Federated Simulation Standard (FSS) Whitepaper

February 24, 2025

Authors

This whitepaper has been created by the Accellera Federated Simulation Standard Working Group (FSSWG) and Federated Simulation User Group (FSUG):

Martin Barnasconi, *Chair*
Mark Burton, *Vice Chair*

The FSSWG and FSUG gratefully acknowledges the contributions of the following participants:

Mohamed Abd El Salam Ahmed, Siemens EDA
Boon Chong Ang - Intel Corporation
Martin Barnasconi, NXP Semiconductors
Mark Burton, Qualcomm
Jean Casteres, Airbus
Eric Jenn, IRT Saint Exupéry
Jean-Marc Talbot, Siemens EDA
François-Frédéric Ozog, Shokubai
Trevor Wieman, Ford Motor Company

Table of contents

1. Introduction 4
1.1. Background and problem statement 4
1.2. Purpose 4
1.3. Scope 4
1.4. Standardization and industry alignment 5
2. Use cases and application scenarios 6
2.1. Pre-silicon sensor-fusion development 6
2.2. Virtual and hybrid testing 6
2.3. Abstraction and virtualization enabling efficient HW/SW co-development 7
2.4. Digital Twin 8
3. General Requirements 10
4. Concept definition 11
5. Demonstrator 12
6. References 13
Annex A Terms, definitions, abbreviations and acronyms 14
A.1 Terms and definitions 14
A.2 Abbreviations and acronyms 14

1. Introduction

1.1. Background and problem statement

Different industry segments (e.g., Semiconductors, Automotive, Space, Avionics, Defense, Industrial, etc.) are working independently to address the integration and interoperability challenges when developing a heterogenous and distributed modeling and simulation ecosystem. Consequently, different and overlapping standards and ecosystems have been developed in each industry segment, which limit cross-industry exchange and reuse of simulation technologies, models, formats, or interfaces.

Although generic and domain-independent standards have been developed to address distributed modeling and simulations ([1],[2]), the adoption of these standards has been very application and industry-specific (mainly applied in Defense industry) and also revealed limitations and constraints [3] which hinder broader cross-industry application and adoption.

Even within a single industry segment, different standards and formats for modeling and simulation have been emerging, often addressing specific needs and use cases in different areas (e.g., software, electrical components/hardware, mechatronics, application/environment, etc.).

Due to this, combining these various standards and formats in a coherent simulation ecosystem becomes a complex and tedious task, because of missing interoperability rules and integration standards.

1.2. Purpose

The Federated Simulation Standard (FSS) aims at establishing cross-industry alignment and standardization to enable interoperability, reuse and integration of established and new modeling and simulation concepts and technologies. In this way, a distributed and federated simulation ecosystem can be established, in which different industry-specific or domain-specific standards and technologies can be combined.

1.3. Scope

The Federated Simulation Standard aims to develop a standard and open infrastructure to enable interoperability of established modeling and simulation standards, technologies and tools as part of a distributed, orchestrated and federated simulation ecosystem.

The interfaces defined by the Federated Simulation Standard should also facilitate integration to physical hardware implementations, supporting concepts such as Hardware-in-the-Loop (HiL), Processor-in-the-Loop (PiL), Software-in-the-Loop (SiL) and Model-in-the-Loop (MiL) setups.

The standard shall address at least the following aspects:

- Standardize the meta-model (‘manifest’) of executable models, such that these models specify their capabilities in terms of supported execution semantics and abstraction level (e.g., interface, model of computation, timing, functionality, etc.).
- Define a set of application programming interfaces (APIs) to facilitate the exchange of data, control and synchronization (incl. time management) between the different environments across the ecosystem.

The standard is made available as a portable document (PDF) as well as interactive and online webpage (HTML).

Along with the standard, supplemental material is made available to encourage industry adoption and support:

- A proof-of-concept implementation acting as reference implementation of the API to verify and validate the concepts and semantics as defined in the standard.
- A regression suite including relevant unit tests and (simple) test cases to verify the API.
- 'Standard adapters' to connect between the FSS and existing standards (e.g., SystemC TLM [4], FMI [5], SMP [6], and ED-247 [7]).

