FMI Layered Standard for Network Communication: Applications in Networked ECU Development

Christian Bertsch[1] Kahramon Jumayev[2] Andreas Junghanns[3] Pierre R. Mai[4] Benedikt Menne[5] Masoud Najafi[6] Tim Pfitzer[1] Jan Ribbe[3] Klaus Schuch[7] Markus Süvern[5] Patrick Täuber[5]

[1]Robert Bosch GmbH, Germany, {Christian.Bertsch, Tim.Pfitzer}@de.bosch.com
[2]Akkodis Germany Consulting GmbH, Germany, Kahramon.Jumayev@akkodis.com
[3]Synopsys, Germany, {Andreas.Junghanns, Jan.Ribbe}@synopsys.com
[4]PMSF IT Consulting, Germany, pmai@pmsf.eu
[5]dSPACE GmbH, Germany, {BMenne, MSuevern, PTaeuber}@dspace.de
[6]Altair, France, Masoud@altair.com
[7]AVL List GmbH, Austria, Klaus.Schuch@avl.com

Abstract

This paper introduces the FMI 3.0 Layered Standard for Network Communication (FMI-LS-BUS), an extension of the Functional Mock-up Interface 3.0 (FMI 3.0) standard designed to address interoperability challenges in simulating distributed, networked systems, particularly in automotive applications. By leveraging FMI 3.0 features such as clocks, clocked variables, and hierarchical terminals, the standard defines two complementary abstraction layers:

Physical Signal Abstraction (High-Cut): Representing physical signal values as clocked variables.

Network Abstraction (Low-Cut): Emulates hardware-level bus protocols (e.g., CAN, Ethernet) using FMI 3.0's clocked binary variables.

Aligning with the V-model development process, we demonstrate how these layers address distinct challenges in different design phases: High-Cut supports requirements engineering and functional testing by simplifying signal exchange during Virtual Electronic Control Unit (vECU) integration. Low-Cut enables later phases of the design validation by replicating network timing and protocol specific properties, such as error handling.

The standard's applicability currently focuses on automotive use cases (e.g., CAN, CAN FD, CAN XL, Ethernet, FlexRay, LIN) but can be extended to industrial automation and IoT, facilitated by its domain-agnostic structure.

Keywords: FMI, layered standard, network communication, bus, V-model, High-Cut, Low-Cut, Physical Signal Abstraction, Network Abstraction, FMU, vECU

1 Motivation

The development of complex cyber-physical systems in the modern automotive industry pushes the boundaries of engineering complexity, demanding new standardized methods, virtual validation, and simulation-based testing practices (Rausch 2022). As vehicles become increasingly software-driven, the need for virtual validation is particularly evident since advanced vehicle functions like Advanced Driver Assistance Systems (ADAS) are extensive and resource-intensive (dSPACE 2025). Real-world testing and hardware-in-the-loop (HiL) systems alone cannot adequately cover all test scenarios. Existing development methodologies, such as the Vehicle Development V-model (Müller-Ettrich 1998), provide a well-established framework for system design and verification. However, these methods must be complemented by adapting processes, methods and tools to the new requirements. Therefore, new approaches are being developed to validate vehicle functions, ECU software components and network communication. Strengthening support for communication protocols and system interactions is essential to ensure performance, reliability, and functional safety in the development of modern vehicles.

Furthermore, the development of these complex, cyber-physical systems and their components is highly distributed, with development of critical functionalities spread across multiple suppliers and engineering teams, introducing significant challenges in requirements specification, system integration and validation (Lawrenz 2013). A single vehicle's electronic architecture may involve numerous Original Equipment Manufacturers (OEMs) and a diverse network of tier suppliers, such as Tier 1 and Tier 2, each with different roles in the development process of ECUs. These ECUs rely on various automotive bus protocols, such as CAN, LIN, FlexRay, and Automotive Ethernet (Rausch 2022), to exchange data and coordinate system behavior. Each supplier may employ different simulation environments, modeling methodologies, and protocol implementations, making it difficult to achieve a unified, end-to-end system simulation. Ensuring seamless interoperability between simulation tools and across different teams and companies has become essential. Engineers encounter substantial challenges in:

- Verifying end-to-end system behavior across diverse

ECUs and networks

- Ensuring consistent communication between heterogeneous components and network technologies
- Validating system performance at different stages of development
- Reducing time-to-market while managing escalating development costs
- Maintaining comprehensive system reliability, safety, and compliance with standards such as ISO 26262 (ISO 2011)

A key enabler for addressing these interoperability challenges is the Functional Mock-up Interface (FMI) (Blochwitz 2011) (Blochwitz 2012) (Junghanns 2021), which is a non-proprietary, free standard for model exchange and co-simulation, allowing simulation artifacts from different sources and different simulation tools to interact seamlessly instead of existing proprietary and incompatible solutions from different software manufacturers. By integrating bus simulations into an FMI-based workflow, engineers can achieve a more modular, reusable, and interoperable simulation environment. Since the FMI 3.0 standard does not cover bus or network communication, the Layered Standard (Bertsch 2023) FMI-LS-BUS extends the capabilities of FMI 3.0 by standardizing network simulations using only features of FMI 3.0. This enables the validation of communication protocols at both the frame level (Low-Cut simulation) and the signal level (High-Cut simulation) while ensuring consistency across different tool chains (dSPACE 2024).

2 Concepts

This section delves into the general concept of the FMI-LS-BUS. First, the existing system compositions are presented. Second, the two specified abstraction layers are explained and differentiated.

