SSP in a Modelica Environment

Dag Brück

Dassault Systèmes, Lund, Sweden, dag.brueck@3ds.com

Abstract

System Structure and Parameterization (SSP) is a tool independent standard to define complete systems. Dymola now supports import and export of SSP files; this paper describes how the SSP support was implemented and discusses some of the constraints and unavoidable compromises.

Keywords: Modelica, SSP, implementation

1 SSP and supporting processes

System Structure and Parameterization (Mai 2019), known as SSP, is a tool independent standard to define complete systems. In a typical use of SSP, the system description consists of several interconnected Functional Mockup Units (FMUs), nested system descriptions, and their parameterizations and other related data, as shown in Figure 1.

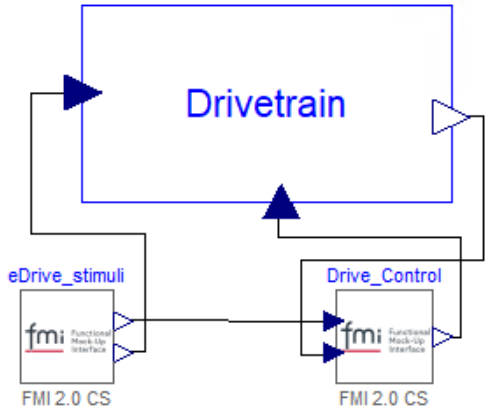

Figure 1. Top-level system comprising two FMUs and a drivetrain subsystem described hierarchically.

SSP is suitable for representing simple model structure with additional support for parameter sets and (some) simulation setup. However, SSP was designed to meet several needs:

- SSP for designing a simulation structure. Components are described with its inputs and outputs and its required parameters, but no behavior.
- SSP as a template for implementation, based on the interfaces and parameters defined in the previous step.
- SSP as central parameterization description and database (Hällqvist et al. 2021). Several data sets can be defined and documented in a portable manner.
- SSP as a ready-to simulate system description. This is the main use we envision in our environment.
- SSP for reuse of system structure during different phases of development, for example, an SSP defined originally for software-in-the-loop can also be reused for hardware-in-the-loop testing.

A key objective is that the SSP files can be transferred between multiple environments for architecture specification, detailed design and model implementation, or post-processing analysis. A demonstrator using such a multi-tool workflow was developed by ProSTEP ivip Smart System Engineering (Rude et al. 2021), as shown in Figure 2. For this application, several tools were used in a collaborative manner: Dymola (Dassault Systèmes 2019; Elmqvist 2014), easySSP (eXXcellent 2022), PMSF FMI Bench (Mai 2023) and Tracy (Vettermann 2021).

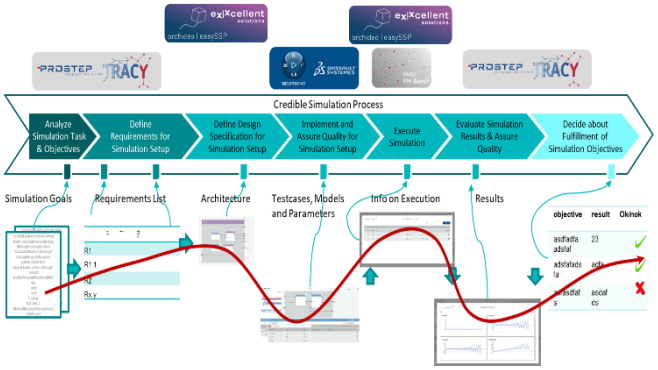

Figure 2. Traceability workflow using SSP from ProSTEP ivip Smart System Engineering.

SSP is commonly stored as a zip-file containing several files and a hierarchy of directories. A simplified view of the SSP structure is given in Figure 3.

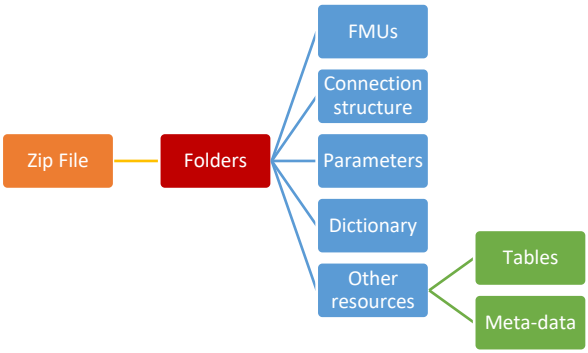

Figure 3. Simplified structure of the SSP file.

