during acceleration, braking, and coasting) are all identical except for the fault insertion time, which is modified inside the Data Acquisition Block, shown in Figure 19. Boxed in red is the error time variable for the when the fault test should start. The figure shows the value of 165 seconds which matches the acceleration time period from the EMC City Cycle, from Figure 16. Since all three test cases are primarily the same with slight modifications, this makes AutomationDesk a quick and easy tool for testing the different faults with the need to only make slight modifications.

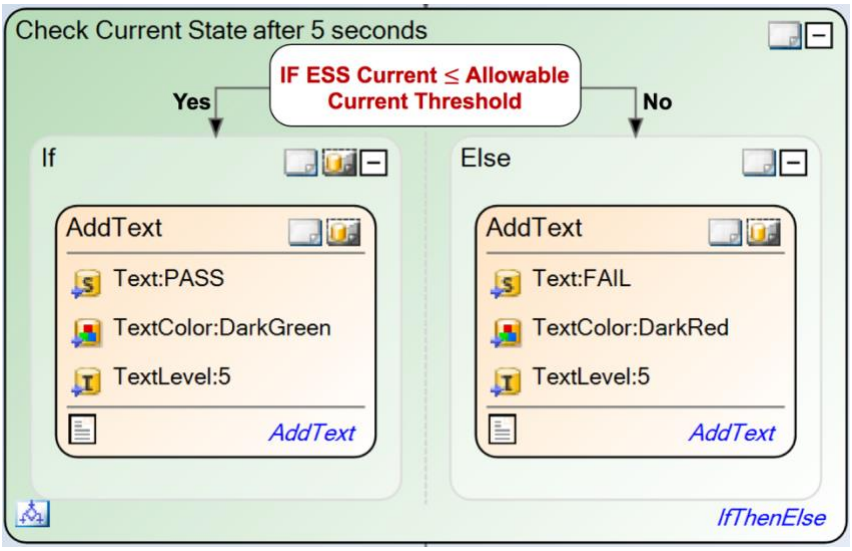

Figure 18 If-Else Block for Pass/Failure Criteria

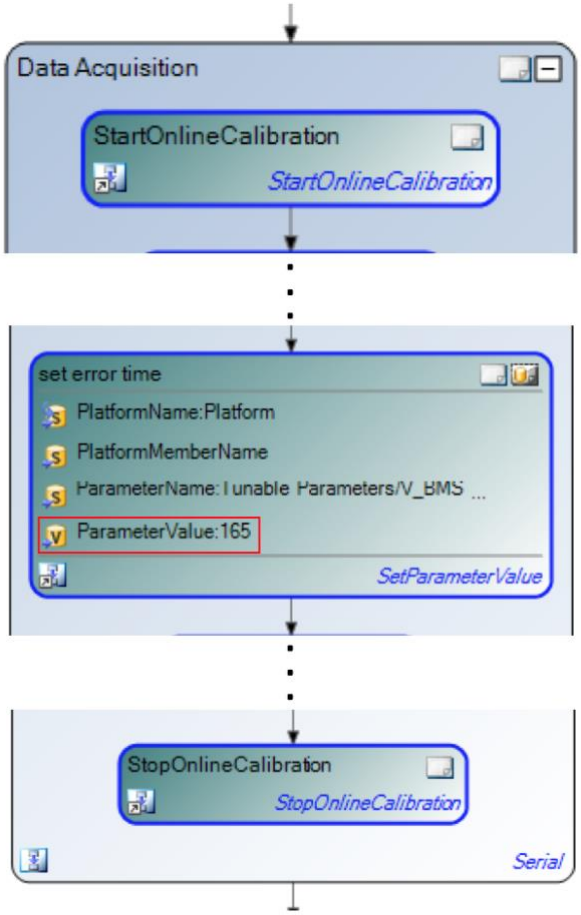

Figure 19 DataAcquisition Block – Set Error Time Section

Chapter 5. Results

The success for this research involved properly using the given software to construct a method for inserting a fault into HIL simulation. All previous Chapters went through the process of understanding and setting up of the software for the test cases, but the focus for this Chapter will be on the actual results obtained by the software. As discussed earlier the requirements set whether the test case passes or fails, which is internally stored inside the RTM. The RTM is an excel sheet of every requirement and test that has been conducted and verified, Figure 20.

The documented report from the AutomationDesk script - for the ESS fault scenario - successfully presented the results of the test. The following figures, Figure 21-Figure 26, show the reported graphs from the ESS fault scenarios; recall that these cases included inserting an ESS fault during an acceleration, braking, and coasting portion of the drive cycle. The current profile on each scenario was less than 5 Amperes within 5 seconds, meeting the requirement. AutomationDesk additionally tested cases for the different component limits and the entire results of the test are stored at the end of the report, Figure 27. A failure in any of the test will require the user to reevaluate the design of the hybrid supervisory controller and ensure that all requirements are satisfied.