The FMI 3.0 standard itself allows interoperability between the tool chains of different software manufacturers and reflects a fundamental need and the necessity of users who have to develop and secure software in a distributed manner. For example, across company boundaries in an OEM and supplier relationship or across departmental boundaries in large companies. Based on this stable foundation, the FMI-LS-BUS (MAP FMI 2025a) specifies an extension of the FMI 3.0 standard that enables the simulation of network communication. This extension supports various bus systems such as CAN, CAN FD, CAN XL, FlexRay, Ethernet and LIN and uses native FMI 3.0 features such as binary variables, clocks and terminals. With two different abstraction layers, the High-Cut and the Low-Cut, it also offers users the necessary flexibility and variety to simulate a network of controllers at their current phase of development.

- 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.
- Network Abstraction (Low-Cut): This abstraction allows for the implementation of virtual bus drivers within FMUs on the level of a 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 both sides of the API: The FMU and the importer. These capabilities can include timing, arbitration, error handling, status monitoring and other effects.

Since the FMI-LS-BUS is constrained to the description of the FMU interface, it remains neutral and agnostic to the actual bus communication process. For supported bus technologies, convenient templates are available to keep the integration effort low.

2.1 System Compositions

The FMI-LS-BUS supports three different ways of system compositions for integration into a specific simulation platform. Each individual system composition offers different advantages that are relevant depending on the user's concrete needs. The integrated FMUs across the different system compositions are the same, so that the same FMU can be integrated in all three communication architectures.

2.1.1 Direct Connection

The first option is to use a common FMU importer. Within this configuration, the FMU importer does not require any special features for simulating buses, apart from supporting FMI variables, clocks and terminals. Figure 1 illustrates the direct communication of two FMUs.

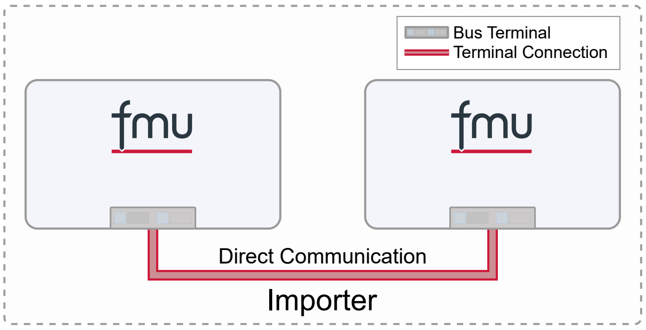


Figure 1. The first system composition allows the connection of exactly two FMUs using a common FMU importer to simulate bus systems.

Direct bus communication is limited to exactly two FMUs. The idealized bus simulation does not consider bus-specific properties like the simulation of transmission timing or arbitration and, therefore, differs from physical networks. The main points resulting from this are as follows:

- Network congestion/bandwidth: The respective bus transmits the specific messages at an infinite speed. Baud rate considerations are not taken into account.
- Arbitration of network frames: All frames are transmitted at the same time without delay, which means that network-specific priority rules are not considered.
- Bus-specific protocol functions: All frames are transmitted at the same time without any delay. Protocol-specific functionalities, such as the acknowledgment of frames in CAN or any other network protocol, are ignored.
- Incoming buffer overflow: The Network FMU will receive all frames, regardless of buffer size or other hardware limitations and would need to handle those internally.

2.1.2 Composition with a dedicated Bus Simulation FMU

To simulate more realistic network characteristics, a bus simulation must usually be integrated, which simulates bus-specific properties like arbitration or the actual transmission time. One option is to implement this behavior in a separate FMU, the so-called Bus Simulation FMU. This FMU is responsible for simulating the respective bus behavior depending on the specified bus type (e.g., CAN, Ethernet, or any other bus) and simulates transmission times and failures of bus messages. Also it realizes the connections between the different FMUs that want to exchange network data (Network FMUs). The primary idea behind this concept is: All Network FMUs send their bus messages to the Bus Simulation FMU, which can confirm, delay or reject, and forward them. This approach allows for detailed bus simulations using common FMU importers. Figure 2 shows two Network FMUs connected to a Bus Simulation FMU; a total of three FMUs are running on an FMU importer.

This type of communication allows for the simulation of all required bus features:

- Transmission timing: An essential and one of the important features of bus-specific simulations is the consideration of transmission times for specific transmissions.
- n:m Bus Communication: The Bus Simulation FMU can provide the appropriate number of inputs and outputs to connect all Network FMUs within the complete bus network.

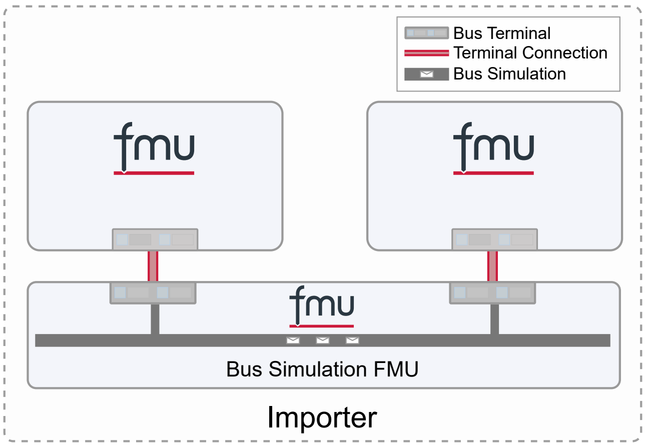


Figure 2. Bus simulation using a dedicated Bus Simulation FMU allows the simulation of n:m bus systems and very detailed simulations based on common FMU importers.

