

Efficient Integration for a Hardware-In-the-Loop (HIL) System

2010-01-0665

Published
04/12/2010Yanan Zhao, Fangjun Jiang and Zhang Yan
Ford Motor Company

Copyright © 2010 SAE International

ABSTRACT

The integration of a HIL system involves many software and hardware elements. During the integration process, various errors may occur. In this paper, the Design For Six Sigma (DFSS) methodology is applied to develop an integration process for a generic type of HIL system. The development follows the DCOV (Design, Characterize, Optimize and Verify) four-phase approach. The result is an efficient process proven to meet the customer's needs, reduce integration errors and deliver a high quality HIL system within less time.

INTRODUCTION

Hardware-in-the-loop (HIL) systems have become a key element in control system development. There are many types of HIL systems [1, 2, 3]. Some HIL systems involve significant amount of hardware, such as hydraulic devices, motor electronics and mechanical chassis. Others, such as the HIL system in this study, involve mostly Electronic Control Unit (ECU) hardware. This system contains a simulator running a vehicle plant model in real-time to emulate a physical automotive vehicle. The simulator is connected through hardwired signals to an ECU hardware module which executes the controller software. The HIL system is used primarily to conduct design verifications of the controller software without a physical vehicle.

Many software and hardware elements, such as a vehicle plant model, an electronic control module and a simulator, are required for a HIL system. The development of a HIL system involves many tasks such as plant model development, interface design and hardware integration. The first time when all these elements from various sources come together is during the HIL integration process. It is during this process that many different types of issues may surface.

Most of the literature related to HIL systems is focused on how to set up a particular HIL system or how to use the system to solve a particular problem. This study is focused on developing a process to integrate a generic HIL system efficiently. Specifically, the Design For Six Sigma (DFSS) approach is applied to develop an efficient integration process for a high quality HIL system with minimum integration errors.

DFSS is a methodology to design a system to meet the very high Six Sigma quality standard. It utilizes a set of statistical tools and systems engineering techniques to meet the customer's needs, avoid mistakes and achieve high quality at the onset of the system. The DFSS process consists of four phases, namely Design, Characterize, Optimize and Verify. The sections in this paper follow these four phases. First in the Define phase, the problem is described, the voice of the customer is collected and the Critical To Quality (CTQ) metrics of the process are defined. In the Characterize phase, a robust concept for HIL integration is first generated. Six Sigma tools such as the boundary diagram, interface matrix, Failure Mode and Effect Analysis (FMEA), Parameter diagram (P-diagram), Robustness Check List (RCL) and Robustness Demonstration Matrix (RDM) are utilized to identify the project scope, system interaction, potential failure modes, noise factors and noise factor management plans. An initial HIL integration process is also proposed in this phase. In the Optimize phase, the HIL integration process is optimized in several major areas to be robust to the critical noise factors. Lastly in the Verify phase, the optimized HIL integration process is verified with two hybrid powertrain projects to demonstrate its effectiveness.

DEFINE

The goal of the Define phase is to define the problem and to understand the customer and the history. The first task is to develop a project charter that defines a problem statement.

business case and scope, project team and work plan. Other tasks include benchmarking industry and quality history, identifying Critical To Satisfaction (CTS) items and related functional targets.

The quality history of the Dual Drive hybrid powertrain [1, 4, 5] Vehicle System Control (VSC) HIL integration was collected. It revealed that no formal integration process was established. Engineers followed an ad hoc process for system integration. Many integration errors occurred, including interface mismatch, incompatible software, and incorrect models and configurations. Overrun was another issue of the above HIL system. Turnaround time is the time it takes the HIL simulator to run the vehicle plant model for one simulation step. Overrun occurs when the turnaround time is larger than the simulation step size. In such a case, the HIL simulation results do not represent the real time situation anymore.

