Engineering Domain Interoperability Using the System Structure and Parameterization (SSP) Standard

Robert Hällqvist[1,4] Raghu Chaitanya Munjulury[2,4] Robert Braun[4] Magnus Eek[3] Petter Krus[4]

[1]System Simulation and Concept Development, Saab Aeronautics, Sweden, robert.hallqvist@saabgroup.com
[2]Technical Management & Maintenance, Saab Aeronautics, Sweden
[3]Technology & Innovation Management, Saab Aeronautics, Sweden
[4]Division of Fluid and Mechatronic Systems (FLUMES), Linköping University, Sweden

Abstract

Establishing interoperability is an essential aspect of the often-pursued shift towards Model-Based System Engineering (MBSE) of, for example, aircraft. If models are to be the primary information carriers during development, the applied methods to enable interaction between engineering domains need to be modular, reusable, and scalable. Given the long life cycles and often large and heterogeneous development organizations in the aircraft industry, one possible solution is to rely on open standards and tools. In this paper, the standards Functional Mock-up Interface (FMI) and System Structure and Parameterization (SSP) are exploited to exchange data between the disciplines of systems simulation and geometry modeling. A method to export data from the 3D Computer Aided Design (CAD) Software (SW) *CATIA* in the SSP format is developed and presented. Analogously, FMI support of the Modeling & Simulation (M&S) tools *OMSimulator*, *OpenModelica*, and *Dymola* is utilized along with the SSP support of OMSimulator. The developed technology is put into context by means of integration with the M&S methodology for aircraft vehicle system development deployed at Saab Aeronautics. Finally, the established interoperability is demonstrated in an industrially relevant use-case. A primary goal of the research is to prototype and demonstrate functionality, enabled by the SSP and FMI standards, that could improve on MBSE methodology implemented in industry and academia.

Keywords: FMI, SSP, Modeling and Simulation, CATIA, OMSimulator, OpenModelica, Dymola

1 Introduction and Motivation

Each engineering domain has its preferred methods and tools for design and analysis, implying that different but often overlapping views of one single system are modeled using several different tools. For some applications, managing all views of interest in a tool suite provided by a single tool vendor may be possible, but for other applications, this may be neither feasible nor desirable. Exchanging information between engineering domains using tool-to-tool connections introduces a set of unwanted drawbacks that may increase in significance with increased product life cycle length. Such connections are fragile and may require significant maintenance. This motivates the utilization of standards to ensure continuity and consistency in the digital thread.

Traditionally at Saab, in-house standardized interface formats are used for the manual exchange of data between different engineering domains. Even though the applied MBSE methods are generally considered successful, such processes are error-prone, tedious, and difficult to maintain; particularly considering the previously mentioned long life cycles of aircraft and aircraft sub-systems. As a result, there is a risk that data may be exchanged less frequently than it should be. This may be a limiting factor in M&S credibility and the risk of taking sub-optimal model-based design decisions is increased as a consequence.

At Saab Aeronautics, aircraft vehicle system models are developed according to the *Saab Aeronautics Handbook for Development of Simulation Models* (Andersson and Carlsson 2012). This handbook highlights aspects such as the definition of model specifications, intended use(s), and the importance of conducting Verification & Validation (V&V). These activities are all highlighted as essential in the literature (Roza, Voogd, and Sebalj 2012; Roy and Oberkampf 2011; International Council on Systems Engineering 2015).

The process described in the handbook can largely be described by the workflow visualized as the *Sub-system Model* abstraction level of Figure 1. Furthermore, the cornerstones of *Sub-system Model* development are analogous to *Simulator* and *Component Model* development if viewed at the level of detail presented in Figure 1. The artifacts at each level of abstraction are executable simulation models, or simulators, of the mathematical sort (Ljung and Glad 2004; Peter Fritzson 2004).

The SSP (Modelica Association 2019) and FMI (FMI Development Group 2020) standards provide standardized formats for establishing interoperability between the different levels of abstraction. The FMI standard concern the export of models for integration in *simulator* (Hällqvist 2019) applications and the SSP standard primarily focuses on the definition and export of simulators for integration into simulator applications at a higher level of abstraction or for exploitation in different frames of reference.

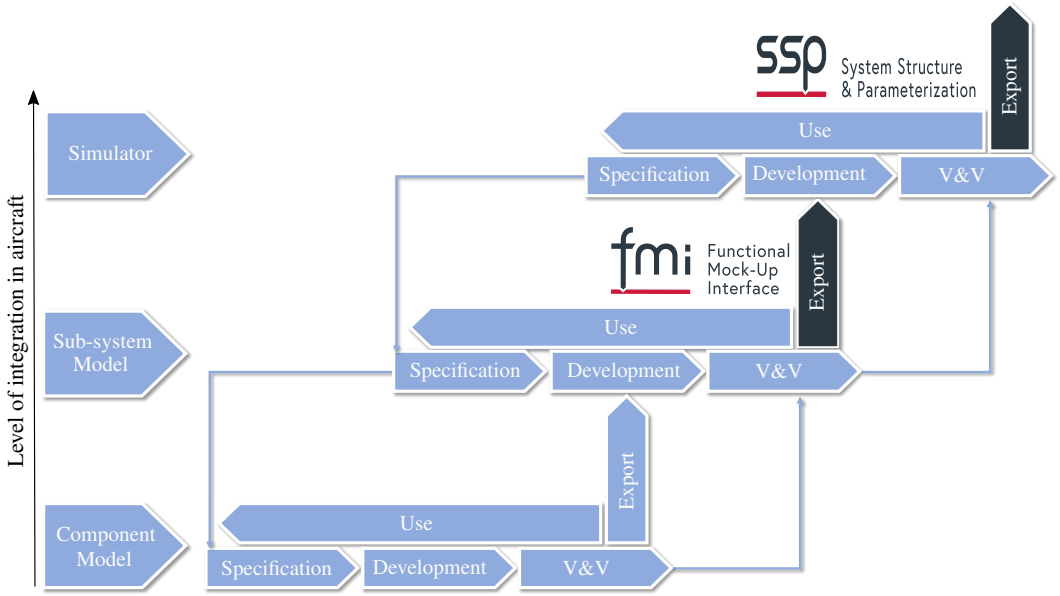


Figure 1. Simulation application development process connecting the Component Model, Sub-system Model, and Simulator levels of abstraction. The term *Model* here refers to mathematical simulation models as specified by Ljung (Ljung and Glad 2004) and Fritzon in (Peter Fritzson 2004). A simulator here is also seen as a mathematical simulation model composed of several connected models or simulators exported from a lower level of hierarchy (Hällqvist 2019). The V&V activities are seen as bottom up, a view that is in line with the *Validation Level per Level* approach as described by (International Council on Systems Engineering 2015). The natural connection between the two standards, FMI and SSP, and the activities of the simulation application development process are highlighted in the figure.