- Failure injection: Both the FMI-LS-BUS and this system composition in which all network data is routed via central Bus Simulation FMU(s) allow the possible targeted injection of transmission errors for simplified testability.
- Arbitration of network frames: Depending on the bus type and the specific arbitration requirements, appropriate implementations can be carried out via the central Bus Simulation FMU.

2.1.3 Importer with integrated Bus Simulation

Within this system composition, the functionality of the Bus Simulation FMU is integrated into the importer or simulator, so it allows the integration of the FMI-LS-BUS into an already existing manufacturer-specific simulator. These simulators may implement network communication with proprietary interfaces.

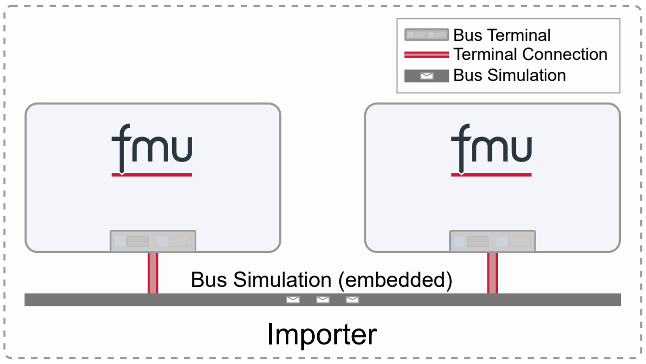


Figure 3. Bus simulation by using an importer with integrated Bus Simulation allows proprietary simulators to integrate FMI-LS-BUS specific FMUs.

This kind of system composition supports the same bus features as the setup with a dedicated Bus Simulation FMU, but with importer-specific benefits or constraints.

Figure 3 shows two Network FMUs integrated by an importer that supports the FMI-LS-BUS standard without the need for an additional Bus Simulation FMU.

2.2 Physical Signal Abstraction ("High-Cut")

Physical Signal Abstraction, or shortly called High-Cut, allows the exchange of unit-based physical signals between FMUs. Variables representing these physical signals are clocked to reflect the bus timing aspects of the information flow. The activation of these clocks always indicate that new network data is available at a given point of time.

As shown in figure 4, the core idea of the High-Cut consists of the bus signals, which are modeled as FMI 3.0 variables. These FMI variables are grouped into Protocol Data Unit (PDU) terminals, which are then grouped into frame terminals to declare their structure according to the network description.

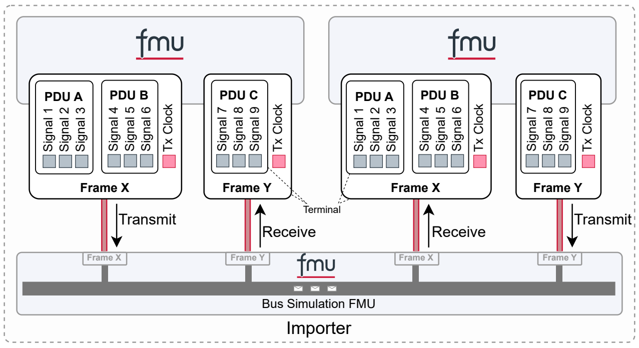


Figure 4. The figure shows an example architecture of a Physical Signal Abstraction. The signals (Signal 1…9) are modeled as clocked FMU variables of a specific type. The corresponding signals are structurally combined using PDUs, which in turn are assigned to frames. The PDU and frame structuring is done via terminals (PDU A, B, C and Frame X, Y).[1]

This approach of loose coupling between the physical signals and the lightweight PDU/frame grouping communication structure allows for easy creation of such High-Cut-based Network FMUs without the need for complex communication stacks. These configurations make them perfect for the rapid creation or prototyping in an agile, iterative development of the final communication structure. The High-Cut is therefore primarily suitable for early development phases, where its data-based simplicity offers efficiency advantages, rather than being disrupted by the fact that it lacks complete accuracy in simulating specific bus systems.

2.3 Network Abstraction ("Low-Cut")

The Network Abstraction, or shortly called "Low-Cut", allows the exchange of bus operations between FMUs over binary clocked variables. Bus operations are defined for different bus types and allow the emulation of the respective bus system above an electrical level. The focus here is on the level of abstraction for virtual scenarios and execution performance. Each operation consists of an OP-code and optional arguments. The primary operation is *Transmit*, which indicates that a Network FMU, connected to the bus, intends to send a payload. The payload content mirrors that of a real bus payload, with additional arguments to provide necessary communication frame data, as shown in figure 5 for a *CAN Transmit* bus operation corresponding to a CAN Standard Data Frame. A corresponding Bus Simulation typically responds with a *Confirm* operation to acknowledge receipt of the frame, but it may also reply with a *Bus Error* or other bus system specific operations. The set of provided bus operations ensures that a particular bus system can be simulated to the required level of detail.

