Beyond FMI - Towards New Applications with Layered Standards

Christian Bertsch[1] Matthias Blesken[2] Torsten Blochwitz[3] Andreas Junghanns[4]
Pierre R. Mai[5] Benedikt Menne[2] Kevin Reim[2] Markus Süvern[2] Klaus Schuch[6]
Torsten Sommer[7] Patrick Täuber[2]

[1]Robert Bosch GmbH, Germany, `Christian.Bertsch@de.bosch.com`
[2]dSPACE GmbH, Germany, `{PTaeuber, MBlesken, MSuevern, BMenne, KReim}@dspace.de`
[3]ESI ITI, Germany, `Torsten.Blochwitz@esi-group.com`
[4]Synopsys, Germany, `Andreas.Junghanns@synopsys.com`
[5]PMSF IT Consulting, Germany, `pmai@pmsf.eu`
[6]AVL List GmbH, Austria, `klaus.schuch@avl.com`
[7]Dassault Systems, Germany `torsten.sommer@3ds.com`

Abstract

The FMI standard — just like any other standard — faces the challenge of balancing generality with enabling specific use cases. Including every domain or use-case specific extension in the core standard would significantly increase its length, making it unreadable and unimplementable. To allow for extensions of the core standard for specific use cases, the Modelica Association developed the concept of layered standards, first in the SSP standard and later in FMI.

This paper presents the concept of layered standards and describes the layered standards currently under development by the FMI Project: XCP support of FMUs, network communication, and structured variables and n-D lookup tables in FMI 3.0.

Keywords: FMI, layered standard, XCP, network communication, regular maps

1 Introduction

1.1 Motivation for Extension Mechanisms of Standards

The versions 1.0 and 2.0 of the FMI standard (Blochwitz 2011) (Blochwitz 2012) already contain many optional features, and FMI 3.0 (Junghanns 2021) has increased their number even more. If additional optional features were continually added to address specific use cases and usage domains, the standard would significantly increase in length and become unreadable and unimplementable.

1.2 Requirements

The layered standards approach is based on a hierarchical structure of standards that meet the following requirements:

- The core standard remains generic to ensure broad usage and tool support.
- The extensions (layered standards) depend on the core standard, but not vice versa. This allows for flexible extension not only by the FMI Project but also by other organizations, independent of the release cycle of the FMI Standard.

1.3 Extension Mechanisms for Standards

A layered standard allows specific, new use cases to be handled, without violating the core standard, but rather building on it. It is realized by using extension points contained in the base standard, that were either already intended for extension via layered standards, or can be used, even if not originally so intended.

Some standards are generally intended to be extended by layered standards, providing just frameworks for this extension: For example, URIs defined in RFC-3986[1] (Berners-Lee, Fielding, and Masinter 2005) gain their expressive power through the extension via scheme definitions, like the file or http schemes.

Other standards provide extension via layered standards on top of a core standard that already provides a robust set of functionality. The HTTP standard RFC-2616 (Nielsen et al. 1999) provides the core functionality behind the web, while allowing among others for extension via additional headers, which have been used to provide, e.g., RFC-2965 (Montulli and Kristol 2000) for HTTP State Management Mechanism — Cookies, or RFC-6797 (Hodges, Jackson, and Barth 2012) for HTTP Strict Transport Security. Similarly, the SMTP standard RFC-2821 (Klensin 2001) provides the core functionality behind email, allowing extension via additional options or services, e.g. RFC-3207 (Hoffman 2002) for STARTTLS.

And some standards can be extended via layered standards even though the base standard only contains very limited extension points: FMI 2.0, for example, provided annotations as a user-defined mechanism in the core XML, and did not prohibit additional files to appear in the FMU archive, thus allowing extension, while not necessarily

[1]Here and in following references to IETF standards, not always the newest incarnation of the RFC is cited, but rather the relevant versions from a historical perspective of layered standard development.

having layered standards in mind. The OSI Sensor Model Packaging (OSMP) (ASAM OSI Project 2022) specification is an example of a layered standard that was layered on top of FMI 2.0.

1.4 The Concept of Layered Standards in the Modelica Association

For Modelica Association standards, the concept of layered standards was first introduced for the System Structure and Parameterization (SSP) Standard version 1.0 (MAP SSP 2019)

They can be defined and released

- by third parties, completely independent from the Modelica Association Project (MAP),
- by third parties that are endorsed by the MAP, or
- by the MAP project itself, making them a MAP layered standard.

