Article

Automatic Failure Mode and Effect Analysis of Electronic Fuel Injection Model

Dong-Woo Lee [1,*], Dong-Min Lee [2] and Jong-Whoa Na [3]

[1] Aerospace and Aviation Electronics Research Center; dongwoo81@kau.kr
[2] Korea Aerospace University; leedongmin93@kau.kr
[3] Korea Aerospace University; jwna@kau.ac.kr
* Correspondence: Jong-whoa Na; Tel.: +82 2 300 0410

Abstract: In the development of the safety-critical systems, it is important to perform Failure Modes and Effects Analysis (FMEA) process to identify potential failures. However, traditional FMEA activities tend to be considered difficult and time-consuming tasks. To compensate for the difficulty of the FMEA task, various types of tools are used to increase the quality and the effectiveness of the FMEA reports. This paper explains an Automatic FMEA tool which integrates the Model-based Design (MBD), FMEA, and Simulated Fault Injection techniques in a single environment. The Automatic FMEA tool has the following advantages compared to the existing FMEA analysis tool. First, the Automatic FMEA tool automatically generates FMEA reports compared to the traditional spreadsheet-based FMEA tools. Second, the Automatic FMEA tool analyzes the causality between the failure modes and the failure effects by performing model-based fault injection simulation. In order to demonstrate the applicability of the Automatic FMEA, we used the electronic fuel injection system (EFI) Simulink model. The results of the Automatic FMEA were compared to that of the legacy FMEA.

Keywords: failure mode and effect analysis (FMEA), model-based design, automatic generation tool, fault injection simulation

1. Introduction

In safety-critical fields such as automotive, aerospace, and railway, it is important to comply with the international standards or guidelines for functional safety to prevent harm caused by potential system failures. Industries employ their own standards such as the SAE ARP 4754(4761A), RTCA DO178C, DO 254 for aerospace, and the ISO26262 for automotive [1–6]. The purpose of these guidelines is to identify hazards and decrease the occurrence of accidents to socially acceptable levels. HAZOP (HAZard and OPerability analysis), FMEA (Failure Modes and Effect Analysis) and FTA (Fault Tree Analysis) activities are key to achieving these goals [7–10]. Among them, the FMEA investigated the causality of fault-failure to improve the safety and reduce the likelihood of recall. However, traditional FMEA activities tend to be considered difficult and time-consuming tasks so that the effectiveness of the results falls short of expectations [11, 12].

To compensate for the difficulty of the FMEA task, various types of tools are used to increase the quality and the effectiveness of the FMEA reports. Traditional FMEA analysis tools use spreadsheet-based FMEA templates [13–16]. The spreadsheet based FMEA tool provides users with the FMEA templates according to the regulations or specifications agreed upon by the stakeholders in their field. Then the users identify each potential failure mode and assess their local and global failure impacts to safety, including the severity or the occurrence. Meanwhile, the complexity of software in electric equipment has increased rapidly, making it very difficult to analyze system fault-failure causalities and their impacts [17]. To address these challenges, leading organizations are pursuing model-based SW development, including UML, SysML, and Simulink [18–24].

Model-based FMEA techniques use design models to support a visible and automated analysis environment. In Model-based Design (MBD), the design model describes the function of the system and its behavior utilizing the graphical objects of various types [UML, SysML]. The MBD and FMEA process can be integrated to simulate the failure effects at the various level from the local level up to the vehicle or system level [21]. This approach can evaluate the causality of the fault-failure for a limited number of cases [25, 26]. In addition, FMEA users expect support for the risk priority matrix (RPN) to respond the unacceptable failure effects.

This paper explains an Automatic FMEA tool which integrates the MBD, FMEA, and Simulated Fault Injection techniques in a single environment. The Automatic FMEA investigates the fault-failure causality of the system using simulated fault injection to find the potential failure modes and their effects at various level from local to global. In addition, it incorporated an AI based failure effect classification module to identify the severity and occurrence rate so that the risk priority matrix can be automatically presented. In addition, the Automatic FMEA is based on MATLAB/SIMULINK to support large users in the safety-critical fields around the world.

The Automatic FMEA tool has the following advantages compared to the existing FMEA analysis tool. First, the Automatic FMEA tool automatically generates FMEA reports compared to the traditional spreadsheet-based FMEA tools [13–16]. These tools provide FMEA templates and traceability, and users write FMEA directly from documents such as design documents and safety guidelines. The Automatic FMEA tool automatically generates FMEA by analyzing the model developed at the system level and analyzing the failure effects due to failure mode. Second, the Automatic FMEA tool analyzes the causality between the failure modes and the failure effects by performing model-based fault injection simulation. The existing model based FMEA analysis tool [21] automatically analyzes the Simulink model and creates FMEA tables for a limited number of cases. Here, each failure data (failure-rate, failure effects, etc.) must be entered directly, and RPN allocation process is manually performed for FMEA analysis. The Automatic FMEA tool performs fault injection simulation so that we can automatically identify the fault-failure causality and allocate the associated RPNs using the fault injection campaign plan.

The rest of this paper is organized as follows. Chapter 2 explains the trends of FMEA analysis support tools and model-based FMEA analysis technology. Chapter 3 describes the Automatic-FMEA environment, support technologies, and analysis procedures. Chapter 4 produced a MILS test model and developed an FMEA report to check the performance of the Automatic FMEA. Finally, Chapter 5 presents the significance of Automatic FMEA research and future research plans.

2. Failure Mode-Failure Effects Trends

Safety certification guidelines (ARP4761, ISO26262, etc.) require safety analysis documents and data of FMEA and FTA that can be reviewed by the certification authorities or their representatives. The safety analysis process is a time and resource consuming task that requires safety engineering technology as well as domain knowledge. FMEA is a process of reviewing thousands of pages of development documents and identifying the potential failure sources and their effects in a detailed format. In addition, the developer must iterate this safety analysis process when there is a modification due to the requirement change or design error.

The legacy FMEA analysis tools generally use the templates required by the corresponding safety specifications [13, 15, 16, 27, and 28]. These tools record the failure modes and their effects manually as well as calculate the criticality, which is the product of the occurrence probability and the severity [13–16, 27, 28]. Unfortunately, as the size of the modern cyber-physical systems or embedded targets are growing rapidly, it is becoming more difficult to create the FMEA reports. The representative template-based FMEA analysis tools include IQ-FMEA, ENCO, and Relex [7].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 May 2022 doi:10.20944/preprints202205.0244.v1

As model-based development technology became popular in the SW development process, the research toward the model-based FMEA took place [39]. Using the design model developed at the MBD process, the Model-based FMEA apply the failure mode to the design model and perform the model simulation to produce the possible failure. Like the traditional FMEA, the model simulation can generate the failure effects at three levels: the local effect, next effect, and end effect. The Hip-Hop FMEA tool automatically analyzes the Simulink model, identifies FTA synthesis, fault modes, and creates FMEA tables [21]. In the process of creating an FTA, the analysis model is optimized with a minimal-cut set. However, each failure data-set (fault rate, failure effects, etc.) must be directly entered, and there are limitations in analyzing severity, incidence, and detection, which are essential for FMEA analysis. MADe FMEA is a commercial FMEA analysis tool that performs model-based failure mode derivation and failure impact analysis by defect injection simulation [25]. In addition, it is possible to establish an integrated safety/reliability analysis environment by the MADe tool. However, it uses its own modeling language so that the general users have to undergo a learning period.

Automatic FMEA is a systematic failure analysis environment using (1) the statistical fault injection simulation and the rule-based failure classification. The Automatic FMEA is built on top of the statistical fault injection environment. Once it sets the statistical confidence level, it automatically calculates the number of faults to be injected and builds the fault list using the HAZOP guide words. After the injection campaign, every fault-failure data is stored and processed to calculate the criticality for each injection event using the severity and occurrence rule set defined by the users.

Currently, the automatic FMEA is built upon the SIMULINK environment because of its large user base around the world. However, it's environment is not limited to SIMULINK but expandable to other simulators such as MODELIA [40]. Table 1 shows the comparison of the existing FMEA analysis support tool and Automatic FMEA.

Table 1. Comparison of FMEA tools

Type	Automatic-FMEA	Hip-HOP	MADE	Relex
Environment	Simulink	Simulink	MADe model	N/A
Automatic Range	Automatic	Semi-automatic	Automatic	Manual
Failure effects analysis	Fault Injection Simulation	User Manual	Fault Injection Simulation	
Severity		FTA	User Manual	Manual
Occurrence	Rule Based	FTA	MADe-FTA	
Detection		N/A	N/A	
Fault Injection Type	Saboteur	N/A	Mutation	N/A

3. Automatic FMEA System

3.1. Automatic FMEA

Automatic FMEA is a safety analysis tool that performs fault simulation in a Simulink environment to evaluate the failure effects of the design target system. Automatic FMEA restructures the Simulink model to perform fault simulation. The Simulink model consists of hierarchically connected functional blocks. In Simulink, one can express the algorithm using the block which is an MDL file in the Simulink internal system. Automatic FMEA parses the Simulink MDL file to restructure into a suitable model for the fault injection process. In the restructured model, we insert a saboteur block between the original system blocks so that it can perform the fault injection simulation. We explain the internal process of the Automatic FMEA shown in Figure 1. First, we use the safety requirements and design documents of the target system to identify the range of tolerance of the design parameters of the system [2, 4]. Using the results, we can identify the severity of the failure effects for the given failure modes which is accomplished by injecting the corresponding faults into the target system.

