FMI Application Guide

Ver. 2.0.2

February 7, 2026

Revision	Date	Main contents
1.0.0	Oct 01, 2018	Ver1.0.0 released
1.0.1	Nov 01, 2018	Chapter 6: Added 6.2.4, corrected a typo in 6.2.5 regarding minimum computation time.
2.0.0	Oct 31,2023	Ver. 2.0.0 released
2.0.1	Jun 1,2024	Ver. 2.0.1 released
2.0.2	Feb 7, 2026	Ver. 2.0.2 released

Table of Contents

The following individuals, companies and organizations have participated and cooperated with the FMI Utilization and Deployment WG of the Automotive Control and Modeling Division Committee of the Society of Automotive Engineers of Japan in the preparation of this guide.

Mutsuhito Eshima	IDAJ Corporation
Junichi Ichihara	AZAPA Corporation
Kazunari Moriya	AZAPA Corporation
Takayuki Sekisue	Ansys Japan K.K.
Takashi Iwagaya	Cybernet Systems, Inc.
Martin Egginton	Siemens AG
Yosuke Ogata	Siemens AG
Yuuichiro Abe	Dassault Systèmes K.K.
VIRY, Guillaume	Dassault Systèmes K.K.
Makoto Koekiba	Chuo Zuken K.K.
Katsuya Tsuzuki	dSPACE Japan Corporation
Takashi Yoshimatsu	dSPACE Japan Corporation
Hiromi Uchida	Dentsu Research Institute Inc.
Keiichi Ueyama	Denso Corporation
Shuya Miwa	Denso Corporation
Dai Araki	Toshiba Digital Solutions Corporation
Yutaka Hirano	HIRANO Research Lab.
Yasufumi Saruki	HEXAGON Corporation
Nobuyuki Hirai	HEXAGON Corporation
Motoaki Muto	Nissan Motor Co.
Haruki Saito	Nissan Motor Co.
Hideaki Jinno	Honda R&D Co.
Shingo Haketa	Honda R&D Co.
Riichi Nagao	PonoSHIP K.K.
Daisuke Akasaka	The MathWorks GK
Akio Takashima	The MathWorks GK
Shun Endo	Mazda Motor Corporation
Ken Komori	Mazda Motor Corporation
Hiroaki Takeuchi	Mazda Motor Corporation
Yuji Sato	Mitsubishi Space Software Corporation
GAO, Rui	Modelon Corporation

(Alphabetical order by Japanese company/organization name)

We would also like to thank the following individuals, companies and organizations for their cooperation in preparing this guide.

Seiji Ishikawa	ETAS Corporation
Kensuke Shibuya	Nagoya University
Koichi Shigematsu	Nagoya University
Yutaka Funabashi	Renesas Electronics Corporation

3V-SG (Study Group for Control Verification Using Virtual Methods)

(Alphabetical order by company/organization name)

Chapter 1 About this Guide

1.1. Purpose of this Guide

This guide is intended for engineers who use simulation in the development of automotive systems to exchange and connect simulation models.

Automotive systems require high functionality and performance, and model-based development utilizing simulation is becoming increasingly important in the development of these complex systems. And in order to verify the behavior of the entire system by combining multiple systems and components, simulation models are required to be distributed among departments within the company or with other companies.

This guide describes the basic techniques and considerations for exchanging and connecting simulation models.

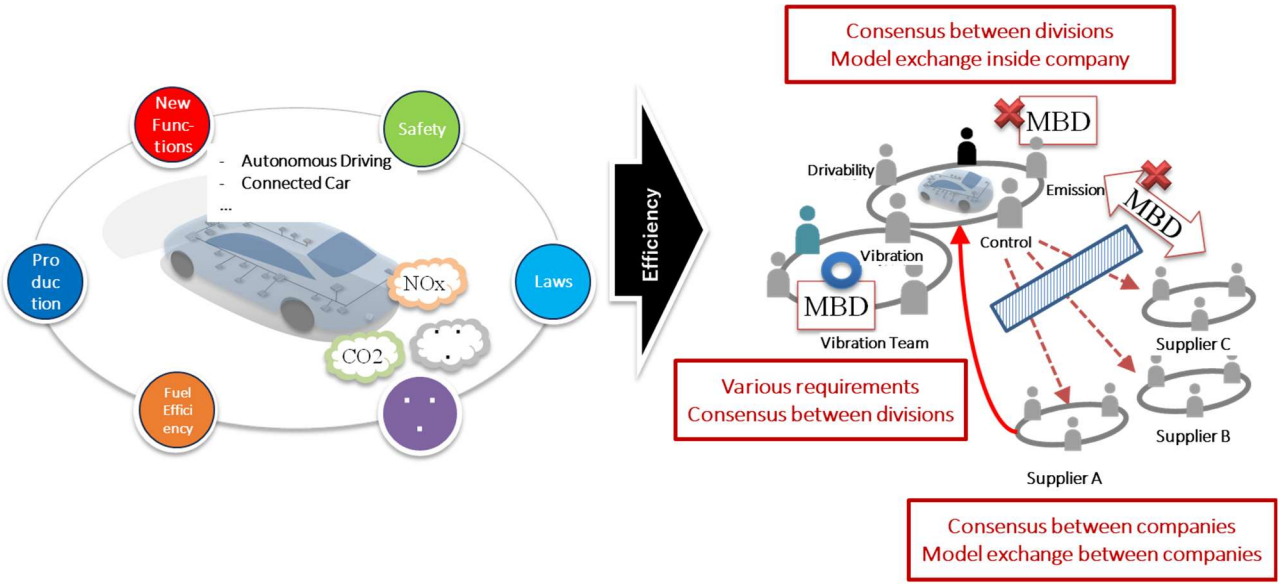


Figure 1.1.1 Figure 1.1 Image of model distribution

(Adapted from Jigikai Symposium on Vibration and Noise (2017), "Toward Model Distribution with FMI") [1].

1.2. Introduction.

In the development of automotive systems, simulation is utilized in all stages of the design/production/testing sequence. As computer performance improves and software advances, simulation is being utilized more and more. The components of a model-based simulation environment are as follows

Table 1.2-1 Components of the Simulation Environment

model	Physical behavior and controller behavior to be simulated Abstracted by ・Graphical Description ・Modeling and programming languages ・Mathematical expressions, tables, maps, etc.
solver	Functions for performing numerical operations such as integration
application software	Simulation tool operating program
operating system	basic computer program
computer	Physical devices executing the program, including network communication environment

1.3. background

1.3.1. Expectations for Model Distribution

Automotive system development has traditionally begun with the design and prototyping of the components that make up the system, followed by testing using a prototype vehicle that combines these components. However, the conventional development style has reached its limits, as it is no longer able to meet the demands for increased complexity and shorter development time due to higher functionality and performance. Therefore, the relationship between specifications and actual parts is being replaced with specifications and corresponding simulation models, and model-based development is changing to testing through simulations using prototype vehicle models that combine sub-models that are the components of the system. And there is a demand for distribution of simulation models among departments within a company or with other companies.

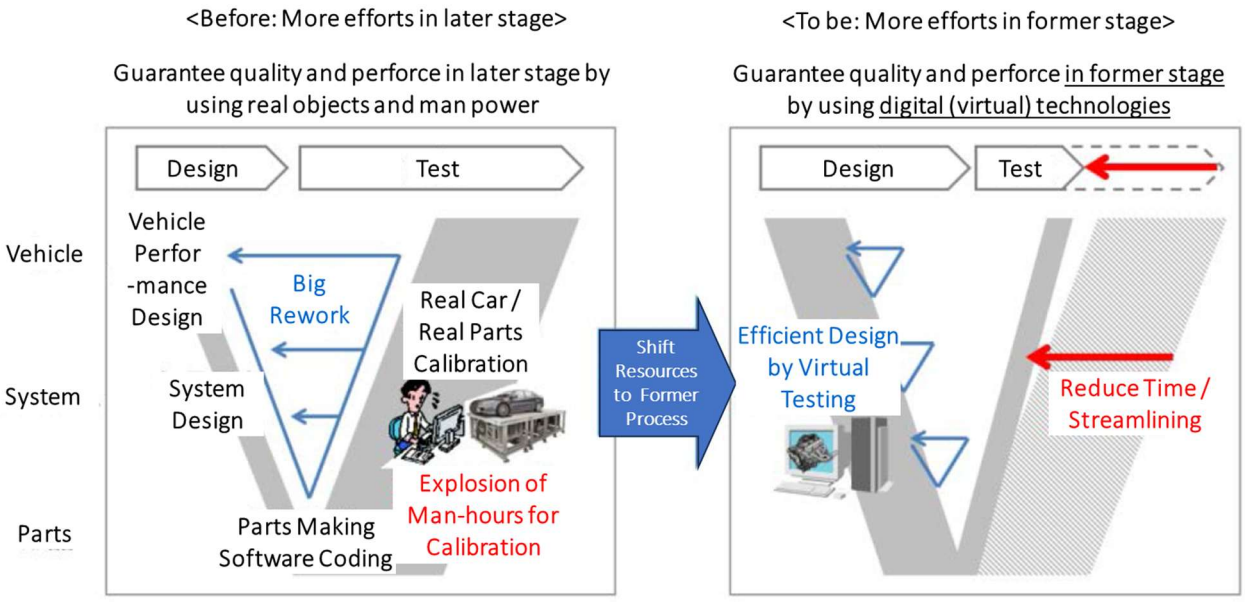


Figure. 1.3.1 Innovations in the way "cars" are made

(Quoted from the Ministry of Economy, Trade and Industry's Automotive New Era Strategy Council (1st meeting) handout) [2].

1.3.2. Model Distribution Issues

Since various simulation tools are used to describe simulation models, it is necessary to unify simulation tools or connect models created by different simulation tools in order to exchange models among departments or companies within a company. In particular, since it is not realistic to unify simulation tools across all companies involved in automotive system development, a common interface for exchanging and connecting models between different simulation tools has become a challenge.

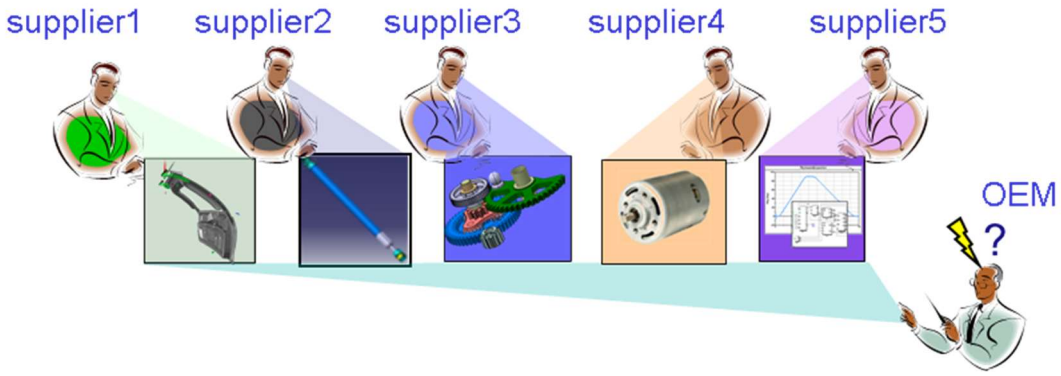


Figure. 1.3.2 Model Distribution Between OEMs and Suppliers

(8th International Modelica Conference 2011 preprints Adapted from) [3].

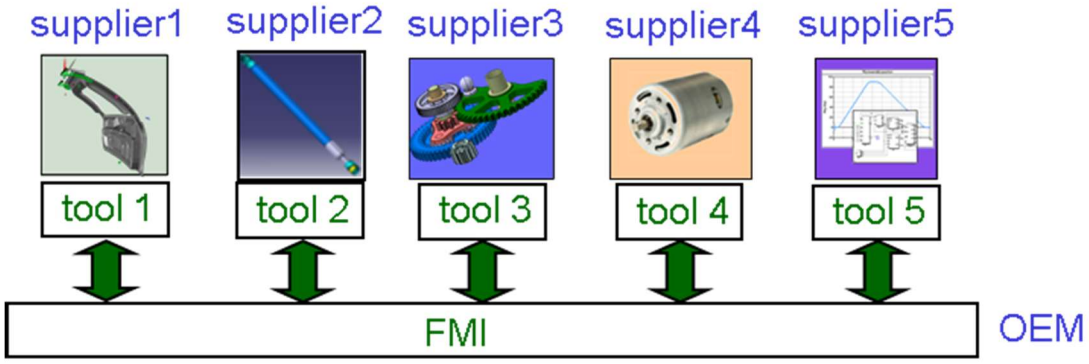


Figure. 1.3.3 Model Distribution between OEMs and Suppliers by FMI

(8th International Modelica Conference 2011 preprints cited from) [3].

1.3.3. Standardization of model distribution

The Functional Mock-up Interface (FMI), standardized by a European public project as a common interface for model connection independent of simulation tools, is gaining popularity worldwide. FMI is also recommended as one of the standard interfaces for model connection and exchange in the guidelines established by ProSTEP iViP Association, a non-profit organization that promotes more efficient system development methods in the European automotive industry. The Society of Automotive Engineers of Japan (JSAE) also established the "Model Development and Distribution Study Committee with International Standard Descriptions" in March 2012, and has been promoting model connection verification, model exchange, and connection using FMI in the Model Connection Technology Study WG. The FMI Utilization and Deployment Study WG of the Automotive Control and Modeling Committee has since taken over this activity and is studying general methods of exchanging and connecting models between different simulation tools without relying on modeling tools. The following objectives are being pursued by the FMI Utilization and Deployment Study WG

① Verification of model connection method by FMI

② Identification of technical issues and study on how to deal with them

③ Development of findings (holding academic lectures and forums, publication of application guides)

④ Cooperation with Modelica Association (a non-profit organization that develops and promotes FMI standards) and related organizations

The results of this work were compiled and published in 2018 in the FMI Application Guide　Ver. 1.0.1 was published and made freely available for download. We have now updated the FMI Utilization Guide as Ver. 2.0 based on the results of updates and extensions to the FMI standard since 2018. It is our sincere hope that this guide will contribute to the promotion of model-based development in the automotive industry through active tool-independent model connection and distribution.

Chapter 2: FMI Basics

This chapter provides basic knowledge and information about FMI.

2.1 explains the origins of the FMI.

2.2 describes the basic structure of FMI's three modes of operation: Model Exchange (ME), Co-Simulation (CS), and Scheduled-Execution (SE). 2.3, 2.4 and 2.5 describe the structure and features of each.

2.6 describes compatibility notes for passing models in FMI format.

In 2.7 section, we will explain some notes on information hiding when passing models in FMI format.

In 2.8 section, we will explain how to find and research tools that support the FMI standard.

2.1. Origin of FMI

Between 2008 and 2011, as part of the EU project Information Technology for European Advancement (ITEA2), a project called MODELISAR was implemented. As a result, the Functional Mock-up Interface (FMI) ver. 1.0 (hereafter referred to as FMI1.0) was developed as a common interface standard for model connectivity that meets the following requirements: FMI1.0 includes Model Exchange (ME) and Co-Simulation (CS) modes of operation, but these two standards were developed separately.

After the completion of the MODELISAR project, the activities of FMI were taken over by the Modelica Association (hereafter referred to as MA), a non-profit organization consisting of business people involved in model-based development, university researchers, tool vendor engineers, etc. In MA, functions that were missing in FMI1.0 were added and ambiguities in the specification were reviewed. FMI for Model Exchange and Co-Simulation Ver. 2.0 (hereafter referred to as FMI2.0) was established in July 2014. Model Exchange (ME) and Co-Simulation (CS) modes of operation were combined into a single standard. Since then, FMI2.0 has undergone minor revisions, and the current version of the FMI2.0 standard is Ver. 2.0.4, published in November 2022.

FMI 1.0 and FMI 2.0 are industry standards for model distribution and are supported by nearly 200 tools to date.

On the other hand, new needs and use cases such as advanced co-simulation algorithms and V-ECU (virtual electronic control unit) packaging and simulation have emerged, and new features to enable these have been considered in MAs, leading to the establishment of the FMI Ver. 3.0 (hereinafter referred to as FMI3.0). In addition to the addition of Scheduled Execution (SE) as the third mode of operation, FMI3.0 has a variety of new features such as expanded data types, clock signal and event driving.

This document also discusses the latest FMI 3.0, but there are still only a few tools that support FMI 3.0, and there are not many opportunities to use FMI 3.0. Therefore, most of the explanations in this document will focus on FMI 2.0.