1.5 The Concept of Layered Standards to the FMI Standard

In FMI, the concept of layered standards was introduced in the version 3.0 (MAP FMI 2022b), and later backported to FMI 2.0.4 (MAP FMI 2022a). The following specific provisions for layered standards were made, see Figure 1:

- The ZIP structure now contains an "extra/" folder where additional files can be placed without disturbing the rest of the FMI mechanisms.
- The XML schema allows extensions to elements to add further information, via Annotation elements.
- For both of these mechanisms the standard provides suggested rules on naming using reverse domain notation to ensure that separately defined extensions do not clash.
- New values for *matchingRule* or *terminalKind* for Terminals can be defined.
- Potentially, layered standards could add new functions to the API of the FMU, however no rules to avoid name clashes have yet been devised as part of FMI 3.0.
- FMI 3.0.1 introduces a recommendation for a standardized fmi-ls-manifest.xml file that allows importing tools to detect which layered standards are supported by the FMU (and which version). This way, the additional capabilities of the FMU can be used more easily and a list of supported layered standards can be displayed to the user.

For FMI 3.0, the standard specifies that an FMU supporting a layered standard on top of FMI 3.0 must at the same time still be a valid FMI 3.0 FMU. This requirement puts certain constraints on a layered standard:

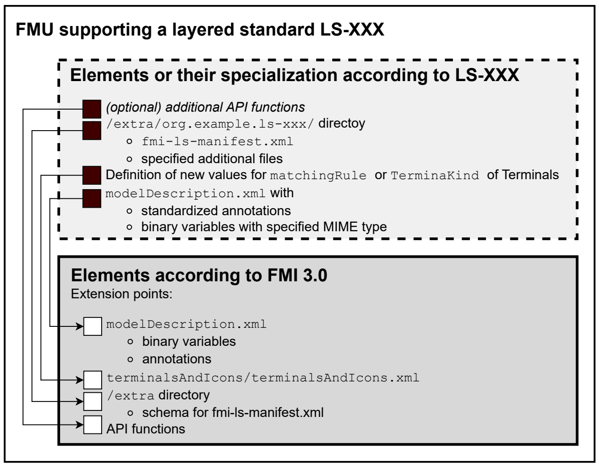
FMU supporting a layered standard LS-XXX
Elements or their specialization according to LS-XXX
(optional) additional API functions
/extra/org.example.ls-xxx/ directoy
fmi-ls-manifest.xml
specified additional files
Definition of new values for matchingRule or TerminaKind of Terminals
modelDescription.xml with
standardized annotations
binary variables with specified MIME type
Elements according to FMI 3.0
Extension points:
modelDescription.xml
binary variables
annotations
terminalsAndIcons/terminalsAndIcons.xml
/extra directory
schema for fmi-ls-manifest.xml
API functions

Figure 1. An FMU supporting a layered standard LS-XXX using the extension mechanisms of FMI 3.0.

- The layered standard can add optional features to the FMU, like additional files inside the FMU's zip file, or it can extend XML files where the base standard schema allows it.
- On the other hand, the layered standard can place additional restrictions on XML elements (e.g. only allow the use of certain Variable types), or mandate an optional FMI 3.0 feature to be required.

In FMI 3.0, some new features that are "orthogonal" to the core FMI functionality (namely *TerminalsAndIcons* and *BuildConfiguration*), were already developed with some features of layered standards in mind, e.g., by having their own XML files. However, as they are of most general interest and considered very important, it was decided to include them in the core FMI standard. This has the benefit of possibly being supported by many tools, but the drawback that updates of these features are limited to the release cycle of the core FMI 3.0 standard.

A layered standard could also extend beyond the core standard, e.g., by introducing additional API functions or side channels, or by removing limitations on the calling sequence of API functions of the FMI state machine of a certain FMI kind. An example would be setting states of a Co-Simulation FMU after initialization, e.g., in order to realize nonlinear Kalman filters.

In the next sections we will give examples of layered standards that are currently in development by the FMI Project and other institutions or companies.

