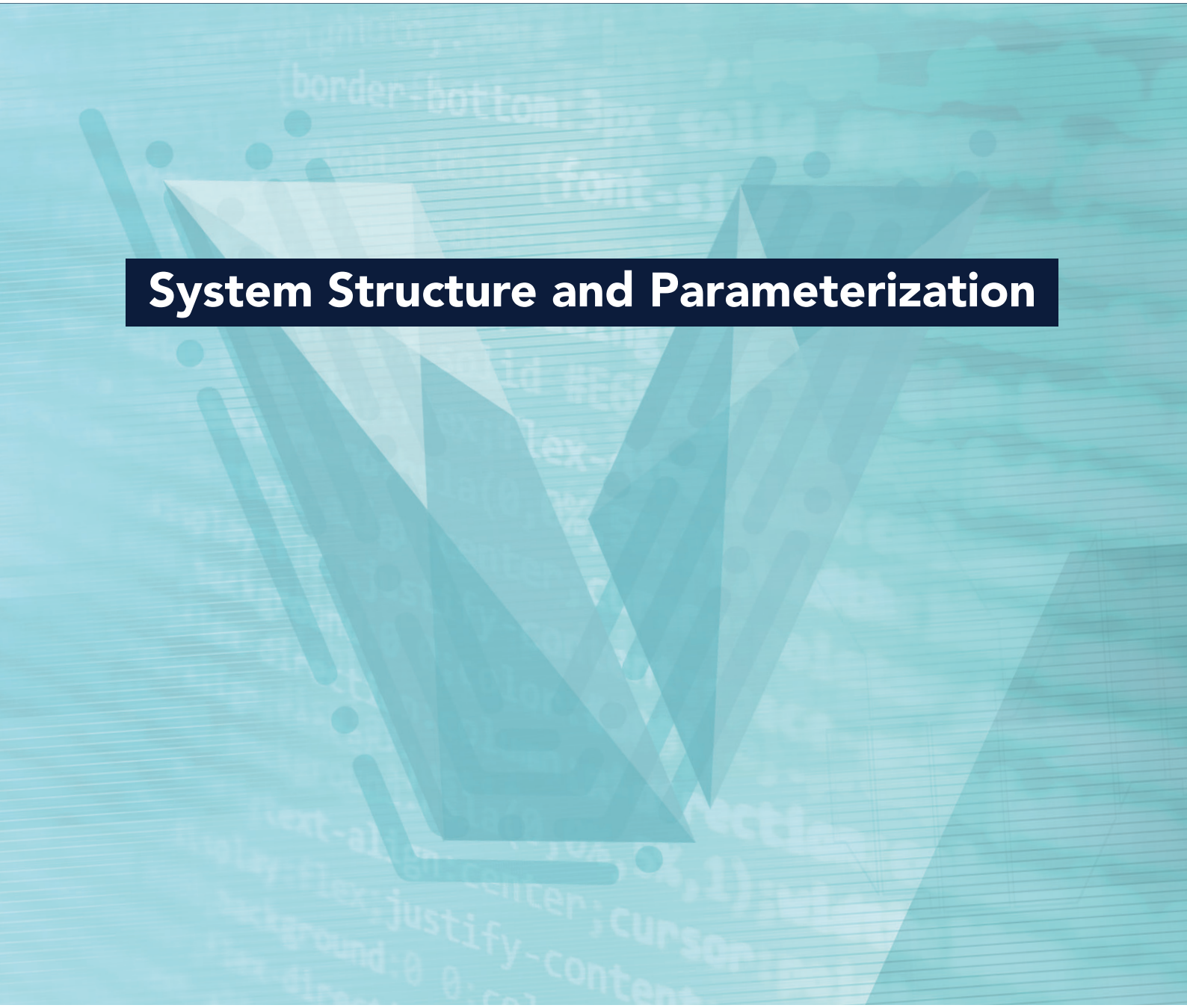
prostep IVIP

White Paper



System Structure and Parameterization

prostep ivip White Paper

Smart Systems Engineering – SmartSE

System Structure and Parameterization

Table of Content

1 Introduction 2
1.1 System Structure and Parameterization (SSP) 2
1.2 Basis for SSP 3

2 Industrialization of SSP 4
2.1 Challenges faced in the past 4
2.2 Data exchange using SSP 4
2.2.1 SSP data exchange in cross-company system development processes 6
2.2.2 Simulation-specific storage of SSP packages and simulation results 7
2.2.3 SSP-based data management for developing mechatronic and autonomous systems 7
2.2.4 Standardized exchange and delivery of component parameter sets 9

3 SSP demonstrator 9
3.1 Motivation for the SSP demonstrator 9
3.2 Details of the SSP demonstrator 9

4 Benefits of using SSP 12

5 Summary, roadmap and next steps 13

6 References 13
6.1 Survey Questions 13
6.2 Survey 13

Figures

Figure 1 Internal structure of a system structure package (image: https://ssp-standard.org/) 3

Figure 2 Cross-company FMU/SSP data exchange 6

Figure 3 Simulation-specific SSP storage 7

Figure 4 Simulation data management based on SSP information structures 8

Figure 5 Systems Modeling Language (SysML) architecture of a Mars Rover 9

Figure 6 Workflow for developing a virtual prototype of a Mars Rover 10

Figure 7 Co-simulation of the Mars Rover virtual prototype in Model.CONNECT 11

Figure 8 Figure 8: Benefits of using SSP according to the survey 12

1 Introduction

The Smart Systems Engineering (SmartSE) project group focuses on distributed collaborative system development between partners using systems engineering methods and standards. A key aspect of this collaborative development is the exchange of simulation models between partners. That is why the FMI standard was of interest to the SmartSE group from the very start and remains so. Complex systems are, however, increasingly being represented by multiple models. This means that numerous individual models are interconnected to form a model network. If these model networks are to be exchanged in practice, it is essential that not only the models themselves but also their interconnections and parameterization are exchanged. This is something that the SSP standard provides for. This white paper is intended to provide an introduction to the SSP standard, the use cases addressed and the benefits of using the standard. The description of a demonstrator supports this objective.