Supplemental material is made available via the Accellera GitHub public repositories [8].

Note: Development of a demonstrator is outside the scope of the FSS working group. Instead, this could be addressed by the user group or industry project. A proposal for a demonstrator is given in section 5.

1.4. Standardization and industry alignment

A Federated Simulation Standard Working Group (FSSWG) has been established, which will closely interact with the Federated Simulation User Group (FSUG).

The FSS Working Group will focus on the development of the standard itself and the supplemental material (e.g., PoC, regression tests, etc.). Working group participation requires Accellera membership. Obviously, alignment with and integration of other standards and technologies originating from Accellera is considered, such as the use of IEEE Std 1666-2023 (SystemC TLM) [4].

The user group offers an open collaboration forum to align and coordinate activities across different industries, organizations and consortia. Participation in the user group does not require Accellera membership, however, written acknowledgement of the IP Rights policy is mandatory.

The user group aims to align with other initiatives and developments happening in various industries and organizations, such as the Eclipse Foundation running the Software Defined Vehicle project [9] and Eclipse Zenoh project [10], the European Cooperation for Space Standardization (ECSS) developing the SMP standard [6], the European Organisation for Civil Aviation Equipment (EUROCAE) developing the ED-247 standard [7], and the Modelica Association developing the FMI [5] and DCP [11] standards.

If recognized by sufficient industry partners, a focused R&D project could be established to help in the definition and implementation of the technologies required to establish the federated simulation ecosystem. An industry-funded R&D project coordinated by the IRT Saint Exupéry [12] is under consideration. The draft project proposal can be found here [13].

2. Use cases and application scenarios

This section describes the various use cases and/or application scenarios which have been identified by the involved industries (Automotive, Space, Avionics and Semiconductors). The use cases and/or application scenarios described in this section have been generalized such that they are applicable in each of these industries. For that reason, implementation-specific details are not described, as the main purpose is to define ***what*** FSS should address, and not ***how***. Furthermore, these use cases and application scenarios are considered complementary and could be combined.

2.1. Pre-silicon sensor-fusion development

This application scenario originates from an automotive application, where different sensor technologies are used in a car (e.g., radar, lidar, ultrasound, vision/camera, etc.) to enable object detection for autonomous driving. Figure 1 depicts the sensor fusion application scenario, where data coming from different sensors is analyzed as part of a sensor fusion ASIC or SoC, which is controlled by an application processor. To enable pre-silicon sensor fusion development, virtual representations of all the elements should be available as part of a heterogeneous simulation environment, which includes physical models of the sensors, a hardware model of the integrated circuit and a processor model including software. To feed the sensor with relevant stimuli, a model of the environment should be incorporated as well.