		Source Traceability (Origin of the Safety Requirements)		Allocation Traceability		Requirement Type	Traceability To V&V Plans and Results				
ID	System Safety Requirement	Tools / Source	Additional Documentatio	Subsystem	Component	Hardware/Software/Both	Test ID	Test Environment	Test Status Pass/Fail/N Tested	Tester Nam	Version Tested/Date Stamp
1	The maximum engine torque demanded by the HSC shall be less than 390 Nm	Component Documentation		Hybrid_Supervisory_Control	Ouput Fault Detection	Software	T101	MIL, HIL	Pass	Hari Ranga	V_04 / 1/15/2020
2	The maximum engine speed resulting due to the ICE torque demand by the HSC shall be less than 6000 RPM	Component Documentation		Hybrid_Supervisory_Control	Ouput Fault Detection	Software	T102	MIL, HIL	Pass	Hari Ranga	V_04 / 1/15/2021
3	The maximum rear electric motor torque demanded by the HSC shall be less than 280 Nm	Component Documentation		Hybrid_Supervisory_Control	Ouput Fault Detection	Software	T201	MIL, HIL	Pass	Hari Ranga	V_04 / 1/15/2022
4	The maximum rear electric motor speed resulting due to the REM torque demand by the HSC shall be less than 12000 RPM	Component Documentation		Hybrid_Supervisory_Control	Ouput Fault Detection	Software	T202	MIL, HIL	Pass	Hari Ranga	V_04 / 1/15/2023
5	The maximum temperature of the rear electric motor windings shall be less than 190 deg F	Component Documentation		Hybrid_Supervisory_Control	Ouput Fault Detection	Software	T203	MIL, HIL	Pass	Hari Ranga	V_04 / 1/15/2024
6	The operating pack voltage of the Energy Storage system shall be more than 300 V and less than 390 V	Component Documentation		Hybrid_Supervisory_Control	Ouput Fault Detection	Software	T301	MIL, HIL	Pass	Hari Ranga	V_04 / 1/15/2025
7	The operating temperature of the Energy Storage system shall be more than 00 deg F and less than 300 deg F	Component Documentation		Hybrid_Supervisory_Control	Ouput Fault Detection	Software	T302	MIL, HIL	Pass	Hari Ranga	V_04 / 1/15/2026
8	The absolute value of current demand of the Energy Storage system shall be less 250 A	Component Documentation		Hybrid_Supervisory_Control	Ouput Fault Detection	Software	T303	MIL, HIL	Pass	Hari Ranga	V_04 / 1/15/2027
9	The state of charge of the Energy Storage system shall be more than 30% and more than 80%	Component Documentation		Hybrid_Supervisory_Control	Ouput Fault Detection	Software	T304	MIL, HIL	Pass	Hari Ranga	V_04 / 1/15/2028
10	The absolute value of the ESS current shall be less than 5 Amps within 4.5 seconds of an ESS fault detection.	Component Documentation		Hybrid_Supervisory_Control	Ouput Fault Detection	Software	T305, T306, T307	MIL, HIL	Pass	Hari Ranga	V_04 / 1/15/2029
11	The maximum belted alternator starter(BAS) torque demanded by the HSC shall be less than 280 Nm	Component Documentation		Hybrid_Supervisory_Control	Ouput Fault Detection	Software	T401	MIL, HIL	Pass	Hari Ranga	V_04 / 1/15/2030
12	The maximum belted alternator starter(BAS) speed resulting due to the BAS torque demand by the HSC shall be less than 21000 RPM	Component Documentation		Hybrid_Supervisory_Control	Ouput Fault Detection	Software	T402	MIL, HIL	Pass	Hari Ranga	V_04 / 1/15/2031
13	The maximum belted alternator starter(BAS) current resulting due to the BAS torque demand by the HSC shall be less than 200 A	Component Documentation		Hybrid_Supervisory_Control	Ouput Fault Detection	Software	T403	MIL, HIL	Pass	Hari Ranga	V_04 / 1/15/2032