Second, we creates a fault model corresponding to the failure modes using the design parameters of the target models and performs fault simulation to evaluate the failure effects. In the model-based design process using the Simulink, the target model has to be verified using the Simulink Design Verifier. After this process, the fault models are built using the selected HAZOP guidewords corresponding to the design parameter of the target model. During fault injection simulation, we inject these fault models representing a specific failure mode and record the results of the fault simulation, which are used to identify the failure effects.

Third, using the results of the fault simulation, we build the failure information that can be used for the FMEA report. Using the results of the fault simulation, it is possible to identify the local effect, next effect, and the end effect of the given failure modes. Also, using the failure decision rules, we can identify the failure effects, which are the severity and occurrence rate of the failure modes. Using these are the information, we may build the preliminary FMEA report that can not only accelerate the FMEA process but also used to find the unidentified failure modes by the FMEA committee members that may happen in the case of the new system development.

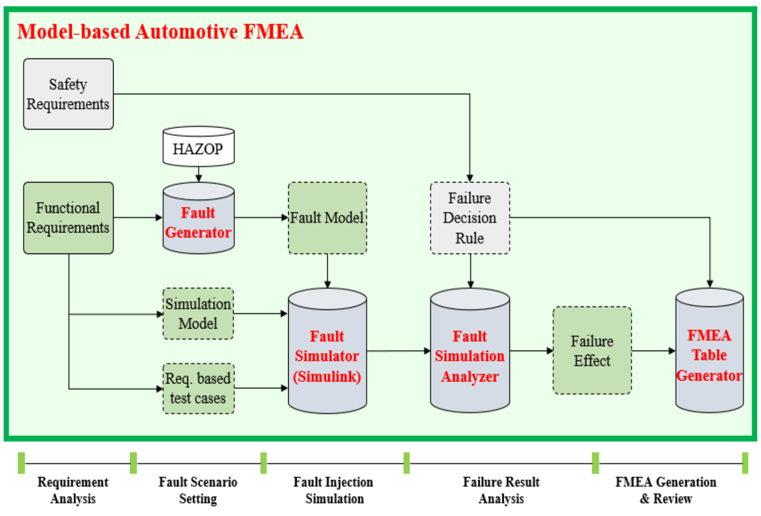


Figure 1. Model-based Automatic-FMEA Architecture Overview

In order to develop automated FMEA analysis tools in the Simulink environment, it is necessary to establish solutions to three development problems. The first is "How to track the failure impact of FMEA's failure mode?" One of the purposes of FMEA analysis is to track the potential impact of the system due to failure. In the traditional FMEA process, the knowledge and experience of the member of the FMEA committee is important for the identification of the failure. In model-based development, the model itself can be used as a tool to track the impact of failure. Automatic FMEA used a Simulink Model Tree generation algorithm to trace the impact of failure in the model. This will be described in detail in 3.2.1.

The second is "How to apply the failure mode to the simulation model?" Automatic FMEA uses the fault simulation corresponding to the failure mode to derive information necessary for the completion of the FMEA report. The fault simulation was implemented using the saboteur technique. This is explained in detail in Section 3.2.2. Finally, the third question is "How to set the severity, detection, and occurrence of failure effects?" Experts allocate the severity, detection, and incidence of FMEA to values between 0 and 10, based on experience. However, Automatic-FMEA can quantify and allocate the severity and detection of failures using the result values of failure simulation. To this end, Automatic-FMEA proposes a method of analyzing simulation result values and determining severity detection using rules and rule engines. The degree of occurrence can be considered to be derived as a statistical defect injection test. Rule-based failure analysis is described in detail in Section 3.2.3.

3.2. The Function of Automatic FMEA

3.2.1. Simulink MDL Parser

The Simulink MDL parser identifies each block and its input/output ports in the Simulink design model and derives connectivity and hierarchy information among them. The Simulink MDL file specifies the system structure as a pair of keys and values [29]. When the Simulink specify a functional unit block of Simulink, the model is configured with a key representing the type, name, and location of the block. Simulink uses three keys such as the system syntax, block syntax, and the line syntax which is used to specify the connectivity and hierarchy of the design model. The system syntax consists of one or more blocks or subsystems in the Simulink model so that we can infer the hierarchical relationship between blocks. The block syntax and line syntax can interpret the connection relationship between each block in the Simulink model. Therefore, parsing the MDL file, we can identify the hierarchy and connection information of the blocks in the Simulink design model. Using this information, we can find the variables and its value to describe the local effect, next higher effect, and the end effect in the FMEA table.

The Simulink MDL parser receives the model of the MDL format as input, interprets the syntax, and outputs a Simulink model tree that can derive FMEA failure traceability analysis. The MDL parser consists of three stages: the Keys & Values Classification stage, Context Interpretation stage, and the Model Re-configuration stage as shown in Figure 2. In the first stage, the Keys & Values Classification process identifies a key in the MDL file and stores a valid key and its value. In Simulink, the MDL model is described using keys such as 'system', 'block', and 'line' and their values. The Simulink MDL parser selects and stores only the information necessary for generating the Simulink Model Tree from the input file.

In the second stage, the Context Interpreter checks the block configuration and connection of the model, and the parent-child relationship structure of the block. The Simulink model uses the 'system' key to define a parent block including the best block or one or more functional blocks. Alternatively, signal connection between blocks is established using the "line" key and each block port information. The Context Interpreter checks the previously processed Key & Value Set information to create a linked block set for Simulink model tree configuration. In the third stage, the Model Reconfiguration process creates a Simulink model tree that can be used to trace the fault propagation path using the linked block sets connected according to the system structure. The Simulink model tree created with the Simulink MDL parser is plotted in the Figure 2. As shown in the figure, we can confirm the hierarchy, configuration, and the input output between blocks of the Simulink design model.

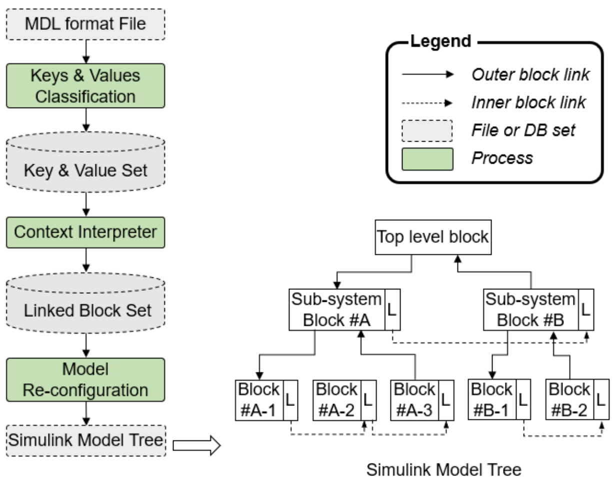


Figure 2. Automatic-FMEA MDL Parser Process & Model

Using the Simulink model tree, we can describe the propagation of the failure effects of the given failure mode, which is the local effect, next higher effect, and the end effect in FMEA table. Figure 3 describes a method of deriving a failure effect from a Simulink model tree. The left side of the figure explains the Inter-block Failure Propagation within the subsystem block, and the right side shows an example of the Intra-block Failure Propagation between the blocks up to the parent block. We can set the initial faulty block to local effect, and identify the next higher effect and the end effect block using the inter-block link information. If the model has the hierarchy, we can append the intra-block link information to the previous failure propagation list so that the expanded failure propagation list can be obtained.