1.1 System Structure and Parameterization (SSP)

The System Structure and Parameterization (SSP) standard was developed by the Modelica Association with the aim of creating and making available a data exchange format for describing simulation-capable system structures based on functional mock-up units (FMU). A single FMU can provide technical and organizational metadata for the FMU in question. However, when it comes to simulating complex systems, individual FMUs are often interconnected to create FMU networks in which the FMUs can be structured both hierarchically and functionally. Functional structuring refers to the unit comprising an FMU and its input and output ports, via which information can be exchanged between the FMUs. Cross-FMU structure and meta information cannot be described or transferred using the Functional Mockup Interface (FMI) standard alone. This is where the SSP standard comes in. The standard is intended to describe the overarching structures and thus makes it possible for them to be exchanged. SSP is an umbrella format comprising multiple subformats that are bundled together in a zip container containing all the components of the system structure package (SSP).

- The system structure description (SSD) is a mandatory part of every system structure package that is used to describe hierarchical and functional structures of a whole FMU network.
- A system structure parameter values (SSV) element is used to describe parameters and external parameter sets that can be applied to an FMU, thus parameterizing it.
- A system structure parameter mapping (SSM) is used to describe parameter mappings that may be required to parameterize FMUs.
- A system structure signal dictionary (SSB) is used to describe signals.

The SSV, SSM and SSB subformats are optional and can be referenced from the SSD subformat. The included FMUs are also referenced from the SSD. Figure 1 shows the structure of a system structure package and also provides a look at the structure of the SSD subformat. The contents of the SSV, SSM and SSB subformats can in principle also be embedded directly in the SSD.

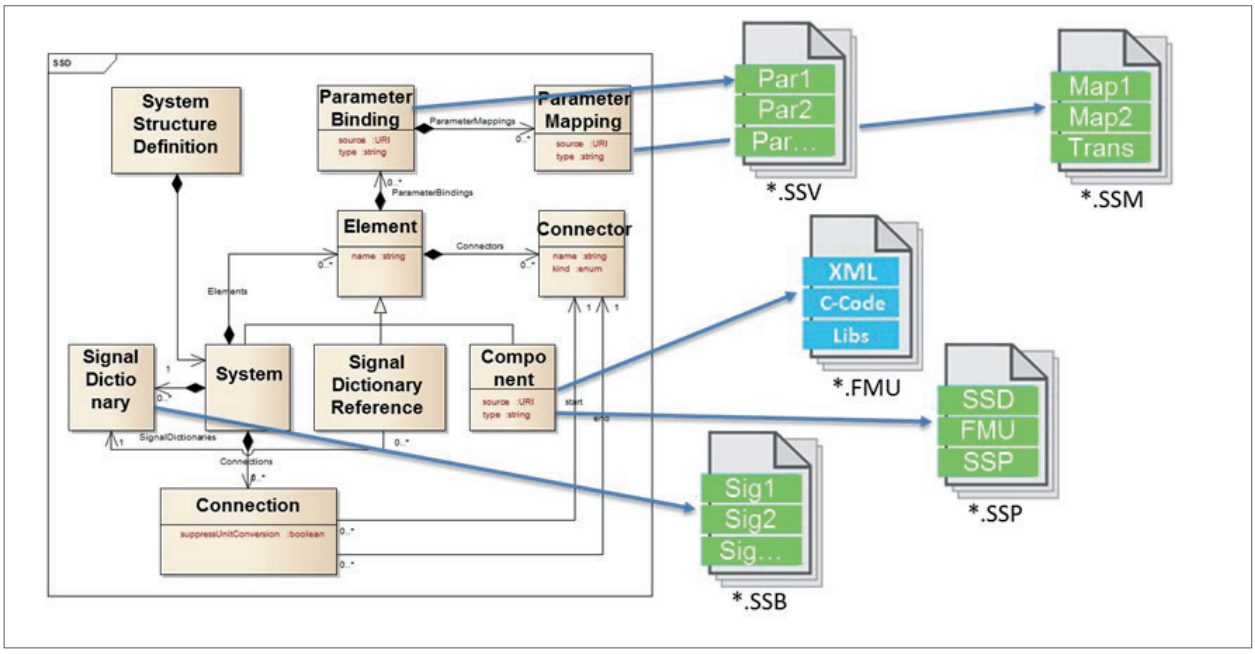


Figure 1: Internal structure of a system structure package (image: https://ssp-standard.org/)

1.2 Basis for SSP

The SSP standard is based on existing standardized formats for structuring information and for data coding, such as XSD (W3C XML Schema Definition Language) and XML (W3C Extensible Markup Language: W3C XML). SSP also references the Modelica Association's Functional Mock-up Interface (FMI) standard.