2 Current Development of Layered Standards for FMI by the FMI Project

Currently three layered standards are in development by the FMI Project. They comprise extensions to FMI that are of general interest. Additionally, they demonstrate and standardize how FMI 3.0 and its mechanism can be used for new important application domains. Furthermore, the provision of these layered standards by the FMI Project will promote the concept and validate the extension mechanisms of FMI, potentially leading to further improvements.

2.1 FMI Layered Standard for XCP (LS-XCP)

2.1.1 Motivation

XCP (Universal Measurement and Calibration Protocol (ASAM e.V. 2017)) is a standardized protocol used in the automotive industry to measure variables and adapt control parameters inside an electronic control unit (ECU) through buses like CAN. When wrapping a virtual ECU (vECU) into an FMU, supporting XCP-based measurement and calibration is required for many simulation use cases. Before this standardization effort, already several tools implemented proprietary XCP support for FMUs. As these are incompatible to each other, the need for standardization became apparent.

2.1.2 Approach

The main idea is to ship an A2L file (see (ASAM e.V. 2018) ASAP2) in a standardized location inside the FMU and to describe the capabilities of the FMU w.r.t. the XCP protocol.

The layered standard describes two alternative implementations depending on the use case and data availability (MAP FMI 2023b):

- The FMU implements an XCP slave which provides access to measurement and calibration variables of the vECU and handles the communication protocol with the XCP master in the MCD tool. The necessary information for an MCD tool is given in a description file which follows the ASAM MCD-2 MC standard (aka A2L, also ASAP2) and customarily carries the file extension .a2l. Figure 2 shows a typical design with an XCP service contained in the FMU.
- An external XCP slave implementation accesses the memory of the vECU to expose the XCP protocol to the MCD tool. In this case, the A2L file is still shipped with the FMU but the importer needs to provide the XCP slave implementation. *fmi3IntermediateUpdateCallback* calls or the Clocks mechanism could be used to synchronize DAQ lists. Figure 3 illustrates a typical design utilizing an external XCP service.

The following extension mechanisms to FMI are used:

- The **"extra/" directory** is used to provide additional files: An fmi-ls-manifest.xml provides information about the capabilities of the FMU and an A2L file for each supported platform describes the memory layout.

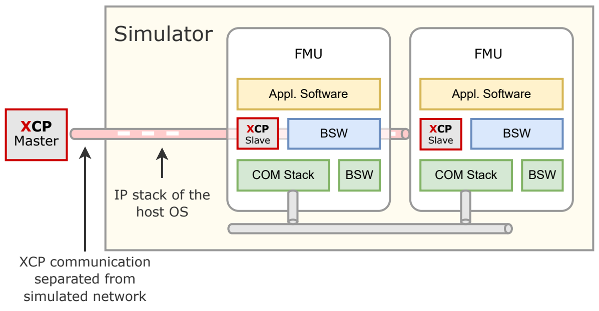

Figure 2. Direct communication of XCP master and XCP slave via the IP stack of the host OS.

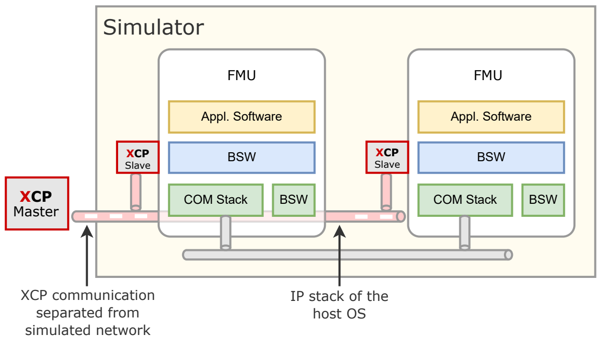

Figure 3. Communication of XCP master and external XCP slave via the IP stack of the host OS.

- A set of **(structural) parameters** for the configuration of the XCP service is introduced, e.g., to change the TCP port the XCP service is listening on. FMUs that provide an XCP service should also provide these parameters that follow a specified naming convention.
- It is specified when the XCP service should be started and stopped in order to have an early access to the XCP calibration parameters.

This layered standard will have no effect on the FMU interface, nor the C-API behavior. While measurement of FMU internal variables does not have a numeric effect on the FMU, so called calibration does. Calibration is the tuning of FMU internal parameters. Such changes will affect the numeric behavior of the FMU. If the FMU contains controller code, numeric stability or energy preservation laws are of lesser concern. On the other hand, plant models offering XCP access for parameter calibration may introduce surprising numerical effects in solvers that might require proper handling, like resetting solvers with every XCP write action.