2.2. FMI operating modes and FMU file structure

2.2.1. FMI operating mode

The FMI standard uses a model part interface called a Functional Mock-up Unit (FMU) to pass models between different tools. (ME), Co-Simulation (CS), and Scheduled Execution (SE).

ME and CS are modes of operation that have existed since FMI 1.0 and FMI 2.0; SE is a new mode of operation introduced in FMI 3.0.

In the case of ME, only the calculation model is exported to FMU and passed on, and when the FMU is imported and executed the solver on the tool side is used. For the CS, both the calculated model and solver are exported as a set to FMU. When passing the model between different tools, FMU is imported and calculated using the solver built in the FMU. This is the major difference between ME and CS.

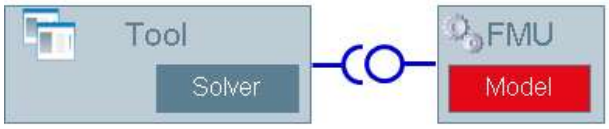


Figure 2.2.1 (a) ME without solver in FMU

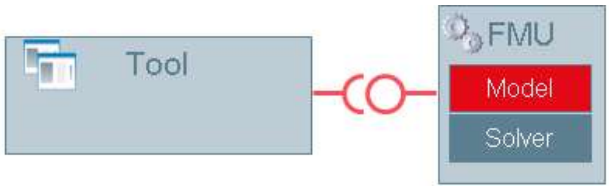


Figure 2.2.1 (b) CS with solver in FMU

(8th International Modelica Conference 2011 Presentation Material Adapted from) [3].

SE is designed to package and simulate V-ECUs (virtual electronic control units) and differs from ME and CS in its structure. The model partition, which corresponds to the task code of the control controller, is passed as a model, and the model is run using the scheduler provided by the tool that imports the model.

2.2.2. FMU File Structure

The extension of FMU is .fmu and its reality is compressed in ZIP format. Concept is shown inFigure 2.2.2 FMU is a combination of an executable dll file for Windows OS or an executable so file for Linux OS and an xml model description file which is a detailed description of the model. As additional description of the executable module, In case of hiding the source file, the tool that generates the FMU deletes the source file after generating the dll to enable concealment. This setting method depends on the tool, so please refer to the manual of the tool you are using.

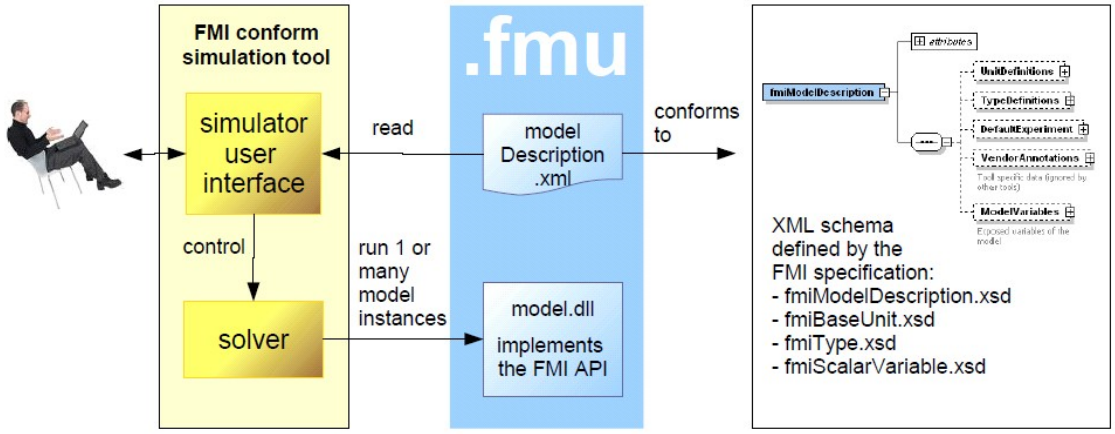


Figure 2.2.2 Structure of FMU

(8th International Modelica Conference 2011 preprints cited from) [5].

Figure 2.2.3 shows the structure of the model description file. As mentioned earlier, in FMI 1.0, ME and CS are planned and formulated separately from the planning stage, and have no common structure. Thus, the structure of FMI 2.0 will be described hereinafter.

Note that compatibility is not maintained between FMI 1.0 and 2.0, so care must be taken when handling them.

At the beginning, there is a description of the distinction between ME or CS, followed by a detailed description of implementation, units, variables, attributes, structures and their attributes, etc.

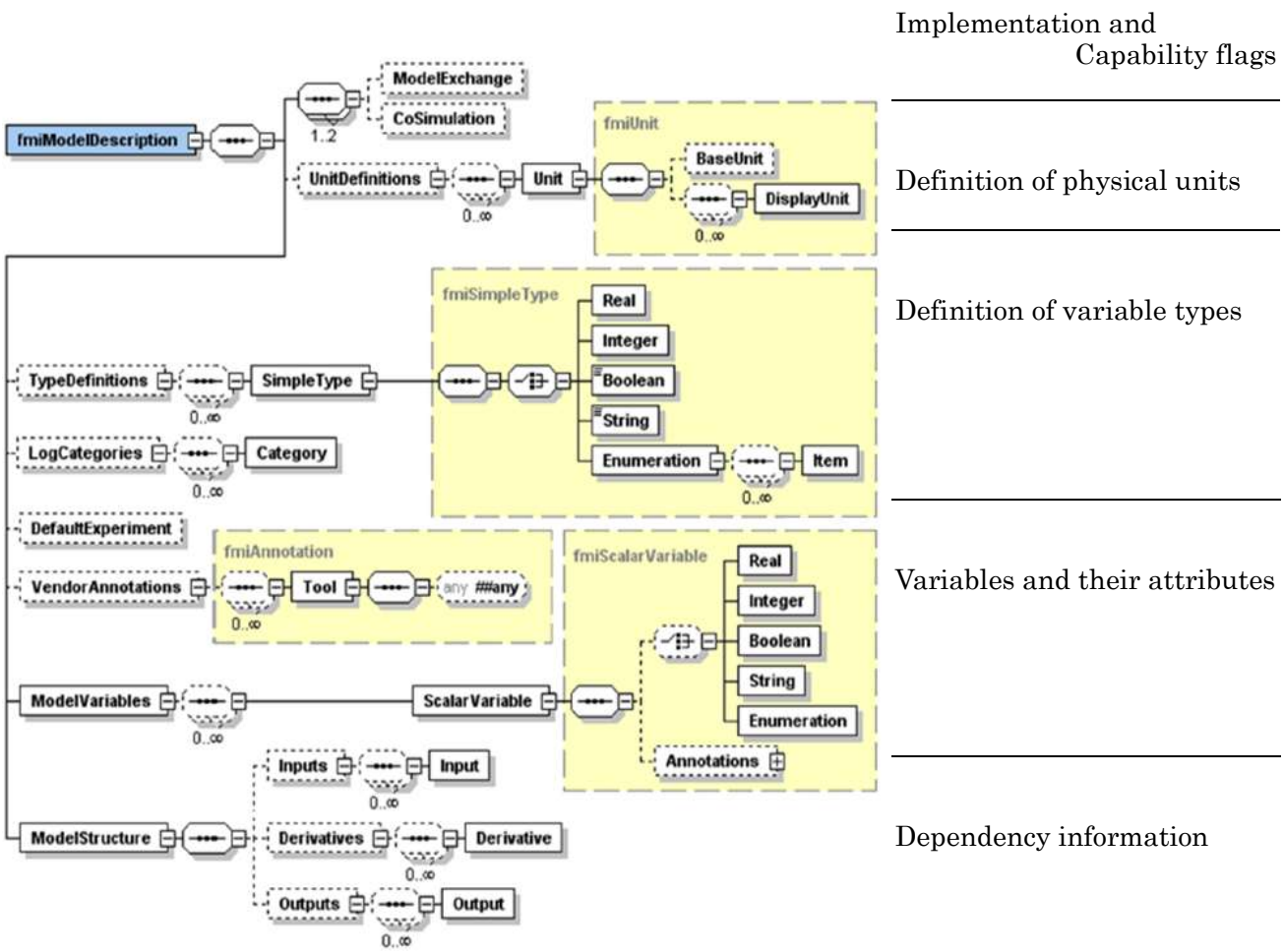


Figure 2.2.3 Structure of the model Description file

(Japanese only) 2nd Japanese Modelica Conference Keynote Speech) [6].

2.3. Model Exchange (ME) Structure and Features

2.3.1. Signal flow in ME model

In ME mode, calculations in the FMU are performed using the solvers in the simulation tool that incorporates the FMU. Thus, time synchronization is guaranteed since the time transitions of the FMU and the surrounding model are performed simultaneously by the same solver.

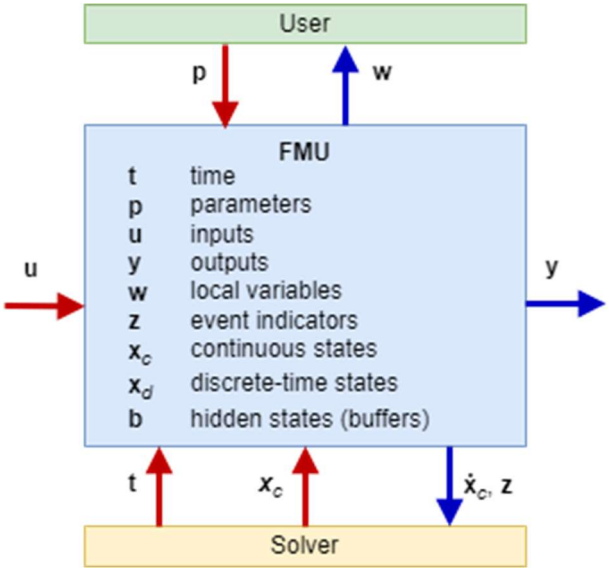


Figure 2.3.1 Signal flow in ME mode

(quoted from FMI Specification: Functional Mock-up Interface Specification Version 3.0 [7])

2.3.2. Example of model connection in ME

Figure 2.3.2 (a) shows an example of an FMU and a signal flow that has been imported into the non-causal system tool and connected via Model Exchange. The model consists of two FMUs.Figure 2.3.2 (b) shows the original model before it is made into an FMU. The red dashed lines correspond to the respective FMUs in Figure 2.3.2(a).Figure 2.3.2 (c) shows a conceptual diagram of the structure. As shown in Figure 2.3.1, the simulation is performed by a solver on the imported tool.

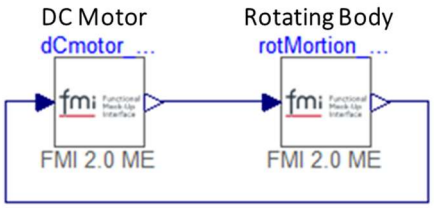


Figure 2.3.2 (a) Example of connection in Model Exchange mode (FMU)

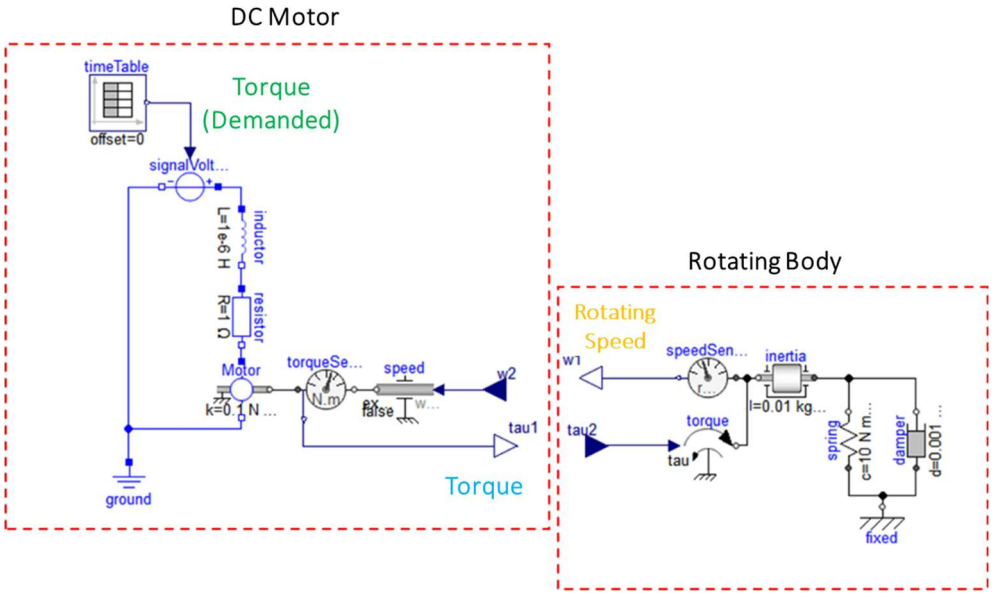


Figure 2.3.2 (b) Example of connection in Model Exchange mode (original model)

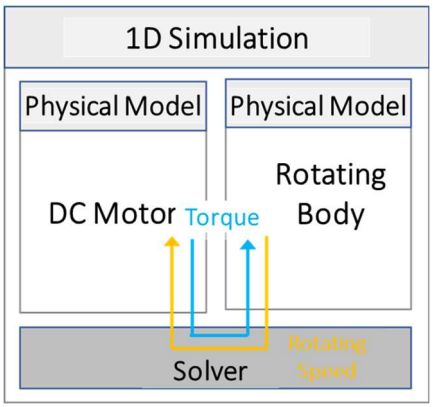


Figure 2.3.2 (c) ME model (Figure 2.3.2 (b)) Supplement

2.4. Co-Simulation (CS) Configuration and Features

2.4.1. Three forms of CS

CSs are classified into three forms depending on their operation.

The most basic is Stand Alone (Figure 2.4.1), where the master tool and FMU run on a single core on the PC.

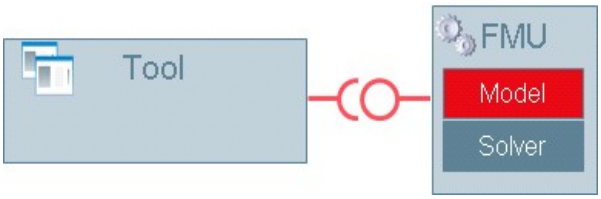


Figure 2.4.1 Stand Alone Co-Simulation

(Cited from 8th International Modelica Conference 2011 Presentation Material) [3].

The second form is Tool Coupling (Figure 2.4.2), where the master tool and FMU run on separate processors or cores. For more information on the FMI Wrapper, please refer to the Functional Mock-up Interface Specification Version 3.0 [7].

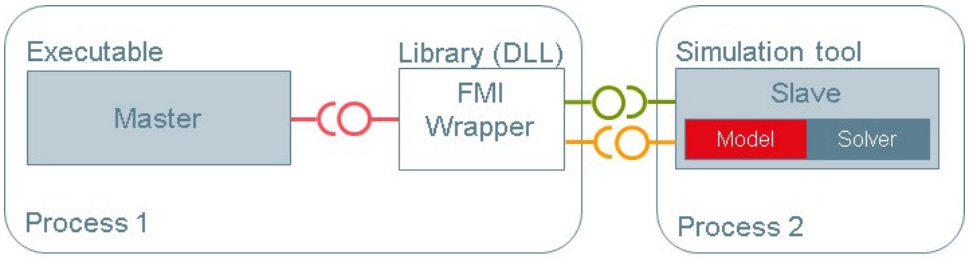


Figure 2.4.2 Tool Coupling Co-Simulation

(Cited from 8th International Modelica Conference 2011 Presentation Material) [3].

The third form is Distributed (Figure 2.4.3), which is applied in large systems that require enormous computer resources. It is based on the basic distributed environment of the client-server method and is configured in compliance with the FMI standard.