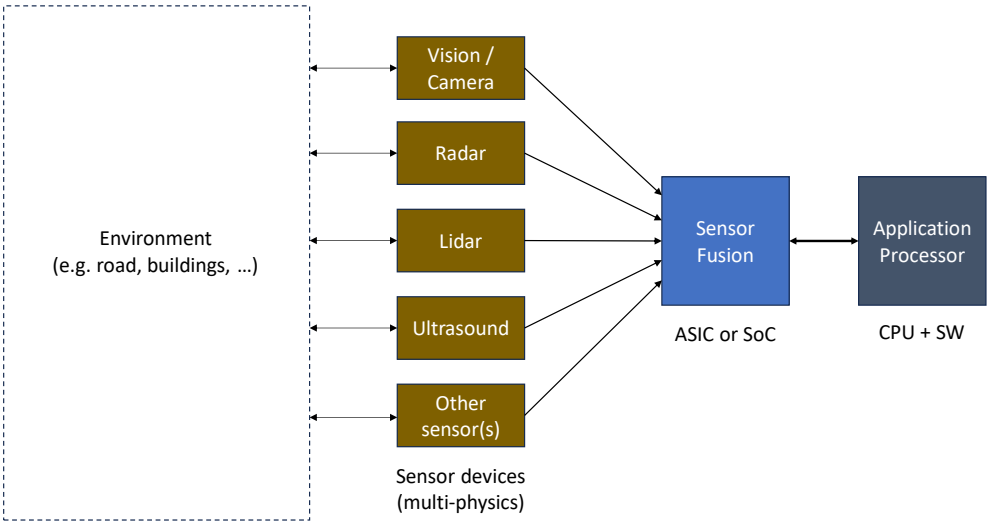

Figure 1 Sensor fusion application

2.2. Virtual and hybrid testing

In an early phase of the product creation process, a virtual environment could be used in which executable models represent the product, to perform tasks such as virtual testing. As soon as the first physical implementation(s) of the hardware or equipment become available, a hybrid setup is often used to validate the combination of models and physical hardware.

Figure 2 shows an abstract representation of such a hybrid test environment in which the virtual and physical environment are interconnected via a shared communication infrastructure. A variant setup could include a distributed structure of hybrid test setups, using internet and/or cloud services for communication, simulation, and even acceleration/emulation technologies.

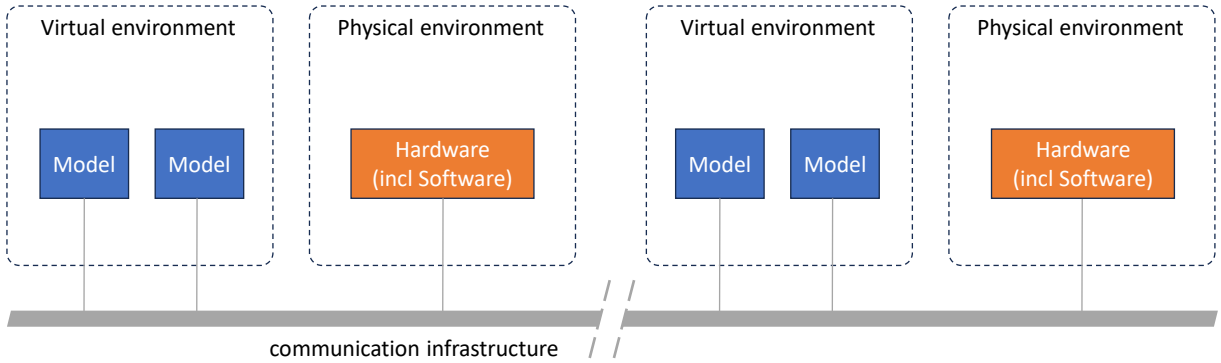

Figure 2 Hybrid testing environment (including distributed option)

Different virtual and hybrid testing approaches exist, as listed below in order of decreasing abstraction:

- Model-in-the-Loop (MiL): The product/system is represented by a model and simulated in a virtual environment. Also called virtual testing.
- Software-in-the-Loop (SiL): The software part of the product/system is compiled with the host compiler within a simulation-based virtual environment. In this case there is no notion of a processor model.
- Processor-in-the-Loop (PiL): The software part of the product/system is compiled with the target compiler and executed on the target processor, which is either the physical implementation or a processor model (e.g., ISS). The latter approach is hybrid testing.
- Hardware-in-the-Loop (HiL): The product/system is represented by a real physical hardware implementation and connected to a simulation-based virtual environment. Also called hybrid testing.
- Vehicle-in-the-Loop (ViL) and proving-ground-vehicle-in-the-loop (PG-ViL)[14] :ViL testing incorporates software and hardware into the vehicle for real-world testing with simulated components. For extensive study and optimization of system components or functionalities, PG-ViL commonly simulates powertrains or processors. For final validations, these stages ensure the system works securely and efficiently in real life.

A combination of these approaches, forming hybrid testing environments, have been proven highly valuable. Combing all approaches is often called Everything-in-the-Loop (XiL).

2.3. Abstraction and virtualization enabling efficient HW/SW co-development

With the increasing software content in embedded systems, many traditionally hardware-centric products are transforming into software-centric systems. This transition requires a different modeling and simulation ecosystem, which shall support different levels of abstraction and the use of virtualization and simulation concepts to enable efficient HW/SW co-architecting, co-design and co-verification.