Figure 20 RTM Excel Sheet

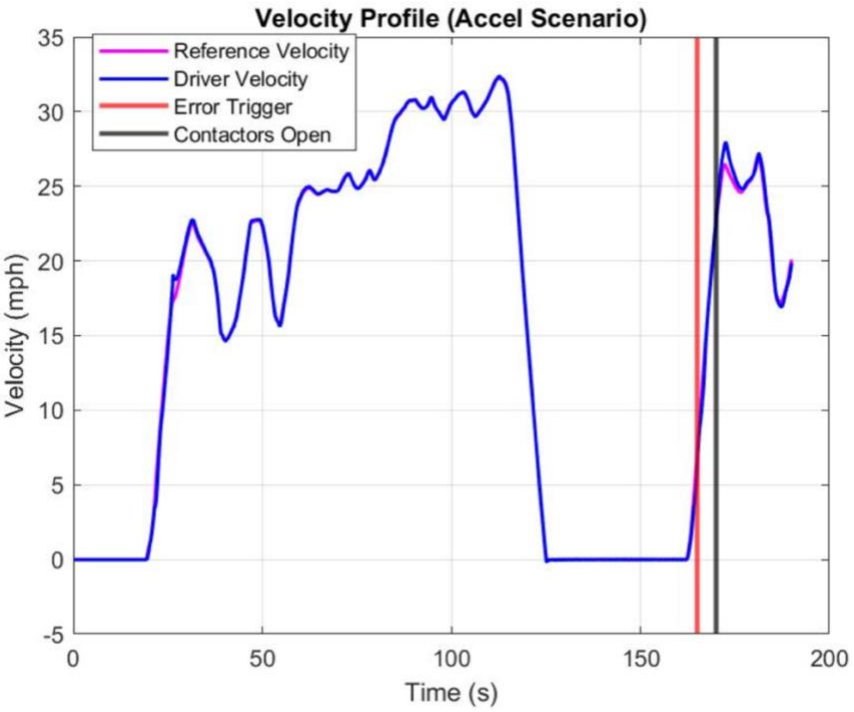

Figure 21 Drive Trace: ESS Fault Insertion During Acceleration

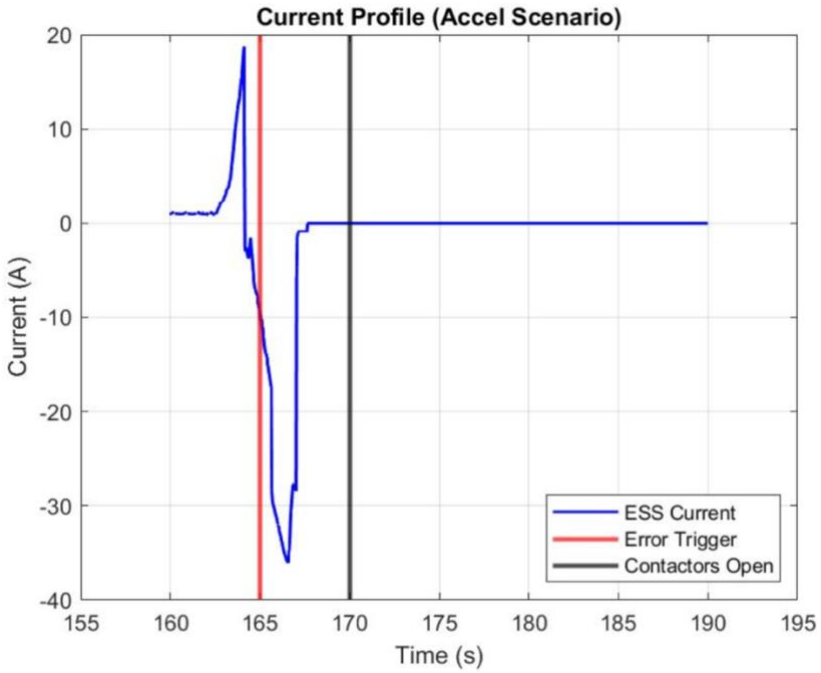

Figure 22 Current Trace: ESS Fault Insertion During Acceleration

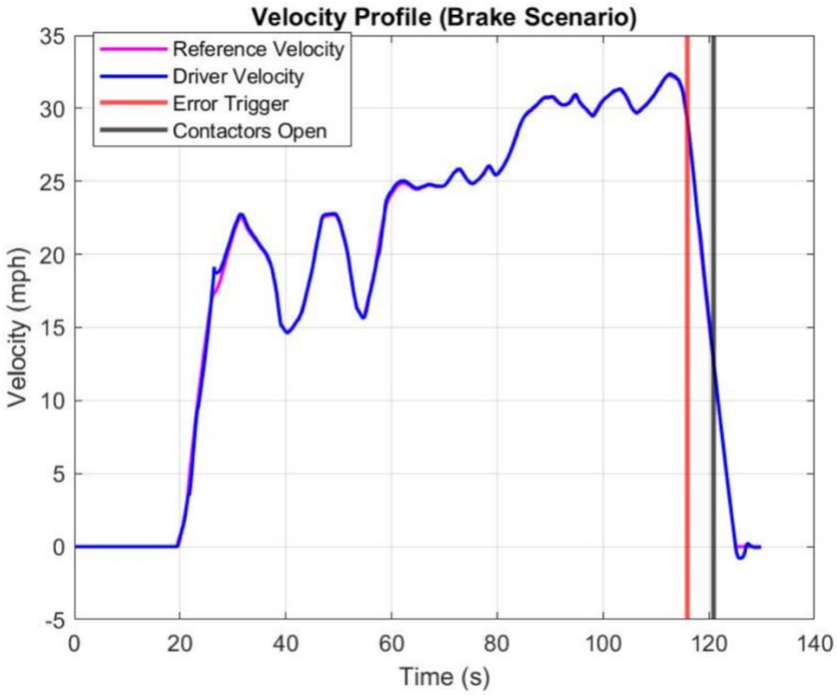

Figure 23 Drive Trace: ESS Fault Insertion During Braking

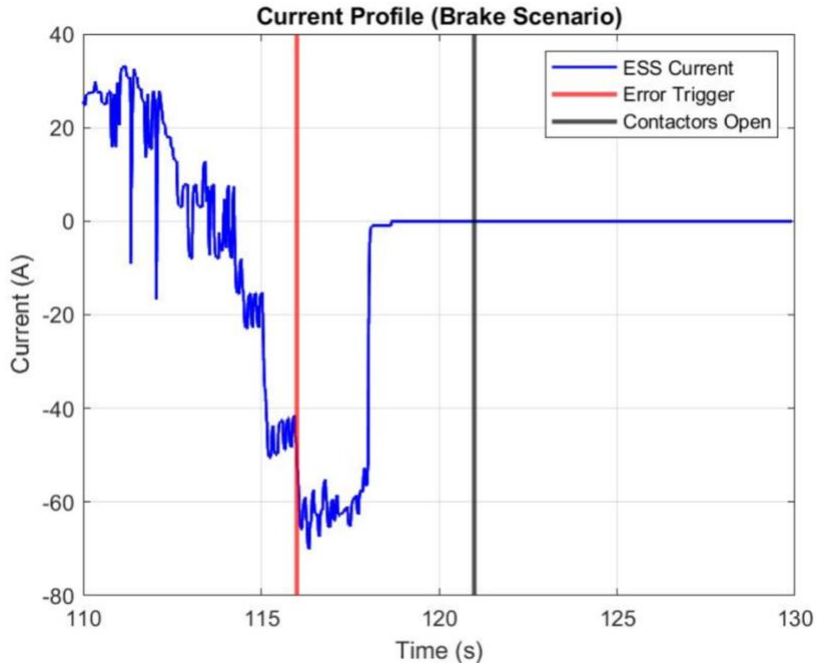

Figure 24 Current Trace: ESS Fault Insertion During Braking

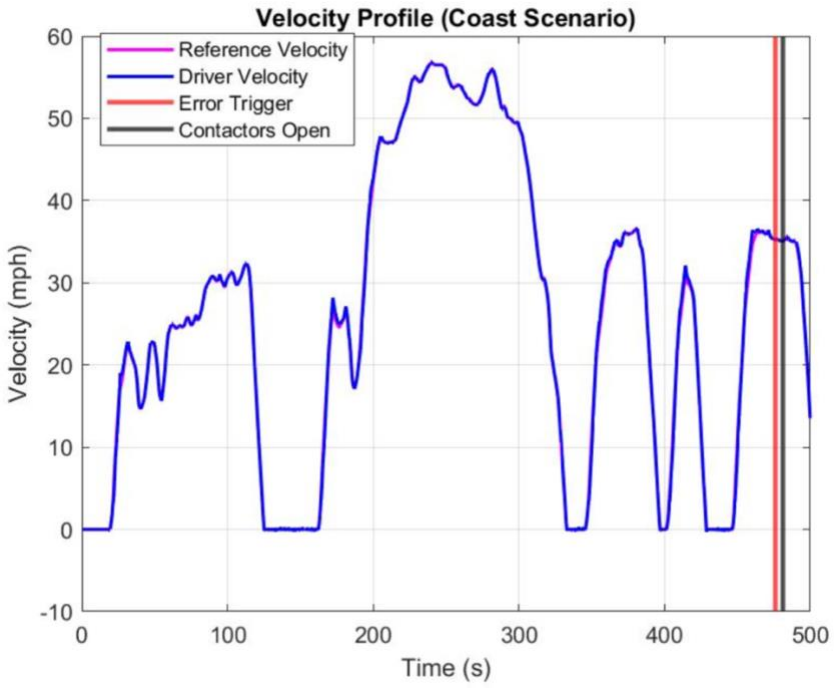

Figure 25 Drive Trace: ESS Fault Insertion During Coasting

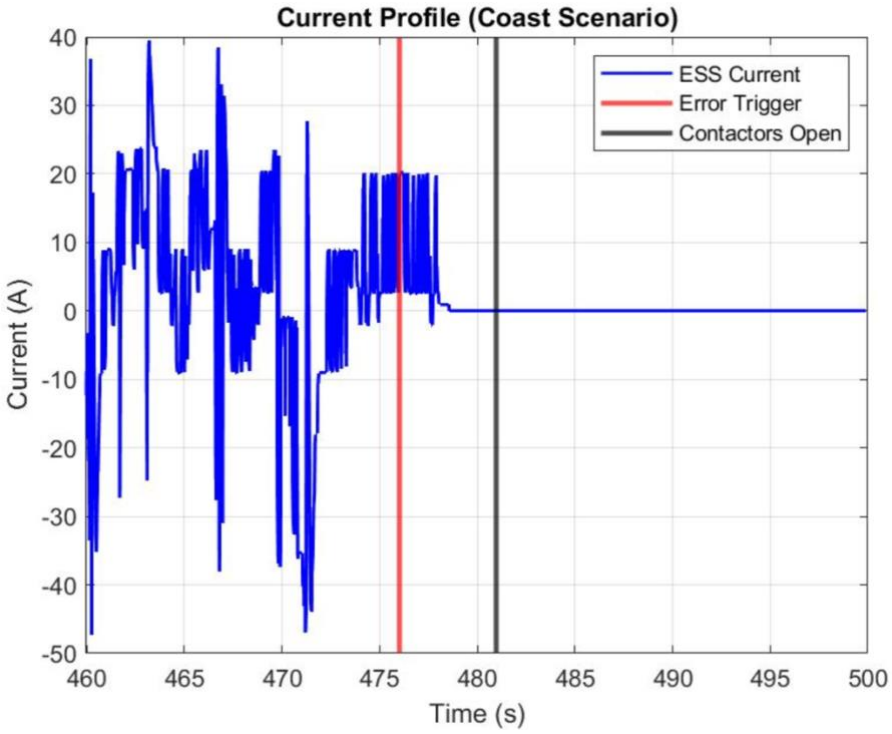

Figure 26 Current Trace: ESS Fault Insertion During Coasting

Summary of Test Case Results

Test Case No.	Test Case Name	Result
T101	Engine Torque Limit	PASS
T102	Engine Speed Limit	PASS
T201	REM Torque Limit	PASS
T202	REM Speed Limit	PASS
T203	REM Temp Limit	PASS
T301	ESS Voltage Limit	PASS
T302	ESS Temp Limit	PASS
T303	ESS Current Limit	PASS
T304	ESS SOC Limit	PASS
T305	ESS Error Accel	PASS
T306	ESS Errror Brake	PASS
T307	ESS Error Coast	PASS
T401	BAS Torque Limit	PASS