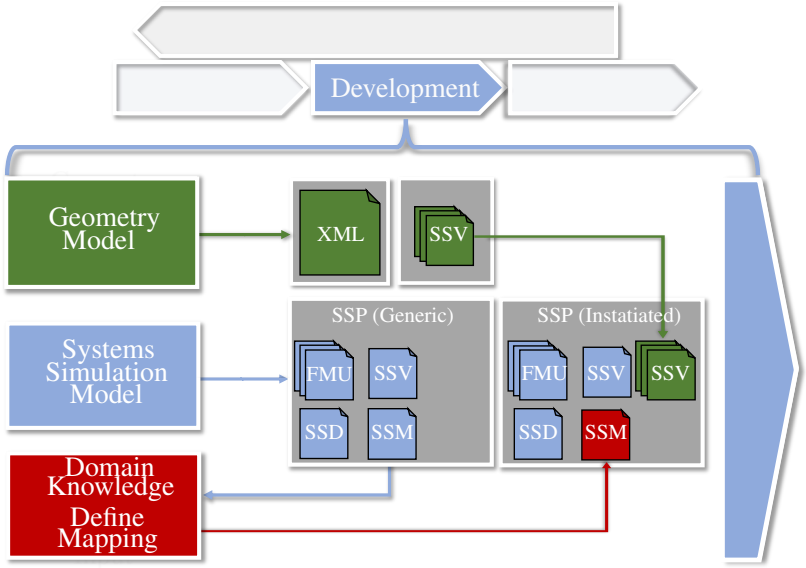


Figure 2. Expansion of the development activity specified in Figure 1. The workflow is designed to be applicable to the three system levels shown in Figure 1: the Component Model, Sub-System Model, and Simulator levels of abstraction. The colors in the figure indicate the source of the information. The rightmost SSP in the figure, *SSP (Instantiated)*, conforms to a executable specified using information from all the relevant engineering domains.

DOI 10.3384/ecp2118137

Additionally, the SSP standard provides a standardized format for specifying the parameters of M&S artifacts representing more than a single realized aircraft system, subsystem, or component. The approach taken here is to exploit these parts of the standard, at all of the presented levels of abstraction, to enable a tool agnostic method to exchange information concerning geometrical parameters and their bindings to the corresponding mathematical simulation models and simulators. As a consequence, Figure 1 is complemented by Figure 2 which is described in detail in Section 4.

This paper is structured as follows. In Section 2, the enablers of the work are introduced and related to the presented research. Furthermore, geometry modeling is related to the targeted standards in Section 3. The proposed refinement of the existing M&S application development methodology implemented at Saab is described in Section 4. In Section 5, the relevant details of the selected application example are described; this application example is then subjected to a use-case presented in Section 6. The use-case results are presented and discussed in Section 7. Finally, the conclusions of the work are stated in Section 8.

2 Interoperability via open tools and standards

Standardized and automated connections of CAD to other interdependent engineering domains is by no means a new research area and a plethora of solutions have been proposed in the literature, including published research led by Saab (Lind and Oprea 2012).

In 1999, Engelson et al. formulated a method to transform mechanical domain *SolidWorks* geometry models into the format of the standardized multi-domain M&S language Modelica (Engelson, Larsson, and P. Fritzson 1999). Similarly, Elmqvist et al. developed a tool for the automatic translation of CATIA multibody models into Modelica code (Elmqvist, Mattsson, and Chapuis 2009). Remond et al. employ a similar approach to generating Modelica models of thermo-fluid piping networks based on CAD data from CATIA (Remond, Gengler, and Chapuis 2015).

Furthermore, Baumgartner et al. developed *Dymola* functionality for generating Modelica models from CATIA multibody models. Their approach explicitly includes the storing of parameter values in a separate text format such that the geometrical data can be modified and updated without re-generating the complete model or package (Baumgartner and Pfeiffer 2014). This conscious separation is of particular interest during the development of aircraft models because such models and libraries often have long life cycles spanning a significant period of the aircraft's development and operation.

These publications all primarily focus on exchanging or generating complete executable models detailed with the geometry model information expressed in the CAD tool of the author's choice. However, during development, and later life cycle stages, the overall structure of the models is less prone to change. A component may be enhanced, the dimensions of a pipe may be modified, but what components are used and what parameters need to be exchanged is likely to be well established information. Lind et al. present such a use-case where the modeled geometry of aircraft fuel tanks is exploited in order to automatically increase the fidelity of the corresponding Modelica models by means of fitting Radial Basis Functions (RBFs) networks to the extracted data (Lind and Oprea 2012).

Even though it is similar, the focus of the research presented here is instead on only exchanging model parameters. That way, the domain specific engineering tools can be used as originally intended but with relevant, and up-to-date, information from the application specific relevant neighboring engineering domains. The SSP standard format is identified as an applicable solution to the above mentioned problem.

2.1 System Structure and Parameterization (SSP)

Three of the SSP Extensible Markup Language (XML) formats are the focal point of the research presented here: the System Structure Description (SSD), System Structure Parameter Values (SSV), and System Structure Parameter Mapping (SSM). The SSD is an XML file primarily specified to describe the architecture of a set of coupled mathematical models, henceforth referred to as a simulator. In the SSV file, it is possible to specify the values of the parameters of the simulator constituent model's. However, the SSV file does not necessarily define the mapping between the parameter values and the parameter names. These bindings can instead be specified in a SSM file. Within this approach, the SSM file is specified once for each constituent model version; changes only need to be made if the parameter interface is modified. The SSV files are changed as soon as the parameter values are modified. A highly parametric model can represent many different physical systems, sub-systems, or components using various SSV input files.

2.2 Use, exchange, and manipulation of SSPs

The FMI and SSP standards are both utilized together with *Transmission Line Modeling (TLM)* (Auslander 1968; Krus et al. 1990) in the simulation environment OMSimulator. The OMSimulator is an open-source master simulation tool originally developed during the ITEA3 project *OpenCPS* (OpenCPS Project Partners 2019). It is based on a simulation framework developed by the Swedish bearing manufacturer SKF for connecting models of bearings with models from external tools (Peter Fritzson et al. 2020; Ochel et al. 2019). The OMSimulator is a stand-alone M&S tool maintained by the Open Source Modelica Consortium (OSMC). It is available as a plugin to the OMEdit Graphical modeling tool which enables graphical, and tex-

tual, development of simulators. Other tools with similar support are available; however, they are few in number because the SSP standard is still young. A list of the tools officially supporting the SSP standard is provided at the SSP project web page (Modelica Association Project System Structure and Parameterization 2021).

The OMSimulator is used as an integrating simulation tool as it supports both of the two targeted standards. Being open-source, it is also used as a platform for implementing the necessary prototype SSP manipulation functionality that is required for the approach described in Section 4.

OMSimulator functionality, using either the available *Lua* or *Python* Application Programming Interface (API) to export template SSM and SSV files is available. A subset of the OMSimulator Python API specifically relevant to the presented research is provided in Listing 1.

Listing 1. OMSimulator Python API commands particularly relevant for the presented research

```
1) oms.exportSSMTemplate("<submodel>.fmu",
"<submodel>.ssm")
2) oms.exportSSVTemplate("<submodel>.fmu",
"<submodel>.ssm")
3) oms.export("<model>", "<model>.ssp")
4) oms.exportSnapshot("<model>")
5) oms.importFile( "<model>.ssp")
```

The API command `oms.exportSSMTemplate` triggers the export of all signals in the *<submodel>.fmu*, that have a start attribute, to an SSM file. The extract of an example SSM file generated using the API command is presented in Listing 2. One *MappingEntry* is generated for each parameter. Each *MappingEntry* has a target and a source attribute. The target is mapped to the name of each *<submodel>.fmu* parameter and the source is left unspecified. The source attribute allows for the manual specification of the mapping to the corresponding parameter value in an SSV file. If a source in the generated SSM is left unspecified, then the *<submodel>.fmu* parameter is left at its default value as presented in the Functional Mock-up Unit (FMU) *<ModelDescription>.xml* file.