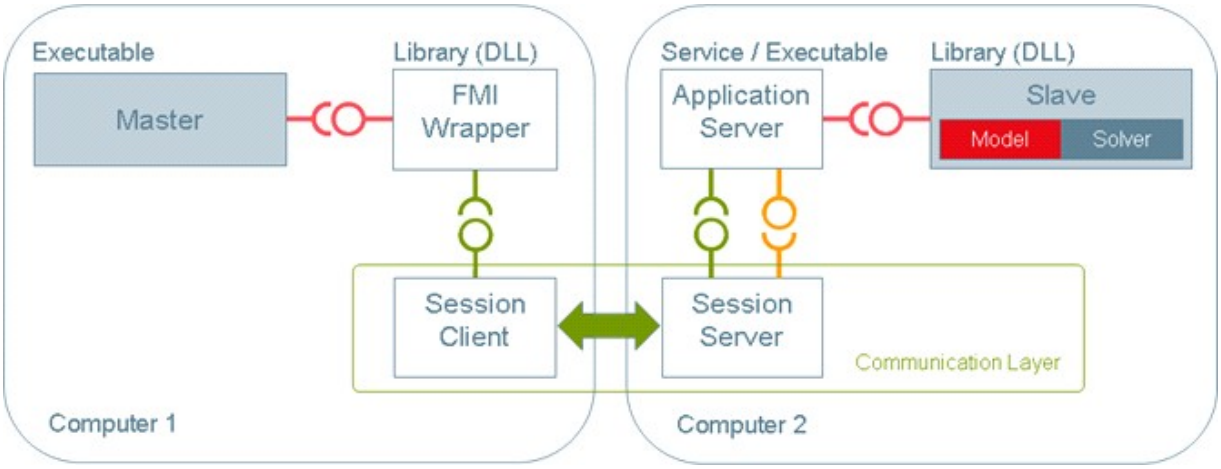


Figure 2.4.3 Distributed Co-Simulation

(Cited from 8th International Modelica Conference 2011 Presentation Material) [3].

2.4.2. Signal flow in CS

The signal flow in the CS is shown in Figure 2.4.4. (u) and (y) are the inputs and outputs of the FMU, and are calculated by the solver inside the FMU. The input (u) and output (y) are exchanged with the tool that has imported the FMU at the time set by the Communication Step Size (see below in Chapter 3).

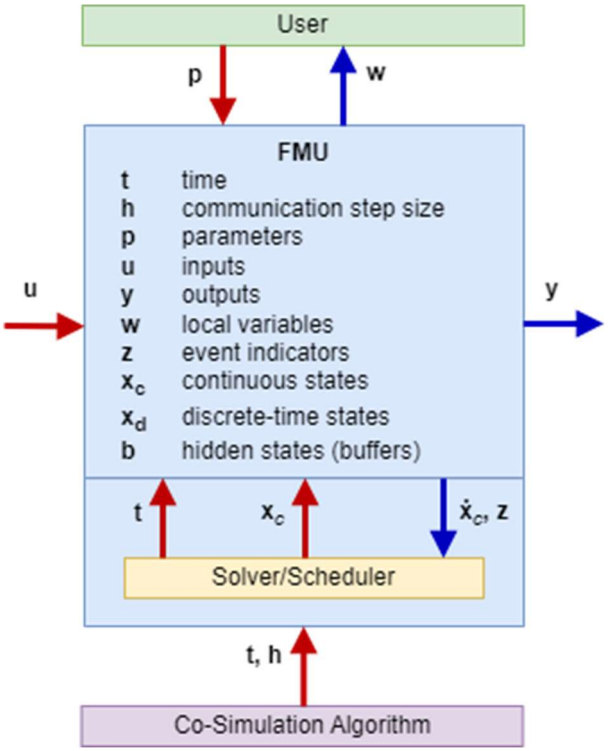


Figure 2.4.4 Signal flow in CS mode

(Quoted from FMI Specification: Functional Mock-up Interface Specification_Version 3.0)[7]

2.4.3. Example of model connection in CS

Figure. 2.4.5 (a) shows an example of FMUs and signal flows connected by Co-Simulation imported into the non-causal system tool. The model consists of three FMUs.Figure. 2.4.5 (b) shows the model before the FMUs, where the red dashed lines correspond to the respective FMUs in Figure 2.4.5(a). In this figure, it appears that each signal is directly exchanged between FMUs, but in realityFigure. 2.4.5 (c), the signals are exchanged through the solver of capturing tool. Each FMU has its own solver, performs its own calculations, and then transfers signals through this solver of the host tool at the timing of each Communication Step Size to perform the overall simulation.

Communication parameters such as Communication Step Size can be set for each FMU by speciifying the properties of each FMU imported to the simulation tool.

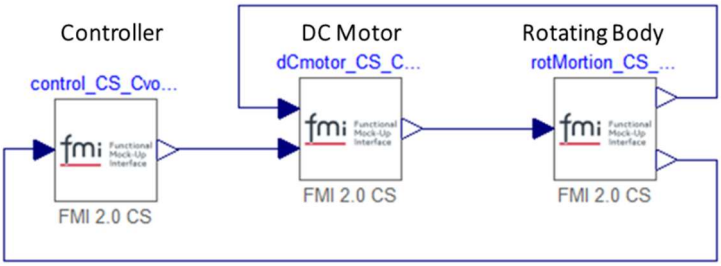


Figure. 2.4.5 (a) Co-Simulation mode (FMU)

Specifically signals to and from the DC motor in Figure. 2.4.5 (a) are sent and received as shown in Figure. 2.4.5_(b). The indicated value of torque from the controller to the power supply is sent to the DC motor via the host tool of the FMU. The DC motor produces torque upon receiving the indicated value, and the revolution speed and angle from the rotor are calculated from this torque. These signals are also fed back to the controller via the host tool solver, respectively.

This signal communication is performed at every time interval set by the Communication Step Size (CSS). Since the torque indication value from the controller is passed through the host tool solver, the DC motor receives the value from the host tool solver at the timing of the next CSS. This is expressed on the time axis asFigure. 2.4.5 (d).

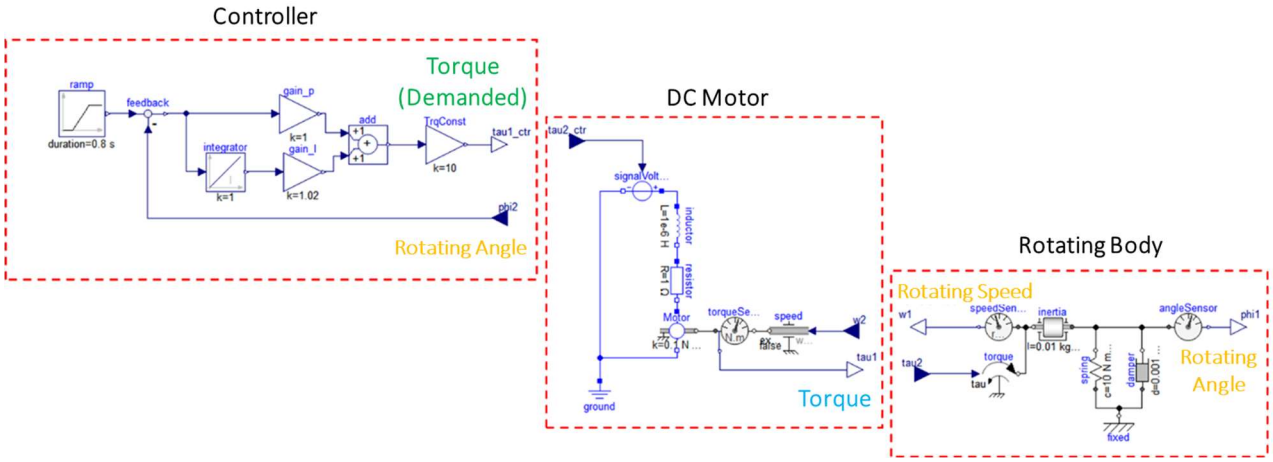


Figure. 2.4.5 (b) Connection example in Co-Simulation mode (original model)

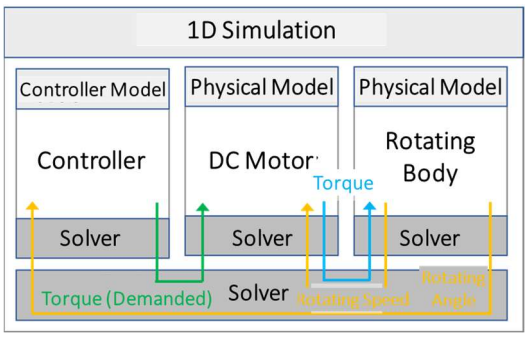


Figure. 2.4.5 (c) Supplement to CS model (Figure. 2.4.5 (b))

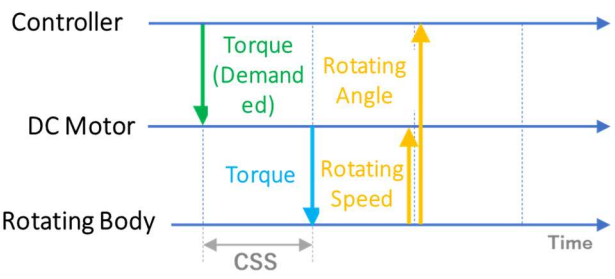


Figure. 2.4.5 (d) Supplement to the CS model (Figure. 2.4.5 (a)~(c))

2.5. Scheduled Execution (SE) structure and features

2.5.1. Configuration of FMU in SE mode

The Scheduled Execution (SE) interface is a new mode of operation introduced in FMI 3.0 and is intended for use in V-ECU (Virtual Electronic Control Unit) simulations.

The FMUs in the SE can be composed by several model partitions, as shown inFigure2.5.1. A model partition is an algorithm corresponding to the task of the controller and the variables used in the algorithm. For each individual model partition, a clock (t) can be associated to drive it, and the clock is used to run the model partition from a scheduler external to the FMU. The scheduler is the mechanism that determines the task execution order and executes them when multiple programs are apparently executed simultaneously in a multiprogram system (or in a multitasking or multiprocessing system). Normally, the scheduler is one of the main functions in the kernel part of the OS, but it is necessary to use such a scheduler to drive the SE's FMU.

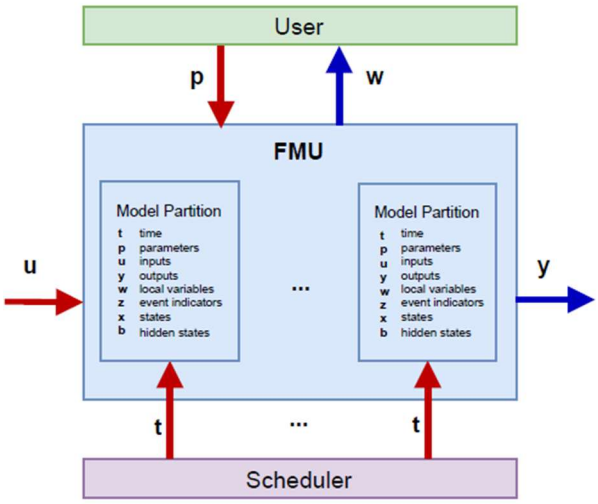


Figure2.5.1　SE mode signal flow

(FMI Specification: Functional Mock-up Interface Specification_Version 3.0])[7]

Thus, the model partition in the SE's FMU can be activated at any time from an external scheduler.

Clocks are a new specification introduced in FMI 3.0. Clocks can be used to control the execution of model partitions and the timing of events. The timing of clock generation can be driven by fixed-cycle driving, variable-cycle driving, or by sudden timing events, etc. Information on what timing model partitions can be activated is provided to the tool that imported the FMU from the SE FMU.

Clock countdown can be used as a way to control the execution of other model partitions from a model partition. The timing generated by the clock countdown in one partition can also be used to control the execution of other partitions.

Model partitions in the FMU can be defined in order of priority for execution. Preemption can also be used to suspend a model partition that is currently being executed and switch to execution of a model partition with a higher priority for execution.

2.5.2. Example of FMU execution in SE mode

Figure 2.5.2 shows an FMU with three model partitions, with time on the horizontal axis, and the execution of the three tasks. We will refer to the three model partitions as "Task 1," "Task 2," and "Task 3". Task 1 is driven by a 10[msec] periodic clock. Task 2 is driven by a non-periodic clock and generates the clock timing to start Task 2 by counting the clocks in Task 1. Task 3 is driven with a 50[msec] periodic clock. As for priority, Task 1 is assigned the highest priority, followed by Task 2, and Task 3 is assigned the lowest priority.

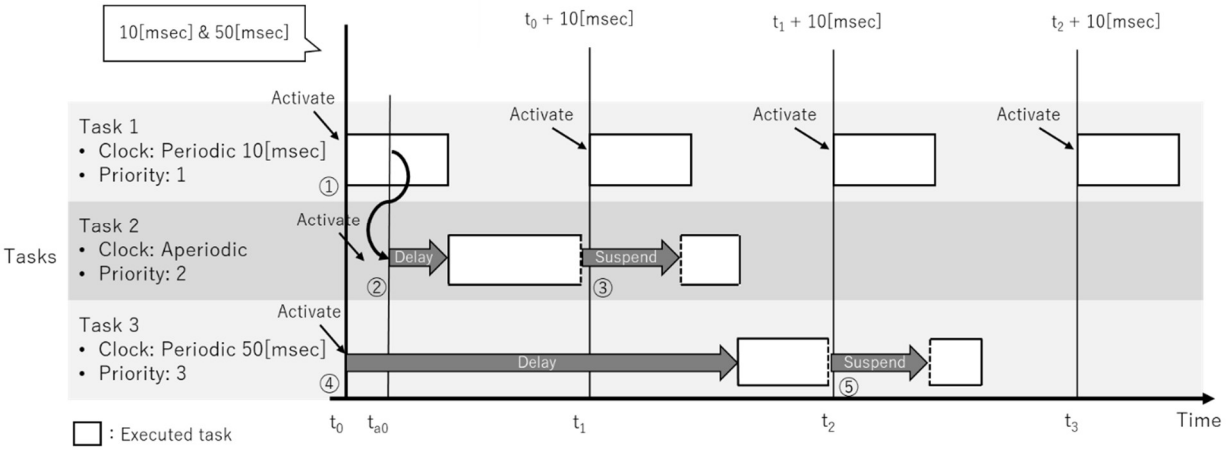


Figure 2.5.2　SE An example of the internal operation of an FMU in SE mode

(FMI Specification: Functional Mock-up Interface Specification_Version 3.0)[7]

① At time t0, 10[ms] and 50[ms] elapse simultaneously for the Task 1 and Task 3 clocks, and Task 1, which has priority over Task 3, is activated.

② The clock for Task 2 is triggered while Task 1 is running, but because Task 2 has a lower priority than Task 1, Task 2 waits until Task 1 finishes to begin execution. When Task 1 finishes, Task 2 starts.

③ Task 2 does not finish by the clock trigger time (t1) of Task 1, so Task 2 suspends execution and Task 1, which has higher priority, starts. After Task 1 finishes executing, Task 2 resumes execution.

④ Task 3 is triggered at t0, but waits to start until Task 1 and Task 2, which have high priority, have finished executing. After Task 2 finishes, Task 3 starts.

⑤ When Task 1 starts at time t2, Task 3 suspends execution. After Task 1 completes, Task 3 resumes.

2.6 General notes on FMI compatibility

2.6.1 Compatibility between different versions of FMI and different modes of operation

There are three versions of FMI: 1.0, 2.0, and 3.0. There is no FMU compatibility between them. Therefore, it is necessary to check whether the tool environment for creating FMUs and the tool environment for loading and executing FMUs are compatible with 1.0, 2.0, or 3.0 versions, and decide which version to use to exchange models.

In addition, the availability of Model Exchange (ME), Co-Simulation (CS) and Scheduled Execution (SE) depends on the tool environment, It is also necessary to check which mode of operation is used between the party creating the FMU and the party reading and executing the FMU.

Many tool environments for creating FMUs support multiple FMI versions and modes of operation, allowing the user to select the FMI version and mode of operation when creating the FMU. There are also tool environments that can create FMUs for both ME and CS modes.

On the other hand, there are many tool environments that load and run FMUs that can mix and match different versions of FMUs, or even FMUs in both ME and CS modes.

Overall, FMI 1.0 is an obsolete standard, so FMI 2.0 is recommended; FMI 3.0 should be used with caution since the standard is newer and only a few tool environments support it.

2.6.2 OS compatibility of FMU operating environment

Inside FMU, there is a folder called "binaries," which contains subfolders that store binary code for a combination of OS and bit number, such as win32, win64, linux32, and linux64. Although there are differences depending on the tool used to import and execute the binary code, these tools usually require a specific OS and bit number in order to execute the binary code. Therefore, it is necessary to create an FMU with a specific OS and number of bits in advance. Since it is allowed to have multiple subfolders under binaries, multiple executable library files (.dll files in Windows or .so files in Linux) can be placed in the FMU.