- XSD: XML Schemas express shared vocabularies and allow machines to carry out rules made by people. They provide a means of defining the structure, content and semantics of XML documents. (Reference: https://www.w3.org/XML/Schema)
- XML: Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the web and elsewhere. (Reference: https://www.w3.org/XML/)
- FMI and FMU: The Functional Mock-up Interface (FMI) is a free standard that defines a container (FMU) and an interface (FMI) for exchanging dynamic models using a combination of XML files, binaries and C code zipped into a single file. It is supported by over 150 tools and maintained as a Modelica Association Project on GitHub. The code has been released under a 2-clause BSD license; the documents under a CC BY-SA license. (Reference: https://fmi-standard.org/)

2 Industrialization of SSP

The initial version of the SSP standard was published by the Modelica Association on 5 March 2019. It made available a normative specification that can be used as the basis for implementing the standard in relevant IT tools. The relevant IT tools can be roughly divided into two categories: authoring tools that can be used to create, utilize and/or process data and data management tools in which data can be managed. Whether implementations are actually performed in industry largely depends on user acceptance. Acceptance can be achieved by, among other things, the fact that using the standard in industrial applications provides added value to the approach already being used. This added value is presumed. The following sections are intended to illustrate the potential added value of using FMUs and show how the use of SSP as a metadata format can improve FMU-based simulation processes. The benefits of SSP will be presented both from the perspective of the standard and from the perspective of the industrial processes.

2.1 Challenges in the usage of FMU for system simulation

Reusing simulation models from different authoring tools across the "V" development process is a basic idea behind model-based development and a key factor in improving the efficiency and the reliability of the process. The Functional Mock-up Interface (FMI) is already widely accepted as the de-facto standard for building component and sub-system models and allowing them to be integrated in larger co-simulation system models. Nevertheless, transferring a whole system set-up, including all the connections between the elements and all the parameterization variants, from one environment to another (i.e. from one tool to another) or from one development stage to the next (e.g. from SiL to HiL), still poses a challenge. Rewiring, converting units and mapping parameters were routine tasks for system simulation engineers integrating components from different teams or external suppliers. These tasks not only involve a huge amount of time but also a large number of hidden risks. A faulty set-up is not usually discovered until a system fails following its launch on to the market. It is obvious that what is needed is an improvement in the form of a standard, or rather a group of standards, that would complement the FMI standard and describe everything related to the configuration and parameterization of an integrative co-simulation set-up beyond the physical content of the building blocks.

2.2 Data exchange using SSP

As described in section 1.1, SSP comprises a set of XML-based formats that can be used to describe a network of FMU component models together with their signal flow and parameterization. This data can be packaged using a zip-based packaging format for efficient distribution of whole system structures.

The SSP standard itself describes five use cases, which are outlined briefly below. The following five sections are excerpts from the SSP standard specification.

Designing a simulation structure (source: SSP Standard Specification 1.0)

For the simulation of complex systems, first a design of this system should be created. From a simulation perspective, each component has to be described with its inputs and outputs and its required parameters. This can be done using SSP by defining the wrapper of this component with an empty "Component" element comprising the connectors for the inputs and outputs and the component's parameters.

The interaction of the components is defined by the connections. Connections in SSP are always causal. Connections can be made directly between components or via signal dictionaries. A signal dictionary is a collection of signals similar to a bus concept (e.g. like a CAN bus). During the design phase of a system, a signal dictionary can be a good way to predefine the available signal connections.

If the system has global parameters that are to be propagated to multiple components, the definition can also be made at system level. The mapping to the parameters of the components can be realized either using connections or through parameter bindings that can include parameter mappings.

SSP as definition of component interfaces and parameterization as template (source: SSP Standard Specification 1.0)

The main result of the design of the complete simulation structure is the definition of all needed components and the used parameterization structure. Each component can be used as a design template for the implementation – including the wanted parameters. The system designer extracts each component into a separate SSP file as preparation for the implementation and sends it to the implementor.

The implementor of the component can import this SSP file as a template in their authoring tool and directly code the behavior using the defined input signals and the definition of the parameters to calculate the defined output signals. After completion of the implementation the component can be returned as a running entity in an SSP package file for insertion into the complete system structure by the integrator. The integrator can decide whether to merge the components from different sources into one file or use the components as references by using the appropriate mechanisms in SSP to just link to the original SSP files. The latter approach has the benefit that the components can be used "untouched" and any "warranty" given by the author of the component is not corrupted. Even traceability information can be retained this way.

SSP as central parameterization description and syntax for other parameterization databases (source: SSP Standard Specification 1.0)

A good system design can be used for various applications. The structure keeps the same for all these applications. The parameter settings are used to differentiate the applications. Therefore a good parameterization concept is important to facilitate this reuse. SSP supports the creation of a central parameterization structure for entire systems. The SSP parameter data model can be used to integrate parameters from various sources, including external parameter databases, which can export their parameter data as System Structure Parameter Value (SSV) data sets. Through the URI-based addressing mechanism, tools can support direct access to such databases from system structure descriptions.

SSP as particular instances of ready-to-simulate simulation systems (source: SSP Standard Specification 1.0)

After implementation of all components and provisioning of the parameter settings for a particular system everything is in place for running simulations. All these entities can be stored in one single SSP package, which can be imported by the executing system for running the simulation. Depending on the execution system it might be necessary to define additional settings for the solver or other execution algorithms. The core SSP standard does not include these execution-specific settings, but layered standards will be defined to include those settings.

These complete instances of simulation systems can also be used as an archive for traceability purposes.

SSP for reuse of system structure elements during development (source: SSP Standard Specification 1.0)

As an example, a system structure defined originally for software-in-the-loop testing can also be reused for hardware-in-the-loop testing. Where FMI enables the reuse of individual models across platforms, SSP enables the reuse of complete systems and subsystems, including their configurations, basic layouts, and parameters.

Data management tools can control the lifecycle of the SSP-based system structures. There is an increasing desire to reuse environment models to provide proven, consistent solutions for the validation of controller models in different projects and development stages (e.g., for virtual validation and HIL simulations).

Data management environments provide capabilities for managing model compositions, handling variants of systems and managing the parameter and signal interfaces of the different model systems.

The SSP approach enables the sharing of standardized system structure descriptions between data management, integration and configuration tools for SIL, MIL and HIL scenarios.
The following sections are intended to show how these use cases can be implemented in industrial practice and how the potential of these use cases can be exploited.