T402	BAS Speed Limit	PASS
T403	BAS Current Limit	PASS

Figure 27 Report Test Results

Finally, the same test cases were implemented in the MIL environment and results were compared with the HIL results. This comparison helps to verify that the control logic is maintained across both the environments. Figure 28 shows the comparison of the MIL and HIL results. The similarities of each method following the same profile is a clear indication that that the HIL fault implementation was a success.

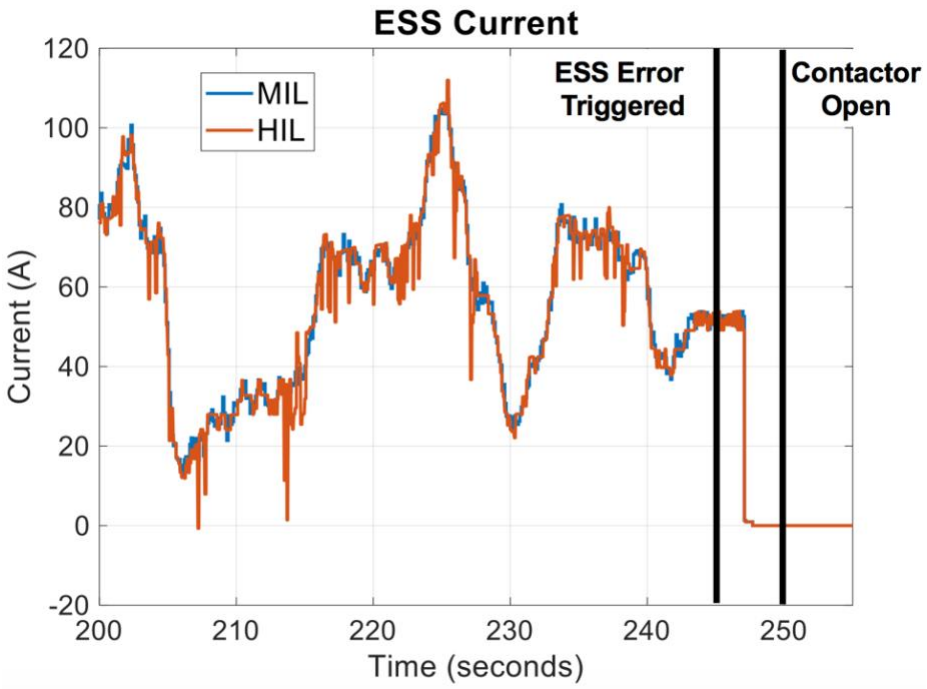

Figure 28 MIL vs HIL Simulation Comparison w/ an ESS Fault

Chapter 6. Conclusion

The focus of this thesis was to develop a systematic procedure for using fault diagnostic tools to be incorporated in HIL. Standard industry fault diagnostic tools were used to help construct the requirements that set the foundation for creating multiple test cases for fault insertion. The faults were used to develop and design a robust and safe controller and HIL simulation was the prime environment in which the faults was inserted. HIL simulation allowed for the physical controller to act as if in a vehicle while additionally removing the unsafe aspect of physically testing faults with component integration. An ESS fault was successfully inserted into the HIL simulation and proper documentation of the process was covered.

The main focus of this thesis was to demonstrate a professional way for using fault diagnostic tools to be incorporated into HIL software in order to design a vehicle controller. Standard industry fault diagnostic tools were used to help construct the requirements that set the foundation for creating multiple test cases for fault insertion. The faults were used to develop and design a robust and safe controller and HIL simulation was the prime In-the-Loop system to insert the fault. HIL simulation allowed for the physical controller to act as if inside a vehicle while additionally removing the unsafe aspect of physically testing faults with component integration. An ESS fault was successfully inserted into the HIL simulation and proper documentation of the process was covered.

