

An Effective Modeling Architecture for MIL, HIL and VDIL Testing

Hamid Alper Oral
Bosch Rexroth Corp.

ABSTRACT

Model based design has proven itself as a key enabler to accelerate the development and enhance the quality of embedded software during the last decade. The tools serving the model based design have continuously improved both in terms of software and hardware, and have achieved a maturity level capable to provide seamless transition across different development phases. This approach provides a capability to ensure the satisfaction of key requirements not only at controls development level but also at system level directly addressing the customer requirements. This paper demonstrates a modeling architecture utilizing this capability, allowing the user work on the complete system model that is capable to maintain the same model architecture starting from pure model simulation phase working its way through vehicle testing, contributing to the reduction of the development time and system integration effort while improving the utilization of the model based tools and methods. The novel approach is presented along with specific examples demonstrating a number of advantages that can be gained through out the development.

CITATION: Oral, H., "An Effective Modeling Architecture for MIL, HIL and VDIL Testing," *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.* 6(1):2013, doi:10.4271/2013-01-0154.

INTRODUCTION

The role of software and associated hardware has been increasing exponentially in automotive industry. Development phases to design and verify controls using high level software with graphical user interface have enabled engineers to build algorithms using blocks with flowing signals and discretely changing states much more effectively compared to writing code by hand. The graphical model design environments have offered a lot of flexibility to the users to build their models. Software Style Guidelines were established to spread the best practices among the users building models [1]. As the size of the system models grew, to avoid duplication of work and improve collaboration among users, it became necessary to come up with a common system level architecture to achieve ease of integration among previously developed and validated subsystems [2], [3]. In 2003 Automotive Open System Architecture (AUTOSAR) was established mainly to standardize the software architecture for the automotive industry [4]. As AUTOSAR gradually found acceptance in industry, the tool developers provided means for users to integrate these standards into their applications [5]. In addition, alternative tools were made available to maximize the model reuse allowing the users to build complete systems across large pool of models [6], [7]. Instead of off the shelf solutions,

some companies built customized model architecture methods targeted for their own application development process [8].

The history of hardware-in-the-loop testing is as old as that of electronic control units. Aerospace HIL applications dating back to 1960's can be seen displayed in museums or in old institutions. Embedded code generation however came after wide spread usage of personal computers for model based development and it has been around for more than a decade. As it is well known embedded code generation tools enable rapid prototyping controls and hardware-in-the-loop testing that execute models in real time. During recent years the capabilities provided by model based embedded code generation tools and the expertise in industry have reached a certain maturity level to build production code for a controller without having to use any hand code [9] [10] [11]; it is now possible to have access to toolboxes available for embedded code generation for production level quality software that are commercially available. That is why the reader is requested to interpret the controls models in figures across this article capable to be embedded into hardware for a controls application without any hand coding requirement as it has been demonstrated in the literature. The implementation referenced in literature has been so far limited to controls software alone. The architecture presented

in this paper is for a complete system that includes not only the controls but also the plant which the controller is designed for. The method presented in this paper is generic enough to be used for system development for any given application. The examples are extracted from a hybrid vehicle hydraulic powertrain rapid prototyping controls application. Given the specific driver blocks used from the controller vendor are replaced to suit the specific hardware, the model architecture, which is the main point of this paper, remains the same for production code development.

MODEL ARCHITECTURE

Method

The main advantage of the model based development approach presented in this paper is that it integrates both embedded software and physical system models allowing the model based developers to have a complete access to all the systems, their components and the associated parameters that goes into characterizing, calibrating and tuning the system at pure simulation environment where model based tools can be implemented to build the steps targeting the satisfaction of customer requirements.

As it is commonly experienced, throughout the development of a system, the knowledge about the system in general and components in particular increase as a result of active learning process. One of the differences of the presented method in this article from the traditional approach is that as the work progresses through the development stages the model changes have to be reflected back at the model created at initial development stage. This could actually be considered as the main strength of this method because of two reasons:

First, the model created at initial stage is where critical decisions are made in regards to the customer requirements. To maintain the initial premises in regards to customer and associated system, subsystem, and component requirements, it is important to update the same model and check if the premises still hold true. A model change introduced during development should be followed by requirements verification testing to see the end effect on the performance and return of investment. The propagation of the model changes can easily be accomplished seamlessly by using the same model libraries and/or model reference blocks across the development stages according to the proposed model architecture.