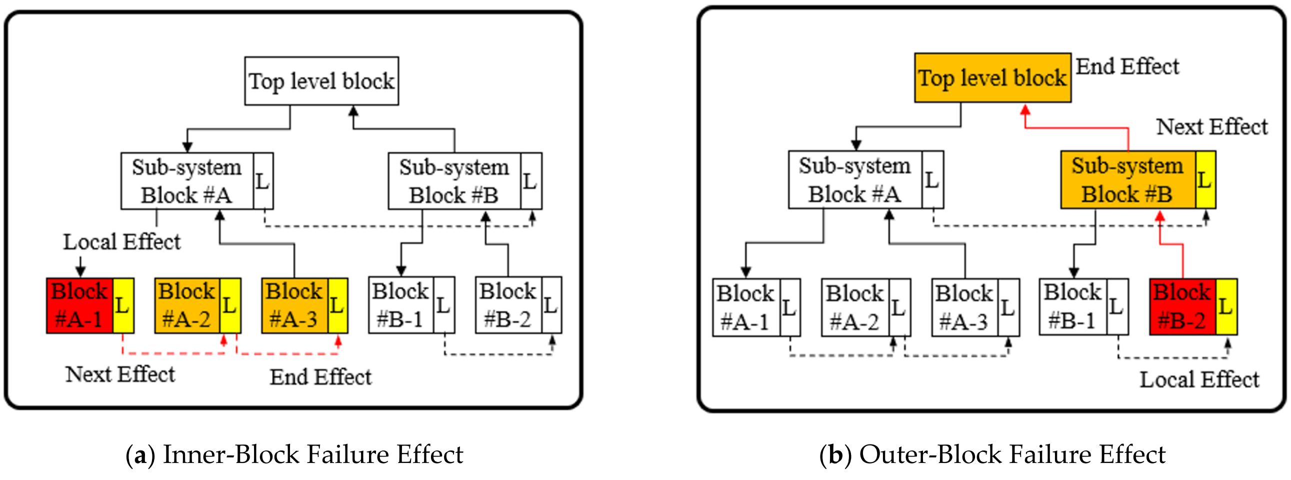


Figure 3. Building the failure propagation list using the inter-block and the intra-block link information of the Simulink Model Tree extracted from the Simulink design model

3.2.2. Saboteur-based fault simulation model creation

Automatic FMEA builds a simulation model for the fault injection campaign to analyze the failure effects for given failure modes by applying the saboteur module to the Simulink MDL file. In the saboteur technique, we may append a saboteur block between the design blocks so that it can insert faults into the design model. The saboteur block converts the value of a variable into a faulty value according to a fault model specified by the fault injection campaign. The saboteur technique has the advantage of being able to create a failure test model in a simple manner. However, in the process of adding saboteur blocks, it may alter the original functionality of the design model. In order to maintain the consistency between the original model and the fault simulation model, we perform a verification simulation using Simulink Design Verifier and the benchmark experiments which compares the result of the simulation without and with the saboteur block in the design model.

The saboteur block of Automatic FMEA injects a fault model from the set of fault list from the fault injection campaign. The fault model illustrates the failure modes using (1) a fault model, (2) a fault occurrence time, and (3) a fault duration time. The fault model is constructed by applying the HAZOP guideword to each and every variable available in the Simulink design model. In Automatic FMEA, we multiplied the value of the design parameter of the Simulink model with the proportional coefficient using the failure keyword as shown in Table 2. For example, suppose the parameter 'A' applied with 'less than requested' failure keyword. Then it builds the three-fault model using the 90%, 80%, and 70% of the value of the parameter 'A' after the consultation with the domain experts. In this way, various types of failure effects may be investigated in a quantitative manner using the derived fault models so that the Automatic FMEA may contribute to discover the unseen failure mode by the members of the FMEA committee.

Table 2. HAZOP based Failure Mode

Failure keyword		Meaning
Loss of Function (LF)		Functions not activated
Incorrect	More Than Requested (MTR)	Functions activated in excess of requested quantity
	Less Than Requested (LTR)	Functions activated in less than requested quantity
	Wrong Direction (WD)	Reverse functions activated
Unintended Activation of Function (UAF)		Functions activated at unintended time
Locked/Stuck Function (SF)		Function is not updated
Timing	Early	Function activated earlier
	Late	Functions activated later

Automatic FMEA integrates the saboteur blocks with the blocks in the simulation models to facilitate the simulated fault injection campaign. Let a block in Simulink is connected with a port key and a line key where the port key is an input or output identifier of a block, and the line key is a link of two blocks. Automatic FMEA searches for the port key and line key of the block corresponding to the fault injection location to add the saboteur block and modifies its value. The original design model and the model prepared for the fault injection experiments are compared in Figure 4. The required number of the fault simulation design model are created and the injection process proceeds automatically so that we may accumulate a large number of the failure mode-failure effects relationship information for the Simulink design model. In the figure, a saboteur block is added between each block for implanting a defect. Automatic FMEA performs the Simulink fault injection simulation of the preparation, execution, and the collection of the simulation log for the fault models described in the fault injection campaign.

Automatic FMEA controls the saboteur block added to the Simulink model to perform a failure simulation as shown in Figure 5. In a normal case, the output from the front logic block is transferred to the back-end logic block via the correct data (CD) block. In the case of the fault injection simulation, the one of the outputs of the failure data (FD) block is transferred to the back-end logic block according to the fault injection scenario.

The Automatic FMEA controls the saboteur block using the MATLAB Command script [30]. The MATLAB command script may probe or update the status of the block in the Simulink model. In addition, the break-point function may be used to pause and resume the simulation at a predetermined time or event. According to the failure scenario, Automatic FMEA simulates failure in following steps: (1) pause the current simulation run when the simulation time reached the fault time, (2) set the failure data (FD), (3) resume the simulation, (4) pause the simulation after the fault duration time, (5) restore the original value, (6) resume the simulation run. Automatic FMEA sets or releases the fault values using the Failure Mode Selector and Failure Data values of the saboteur block. The Failure Mode Selector of the saboteur Block selects one of the six HAZOP fault types, and the Failure Data sets the value of the fault model. For example, if the failure mode is the sensor degrades and its signal is reduced to 70% from the nominal value, the Failure Mode Selector enables the 'Less Than Requested' and set the Failure Data to 70%.

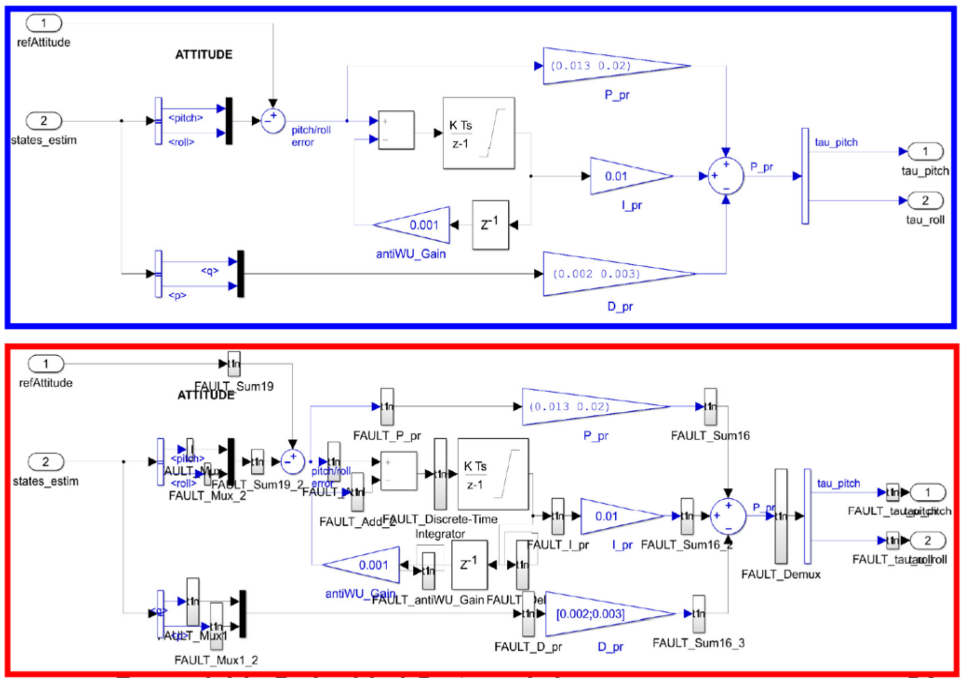


Figure 4. Normal Model (Up) & Mutation Model (down)

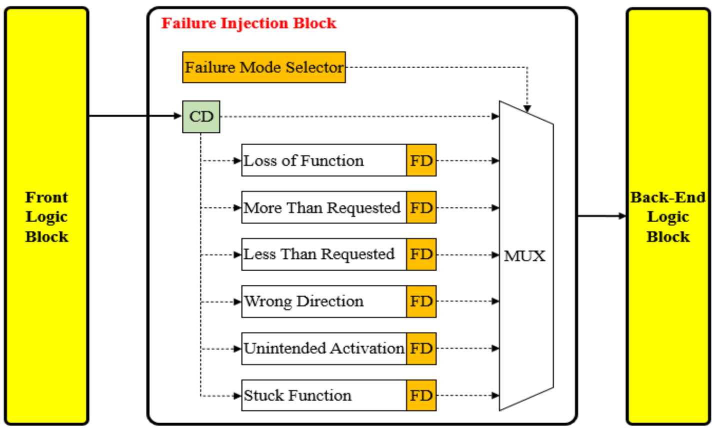


Figure 5. Fault Injection Block Structure (CD: correct data, FD: fault data)

3.2.3. Rule-based failure determination function

Automatic FMEA uses rule-based engines to analyze the result of the fault injection simulation and determine the severity of the failure effects for the given failure mode. Automatic FMEA determines the effect of a failure mode by comparing the results of a golden run simulation to that of the fault simulation. In addition, it also examines the value, state, and the condition of the parameter of the fault simulation.