The fault diagnostic procedure covered in this thesis was developed by working towards the 2019 Winter Workshop Deliverable for the Ohio State EcoCAR Team. This

procedure was presented as a technical video and report which were reviewed by industry experts from GM, MathWorks and dSPACE and other competition sponsors. OSU received an outstanding near perfect score in this deliverable. OSU plans to expand this tool to encompass other propulsion components and perform fault diagnostics to ensure operation safety and controller robustness. The HIL procedure is developed to further improve upon automotive industries across the globe and familiarize both student and engineers on how X-in-the-Loop systems, specifically HIL, can support the work for designing and testing the development of a controller. The HIL environment additionally supports any component with electrical communications and can additionally be used for hardware integration and for lifecycle/endurance testing.

6.1. Future Work

The research will continue to integrate additional test cases while designing the controller. The requirements will be continually updated across the design phase, and more requirements will be added potentially all components that will need to be tested. With the increasing number of test cases, keeping track of impact of controller algorithm updates on meeting test requirements will become cumbersome. Therefore, the use of dSPACE Synect to manage test cases is recommended in future.

Synect allows for multiple AutomationDesk tests to be compiled together. Automation Desk lacks the capability of tracking changes and different versions of the controller algorithm along with test case results. Fixing the AutomationDesk limitations, Synect can maintain an overall database to simultaneously track controller algorithm updates along with test case results. This will finally aid the fault diagnostics by

providing a better understanding of the effect of updated control strategy on controller robustness to faults.

Bibliography

[1] R. Isermann, Fault-Diagnostics Systems, Darmstadt, Germany: Verlag Berlin Heidelberg, 2006.

[2] K. R. Butts and A. Ohata, "Improving Model-based Design for Automotive Control Systems Development," Seoul, Korea, 2008.

[3] S. S. E. Jim Hays, *Writing Effective Requirements,* IBM Corporation, 2007.

[4] D. M. Stamatelatos, Fault Tree Handbook with Aerospace Applications, Washington, DC, 2002.

[5] Ford Motor Company, Failure Mode and Effects Analysis, Dearborn, MI: Ford Motor Company, 2011.

[6] C.-Y. Lin, P.-L. Li, C.-H. Li and W.-S. Chiang, "Investigation on IGBT Failure Effects of EV/HEV Inverter Using Fault Insertion HiL Testing," KINTEX, Korea, 2015.

[7] J. Luo, K. R. Pattipati, L. Qiao and S. Chigusa, "An Integrated Diagnostic Development Process for Automotive Engine Control Systems," 2007.

[8] D. Shang, E. Eyisi, Z. Zhang, X. Koutsoukos, J. Porter, G. Karsai and J. Sztipanovits, "A Case Study on the Model-Based Design and Integration of Automotive Cyber-Physical Systems," 21st Mediterranean Conference on Control & Automation, Platanias-Chania, Crete, Greece, 2013.

[9] G. Tibba, C. Malz, C. Stoermer, N. Nagarajan, L. Zhang and S. Chakraborty, "Testing Automotive Embedded Systems under X-in-the-Loop Setups," Austin, 2016.

[10] A. Mouzakitis, D. Copp, R. Parker and K. Burnham, "Hardware-in-the-loop system for testing automotive ECU diagnostic software," Coventry, UK, 2009.

Appendix A. List of Abbreviations

CAN	Computer Area Network
CIL	Component-in-the-Loop
ECU	Electronic Control Unit
EMC	EcoCAR Mobility Challenge
ESS	Electronic Storage System
FMEA	Failure Mode and Effect Analysis
FTA	Fault Tree Analysis
HIL	Hardware-in-the-Loop
MABx	MicroAutoBox
MIL	Model-in-the-Loop
RPN	Risk Priority Number
RTM	Requirement Traceability Matrix
SIL	Software-in-the-Loop
VIL	Vehicle-in-the-Loop
XIL	X-in-the-Loop