SSP is open for any extensions using standardized or vendor-specific annotations and directories with additional meta-data. Under the catch-phrase "Credible Decision Process" (CDP), the SSP community has expressed great interest in adding standardized meta-data. The SET Level project has developed a process framework for integrating simulation into the development and validation of system models (Heinkel and Steinkirchner 2022a; Heinkel and Steinkirchner 2022b). The process supporting CDP is shown in Figure 4.

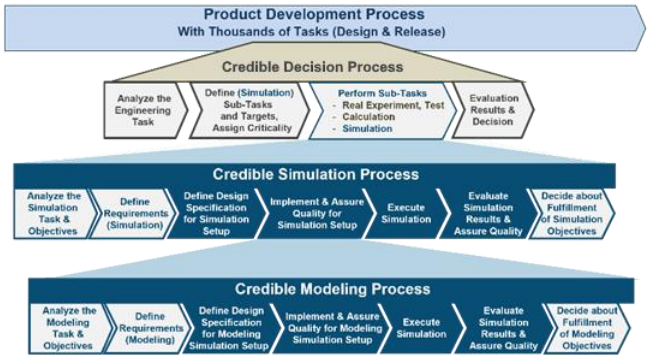

Figure 4. SET Level credible decision process.

An extensive Simulation Resource Meta Data (SRMD) file format was defined to store meta-data for simulation resources as well as meta-data for models, tools, maps and scenarios, and simulation tasks (Heinkel et al. 2022c). Similar approaches are LOTAR (LOTAR International 2023), MIC (IRT SystemX 2020) and MoSSEC (MoSSEC Project 2021), each shaped by the needs of their respective communities.

It is worth noting that SSP does not specify any simulation semantics, although you can argue that one is implied because much of the definition is based on FMI and the interconnection structure. However, including components of other types (e.g. Modelica) is within the scope of SSP.

SSP is developed as a project in the Modelica Association (MAP-SSP). The first version of the SSP standard was published in March 2019 (Mai 2019), and progress is now made to align SSP 2.0 with the FMI 3.0 specification.

Several tools supporting SSP are now emerging; the remainder of this paper describes how SSP support was implemented in Dymola and discusses some of the constraints and unavoidable compromises. A detailed comparison with other Modelica tools could not be made due to Dassault Systèmes policies.

2 Mapping SSP to Modelica

Dymola is a development and simulation environment for Modelica models, with support for importing Functional Mockup Units (FMUs) and running simulations. This means that much of the support needed for SSP "comes for free", the remaining task being to map SSP structures to Modelica models in a sensible way. The mapping is summarized in Figure 5.

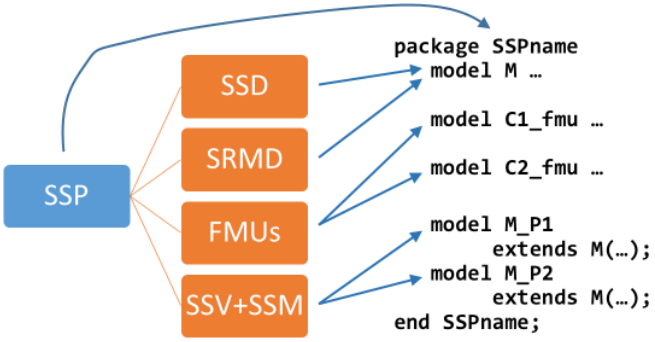

Figure 5. Mapping SSP key elements to Modelica

2.1 Package as container for all artifacts

The first design choice was that everything imported from an SSP would be collected in a single Modelica package in order to prevent contamination of the namespace.

The package name is derived from the SSP name, and certain SSP documentation is stored as package documentation.

2.2 System Structure Description (SSD)

The top-level system, i.e. the `SystemStructure.ssd` file, is converted to a Modelica model, including the internal component, connector and connection structure. A restriction at present is that Dymola only supports one top-level SSD.

Embedded systems in the form of nested SSDs are also imported as locally defined models, and the corresponding component is created in the top-level model.

