Consistent Simulation Environment with FMI based Tool Chain

Edo Drenth[1], Mikael Törmänen[2], Krister Johansson[2],
Bengt-Arne Andersson[1],Daniel Andersson[1], Ivar Torstensson[1], Johan Åkesson[1]

[1]Modelon AB
Ideon Science Park, Lund, Sweden
info@modelon.com

[2]Volvo Car Corporation
Dept. Complete Powertrain
Göteborg, Sweden

Abstract

Systems engineers face the ever increasing chase for reduced time to market, while the systems to develop ever increase in complexity. Software systems design and integration processes have therefor evolved along the well-known V-cycle.

This paper will focus on the software integration for mechatronic systems as they develop fast due to high demands and challenging requirements in the automotive industry.

The development order of model in the loop (MIL), software in the loop (SIL), processor in the loop (PIL) and hardware in the loop (HIL) can be seen as state of the art practised by many systems engineers. Driver in the loop (DIL) may be in its infancy, but rapidly growing.

The novelty presented in this paper is the consistency of the plant models used in the integration chain supporting consistent model data propagation: Functional Mock-up Units (FMU) defined by the open standard of the Functional Mock-up Interface[1] (FMI).

Keywords: FMI, FMU, MIL, SIL, PIL, HIL, plant models, Modelica

1 Background

Volvo develops and calibrates its own engine control software. The model based design (MBD) process has been deployed for many years. The legislation on exhaust emissions and fuel consumption has become significantly stricter the past years. The change in legislation increases the burden of developing control software, calibration of parameters and validation of the mechatronic system. As a result efficiency improvements to the MBD process are required.

Experiences tell that the average development and project engineer does not feel comfortable with all aspects of MBD. It might simply be out of their comfort zone. Part of the project assignment was to bring MBD to the test and calibration engineer instead. These engineers shall be able to work with their de facto industry standard measurement, calibration and diagnostic (MCD) tools. The aim is to have a transparency for the software calibration tools as depicted in Figure 6.

2 Introduction

The FMI technology has been adapted fast by many modelling and simulation software vendors. This rapid adaptation of this open standard clearly is proof of an industry demand for (plant) model exchange. In the past the chain from MIL to HIL has been bridged by

many hours of manual labour to mix-and-match many, often for reasons of IP black-box, models from different sources and developed on different platforms. This was tedious work and error prone.

In the ideal case the plant model follows the entire integration process without any model modification. With the introduction of the FMI toolbox for MATLAB® Coder (FMIT-Coder) this vision is achieved.

Yet, the entire development chain from model in the loop to hardware in the loop can make use of one and the same source for a plant model exported as a Functional Mock-up Unit, FMU.

The main contributions of this paper are the use of consistent models throughout the integration workflow from desktop to test bed in the engine controller development, which by the introduction of the FMIT Coder has been made available. It is a solution to Volvo's mission to bring simulations to the test engineer. The engine test and calibration engineer can with help of industry standard protocols and tools, like the INCA[9] based product suite, do his/her calibration work against models or hardware.

The paper is outlined as follows. An introduction to the FMI technology is briefly drawn up, with emphasis on the toolbox available in MATLAB®. The engine and vehicle plant model based on commercially available Modelica libraries are introduced. This is to follow of a more detailed integration flow discussion with help of consistent use of FMU based plant models.

3 The Functional Mock-up Interface

3.1 Introduction[8]

The FMI is a tool independent standard to support both model exchange (ME) and co-simulation (CS) of dynamic models using a combination of xml-files and compiled code. The first version was published in 2010. The FMI development was initiated by Daimler AG with the goal to improve the exchange of simulation models between suppliers and OEMs. FMI is supported by many CAE tools and is used by automotive and non-automotive organizations throughout Europe, Asia and North America.

3.2 FMI Toolbox for MATLAB

FMI Toolbox for MATLAB enables users to import FMUs into Simulink® models by means of a block-set supporting FMI for Model Exchange 1.0[6] and FMI for Co-simulation 1.0[7]. The FMU blocks can then be connected to native Simulink blocks, e.g., to support development of control systems and MIL scenarios. The FMU blocks offer a graphical user interface to parameterize the FMU, set initial conditions, configure outputs and to set the FMU log level, see Figure 1.