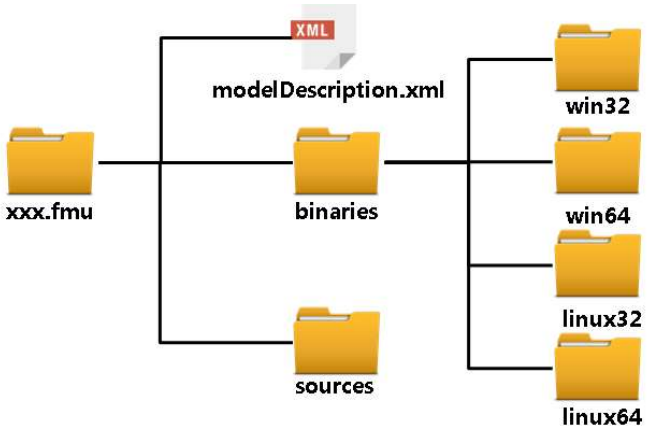


Figure 2.6.1 Structure of FMU

2.7. General notes on information concealment for FMIs

2.7.1. Observation of internal variables

The degree of concealment depends on the FMU generation tool. However, only inputs, outputs, parameters, and disclosed variables can be viewed externally. For this reason, variables that need to be observed must be consciously made available for external reference.

2.7.2. Source code passing

FMU has a folder "sources" where source code can be stored (Figure 2.7.1). Since storing source code is optional, source code should not be stored in the folder if concealment of information is considered. If you want to hide information, you should not store source code. On the other hand, as mentioned in the OS compatibility section, if the executable format is not stored and provided in binaries, it cannot be executed. For example, if you want to run an FMU on an ARM processor, most tools cannot create an executable FMU. In such cases, the source code is created internally, and then compiled and linked by the importing tool.

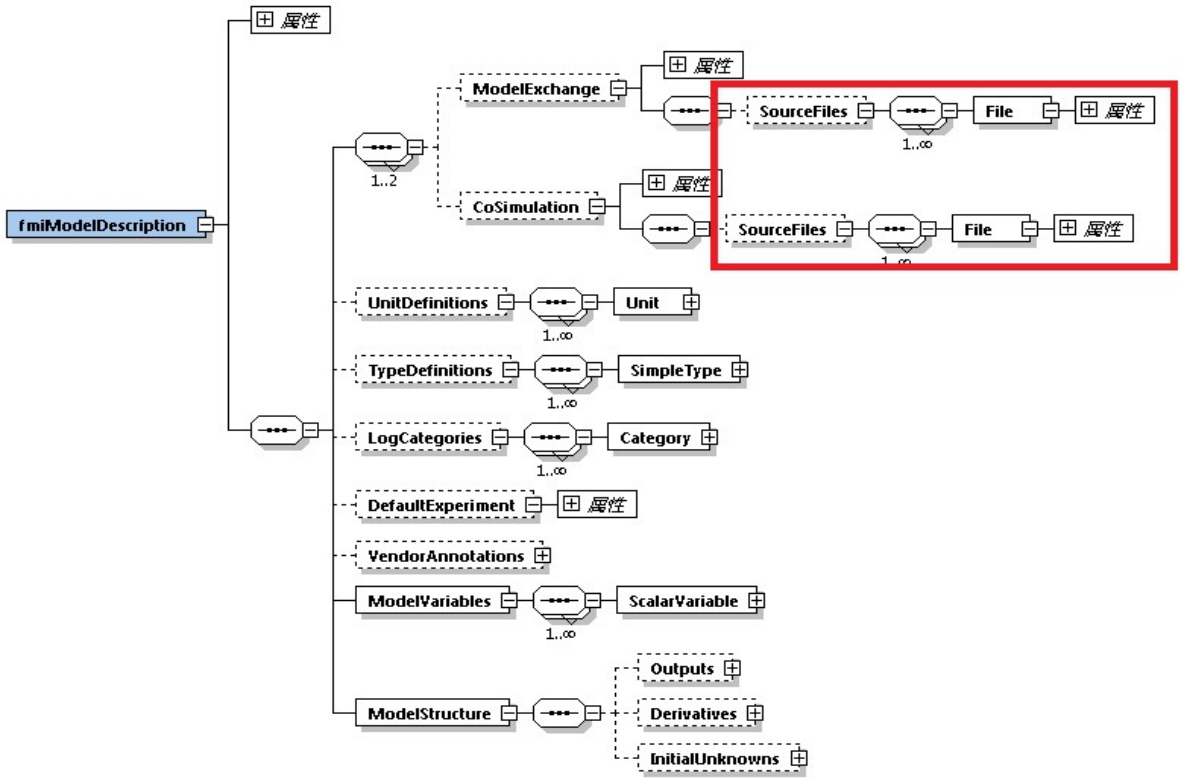


Figure 2.7.1　Structure of modelDescriptionFile (only the target part is expanded; in the red frame)

(FMI Specification:FMI_for_ModelExchange_and_CoSimulation_v3.0 :) [7].

2.8. General notes on tool environments that use FMI

2.8.1. FMU Execution License

In order to run an FMU, some tools that generated the FMU may require an execution license specified by the generating environment. If a license is required, please check the settings of the generating tool and the licensing environment of the importing tool, as simply passing an FMU is not enough to use it.

2.8.2. FMU execution environment

The FMU must be generated to match the OS and number of bits of the capturing tool, as described in Section 2.6.2. Additionally, in many cases, CS FMUs have an internal solver. However, in some cases, the FMU itself has only communication functions, and an environment in which the solver itself can be used for simulation is required. In this case, not only a licensed environment but also an environment in which the solver itself is installed and required.

2.8.3. Parameter Change

FMI regulations allow FMUs to define values (numeric, boolean, string) as parameters that can be changed before execution. Some of these values cannot be changed by the importing tool. If the value cannot be changed in the importing tool, the FMU must be re-generated and passed to the importing tool. The default values are written in the xml file of the FMU, but even if these values are changed, the parameter values cannot be reflected.

2.8.4. name

FMI has a fixed naming rule: strings that can be used in the name of the FMU itself, variable names (input/output, parameters), etc. are considered to be Unicode strings.

To avoid unwanted errors when importing into the tool, it is recommended to use a combination of letters, numbers and "_", starting with a letter of the alphabet.

2.9. About FMI Official Website

This section describes the official FMI website (https://fmi-standard.org/).

2.9.1. FMI's standards documents

Two types of standard documents, "Complete Package" and "Specification" for FMI 1.0 to 3.0.1 published so far, can be obtained from the official FMI website.

The "Specification" contains only the specification of the standard, while the "Complete Package" contains the specification of the standard, plus a schema definition file in XML format file and a common header file. The schema definition file can be used to verify that the data in the XML document is formatted and valued in accordance with the FMI standard. The common header file defines the API functions to be used when executing the FMU file. These two types of files are useful when creating or processing FMU files yourself, or when creating your own tools to read the internal information of FMU files.

2.9.2. Find out which tools are compatible with FMI standards

“Tools" page on the official FMI website allows you to search for tools that comply with FMI standards.

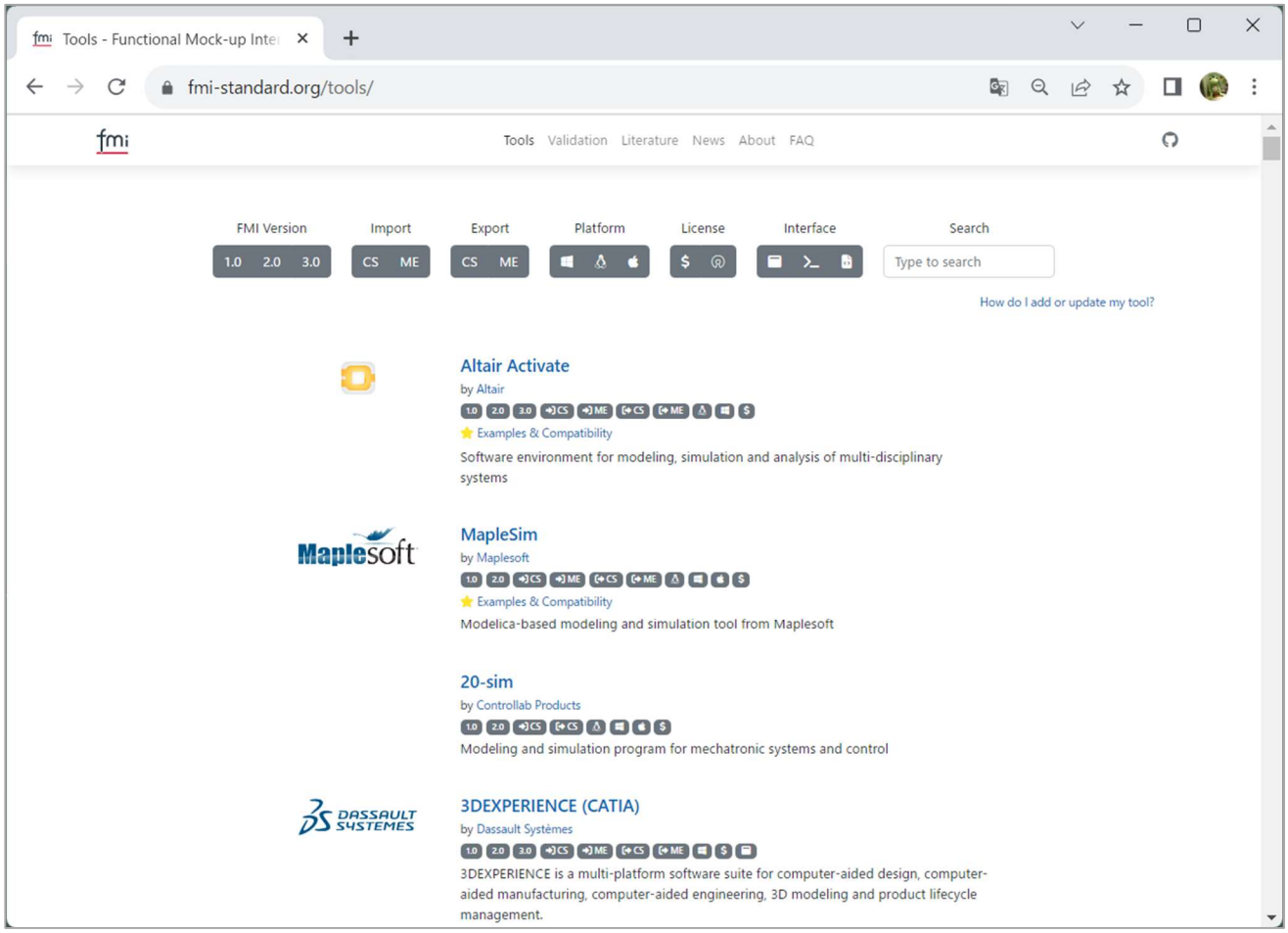


Figure2.9.1 Search function for tools compatible with FMI standards (FMI official website) [10].

The above page of the website allows users to search for FMI standard-compliant tools using free keywords, and also categorizes each tool's FMI standard compliance status using icons, making it easy to examine available tools according to the user's environment and application.

- Which FMI standard (FMI 1.0/2.0/3.0) is supported?
- Can import CS (Co-simulation mode) or ME (Model Exchange mode) FMUs?
- Ability to generate (export) FMUs in CS (Co-simulation mode) or ME (Model Exchange mode)
- Operating environment (Windows, Linux, iOS)
- Either paid or free tools
- The user interface of the tool is either a GUI method, command line method, or API library

Please check the FMI official website, which is updated daily.

2.10 FMI technical activities and events

2.10.1. Modelica Conference

FMI is one of the main topics of the Modelica Conference [20], as it is a part of the Modelica Association Project (MAP) [19], which is responsible for specification revision and deployment activities. The Modelica Conference is held every two odd-numbered years in Europe, and is a forum for the development and implementation of the FMI [21] and its derivative projects: SSP (System Structure and Parameterization) [22], DCP (Distributed Co-Simulation Protocol) [23], and eFMI (Functional Mock-up Interface for embedded Systems)[24], as well as the latest developments in other MAPs. In even-numbered years, the American Modelica Conference and Asian Modelica Conference are held in North America and Asia, respectively, to complement each other. More information on each Modelica-related topic can be found on the Modelica Association's website [1]..

Figure 2.10.1 shows the transition of the Modelica Conference and Modelica Association Project.

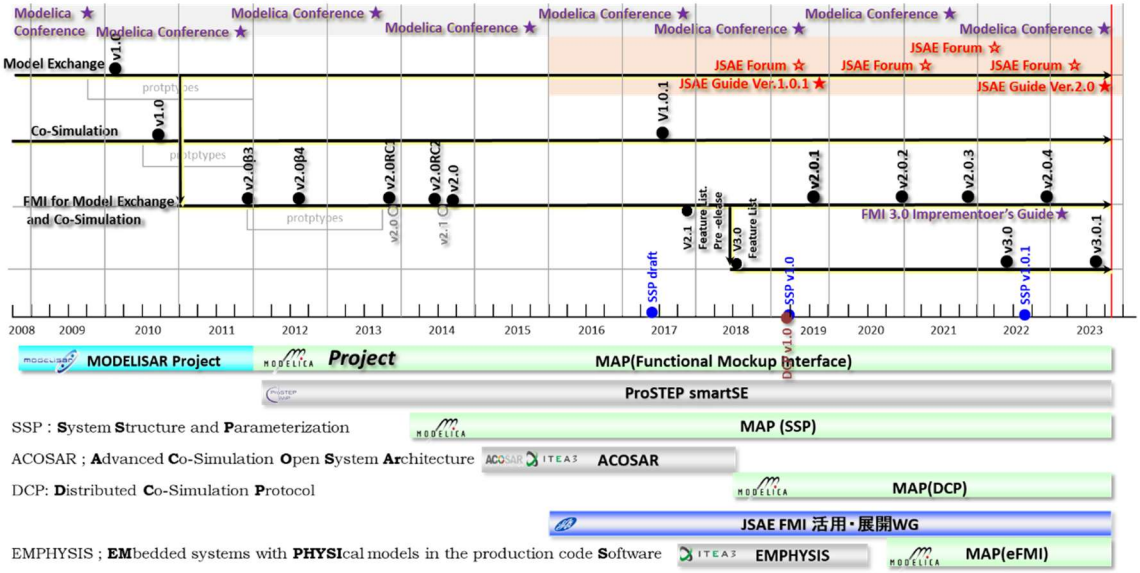


Figure 2.10.1 Modelica Conference and Modelica Association Project Transition

2.10.2. prostep ivip SmartSE Project

prostep ivip [25] is an industry association in Germany and other European industries that organizes conferences, publishes technical standards, and plans to raise awareness and deploy methods related to the efficiency of development work using digital development. Its scope includes standardization of 3D-CAD data, standardization of requirement specification description methods and management methods for MBSE (Model Based System Engineering), and standardization of model exchange/connection specifications and model distribution management methods for MBD (Model Based Development). The Smart Systems Engineering (SmartSE) Project [26] focuses mainly on activities to promote MBD in the automotive industry. Guidelines for Connecting and Exchanging Models are in effect. The latest version is Smart Systems Engineering Collaborative Simulation-Based Engineering Version 3.0 [27], which can be downloaded from the prostep ivip web page.

2.10.3. FMI Implementers' Guide

The FMI Implementers' Guide is a guidebook for implementing model import and export functions by　FMI in tools, and currently the FMI 3.0 Implementers' Guide [28] for FMI Ver. 3.0 is published by the collaboration of the　Modelica Association Project FMI and prostep ivip.

2.10.4. Others

Detailed specifications for all versions of FMI can be downloaded from the FMI Project homepage [20]. A list of the latest FMI-compatible tools can also be found on the homepage [29].

FMI is also being integrated with the Python environment, and free libraries for this function are included by　FMPy [30] and PyFMI [31].

Chapter 3: Guide to Selecting FMU Operation Mode and Flow of Each Process

In this chapter, we will discuss Model Exchange and Co-Simulation in details.

FMI has three methods: Model Exchange, Co-Simulation, and Scheduled Execution. Scheduled Execution was added from FMI 3.0 and is not included here because the tools that support it are limited. 3.1 section explains the guidelines for selecting the appropriate method by learning about the characteristics of each. 0 section describes the process from model creation to simulation execution, 3.3 and 3.4 sections describe possible problems and countermeasures in each process in order to find clues to solve problems when they occur.

3.1. Guideline for selecting FMU operating mode

Characteristics of each Model Exchange and Co-Simulation are shown inTable 3.1-1. The last part of the table summarizes the working group's recommended applications. Recommended application of Model Exchange mode is a controller. This is because control models generally do not contain algebraic loops, no communication delays are desired, and models are generally calculated by using explicit fixed time step solvers. Co-Simulation is useful for complex physical system models because it allows FMUs to be connected to other FMUs whose models and solvers have been confirmed to be compatible with each other using appropriate tools. In addition, since each FMU can have its own process, it is possible to perform parallel calculations to improve the simulation calculation speed, and multi-rate execution is also possible. When performing a Hardware in the Loop Simulation, FMU can also be distributed independent of the modeling tool and the hardware on which it is run.