Figure 3 shows a number of use cases to develop software on top of a virtualized, simulated, or emulated infrastructure:

- Use case A: Algorithm development: Algorithms developed in languages like C/C++ or Python are compiled (or interpreted) on the host computer.
- Use case B: Application development using the API to the middleware, however, this middleware substitutes the real middle-layer and might even replace the OS for an functional equivalent approach.
- Use case C: Both application and middleware (which might include the operating system) are implemented on top of a Hardware Abstraction Layer (HAL).
- Use case D: The full software stack, including application-layer, middle-layer (OS) and HAL are executed on an Instruction Set Simulator (ISS). The software is compiled for the target MCU platform.
- Use case E: This use case introduces the hypervisor to facilitate the creation of multiple virtual machines in which independent software stacks are executed.
- Use case F: Similar as use case E, except the processor model is the real physical implementation, and not an abstract instruction-accurate model or emulator.

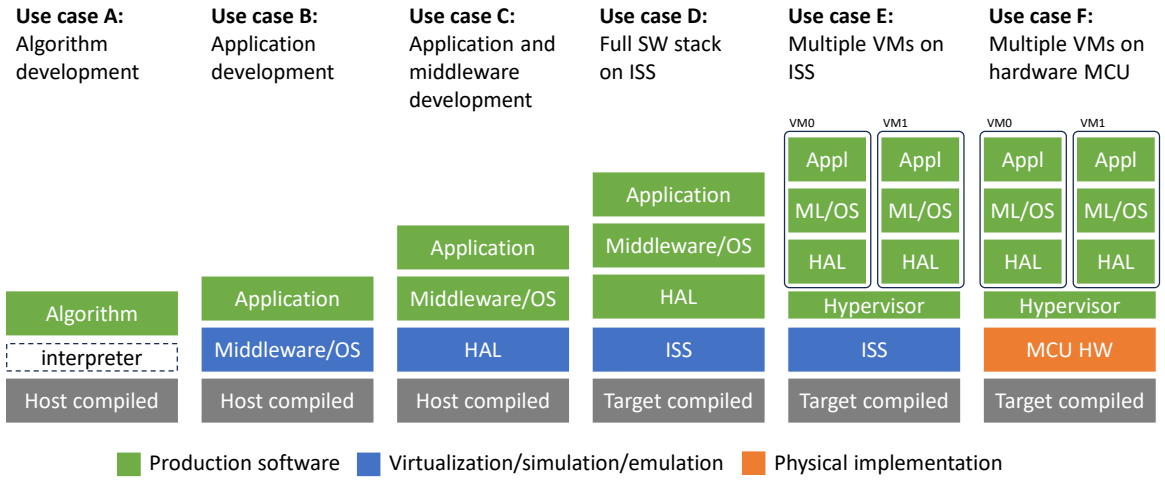

Figure 3 Abstraction and virtualization concepts

2.4. Digital Twin

A digital twin is a virtual representation of a physical product, production plant, or process. It enables engineers to design, simulate, and optimize products, machines, production plants or processes in the digital (virtual) world before implementing them in the real (physical) world.

A digital twin integrates abstract equivalent representations of the electronics, embedded software, mechanical parts, and sensors to create live simulations that update and change as their physical twin counterparts change. This technology is widely used in various industries including automotive, robotics, avionics, space, and healthcare to name a few, to master complexity, enhance efficiency, predict problems before they occur, and provide insights into how products can be improved.

A scalable modeling and simulation ecosystem should enable a seamless interaction between the different digital twin components (e.g., models, simulation/emulation technologies, physical hardware, software, etc.), maintaining accurate timing across different simulation environments, and encouraging collaboration to address complex, multi-domain design, verification and manufacturing challenges.

As software-defined products are gaining momentum, as exemplified by the automotive industry, digital twins need to cover several product components from chip to system. Virtual and hybrid platforms are increasingly used to create these digital twins to run the full software stack.