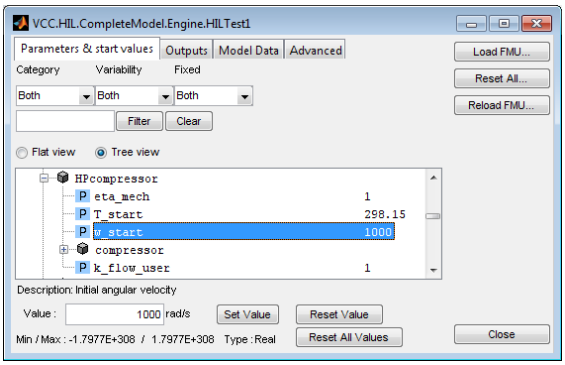

Figure 1 FMIT Dialog

3.3 The FMI ME Calling Sequence

The option for co-simulation is a self-contained sampled system, but the model exchange alternative requires some tight interfacing with the master solver.

The FMI standard defines a calling sequence for simulating the FMU, see Figure 2. The FMI functions are executed in the appropriate order from the FMU block in the Simulink model. The FMU block is based upon an S-Function block which defines a list of call back functions. When the simulation loop is started, the S-function's call back functions are called which then calls the FMI functions.

DOI 10.3384/ECP140961277

3.4 FMIT Coder for MATLAB

FMIT Coder, which is an extension to FMIT, supports export of FMUs from Simulink (plant) models. Export of FMUs according to FMI for Model Exchange 1.0 and FMI for Co-simulation 1.0 (using the fixed-step solvers available in Simulink) is supported. This feature, which relies on Simulink Coder™, enables easy integration of Simulink models in FMI compliant tools.

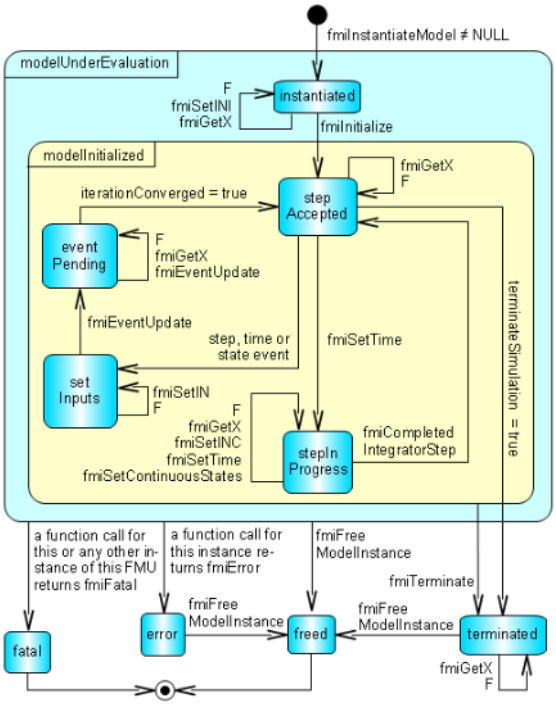

Figure 2 FMI for Model Exchange 1.0 state machine calling sequence.

FMIT Coder also supports code generation from Simulink models containing FMU blocks, possibly from other sources than Dymola, by means of Simulink Coder. Currently only source code FMUs are supported. Supported targets include besides S-functions one real-time platform. The latter target enables HIL simulation, where a source code FMU is connected to native Simulink blocks and the aggregated Simulink model is compiled into binaries that are executed on a real-time computer. This solution provides a tool independent approach to FMI-based HIL simulation.

A target definition in Simulink Coder controls the code generation and can invoke function calls into the build process at the different stages of building. An S-function block may also hook into the build process that affects the code generation. By defining the S-function callback function mdlRTW, the S-function has to define a TLC-file and may also write data to the *.rtw file. The Target Language Compiler can then use this data to generate code for the FMU block. To control the compilation and linkage of the FMU block, the rtwmakecfg.m can be used. The rtwmakecfg returns a predefined structure with libraries and source code to use in the build process.

The FMIT GUI allows the user to edit output signals from the FMUs and make these accessible on the real-time target together with the other Simulink signals. This allows debugging for FMU internal signals on the HIL platform too.

4 Plant model

To support the different steps in the systems integration process, the plant model is ideally configured with *multi-fidelity*[4] configurability in mind.

As an example the engine model, shown in Figure 3, is implemented using the Dymola® Engine Dynamics Library®. The model accounts for 1D gas dynamics, lumped thermal masses and inertia of the turbo machinery. Thanks to the flexibility of the Modelica language, relevant modeling assumptions can be modified by setting model parameters or switching the gas property model. The user can for example, within the same model, choose to disregard the thermal dynamics of metal masses and the gas, disable generation of events at flow reversal and adjust time constants of gas dynamics. From a parameterized plant model with detailed dynamic representation, by setting flags related to physical modeling assumptions, a simplified model well suited for fixed step solvers and HIL can be obtained.