The rules in the Automatic FMEA are designed to support the fault models and the Fault Injection Block of Figure 5. Each rule evaluates the failure mode and the value of the parameter. First, the condition part of the rules is designed to identify the failure mode of the current simulation such as (a) the under-estimation values check for system degradation, (b) over-estimation values check for system overload, (c) error margin check for the stability of a performance parameter, or (d) failure event check for the detection of the specific failure effect.

Next, the action part of the rules is used to determine the value of the entries of the severity, occurrence, and the controllability in the FMEA table. For the given failure mode, the Automatic FMEA performs a large number of fault injection simulations to find the behavior of the failure effect. Thus, we can compare the distribution of the parameter from the current simulation runs to that of the golden run simulation so that the action part of the rule can assign the resulting severity between 0 and 10. The occurrence and detection entries of FMEA table is not yet implemented so that it is necessary to have a meeting with the domain experts and the designers of the model. This process of identification of the severity, occurrence, and the detectability of the failure effects is summarized in Figure 6.

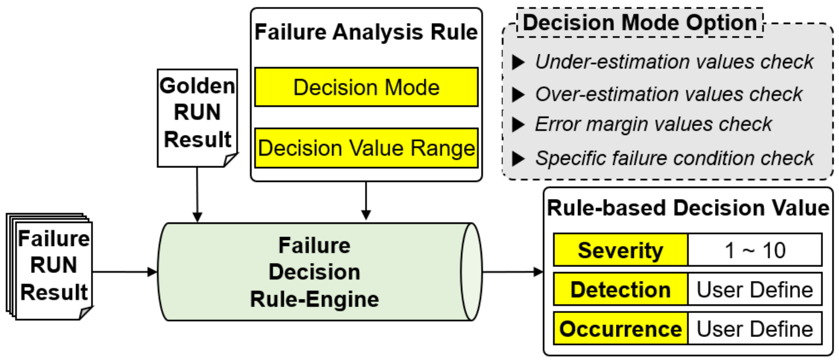
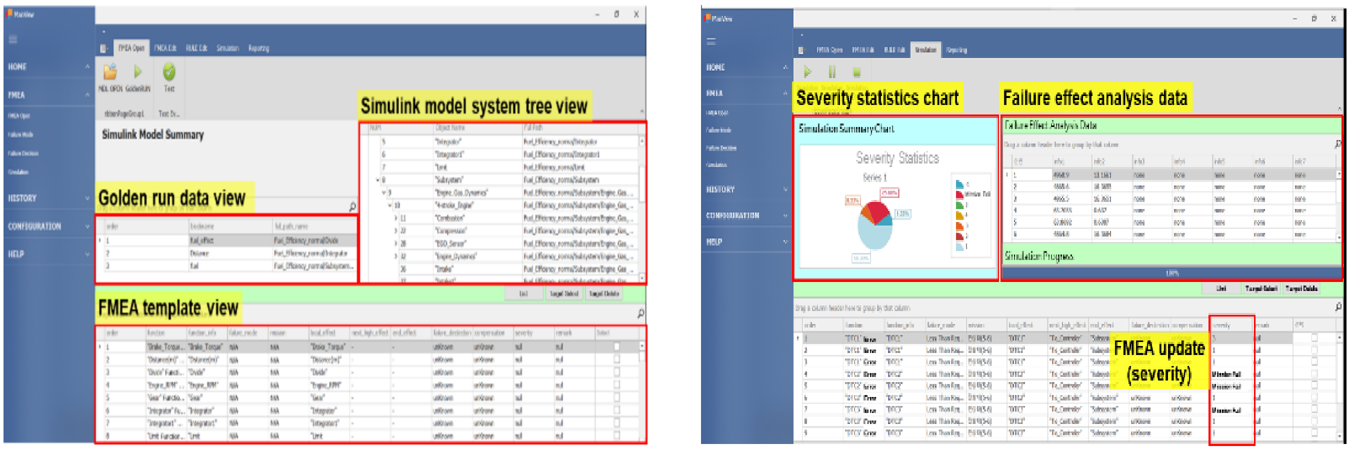

Failure Analysis Rule
Decision Mode
Decision Value Range
Decision Mode Option
Under-estimation values check
Over-estimation values check
Error margin values check
Specific failure condition check
Golden RUN Result
Failure RUN Result
Failure Decision Rule-Engine
Rule-based Decision Value
Severity
1 ~ 10
Detection
User Define
Occurrence
User Define

Figure 6. Automatic FMEA Failure Decision Rule-Engine Architecture

3.2.4. Implementation of Automatic FMEA

The operation of Automatic FMEA is in four steps as follows: 1) Simulink model analysis, 2) failure mode setting, 3) failure decision rule setting, and 4) fault injection simulation. The first Simulink model analysis step analyzes the input Simulink design model, prepares an initial FMEA template, and performs a golden-run simulation. In the second failure mode setting step, one failure mode from the failure mode list is selected and injected into the design system. The third failure effect determination rule setting step analyzes the fault simulation result and defines a rule for determining the severity. Fourth, in the failure injection simulation step, the fault injection simulation is automatically performed according to the previously set failure (operation) mode, and the result is analyzed according to the failure impact analysis rule. The screenshot of the Automatic FMEA user interface and the final results are shown in Figure 7.



(**a**) User Interface (**b**) Fault Injection Simulation Results

Figure 7. Automatic FMEA Software screen capture

4. Case study: Electronic Fuel Injection System

4.1. Electronic Fuel Injection System

4.1.1. Overview of Electronic Fuel Injection System

The Electronic Fuel Injection (EFI) System is a system that calculates the amount of fuel injected into the engine. The EFI system may be described using the Sensor, Engine Control Unit (ECU) and the Engine as shown in Fig. 11. The Sensor sub-system measures the engine state and transmits the measurement result to the ECU sub-system. The ECU sub-system calculates the amount of air and fuel using the Sensor information. The Engine sub-system generates the power of a vehicle through four-stroke cycles using fuel and air.

The Sensor sub-system consists of four sensors. (1) The Throttle Position Sensor (TPS) measures a throttle valve angle. (2) The Crankshaft Position Sensor (CPS) measures the rotational speed per second of the crankshaft of the engine unit. (3) The Exhaust Gas Oxygen sensor (EGO) measures the oxygen concentration in the exhaust gas. (4) The Manifold Absolute Pressure sensor (MAP) measures the internal pressure of the intake manifold. The measured sensor data are connected to the data bus and transmitted to the ECU sub-system.

The Engine Control Unit (ECU) sub-system calculates the amount of fuel by performing three processes from the input sensor sub-system data. First, the Sensor Data Filter process checks whether there is any abnormality in data received from the data bus. Second, the Control Logic process sets the air ratio using the verified sensor data and current vehicle state information. Third, the Fuel Rate Calculator process calculates the amount of air intake using vehicle condition information and throttle position data and calculates the amount of fuel based on the set air ratio.

The Engine sub-system consists of eight devices. (1) The fuel injector is a device that injects fuel into the intake manifold according to the fuel measured by the ECU. (2) The throttle body is a device that controls the amount of air using the throttle valve. (3) The intake manifold is a pipe that supplies the cylinder with a gas mixture that consists of air and fuel. (4) The intake valve sends the gas mixture from the intake manifold to the cylinder. (5) The cylinder is a space in which four strokes (intake, compression, explosion, and exhaust) of the engine are performed. (6) A crank converts energy obtained from the four strokes of an engine into kinetic energy. (7) The exhaust valve then exhausts gas burned in the cylinder to the exhaust manifold. (8) The exhaust manifold is a pipe that sends the discharged exhaust gas to the exhaust pipe.