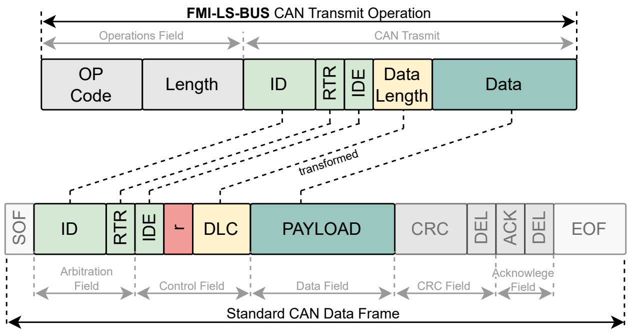


Figure 5. Shows the mapping of the FMI-LS-BUS *CAN Transmit* bus operation to a CAN Standard Data Frame. The information that can be mapped directly is shown in green. The yellow representation describes transformed values and the information shown in red is not used.

In contrast to the High-Cut approach, only one terminal per bus is needed. It consists of a binary variable and a clock variable for each direction of communication (transmission and reception). With the aid of bus specific operations, definition any message on the particular bus can be transported.

Thus, the sum of the instruments of the bus operations, as well as the bus terminals specified for the interconnection of Network FMUs, provides the possibility to use the Network Abstraction within the presented system compositions (see section 2.1). Figure 6 illustrates two Network FMUs integrated via an importer with an integrated Bus Simulation.

Network Abstraction allows a detailed and precise simulation of bus systems. This allows extensive tests using a suitable simulator in later development stages, when it comes to ensuring the functionality of a communication stack and software based on it. Currently, the bus systems planned for Network Abstraction are CAN, CAN FD, CAN XL, FlexRay Ethernet and LIN. However, this list might be extended to include other domains in the future.

[1]The term *Tx Clock* was chosen as an abbreviation for *Transmission-Clock* within the figure.

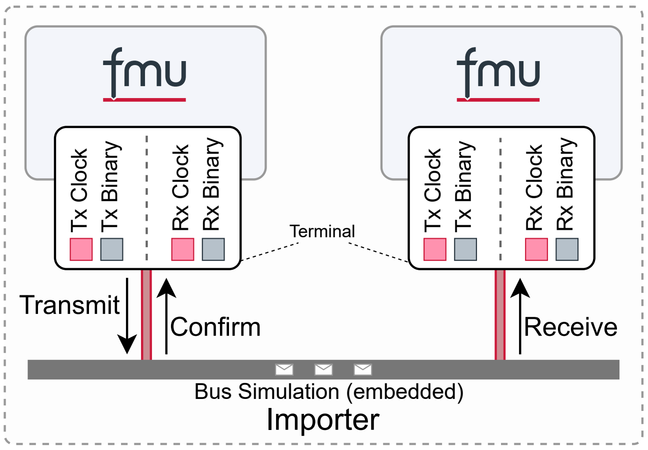


Figure 6. Two Network FMUs integrated via an importer which is coming with integrated Bus Simulation support, based on the Network Abstraction.

2.4 Principle of FMI-LS-BUS

Based on FMI 3.0, FMI-LS-BUS defines the API between FMU and importer and its semantics: What a variable and its values mean, how and when to exchange values and how to interpret those values.

Neither FMI 3.0, nor FMI-LS-BUS, specify what happens inside FMUs and importers. Both standards are agnostic of the FMU generation process (e.g. hand written, generated, wrapper or tool connector) or its content (e.g. AUTOSAR (AUTOSAR R24-11) vECU, file access for replay, aerospace line-replaceable unit (LRU)).

By not defining the internals of either side of the FMI-LS-BUS API's participants, FMI-LS-BUS can be used for many different use cases, as described above: Rough or detailed functional behavior, function emulations or production behavior wrapped in a thin FMI-LS-BUS wrapper.

3 Benefits of Integrating FMUs with FMI-LS-BUS within the Development Process

This section proposes an enhanced V-model, tailored to address the complexities of modern automotive development (see figure 7). This approach uses FMUs as central, standardized exchange artifacts across the entire development lifecycle. Being inherently tool-agnostic, FMI proves particularly valuable in distributed development by facilitating seamless integration at the interfaces between diverse domains (e.g., controls, sensor fusion, vision), distinct development stages (design, implementation, testing), and across heterogeneous tool chains.

During the initial design phases on the left side of the V-model, requirements decomposition occurs, leading to system and architectural decisions, including function distribution. This involves the definition of software components (SWCs) and their specific network interactions. These designs, including functional behavior, parameters, interface definitions, and network information, are then abstracted and encapsulated into executable FMUs, potentially leveraging FMI 3.0 features and the FMI-LS-BUS.

The availability of executable FMUs enables early and continuous virtual integration testing (e.g., MIL/SIL) on the verification branch of the V-model. For example, FMUs representing distinct ECUs, potentially created from multiple suppliers, can be simulated together using FMI-LS-BUS. By enhancing conventional methods (which typically rely on static documentation and interface specifications) by integrating executable specifications, we can substantially reduce ambiguity and error potential. A key benefit of this integration is the ability to formulate and apply test cases early on, enabling the early identification and resolution of inconsistencies that might otherwise arise from interpretation mismatches or format translations between different development stages and tools. This includes managing inconsistent signal definitions across different tool domains, simulating realistic network requirements, and validating end-to-end timing, interactions, and protocol adherence.