Figure 4 shows an industrial use case for a digital twin of an Autonomous Mobile Robot (AMR) in factory logistics. It allows system design and verification using a virtual environment in the cloud and once the embedded software of the AMR is verified in its system and factory context, it can be easily deployed to the real physical environment using an over-the-air (OTA) update.

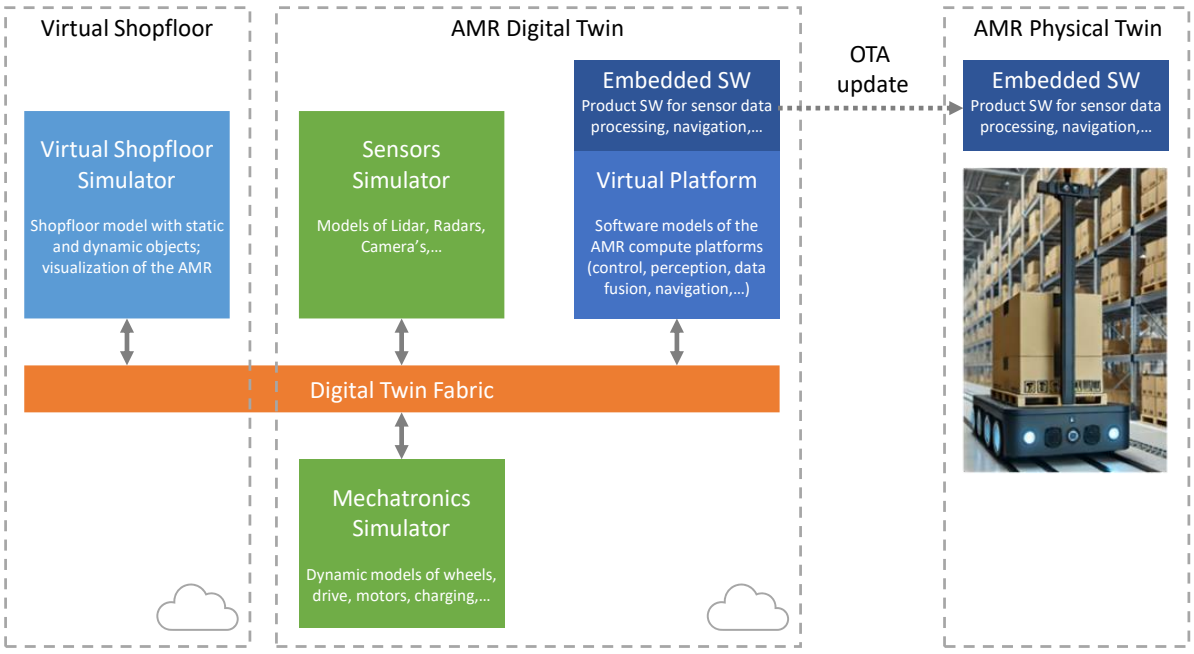

Figure 4 Autonomous Mobile Robot Digital Twin in factory logistics

3. General Requirements

This section lists the general requirements [15] for the Federated Simulation Standard:

1. FSS shall enable the composition of a distributed and orchestrated modeling and simulation environment to specify, architect, analyze, integrate, verify, and/or validate complex heterogenous systems or systems-of-systems.
2. FSS shall define (the use of) a standardized interface(s) and/or communication protocol(s) to interconnect simulation or physical environments (and models) in any topology (e.g., LAN, WAN, cloud, cloud-to-cloud).
3. FSS shall be agnostic to the way the behavior of a simulation component is implemented (software, hardware, algorithm/functional, etc.).
4. FSS shall support modeling and simulation of different design representations according to the OSI model/stack.
5. FSS shall enable interoperability between models, models-to-infrastructure, infrastructure-to-infrastructure, etc. to facilitate model reuse, exchange and portability.
6. FSS shall enable embedding/integration of existing standards (e.g., SystemC/IEEE1666 [4], FMI [5], SMP [6], ED-247 [7]).
7. FSS shall support the creation and integration of dedicated actors/services for specific (shared) functions or tasks (e.g., simulation, memory/data storage, time sync, logging, reporting, checkpointing, post-simulation analysis, etc.).
8. FSS shall support hybrid modeling and simulation concepts (e.g., MiL, SiL, PiL, HiL, ViL, etc.).
9. FSS shall support QoS for network communication.
10. FSS shall support time management and temporal decoupling concepts (e.g., time quantum).
11. FSS shall be based on standards which are proven to be high-performance in terms of data throughput, low-latency, deterministic, real-time data processing, low overhead.
12. FSS shall offer an API available in different implementation languages (e.g., C++, Python, etc.) to facilitate integration with domain/discipline specific languages.
13. FSS shall enable seamless integration of 3rd party models and simulation kernels/engines.
14. FSS shall support different levels of security (e.g., authorization, authentication, access control, encryption, secure connections).