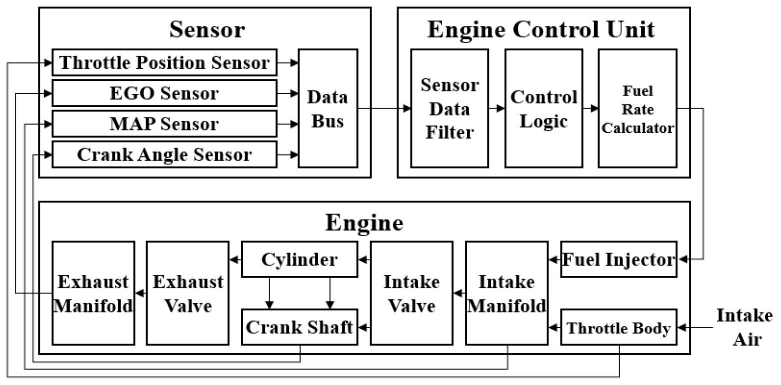


Figure 8. Electronic Fuel Injection System Architecture

4.1.2. Simulink Design Model of Electronic Fuel Injection System

The Electronic Fuel Injection (EFI) Simulink model from the Fault-Tolerant Fuel Control System Model was used to as a case study for the evaluation of the Automatic FMEA [31]. The EFI Simulink model consists of the Sensor, Fuel Rate Controller, and Engine Gas Dynamics sub-systems to calculate the air-fuel ratio as shown in Figure 9. The Sensor sub-system receives data from the four sensors (TPS, EGO, MAP, and CPS) and outputs them to the data bus. The EFI sensor data are throttle valve angle, number of rotation per crank (rad/s), exhaust oxygen concentration (volts), and absolute pressure (bar) in the intake manifold. The EFI block is interfaced with the Electronic Fuel Injector System and the Vehicle model to build the operation environment. Test Scenario and the Data Logging block are interfaced with the model to provide the testing purpose.

The Fuel Rate Controller sub-system receives a sensor data-bus and calculates the fuel flow rates per second. The Fuel Rate Controller consists of three blocks: control logic, airflow calc, and fuel calc. The control logic block verifies sensor data-bus and decides on the driving mode. The airflow calc block predicts air flow rate based on sensor data and driving data. The fuel calc block decides on the fuel flow rate using the driving mode and air flow rate.

The Engine Gas Dynamics sub-system outputs engine information from air flow rate and fuel flow rate. The Engine Gas Dynamics sub-system consists of the throttle & manifold block and the 4-stroke engine block. The throttle & manifold block calculates the air volume and absolute pressure information in the intake manifold. The 4-stroke engine block represents the engine dynamics in engine cylinder. This block calculates engine power by combustion dynamics of air and fuel mixtures, and outputs crank rotational speed and engine RPM. It also outputs the saturated oxygen concentration of the exhaust gas after combustion. This information becomes an input to the Sensor subsystem.

4.2. Electronic Fuel Injection analysis using Automatic FMEA

As a case study, we applied the proposed Automatic FMEA to the EFI Simulink model to find the FMEA report for the Sensor part of the EFI. The internal operation of the Automatic FMEA is explained in four steps: (1) Simulink model analysis step, (2) definition of failure mode step, (3) Failure effect identification step, and (4) fault injection simulation step.

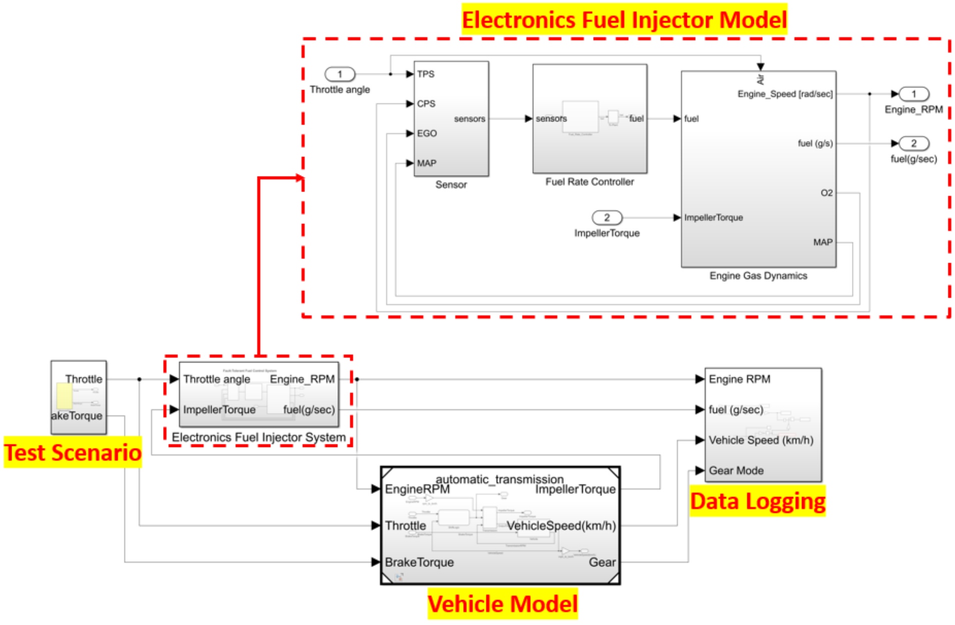


Figure 9. Simulink Model of the Electronic Fuel Injection System

4.2.1. Step 1: Simulink model analysis

When the Simulink design model is ready and input to the Automatic FMEA, it identifies the sub-system and blocks in the design model to build an internal data structure of tree type. Then the nodes of the tree represent the variables of the blocks in the model and the edges represent the relationship between the input-output relationships between variables of the block in the model. If the user wants to inject a fault model into the variable of a block as a fault injection campaign, the selected edges of the model will be modified with the saboteur block to enable the insertion of the specified fault model.

The EFI Simulink model consists of three sub-systems and 163 functional blocks. Since we are interested in the failure effect of the Sensor model, we selected the Sensor blocks of the four sensors (TPS, EGO, MAP, and CPS). The FMEA template of the MIL-STD-1629 standard was also selected. We identified the fault-failure traceability of the target system using the tree data structure of the Simulink design model. The entries of the FMEA template are filled after the execution of the fault injection simulation explained in the following steps. In step 2, for each of the fault models of the fault injection campaign, we enter the entries of the FMEA table such as the function and failure modes. In step 3, the entries of local effect, next high effect, and the end effect information are identified using the results of the fault injection simulation. In step 4, the severity entry is filled after the execution of the failure diagnosing rule-based systems explained in the previous chapter. In the following, the process of completing the entries of the FMEA table are explained in detail.

4.2.2. Step 2: Definition of Failure Mode

In this step, we determine the failure mode of the sensor unit of the EFI model. The four sensors of the EFI may cause open circuit (or short circuit) faults in sensor connector pins due to factors such as wearing out, vibration etc. in the vehicle driving environment [32, 33]. These faults cause sensor failure modes such as the Drift, Hard-over, and the Stuck [34–37]. Automatic FMEA analyzes the characteristics of these three sensor failure mode and defines the failure mode as follows.

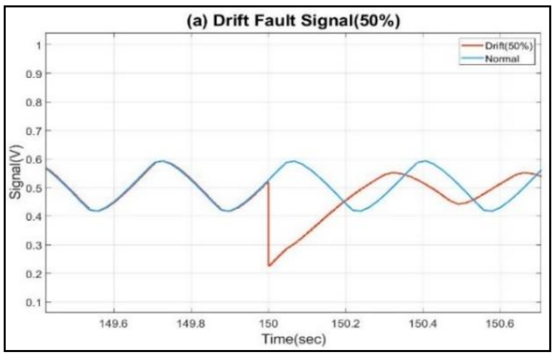
First, the Drift failure mode outputs higher or lower than the value of the normal signal. Automatic FMEA set the drift failure mode using the ‘More Than Requested’ or ‘Less Than Requested’ failure mode. For example, the ‘signal output degradation due to drift’ failure mode of the EGO sensor is explained in the top two rows of Table 3. The first row represents the failure mode of ‘EGO sensor signal output is lowered to 50%’. In this failure mode, the EGO sensor signal output is lowered to 50% as shown in Figure 13-(a). Next, in the second row, we illustrate the failure mode of 'EGO sensor signal output increased to 150%' in Figure 13-(b).

Second, the Hard-over failure mode is characterized by outputting more than the maximum value of the sensor. The Automatic FMEA set the hard-over failure mode using the 'More Than Requested' failure mode. For example, in order to define the maximum value (1.0 V) of the EGO sensor due to hard-over failure mode, the Automatic FMEA defines it as shown in Table 3. In this mode the EGO sensor output is set to 1.0 V as shown in Fig. 13-(c).

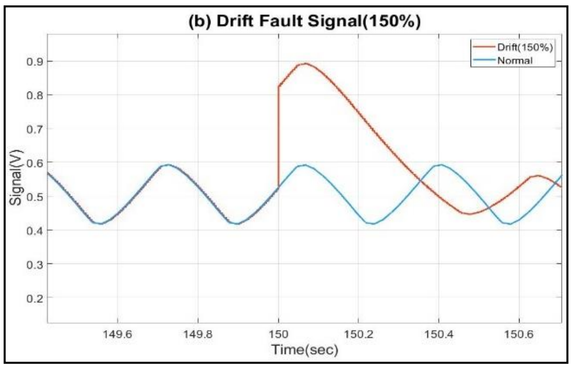
Third, the Stuck has the feature of outputting a fixed signal. The Automatic FMEA uses the 'Locked/Stuck Function' failure mode. For example, in order to explain the 'signal output failure mode due to Stuck' of the EGO sensor, the Automatic FMEA set the output of the sensor to 0.0V as shown in Table 3. In this mode, the output of the EGO sensor is set to 0.0 V as shown in Figure. 13-(d).