<table 1 here>

It was clear that the quality and efficiency of the HIL integration process needed improvement. The problem statement is quite simple. It is to develop an efficient process for integrating a high quality HIL system with minimum integration errors. The business case is also very strong. The process developed can be broadly applied to various vehicle programs in many organizations. The DFSS project assumes that the system is a generic HIL system with vehicle plant model and controller software and hardware provided. It also assumes that the hardware and software environment for the HIL laboratory already exist.

The voice of the customer was then collected and analyzed. The Kano model and Quality Function Deployment (QFD) tool were used to prioritize and map customer CTS to technical metrics called Critical To Quality (CTQ). Table 1 shows the QFD matrix. Kano model analysis indicated that the first three CTSs are basic quality requirements. The fourth and fifth are performance quality requirements while the last one is an excitement quality requirement. Thus different values were given to these CTSs to indicate their importance. The CTQs are basically how those CTSs can be met. The rating of High (H), Medium (M) and Low (L) in the rest of the matrix is called the Cause and Effect Matrix indicating the relationship between CTS and CTQ. By giving a weight to the ratings (H=9, M=3 and L=1), the weighted sum of each CTQ was calculated at the last row. Two CTQs with the highest score were selected as the metrics for the DFSS project. Since the vehicle plant model is provided as an input to the HIL integration process, it was not chosen as a metric.

The selected CTQs are: 1) Integration errors. The target is to reduce the number of integration errors by at least 60%. (It was analyzed that 60% was desired for the objective of this

project.) 2) Overrun issue. The target is to have zero overrun issues.

CHARACTERIZE

The objective of the Characterize phase is to generate robust concepts, understand the system and characterize robustness opportunities. This phase applies many tools such as the boundary diagram, interface matrix, FMEA, P-diagram, RCL and RDM. Each will be explained in detail in the following sub-sections.

GENERATE ROBUST CONCEPT

The robustness of a system indicates its consistency of performance of the ideal function around the target with minimal error states in spite of the effects of noise. To generate a robust concept, a survey and a benchmarking were conducted on seven internal HIL systems developed for various hybrid and fuel cell programs. The survey studied the HIL systems in terms of simulator hardware, ECU module, experimental software, control software, simulation needs and testing capabilities. Based on the survey and analysis, two concepts were proposed for HIL integration. One is called the one-step approach and the other is called the two-step approach. The two-step approach involves integrating the vehicle plant model and controller software into a Model-In-the-Loop (MIL) system first and then integrating the MIL system into the HIL system. A one-step approach, on the other hand, involves integrating the vehicle plant model and controller software directly into the HIL system. It usually starts with adding hardwired and Controller Area Network (CAN) I/O to the vehicle plant model right away without the intermediate MIL step.

A MIL model comprises a vehicle plant model, a controller model and an interface mapping model in between. The interface mapping matches I/O signals between the vehicle plant model and the controller model and performs necessary unit conversion, signal grouping, signal mapping and additional signal sourcing. The HIL model is essentially the same as the MIL model except that the controller model is taken out from the MIL and run in the ECU. Thus, additional blocks need to be added for the HIL to access hardware devices. These blocks are usually Low/High Level Drivers (LLDs/HLDs) representing the I/O interface between the HIL simulator and the ECU hardware.

The vehicle plant model has an established model structure called the Vehicle Model Architecture (VMA) [5]. It contains a "HIL-Online" block designed for LLDs/HLDs. When moving from the MIL to the HIL, the controller model is removed and the empty block is replaced by a "HIL-bypass" block to keep the integrity of the VMA. The "HIL-bypass" block contains mostly signal routing which re-routes the input and output signals of the plant model.

The Pugh concept selection tool was used to compare the one-step approach and the two-step approach. Based on the key attributes such as the ability to analyze real-time overrun issues early on, the ability to debug HIL integration errors, the ability to validate the HIL system and the timing of the HIL system availability, it was determined that overall the two-step approach is better than the one-step approach.