Second, the methods that allow or even force the developers spread the information across different xIL phases disconnects the previously developed models from the key information learned during the previous development. Eventually the accumulated information end up being distributed across various models used at different stages. It becomes then a challenge to account for all the knowledge gained to be leveraged in a robust manner to the next development project. Following the presented approach in this paper at the end of the development phases, a complete

system model is obtained, which is a whole representation of the accumulated knowledge with a smaller foot print organized in a manner that can be executed in pure simulation environment and ready to be leveraged for the next project. With this approach the speed of the development can be increased a lot faster than traditional methods as the number of projects increase.

Requirements for the Model Architecture

Usually the term model architecture means architecture for a controls software embedded in controller hardware. However the work presented here targets the complete system architecture that includes both the embedded software and the model of the physical system that controller is designed for.

There are several factors affecting a system model and its architecture in regards to model based development. First of all it is critical to have a system model that captures all the requirements. That is because most of the controls defects occurring during the development are reported to be mainly of two causes: failure to satisfy the requirements and improperly structured algorithms [12] [13]. The model-in-the-loop (MIL) development phase where both controls and plant model are simulated together is therefore ideal to ensure that all the requirements testing can be accomplished within, which should also be the first and foremost requirement for system model architecture.

The second requirement that needs to be considered determining the model architecture is the seamless transition capability moving from one development phase into another in a coherent manner maintaining all the interfaces and system and environment parameters. This is because one of the biggest advantages of the model based development is the capability to test every step of the way to ensure error free development satisfying the requirements. The development initially starts with the model-in-the-loop (MIL) and later moves into the hardware-in-the-loop (HIL) and finally moves into vehicle-and-driver-in-the-loop (VDIL), all of which are referred here as “x” in-the-loop (xIL). Sometimes it is tempting to skip some of those steps, however unless there is no risk, this sequence should be followed to ensure successful implementation. Having an architecture that provides seamless and coherent transition from MIL through VDIL saves time by eliminating all potential problems during integration of components and subsystems that is repeated for each xIL phase. Since these model components have to integrate seamlessly, their architecture is designed to address all the I/O and parameters requirements at all phases.

The last but not the least requirement is avoiding redundant modeling effort. Having common model architecture, the users can leverage previous information and avoid the proliferation of models that all serve the same purpose. In addition to redundancies in modeling effort, even more is spent on testing; due to the nature of the model based development there are series of tests executed for verification and validation repeatedly. Maintaining the same model architecture provides the edge for the developers to reuse the

same verification tests again instead of duplicating the effort for every development phase. These common tasks across the development phases also support and simplify the automation that can lead to significantly reduced turnaround times.

Implementation

Initial step of model based development is the MIL which includes both the controls and the plant to be controlled (Figure 1). This structure satisfies the first point made above since it contains all the components of system. For ease of understanding the principle approach, initially a system with only one controls module and one plant model is used as shown in the figure. The controller block is labeled “00 CONTROLS”, and the plant block is labeled “01 PLANT”.

The signal bus Sim contains all the signals exchanged between the controls and the rest of the system. Usually the controls output the commanded drive signals and receive the feedback signals from the sensors located in the plant. There is also signal exchange between the controls and the rest of the controllers that are emulated inside the plant including the driver.

The model architecture here is built on the premise that once this system model performs satisfying all the requirements, the 00 CONTROLS portion drops in and executes on target controller hardware and 01 PLANT block drops in and executes on the HIL PC without any changes to the blocks. That is the prerequisite to satisfy the second and third model architecture requirements mentioned above.

The PLANT2CONTROLS and CONTROLS2PLANT blocks are target dependent I/O models for which the embedded code has to be generated according to their corresponding target hardware. Since the plant and the controls software are embedded in separate targets, the I/O blocks are labeled separately for the plant and the controls in the figure. The rate transition blocks are placed grouping the models that have to execute at the same rate.

