講演番号 | 173

文献番号 | 20225173

Development of C ++ library for converting a simulator to FMU

Hajime Sato [1)] **Takashi Yamashita** [1)] **Kunihiro Matsuzawa** [1)]

1) AdvanceSoft Corporation
4-3 Kanda Sugugadai, Chiyoda, Tokyo, 101-0062, Japan (E-mail: h-sato@advancesof.jp)

KEY WORDS: **vehicle development, computer aided engineering, tool,** Functional Mock-Up Interface [B2]

We have developed a C ++ interface library for converting simulators written in Fortran and C / C ++ to FMU. By rewriting the legacy program into an FMU block, it can be reused by MBD tools. The details of this library and the case studies that contributes to model exchange by FMI are introduced.

Beyond model-based development (MBD), which is currently prevalent in the automotive industry, the era of model-based design information exchange (model exchange) will arrive. However, it is difficult to develop new practical MBD models because it requires a long period of time and a great deal of human resources. If stand-alone simulation software that has been developed and used for many years can be ported to MBD models, it will be a boost to model exchange.

Two methods of building stand-alone simulator programs developed in C/C++ and Fortran languages into FMI-compliant fmu files were examined as candidates: an indirect method and a direct method. The indirect method is to use an MBD tool that has both functions of importing a program in a general-purpose programming language as a model and exporting the model to an fmu file. The indirect method is to convert once to a model in the MBD tool to obtain an fmu file. In the indirect method, the compatibility between the two functions in the MBD tool can be a problem. In the case of incompatibility, there are cases where the behavior of the fmu file is different from that of the stand-alone simulation program, or the fmu file does not work in the first place, even though there seems to be no problem in each conversion. In this case, it is necessary to verify the behavior of the intermediate model or the MBD tool in order to make the fmu file work, which is a technical problem far from the original purpose. On the other hand, the direct method has the advantage that it does not depend on the compatibility of MBD tools, but it is more expensive than the case where MBD tools are used.

To simplify the direct fmu build process, we developed fmi2++, a C++ library that directly builds stand-alone simulator programs developed in C/C++ and Fortran languages into FMI-compliant fmu files. By using fmi2++ to build existing program assets into fmu files, they can be used in MBD tools compliant with the FMI standard. The specification of the FMI standard(1) is available on the official web page, fmi-standard.org. Since there are several versions of the FMI standard, and the latest FMI 3.0 is in the draft stage as of March 2022, the following description follows the contents of the FMI 2.0.3 distribution. There are two execution formats of the FMI standard, Model Exchange and Co-Simulation, but this paper focuses on Model Exchange and introduces the contents of chapters 1 through 3 of the FMI standard. Unlike Co-Simulation, Model Exchange does not require embedding the integral solver in the fmu file, which has the advantage that it can be run as an fmu file without the need to port integral calculations from the standpoint of porting a single simulator program. The following is a list of the most common problems with this method. The fmu file is a ZIP file containing the directory structure and files defined by the standard. The standard defines the required files and their meanings for each file. fmi2++ generates two files, a model description file and a model implementation library, according to the contents of the standard. The model description is included in the fmu file as an XML format file named modelDescription.xml. The contents of the model description file are defined in the xsd file included in the FMI standard distribution. fmi2++ library has a template file with all necessary contents, so when built, the fmu file contains the result of embedding values in the template file. The system is designed to be used in the same way. The model implementation library must be saved in source code or binary format in a directory with a fixed name. Since the fmi2++ library contains default implementations of all required functions, building the library itself as is will build an empty fmu implementation that complies with the standard. Since the fmi2++ library provides templates for the above two files, library users can build an FMI-compliant fmu file simply by combining the necessary parts of the implementation of a unit simulation model with fmi2++.

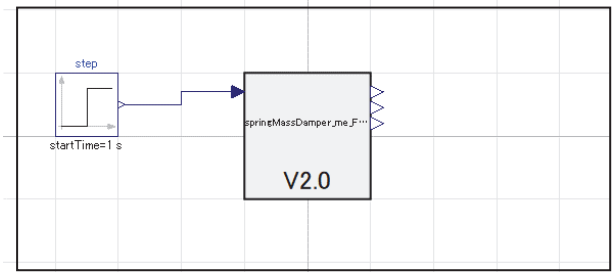

Fig.1 Loaded fmu by fmi2++

As an example of the application of the fmi2++ library, a spring-mass-damper system implemented in C++ was built as an fmu, imported and executed by OpenModelica. The input variable is the external force f, and the output variables are position, velocity, and acceleration. We verified that the built fmu outputs the correct results.

(1) https://github.com/modelica/fmi-standard/releases/download/v2.0.3/FMI-Specification-2.0.3.pdf