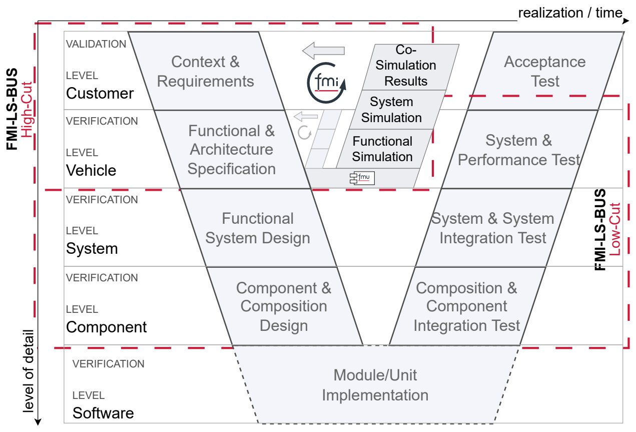

realization / time
level of detail
FMI-LS-BUS High-Cut
FMI-LS-BUS Low-Cut
VALIDATION LEVEL Customer
VERIFICATION LEVEL Vehicle
VERIFICATION LEVEL System
VERIFICATION LEVEL Component
VERIFICATION LEVEL Software
Context & Requirements
Functional & Architecture Specification
Functional System Design
Component & Composition Design
Module/Unit Implementation
Acceptance Test
System & Performance Test
System & System Integration Test
Composition & Component Integration Test
Co-Simulation Results
System Simulation
Functional Simulation
fmi

Figure 7. Enhanced V-model incorporating FMI-LS-BUS. This model illustrates the application of High-Cut and Low-Cut FMUs throughout distinct phases of the automotive development and verification life cycle.

This tight coupling between design and validation aspects of the V-model, facilitated by executable FMUs, enables an agile, iterative methodology, thereby supporting Continuous Integration/Continuous Testing (CI/CT) practices. Smaller, faster development cycles for components or subsystems provide rapid feedback. This significantly reduces integration risks later in the process and enhances overall efficiency, traceability, and the capacity to manage the complexity inherent in modern vehicle compositions. Furthermore, the standardized nature of FMUs promotes model reuse across different projects, simulation environments, and even between collaborating organizations (e.g., OEM and Tiers), maximizing the return on modeling effort.

Figure 7 illustrates this enhanced V-model framework that integrates FMI-LS-BUS to support both High-Cut and

Low-Cut FMUs, targeting different phases in the development process. Typically, High-Cut FMUs encapsulate application logic or functional behavior (often derived early in the design), while Low-Cut FMUs focus on detailed network interface behavior and network stack implementations (like AUTOSAR Classic), enabling verification at different integration levels as development progresses.

3.1 System Requirements and Architecture Definition

The High-Cut variant addresses the "top-left" of the development process, including requirements decomposition, system architecture definition, and the allocation of functions to logical or preliminary architectural components (see figure 8).

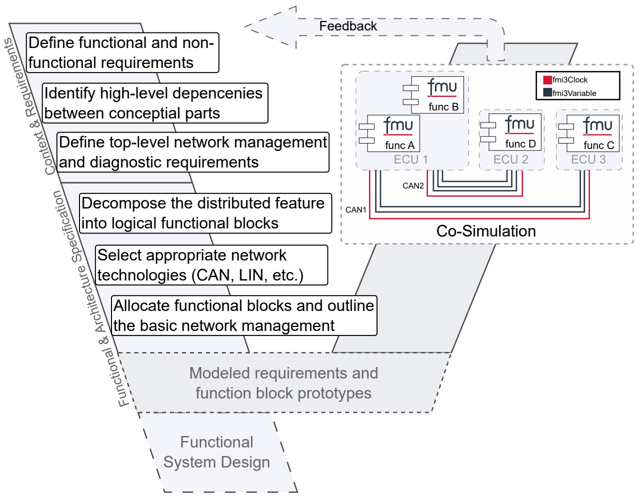

Feedback
Context & Requirements
Functional & Architecture Specification
Define functional and non-functional requirements
Identify high-level depencenies between conceptial parts
Define top-level network management and diagnostic requirements
Decompose the distributed feature into logical functional blocks
Select appropriate network technologies (CAN, LIN, etc.)
Allocate functional blocks and outline the basic network management
Modeled requirements and function block prototypes
Functional System Design
fmu
func B
func A
func D
func C
ECU 1
ECU 2
ECU 3
fmi3Clock
fmi3Variable
CAN2
CAN1
Co-Simulation

Figure 8. High-Cut use cases in the V-model. A proposed workflow for a quick development testing feedback loop using FMI-LS-BUS High-Cut features.

In this context, FMUs become executable design objects that embody requirements and components, offering a precise and verified definition early in the development lifecycle. The relationship between requirements and executable models improves traceability, verification, and impact analysis. The interoperability of the FMI standard can be a significant improvement in addition to traditional document-centric methods, allowing experts to integrate models from diverse tools like MBSE platforms and domain-specific simulators. These models serve as preliminary executable definitions that yield insights more rapidly and accelerate the specification process.