4. Concept definition

Figure 5 shows the concept definition of the Federated Simulation Standard. The standard itself is depicted in blue and covers the API and execution semantics such that different modeling and simulation environments can interact.

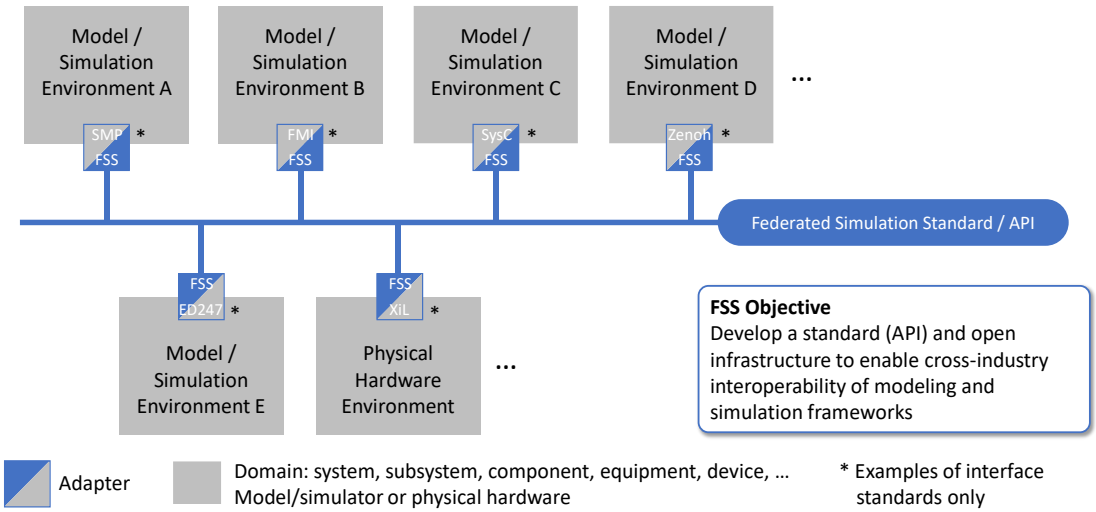

Figure 5 FSS concept definition

The elements in grey show the existing modeling and simulation environments, which could connect to the FSS infrastructure by means of adapters. Although the definition and implementation of adapters is outside the scope of the standard, the development and availability of adapters to commonly used standards and interfaces is considered in scope of the working group, as part of the development of the proof-of-concept implementation, to encourage industry adoption and support of an open infrastructure and ecosystem.

It is expected that the underlying communication protocol (i.e. transport/network layer) is implementation defined. The FSS standard will primarily focus on standardizing the interface, and not on the definition and implementation of the underlying transport/network layer. This means established technologies, standards or protocols could be used to implement the transport layer. To enable the use of an existing transport layer, a middleware abstraction layer will be defined and developed, which is part of the proof-of-concept implementation. Figure 6 shows the how the various elements are layered.