Table 3. Sensor Subsystem Failure Mode Table

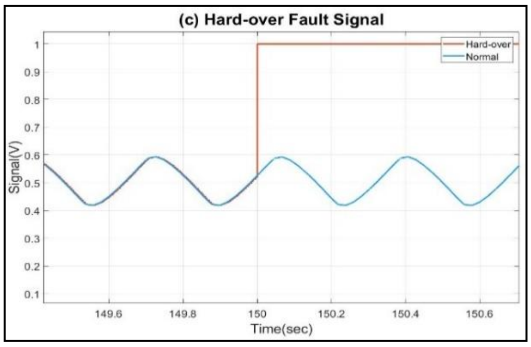
No	Sensor Failure Mode	A-FMEA Failure Mode Condition		
		Fault Type	Value	Time (sec)
1	Drift fault 50%	Less Than requested	$V_{out} = V_{Normal} \times \alpha$, $\alpha = 0.5$	505
2	Drift fault 150%	More Than requested	$V_{out} = V_{Normal} \times \beta$, $\beta = 1.5$	505
3	Hard-over	More Than requested	$V_{out} \geq V_{MAX}$, $V_{MAX} = 1.0$	505
4	Stuck 0	Locked/Stuck function	$V_{out} = V_{Const}$, $V_{Const} = 0.0$	505



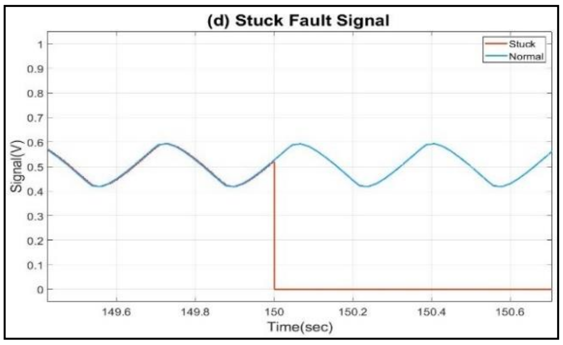
(a) Drift 50%



(b) Drift 150%



(c) Hard-over



(d) Stuck

Figure 10. EGO Sensor failure modes

After defining the failure mode, we define the operation mode of the vehicle model to test the behavior of the EFI sensors. The operation mode refers to a time period in which an EFI failure mode occurs during vehicle operation. The test scenario of the case study consists of two modes: the acceleration mode and the cruise mode. First, in the acceleration mode, we increase the speed of the test vehicle to reach the target speed of 30.6 km/h for 100 seconds. In the next cruise mode, we maintain the speed of 32.2 km/h for 405 seconds. Table 4 explains these two operation modes and vehicle information accordingly. The failure mode and operation mode are recorded to describe the failure mode and mission column of the FMEA template.

Table 4. Operation modes and vehicle information for the test scenario

Test	Operation Mode		Vehicle Information		
Scenario	Start Time	End Time	Velocity	Acceleration	Distance
Accel. Test	0	100	30.6 km/h	0.31 km/s^2	850 m
Cruise Test	100	505	32.2 km/h	0 km/s^2	4475 m

4.2.3. Step 3: Failure effect identification

In this step, we build the rule base to identify the failure effects of the target Simulink model to analyze the failure effects and the severity of the FMEA using the results of the fault injection campaign. The Automatic FMEA user prepares failure effects determination rules according to the three-step procedure. The first step is to determine the type of the failure effect. When a system failure occurs, a target function may halt its execution, or an unexpected malfunction occurs. All malfunctions in the fault simulation are recorded as files. Among these malfunctions, those related to safety are collected as candidates for the next step. The second step determines the output value of the selected candidates to analyze the failure effect. The analyzer identifies the output of the model related to the failure effects

Finally, to assess the severity, the allowable range of data is determined. Automatic FMEA analyzes the failure effect by comparing the normal value and the failure value. The user analyzes the failure effects by presenting an acceptable range of failure values in a quantitative manner. Note that the occurrence of the FMEA can be identified using the similar rule-based system and the fault tree analysis to calculate the failure rate for each failure mode.

We now explain the rule-building process with the EFI Simulink model. In this section, we explain the Sensor failures in the EFI model causing (1) the engine stop, (2) engine hesitation, and (3) gas mileage reduction.

First, the engine stop occurs when the fuel cannot be injected into the cylinder due to a stuck failure of the EFI sensors. If the engine stop failure occurs while driving the vehicle, the driver cannot control the vehicle and the risk of an accident is high. When an engine stop failure occurs, the engine RPM value is reduced to 0. Therefore, we may build a rule for engine stop failure effect with the condition part as the conjunction of the vehicle is running and the engine RPM value converges to 0, which is rule 1 in Table 5. As shown in Table 5, the failure mode category is a specific failure condition check where the condition part of the failure diagnosis rule is 'Engine RPM == 0'. If this condition is met, we set the severity to 8.

Table 5. Failure Effect Table for Severity (RPM represents revolution per minute, FE represents fuel economy figure assigned by EPA [38]

No	Failure Effect	Decision mode option	Condition	Severity
1	Engine stop	specific failure condition check	RPM == 0	8
2	Engine Hesitation	error margin values check	(RPM <1120) && (RPM>1680)	7
3	Engine Hesitation	error margin values check	(RPM <1190) && (RPM>1610)	6
4	Engine Hesitation	error margin values check	(RPM <1260) && (RPM>1540)	5
5	Engine Hesitation	error margin values check	(RPM <1330) && (RPM>1470)	4
6	Gas mileage low	under-estimation values check	$mpg < 20$ (FE 1,2,3)	4
7	Gas mileage low	under-estimation values check	$mpg < 27$ (FE 4,5)	3
8	Gas mileage low	under-estimation values check	$mpg < 33$ (FE 6,7)	2
9	Gas mileage low	over-estimation values check	$mpg > 33$ (FE 8,9,10)	1

Second, the engine hesitation failure effect may occur when the amount of fuel injected into the cylinder fluctuates due to a drift failure of the EFI sensor unit. The engine hesitation may cause noise and vibration while driving the vehicle and may be regarded as a potential threat for the driving safety. We may use four rules from rule 2 to rule 5 in Table 5 to diagnose the failure mode of the engine hesitation. In the rule, the condition elements are used to check the degree of deviation in the input RPM from the average RPM of 5% up to 20% of the golden run simulation. These deviations are used to assign the severity of the failure effects.

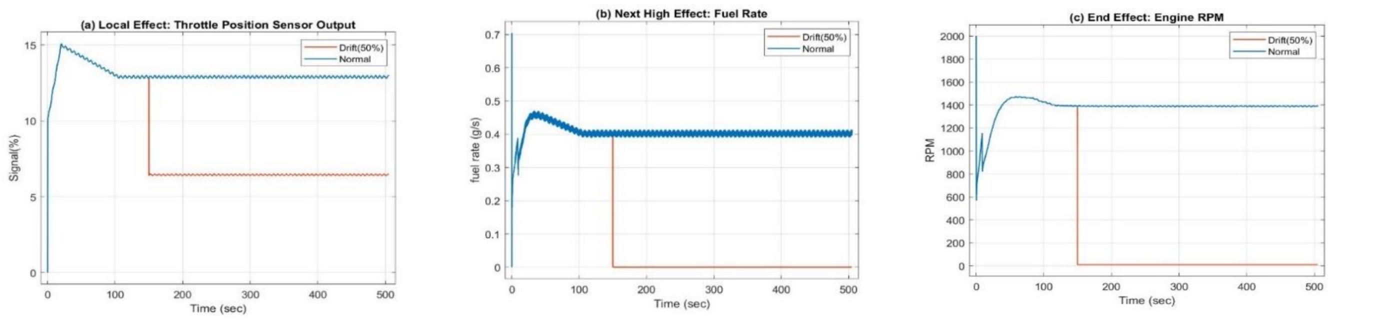
Third, the reduction in gas mileage may occur when the amount of fuel injected into the EFI fluctuates due to the drift failure of the EFI sensor unit. The mileage reduction failure has no direct impact to the safety of drivers but may decrease the performance of the vehicle. We identify the mileage reduction phenomena using the mpg variable of the EFI model. The mpg is calculated by dividing the travelled distance by fuel consumed. Using the fuel economy guideline from the Environmental Protection Agency (EPA), we assign the severity of the gas mileage reduction from 1 to 4 as shown in Table 5 [38].

4.2.4. Step 4: Fault Injection Simulation

The final step of Automatic FMEA performs fault simulation according to the failure mode scenario according to the Table 4. For each failure mode, the corresponding entries of the FMEA template are completed using the results of the automatic FMEA process. The three failure effects for the failure mode of in the Throttle Position Sensor (TPS) are presented in Figure 10. The local effects are measured at the TPS, the next higher effects are measured at the Fuel Rate Controller module, and the end effect are measured at the engine RPM to represent the status of the Vehicle as shown in Figure 10 (a), (b), and (c), respectively.