Appendix B. FTA: Key Block Diagrams

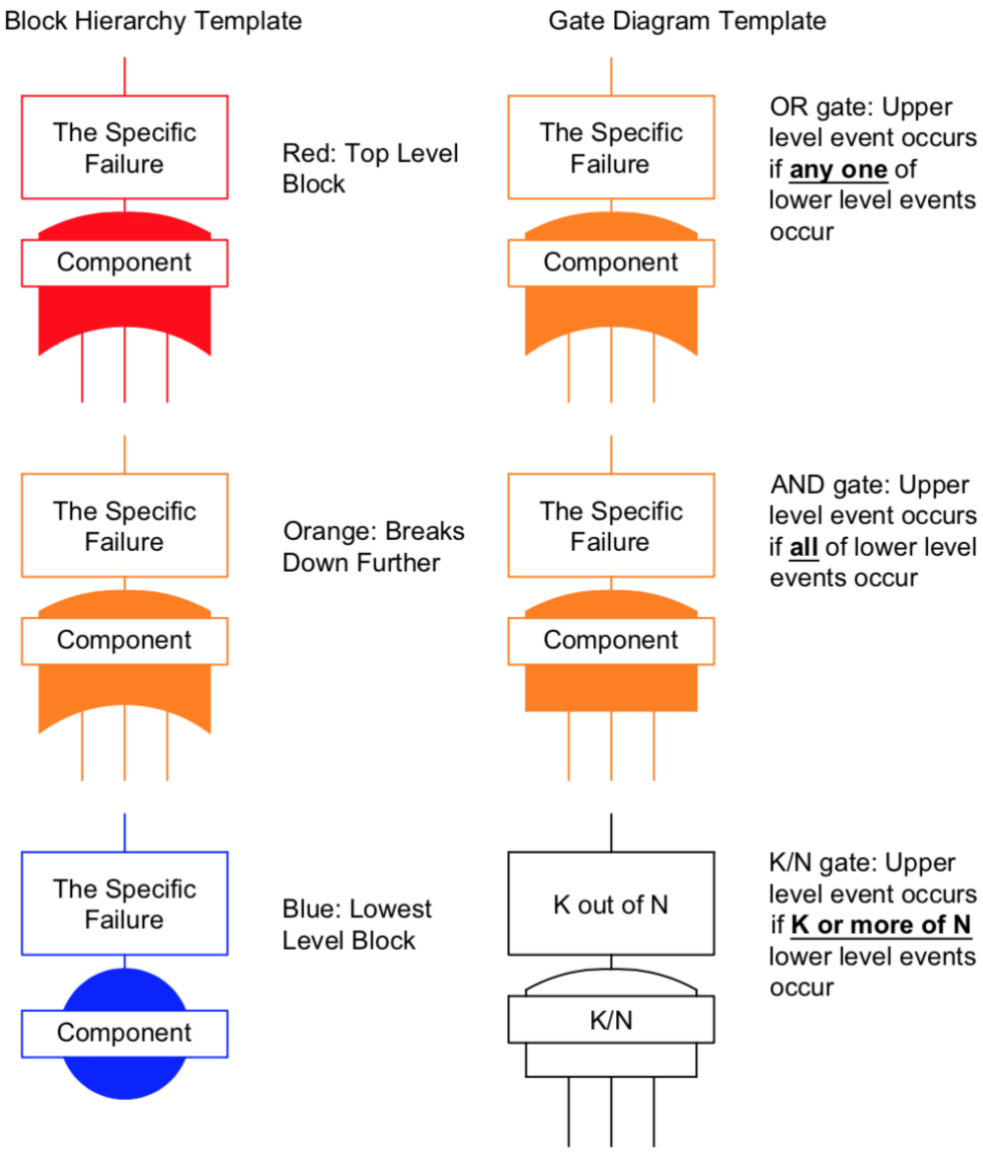

Figure 29 FTA: Block KEY

Appendix C. FMEA Rankings

Note: All Industrial rankings charts (Table 2-Table 4) were grabbed from Ford's FMEA 2011 Handbook [5]. Table 5-Table 7 were used for the rankings in FMEA example in Chapter 3, Section 3.2.2.

Table 2 Automotive Industrial Severity Rankings

Effect	***Criteria: Severity of Effect***	***Ranking***
Failure to Meet Safety and/or Regulatory Requirements	*Potential failure mode affects safe vehicle operation and/or involves noncompliance with government regulation* ***without warning****.*	*10*
	Potential failure mode affects safe vehicle operation and/or involves noncompliance with government regulation ***with warning****.*	*9*
Loss or Degration of Primary Function	*Loss of primary function (vehicle inoperable, does not affect safe vehicle operation).*	*8*
	Degradation of primary function (vehicle operable, but at reduced level of performance).	*7*
Loss or Degration of Secondary Function	*Loss of secondary function (vehicle operable, but comfort /convenience functions inoperable).*	*6*
	Degradation of secondary function (vehicle operable, but comfort /convenience functions at reduced level of performance).	*5*
Annoyance	*Appearance or Audible Noise, vehicle operable, item does not conform and noticed by most customers (> 75%).*	*4*
	Appearance or Audible Noise, vehicle operable, item does not conform and noticed by many customers (50%).	*3*
	Appearance or Audible Noise, vehicle operable, item does not conform and noticed by discriminating customers (< 25%).	*2*
No Effect	*No discernible effect.*	*1*

Table 3 Automotive Industrial Occurrence Rankings

Probability of Failure	***Criteria: Occurrence of Cause - DFMEA (Design life/reliability of item/vehicle)***	***Criteria: Occurrence of Cause - DFMEA (Incidents per items/vehicles)***	***Ranking***
Very High	*New technology/new design with no history.*	*≥ 100 per thousand* *≥ 1 in 10*	*10*
High	*Failure is inevitable with new design, new application, or change in duty cycle/operating conditions.*	*50 per thousand* *1 in 20*	*9*
	Failure is likely with new design, new application, or change in duty cycle/operating conditions.	*20 per thousand* *1 in 50*	*8*
	Failure is uncertain with new design, new application, or change in duty cycle/operating conditions.	*10 per thousand* *1 in 100*	*7*
Moderate	*Frequent failures associated with similar designs or in design simulation and testing.*	*2 per thousand* *1 in 500*	*6*
	Occasional failures associated with similar designs or in design simulation and testing.	*.5 per thousand* *1 in 2,000*	*5*
	Isolated failures associated with similar design or in design simulation and testing.	*.1 per thousand* *1 in 10,000*	*4*
Low	*Only isolated failures associated with almost identical design or in design simulation and testing.*	*.01 per thousand* *1 in 100,000*	*3*
	No observed failures associated with almost identical design or in design simulation and testing.	*≤.001 per thousand* *1 in 1,000,000*	*2*
Very Low	*Failure is eliminated through preventive control.*	*Failure is eliminated through preventive control.*	*1*