Listing 2. Example of SSM file template generated using the OMSimulator API command `oms.exportSSMTemplate`

```
<ssm:ParameterMapping xmlns:ssc="http://ssp-
    standard.org/SSP1/SystemStructureCommon"
    xmlns:ssm="http://ssp-standard.org/SSP1/
    SystemStructureParameterMapping" version="
    1.0">
<ssm:MappingEntry source="" target="<submodel.
    parameterA>"/>
</ssm:ParameterMapping>
```

The API command `oms.exportSSVTemplate` enables the default parameter values of the *<submodel>.fmu* to be exported to an SSV file. An extract of an example SSV file generated using the API command is presented in Listing 3. A *Parameter* entry is generated for every signal with a start attribute in the FMU *<ModelDescription>.xml* file. Each parameter entry has a *name* attribute corresponding to the *<submodel>.fmu* parameter name, a *unit* attribute, and a *value* attribute corresponding to the default parameter value, i.e. the value of the *<ModelDescription>.xml* start attribute. The SSV file parameters are mapped via name matching to the FMU parameters if a SSM file is unavailable.

Listing 3. Example of SSV file template generated using the OMSimulator API command `oms.exportSSVTemplate`

```
<ssv:ParameterSet name="
    modelDescriptionStartValues">
 <ssv:Parameters>
  <ssv:Parameter name="<submodel.parameterA>">
   <ssv:Real value="<submodel>.<ModelDescription
       >.parameterA.value>" />
  </ssv:Parameter>
 </ssv:Parameters>
</ssv:ParameterSet>
```

Additionally, In Listing 1, the Python API command for exporting an SSD description of the architecture (`oms.exportSnapshot`) is presented, along with the command for exporting, and importing, a complete SSP package (`oms.export`). An extract of an example SSD file generated using either of the two export APIs is presented in Listing 4. The SSD extract visualizes how the SSV and SSM files are incorporated into the model or simulator architecture. The *ssd:ParameterBinding* element has a *source="resources/<values>.ssv"* attribute pointing to the SSV file used. This element also has a child that, again via the *source="resources/<bindings>.ssm"* attribute, specifies the SSM file used. A complete description of all the available OMSimulator API functions is provided in the user manual (Ochel 2021).

Listing 4. Example of SSD file generated using the OMSimulator API command `oms.exportSnapshot`. The SSD file connects the FMU parameters to the parameter values in the SSV file via the mappings expressed in the SSM file.

```
<ssd:SystemStructureDescription
 <ssd:System name="root">
  <ssd:ParameterBindings>
            <ssd:ParameterBinding source="
                resources/<values>.ssv">
                <ssd:ParameterMapping source="
                    resources/<bindings>.ssm"/>
            </ssd:ParameterBinding>
        </ssd:ParameterBindings>
    </ssd:System>
</ssd:SystemStructureDescription>
```

3 Geometry modeling and SSP

At Saab Aeronautics, geometry models are developed according to the *Knowledge-Based Engineering (KBE)* methodology MOKA (Stokes 2001). MOKA specifies an iterative process whereby the developed models can be added or updated in steps at each stage of the methodology. The geometry models created are parametric in nature. For example, the sizes of all the components in the application example coolant distribution system can be modified alongside any changes in the specification. The *User Defined Features (UDF)* created in CATIA encapsulate the design intent and design automation is applied

DOI 10.3384/ecp2118137

to instantiate the pipes based on the input points. All the parameters needed for simulation are computed automatically because the knowledge/calculations are embedded in the UDF.

Inspired by the previous work conducted at Saab and Linköping University (Lind and Oprea 2012; Munjulury et al. 2016; Munjulury 2017), Visual Basic for Applications (VBA) is exploited to extract parameter information from CATIA via a set of macros. Additionally, the implemented macros convert the extracted information to the SSV format.VBA macros can either be executed directly from CATIA or via, for example, a User Interface (UI) in Microsoft Excel. Application-specific mapping, as described in Section 3.2, is applied to the intermediate XML.

3.1 Parameter extraction and SSP support

An approach similar to that of Munjulury et. al. (Munjulury et al. 2016), exploiting the Document Object Model (DOM) objects available in VBA, is used to create an intermediate CATIA XML. This intermediate CATIA XML conversion is tailored to reduce the number of data interfaces enabling a robust and seamless exchange to other formats, such as SSV. For every entity (point, line, plane, surface, etc.,) at least three parameters are created when designing a component in CATIA. The functionality to extract and save all the geometry model parameter values into an XML file is available in, for example, *3D experience*; however, it is all the more challenging to find, extract, and store specific parameters. First of all, the latter requires a list of identifiers. Identifiers in this scenario are the *Parameter sets* or *Geometrical sets* that contain the parameters in the respective identifier list to help narrow down the search. The following are the steps involved in creating the SSV file.

- The user provides the Identifiers in the UI in order to narrow down the total number of parameters that need to be saved to the intermediate CATIA XML.
- A first VBA macro is executed which reads all the parameters available in the CATIA geometry model. The parameters specified in the identifier list mentioned above are then extracted from the geometry model and saved in the CATIA XML. The CATIA XML structure maps to the CATIA Product tree structure.
- A second VBA macro converts the CATIA XML to an SSV file.

3.2 Integration of application specific functionality

Functionality enabling application specific mappings has been developed. This functionality is kept separate from the SSP support macros such that aggregation methods can be exchanged and tailored to different applications. The methodology to instantiate pipes and insulation using the UDF is an add on to the methodology currently used at Saab Aeronautics to create the respective components. This add on reduces the time needed for the design process as most of the process is successfully automated; the only additional modeling requirement is to include the bend points of the pipes. With this automation, the parameters needed by the mathematical models of the application example are created and recursively used to compute the aggregated parameter values needed for the lumped parameters. The developed application specific functionality is,

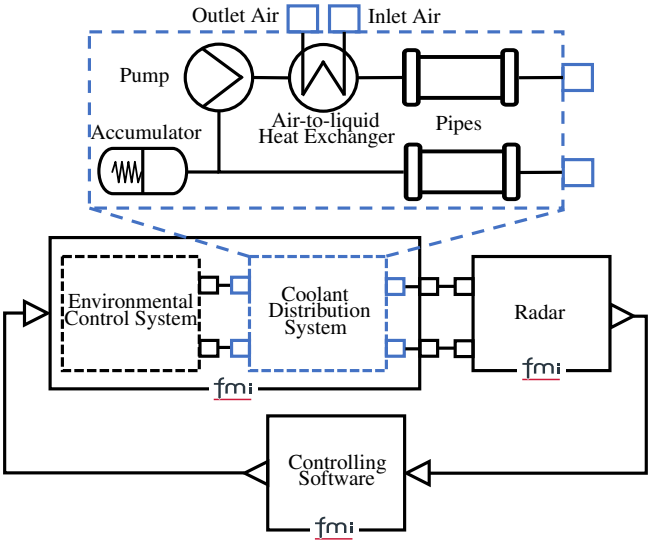
- The combination of parameter values, such as fluid pipe lengths, enabling lumped parameter dynamic simulation components
- Unit transformations,
- Interpolating in application specific interpolation tables used to, for example, convert bends in pipes to pressure loss coefficients.