Table 3.1-1 Model Exchange/Co-Simulation Features

	Model Exchange	Co-Simulation
Algebraic Loop containing FMU	Deprecated due to inability to generate FMU or loss of stability	Stable calculations are possible by the solver of generator tool. Possible increase in computation time.
Communication delay	no occurrence	Appropriate settings are required for Communication Step Size.
Variable time step SOLVA When used	Need to check model / solver compatibility.	Model / solver Compatibility has been checked at the time of FMU generation
When explicit fixed time step solver is used	Need to suppress model maximum eigenvalues	Need to suppress model maximum eigenvalues
Core Division	Impossible Single process for the entire model	Possible Processes can be generated for each

		FMU
Recommended Use	control model	Composite Physical System Parallel processing of large models Multi-rate model execution Hardware in the Loop Simulation

3.2. Work flow from FMU creation to exchange and simulation execution

For successful simulation, it is useful to understand the flow of FMU generation, exchange, and simulation execution in order to investigate the cause of a problem. The general flow is summarized inFigure 3.2.1.

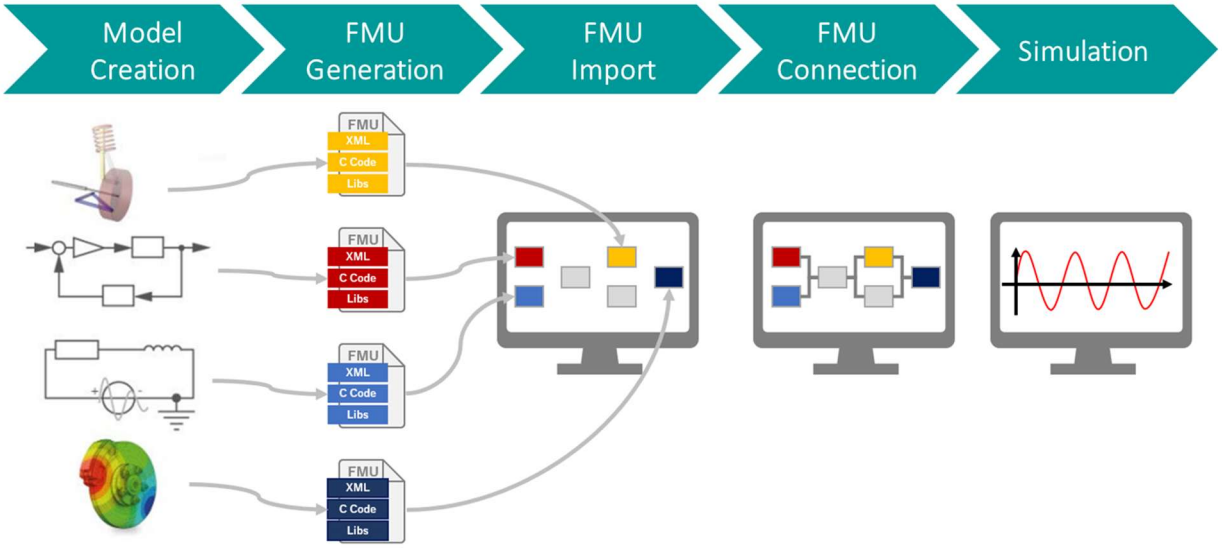


Figure 3.2.1 Flow from FMU generation to exchange and simulation execution

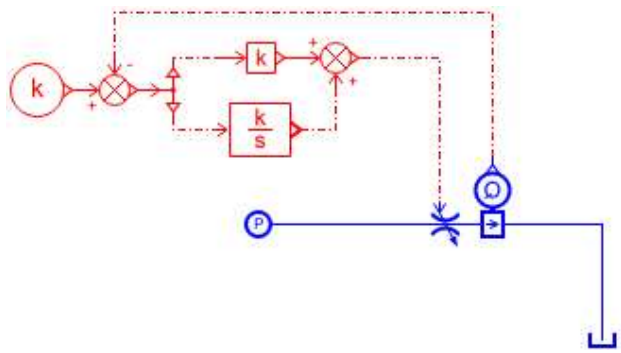
(Some of the figures are based on13th International Modelica Conference) [32].

3.3. Possible problems and countermeasures in each process (Model Exchange)

3.3.1. During Model Creation

It is recommended that there be no algebraic loops when creating Model Exchange FMU. If your model has an algebraic loop, identify the cause of the loop, take corrective action, and change the model to an explicit model.

Let's look at an example. In the model inFigure 3.3.1a, the variable orifice is PI-controlled to vary the opening diameter to achieve the target flow rate. However, an algebraic loop is occurring in the hatched area of Figure 3.3.1b.



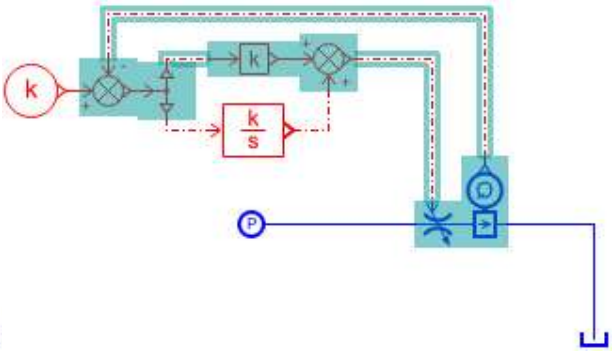


Figure 3.3.1a Example of PI control of flow rate **Figure 3.3.1b PI control example of flow rate: Algebraic loop Generating point**

As shown in Figure 3.3.2, the algebraic loop can be eliminated by inserting an integral calculation in-between the loop . One possible solution is to insert a first-order delay at the output to simulate the time response delay of the Actuator.

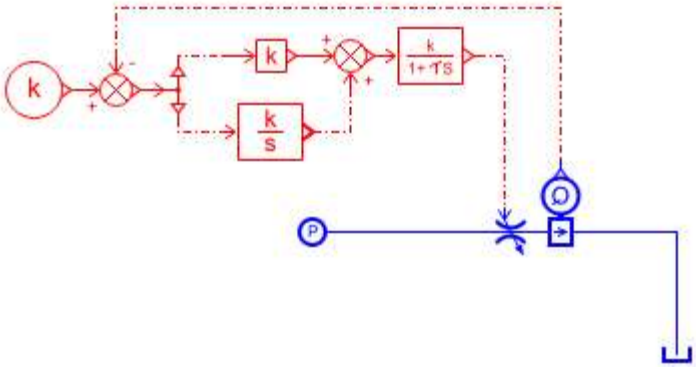


Figure 3.3.2 Insert a first-order delay

Note that this can occur when PI control is used within the physical model. Also, an algebraic loop in the physical model alone may also occur. When using calculation elements with constraining conditions such as constant potential or constant displacement, check for algebraic loops before generating FMUs.

3.3.2. During FMU generation and import

FMU parameters, state variable names, input/output information, etc. are stored in the model Description File, a file named modelDescription.xml. The FMI Specification describes how the model description file should be written, and the Compliance Checker is available on the FMI homepage [13].

3.3.3. When FMU is connected

The first challenge is to correctly connect the inputs and outputs of the FMUs. In the example in the attached tutorial, there is only one combination of connections between two FMUs, and the connections were made without any problems. However, when dealing with FMUs that have multiple inputs and outputs, it is necessary to check which inputs and outputs are to be connected. To

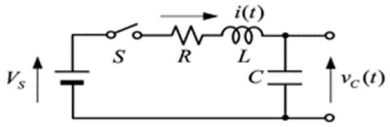
determine what variables should be exchanged between FMUs and what units should be used when exchanging various control and physical models, "Plant Model I/F Guidelines for automotive development (ver 4.0)" [34] can be utilized. Input/output rules based on these guidelines need to be agreed among model exchangers in advance. This point is discussed in detail in Chapter 4.

One problem which may occur when using Model Exchange is that unnecessary algebraic loops that do not really exist may occur when there occur loops in the input/output connections when FMUs are connected. In this case, the user can change the Dependency in the model Description File to eliminate the algebraic loop. For more information on this point, please refer to4.3.3 . Examples and countermeasures for this point will be introduced in 4.3.3.

3.3.4. During simulation run

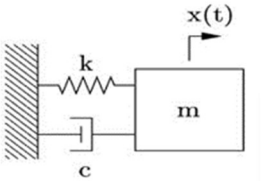
The simulation is performed using the solver of the importing tool when using Model Exchange mode. The success of the simulation can be described as the compatibility of the model and the solver. To understand this compatibility, knowledge about the stability of the simulation is helpful. Here is a brief introduction to the technical details.

The models exchanged using FMI are primarily represented only by ordinary differential equations related to time.



$$L\frac{d^2q(t)}{dt^2} + R\frac{dq(t)}{dt} + \frac{1}{C}q(t) = V_S \quad (1)$$

Figure 3.3.3 LRC circuit



$$M\frac{d^2x(t)}{dt^2} + C\frac{dx(t)}{dt} + Kx(t) = F_{ext} \quad (2)$$

Figure 3.3.4 Single-vibration system

Thus, in various physical domains, the behavior of the modeled object is represented by the second-order ordinary differential equations as in equations (1) and (2). As LRC circuit in Figure 3.3.3. and single-vibration system in Figure 3.3.4, these forms have eigenvalues and resonant frequencies.

The fact that the eigenvalues of the model being solved are within the convergence region of the simulation corresponds to that the simulation is performed in a solvable manner and not diverging.

Also, when dealing with detailed physical models, the system is most often nonlinear rather than linear time-invariant. In such cases, variable time step solvers are sometimes used to solve the system while changing the appropriate time step, solver order, etc. Variable time-step solvers are devised by each simulation tool and tuned to easily solve the model created by the tool. Therefore, when an FMU created by one tool is outputted via Model Exchange and then run the simulation by another tool, problems may occur, such as poor convergence or differences in the results. This is a model-solver compatibility issue.

3.4. Possible problems and countermeasures in each process (Co-Simulation))

3.4.1. During Model Creation

Co-Simulation outputs the model and solver together in the output. Therefore, even if there is an algebraic loop in the model, there is no problem. This can be useful when using a mechanism or electrical circuit model with constrained equations, or when using a model that requires a special solver, such as 3D-CFD.

3.4.2. During FMU generation and import

The problems of the successful exchange of FMU parameters, state variable names, and input/output information are similar to that of Model Exchange.

3.4.3. When FMU is connected

FMU input/output matching is similar to Model Exchange, but unnecessary algebraic loops are less likely to occur during Co-Simulation. This is because there is a communication delay due to Communication Step Size, which eliminates the algebraic loop. In most tools importing FMUs using Co-Simulation mode, even when there is a loop in the input-output connection, it is not considered as an algebraic loop.

3.4.4. During simulation run

There are three ways to perform Co-Simulation: The first is Coupling with System Models or simply called Stand Alone, where the solver and model are output together. The second and third are called Tool Coupling or Distributed, and are Co-Simulation, where only the interface between the tools is used and both software are launched. Note that different methods require different software and installation environments.

The calculation results may diverge or the calculation speed may slow down. This is a problem caused by the Communication Step Size in Co-Simulation. Also, the simulation results may differ depending on the Communication Step Size setting.

In Co-Simulation, communication occurs between the master and slave at set time intervals. During the communication interval, the input value of each FMU is treated as constant, so a step response delay occurs between the FMUs. Some tools perform interpolation to make the input continuous, but the insertion of this step response delay can cause the following issues:

1) The results may diverge or be slow due to the Communication Step Size setting in Co-Simulation. Simulation results may differ depending on the Communication Step Size setting.

2) In addition, the smaller the communication interval, the smaller the effect of the step response delay on the entire model system, and thus the lower the calculation error.

3) However, setting a fine communication interval increases the number of discontinuities in communication, which increases the number of convergence operations in the solver and thus the computation time.

In order to keep the communication interval as large as possible, it is desirable to connect models at points where their coupling is weak. In addition, to reduce the number of convergence computation operations, an explicit Fixed time step solver. However, it is necessary to confirm that all eigenvalues of the model are within the solver convergence region and do not diverge.

Depending on the tool used, the functionality may or may not be available, but if the computer has multiple cores or threads, Co-Simulation allows separate executables to be executed for each FMU, enabling distributed processing. In contrast, Model Exchange is basically a single executable for the entire model, so it is not suitable for distributed processing. This feature can be utilized to improve the calculation speed.

Chapter 4: What You Need to Know for Practical Application

In this chapter, we will explain the points to note in generating FMUs to actually use FMI, as well as errors up to the execution of reading.4.1 In Section 4.1, the JAMBE guidelines for determining the input/output interfaces of a plant model are explained. 4.2 lists the precautions to be taken when dividing the model into various parts.4.3 4.3 describes possible errors in using FMI.

In order to distinguish between the notes on the use of model linkage between different tools in general and those specific to FMI, we use the terms [Common] and [FMI].

4.1. Guideline for determining plant model inputs and outputs

When multiple models are combined into one overall model, the input-output relationship of the models is important; the same is true when using FMI. In the following section4.2, we will discuss the four relationships between plant and controller and explain some notes on connection and partitioning. Before doing so, we will refer to the JAMBE (MBD Promotion Center) website Plant Model I/F Guidelines for Automotive Development (hereinafter referred to as "JAMBE Guidelines"), which are available on the JAMBE website. Here, we will first briefly explain the five principles presented, and then explain the inputs and outputs of the models to be converted to FMUs in relation to these principles.

Table 4.1-1 JAMBE Guidelines Principles

Basic principles	
First	Plant models should be connected by across variables and through variables. Also, the across variables and through variables shall be oriented opposite to each other.
Second	Energy source to energy sink is the positive direction of energy.
Third	Consider the overall interface based on the element that accumulates the amount of through variables and across variables.
Fourth	The sign of the through variable is positive when the direction of positive flow of energy is the same as the direction of input and output of the through variable.
Fifth	SI unit system and SI assembly unit system are used for input and output units. JIS standard is used for quantity symbols.

(Adapted from Plant Model Interface Guidelines for Automotive Development (ver. 4.0))

4.1.1. Selection of connection signal [Common].

This item is the application of the first and third principles of JAMBE Guideline. According to the first principle, in plant-to-plant connections, if one plant has a through variable input, that plant outputs an across varialbe quantity as shown in Figure 4.1.1. The another plant to be connected will receive the amount of the across variable as input and will output the amount of through variable that the first plant receives. By the third principle, as shown inTable4.1-2 (a) and Table4.1-2 (b), the physical parts represented at the connection site are divided into two parts.

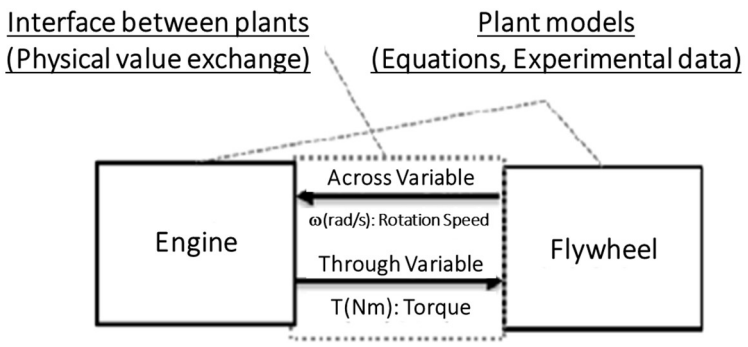


Figure 4.1.1 First principle (adapted from JAMBE guidelines)

Table4.1-2(a) Through Factors that accumulate quantities (with the through variable as input)

Physical area	Mechanical (translational)	Mechanical (rotating)	Electric	Thermal
Physical Components	Mass	Moment of inertia	Electric capacity (capacitor)	Heat capacity
Input: Through Variables	Power	Torque	Electric current	Heat flow rate
Output: Across Variables	Velocity	Angular velocity	Voltage	Temperature

Table4.1-2 (b) Across Factors that accumulate quantities (with the across variable as input)

Physical area	Mechanical (translational)	Mechanical (rotating)	Electric	Thermal
Physical Components	Spring	Torsion spring	Electric induction (inductor)	na
Input: Across Variables	Velocity	Angular velocity	Voltage	na
Output: Through Variables	Power	Torque	Electric current	na

As an example, consider the following for a mechanical system (rotational).
Moment of inertia accumulates the through amount (torque) (seeTable4.1-2 a). For this reason, if the moment of inertia is attached to a connection in the model, it is recommended that the torque be the input and the angular velocity (across amount) be the output. If you try to generate an FMU with a velocity input with a non-causal tool, an error will occur during generation, and you will need to change to an acceleration input to avoid this. However, generating acceleration requires differentiation, which is considered unstable in terms of numerical simulation. For this reason, JAMBE guidelines require, in principle, the input of through variables rather than across variables. If the plant model is created using a non-causal modeling tool, it may be possible to achieve a

connection different from this principle by appropriately using a non-causal adapter as described in the Society of Automotive Engineers of Japan's " Guidelines for FMI Model Connection Using Non-causal Modeling Tools Ver. 1.0 " [4].