Therefore, it is necessary to synchronize XCP variable access (read and write) with the state of the FMU.

2.1.3 Status and Outlook

A proposal for such a layered standard is being developed on GitHub (https://github.com/modelica/fmi-ls-xcp).

With small limitations, this layered standard can also be used with FMI 2.0.4.

Most mechanisms have been agreed upon and first prototype implementations have been realized. The cross-check of generated FMUs has been started as of writing this paper by companies such as dSPACE GmbH, ETAS GmbH, PMSF IT Consulting and SYNOPSYS.

2.2 FMI Layered Standard for Network Communication (LS-BUS)

2.2.1 Motivation

Simulation of modern automotive systems requires network communication between vECUs. Traditional simulations according to FMI 2.0 or other formats deal either with continuous signals or with non-standardized and proprietary solutions for exchanging network messages. In practice, such proprietary solutions often lead to interoperability issues when creating simulation systems with vECUs provided by different suppliers.

To minimize the resulting effort, the FMI layered standard for Network Communication (MAP FMI 2023a) was introduced. By using FMI 3.0 core standard features such as Co-Simulation, Clocks, clocked variables and terminals, the layered standard specifies a common bus interface and defines how to emulate a transport layer for several bus types in detail. While this layered standard has been initiated with automotive use cases in mind (with automotive network technologies such as CAN, LIN, FlexRay, CAN FD, CAN XL and Ethernet), the used concepts are kept general. This way the layered standard could also be applied to other domains such as industrial automation.

2.2.2 Approach

General concepts: The proposal on GitHub (https://github.com/modelica/fmi-ls-bus) provides two abstraction layers for different use cases:

- Physical signal abstraction ("high cut"): Use individual, clocked signal variables to transport logical, unit-based values between vECUs, ignoring transport layer-specific properties. The layered standard for this abstraction basically defines how bus signals have to be described in the model description file. It should be noted that creating FMUs with this abstraction layer typically requires a network description.

- Network abstraction ("low cut"): This abstraction allows the implementation of virtual bus drivers within FMUs on the level of the hardware abstraction layer. It uses clocked binary variables to exchange bus operations between FMUs based on a lightweight protocol defined by the layered standard. Bus operations are used to transmit bus messages as well as bus events like acknowledge or error events. This enables both ideal and more realistic bus simulations, depending on the capabilities of the FMU and importer. These capabilities can include timing, arbitration, error handling, status monitoring and other effects.

It is assumed that FMUs will provide only one of these abstraction layers although it would be technically feasible to support both.

System composition: In the simplest use case, the importer does not need to provide specific bus semantics of certain variables of an FMU; it simply forwards variable values between two FMUs according to the FMI standard. Such a simulation is shown in Figure 4. In this case, however, the bus simulation is idealized, i.e., effects like transmission time, arbitration or any other bus-specific behavior are not taken into account.

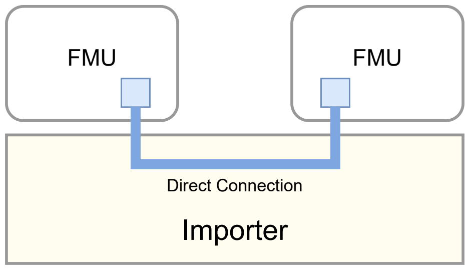

Figure 4. Direct communication of two FMUs, e.g., vECUs, on a common importer.

Only if more than two FMUs should be connected to a single network or a detailed bus simulation is desired in the "low cut" case, a dedicated bus simulation component is required. This bus simulation component then forwards bus operations between multiple senders and receivers and emulates the bus behavior. This type of communication allows the simulation of complex bus features, such as arbitration, the simulation of timing or the injection of bus failures. The supported bus features cannot be specified explicitly, but refer to a specific implementation of the bus simulation component and depend on the requirements of the bus simulation. The implementation of the bus simulation component could be done either by special capabilities of the importer, or by the provision of a Bus Simulation FMU. See Figure 5 for the simulation with such a Bus Simulation FMU. Here, the importer does not require any special features for bus simulation.

It should be explicitly noted that the FMUs integrated in the respective use case do not necessarily have to be different, which means that the same FMU can be integrated across all system compositions. The interface of the FMU to the importer is always the same, but a different subset of the features may be used.