4 Proposed concept

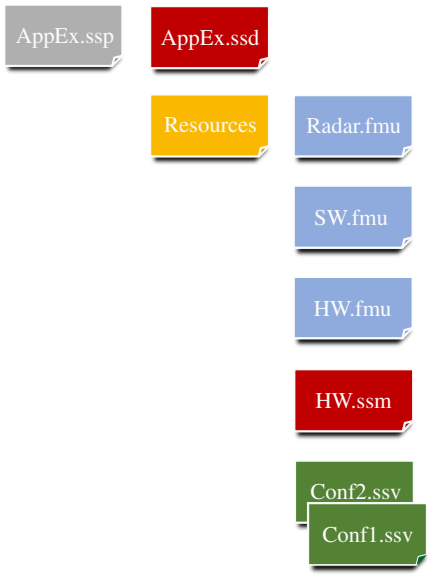
This section describes the proposed methodology for exploiting the established connection between the domains of geometrical modeling to that of mathematical modeling and system simulation. The methodology is here related to the general simulation application development process presented in Figure 1. In Figure 2, the proposed additions to the *Development* activity of Figure 1 are described in detail. The process visualizes, see the light blue artifacts in the figure, how the simulation application is first exported in the SSP format, including an SSD architecture description, an SSV file containing default parameter values of the parameterized simulator, sub-system model, or component model, and a template SSM file containing empty bindings to the aforementioned parameters. In parallel, the parameter values of the corresponding geometry model are expressed in the SSV format, visualized as green artifacts in the figure. In the next step, the SSM file is populated with the geometry model names of the corresponding simulation application parameters such that the geometry model SSV file is specified as the source of the parameter values. Please note that the process of Figure 2 needs to be iterated. However, the user input specified SSM file only needs to be updated if the parameter interface is modified. The rightmost SSP in the figure corresponds to a set of fully specified executable entities ready for V&V, as-is use, and integration into a simulation application at a higher level of abstraction.

The process of Figure 2 is described as applicable for the development activities at all of the levels of abstraction presented in Figure 1. However, the work here is primarily focused on the *Sub-system Model* and *Simulator* levels. The available tool support of the SSP standard is primarily focused on FMI applications. If parameters in components, present in for example Modelica component libraries, are to be specified using the SSP standard,

then tool support for this type of functionality, in the relevant simulation application modeling tools, could render a more efficient exchange of information.



(a) Application example schematic description. The dashed *Coolant Distribution System* is highlighted as its parameters are specified in the geometry modeling domain and exchanged using the concepts of the SSP standard. The individual parts of the application example are exported using the FMI standard.



(b) Application example SSP file structure. The SSP file represents two different simulator geometrical configurations via the two different incorporated SSV files *Conf1.ssv* and *Conf2.ssv*. The simulator architecture is described in the *AppEx.ssd* file.

Figure 3. Application example description. A schematic description of the application example architecture and its constituent executable models is provided in Figure 3a. The structure of the resulting application example SSP file is provided in Figure 3b.

5 Application example

The application example incorporates the targeted engineering domains and M&S tools; a schematic overview is provided in Figure 3. The application example is separated into three different main parts; each of these parts is exported as an FMU from its original development tool. The modeled *Coolant Distribution System* is highlighted in as dashed and blue in Figure 3a as it here is the target of the established interoperability between systems simulation and geometry modeling.

The FMUs of the application example are packaged in a SSP file providing a complete and executable description of the targeted simulator configurations, see Figure 3b. The depicted SSP file includes two different geometrical configurations. These configurations are specified through the two included SSV files *Conf1.ssv* and *Conf2.ssv*.

5.1 Mathematical modeling

Two of the three constituent parts of the application example are individual models of the mathematical sort. These two mathematical models include an interpolation based representation of a Environmental Control System (ECS), a liquid coolant distribution system, and a consumer of generated cooling power. The first two modeled coupled sub-systems, the ECS and the liquid coolant distribution system, are lumped together in a single exported FMU, whereas the consumer is separated from the other two, see Figure 3. The development of these aircraft sub-systems is typically conducted by different departments at Saab, or by suppliers, and this partitioning is intended reflect a likely situation during development.

Even so, all three modeled sub-systems are developed using components from the Saab Aeronautics in house Modelica library *Modelica Fluid Light* (Eek, Gavel, and Ölvander 2017) and the *Modelica Standard Library* (The Modelica Association 2019). The mathematical modeling is conducted in the Dymola and OpenModelica SW.

5.1.1 Environmental Control System (ECS)

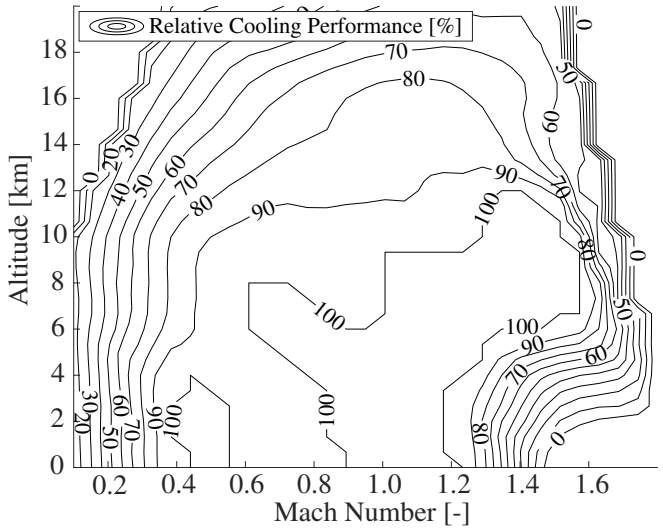
A simulator enabling detailed studies of pilot thermal comfort was presented in (Hällqvist et al. 2018). The ECS presented here is based on the aircraft cooling system of that simulator. The ECS model incorporated in the application example is intended to provide a connection between the operating conditions and the cooling power available for distribution to the included consumer.

The results of maximum available steady-state relative cooling power $\dot{Q}_{rel}$, along with the corresponding available mass flow of conditioned cold air $\dot{m}$, are exploited in an interpolation based Modelica component, see Figure 4a and Figure 4b. Provided the characteristics of Figure 4, the minimum possible cold air temperature can be calculated as

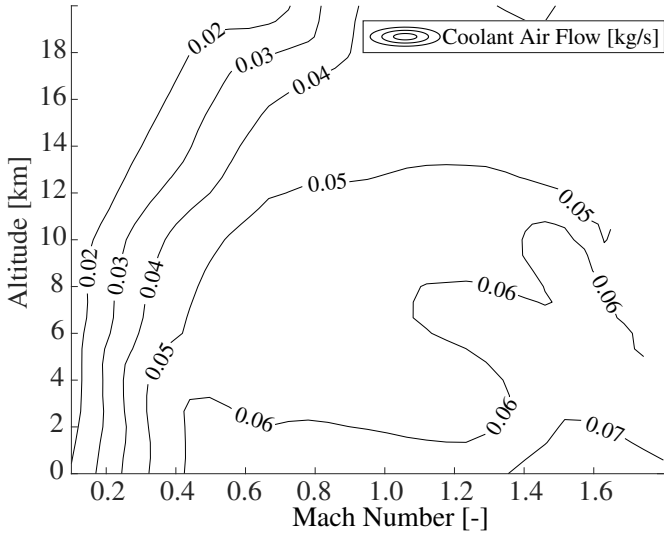
$$T_{in}^{min} = T_{out} - \frac{\dot{Q}_{max}^{current}}{\dot{m} \cdot C_p} \tag{1}$$

where T_{out} is the current exhaust air temperature and $\dot{Q}_{max}^{current} = \dot{Q}_{rel} \cdot \dot{Q}_{max}$. The parameter $\dot{Q}_{max}$ specifies the maximum available cooling power independent of the operating conditions. The specific heat at constant pressure is denoted C_p in Equation 1. The ECS is here modeled

DOI 10.3384/ecp2118137



(a) ECS available relative cooling power ($\dot{Q}_{rel}$) as function of altitude and Mach number



(b) ECS available coolant flow ($\dot{m}$) as function of altitude and Mach number.