<figure 1 here>

BOUNDARY DIAGRAM AND INTERFACE MATRIX

A boundary diagram graphically defines the scope of a system under study. It is an important tool to identify the major components of the system, understand how they interact with each other and how they interact with outside systems and environments. It provides detailed information to be used as input to the interface matrix, P-diagram and FMEA.

A boundary diagram as shown in [Figure 1](#) was constructed for the HIL integration study. There are 22 major components representing the tasks involved in the MIL and HIL integration. Each component was identified with its name plus a sequence number ranging from “A1” to “A22”. The lines represent interactions. The solid blue lines represent “in system direct interface”. The dotted blue lines represent “in system indirect interface”. The solid red lines represent “direct interface” and the dotted red lines represent “indirect interface”.

A system interface matrix was then created to document the details, such as the type, the strength or importance and the potential effect of the interfaces. This interface matrix acts as an input to the creation of FMEA and P-diagram.

DESIGN FMEA

FMEA is a method to identify failure modes within the design and prioritize them according to the severity of the effect, the frequency of the occurrence and the ability of detection. It also identifies what type of prevention and detection activity can be applied to prevent the cause (mechanism) of the failure mode from occurring.

A design FMEA was conducted for the HIL integration process. It shows that the following items have the highest rank of Risk Priority Number (RPN): 1) signal conversion, RPN=280; 2) signal mapping, RPN=280; 3) high level driver of hardwired signals, RPN=224. These prioritized items were studied along with important noise factors in the Optimize phase to improve the process.

<figure 2 here>

P-DIAGRAM, RCL AND RDM

A P-diagram is used to identify the ideal outputs of the product or process in the context of inputs and noises. It also identifies the possible error states that the product or process might lead to and the control factors that can be used to mitigate the noises and error states. By brainstorming, the P-diagram of the HIL integration process was created in [Figure 2](#).

Next, a tool called the Robustness Check List (RCL) was used to 1) identify high impact noise factors, 2) associate error states with noise factors, 3) identify a Noise Factor Management Strategy (NFMS), 4) record important Verification Methods (VMs) Plans, 5) correlate VMs with error states to show that a test exists to detect the error state, 6) correlate VMs with noise factors to show that the noises that induce the error state have been included in the test. In the RCL created for the HIL integration, all the error states identified in the P-diagram were classified as important. A NFMS was proposed and classified into categories such as “change technology”, “apply parameter design”, “reduce/remove noise”, etc. The four VMs proposed were: 1) model compile/build; 2) MIL desktop simulation, 3) HIL test bench simulation and 4) vehicle testing. The vehicle testing is identified as a non-high impact VM.

<figure 3 here>

The RDM usually contains VM descriptions and targets, high impact error states and high impact noise factors associated with each VM, demonstrated results and issues. A RDM for the HIL integration was initiated by taking inputs from the RCL in the Characterize phase and was finalized in the Verify phase.

PROCESS MAP

Based on the existing work flow and the decision to take the two-step approach, a process map for the HIL integration was proposed in [Figure 3](#). It shows the inputs and outputs of the process. It also shows many of the integration tasks and decision points needed to complete the HIL integration. This process serves as the initial design and was optimized in the Optimize phase to achieve the Six Sigma quality standard.

OPTIMIZE

The objective of this phase is to optimize system performance and minimize process sensitivity to noise factors.

IMPORTANT NOISE FACTORS

To identify the most important noise factors, the quality history of the MIL/HIL integration for the Dual Drive hybrid powertrain VSC was analyzed in [Table 2](#). The errors were tracked to different categories and to the locations in the

process map where they occurred. Based on the opportunities for each particular step, the throughput yield was calculated. A Pareto chart was then created to analyze which steps contributed the most to the integration errors. It was revealed that all of these steps have roughly equivalent throughput yields with an exception of A17 (due to its low number of opportunities). It was also revealed that I/O interfacing work including signal mapping, signal conversion and additional signal sourcing between plant/MIL/HIL caused most of the errors.