2.3 Functional Mockup Unit (FMU)

All FMUs embedded in the SSP file are imported into the enclosing package. This import relies entirely on the available FMU import capabilities of Dymola to create a Modelica wrapper model, which means that any combination of ME and CS, or FMI version is supported.

The FMU import processes the port definitions of the FMU and creates the corresponding Modelica connectors.

For that reason, any connector definitions defined in the corresponding SSP component are resolutely discarded; the FMU ports are used instead.

2.4 Parameters and connectors

SSDs can have both parameters and connector definitions. They are mapped to the closest matching Modelica type. For example, a connector of type `ssc:Real` and kind "input" is represented by the MSL type `RealInput`. Units in the SSD are applied to the Modelica model without any checking until the model is translated in Dymola.

The Binary type is used as a special case to represent more complex Modelica types (see below), but in the general case represent a problem for a Modelica environment. There is no natural mapping to an anonymous array of bytes; this is an unsolved problem that SSP shares with FMI.

2.5 Parameter sets

One of the strong points of SSP is that it combines simulation models with parameter sets, stored in one or more SSV-files. There are also optional mapping files (SSM) that allows parameters to mapped to model variables with different names, and in that process perform linear transformations for e.g. unit conversion.

In Dymola, we apply a common Modelica idiom to parameter sets. For each parameter set, we create a new model that extends from the main model defined in the SSP, and then we provide the parameters as a modifiers in the extends-clause. This process allows us to keep the natural structure of the SSP file in a manner that maps naturally to Modelica. It also allows further parameter changes by inheritance.

2.6 Documentation and meta-data

The question of simulation quality, from measurement data via modeling assumption to simulation setup, has received considerable attention. As a result, the Credible Simulation Process (Heinkel and Steinkirchner 2022a) is applied to SSP. A key aspect of CSP is the ability to store meta-data in the form of key-value pairs, following standardized templates defined by organizations or corporations.

Dymola extracts meta-data stored in the proposed Simulation Resource Meta Data (SRMD) format and converts it to model annotations that are displayed and edited as part of the documentation.

2.7 Simulation setup

SSP does not have any simulation semantics in itself, although possibly something can be derived from the underlying reliance on FMU components. The simulation setup in SSP is conversely restricted to simulation start and stop times.

Dymola has the capability to store a more extensive "experiment" annotation in Modelica models, and that information is shared in SSPs by the use of a proprietary SSP annotation (lacking further standardization).

3 Example of an imported SSP

Using a simple SSP example, the XML code with many details omitted is shown in Listing 1.

```
<?xml version="1.0" encoding="UTF-8"?>
<ssd:SystemStructureDescription >
  <ssd:System name="Example">
    <ssd:Connectors>
      <ssd:Connector name="pos"
         kind="output" />
    </ssd:Connectors>
    <ssd:Elements>
      <ssd:Component name="stimulus"
        type=
          "application/x-fmu-sharedlibrary"
        source=
          "resources/StimulusFMU.fmu">
        <ssd:Connectors>
          <ssd:Connector name="y"
             kind="output">
             <ssc:Real/>
             <ssd:ConnectorGeometry ... />
          </ssd:Connector>
        </ssd:Connectors>
        <ssd:ElementGeometry ... />
      </ssd:Component>
      <ssd:Component name="controller" .. />
      <ssd:Component name="drivetrain" .. />
    </ssd:Elements>
    <ssd:Connections>
      <ssd:Connection
         endElement="controller"
         endConnector="ref"
         startConnector="y"
         startElement="stimulus">
         <ssd:ConnectionGeometry ... />
      </ssd:Connection>
      <ssd:Connection ... />
      <ssd:Connection ... />
      <ssd:Connection ... />
    </ssd:Connections>
    <ssd:SystemGeometry ... />
  </ssd:System>
  <ssd:DefaultExperiment stopTime="4" />
</ssd:SystemStructureDescription>
```

Listing 1. Simple SSP example.

The representation of parameters in SSP is currently subject to discussion; see also (Brück 2023).

Importing the SSP file with three FMUs, the generated Modelica model is displayed as Figure 6.

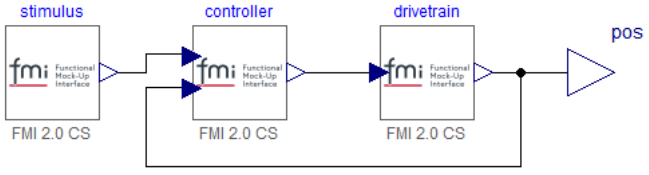

Figure 6. Diagram of SSP file imported into Dymola.