2.2.1 SSP data exchange in cross-company system development processes

Simulation models are also exchanged in the context of development partnerships. These can be both native models and FMUs. FMU metadata and FMU structural information are typically also exchanged when FMUs are exchanged. The use of system structure packages is ideal for this purpose because this is exactly what SSP was developed for. System structure packages do not necessarily have to contain FMUs and the system structure description (SSD), an SSP subformat for storing metadata and structure information, does not necessarily have to contain references to FMUs. This provides enormous potential for controlling FMU data exchange both within a company and across different companies.

This potential lies primarily in the approach of, on the one hand, using an SSD as a structure template for integrating FMUs in a given system structure and, on the other hand, using SSPs to specify the signal definitions to be used and the ports and parameters so that these can be used consistently and in a uniform manner throughout the development group. This type of development group might begin its work by creating a system structure package that defines all of these specifications but which does not yet include any FMUs. This SSP will be made available to the partners, who can then integrate their FMUs in the package and either pass the SSP on to other partners or send it back to the client. In addition, the SSP that has been sent can be used by the recipient to for example import the simulation structures, the FMU, parameters and their metadata into data management systems that have been designed for this purpose.

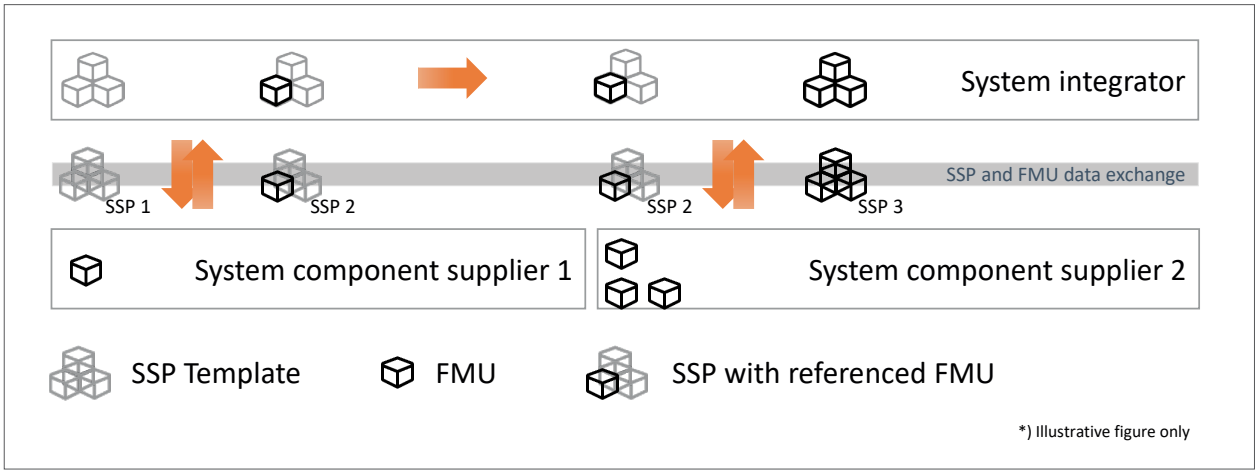


Figure 2: Cross-company FMU/SSP data exchange

Figure 2 shows an example of cross-company FMU data exchange using SSP files. In this example, the overall scenario involves the integration of multiple components from multiple suppliers in an overall system structure by a system integrator.

The system integrator first sends an SSP without FMUs to a component supplier and asks that an FMU of their component is integrated. The integrator can store the required information, for example about the ports and signals to

be used, in the SSP. The component supplier sends the desired FMU, which has now been integrated in the SSP at the intended location, back to the integrator. This interaction is then repeated for the three other FMUs belonging to component supplier 2, who also integrates the FMUs in the SSP at the designated points. Ultimately, an SSP that contains all the FMUs is returned to the integrator. The integrator can import the SSP containing the FMUs and all the structural information and metadata into their data management system, provided that the system has been designed for this purpose.

2.2.2 Simulation-specific storage of SSP packages and simulation results

Even if no simulation data management solution that can break down SSP information structures and manage them granularly is available, a system structure package is well suited for making the relevant simulation-specific information, such as models, parameters, signals and their structural relationships, available in a compact way. In this case, benefits are gained from the fact that the SSP standard not only defines a data format for mapping information and information structures but also a packaging format that allows all the information to be handled using a monolithic approach in the form of a zip file. This zip file can in turn be managed in a data storage solution or document management systems if no simulation data management system is available. All the information required for the simulation, with the exception of the solver settings, can be compiled within the zip container and made available for simulation. It can not only be used across different companies but also within a single company. This can be understood in the broadest sense as a simulation-specific configuration that can be versioned and archived as a baseline. Figure 3 shows a simple representation of a concept for storing and versioning simulation-specific configurations of simulation data in the form of versioned SSP zip files.