Figure 4. Steady-state characteristics of the application example ECS

as a source with a prescribed mass flow corresponding to that of $\dot{m}$. The temperature of the air expelled from the source is regulated by the incorporated control system, see Figure 3; however, a lower bound corresponding to that of T_{in}^{min} is specified in the ECS model.

5.1.2 Coolant distribution system

The application example's coolant distribution system is schematically described as the dashed and highlighted model in Figure 3. The cooling distribution system interfaces the modeled ECS via the Air-to-Liquid Heat Exchanger (LHEX). The LHEX transfers heat from the liquid circulated by the included pump to the ECS coolant air.

The Modelica components with parameters specified by the geometry model are all part of the *Modelica Fluid Light* library. The considered modeled components are a pipe, a pump, an LHEX, and an accumulator. The parameters of each modeled library component, here specified via the application example's geometry model, are presented in Table 1. A sub-set of the included model component equations are described in detail in the following paragraphs, in order to highlight how the selected parameters impact upon the characteristics of the coolant distribution system model.

Pipe	D_h	l	z
Accumulator	V_{acc}		
LHEX	h	b	l

Table 1. Summary of the parameters that are exchanged in the application example. The pipe parameters D_h and z represent the pipe hydraulic diameter and its lumped pressure loss coefficient. The pressure loss coefficient is a result of pipe bends and contractions/expansions. The parameter l represents pipe or LHEX length. The LHEX height and width are denoted h and b and the accumulator volume V_{acc}.

The pipe model algebraic equation relating the component parameters to mass flow ($\dot{m}$) and pressure drop (Δp) is

$$\Delta p = \frac{(z + c \cdot l/D_h)}{A^2 \cdot 2\rho} \dot{m}^2 \tag{2}$$

where A is the pipe's cross sectional area, D_h is the hydraulic diameter, and l is the pipe length (Miller 1990). The friction coefficient is denoted by c. The one-time pressure losses occurring as a result of pipe bends and contractions/expansions are incorporated via the parameter z. The pipe component parameters of Table 1 also impact upon the relationship between the air temperature surrounding the pipe and the specific enthalpy of the fluid itself. This relationship is described by a system of differential and algebraic equations:

$$\begin{aligned} q_{ep} &= A_o h_{ep}(T_e - T_p), \\ q_{pf} &= A h_{pf}(T_p - T_f), \\ \dot{T}_p &= (q_{ep} + q_{pf})/(C_p \cdot M), \\ \dot{h} &= \dot{m}/(\rho \cdot V)(h_{in} - h), \end{aligned} \tag{3}$$

where the pipe hull outer surface area is $A_o = \pi D_o L$ and the pipe hull inner surface area is $A = \pi D_h L$. The specific enthalpy h is modeled as a nonlinear function of the fluid temperature T. The pipe input specific enthalpy is denoted by h_{in}. The variable T_e represents the temperature of the media surrounding the pipe's outer surface whereas the variable T_p represents the pipe's hull temperature. The intermediate variables q_{ep} and q_{pf} represent the heat transferred from the pipe's external environment to its hull and from the pipe's hull to the fluid, respectively.

The modeled accumulator component exploits the Modelica inner/outer concept to access temperature dependent information concerning the system's total volume. The accumulator pressure is then related to the fluid temperature via linear interpolation as

$$p = \frac{p_f - p_e}{V_{acc}} \left[\sum_{i=1}^{N} (V_i(T) - V_i(T_0)) + V_{acc}(T_0) \right] + p_e \tag{4}$$

where $V_i(x)$ is the fluid volume in connected component model i, at the current temperature $x = T$ or the temperature at filling $x = T_0$. The pressure in the accumulator when it is full and empty are denoted p_f and p_e respectively. The two accumulator parameters V_{acc} and $V_{acc}(T_0)$ represent the total accumulator volume and the volume of liquid in the accumulator at filling temperature.

The LHEX is a plate fin cross-flow type heat exchanger where the model parameters length, width, and height determine the total heat transfer area of both the hot and cold side. Such a heat exchanger is a common and appropriate selection for gas-to-liquid applications, where the optimal arrangement conforms to maximizing the surface area on the gas side. The presented component model is founded on the theory presented by Kays et al. in (Kays and London 1984).

The assumed flow arrangement in this model component is that of one fully mixed fluid (the gas) and one unmixed fluid (the liquid). This assumption translates to that the gas temperature is constant perpendicular to the direction of the flow. The hot and cold side, $i = h$ and $i = c$ respectively, heat transfer rates can be expressed as

$$h_i = \left[\frac{C_p S_t \dot{m} \frac{4l}{D_h}}{A} \right]_i \tag{5}$$

where A_i is the total heat transfer area of side i. The Stanton number S_t in Equation 5, named after Thomas Stanton, is a heat transfer modulus that is used to characterize heat transfer (Ackroyd 2007). In this particular model, the Stanton number is seen as a tunable exponential function of the Reynolds number that is calibrated against efficiency measurements.

The LHEX overall thermal resistance can now be expressed as

$$R = \sum_{i=c}^{h} \frac{1}{h_i A_i}. \tag{6}$$

assuming negligible thermal resistance of the walls and no extended fin surface on either the hot or cold sides of the LHEX.

The thermal efficiency, for a heat exchanger with this particular assumed flow arrangement, is expressed as either

$$\varepsilon = 1 - e^{-\left[1 - e^{-N_{tu}} \frac{C_{min}}{C_{max}}\right] \frac{C_{max}}{C_{min}}} \tag{7}$$

or

$$\varepsilon = \frac{C_{max}}{C_{min}} \left[1 - e^{-\left[1 - e^{-N_{tu}}\right] \frac{C_{min}}{C_{max}}} \right] \tag{8}$$

depending on which of the side's flow capacity rates C that are limiting. If $C_{min} = C_c = \dot{m}_c C_{p_c}$ then Equation 7 is applicable, and if $C_{min} = C_h = \dot{m}_h C_{p_h}$ then Equation 8 is applicable. The thermal resistance influences the efficiency via the heat transfer parameter

$$N_{tu} = \frac{1}{RC_{min}} \tag{9}$$

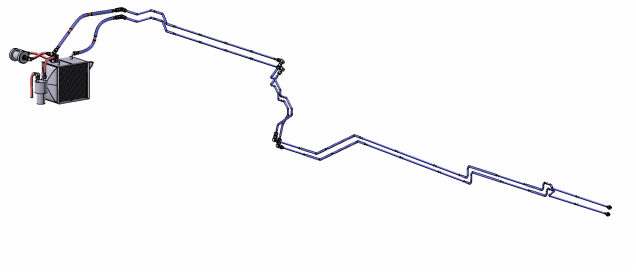
which connects the geometrical parameters to the efficiency. In the application example LHEX model $l_h = 2 \cdot b$, as the hot liquid passes the cold side surface twice, and $l_c = l$. Additionally, the parameters b, h, and l affect the efficiency through S_t which here is modeled as an exponential function of the Reynolds number. This exponential function is tuned such that the model complies with supplier data.