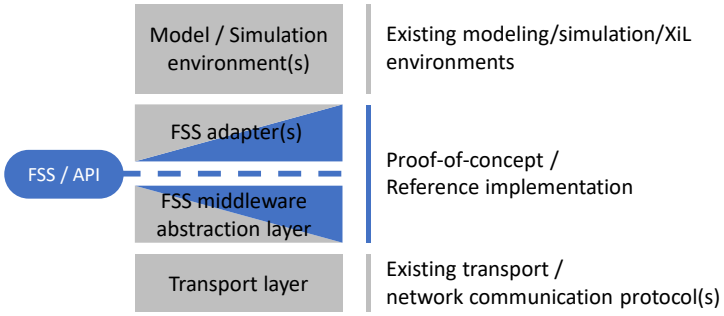

Figure 6 FSS layered elements

5. Demonstrator

Figure 7 shows the definition of an automotive-centric demonstrator where existing modeling and simulation environment and technologies are interconnected using the FSS API and adapters. Depending on the selected use case and application scenario (see section 2), only the relevant environments and interfaces are established.

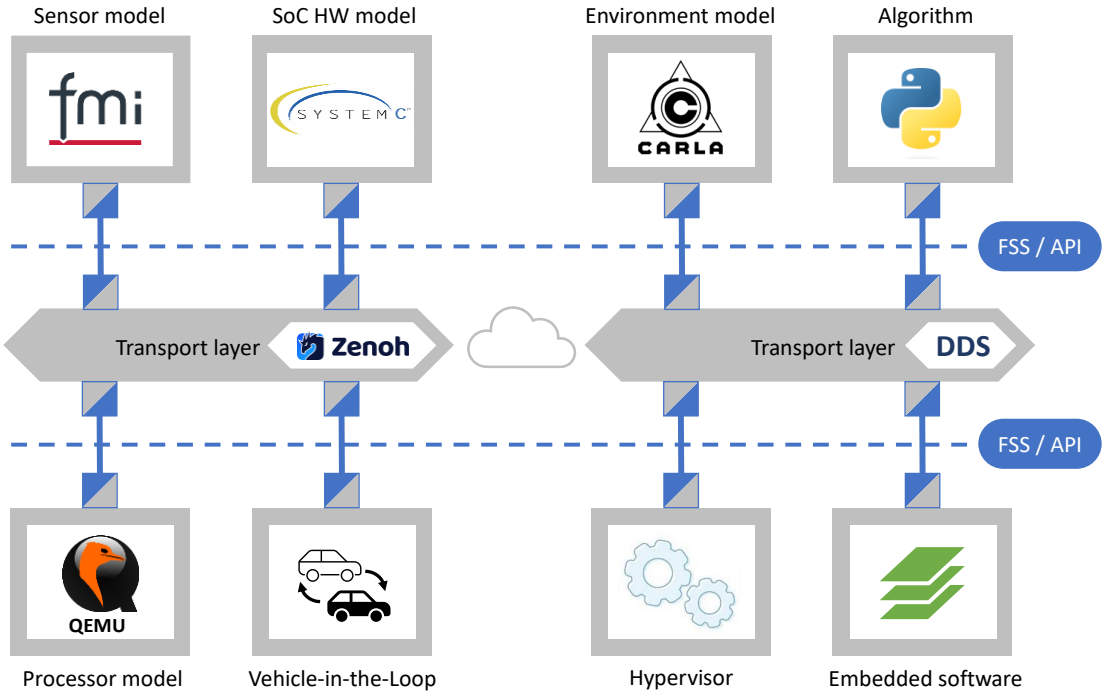

Figure 7 Demonstrator definition (Automotive application)

It is expected that not all modeling and simulation environments are located in single compute farm (on-premise). Instead, different organizations in the supply-chain will use their own environment, which needs to be interconnected to compose the overall federated simulation ecosystem. Therefore, the transport layer should support a secure and reliable communication infrastructure supporting multi-prem use models.

The modeling and simulation environments used in the demonstrator will be selected by the FSS User Group and Working Group. This selection might depend on possible technology contributions of Accellera members to the FSS Working Group, as part of the standard/API definition process. The actual implementation of a demonstrator is outside the scope of the FSS Working Group. Instead, a demonstrator can be developed by the user group or separate industry-funded project.

To support the creation of an open federated modeling and simulation ecosystem, the use of proven web- and cloud-based technologies is encouraged.