The way the models are configured for real time applications are displayed in Figures 2 and 3. The controls and the plant blocks are dropped into these models without any changes as planned before embedded code generation.

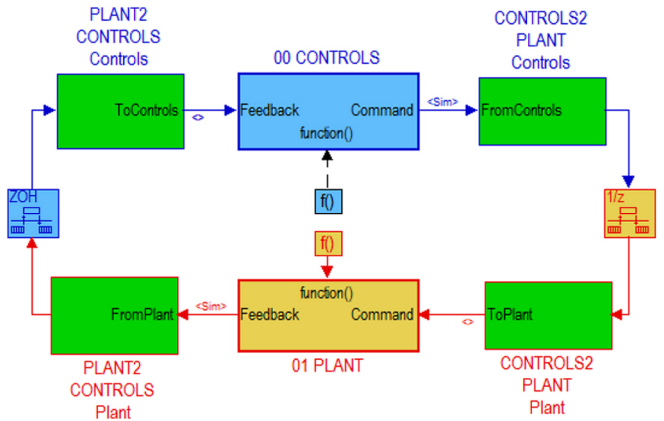


Figure 1. Top level model architecture in MIL phase



Figure 2. The model that executes on the target Controller during HIL and VDIL phases

The controls model should remain the same through HIL and VDIL since that is the purpose of the HIL, the capability to test the controls as they would execute in actual vehicle environment.

The only part of the model that changes as the development makes transitions across the xIL phases is the content of the target dependent interface blocks; even though their interface to the core part of the model (Sim) remain the same, as the transition is made from MIL to HIL, their contents are replaced with actual driver software which can be embedded into target without hand coding. In this particular example in Figure 2 the target hardware is the dSpace module dS1401 and in Figure 3 the target is the dSpace module dS1006

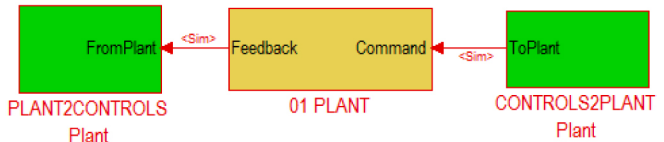


Figure 3. The model that executes on the HIL target PC

Execution Rates

To realistically simulate a complete system, the execution rates have to be set as shown in Figure 1 for the controls and the plant. The plant and controls models are executed separately at their corresponding rates. The controls execution rate is set to satisfy the controller algorithm requirements. Plant execution rate on the other hand depends on what dynamics are desired to be simulated. It is recommended to execute a model at about ten times faster than the simulated dynamics behavior. It is also expected to have a number of subsystems inside controls and plant models, running at different execution rates, which are set up internally in a similar manner since available tools provide flexibility to execute different subsystems at desired execution rates to make the best of out the available computational capability.

Signals

The models represent mathematical representation of systems. For example for a controls model the feedback is usually from a sensor that has a physical quantity such as speed or pressure, associated with an engineering unit such as RPM or BAR. It is a common practice to build the conceptual models at MIL phase using only engineering units both for the controls and the plant models. However the proposed

model architecture in this paper recommends having the signal transition information to be exclusively modeled at MIL phase to help identify potential issues as early as possible. That is why both target independent and dependent subsystems are modeled as early as MIL phase. A number of signal examples are provided below.

Analog Signals

Figure 4 demonstrates analog signal flow between the plant and controls models thru hardware while conditioned through the signal conditioning module during transmission. The source of the signal from the plant model is in engineering units, and the destination of the unit in the controls also is in engineering unit; if it was not for this particular model architecture, there would not be any of the components seen in the figure since the destination in the controls could automatically be hooked to the source of the signal at MIL phase, which has been the common approach. However adding all the transition including sensor dynamics, the signal conversions and conditionings the model is empowered with all the calibration settings and sensor specifications for the interface. Any issues in regards to sensor selection or scaling or calibration can be identified before testing. Also the controls and plant models become ready to be embedded into the controls and HIL target hardware.