For defining the engine load a vehicle model created with help of the Vehicle Dynamics Library® is deployed. The deployment of the vehicle model can either be in the same FMU as the engine model, because both used libraries are supported in Dymola, but may be two separate FMU's. The latter supports a transparent and consistent use of models all the way from desk top to an engine test bed.

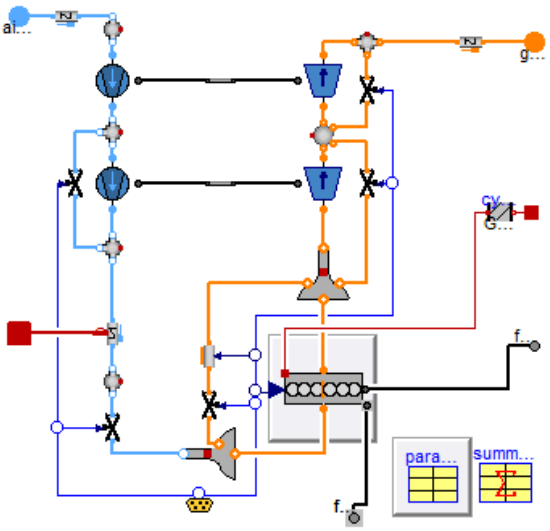

Figure 3 Engine model diagram layer

5 Systems integration process

5.1 Introduction

For completeness many integration steps are outlined and discussed below, but a deployed development process does not necessarily include all presented steps in this document.

5.2 Model in the loop

The accessibility and quality of code generators has improved tremendously during the nineties. This has been an enabler for development of controllers in a simulation environment.

Developing control software in a simulation environment, like Simulink® in Figure 4, allows algorithm testing in a very early stage of development. The design iterations can be much faster, because the simulation environment allows the control engineer to quickly change algorithms and instantly simulate and thus test.

Because the plant model is supplied as an FMU and run with help of the FMI Toolbox for MATLAB[5] (FMIT), the development engineer that actually masters the domain of the hardware can produce a plant model in his favourite and ideal modelling environment. In this case Dymola® Engine Dynamics Library®.

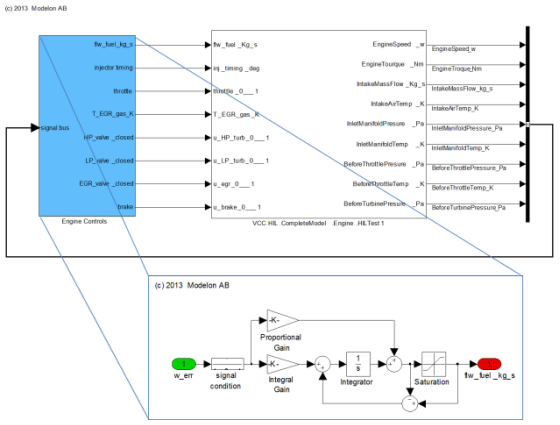

Figure 4 A MIL (blue block) example of a simple engine controller and an EDL based engine model exported as an FMU (white block)

5.3 Software in the loop

Once the algorithms in the MIL stage perform as designed the controller model can be transformed into c-code with a coder like Simulink Coder or TargetLink® including debug information. This exported software code can in its turn be linked with external sources of code or manual written code if desired.

The created code can firstly be tested in Simulink as shown in Figure 5. This solution mimics the MIL solution very closely.

The created code can also be hooked up to communicate with other systems and sub-systems in for instance Silver™ by QTronic[3]. Silver supports virtual module integration and test automation. An important benefit for Volvo using Silver is the support it has for the already deployed engine calibration tools and protocols. The calibration engineer won't see the difference if s/he is calibrating a virtual or a real engine.

DOI 10.3384/ECP140961277

The great benefit of SIL is the possibility to debug code intended for implementation in a mechatronic system. Of course, code can be debugged on HIL rigs too, but with SIL the algorithms can be tested by means of virtually holding time in contrast to HIL based testing where time marches on and the external signals continue to be updated.

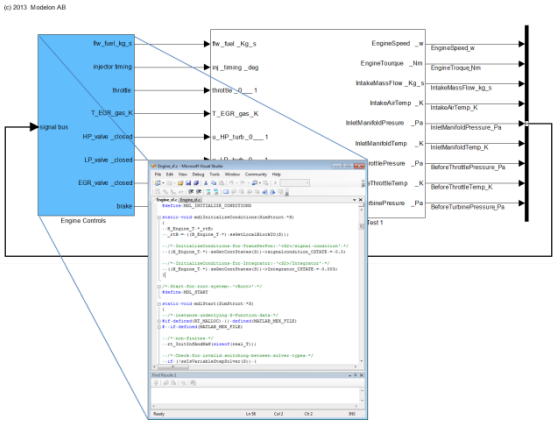

Figure 5 A SIL (blue block) example of a simple engine controller and an EDL based engine model exported as an FMU (white block)

All SIL testing can be performed against the same plant model FMU as in MIL.