We performed the simulated fault injection using the fault model to represent the Drift failure mode where the input signal is reduced to 50% of the original signal and injected into the input variable at 150 seconds for about one second. The function, mission, the three failure effects, and the severity of the Drift failure mode at the Throttle Position Sensor are summarized in Table 6.

We performed the simulated fault injection using the Drift failure mode at the Throttle Position Sensor to find the function, mission, the three failure effects, and the severity as shown in Table 6. The fault model is the Drift failure mode where the input signal is reduced to 50% of the original signal and injected into the input variable at 150 seconds to 505 seconds. Once this fault model is injected into EFI model, we can find the local effect of the failure effect by monitoring the TPS where the value of the signal fell from 13% to 6.5%, which is shown in Figure 10(a). The next effect is detected at the Fuel Rate Controller module where the value of fuel rate drops to 0 g/s at 150 seconds as shown in Figure 10(b). Finally, the end effect is observed at the Engine module which shows sudden drop from 1400 RPM to 0 RPM as shown in Figure 10(c). These three pieces of information illustrates the occurrence of the event of the vehicle stop so that we can assign the severity of this failure effect to 8.



(**a**) Local Effect (**b**) Next High Effect (**c**) End Effect

Figure 11. The failure effect of TPS fault mode after fault injection simulation. (a) Local effect at the Sensor module, (b) next higher effect at the Fuel Rate Controller module, (c) end effect at the Engine Gas Dynamics module

Table 6. FMEA table of TPS fault mode of EFI Simulink model

Item	**Function**	**Function info**	**Failure Mode**	**Mission**	**Local effect**	**Next high effect**	**End effect**	**Severity**
Throttle Position Sensor	Data Sensing	Throttle Position Sensing	Drift (505)	Cruise (150-151)	TPS output *($\leq 3^\circ$)	Fuel rate *($0\ g/s$)	Engine RPM *($0\ rpm$)	8

Automatic FMEA can analyze the failure effects by including FMEA severity analysis using failure simulation. We performed a total of 328 fault injection simulation on four sensors in the EFI sensor unit for the Drift, Hard-over, and Stuck failure modes and summarized the results in Table 7. In the table, the failure effects were grouped into three categories: engine stop, engine hesitation, and gas mileage reduction. From the table, we can show that the contribution of the four sensors to the criticality to the EFI system is different from each other.

Among the four sensors, the sensor with the highest number of failures of 59 is the CPS sensor. On the other hand, the EGO sensor was the most vulnerable to the critical failure effects such as the Hard-over and the Stuck failure effect. From this information, we should emphasize the quality of the EGO and CPS sensor are more important than the others.

Table 7. Analysis results of the failure effects of the TPS, CPS, EGO, and MAP sensor

Sensor Failure Effects	**TPS**			**CPS**			**EGO**			**MAP**		
	Drift	**Hard over**	**Stuck**	**Drift**	**Hard over**	**Stuck**	**Drift**	**Hard over**	**Stuck**	**Drift**	**Hard over**	**Stuck**
Total Tests	40	40	2	40	40	2	40	40	2	40	40	2
Masked	30	31	0	12	11	0	30	0	0	17	18	0
Engine stop	6	5	1	14	23	2	10	40	2	15	15	1
Hesitation	4	4	1	11	2	0	0	0	0	6	6	1
Low Mileage	0	0	0	3	4	0	0	0	0	2	1	0
Total Failure	10	9	2	28	29	2	10	40	2	23	22	2

4.3. Comparison of Automatic FMEA tool with Legacy FMEA tool

In order to illustrate the impact of the presented tool, we built two FMEA for the drift failure of the throttle angle position sensor (TPS) using a legacy FMEA tool and the Automatic FMEA tool [32]. First, the legacy FMEA is presented in Table 8(a). Here we should organize the FMEA committee with the related stakeholders including the developer and maintenance experts to identify the failure modes and failure effects of the TPS. The human expert may identify the drift error of TPS may cause the problem with the air/fuel mixture in the fuel injector, resulting in engine hesitation. In addition, based on expert opinions, the severity, occurrence, and detection of the failure effects are presented.

Table 8. Analysis results of the failure effects of the for Automatic FMEA sensor of TPS, CPS, EGO, and MAP.

(a) Legacy FMEA Table

Item	**Function**	**Function Info.**	**Failure Mode**	**Mission**	**Effect**			**S**	**O**	**D**	**RPN**
					Local	**Next**	**End**				
TPS	Sensor	Throttle Position Sensing	Signal Error	Cruise	TPS Signal Error	Lack of Fuel	Engine Hesitation	6	2	1	12

(b) Automatic FMEA Table

Item	**Function**	**Function Info.**	**Failure Mode**	**Mission**	**Effect**			**S**	**O**	**D**	**RPN**
					Local	**Next**	**End**				
TPS	Sensor	Throttle Position Sensing	Signal Error: Type: Permanent Start time: 100s End time: 505s Signal size: 10%	Cruise	TPS Signal Error: Avg. error: 10% Max. error: 10% Min. error: 10%	Lack of Fuel Avg. error: 10% Max. error: 11% Min. error: 9%	Low Gas Millage Avg. error: 9% Max. error: 9% Min. error: -	4	2	2	16
			Signal Error: Type: Transient Start time: 100sec End time: 505sec Signal size: 40%	Cruise	TPS Signal Error Avg. error: 20% Max. error: 20% Min. error: 20%	Lack of Fuel Avg. error:20% Max. error: 27% Min. error: 18%	Engine Hesitation Avg. error: 15% Max. error: 25% Min. error: 7%	6	2	1	12
			Signal Error: Type: Permanent Start time: 100sec End time: 505sec Signal size: 60%	Cruise	TPS Signal Error Avg. error: 60% Max. error: 60% Min. error: 60%	Lack of Fuel Avg. error:100% Max. error: 100% Min. error: 100%	Engine stop: Avg. error: 100% Max. error: 100% Min. error: 100%	8	1	1	8

Next, the same Drift failure mode in the TPS is analyzed using the Automatic FMEA as shown in Table 8(b). Unlike the legacy FMEA, the Automatic FMEA specifically investigates the drift failure mode of the TPS thoroughly. Here, many tests with various types, frequencies, and failure durations can be executed in the fault injection simulation environment. The results of each simulation can be summarized to build the failure effects of the failure mode of the TPS. Using these data, we can not only record the entries of the FMEA templates but also identify the threshold value, average value, minimum/maximum value of the variable of interest.

As shown in Table 8(b), when the size of the TPS signal was reduced by 10% of the original signal, there was no significant impact to the safe operation of the vehicle, but the value of the engine RPM was reduced into 9% of the golden run simulation. However, when the size of the TPS signal reduced to 40% and 60%, the vehicle runs into the safety-threatening mode such as the engine hesitation mode and the engine stop mode, respectively. By proper and extensive preparation of the test cases for the given failure mode, Automatic FMEA can calculate the value of the key performance variables to decide the failure effects and the severity for the given failure mode.

We can summarize the advantages of the Automatic FMEA as follows: First, using the fault injection campaign, we may find the detailed causality information such as the threshold or the acceptable range of the key performance variables. In the case of the legacy FMEA, one may identify the known failure effect of the given failure modes. Thus, the Automatic FMEA may provide the in-depth causality between the failure modes and the failure effects.

Second, data driven failure diagnosis can be utilized. In the target design model, we may setup the monitoring probes, as many as desired, to build the database of the sensor data. These data can be used to diagnose the failure event using deep learning or neural network. Unlike the legacy FMEA, the proposed FMEA may open a new horizon toward the failure diagnosis or a prognosis for the future failure events.

Third, we can setup various fault models in many different operating environments. While the legacy FMEA is used to identify the failure effects using the knowledge and experience of the participating member of the FMEA committee, the proposed FMEA can setup the various test cases and the tool will provide summarized results to the experts to find rare failure events that could be overlooked.

5. Conclusions

We explained the Automatic FMEA that integrates the model-based design and the simulated fault injection tool to facilitate the failure modes and effect analysis (FMEA) process. Although the FMEA process is the key activity in the development of the safety-critical systems, we could not utilize the merit of this process due to the various difficulties such as the difficulty in organizing and proceeding of the FMEA committee and scarcity of the experts especially when the target system is a new product. To improve these limitations, Automatic FMEA may contribute to overcome these processes by integrating the model-based development, the fault injection simulation, and the rule-based failure diagnosis.