Finally, the LHEX hot side outlet temperature T_h^{out} can be described as function of the efficiency

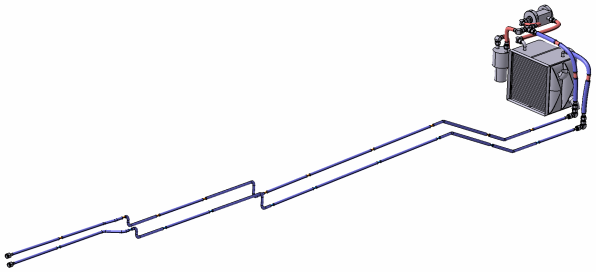
$$T_h^{out} = T_h^{in} - \varepsilon(T_h^{in} - T_c^{in}) \tag{10}$$

where the superscipt indicates inlet or outlet temperature.

5.2 Geometrical representations



(a) Geometry model of cooling power distribution system routing option one (*Configuration 1*). The ECS is located in the aft of the aircraft.



(b) Geometry model of cooling power distribution system routing option two *Configuration 2*. The ECS is located immediately behind the aircraft's cockpit.

Figure 5. Use-case geometry models representing the two different routing options under investigation. The piping reaches from the LHEX to the front of the aircraft where the radar is located. The radar is not included in either Figure 5a or Figure 5b.

Two different configurations of the coolant distribution system are modeled in CATIA, see Figure 5. The resulting geometry models include geometrical representations of all the parts of the coolant distribution system model. The main components, the LHEX, Pump, and Accumulator are the same in both configurations.

In both configurations, the piping reaches from the LHEX to the front of the aircraft where the radar is located. The main difference between the configurations

DOI 10.3384/ecp2118137

lie in the positioning of the ECS and coolant distribution system core, i.e. the accumulator, pump, and LHEX. In *Configuration 1*, the routing extends from the aft, via the aircraft ridge, to the radar. This configuration results in a significantly longer routing with more bends compared to *Configuration 2*, where the ECS is located immediately behind the cockpit. Both configurations have advantages and disadvantages. For example, the potential increased pressure drop of *Configuration 1* could be outweighed by the reduced need for transporting engine bleed air to the, in this case, bleed-air-driven ECS.

6 Use-case

A use-case, presented in this section, is formulated to demonstrate the functioning and benefits of the developed technology. An Operational Concept (OpsCon) (International Council on Systems Engineering 2015) mission, along with a sub system requirement posed by the hypothetical developer of the application example radar, together compose the use-case requirements on the coolant distribution system.

6.1 Prerequisites

The application example described in Section 5 naturally serves as the primary use-case prerequisite. Here, the application example is available in the form of a generic SSP, including a template SSM file generated using the functionality provided in Listing 2.

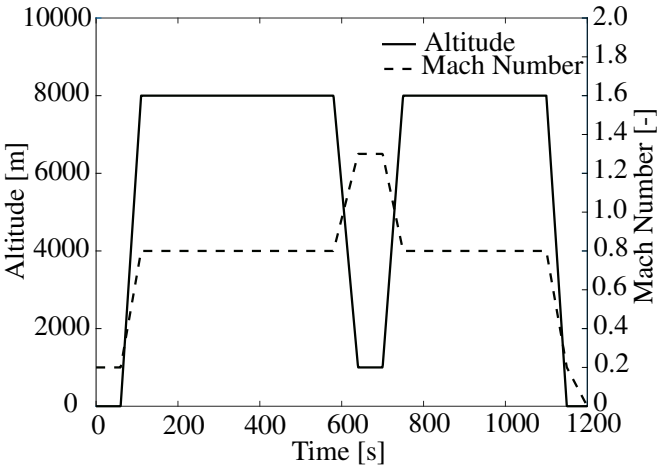
In addition, the OpsCon mission is seen as a top-level requirement which the application example should fulfill. The application example boundary conditions of the OpsCon mission is presented in Figure 6. The mission altitude and Mach number profiles are presented in Figure 6a, and the radar heat load and SW input aircraft state in Figure 6b.

The application example aircraft leaves the runway after approximately 100 s with the goal of identifying an unknown aircraft known to be present approximately 115 km from the base. A climb and acceleration to cruise conditions are then initiated and realized. The aircraft operates at cruise conditions until it reaches the specified location. A loitering phase is commenced and the radar is shifted from stand-by to active, see the power transient depicted in Figure 6b. The aircraft being sought is located after 60 s of searching and the operating conditions are then matched to those of the foreign aircraft via a speed increasing dive. Once contact has been established, the radar is shifted to stand by mode and the aircraft is returned to base via a second low fuel consumption cruise phase.

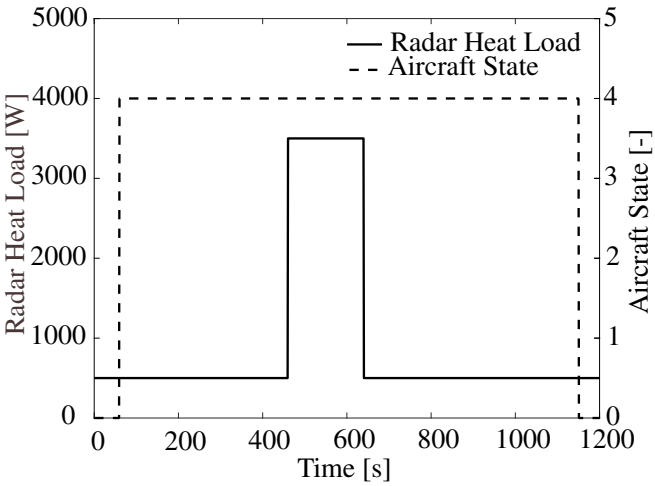
The OpsCon mission specifies the use and functioning of a radar. This radar functions provided that the subsystem requirement

- The difference in radar coolant inlet and outlet temperature shall not exceed 13°C

is fulfilled throughout the OpsCon mission.



(a) Altitude and Mach number of the OpsCon mission profile



(b) Heat load exerted by the application example radar component during the OpsCon mission. The radar can here operate in two different discrete modes: a stand by mode corresponding to 500 W of power and a active mode corresponding to 3500 W. The dashed line represents the SW input signal *AircraftState* which indicates whether the aircraft is situated on the ground (*AircraftState*= 1), or if it is airborne (*AircraftState*= 4).

Figure 6. Boundary conditions corresponding to the specified OpsCon mission profile

6.2 Sunny day scenario and expected outcome

The engineering team responsible for the acquisition and tailoring of the ECS to be used in the aircraft, in collaboration and agreement with the ECS supplier, has identified two different possible system locations: below the fin in the aft of the aircraft, and immediately behind the cockpit.

Each ECS position results in a different routing of the liquid coolant distribution system as the consumer of coolant power is located in the nose of the aircraft. The engineer responsible for specifying the installation of the liquid coolant distribution system is proposing two different routing options, see Figure 5 and Section 5.2. Geometry models are developed for both of the two different routing options and the corresponding parameters are exported, using the functionality presented in Section 3, as two different SSV files. These SSV files are placed in

the resources of the application example SSP as shown in Figure 3b. An extract of the exported geometry information in the SSV format is provided in Listing 5. An extract of the corresponding intermediate CATIA XML is provided in Listing 6. The presented parameter value is common to both *Configuration 1* and *Configuration 2*.