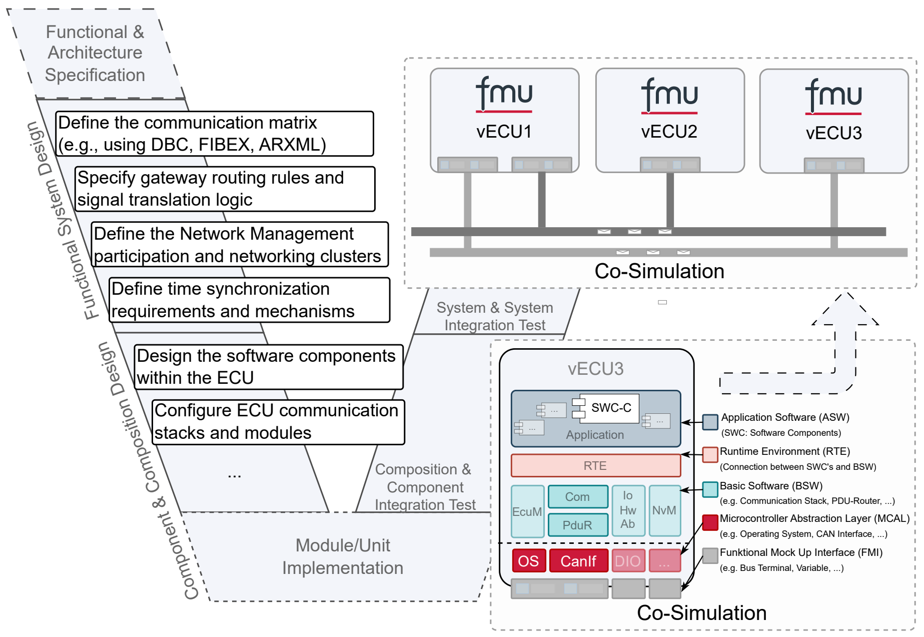
One example usecase is the allocation of distributed functions within a vehicle network, where a function is spread across multiple components. Simulation can be used to assess the impact of moving a function from one control unit to another, revealing how the change affects system behavior and communication. FMI-LS-BUS describes the interaction between these functional FMUs as part of automotive networks at a High-Cut abstraction level by modeling the interaction between these FMUs at the "signal level. This abstraction emphasizes the validation of functional data exchange between components while omitting the characteristics of network protocols.

The primary validation objective in this design phase is to confirm the architecture's functional accuracy, validate data flows, and detect conceptual errors in distributed logic before they are implemented. By enabling Verification and Validation activities much earlier than conventional, late-stage integration methods, High-Cut simulations with FMI-LS-BUS allow teams to identify and correct architectural flaws before they propagate into more costly development stages. Crucially, even with signal-level abstraction, this approach facilitates the early evaluation of closed-loop behavior, capturing essential timing factors and discretization effects that might otherwise be overlooked. Moreover, these FMUs serve as a shared, executable specification that improves collaboration between system architects and implementation teams. This single source of truth provides a clear, unambiguous, and validated artifact that can be shared with suppliers, which is especially beneficial when developing functions across organizational boundaries.

3.2 ECU Software Development and Component Testing

Building on the validated architecture, this iteration concentrates on the detailed development of Software Components (SWCs) and their integration into virtual ECUs (vECUs). Function developers and software engineers refine the FMUs with the executable specification or create new, more detailed FMUs representing concrete SWCs or vECUs, potentially incorporating auto-generated or handwritten code. The adoption of FMUs supporting FMI-LS-BUS substantially enhances the workflow, enabling early virtual integration and testing of software long before hardware is available. For instance, an OEM can easily integrate and test vECUs from multiple suppliers in a common simulation environment. This early, hardware-independent validation results in robust vECUs that serve as reliable deliverables for the next stage, significantly accelerating the overall development cycle.

In advance, network communication can be verified at a more detailed "Low-Cut" abstraction level using FMI-LS-BUS. This involves simulating the AUTOSAR communication stack, including the packing and unpacking of signals into network messages (e.g., CAN frames) based on standard description formats like DBC or ARXML. Simulating these vECUs together allows for thorough testing of the communication software, ensuring the implementation meets network interface requirements. This helps developers identify critical errors, such as incorrect signal packing or protocol deviations, preventing them from becoming costly problems during later hardware integration phases. The simulation can also be extended to model key characteristics of the bus itself, enabling the analysis of network-level phenomena such as timing effects and the correctness of error-handling mechanisms like mes-



Functional & Architecture Specification
Functional System Design
Define the communication matrix (e.g., using DBC, FIBEX, ARXML)
Specify gateway routing rules and signal translation logic
Define the Network Management participation and networking clusters
Define time synchronization requirements and mechanisms
Component & Composition Design
Design the software components within the ECU
Configure ECU communication stacks and modules
...
Module/Unit Implementation
Composition & Component Integration Test
System & System Integration Test
fmu
vECU1
fmu
vECU2
fmu
vECU3
Co-Simulation
vECU3
SWC-C
...
Application
RTE
EcuM
Com
PduR
Io Hw Ab
NvM
OS
CanIf
DIO
Application Software (ASW)
(SWC: Software Components)
Runtime Environment (RTE)
(Connection between SWC's and BSW)
Basic Software (BSW)
(e.g. Communication Stack, PDU-Router, ...)
Microcontroller Abstraction Layer (MCAL)
(e.g. Operating System, CAN Interface, ...)
Funktional Mock Up Interface (FMI)
(e.g. Bus Terminal, Variable, ...)
Co-Simulation

Figure 9. Low-Cut use cases in the V-model. This figure illustrates a proposed workflow for a quick development testing feedback loop using FMI-LS-BUS Low-Cut features.

sage retransmission.

3.3 System Testing

Proceeding from the integrated vECU developed and validated in section 3.2, this final virtual iteration assembles the individual vECUs into a complete system simulation. The primary objective is comprehensive testing in a realistic environment to create a high-confidence virtual release candidate. System integration and test teams utilize the collection of vECUs alongside potentially more sophisticated Bus Simulation FMUs (see section 2.1.2) or use an integrated Bus Simulation (see section 2.1.3) representing the network bus itself. These Bus Simulations, also leveraging FMI-LS-BUS (Low-Cut), incorporate enhanced capabilities beyond simple message transport, such as detailed timing models and error injection mechanisms, providing a realistic virtual testbed. The standardized FMU interface within this iteration affords significant flexibility, allowing test teams to readily combine vECUs from various sources (different internal teams or external partners) with advanced bus models and test harnesses, potentially originating from different tool vendors. This enhances test flexibility, coverage, and the ability to simulate realistic system conditions, ensuring functional requirements and communication specifications are met at the system level.