Automatic FMEA is designed to perform the fault injection simulation on the Simulink model which has the largest users in the domain of model-based development. It automatically generates FMEA reports by performing the fault injection simulation for each failure mode and produces the three failure effects. The severity information for the risk priority number is convincing since we can use the traceability causality between any failure modes and the failure effects and the rule-based failure diagnosis technique. The tool can investigate those events that may be overlooked by building a large number of test cases based upon the HAZOP keywords and performing simulated fault injections accordingly. These processes are integrated into a single environment to help the safety engineers. In order to demonstrate the applicability of the Automatic FMEA, we used the electronic fuel injection system (EFI) model using four sensors. The results of the Automatic FMEA were compared to that of the legacy FMEA.

The use of Automatic FMEA will facilitate the development process of the safety-critical field such as aerospace, automotive, and railway. We are in the process of integrating the fault tree analysis tool with our tool so that we can calculate the occurrence parameter for the given failure mode. Using the severity and the occurrence figure, we can calculate the criticality figure for every possible failure mode and rank the failure mode according to the criticality. This result can be crucial for the certification authorities, developers, as well as consumers in the evaluation of the safety in the system.

1. International Standard, "Railway Application: Communications, signaling and processing systems - Safety related electronic system for signaling", IEC 62425 Ed. 1, 2005.

2. SAE ARP 4761, "Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment", December 1996.
3. RTCA Inc., DO-178C: Software Considerations in Airborne Systems and Equipment Certification, Washington, DC:RTCA Inc., 2011.
4. International Standard Office (ISO), "ISO 26262 Road Vehicles - Function Safety - Part 5: Product development at the hardware level", International Standardization Organization Std., Dec. 2018.
5. RTCA Inc., "DO-254, Design Assurance Guidance for Airborne Electronic Hardware", DC:RTCA Inc., 2000.
6. SAE ARP4754, "Certification Considerations for Highly-Integrated or Complex Aircraft Systems", 1996.
7. STEINKE, Alexander; KOPP, Bruno. "RELEX: An Excel-based software tool for sampling split-half reliability coefficients. Methods in Psychology", 2020, 2: 100023.
8. ETI, M. C.; OGAJI, S. O. T.; PROBERT, S. D. Integrating reliability, availability, maintainability and supportability with risk analysis for improved operation of the Afam thermal power-station. Applied Energy, 2007, 84.2: 202-221.
9. TORTORELLA, Michael. Reliability, maintainability, and supportability: best practices for systems engineers. John Wiley & Sons, 2015.
10. B. S. Blanchard, W. J. Fabrycky and W. J. Fabrycky, Systems engineering and analysis, vol 4. Prentice hall Englewood Cliffs, NJ, 1990.
11. COOPER, Howard C. Capture all critical failure modes into FMEA in half the time with a simple decomposition table (Actual case study savings= $4,206,000). In: 2015 Annual Reliability And Maintainability Symposium (RAMS). IEEE, 2015. p. 1-6.
12. FENG, Xiangli, et al. Functional model-driven fmea method and its system implementation. In: 2018 12th International Conference on Reliability, Maintainability, and Safety (ICRMS). IEEE, 2018. p. 345-350.
13. HECHT, Herbert; AN, Xuegao; HECHT, Myron. Computer aided software fmea for unified modeling language based software. In: Annual Symposium Reliability and Maintainability, 2004-RAMS. IEEE, 2004. p. 243-248.
14. HODKIEWICZ, Melinda, et al. An ontology for reasoning over engineering textual data stored in FMEA spreadsheet tables. Computers in Industry, 2021, 131: 103496.
15. MARIANI, Riccardo; BOSCHI, Gabriele; COLUCCI, Federico. Using an innovative SoC-level FMEA methodology to design in compliance with IEC61508. In: 2007 Design, Automation & Test in Europe Conference & Exhibition. IEEE, 2007. p. 1-6.
16. SELWYN, T. SUNDER; KESAVAN, R. FMECA Analysis of Wind Turbine Using Severity and Occurrence at High Uncertain Wind in India. International Journal of Applied Engineering Research, 2015, 10.11: 2015.
17. KELLNER, Darryl W. Software FMEA: A successful application for a complex service oriented architecture system. In: 2017 Annual Reliability and Maintainability Symposium (RAMS). IEEE, 2017. p. 1-5.
18. ZHANG, Xu, et al. Product model-based design process modeling in collaborative design. In: 2010 IEEE International Conference on Industrial Engineering and Engineering Management. IEEE, 2010. p. 315-319.
19. BESHEARS, Raymond; BOUMA, Andrew. Engaging Supportability Analysis through Model-Based Design. In: 2020 Annual Reliability and Maintainability Symposium (RAMS). IEEE, 2020. p. 1-5.
20. KELLNER, A., et al. Design and use of system models in mechatronic system design. In: 2015 IEEE International Symposium on Systems Engineering (ISSE). IEEE, 2015. p. 142-149.
21. FIORUCCI, Tiziano, et al. Automated Dysfunctional Model Extraction for Model Based Safety Assessment of Digital Systems. In: 2021 IEEE 27th International Symposium on On-Line Testing and Robust System Design (IOLTS). IEEE, 2021. p. 1-6.
22. HUANG, Zhao; HANSEN, Raymond; HUANG, Zhaofeng. Toward FMEA and MBSE integration. In: 2018 Annual Reliability and Maintainability Symposium (RAMS). IEEE, 2018. p. 1-7.
23. KAUKEWITSCH, Christof, et al. Automatic generation of RAMS analyses from model-based functional descriptions using UML state machines. In: 2020 Annual Reliability and Maintainability Symposium (RAMS). IEEE, 2020. p. 1-6.
24. GAO, Ting, et al. Circuit FMEA Method by Fault Simulation Based on Saber. In: 2019 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE). IEEE, 2019. p. 1039-1045.
25. A. Hess, J. S. Stecki and S. D. Rudov-Clark, "The Maintenance Aware Design Environment: Development of an Aerospace PHM Software Tool", PHM Technology, 2008.
26. REKIK, Fadwa, et al. Model-driven consistency verification for service-oriented applications. In: 2015 IEEE 8th International Conference on Service-Oriented Computing and Applications (SOCA). IEEE, 2015. p. 180-187.
27. "APIS IQ-Software | FMEA | DRBFM | Functional Safety", APIS Informationstechnologien GmbH. https://www.apis-iq.com/ (11 18, 2021).
28. TANG, Ting, et al. FTA and FMEA of braking system based on relex 2009. In: Proceedings of International Conference on Information Systems for Crisis Response and Management (ISCRAM). IEEE, 2011. p. 106-112.
29. CHOI, Yong-Jae; CHOI, Gyeong-Hee; CHUNG, Gi-Hyeon. Parsing simulink mdl file and C# data structure for Testing. In: Proceedings of the Korean Information Science Society Conference. Korean Institute of Information Scientists and Engineers, 2010. p. 414-418.
30. TOOLBOX, Symbolic Math, et al. Matlab. Mathworks Inc, 1993.

31. "Modeling a Fault-Tolerant Fuel Control System", https://kr.mathworks.com/help/simulink/slref/modeling-a-fault-tolerant-fuel-control-system.html (17 09 2021).
32. Il-Kwon Lee, et al. A study on the sensor of electronic control system. Korea Tribology Society Conference, 2008, 9-16.
33. Seong-Yong Lim, In-Hyeok Jang, Young-Ju Lee/Hong-Woo Lee/Im Hong-Woo, "Improvement through failure mode and failure mechanism of piezo sensors used in AVC", Proceedings of the Korean Reliability Society Conference, pp 313–320, 2015.
34. JAN, Sana Ullah, et al. Sensor fault classification based on support vector machine and statistical time-domain features. IEEE Access, 2017, 5: 8682-8690.
35. KIM, Sang-Chul; KIM, Doo Hyun; KANG, Shin Uk. Analysis of Voltage, Current and Temperature Signals for Poor Connections at Electrical Connector. Journal of the Korean Society of Safety, 2014, 29.2: 12-17.
36. YANG, Jae-Wan; LEE, Young-Doo; KOO, In-Soo. Timely Sensor Fault Detection Scheme based on Deep Learning. The Journal of the Institute of Internet, Broadcasting and Communication, 2020, 20.1: 163-169.
37. Yang, Jae-Wan; Lee, Young-Doo; Gu Insoo. "Deep learning and support vector machine-based sensor failure detection techniques.", Journal of the Korean Internet and Telecommunication Society, 2018, 18.2: 185-195.
38. "Learn More About the Fuel Economy Label for Gasoline Vehicles". https://www.fueleconomy.gov/feg/label/learn-more-gasoline-label.shtml (01 22 2022).
39. Joshi, A., Heimdahl, M. P., Miller, S. P., & Whalen, M. W. (2006). Model-based safety analysis (No. NASA/CR-2006-213953).
40. FRITZSON, Peter. Introduction to modeling and simulation of technical and physical systems with Modelica. John Wiley & Sons, 2011.