Table 4 Automotive Industrial Detection Rankings

Opportunity for Detection	***Criteria: Likelihood of Detection by Design Control***	***Ranking***	***Likelihood of Detection***
No detection opportunity	*No current design control; Cannot detect or is not analyzed*	*10*	*Almost Impossible*
Not likely to detect at any stage	*Design analysis/detection controls have a weak detection capability; Virtual Analysis (e.g., CAE, FEA) is not correlated to expected actual operating conditions.*	*9*	*Very Remote*
Post Design Freeze and prior to launch	*Product verification/validation after design freeze and prior to launch with pass/fail testing (Subsystem or system testing with acceptance criteria, such as ride and handling, shipping evaluation).*	*8*	*Remote*
	Product verification/validation after design freeze and prior to launch with test to failure testing (Subsystem or system testing until failure occurs, testing of system interactions, etc.).	*7*	*Very Low*
	Product verification/validation after design freeze and prior to launch with degradation testing (Subsystem or system testing after durability test, e.g., function check).	*6*	*Low*
Prior to Design Freeze	*Product validation (reliability testing, development or validation tests) prior to design freeze using pass/fail testing (e.g., acceptance criteria for performance, function checks).*	*5*	*Moderate*
	Product validation (reliability testing, development or validation tests) prior to design freeze using test to failure (e.g., until leaks, yields, cracks).	*4*	*Moderately High*
	Product validation (reliability testing, development or validation tests) prior to design freeze using degradation testing (e.g., data trends, before/after values).	*3*	*High*
Virtual Analysis - Correlated	*Design analysis/detection controls have a strong detection capability. Virtual analysis (e.g., CAE, FEA) is highly correlated with actual or expected operating conditions prior to design freeze.*	*2*	*Very High*
Detection not applicable; Failure Prevention	*Failure cause or failure mode can not occur because it is fully prevented through design solutions (e.g., proven design standard, best practice or common material).*	*1*	*Almost Certain*

The Following Rankings were used to construct the FMEA used for this Thesis in Chapter 3, Section 3.2.2.

Table 5 EMC Severity Rankings

Severity Rating Scale		
Rating	**Description**	**Definition (Severity of Effect)**
10	Dangerously High	Failure would result in total vehicle loss, and or severe injury/death to driver or spectator
9	Extremely High	Failure results in complete loss of vehicle functionality
8	Very High	Failure of the vehicle's subsystem
7	High	Failure is apparent to driver and would affect vehicle's performance
6	Moderate	Failure is apparent to driver and would minorly effect the vehicle's performance
5	Low	Failure is enough to constitude perfomance issue, but is able to be overcome with slight modification
4	Very Low	Failure would be annoying to the operation of the vehicle, but would not be sufficent enough to cause major performacne issues.
3	Minor	Failure is not apparent to driver, but would minorly affect vehicle's performance
2	Very Minor	Failure is not apparent to driver, but would minorly affect evironment or fuel consumption
1	None	Failure doesn't not consititute issue with driver or the team's goals

Table 6 EMC Occurrence Rankings

Occurence Rating Scale		
Rating	**Description**	**Definition (Severity of Effect)**
10	Very High: Failure almost inevitable	Failure is expected to occur everytime the vehicle is riden
9	High: Failures occur almost as often as not	Failure is **expected** to occur during light or medium testing
8	High: Repeated Failures	Failure is **expected** to occur during competition and/or high intensity testing
7	High: Failures occur often	Failure **could** occur while riding the vehicle during light or medium testing
6	Moderately High: Frequent failures	Failure **could** occur during competition and/or high intensity testing
5	Moderate: Occasional failures	Failure **could rarely** occur during high intesnsity test or a competition, but should never occur during light or medium testing
4	Moderately Low: Infrequent failures	Failure could occur once to three times throughout the life of the vehicle
3	Low: Relatively few failures	Failure could occur during initial testing and potentially once, but highly unlikely, during the life cycle of the vehicle.
2	Low: Failures are unlikely	Failure could only occur during initial testing and shouldn't occur again during the life of the vehicle
1	Very Low: Failure rarely occurs if ever	Failure is expected to never occur during the life of the vehicle.

Table 7 EMC Detection Rankings

Detection Rating Scale		
Rating	**Description**	**Definition (Severity of Effect)**
10	Absolute Uncertainty	The defect from the system is undetectable
9	Very Remote	Effect of failure may be noticed but the defect is difficult to locate
8	Remote	Effect of failure is noticeable but defect is difficult to locate
7	Very Low	The defect is not obvious and may be caught by electrical testing
6	Low	The defect is not obvious but will be detected by physical observations
5	Moderate	The defect might be obvious but electrical testing is necessary
4	Moderately High	The defect might be obvious and could be found from physical observations
3	High	The defect is obvious and will be detected by physical observations or electrical tests
2	Very High	The defect is obvious and will be found quickly from physical observations
1	Certain	The defect is obvious and will be detected immediately without any need for observations