Used FMI 3.0 concepts: The following FMI 3.0 concepts are used by this layered standard:

- **FMI for Co-Simulation**: The layered standard is currently being described in the context of the Co-

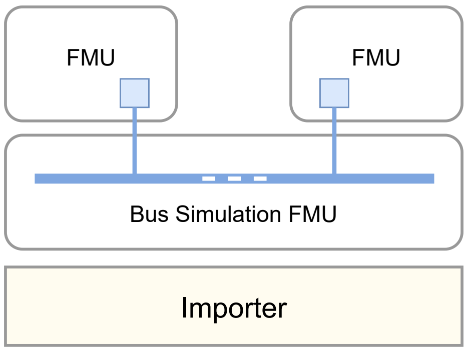

Figure 5. Realization of complex bus simulations by using a Bus Simulation FMU in addition to the communicating FMUs (e.g. vECUs).

Simulation mode, as this is best suited for the identified use cases.

- Hierarchical **Terminals** are used for grouping of variables. For the "high cut", the signals are nested hierarchically in several Terminals. The signals, together with their associated Clocks, are grouped based on the PDU and frame to which they belong. In the final aggregation, all frame terminals are combined in a Bus Terminal depending on their specific bus type. Within the "low cut" two binary variables and two Clocks are coupled into a common bus interface aggregated by a Terminal.
- **Clocks** and **clocked variables** are used to synchronize the exchange of network data among all FMUs in the simulated network.
- **Binary variables** with a dedicated MIME-type are used for the "low cut" to define the data exchanged for a specific bus type such as CAN.
- In the "high cut" variant, the **"extra/" directory** can optionally be used to ship network description files such as ARXML, DBC, LDF, Fibex, or others.

Timing aspects: Clocks are used to inform the importer about new signal values ("high cut") or binary bus operations ("low cut") to be exchanged. FMUs that want to send out those values exactly in time can use time-based Clocks and should support a variable communication step size. Periodic (fixed-time) FMUs are also supported, but in this case, multiple sends might fall into one communication step. While "high cut" signal variables will miss all but the last value sent, in the "low cut" case, all bus operations will be buffered in the binary variables.

Example for a "low cut" network interface: Figure 6 shows an example FMU with two binary variables `Rx_Data` and `Tx_Data`, and two Clock variables `Rx_Clock` and `Tx_Clock` that are aggregated to a `Bus Terminal`. Independent of the bus feature `Input` and `Output` represent exemplary additional FMI variables of the example FMU.

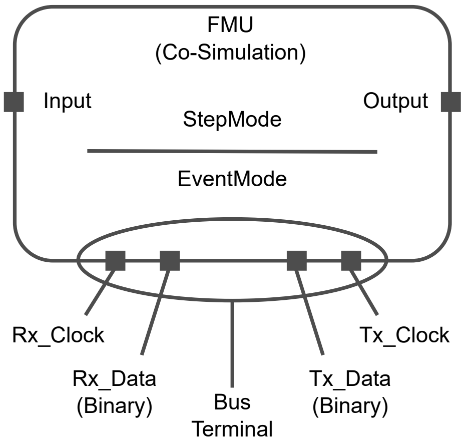

Figure 6. Example for a "low cut" Bus Terminal.

Based on this generic bus interface, an FMU can either be connected to a bus simulation component or directly to another FMU via an importer.

2.2.3 Status and Outlook

The first iteration primarily focuses on the basic concepts of the layered standard and the support for simulating CAN, CAN FD and CAN XL buses. Adding support for Ethernet, LIN and FlexRay is planned for upcoming iterations. Other bus systems from various domains can be specified in the future as required.

2.3 FMI Layered Standard for Structuring of Variables, Maps and Curves (LS-Struct)

2.3.1 Motivation

Grouping of variables, especially parameters: For many use cases, grouping of several parameters is very important. In FMI 1.0, 2.0 and 3.0, this can be partially realized with the "structured naming convention" (MAP FMI 2022a) which can be used, e.g., to represent array variables and/or hierarchical structures of variable. However, FMI 1.0 and 2.0 define scalar variables only, so for arrays this is not efficient, and generally the "structured naming convention" is not flexible enough for many applications.