The generated Modelica code, with graphical annotations removed for clarity is shown in Listing 2.

```
model Example "Model for position servo"
  // Connectors
  Modelica.Blocks.Interfaces.RealOutput
      pos annotation (...);
  // Components
  parameter Real k(value=200)
      "Gain of controller";
  parameter Modelica.Units.SI.Time
      T(value=10)
      "Time constant of controller (T>0)";
  parameter Modelica.Units.SI.Radius
      r(value=0.5) "Radius of load";
  parameter Modelica.Units.SI.Mass
      m(value=80)  "Mass of load";
  StimulusFMU_fmu stimulus
      annotation (...);
  ControllerFMU_fmu controller
      annotation (...);
  DrivetrainFMU_fmu drivetrain
      annotation (...);
equation
  // Connections
  connect(stimulus.y, controller.ref)
      annotation (...);
  connect(controller.y, drivetrain.u)
      annotation (...);
  connect(drivetrain.pos, controller.pos)
      annotation (...);
  connect(drivetrain.pos, pos)
      annotation (...);
  annotation (experiment(StopTime=4));
end Example;
```

Listing 2. Generated Modelica code after import.

If the SSP file contains parameter sets (SSV and optionally SSM files), then additional Modelica models are created that provide parameter settings, see Listing 3.

```
model HighGain "Servo with high gain"
  extends Example(k=400, T=12);
end HighGain;
```

Listing 3. Applied parameter set.

It is worth noting that Dymola creates a Modelica wrapper model for each imported FMU. These models are however no different for FMUs in SSP-files from other imported FMUs.

4 Mapping Modelica to SSP

Being a modeling and simulation environment for Modelica, Dymola has to support model export to SSP.

4.1 Models with FMUs

The most straightforward use case is a top-level Modelica model populated with interconnected FMUs as components. This kind of structure can be exported to a standard SSP file under the assumption that you only use types that can be expressed in SSP. For example, Modelica.Blocks.Interfaces.RealInput and RealOutput can be mapped to `ssc:Real` with kind=”input” and “output” respectively. FMUs are copied from wherever they are located into the SSP file’s resource directory.

Similarly, local variables and parameters of built-in types, connections and description string have corresponding attributes in SSP. More advanced concepts such as inheritance (*extends*) and model templates (*replaceable/redeclare*) cannot be represented.

4.2 Need for annotations

SSP has a general escape mechanism called “annotations” that allows a tool to store arbitrary information with a proprietary encoding. Each annotation is tagged by the tools, e.g. com.3ds.dymola.

Dymola uses such annotations to store model documentation, the full experiment (simulation) setup, commands and standardized Modelica figure annotations. It is worth noting that the only simulation setup attributes defined in SSP are start and stop times.

Dymola can store model meta-data (key-value pairs, in user-defined groups). Most meta-data are stored in SSP annotations, the exception being meta-data groups being identified as being in SRMD format. Such SRMD meta-data is stored in separate SRMD files in the SSP extra directory.

Graphical annotations in Modelica models are not stored in SSP. The possibility to store the entire model text as an annotation is not used by Dymola.

4.3 Native Modelica models

A natural extension (in a Modelica tool) is the possibility to use Modelica components in addition to FMUs and export it as an SSP file. Such an extension would increase the expressive power of SSP, for example by connecting physical connectors such as mechanical flanges or fluid pipes.

SSP was designed with such openness in mind, although the specification only mentions FMI by name and provides a restricted set of types appropriated from FMI. A noteworthy point is that SSP does not define any simulation semantics, this follows implicitly from the semantic meaning of connecting its components.

Dymola supports an extended SSP format that has been proposed as part of SSP 2.0 (Brück 2023). The proposal aims to be a practical compromise with the following key properties:

- Components and connectors can have native Modelica types, such as a rotational inertia. We have chosen to represent them with SSP’s Binary type as a generalization of the concept.
- Acausal connectors have been added (in addition to in, out and inout).
- Parameter values can be arbitrary expressions.

- Modelica components use references to Modelica types; the Modelica models themselves are not stored in the SSP file.