For the connection based on JAMBE Guidelines, the moments of inertia cannot be connected to each other. In this case, the WG considers it preferable to combine the moments of inertia on either side. If it is unavoidable to separate them in a way that both sides have moments of inertia, a rotational spring element should be inserted in one of them to allow from through variable input to across variable input. In this case, if the magnitudes of the rotational spring and moment of inertia are not appropriately related, the result will be an unstable model with small time constants and long simulation times.

Contrary to this, for elements with a torsion spring at the connection, the amount of across variable (angular velocity) can be used as the input. The torque, which is a through variable, can be used as the output.

In addition, by the JAMBE Guideline, If the damping (torsional damper) is at the connection, the through variable input or across variable input is acceptable. In mechanical systems, it is not common to actually have only damping at the connection. In electrical systems, electrical resistance can be both a through variable input and an across variable input, and this is a connection that is quite conceivable.

4.1.2. Signal Arrangements [Common].

4.1.2.1. Sign of Input/output signals

This is the part where the second and fourth principles apply: as an example of the second principle, the JAMBE Guidelines define the flow of energy (work rate or power) from the engine side to the drive train and running resistance as positive. According to this, the relationship between torque input from the transmission to the differential (and angular velocity input from the differential to the transmission) is that positive torque is sent to the differential during acceleration and negative torque is sent to the differential during deceleration.

4.1.2.2. Unit system

The fifth principle states that the unit system should be the SI unit system. FMI allows the unit system to be described in the modelDescription.xml file and to be held explicitly internally. However, the FMI standard does not require the use of a unit system, so the generated FMU does not necessarily contain this description. It is also not always the case that the unit system definition is used on the importing side.

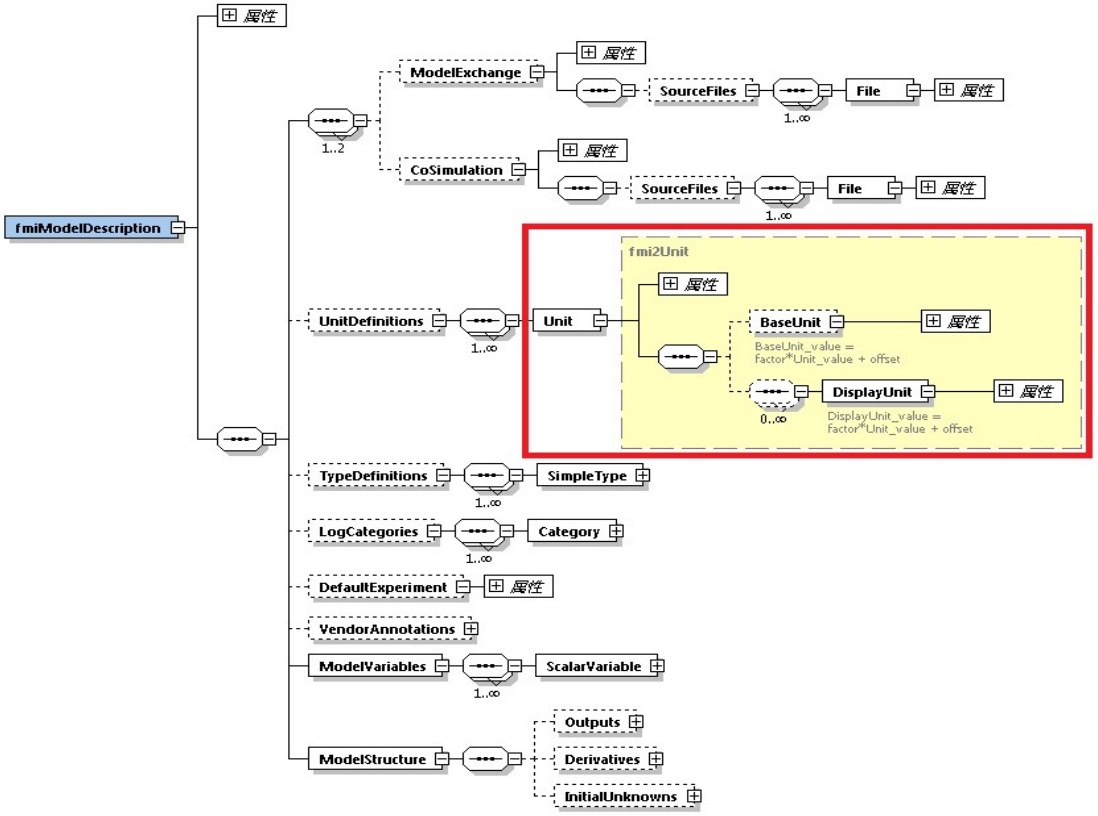


Figure4.1.2 Structure of modelDescriptionFile (only the target part is expanded; in the red frame)

(Adapted from FMI Specification: FMI_for_ModelExchange_and_CoSimulation_v3.0) [7].

In automotive development, [rpm] is often used to express rotational speed instead of [rad/s]. Temperature is often expressed in degrees Celsius [°C]. Because this system of units can be easily misunderstood, The JAMBE Guidelines suggest that the units be given and received as a "Subsystem Interface Definition Sheet". The WG also considers it desirable to communicate this information outside of the FMU, which is the model.

4.2. Notes on model splitting

In the previous section, we discussed the division of the plant model. In this section, we will explain the division of the sub-model by dividing the two connections into a controller and a plant.

4.2.1. Controllers and Controllers [FMI].

The analog (continuous) signal input/output and the digital (discrete) signal input/output need to be considered separately. In the case of ME, the actual analog signals are simulated without any delay between models. Since digital signals originally have a delay due to sampling, we must consider the effect of the delay in relation to the communication interval of the CS.

Procedure for CS:

The processing performed in one controller is done in one FMU without separating the FMUs.

Signals that only pass through are not transmitted via the FMU. As far as possible, the controller that is the signal source should be directly connected to the receiving controller.

Until FMI2.0, bus (or vector) terminals did not exist (FMI3.0 supports bus terminals). When connecting controllers, FMI2.0 requires that all signals be connected one at a time, resulting in a large number of connections. Make connections only to those inputs and outputs that are necessary. In particular, unnecessary input terminals should not be provided because many tools will generate an error if there is no input signal.

The FMI convention defines a disclosure variable (v) in addition to the output variable (y). If there is a variable that you want to observe on the importing tool, you need to define it as (y) or (v) at the time of generating the FMU. On the other hand, some read-in tools cannot observe (v). In this case, variables that are not connected may also need to be set as output variables.

4.2.2. Controllers and Plants [Common].

4.2.2.1. Signals between controller and plant

In general, the indication and observation value signals transmitted and received between the controller and the plant are electrical signals through cables. But in many cases they are not modeled as a circuit. Strictly speaking, it is necessary to consider signal delays in electrical circuits. It is necessary to divide the controller and the plant by considering at what part the instruction signal to the actuator or the sensor signal output by the plant is discretized. Especially in the case of CS, it is necessary to consider whether the delays are correctly represented as discussed in the previous section.

4.2.2.2. Physical quantity between controller and plant

On the model, it is similar to the signal written in 4.2.2.1, but when exchanging physical values that are actually connected as through and across variables, the situation is the same as the plant-to-plant connection shown in 4.2.3. For this reason, it is necessary to determine if the signal is the same as the one written in 4.2.2.1. For exampleFigure 4.2.1 (a), where the controller (FMU1) conveys the current value to the motor in the plant (FMU2) and the voltage returns from the plant, even though the creator considered FMU1 as the controller, the controller itself has the characteristics of the plant, so the result is the behavior shown in Figure 4.2.1(b).

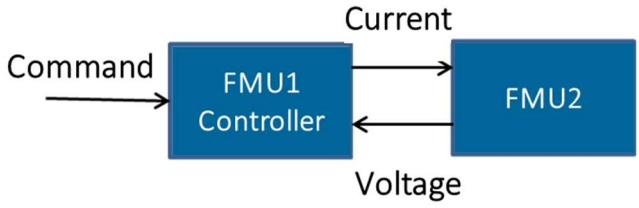


Figure 4.2.1 (a) Example model of controller and plant

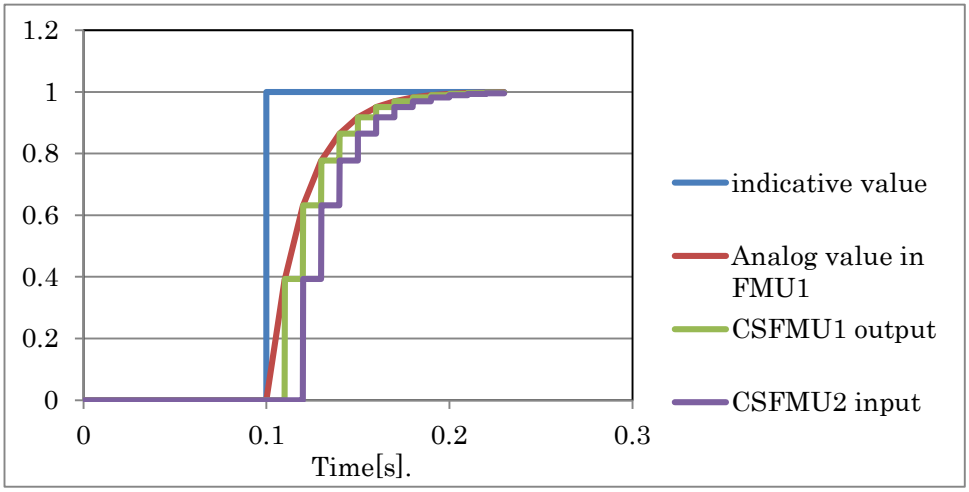


Figure 4.2.1(b) Example of delay in CS (communication interval = 0.01s)

4.2.3. Plant and Plant [Common].

The most attention should be paid to plant-to-plant connections where the mechanical, electrical, and other physical joints are split into different models. For inputs and outputs, the WG recommends following the JAMBE guidelines. If the tool that loaded the FMU is a tool that allows non-causal connections, there is also a way to use the non-causal adapter mentioned in 4.1.2. When dividing and connecting according to the JAMBE guidelines as explained in 4.1, it is not possible to directly connect parts that have a through variable as input to each other and parts that have an across variable as input.

4.2.3.1. Location where model splitting is to be avoided

We believe that division at sites with fast movements (small time constants) or high discontinuities in the original movement should also be avoided as much as possible.

・Example of a part with a small time constant: In a mechanical system, the translational/rotational spring constant is large in relation to the mass/moment of inertia.

・Examples of discontinuous parts: Friction and thrust (collision, limitation of mechanical range of motion) in mechanical systems, and the diode or other switching devices in the electrical systems.

In the case of a diode or other switching device, it is the diode or other switching device.

4.3. FMI and Error

Errors in models using FMI can be FMI-specific or general tool-related. 4.3.2 onward will explain the various types of errors, but if the generated tool can import the model, the first thing to do is to check that there are no problems by importing and executing the model with the generated tool itself.

There are two types of error messages: one is generated by the tool that imported the data, and the other is generated by FMU itself. If we can identify which one is producing the problem, we can find a solution. The error message from the FMU may be in the source code that generated the FMU.

4.3.1. FMU's FMI Standards Compliance Verification Tool [FMI]

Compliance Checker is available on the FMI official website The Compliance Checker can be run on the FMI official website. Figure 4.3.1 is the execution page. When clicking “Validate your FMU ➔" in the "FMU Check" section ofFigure 4.3.1, Figure 4.3.2 will be displayed. Click the "Select" button to specify the FMU you wish to check. Sample models can be downloaded from the Automotive Controls and Models Section Committee page of the Society of Automotive Engineers of Japan website.

File name : Sample_3p1.zip　　: Extract FMU_J1_Case1_Dymola_ME_2.fmu from the ZIP file and check it.

An example of the results is shown inFigure 4.3.3. No problems are reported. Now click on the "Model Info" tab, then the file information of the FMU is displayed as shown in Figure 4.3.4. Also, if you click on the "Variabes" tab, the information on variables included in the FMU will be shown as in Figure 4.3.5. Clicking on the "Files" tab will show information on files stored in the FMU as in Figure 4.3.6.