5.4 Processor in the loop

The next integration step would be processor in the loop. Processor in the loop may be seen as a half-way house for HIL. The actual target processor is used. High speed IO is directly connected to the simulation environment and the real sensors and actuators are omitted. PIL enables the developer to analyse stack memory and CPU load analyses for instance.

PIL requires a plant model which runs in real-time and from an FMI perspective would have the same requirements as the HIL solution (see section below).

In order to allow high fidelity plant models, Virtual PIL[2] can be deployed because there is no real-time constraint in this configuration. This solution creates the ability of target code debugging on algorithm level against high fidelity signals because time can be stepped through, in contrast to PIL and HIL where time marches on after a break point in the code.

5.5 Hardware in the loop

First, if not foregone by PIL, when the actual electronic controller is to be tested on a HIL rig with its actual sensors and actuators (these are possibly emulated), a demand for real time capable FMU's arises. At the same time there exists a wide variety of RT platforms and hence the plant model FMU needs to be exported as source code. Many of FMU export compliant tools will be able to produce source code, but may need a special license module.

Access to source code allows cross compiling to RT systems for HIL simulations.

5.6 Engine calibration support

Test and calibration engineers use typical automotive measurement, calibration and diagnostic applications that communicate with embedded targets with standardised protocols. Silver connects with INCA and thus all alternatives from SIL onwards to test bed (and in vehicle actually) can use the same calibration and measurement tool. Please refer to Figure 6 to get an overview of the process.

With the deployment of FMUs across all integration levels, Volvo is able to offer its test and calibration engineers one and the same interface independent of the integration phase. The engineer won't necessarily know if s/he is calibrating virtual or in real life.

A high fidelity engine plant model consisting of a data-driven combustion model and first principles air charge model will support virtual engine calibration at SIL/VPIL level and reduce the number of test beds (and vehicle prototypes for that matter) necessary for calibration.

5.7 Test cell support

The above discussed engine plant model in the MIL to HIL chain can be replaced by a physical engine in a test bed. The loads in the test bed are determined virtually with the FMU-Vehicle representing the drive line and road loads on the vehicle. This test cell support enables calibration of engine controllers with respect to different driving cycles. One benefit

of using virtual vehicle models is that the environmental conditions are more controllable than in real vehicles tests.

The consistent usage of plant models across the different experimental domains allows propagation of parameters for the physical models and data for the empirical models for the plant models at different fidelity level all the way back to MIL.

(CSW) is interacting with the physical plant models.

Simulink Coder or TargetLink allow the CSW to be exported to Silver. This solution allows for SIL and VPIL. Silver supports standard CAN protocols and thus standard already available measurement, calibration and diagnostic (MCD) software can be used.