Listing 5. Extract of application example geometry information in the SSV format.

```
<ssv:ParameterSet name="product_ECS">
  <Units>
    <Unit name="m">
      <BaseUnit m="1"/>
  </Units>
  <ssv:Parameter name="part_LHEX.
      parameterSet_inputParameters.
      parameter_width">
    <ssv:Real unit="[m]" value="0.3"/>
  </ssv:Parameter>
</ssv:ParameterSet>
```

Listing 6. Extract of application example geometry information in the intermediate XML format described in Section 3.1.

```
<product name="ECS">
  <part name="LHEX">
    <parameterSet name="inputParameters">
      <parameter name="width" type="Double">
        <value>300</value>
        <unit>[mm]</unit>
      </parameter>
    </parameterSet>
  </part>
</product>
```

In parallel, the involved stakeholders agree upon a mapping between parameter values and the parameters of the FMUs relevant to the application example; thus updating the SSM from template to the final version of the instantiated SSP.

The sub-system requirements need to be verified during the presented OpsCon. The analysis is suggested to provide feedback on the design in terms of suggestions concerning the ECS positioning and the accompanying routing. The feasibility is determined with respect to the presented system and sub-system level requirements.

7 Results and discussion

The application example is simulated for the mission profile described in Section 6.1. The geometry settings of the two different modeled configurations are summarized in Table 2 and Table 3. The parameters that differ between

	Feed line piping		Return line piping	
	l[m]	z[-]	l[m]	z[-]
Configuration 1	7.393	2.491	7.412	2.417
Configuration 2	4.614	0.985	4.571	0.880

Table 2. Summary of parameter values that differ between the two configurations. The cooling distribution system routing is subject to modification. The remaining coolant distribution system components, along with their constituent parameters, remain unchanged

	D_h[m]	h[m]	b[m]	l[m]	V_{acc}[m^3]
Piping	0.01				
Acc.					$1.71 \cdot 10^{-3}$
LHEX		0.3	0.3	0.3	

Table 3. Summary of parameter values that are specified by the geometry model but identical for the two different configurations.

configurations are presented in Table 2. Parameter values that remain unchanged in the different configurations, but are specified by the geometry models, are presented in Table 3. The remaining system simulation model parameters are kept at their default values, as specified in their original M&S development environment.

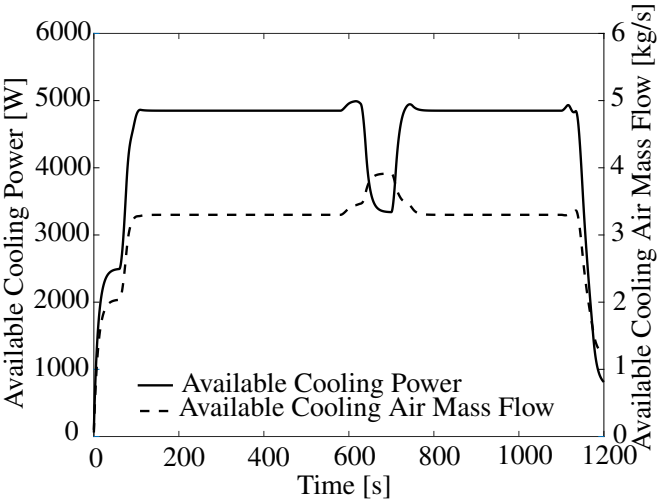
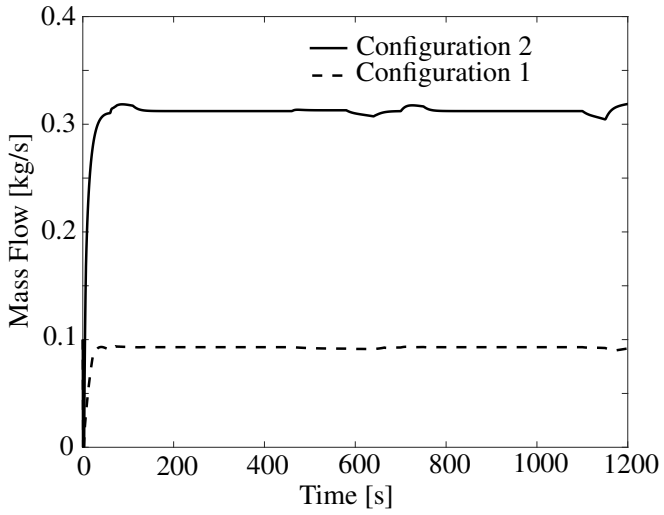


Figure 7. ECS model available cooling power (solid) and the available coolant mass flow as (dashed) for the OpsCon simulations

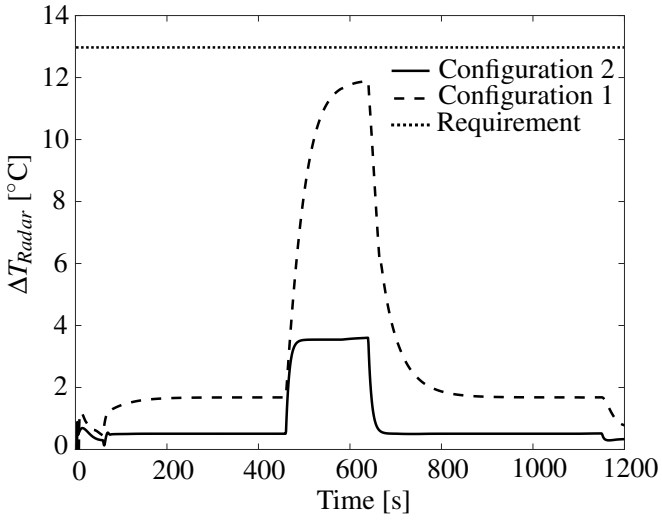
The simulation results used to assess the feasibility of the two different configurations, with respect to the requirements, are shown in Figure 7 and Figure 8. Figure 7 quantifies the performance limits of the included ECS model. The available cooling power is shown as solid in the figure and the available coolant mass flow as dashed. Note that the available cooling power is significantly greater than the OpsCon radar heat load, see Figure 6b, indicating that the ECS performance is sufficient for executing the mission.

The simulated radar inlet mass flow is shown in Figure 8a and the temperature increase over the radar in Figure 8b. The temperature increase remains below the required differential temperature level, dotted line in Figure 8b, throughout the simulated OpsCon for both configurations. Even so, the temperature increase is shown as significantly higher for *Configuration 1* than *Configuration 2*. This is a result of the corresponding lower levels of coolant mass flow, see Figure 8a. The coolant distribution mass flow depends on the pipe length l and pressure loss coefficient z, according to Equation 2, that are presented in Table 2.

DOI 10.3384/ecp2118137



(a) Coolant distribution system mass flow during the two OpsCon simulations



(b) Temperature increase over the radar during the two OpsCon simulations

Figure 8. Compilation of simulation result relevant during use-case requirement verification. The results stem from simulations of the two different configurations. *Configuration 1* is depicted as dashed, and *Configuration 2* as solid

8 Conclusions

There is much to gain already in adopting a single established standardized format for information exchange internally, within the confines of the organization, that can be version controlled and compared to previous versions using well established tools. This benefit can be increased if the standardized format is supported by the modeling tools such that the parameters can be automatically exchanged at manually, or automatically, generated events such as the commit of a model update to a repository.