6. References

[1] IEEE Std 1516-2010, Standard for Modeling and Simulation (M&S) High Level Architecture (HLA), https://standards.ieee.org/ieee/1516/3744/
[2] Simulation Interoperability Standards Organization (SISO), WebLVC, SISO-STD-017, https://www.sisostandards.org/resource/resmgr/standards_products/siso-std-017-2022_weblvc_pro.pdf
[3] ADLink, Simulation Whitepaper, 2019, https://www.omg.org/news/whitepapers/Simulation-Whitepaper-v2.0.pdf
[4] IEEE Std 1666-2023, SystemC, https://standards.ieee.org/ieee/1666/7293/
[5] FMI standard, https://fmi-standard.org/
[6] ECSS SMP, https://ecss.nl/standard/ecss-e-st-40-07c-simulation-modelling-platform-2-march-2020/
[7] EUROCAE, ED-247 Rev B, https://eshop.eurocae.net/eurocae-documents-and-reports/ed-247b/
[8] Accellera GitHub repositories, https://github.com/accellera-official/
[9] Eclipse Software Define Vehicle project, https://sdv.eclipse.org/
[10] Eclipse Zenoh project, https://zenoh.io/
[11] DCP standard, https://dcp-standard.org/
[12] IRT Saint Exupéry, https://www.irt-saintexupery.com
[13] IRT project proposal (draft), https://workspace.accellera.org/wg/FSS-WG/document/13110 (Accellera members only)
[14] IEEE SA Autonomous Driving Working Group (ADWG), Simulation, Testing, Verification, and Validation (STV2) of Autonomous Driving, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10709902
[15] FSS Proposed Working Group, General requirements, https://workspace.accellera.org/wg/FSS-WG/document/12342 (Accellera members only)

Annex A Terms, definitions, abbreviations and acronyms

This annex lists the terms, definitions, abbreviations, and acronyms used in this document. If applicable the standard defining a term is listed.

A.1 Terms and definitions

ecosystem: the constellation of different environments forming a coherent arrangement for study and analysis.

environment: the constellation of different elements (e.g., simulation models, physical devices) forming a coherent set for study and analysis. See also: simulation environment, virtual environment, physical environment.

federated simulation: an ecosystem which contains a set of distributed simulation or physical environments which together form a simulation ecosystem to enable study and analysis of the entire system.

heterogeneous simulation: simulation which combines different modeling and simulation concepts (e.g., different models of computation)

infrastructure: collection of physical hardware components or virtual components which form the foundation of a physical or virtual environment.

physical environment: environment where physical hardware components (or equipment) are combined forming a real-world process or system. See also: simulation environment.

simulation environment: a virtual environment created to imitate the operation of a real-world process or system over time by means of simulation. See also: virtual environment.

system: combination of interacting elements organized to achieve one or more stated purposes. [ISO/IEC/IEEE 15288:2015]

systems-of-systems: a large system that delivers unique capabilities, formed by integrating independently useful systems. [ISO/IEC/IEEE 24765:2010]

virtual environment: see simulation environment.

A.2 Abbreviations and acronyms

AMR	Autonomous Mobile Robot
API	Application Programming Interface
ASIC	Application Specific Integrated Circuit
DCP	Distributed Co-Simulation Protocol
FMI	Functional Mockup Interface
FSS	Federated Simulation Standard
FSSWG	Federated Simulation Standard Working Group
FSUG	Federated Simulation User Group
HAL	Hardware Abstraction Layer

HIL	Hardware in the loop
HW	Hardware
ISS	Instruction Set Simulator
LAN	Local Area Network
MCU	Microcontroller Unit
MIL	Model in the loop
OS	Operating System
OSI	Open Systems Interconnection
PIL	Processor in the loop
PGVIL	Proving Ground Vehicle in the loop
POC	Proof of Concept
RTOS	Real Time Operating System
SIL	Software in the loop
SMP	Simulation Model Portability
SOC	System on a Chip
SW	Software
TLM	Transaction Level Modeling
VIL	Vehicle in the loop
WAN	Wide Area Network
WG	Working Group
XIL	Everything in the Loop