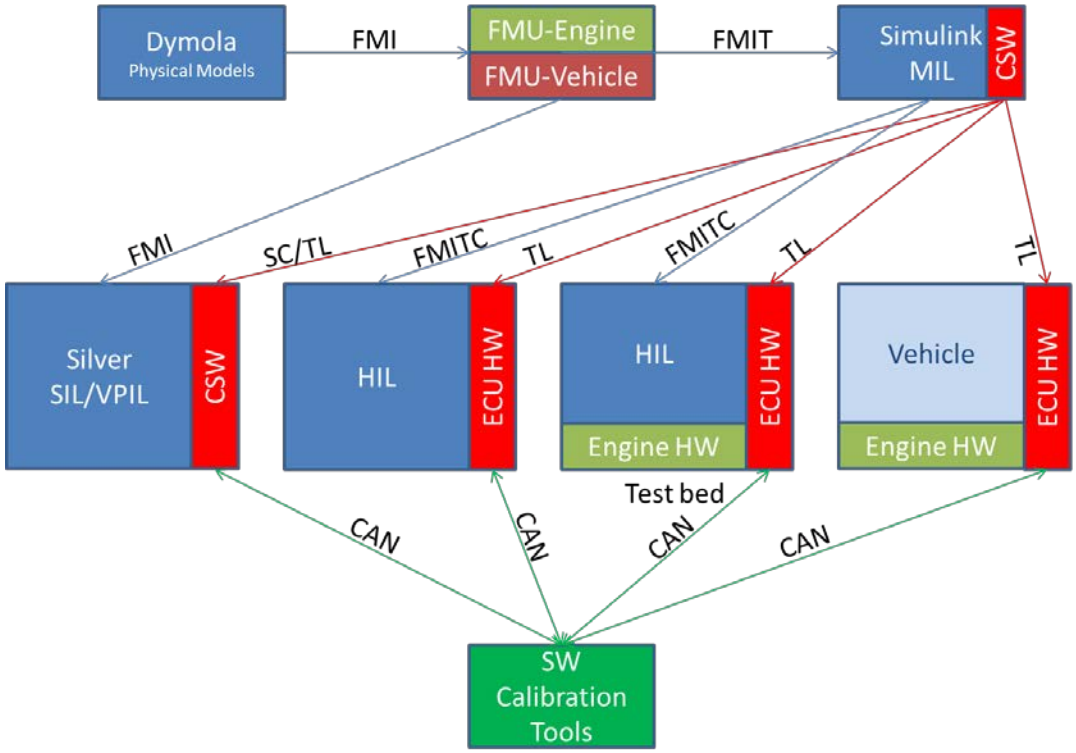

Figure 6 FMU's central role in control system integration process. For sake of simplicity the PIL solutions are omitted. TL=TargetLink, SC=Simulink Coder, CSW=Control Software.

Data retrieved from tests will be part of data- and model regression and the process is thus improving quality of results over time.

6 Process summary

The process summary is depicted in Figure 6. Dymola is the environment where the plant models for the physical systems, engine and vehicle in this case, are created. These models are exported as FMU's. On binary level these FMUs can with help of FMIT be used in Simulink and can also be natively imported to Silver. In Simulink the control software model

The next logical step is the export of plant models to the HIL environment with FMIT Coder. The model fidelity might be lower to account for the real time constraints this solution has. The CSW is in the real ECU hardware exported with TargetLink. Of course the standard MCD software can be invoked to alter the ECU calibration.

Currently the final stage will be the engine test bed that has its loads determined with an FMU for the vehicle with driveline. For the MCD software this is exactly the same use case as the HIL solution.

DOI 10.3384/ECP140961277

The test bed results can be fed back into the model parameters and data to evolve fidelity and quality of models. An iterative process is created to continuously improve model fidelity and quality.

At the end of the process of course, the engine and engine controllers are assembled in the vehicle. The same MCD-toolset can be used.

7 Conclusions

With the speed FMI technology is embraced in the industry is clear sign it has solved a long existing challenge for systems integration and validation engineers. Often tedious and error prone manual modifications to adopt the supplied models and data to different simulation environments have become a technology of the past with the introduction of FMI compliant FMUs.

With the FMIT-Coder the FMI based chain of FMU deployment is complete. Yet the entire suite of validation and verification development stages is covered by FMI technology.

The consistent use of FMI compliant models has been an enabler for improved work flow efficiency and model quality of the MBD process.

Part of the project assignment was to bring MBD to the test and calibration engineer. These engineers shall be able to work with their de facto industry standard measurement, calibration and diagnostic tools. The aim to have a transparency for the MCD tools is accomplished and depicted in Figure 6.

8 Copyright notice

9 References

1. Blochwitz, T., Otter, M., Arnold, M., Bausch, C., Clauß, C., Elmqvist, H., Junghanns, A., Mauss, J., Monteiro M., Neidhold, T., Neumerkel, D., Olsson, H., Peetz, J.-V., Wolf, S., *The Functional Mockup Interface for Tool independent Exchange of Simulation Models*, 8th Modelica Conference, Dresden, Germany, 2011
2. Liebezeit, Bräuer, Serway, Junghanns: *Virtual ECUs for developing automotive transmission software*, 10th CTI Symposium Innovative Fahrzeug-Getriebe Hybrid- und Elektro-Antriebe, 5.-8.12.2011, Berlin, Germany
3. Junghanns, A., *Virtual integration of Automotive Hard- and Software with Silver*, Qtronic GmbH, Berlin
4. Andreasson, J., Andersson, D., Batteh, J., Gohl, J., Griffin, J., Krueger, I., *Integrated simulation of a e4WD vehicle using Modelica*, Advances in Automotive Control, Volume#7, Part#1, 2013
5. *FMI Toolbox User's Guide 1.7*, Modelon AB, Lund, Sweden, 2013
6. *Functional Mock-up Interface for Model Exchange, Version 1.0*
7. *Functional Mock-up Interface for Co-Simulation, Version 1.0*
8. https://www.fmi-standard.org/
9. *Measurement, ECU Calibration, and Diagnostics –Development Solutions for Automotive Embedded Systems*, ETAS GmbH, Stuttgart, Germany, 2010