Realizing n-D lookup tables: A special case of grouping variables is the representation of n-D lookup tables. In the context of the layered standard, an n-D lookup table is a sampled representation of a function of n input variables $y = F(x_1, x_2, x_3, \ldots, x_n)$ sampled on the vertices of a rectlinear grid. Such an n-D lookup table could be also called a map from the n-dimensional domain to a codomain. In (ASAM e.V. 2018) a 1-D lookup table is called `CURVE`, a 2-D lookup table is called `MAP` (see Figure 7), and a 3-D lookup table is called `CUBOID`. 4-D and 5-D lookup tables are called `CUBE_4` and `CUBE_5`, respectively. Higher dimensional lookup tables are not defined in (ASAM e.V. 2018).

FMI 3.0 introduces array variables (optionally with sizes that can be changed via structural parameters). However, an n-D lookup table is more than just one array parameter but it can be represented as a combination of several array variables.

Some tools started to implement proprietary solutions to represent lookup tables with FMI, e.g., through vendor-specific annotations or naming conventions, and the need for standardization became apparent.

2.3.2 Approach

Grouping of variables: FMI 3.0 includes a mechanism for grouping variables through the definition of Terminals ((MAP FMI 2022b). This concept was originally designed to allow for the connection of a group of variables, typically inputs or outputs, but also for parameter propagation. However, the semantics of Terminals are kept general, and thus they can be used just for grouping of variables. In this case connecting these "terminals" isn't the main focus.

n-D lookup tables: Their representation is a specialization of the more general parameter grouping concept. An exposed n-D lookup table within an FMU contains the following information:

- Domains: For each of the n dimensions of the lookup table an array variable (typically a parameter or a constant) with the sampling points (along this dimension) of the lookup-table must be referenced.
- Codomain: The sampled function values are stored in this references n-dimensional array.
- Optionally, for each dimension a variable (typically an input or a local variable) can be referenced that represent the current operating point (along this dimension)
- Optionally, additional variables containing related information can be referenced. (e.g., the interpolation algorithm in between the sampling points)

It is intended to use the concept of Terminals for this purpose by defining specific values for the terminalKind, variableKind and matchingRule attributes. Note that, e.g., CombiTable1D or CombiTable2D blocks of the Modelica standard library used within an FMU can be exposed with this approach as well.

2.3.3 Status and Outlook

At the time of writing this paper, this layered standard is in an early stage. Its development can be followed on GitHub (MAP FMI 2023c).

3 Layered Standards by other Organizations and Companies

The concept of layered standards is especially suitable for specific extensions to FMI that will not become part of the FMI Core standard and that are not developed by the

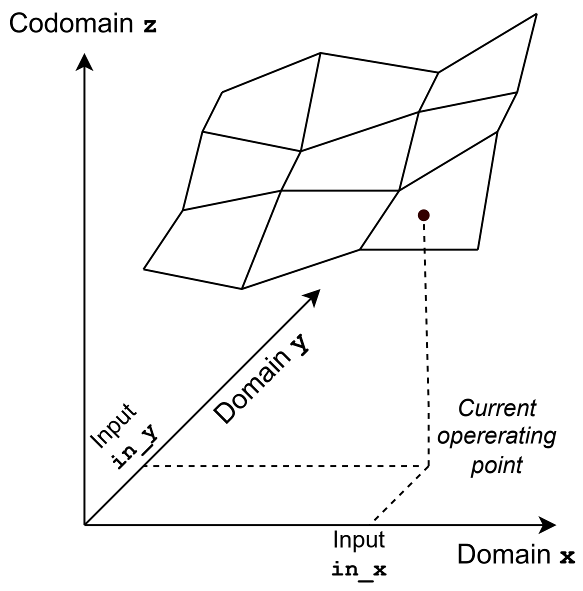

Figure 7. A 2-D lookup table with the current operating point

FMI Project. Thus, companies and other organizations are encouraged to develop their own layered standards.

First drafts for such layered standards developed have been already created, e.g., for exchangeable binary codecs and the realization of binary variables in FMI 2.0 via string variables (Bosch 2023).

4 Summary and Outlook

The FMI 3.0 standard is currently in a phase of rapid adoption by tool vendors. However, many existing use cases for FMI-based simulation can already be handled with FMI 2.0. The switch to FMI 3.0 will be driven by new use cases with new requirements, such as detailed simulations of vECUs. For their realization with FMI, additional information and extended capabilities for certain domains are beneficial and necessary. In this paper we presented the concept of layered standards to FMI. We expect their broad adoption in the near future, especially after the publication of the layered standard examples currently being created by the FMI Project. During their development the concept of layered standards and the foreseen extension mechanisms in FMI 3.0 have already proven very effective.