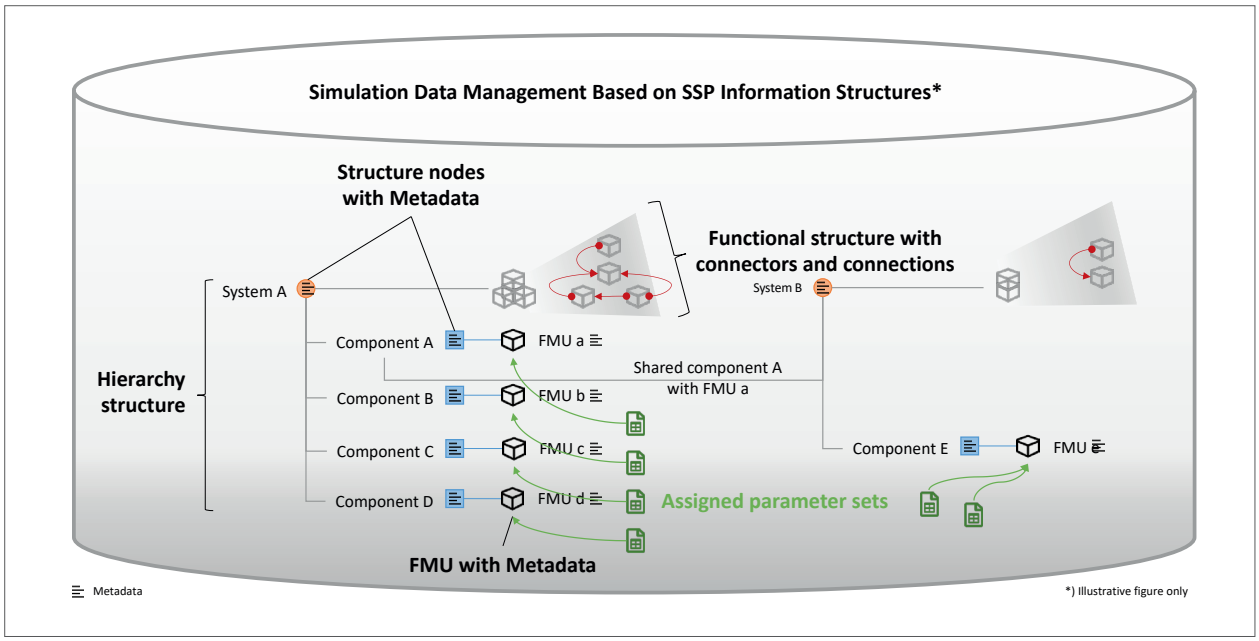


Figure 3: Simulation-specific SSP storage

2.2.3 SSP-based data management for developing mechatronic and autonomous systems

Functional mock-up units should be seen as objects of simulation data management in the same way as the simulation models used for crash simulations, flow simulations, the simulation of mechanical multibody systems, thermal simulations, or noise and vibration simulations. Models for the latter simulations are managed to an increasing extent in simulation data management systems designed for this purpose. These models usually have a monolithic structure, i.e. one or more models are created, used and managed accordingly for a given simulation task. The traces along the

process chain are also clearly mapped in the data management systems, i.e. it is possible to determine which models were used under which simulation conditions to generate which simulation results.

The model structures in what is referred to as equation-based simulation (ESB) are usually more marked and less monolithic. The structural relationships between the FMUs are reflected to a greater extent. This can occur in three ways:

- FMUs are structured in hierarchies, similar to assembly structures in mechanical development, i.e. multiple FMUs that represent the components of a system together form a system structure from a simulation perspective.
- FMUs are connected functionally, either directly or via signals that exchange or transfer information between the FMUs. In this context, the SSP standard refers to "Connections" that link two FMU connectors.
- Furthermore, there is usually a separation between the internal structure of an FMU and its parameterization. The assignment of a parameterization to an FMU is also a type of structural relationship.

In figure 4, these structural relationships are shown in the context of simulation data management.

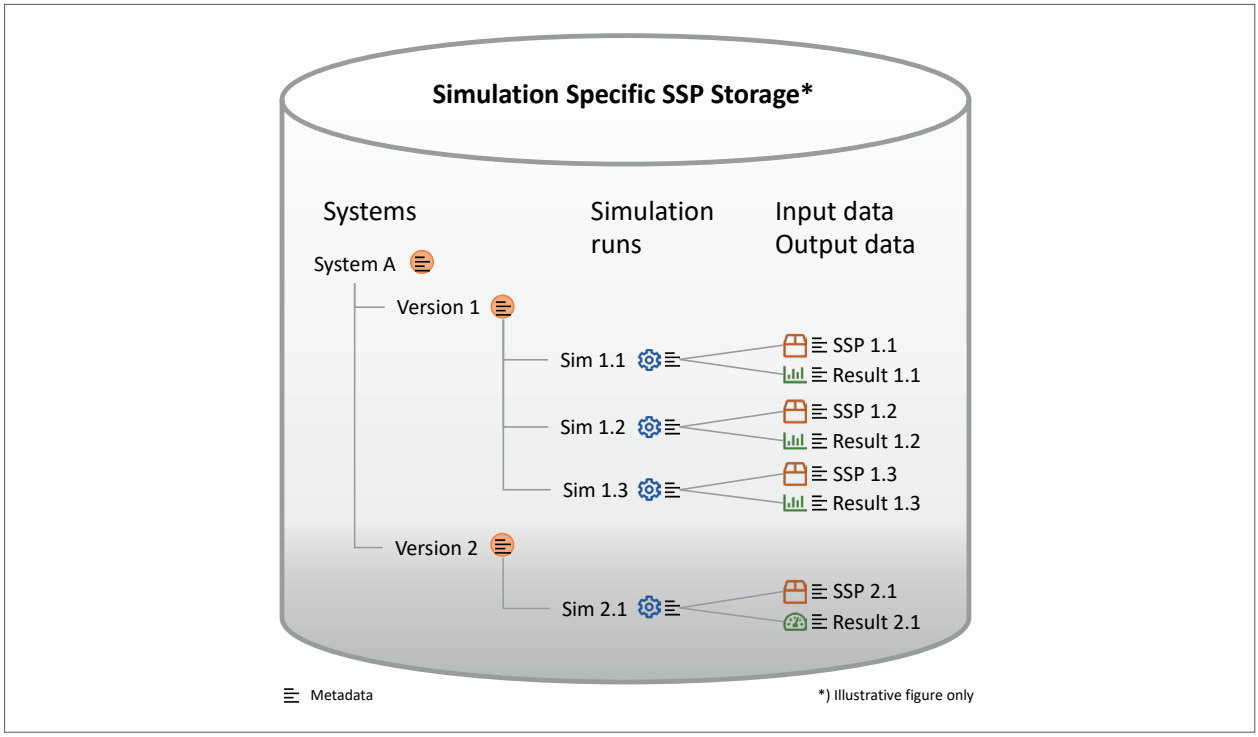


Figure 4: Simulation data management based on SSP information structures

Revealing these structures at data management level makes it possible to exploit considerable potentials that are also familiar from the context of product data management and product lifecycle management.