The validation at this system level is thorough, intended for ensuring end-to-end correctness and identifying integration problems that isolated component tests may not detect. The procedure include validating functional performance throughout the complete virtual system and evaluating detailed network attributes such as latencies, scheduling conformity, and bus load. By systematically injecting network errors—such as message loss, data corruption, or node failures—through the advanced bus simulation, the system's reaction can be properly assessed against its requirements. Successful completion of this virtual testing phase provides high confidence in the system's correctness, performance, and robustness. The validated set of integrated FMUs represents a comprehensive virtual prototype, ready to be shared with other stakeholders (e.g., for calibration or HiL testing) as part of the distributed development and release process, ensuring consistency prior or in parallel to hardware deployment.

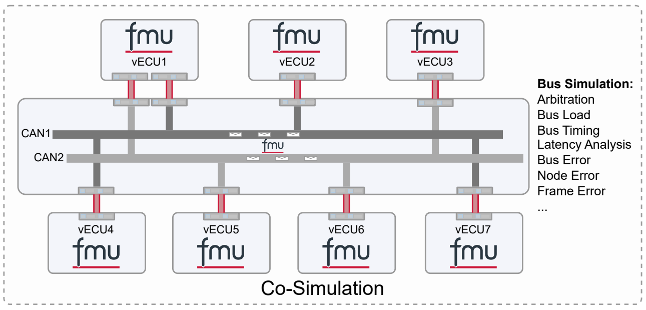


Figure 10. This figure illustrates a total of seven VECU FMUs connected via two virtual CAN bus lines (CAN1 and CAN2). A portion of the VECU FMUs is connected via both bus lines and thus communicates in both CAN networks. Using the Low-Cut approach, very detailed and global tests on a system level are possible to ensure the functionality of the various vECU FMUs in various scenarios.

3.4 Operation and Maintenance

The integration of FMUs connected via FMI-LS-BUS extends its benefits beyond initial development and testing into the vehicle's operational phase and subsequent maintenance activities. Virtual models and test environments created during the development life cycle, encompassing a wide range of potential vehicle configurations, become valuable assets for post-release tasks.

Managing Variant Complexity with Virtual Fleets: A virtual fleet enhances troubleshooting capabilities. When addressing field-reported issues potentially specific to certain variants, the corresponding virtual configuration can be quickly instantiated to reproduce the fault condition. Validated system representations, containerized as FMUs, are used within these simulations. Field recordings are utilized to stimulate the simulation. Therefore, the Low-Cut abstraction is especially useful since the captured traces can be easily transformed to the FMI-LS-BUS datagram. Furthermore, operational data collected from vehicles in the field can then be added into the simulation database to improve and correlate the original FMU models, gradually increasing accuracy over time.

Quality Assurance of Updates Across Variants: A virtual system environment, representing the necessary variants from the virtual fleet, is crucial for validating software upgrades before deployment, especially for Over-the-Air (OTA) updates. Prior to releasing updates, their compatibility and performance can be rigorously tested across all relevant deployed configurations within the FMI-LS-BUS simulation system. This pre-deployment validation against the virtual fleet acts as essential quality assurance, significantly reducing the risk of introducing variant-specific defects or regressions into operational vehicles, thereby safeguarding the safety and reliability of OTA updates across the diverse product range.

Digital Twin for Long-Term Maintenance: The containerized nature of FMUs simplifies archiving specific system configurations and component baselines for each variant (Coïc 2021). This supports consistent, traceable long-term maintenance across the entire vehicle portfolio. A specific configuration from the virtual fleet, potentially updated with operational data for a particular physical vehicle, can function as a key component of that vehicle's Digital Twin, enabling variant-specific analysis and prediction.

Service Training: The library of FMU models, covering various vehicle configurations, serves as an effective tool for the supplementary training of service technicians. They can learn about variant-specific systems and diagnostics in a virtual environment, enhancing competence without the constraints and costs associated with accessing diverse physical hardware.

3.5 Conclusion

By establishing FMUs as standardized, executable exchange artifacts throughout the proposed "enhanced" V-model development process, FMUs close critical gaps between development domains, stages, tools, teams and companies. The iterative application of this methodology, leveraging both High-Cut (signal-level) and Low-Cut (message-level) FMI-LS-BUS abstractions, enables the following significant benefits:

This methodology improves efficiency, reduces integration risks and costs, enhances quality and reliability, fosters collaboration, and manages the inherent complexity in developing modern, distributed automotive electronic systems through standardization, virtualization, and early, continuous validation.

4 Validation of FMI-LS-BUS

Interoperability is an essential aspect of modern model-based systems engineering, enabled by standards like the FMI. FMI encourages exchange and simulation of dynamic models created with various authoring tools. The FMI 3.0 Layered Standard for Network Communication (FMI-LS-BUS) has been developed to extend these capabilities to distributed systems and network-based simulation architectures. The demonstrated interoperability across a wide range of technologies and organizations is an important measure of such standards' success and practical adoption.