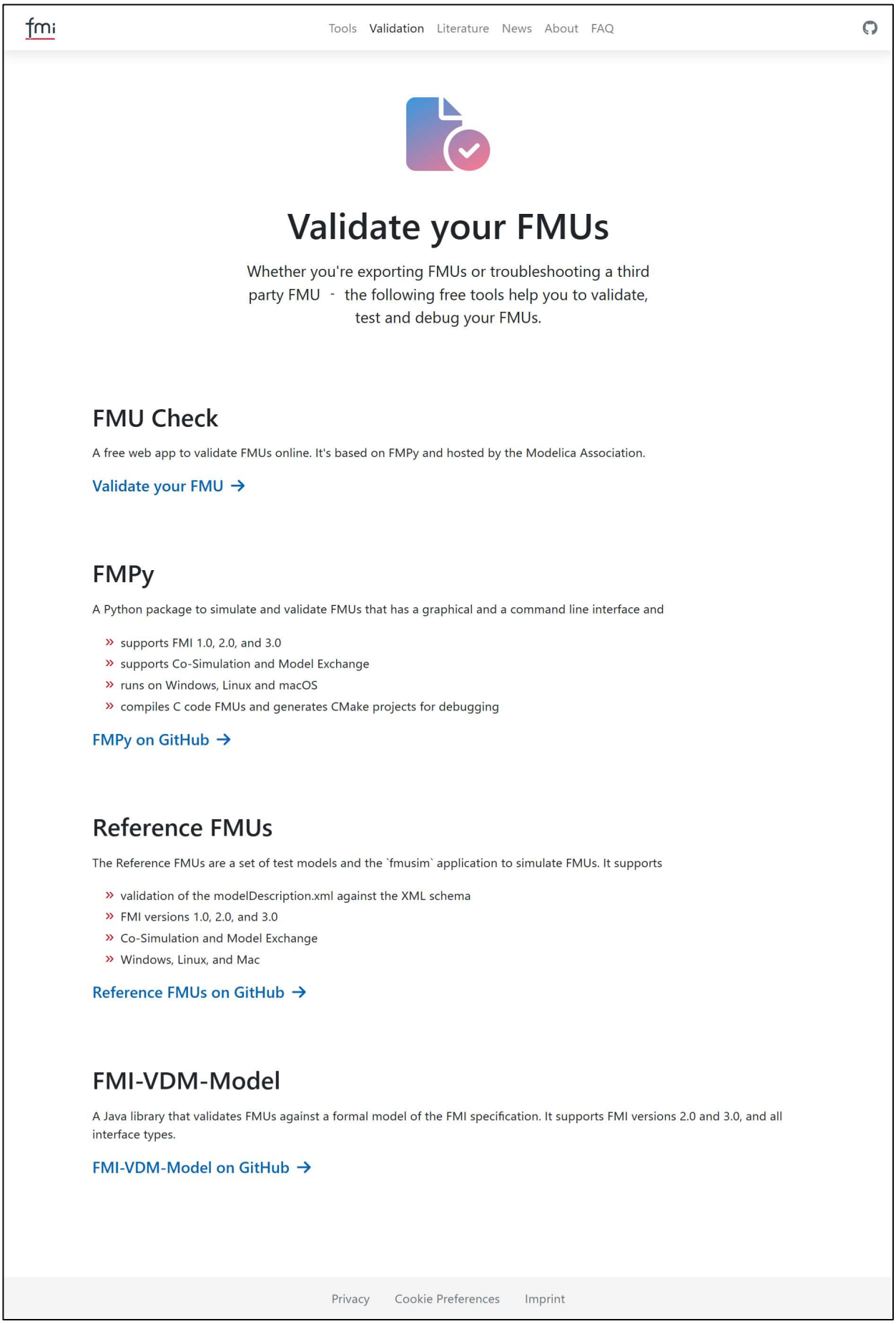


Figure 4.3.1 Compliance Checker Execution Web Page

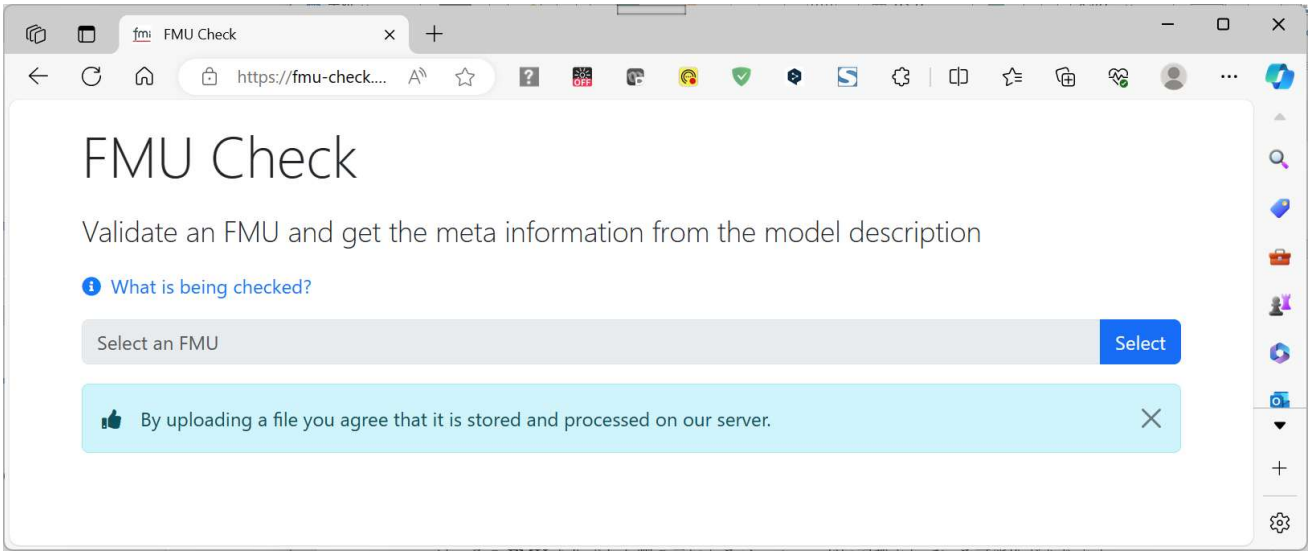


Figure 4.3.2 Compliance Checker Execution screen of Compliance Checker

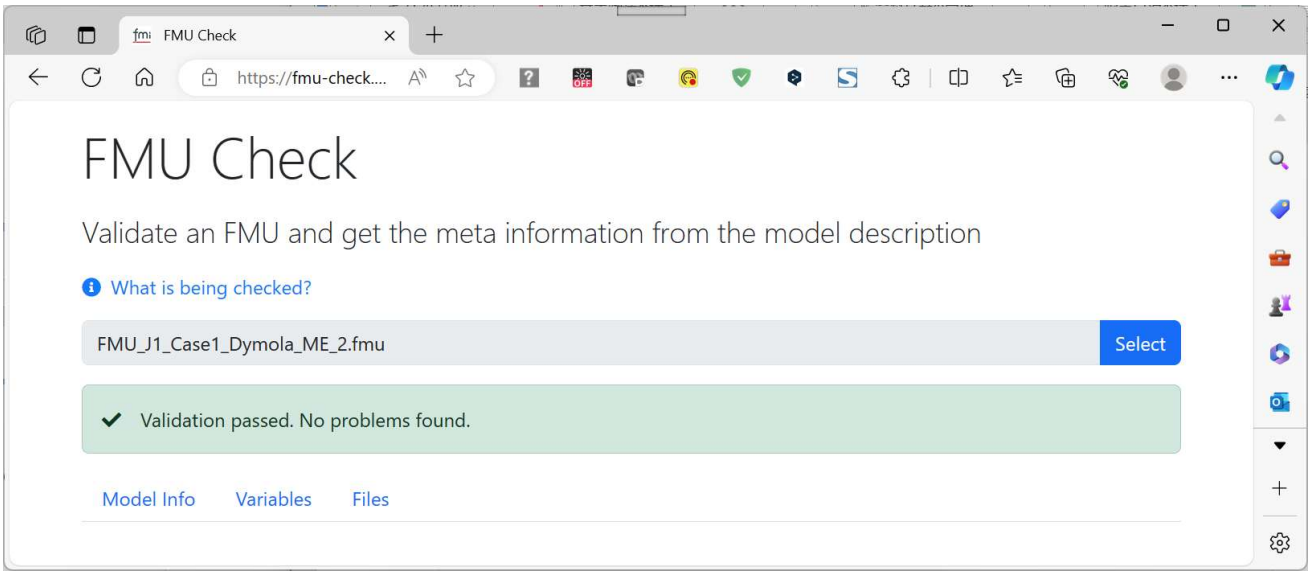


Figure 4.3.3 Compliance Checker Execution Result Screen

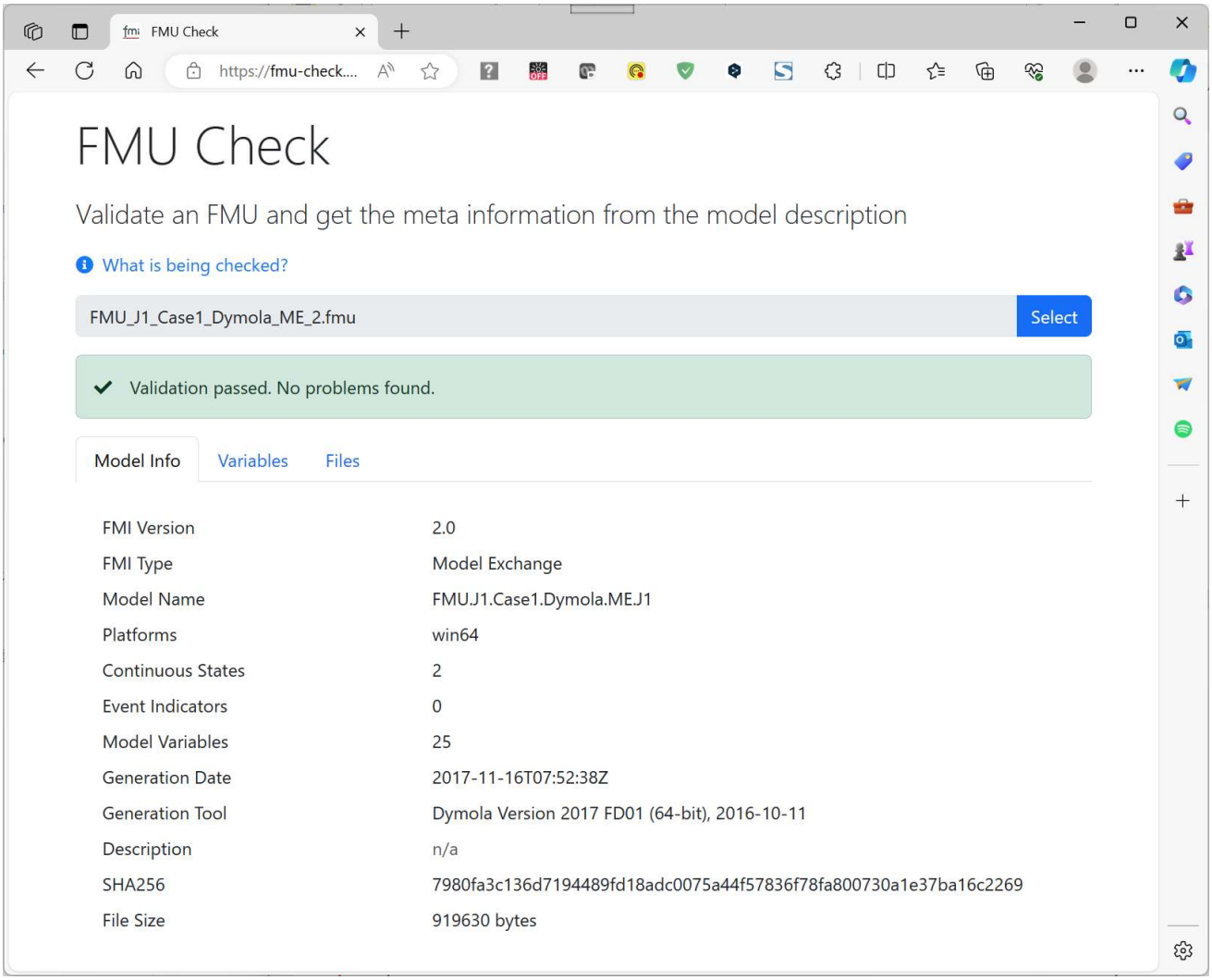
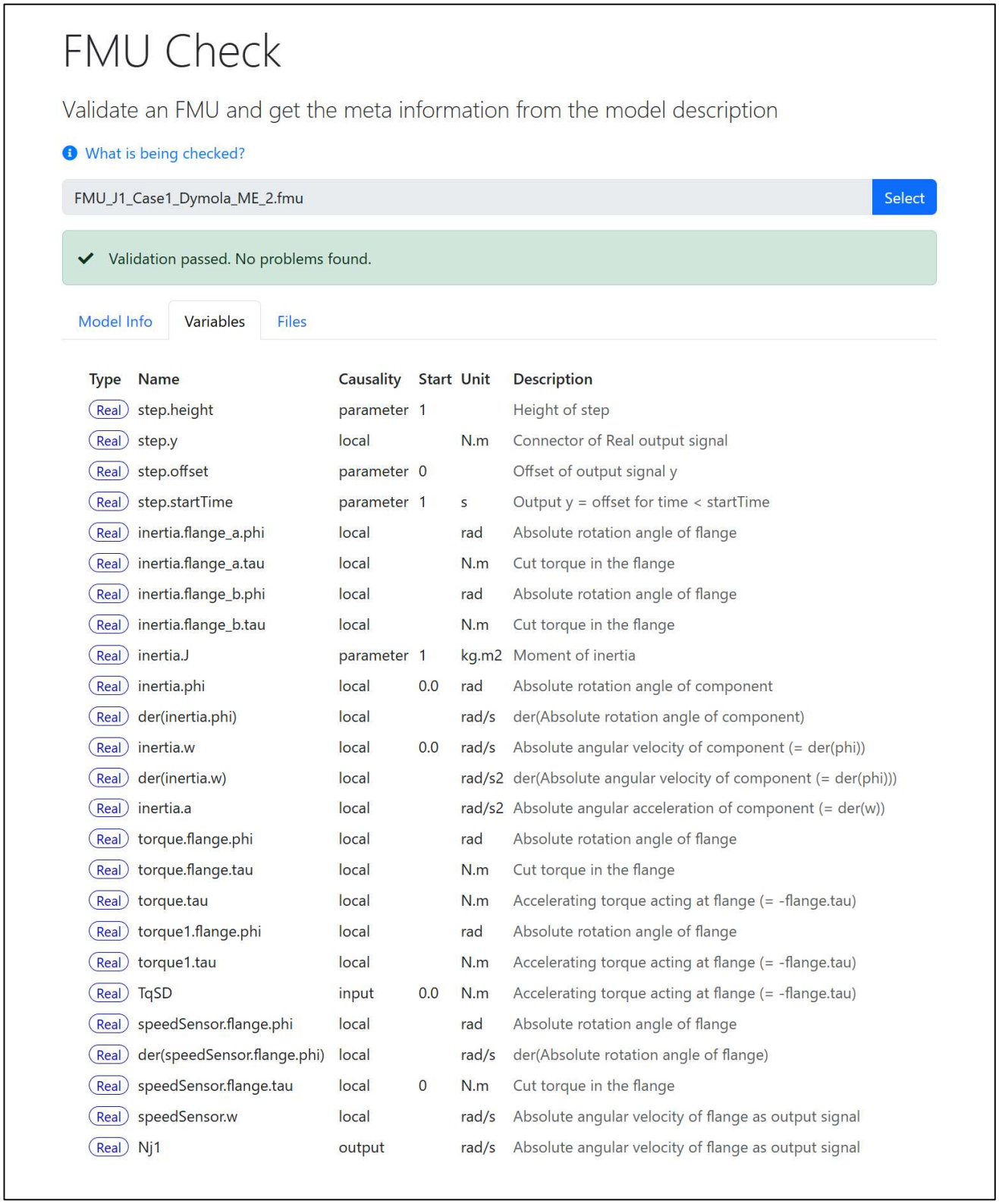


Figure 4.3.4 Model Info screen of Compliance Checker



Type	Name	Causality	Start	Unit	Description
Real	step.height	parameter	1		Height of step
Real	step.y	local		N.m	Connector of Real output signal
Real	step.offset	parameter	0		Offset of output signal y
Real	step.startTime	parameter	1	s	Output y = offset for time < startTime
Real	inertia.flange_a.phi	local		rad	Absolute rotation angle of flange
Real	inertia.flange_a.tau	local		N.m	Cut torque in the flange
Real	inertia.flange_b.phi	local		rad	Absolute rotation angle of flange
Real	inertia.flange_b.tau	local		N.m	Cut torque in the flange
Real	inertia.J	parameter	1	kg.m2	Moment of inertia
Real	inertia.phi	local	0.0	rad	Absolute rotation angle of component
Real	der(inertia.phi)	local		rad/s	der(Absolute rotation angle of component)
Real	inertia.w	local	0.0	rad/s	Absolute angular velocity of component (= der(phi))
Real	der(inertia.w)	local		rad/s2	der(Absolute angular velocity of component (= der(phi)))
Real	inertia.a	local		rad/s2	Absolute angular acceleration of component (= der(w))
Real	torque.flange.phi	local		rad	Absolute rotation angle of flange
Real	torque.flange.tau	local		N.m	Cut torque in the flange
Real	torque.tau	local		N.m	Accelerating torque acting at flange (= -flange.tau)
Real	torque1.flange.phi	local		rad	Absolute rotation angle of flange
Real	torque1.tau	local		N.m	Accelerating torque acting at flange (= -flange.tau)
Real	TqSD	input	0.0	N.m	Accelerating torque acting at flange (= -flange.tau)
Real	speedSensor.flange.phi	local		rad	Absolute rotation angle of flange
Real	der(speedSensor.flange.phi)	local		rad/s	der(Absolute rotation angle of flange)
Real	speedSensor.flange.tau	local	0	N.m	Cut torque in the flange
Real	speedSensor.w	local		rad/s	Absolute angular velocity of flange as output signal
Real	Nj1	output		rad/s	Absolute angular velocity of flange as output signal

Figure 4.3.5 Variables screen of the Compliance Checker

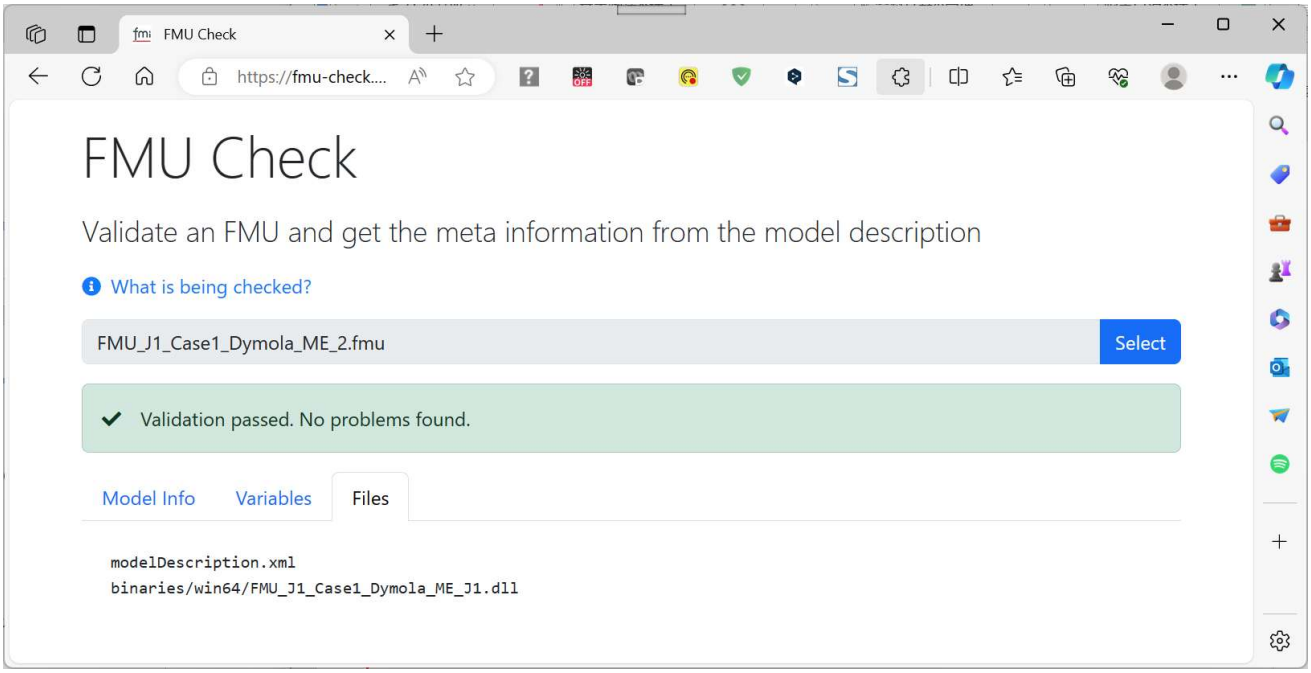


Figure4.3.6 Files screen of the Compliance Checker

4.3.2. Importing error [FMI].

This is a problem specific to FMI. When you try to import an FMU with the FMU importing tool, you may encounter an error. In this case, it is recommended to check using the Compliance Checker in 4.3.1.2.8.4 There are many cases of errors due to the naming scheme described in Section 2.8.4.