- Reusability of component FMUs
- Traceability of FMUs used in simulation structures
- Reuse of parameters and parameter sets
- Scaling of structurally identical models using different parameter sets

2.2.4 Standardized exchange and delivery of component parameter sets

SSP offers the option of separating models from parameter sets and reciprocally replacing parameter sets for the models, thus supporting flexible and efficient simulation using different parameter sets but the same internal model structure, e.g. for scaling component models. The SSV subformat can be used to represent and store component parameters in a standardized format. It can be used in a number of different ways.

One possible scenario would be for component manufacturers to provide data sheets for technical components (e.g. small electric drives used in vehicles as drives for technical components) as *.ssv files, which can then be downloaded from the website either individually or all at once. A similar use case would be to use a *.ssv file to parameterize virtual ECUs.

Depending on whether you only want to describe the data sheets or provide parameterized models, you would either use only the *.ssv file without an SSP package or an SSP package that contains an SSD file, the FMU and an SSV (parameter) file.

3 SSP demonstrator

3.1 Motivation for the SSP demonstrator

The prostep ivip SmartSE project group has developed a demonstrator to demonstrate use of the SSP standard when exchanging simulation models in cross-company system development processes. The demonstrator is intended to help provide a clear overview of the possibilities that the SSP standard offers. It demonstrates the use of SSP in a collaborative scenario involving a joint simulation with multiple development partners. Individual components for the simulation are developed by different partners and exchanged via SSP. The demonstrator makes it easy to reconstruct this scenario. In addition, the models and SSPs involved can be used by other partners to replicate the described collaboration process in their own company. The collaboration process shown is the one between a system integrator and multiple component suppliers introduced in section 2.2.1 .

3.2 Details of the SSP demonstrator

The starting point for the demonstrator is a Systems Modeling Language (SysML) model that represents the system architecture of a Mars Rover. The system comprises multiple modules (sensor, perception, planning and plant model) and the signals that need to be exchanged between the modules during runtime. The SSP standard uses the system structure description (SSD) to define co-simulation configurations comparable to the SysML architecture. As both standards are based on the XML standard, it is only a matter of converting one format into the other. (Please note that SysML has different flavors that requires modified conversion scripts for each vendor flavor).

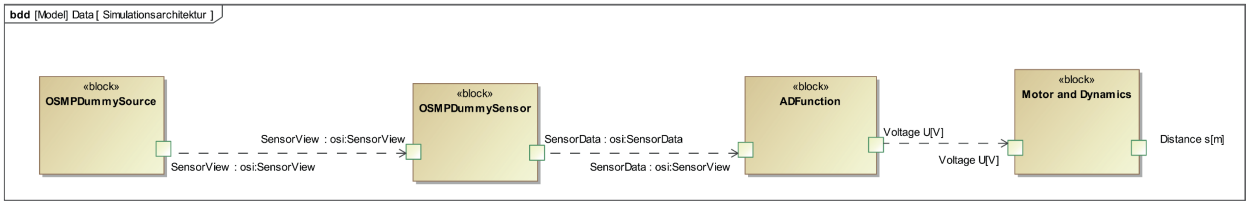


Figure 5: Systems Modeling Language (SysML) architecture of a Mars Rover

Having converted the formal system architecture into the SSP standard, the first version of the SSP file is a representation of the co-simulation configuration that does not contain any models (Figure 6 -1). As there is a huge variety of simulation tools, the FMI standard provides a standardized method for bundling simulation models and solvers into black-box containers called FMUs. As the global system architecture has already been defined, the interfaces between the subsystems are specified from the beginning. This means that the project partners can develop their models knowing they will fit into the co-simulation topology.

Figure 6: Workflow for developing a virtual prototype of a Mars Rover

The Open Simulation Interface (OSI) is an emerging standard for the co-simulation of environment and traffic simulators, sensor models and autonomous driving functions. In this concrete demonstrator use case, the empty SSP package was filled with two OSI Sensor Model Packaging (OSMP) FMUs (Figure 6 - 2). The OSMPDummySource generates random traffic objects that represent a straight two-lane road. The OSMPDummySensor is an ideal sensor that translates the OSI SensorView from the dummy source into an OSI SensorData stream that contains a list of detected moving objects. The source code for these FMUs can be found in the official OSI GitHub repository [1]. At the same time, the empty SSP system definition was passed on to the system component supplier, who was tasked with developing the simulation module that represents the motor and dynamics (Figure 6 - 2). This parallelization is made possible by the fact that the interface between the different modules has already been specified by the system architect. This guarantees that the modules from different suppliers or departments can be integrated in the whole system without problems. The last step was again performed by the system integrator and involves several tasks. First, the SSP packages from steps 2 and 3 were merged into a single system. This task has not yet been automated and needs to be done in a tool that supports the SSP standard [2]. In addition, an autonomous driving function that complies with the interfaces as specified needed to be developed. The function was developed in C++ and is based on the OSMPDummySensor. It is a simple prototype that filters the list of moving objects detected for the closest vehicle and outputs a control signal. The final result is a SSP package containing all the modules integrated as standardized simulation containers (FMUs). This package can be used as a neutral input file for co-simulation engines for testing the functionality of the virtual prototype.

