FMU-proxy: A Framework for Distributed Access to Functional Mock-up Units

Lars Ivar Hatledal[1] Houxiang Zhang[1] Arne Styve[2] Geir Hovland[3]

[1]Department of Ocean Operations and Civil Engineering, NTNU, Norway, {laht,hozh}@ntnu.no
[2]Department of ICT and Natural Sciences, NTNU, Norway, asty@ntnu.no
[3]Department of Engineering Sciences, UiA, Norway, geir.hovland@uia.no

Abstract

The main goal of the Functional Mock-up Interface (FMI) standard is to allow simulation models to be shared across tools. To accomplish this, FMI relies on a combination of XML-files and compiled C-code packaged in a zip archive. This archive is called an Functional Mock-up Unit (FMU) and uses the extension *.fmu*. In theory, an FMU can support multiple platforms, however this is not always the case and depends on the type of binaries the exporting tool was able to provide. Furthermore, a library providing FMI support may not be available in a particular language, and/or it may not support the whole standard. Another issue is related to the protection of Intellectual Property (IP). While an FMU is free to only provide the C-code in binary form, other resources shipped with the FMU may be unprotected.

In order to overcome these challenges, this paper presents FMU-proxy, an open-source framework for accessing FMUs across languages and platforms. This is done by wrapping one or more FMUs behind a server program supporting multiple language independent Remote Procedure Call (RPC) technologies over several network protocols. Currently, Apache Thrift (TCP/IP, HTTP), gRPC (HTTP/2) and JSON-RPC (HTTP, WebSockets, TPC/IP, ZeroMQ) are supported. Together, they allow FMUs to be invoked from virtually any language on any platform. As users don't have direct access to the FMU or the resources within it, IP is more effectively protected.

The software architecture is shown in Fig. 1 and consists of three main parts:

1. *Discovery service(s)* - Provides users with the required information needed to connect to a remote FMU. Available FMUs can be listed through a web page or be querying it through HTTP.

2. *Server(s)* - Exposes locally available FMUs through one or more RPCs, possibly over several network protocols. Optionally, publishes information to a discovery service, making the server *discoverable*.

3. *Clients* - Interacts with the remote FMU(s), and may be implementing in virtually any language.

Some features of FMU-proxy include:

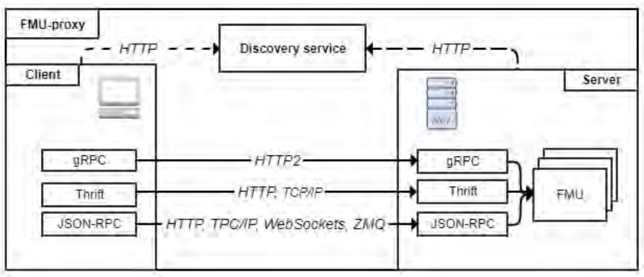

Figure 1. Software Architecture

- Brings FMI capabilities to previously unsupported languages and otherwise incompatible platforms.

- By implementing the RPC functions directly, FMI compliant models can be implemented without having to package them into FMUs.

- Allows re-use of code between software projects that requires distributed execution of FMUs, independent of implementation language.

- Enables companies to securely share FMUs. By hosting their own proxy server and directory service, neither the FMUs nor the knowledge about them leaves the company controlled servers.

Server implementations exists for C++ and the JVM, while client implementations exists for JavaScript, Python, C++ and the JVM. Due to the language independent nature of the RPC frameworks and protocols used, and especially the code-generation feature of selected RPC frameworks, client implementations in other languages require little effort.

FMU-proxy is available from GitHub at https://github.com/NTNU-IHB/FMU-proxy. Here, pre-built server executables can be obtained. Client libraries for Java are available through *maven* at https://jitpack.io/#NTNU-IHB/FMU-proxy, while client libraries for C++ will be available through *vcpkg*.

Keywords: RPC, FMI, Co-simulation, Model Exchange