The sensor models in the plant model receive physical values with engineering units and output electrical voltage values. Then the voltage values have to be scaled according to the DAC card converting the voltage to a percentage of the full scale value that is labeled as Sim. The components located inside the PLANT2CONTROLS blocks consist of target dependent components from both target hardware where plant and control model are embedded. Since this is only an MIL model, everything is simulated hence the PLANT2CONTROLS blocks represents an analog signal going thru DAC module on the plant target side, going thru a signal conditioning unit, which is an external hardware modeled in between, and finally going into the ADC module on the controls target side. Into the controls, the Sim value is the raw signal, where value 1 represents full scale voltage into ADC unit, goes thru inverse models of ADC unit, signal conditioning unit and sensor model to become the value that it started, such as a pressure value or a displacement value.

All analog signals sent from the controller to the plant are implemented in a similar manner to what is shown in Figure 4 using the CONTROLS2PLANT blocks after the controls and before the plant.

In Figure 5 both forward and reverse models of the signal conditioning unit are shown. The model is built strictly following the subsystem boundaries and the associated signal labels and conditioning while the calibration values match the way they are set up using the Configuration Desk¹.

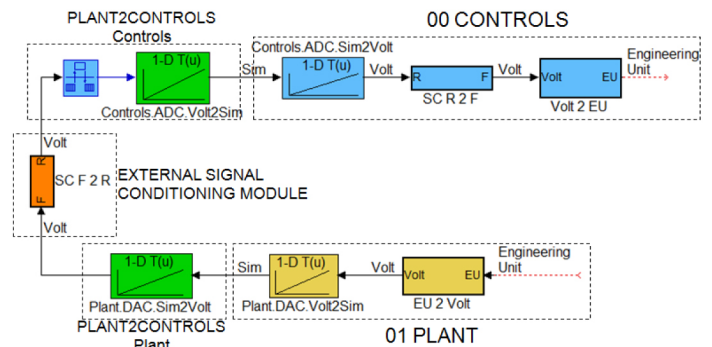


Figure 4. Analog feedback from the plant to the controller in MIL model

Having actual pin numbers as signal names greatly simplifies the layout generation and testing using Control Desk¹ later in HIL and VDIL phases. Especially labeling all the signals in the hardware interface helps to easily trace the signals in experiment layouts and verify their calculated values against the actual measurements.

The implementation for HIL set up for analog signals are shown in Figure 6. The controls and the plant blocks represent the embedded code generated from the same controls and plant blocks shown in Figures 1, 2, 3. The target dependent code is clearly separated from the target independent code from controls and plant blocks. The I/O modules for both target hardware along with the external signal conditioning unit in between are also shown

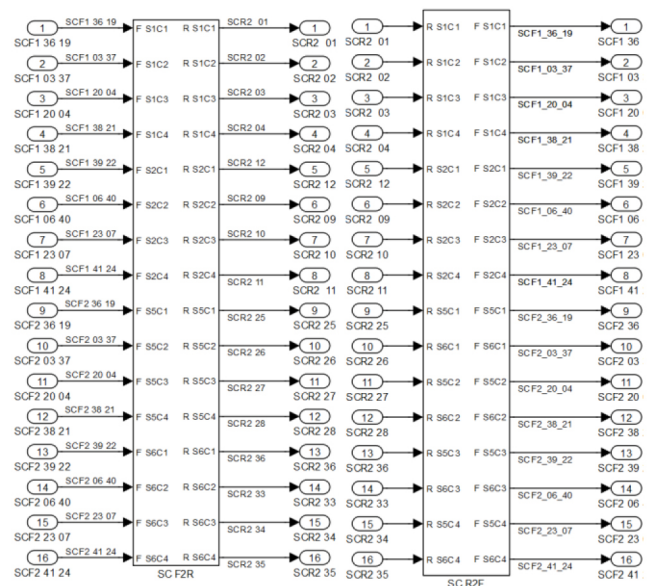


Figure 5. The Rapid Pro¹ Signal Conditioning Unit (RPSCU) modeled front to rear (SC F2R) and rear to front (SC R2F).

¹Rapid Pro, Configuration Desk and Control Desk are the trade marks of dSpace [14]