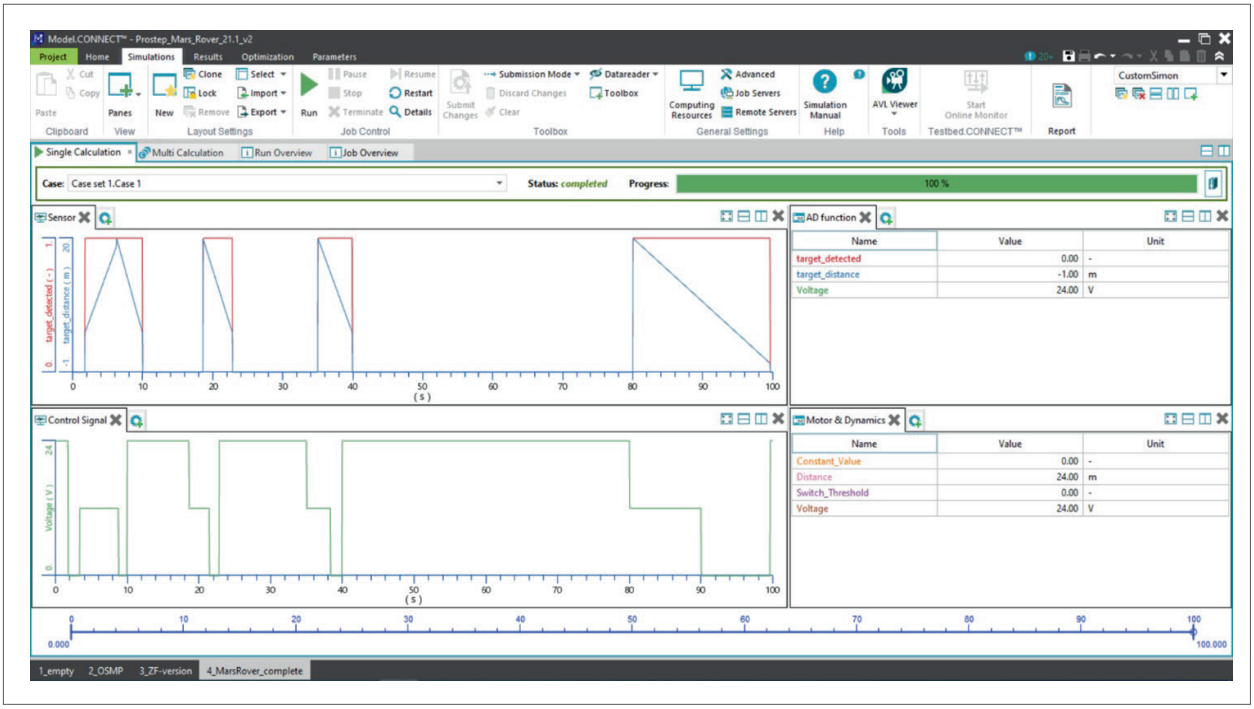


Figure 7: Co-simulation of the Mars Rover virtual prototype in Model.CONNECT

Figure 7 shows the results of the functional test involving the SSP package. As the OSMPDummySource generates random traffic objects, there is no clear use case that can be mapped to a Mars mission. The diagram at the top shows the sensor signals. The red line indicates whether or not a target is detected and the blue line indicates the relative distance between the detected target and the Mars Rover over the simulation time. Depending on the distance of a detected object, the Mars Rover receives a "Voltage" signal that controls the electric engine that moves it forward (green line in the bottom diagram). This simplified setup has three voltage levels.

On the whole, it was shown that the SSP standard improves collaboration between companies and departments. The main advantage is the up-front system definition with clearly specified interfaces between the components. This allows for parallelized development and ensures seamless system integration. Providing an empty (and perhaps reduced) SSP package to suppliers can help create a clear common understanding of the development task. Although figure 7 shows an ideal collaboration scenario, in reality models are more complex and are developed in multiple iterations. This means that the linear process shown above has to be turned into a continuous integration loop, in which every new model variant is added to the SSP package. As SSP packages are a combination of small configuration files and large binary archives, a smart versioning management system is required. One option would be to manage an unpacked SSP package in a Git repository. This is however a key topic that needs to be addressed within the framework of further developing the SSP standard.

4 Benefits of using SSP

The aim of a survey conducted within the framework of the SmartSE project group was to highlight and evaluate the benefits of using SSP. Twelve participants were interviewed during a workshop, one of whom was from the research community and eleven from industry. All the participants are part of the Smart Systems Engineering project group. They are familiar with the SSP standard and have an interest in the industrialization of SSP. The results of the survey can therefore be considered representative of the project group. For each question, the answers 0 to 3 were available for rating the added value of SSP. The answer 0 indicates no benefit, while the answer 3 indicates a very great benefit. The following figure represents the averaged responses from all participants.