At the ASAM International Conference 2024 (Bertsch 2024), the FMI-Design Group demonstrated the seamless interoperability between different tools. This demonstration showcased the High-Cut and Low-Cut capabilities based on automotive simulation use cases.

Well-known SiL software tool vendors such as Akkodis, Altair, AVL, Bosch, dSPACE, PMSF IT Consulting and Synopsys participated in this demonstration with their tools (Bertsch 2024).

5 Other usage domains

The current focus of standard network protocols predominantly targets technologies utilized within the automotive sector. However, other industries, such as shipping, aerospace, off-road machinery, and agriculture, employ distinct network technologies tailored to their specific requirements. Examples include CANaerospace, ARINC 667, AFDX, MIL-STD-1553, and ISOBUS.

With the successful integration of automotive network technologies into the FMI-LS-BUS, there is potential for incorporating additional domain-specific technologies. This process can be facilitated by interested parties from both within and outside the FMI community. Such integrations may occur with or without formal endorsement from the Modelica Association Project FMI.

6 Summary and Outlook

The increasing complexity of products and development processes necessitates not only the standardization of signal exchange but also the harmonization of network message exchange for simulation artifacts. The FMI-LS-BUS addresses this need by providing a standardized framework for network simulation, building upon the widely adopted FMI 3.0 standard. By introducing two abstraction layers, the standard enables flexible utilization of network communication throughout all phases of product development. Early development stages can leverage high-level abstractions (High-Cut), while later stages can adopt low-level details (Low-Cut) as more specific network behaviors are defined.

FMI is already a standard for many OEMs and suppliers when developing and exchanging vECUs. The FMI-LS-BUS, as a key technology for cross-manufacturer description of bus simulations in FMUs, will strengthen the core FMI standard itself and allow users to use it in new and unprecedented areas of application. To facilitate implementation, a range of resources is available, including an FMI-LS-BUS Implementers' Guide (MAP FMI 2025b) and reference FMUs, which are accessible via the FMI website *www.fmi-standard.org*. These resources aim to streamline integration efforts for tool providers and their developers.

Regarding a possible description format for the system interconnection of buses described in FMUs for the FMI-LS-BUS, the SSP standard (MAP SSP 2024) is a suitable option. Current analyses regarding the possible specification are part of ongoing work. This could offer further simplifications for users.

Additionally, stakeholders are encouraged to participate in the FMI community or develop their own layered standards on top of FMI 3.0. Such initiatives can occur independently or in collaboration with the FMI project, fostering innovation and extending the applicability of FMI standards across diverse domains.

Acknowledgments

We would like to thank all contributors to the FMI Standard and the Layered Standards built on top of it. Numerous ideas and feedback from FMI users and the Advisory Board supported us throughout the design process and provided valuable insights. We thank the companies, especially those of the authors, for their support, which enable us to continue the standardization.

References

AUTOSAR (R24-11). *AUTOSAR Layered Software Architecture*. URL: https://www.autosar.org/.

Bertsch, Christian et al. (2023). "Beyond FMI - Towards New Applications with Layered Standards". In: *15th International Modelica Conference*. URL: https://doi.org/10.3384/ecp204381.

Bertsch, Christian et al. (2024). *FMI, Layered Standards and ASAM Standards - Enabling Seamless SiL Simulation of Virtual ECU, Presentation at International ASAM Conference*. URL: https://fmi-standard.org/assets/literature/FMI-LS-ASAM_2024.pdf.

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: http://dx.doi.org/10.3384/ecp12076173.

Coïc, Clément et al. (2021). "Linking Design Requirements to FMUs to create LOTAR compliant MBSE models". In: *14th International Modelica Conference*. URL: https://doi.org/10.3384/ecp2118149.

dSPACE (2024). *(R)evolution in the Exchange of V-ECUs*. URL: https://www.dspace.com/en/pub/home/news/engineers-insights/fmi-v-ecu-standard.cfm.

dSPACE (2025). *Virtual Validation Redefined*. URL: https://www.dspace.com/en/pub/home/news/engineers-insights/fmi-v-ecu-standard.cfm.

ISO (2011). *ISO 26262: Road vehicles – Functional safety*. Norm. URL: https://www.iso.org/obp/ui/#iso:std:iso:26262:-1:ed-2:v1:en.

Junghanns, Andreas et al. (2021). "The Functional Mock-up Interface 3.0 - New Features Enabling New Applications". In: *14th International Modelica Conference*. URL: https://doi.org/10.3384/ecp2118117.

Lawrenz, Wolfhard (2013-12). *CAN System Engineering*. Springer Science and Business Media.

MAP FMI (2025a). *FMI Layered Standard for network communication v1.0.0rc.1*. URL: https://github.com/modelica/fmi-ls-bus.

MAP FMI (2025b). *FMI LS BUS Implementers' Guide*. URL: https://modelica.github.io/fmi-guides/main/ls-bus-guide/.

MAP SSP (2024-03). *System Structure and Parameterization (SSP) Standard 2.0*. URL: https://ssp-standard.org/docs/2.0/.

Müller-Ettrich, Gunter (1998). "System Development with V-Model and UML". In: *The Unified Modeling Language*. Ed. by Martin Schader and Axel Korthaus. Heidelberg: Physica-Verlag HD, pp. 238–249.

Rausch, Mathias (2022). *Kommunikationssysteme im Automobil - LIN, CAN, CAN FD, CAN XL, FlexRay, Automotive Ethernet*. M: Carl Hanser Verlag GmbH Co KG. ISBN: 978-3-446-47457-4.