<table 2 here>

Based on the above observation and the noise factors listed in the P-diagram, it was identified that the majority of the integration errors were caused by the “System Interaction” noise factors, specifically, item a) “unclear/incorrect/mismatched signal documentation and I/O specification” and item d) “inadequate specification regarding deliverables among development groups”.

ROBUSTNESS DESIGN

In the P-diagram, both noise factors and control factors were identified. The functions of the control factors are to eliminate or minimize the effects of the noise factors. In the RCL, a general noise factor management strategy was proposed. In [Table 3](#) some design details are given as examples for the robustness design. Issue #1 lists the most important noise factor. The solution is to adopt the CAN database as the specification for signal interfaces because it is usually well maintained and the majority of the I/O signals are CAN signals. Also, integration automation is available through the use of the CAN database. With this design, the roles, responsibilities and deliverables among four development groups (plant modeling group, controller software group, MIL integration group and HIL integration group) can be clearly specified. For example, the plant modeling group shall deliver a vehicle plant model with the CAN message packing and unpacking represented in a CAN bus.

<table 3 here>

For Issue #2, the incorrect or inadequate modeling and incorrect calibration for the plant model can be detected by verifying whether the plant model meets modeling requirements. For an overly complex vehicle plant model, this problem usually is also part of the HIL system overrun issue. Issue #3 indicates that the noise factor “insufficient HIL simulator processor capability” sometimes leads to HIL system overruns. The root cause can be traced down with a fault-tree diagram. Based on the diagram, a procedure for analyzing and solving the overrun issue was developed. First, the HIL integration group shall provide a HIL simulator capability specification to the vehicle plant modeling group.

The HIL group shall then estimate the real-time execution time of the plant model immediately after the plant model is released and also verify the HIL simulator capability after HIL plant model integration is completed. In the case of an overrun, a MIL Simulink Profiler and HIL execution time analysis can be used to find the root cause. The problem can generally be solved by simplifying the vehicle plant model, increasing modeling efficiency, or setting the optimal combination of simulation step size, solver type and fidelity of the vehicle plant model. As a last resort, the HIL simulator hardware could be upgraded.

OPTIMIZED PROCESS MAP

With the above proposed robustness design, the HIL integration process was optimized with several new steps and simplified steps as shown in [Figure 4](#). The new process provides improvement by four measures: 1) Integrate the I/O interface through a standard method by automation, 2) Eliminate re-work on the I/O interface between MIL and HIL, 3) Conduct early-on studies on turnaround time and 4) Conduct model compliance checks and additional milestone reviews.

<figure 4 here>

[Figure 5](#) illustrates the idea of integrating the I/O interface through a standard method by automation. The left half illustrates the “Before” practice. A “Wrapper” was used to provide signal mapping, conversion and additional signal sources between the vehicle plant model and controller model. Quite often the “Wrapper” has to be modified many times to fix missing signals, mismatched signal mapping and attributes conversion. The “After” practice is illustrated in the right half. The I/O interface mapping is CAN message-based and the CAN packing and unpacking blocks are automatically generated using the released CAN database file. This approach not only guarantees the completeness of the I/O interface signals, but also provides the MIL simulation with enhanced capability since it now represents the actual CAN communication in the model. The integration of the I/O interface is now more efficient and robust to human errors and modeling style variations.

[Figure 5](#) also compares the work needed to move from MIL integration to HIL integration. In the “After” process with the MIL interface resembling the CAN I/O and the hardwired I/O signals (see steps O5 to O10 in [Figure 4](#)), the MIL and HIL interfaces are essentially equivalent. The same I/O interface eliminates the re-work from MIL to HIL and shortens the time required for the HIL integration. The new process also allows the I/O to be debugged earlier during the MIL integration. Note that the HIL bypass block is not shown. In general it still exists but is significantly simplified.