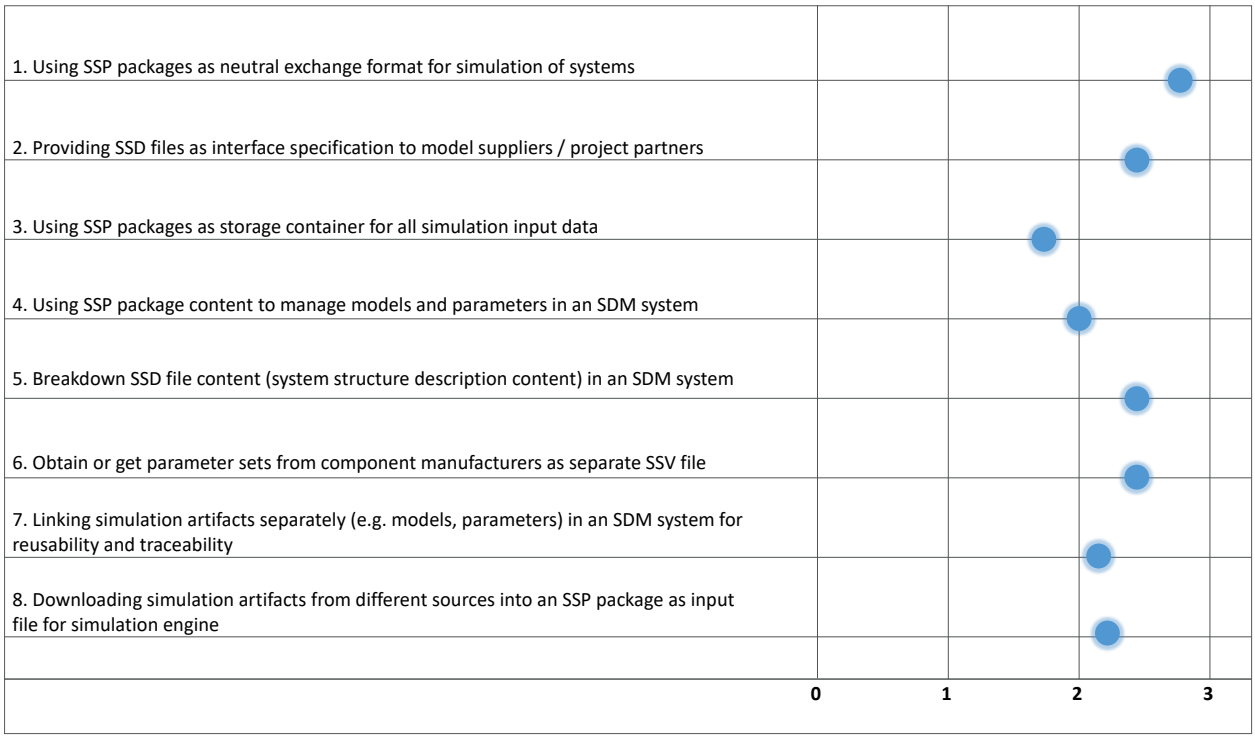


Figure 8: Benefits of using SSP according to the survey

Generally speaking, it can be said that an added value of SSP could be observed across all the questions. The respondents see the greatest added value (2.75) when SSP packages are used as a neutral exchange format for system simulations. Using SSD files as interface specifications is also seen as offering great added value (2.42), both for exchange with suppliers but also for exchange between project partners. Respondents see the same added value in breaking down the contents of the SSD files in SDM systems, as well as in obtaining parameter sets from component manufacturers using separate SSV files. With an average response of 2.25, this is followed by downloading simulation artifacts from different sources into an SSP package as an input file for the simulation to be performed. The traceability and reusability of simulation artifacts by means of separate links within the SDM systems was rated with an average value of 2.17. Respondents saw the least added value in using SSP package content to manage simulation models and parameters in SDM systems (2) and in using SSP packages to store all simulation input data (1.75).

5 Summary, roadmap and next steps

The SSP standard tackles the challenges currently posed by collaboration in the simulation industry as it provides a neutral format that can be used to define a "ready-to-run" co-simulation configuration. However, its use in practice raises the question of how to deal with the different versions of the SSP packages when multiple parties are involved. So, how good an idea is SSP in reality? The best way to answer this question is to create a realistic demonstrator using the tools that already support SSP and define and run a heterogeneous co-simulation configuration using different model sources and parameterization sets.

This white paper first of all presents the basics of SSP and its history. It then takes a closer look at conceivable approaches to the industrial implementation of solutions involving the topics "data exchange in cross-company system development processes", "simulation-specific storage of SSP packages and simulation results", "SSP-based data management for the development of mechatronic and autonomous systems" and the "standardized exchange and delivery of component parameter sets", based on the use cases described in the standard. A demonstrator illustrates use of the standard by way of an example. Finally, the results of a survey conducted within the project group among the project participants are presented and serve to underscore the fact that the value added by the standard is definitely seen.

However, complex projects will require a version management system that keeps track of all model versions and allows suppliers to work in parallel. This includes automation of the development process in collaborative environments, taking account of concrete challenges in industry (e.g. data management, IP protection, traceability, etc.), and the evolution of the SSP to provide support. Some of the use cases presented suggest possible data management solutions for this. This may be one of the focal points of further development of the standardization framework.

6 References

https://github.com/OpenSimulationInterface/osi-sensor-model-packaging
https://ssp-standard.org/tools/

6.1 Survey Questions

Please rate the benefits of SSP you see for...

1. Using SSP packages as neutral exchange format for simulation of systems
2. Providing SSD files as interface specification to model suppliers / project partners
1. Using SSP packages as storage container for all simulation input data
2. Using SSP package content to manage models and parameters in an SDM system
3. Using SSD file content to manage hierarchical system structures in an SDM system
4. Using SSD file content to manage model connections in an SDM system
5. Obtain or get parameter sets from component manufacturers as separate SSV file
6. Linking external FMUs in SSD instead of embedding them in the SSP package

Response options: (no benefit) 0 - 1 - 2 - 3 (very high benefit)

6.2 Survey

https://docs.google.com/forms/d/e/1FAIpQLSePN9CAxJ-DO5BV4ovZ0krQyu1MD9vKypIOlXeFckeWkj3IJw/viewform?usp=sf_link

prostep IVIP

prostep ivip association

Dolivostraße 11
64293 Darmstadt
Germany

Phone +49-6151-9287336
Fax +49-6151-9287326
psev@prostep.com
www.prostep.org

ISBN 978-3-948988-19-7
Version 1.0, 2022-1