4.3.3. Connection error [FMI].

A possible issue with Model Exchange connections is that when connecting FMUs and there is a cycle in the input/output chain, an unnecessary algebraic loop may be created that does not actually exist. In this case, the user can change the Dependency in the model Description File to eliminate the algebraic loop.

Let's look at a concrete example. This is an example of a model description file generated as the rotational inertia side (FMU1) in the attached example.

```
<ModelStructure>
    <Outputs>
        <Unknown index="9"/> ⟶ Output signal (speed)
    </Outputs>
    <Derivatives>
        <Unknown index="2"/> ⟶ Acceleration (derivative of speed)
        <Unknown index="4"/> ⟶ Speed (derivative of displacement)
    </Derivatives>
    <InitialUnknowns>
        <Unknown index="1"/> ⟶ Speed
        <Unknown index="3"/> ⟶ Displacement
        <Unknown index="9"/> ⟶ Output signal (speed)
    </InitialUnknowns>
</ModelStructure>
```

Variables handled inside the FMU have an index set. This index is set for the state variable to be integrated and for other variables used in the model from which the FMU is generated . In this case,

the output signal is the rotation speed (index="9"), which is the value of the state variable (index="1") detected.

By declaring the dependency between these two variables, the importing tool knows that the output signal is a state variable that is being integrated, and that the relationship between the input and output is not an algebraic equation. However, in the above example, the dependency is not expressed, so if the importing tool understands that the input/output relationship may be an algebraic equation, and applies a similar understanding to FMU2, it will understand that an algebraic equation loop occurs between FMU1 and FMU2. In reality, the output value of each FMU is the integrated value of the input signal, and it should be possible to solve it using an explicit solver, but we will use an implicit solver that can solve algebraic loops.

When solving using the implicit method, the close-loop transfer function gain must be less than or equal to 1.

To resolve the algebraic loop in this example, one countermeasure is to change the model Description File as shown below.

```
<ModelStructure>
    <Outputs>
        <Unknown index="9" dependencies = "1"/>
    </Outputs>
    <Derivatives>
        <Unknown index="2"/>
        <Unknown index="4"/>
    </Derivatives>
    <InitialUnknowns>
        <Unknown index="1"/>
        <Unknown index="3"/>
        <Unknown index="9"/>
    </InitialUnknowns>
</ModelStructure>
```

By declaring "dependencies", it is understood that the output value (index="9") is a state variable to be solved integrally, preventing unnecessary algebraic loops.

Please note that whether or not measures such as the above example are necessary, or whether or not measures to eliminate algebraic loops when they occur are possible, depends on the tool. The following is an example of countermeasures to be taken when problems related to algebraic loops occur.

4.3.4. Initial value error [Common]

Initial value errors can be classified into two types. Neither of these problems is unique to FMI

4.3.4.1. Initial value inconsistency

This is an error when two FMUs or both the FMU and the model of the capturing tool have different initial values for the same variable. The workaround is to unset the initial value for one of them.

4.3.4.2. Initial value calculation error

If the initial values are not given explicitly, some tools that run the simulation perform initial value

estimation. This error occurs when the initial value cannot be estimated, and can be resolved by explicitly specifying the initial value. In FMI, the initial value can be explicitly specified when generating the FMU, but it is necessary to set it so that the "initial value inconsistency" mentioned above does not occur. If the FMU generation tool allows you to specify the initial value as a parameter, it is preferable to specify it as a parameter.

4.3.5. Run-time error

4.3.5.1. Initial simulation error [FMI].

This is similar to the initial value calculation error, but it is different. According to the FMI regulations, the value input to the FMU at the first simulation (for example, when the calculation start time is set to 0, at time=0) is 0. It has been reported that an error due to division by zero occurs as a result of running a simulation using this input signal value.

4.3.5.2. Divergence [Common]

Typical examples of errors are shown in 1) and 2) below. None of these problems are unique to FMI.

1) The value gradually increases during the simulation and the simulation terminates abnormally.

This section will be divided into two cases: ① those caused by the method of numerical calculation of individual models, and ② those caused by the method of connection between models.

①The model has eigenvalues and resonant frequencies, but the condition for no divergence is that the eigenvalues are in the convergence region of the solver being used. For reference, the stability region of the Runge-Kutta solver is shown inFigure4.3.7. The condition for convergence is that the eigenvalues are in the region represented on the complex plane. The region varies with the Δt of the solver time step. Measures can be taken such as making the Δt finer and keeping it within the convergence region, or using a solver with a wider convergence region.

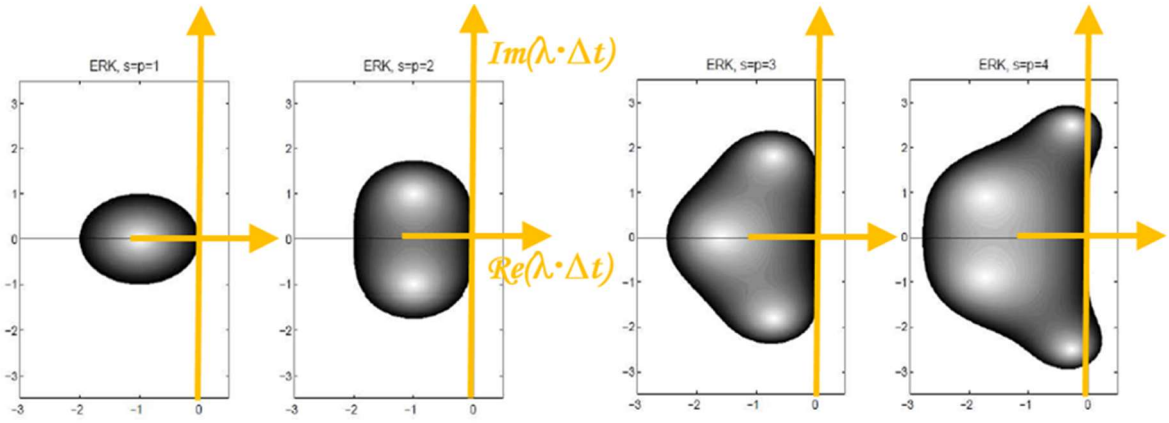


Figure4.3.7 Stability Regions (1st to 4th Order) for Runge-Kutta solve

②A case frequently seen in plant-to-plant connections is when the sign of a physical quantity is reversed. This is a result of positive feedback when it should be negative feedback (Figure 4.3.8, middle panel). The relationship with the direction of energy must be sorted out as shown in 4.1.2, and either generate the FMU again after correcting it with the generator tool, or invert the sign

with the reader tool.

2) Midway through the simulation, the signal oscillates and the simulation terminates abnormally.

It is possible that the high natural frequencies in the plant model affect the signal transfer (delay due to the signal transfer) at frequencies lower than the natural frequencies, resulting in an unstable case (see Figure 4.3.8, bottom panel). In most cases, the Co-Simulation FMU can be avoided by reducing the Communication Step Size. In the Model Exchange FMU, even if the solver in the generation environment is stable when executed, errors often occur in the imported execution environment solver due to differences in stability. It is therefore desirable to understand the solver environment of the execution tool and check the execution using a solver that is as close in type as possible (see 3.3.4).

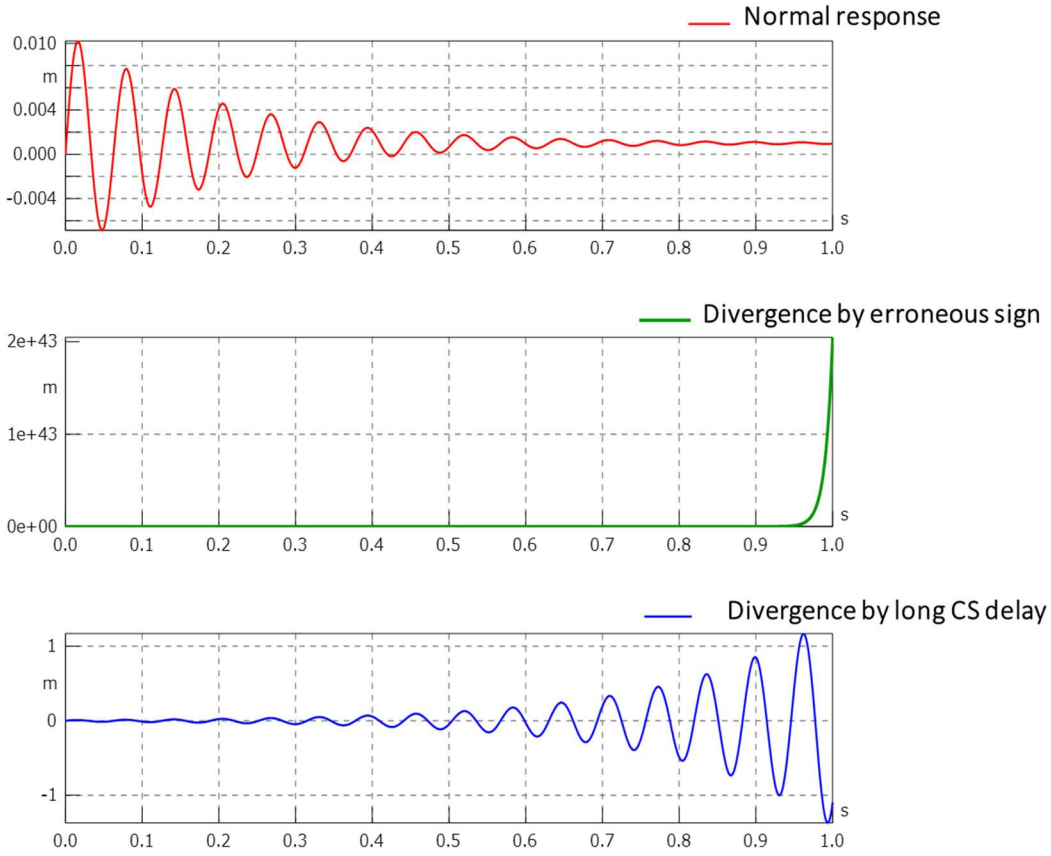


Figure4.3.8 Divergence

4.3.5.3. Non-convergence [Common].

This is when the simulation does not go ahead and the calculation is terminated.

This error often occurs when using Model Exchange FMUs with tools that use variable step solvers. Countermeasures such as decreasing the minimum time step width can be taken. In many

cases, the cause is the occurrence of an event that causes a discontinuity in the phenomenon, so it is necessary to find the part of the model itself that causes the event and correct it. For plant connections, please refer to “Examples of Highly Discontinuous Areas” as described in 4.2.3. In addition, when algebraic constraint conditions (algebraic loops) are given by non-causal tools, etc., the solver tries to solve algebraic equations in the same time, and this phenomenon occurs because the solution cannot be obtained under some conditions (see 4.3.1).

Chapter 5 References

[1] Yutaka Hirano : Toward Model Distribution by FMI, Society of Automotive Engineers of Japan, Vibration and Noise Division Committee No. 10-17 Symposium,(2017)

[2] Ministry of Economy, Trade and Industry, Strategy Council for a New Era of Automobile : Materials for the Strategy Council for a New Era of Automobile (1st meeting), (2018)
http://www.meti.go.jp/shingikai/mono_info_service/jidosha_shinjidai/pdf/001_01_00.pdf

[3] Blochwitz Torsten, Martin Otter, et al. : The Functional Mockup Interface for Tool independent Exchange of Simulation Models, Preprint of the 8th International Modelica Conference,(2011)
https://ep.liu.se/ecp/063/ecp11063.pdf

[4] Society of Automotive Engineers of Japan, Committee on Model Development and Circulation with International Standard Descriptions, Model Connection Technology Study WG : FMI Model Connection Guidelines Using Non-Causal Modeling Tools Ver. 1.0, (2015)

[5] Blochwitz Torsten, Martin Otter, et al. : The Functional Mockup Interface for Tool independent Exchange of Simulation Models, Proceedings of the 8th International Modelica Conference,(2011)

[6] Junghanns, T. Blochwitz : 10_Years_of_FMI Where are we now? Where do we go? , Keynote speech of the 2nd Japanese Modelica Conference, (2018)
https://www.modelica.org/events/modelica2018japan/presentation/10_Years_of_FMI.pdf

[7] Modelica Association : Functional Mock-up Interface Specification version 3.0

[8] https://fmi-standard.org/docs/3.0/

[9] Yutaka Hirano, Junichi Ichihara, et al. : Toward the actual using FMI in practical use cases in the Japanese automotive industry [p195-203], Program of the 2 ndJapanese Modelica Conference, (2018)

[10] Nikkei Digital Health Home Page : Modelica and FMI - CAE standards useful in the conceptual design stage (2013)
https://tech.nikkeibp.co.jp/dm/article/COLUMN/20131107/314661/

[11] FMI Home Page :https://fmi-standard.org/

[12] https://fmi-standard.org/tools/

[13] https://trac.fmi-standard.org/browser/branches/public/Test_FMUs/Compliance-Checker

[14] Study Group on the Use of Models in the Automotive Industry, Ministry of Economy, Trade and Industry :.

[15] Plant Model Interface Guidelines for Automotive Development (2017)

[16] Yosuke Ogata, Shintaro Murakami, et al : FMI, Model Exchange , A Study on Simulation Stability in Co-simulation, Society of Automotive Engineers of Japan, 2018 Spring Meeting S4-4,(2018)

[17] Jun-ichi Ichihara, Haruki Saito, et al : Co-Simulation with FMI by Multiple Modeling Tools Introduction of Model Connection Activities on (2nd Report), Society of Automotive Engineers of Japan, 2018 Spring Meeting S4-2, (2018)

[18] Yutaka Hirano, Takayuki Sekisue : Toward Model Distribution with FMI, Society of Automotive Engineers of Japan 2018 Spring Meeting S4-1, (2018)

[19] https://modelica.org/projects.html

[20] https://modelica.org/events.html

[21] https://fmi-standard.org/

[22] https://ssp-standard.org/

[23] https://dcp-standard.org/

[24] https://www.efmi-standard.org/

[25] https://www.prostep.org/en/

[26] https://www.prostep.org/en/projects/smart-systems-engineering/

[27] https://www.prostep.org/fileadmin/downloads/PSI_11_V3_SmartSE_Rec_and_Part_A-I.zip

[28] https://modelica.github.io/fmi-guides/main/fmi-guide/

[29] https://fmi-standard.org/tools/

[30] https://github.com/CATIA-Systems/FMPy

[31] https://pypi.org/project/PyFMI/

[32] Torsten Blochwitz, Andreas Junghanns,Jochen Köhler, Martin Krammer:Overview on FMI, SSP, DCP , 13th International Modelica Conference (2019)
https://fmi-standard.org/assets/literature/FMI_User_Meeting_2019/001_Overview_FMI_SSP_DCP.pdf

[33] FMI Home Page: https://fmi-standard.org/validation/

[34] MBD Promotion Center (JAMBE): Plant Model Interface Guidelines for Automotive Development:
https://www.jambe.jp/system/download