It is hoped that the community can gain experience of using Modelica components in SSP and that it eventually lead to adaption by SSP.

4.4 Round tripping

In this context, round tripping means the ability to read and write between internal and external data formats without any loss of information. A stricter sense of the term would require an identical external representation.

Dymola is only partially successful in this respect. Overall structure and simulation behavior is very well preserved, which can be expected as FMUs are copied in and out.

The read-edit-store cycle for SSPs is not so well developed and will need improvement in the future. In particular, annotations from other tools may be lost during editing.

5 Conclusions

Reasonable, although not complete, support for SSP has been implemented in Dymola. It takes advantage of earlier functionality for e.g. importing FMUs, hence is able to map most SSP concepts to Modelica models.

The development effort is entirely guided by practicality, with the aim of providing high-quality simulation capabilities to SSP. The degree of success has to be judged by its users.

Acknowledgements

The author wants to thank the members of the SSP design group, in particular Pierre Mai and Peter Lobner, for constructive collaboration.

References

Brück, Dag (2023). "Modelica Models in SSP" in Proc. 15th International Modelica Conference, Aachen, Germany.

Dassault Systèmes (2019). "What is Dymola?", https://www.3ds.com/fileadmin/PRODUCTS/CATIA/DYMOLA/PDF/What-is-Dymola-2020x.pdf.

Elmqvist, Hilding (2014). "Modelica Evolution - From My Perspective" in Proc. 10th Modelica Conference, Lund, Sweden.

eXXcellent (2022). "orchideo | easySSP", https://www.easy-ssp.com/.

Heinkel, Hans-Martin, P. R. Mai, R. Aue, J. Bou, C. Bühler, C. Franke and A. Puetz (2022). "SRMD format and classifications for metadata". https://gitlab.setlevel.de/open/processes_and_traceability/traceability_data/-/blob/main/SETLevel_SRMD_and_classifications_for_metadata.pdf.

Heinkel, Hans-Martin and K. Steinkirchner (2022a). "Credible Simulation Process". https://gitlab.setlevel.de/open/processes_and_traceability/credible_simulation_process_framework/-/blob/main/Credible-Simulation-Process-v1-2.pdf.

Heinkel, Hans-Martin and K. Steinkirchner (2022b). "Credible Modeling Process", https://gitlab.setlevel.de/-open/processes_and_traceability/credible_simulation_process_framework/-/blob/main/Credible-Modeling-Process-v1-0.pdf.

Heinkel, Hans-Martin, P. R. Mai, R. Aue, J. Bou, C. Bühler, C. Franke and A. Puetz (2022c). "SRMD format and classifications for metadata", https://gitlab.setlevel.de/open/processes_and_traceability/traceability_data/-/blob/main/SETLevel_SRMD_and_classifications_for_metadata.pdf.

Hällqvist, Robert, R. C. Munjulury, R. Braun, M. Eek and P. Krus (2021). "Engineering Domain Interoperability Using the System Structure and Parameterization (SSP) Standard" in Proc. of the 14th International Modelica Conference, Linköping, Sweden.

IRT SystemX (2020). "Model Identity Card (MIC)", https://mic.irt-systemx.fr/mic.

LOTAR International (2023). "LOTAR - Long Term Archiving and Retrieval", https://lotar-international.org/.

Mai, Pierre R. et al. (2019). "System Structure and Parameterization". https://ssp-standard.org/publications/SSP10/SystemStructureAndParameterization10.pdf.

Mai, Pierre R. (2023). "PMSF FMI Bench", https://pmsf.eu/products/fmibench/.

MoSSEC Project (2021). "Modelling and Simulation information in a collaborative Systems Engineering Context," ISO 10303-243:2021, http://www.mossec.org/.

Rude, Stefan, F. Fischer, H. Esen, P. R. Mai, K. Steinkirchner, P. Lobner, D. Brück and H.-M. Heinkel (2021). "prostep ivip SmartSE: Traceable simulation results in a heterogeneous environment", https://youtu.be/qVXD0sZG5f8.

Vettermann, Steven, C. Bühler and K. Steinkirchner (2021). "Traceability Software Demonstrator TRACY," 29 April 2021, https://setlevel.de/assets/mid-term-presentation/Traceability.pdf.