Acknowledgements

We would like to thank all contributors to the Modelica Association standards, particularly FMI and SSP. The concept of layered standards for FMI is based on their valuable work and contributions.

References

ASAM e.V. (2017-11). *ASAM MCD-1 XCP v1.5.0.* URL: https://www.asam.net/standards/detail/mcd-1-xcp/.

ASAM e.V. (2018-03). *ASAM MCD-2 MC (aka ASAP2) v1.7.1.* URL: https://www.asam.net/standards/detail/mcd-2-mc/.

ASAM OSI Project (2022-07). *ASAM OSMP (Open Simulation Model Packaging) V1.3.0, part of ASAM OSI (Open Simulation Interface) v3.5.0*. URL: https://opensimulationinterface.github.io/osi-documentation/#_osi_sensor_model_packaging.

Berners-Lee, Tim, Roy T. Fielding, and Larry M Masinter (2005-01). *Uniform Resource Identifier (URI): Generic Syntax*. RFC 3986. DOI: 10.17487/RFC3986. URL: https://www.rfc-editor.org/info/rfc3986.

Blochwitz, Torsten et al. (2011). "The Functional Mockup Interface for Tool independent Exchange of Simulation". In: *8th International Modelica Conference*. URL: http://www.ep.liu.se/ecp/063/013/ecp11063013.pdf.

Blochwitz, Torsten et al. (2012). "Functional Mockup Interface 2.0: The Standard for Tool Independent Exchange of Simulation Models". In: *8th International Modelica Conference*. URL: https://lup.lub.lu.se/search/ws/files/5428900/2972293.pdf.

Bosch (2023-05). *FMI Layerd Standard Drafts by Robert Bosch GmbH*. URL: https://github.com/boschglobal/dse.standards/tree/main/modelica.

Hodges, Jeff, Collin Jackson, and Adam Barth (2012-11). *HTTP Strict Transport Security (HSTS)*. RFC 6797. DOI: 10.17487/RFC6797. URL: https://www.rfc-editor.org/info/rfc6797.

Hoffman, Paul E. (2002-02). *SMTP Service Extension for Secure SMTP over Transport Layer Security*. RFC 3207. DOI: 10.17487/RFC3207. URL: https://www.rfc-editor.org/info/rfc3207.

Junghanns, Andreas et al. (2021). "The Functional Mock-up Interface3.0 - New Features Enabling New Applications". In: *14th International Modelica Conference*. URL: https://doi.org/10.3384/ecp2118117.

Klensin, Dr. John C. (2001-04). *Simple Mail Transfer Protocol*. RFC 2821. DOI: 10.17487/RFC2821. URL: https://www.rfc-editor.org/info/rfc2821.

MAP FMI (2022a-05). *Functional Mock-up Interface (FMI) Standard 2.0.4*. URL: https://github.com/modelica/fmi-standard/releases/download/v2.0.4/FMI-Specification-2.0.4.pdf.

MAP FMI (2022b-05). *Functional Mock-up Interface (FMI) Standard 3.0*. URL: https://fmi-standard.org/docs/3.0/.

MAP FMI (2023a). *Layerd Standard for Bus (unreleased)*. URL: https://modelica.github.io/fmi-ls-bus/main/.

MAP FMI (2023b). *Layered Standard for XCP (unreleased)*. URL: https://modelica.github.io/fmi-ls-xcp/main/.

MAP FMI (2023c). *Layered Standard Structuring of Data (unreleased)*. URL: https://modelica.github.io/fmi-ls-struct/main/.

MAP SSP (2019-03). *System Structure and Parameterization (SSP) Standard 1.0*. URL: https://ssp-standard.org/publications/SSP10/SystemStructureAndParameterization10.pdf.

Montulli, Lou and David M. Kristol (2000-10). *HTTP State Management Mechanism*. RFC 2965. DOI: 10.17487/RFC2965. URL: https://www.rfc-editor.org/info/rfc2965.

Nielsen, Henrik et al. (1999-06). *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616. DOI: 10.17487/RFC2616. URL: https://www.rfc-editor.org/info/rfc2616.