The results of the research presented here indicate that the FMI and SSP standards show great promise for achieving such an automated simulation application development method. A method for exchanging parameter information between the engineering domains of system simulation and CAD has been established exploiting the, in this context, suitable open tools and standards. The method has been developed while keeping the aim of minimizing the impact on the modeling methodologies, mathematical or geometrical, in mind. The application example's aggregated pressure loss coefficients, for example, could have been computed in the components of the modeling library compared to in the developed VBA macros, see Section 3. This would, however, constitute a major change to any library that is mature and used in several different models.

The presented method has been contextualized to the simulation model development and maintenance processes currently deployed at Saab Aeronautics. Furthermore, the work has resulted in the specification of necessary functionality for manipulating SSPs. Prototype functionality is implemented in, and tested using, the OMSimulator tool. The presented methodology would benefit greatly if the presented functionality were made available in the modeling tools best suited for each considered modeling domain.

Additionally, the presented work targets the exchange and specification of parameters exposed at the interface of FMUs. An FMU generated from Modelica only allows modification of *non-structural* parameters, i.e. parameters that do not impact upon the internal structure of the system of equations. This delimitation could be avoided if the available M&S tools developed SSP support not only coupled to FMI but also to the tool's native modeling language. In such a case, the parameters could be exchanged prior to code generation and compilation.

Acknowledgements

The presented research was conducted within the frame of the ITEA3 project *EMBrACE* and the NFFP7 project *Digital Twin for Automated Model Validation and Flight Test Evaluation*. The research was funded by Vinnova and Saab Aeronautics. The authors would specifically like to thank Lennart Ochel and Arunkumar Palanisami for their help in implementing the necessary OMSimulator functionality. In addition, the authors would like to thank Dan Louthander for his help with the details of the application example and reviewing of the manuscript.

References

Ackroyd, J. A. D. (2007). "The Victoria University of Manchester's contributions to the development of aeronautics". In: *The Aeronautical Journal (1968)* 111.1122, pp. 473–493. DOI: 10.1017/S0001924000004735.

Andersson, Henric and Magnus Carlsson (2012). *Saab Aeronautics Handbook for Development of Simulation Models : Public Variant*. Tech. rep. 12/00159. Linköping University, Machine Design.

Auslander, D. M. (1968). "Distributed System Simulation With Bilateral Delay-Line Models". In: *Journal of Basic Engineering* 90.2. DOI: https://doi.org/10.1115/1.3605079.

Baumgartner, Daniel and Andreas Pfeiffer (2014-03). "Automated Modelica Package Generation of Parameterized Multibody Systems in CATIA". In: *Proceedings of the 10th International Modelica Conference, March 10-12, 2014, Lund, Sweden*. Linköping University Electronic Press. DOI: 10.3384/ecp14096913.

Eek, Magnus, Hampus Gavel, and Johan Ölvander (2017-02). "Definition and Implementation of a Method for Uncertainty Aggregation in Component-Based System Simulation Models". In: *Journal of Verification, Validation and Uncertainty Quantification* 2.1. DOI: https://doi.org/10.1115/1.4035716.

Elmqvist, Hilding, Sven Erik Mattsson, and Christophe Chapuis (2009-10). "Redundancies in Multibody Systems and Automatic Coupling of CATIA and Modelica". In: *Proceedings of the 7 International Modelica Conference Como, Italy*. Linköping University Electronic Press. DOI: 10.3384/ecp09430113.

Engelson, V., H. Larsson, and P. Fritzson (1999). "A design, simulation and visualization environment for object-oriented mechanical and multi-domain models in Modelica". In: *1999 IEEE International Conference on Information Visualization (Cat. No. PR00210)*. IEEE Comput. Soc. DOI: 10.1109/iv.1999.781557.

FMI Development Group (2020-12-15). *Functional Mock-up Interface for Model Exchange and Co-Simulation*. Report 2.0.2.

Fritzson, Peter (2004-01). *Principles of Object Oriented Modeling and Simulation with Modelica 2.1*. Wiley-IEEE Press. ISBN: 9780470545669. DOI: 10.1109/9780470545669.

Fritzson, Peter et al. (2020). "The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development". In: *Modeling, Identification and Control: A Norwegian Research Bulletin* 41.4, pp. 241–295. DOI: 10.4173/mic.2020.4.1.

Hällqvist, Robert (2019). "On Standardized Model Integration : Automated Validation in Aircraft System Simulation". Licentiate Thesis. Linköping University, Faculty of Science and Engineering. ISBN: 9789179299293. DOI: 10.3384/lic.diva-162810.

Hällqvist, Robert et al. (2018). "A Novel FMI and TLM-based Desktop Simulator for Detailed Studies of Thermal Pilot Comfort". In: *Proceedings of the 31st Congress of the International Council of the Aeronautical Sciences*. International Council of the Aeronautical Sciences. ISBN: 978-3-932182-88-4.

International Council on Systems Engineering (2015). *Systems Engineering Handbook*. 4th ed. John Wiley and Sons, Inc. ISBN: 9781118999400.

Kays, W.M. and A.L. London (1984). *Compact heat exchangers*. Krieger. ISBN: 9781575240602. URL: http://books.google.de/books?id=A08qAQAAMAAJ.

Krus, Petter et al. (1990-01). "Distributed Simulation of Hydromechanical Systems". In: *Third Bath International Fluid Power Workshop*.

Lind, Ingela and Alexandra Oprea (2012-11). "Detailed geometrical information of aircraft fuel tanks incorporated into fuel system simulation models". In: *Proceedings of the 9th International MODELICA Conference, September 3-5, 2012, Munich, Germany*. Linköping University Electronic Press. DOI: 10.3384/ecp12076333.

Ljung, Lennart and Torkel Glad (2004). *Modelbygge och Simulering*. Vol. 2. Studentlitteratur. ISBN: 91-44-02443-6.

Miller, Donald (1990). *Internal Flow Systems*. Cranfield, Bedford: BHRA (Information Services. ISBN: 978-0956200204.

Modelica Association (2019-03-05). *System Structure and Parameterization*. Report 1.0.

Modelica Association Project System Structure and Parameterization (2021). *System Structure and Parameterization*. URL: https://ssp-standard.org (visited on 2021-05-08).

Munjulury, Raghu Chaitanya (2017). "Knowledge-Based Integrated Aircraft Design : An Applied Approach from Design to Concept Demonstration". Linköping. ISBN: 9789176855201.

Munjulury, Raghu Chaitanya et al. (2016). "A knowledge-based integrated aircraft conceptual design framework". In: *CEAS Aeronautical Journal* 7.1, pp. 95–105.

Ochel, Lennart (2021). *OMSimulator's documentation*. Accessed: 2021-02-18. URL: https://openmodelica.org/doc/OMSimulator/master/html (visited on 2021-02-18).

Ochel, Lennart et al. (2019-03-04). "OMSimulator – Integrated FMI and TLM-based Co-simulation with Composite Model Editing and SSP". In: *Proceeding of the 13th International Modelica Conference*. DOI: 10.3384/ecp1915769.

OpenCPS Project Partners (2019). *Project 14018:Open Cyber-Physical System Model-Driven Certified Development*. Accessed: 2018-06-21. URL: https://www.opencps.eu/ (visited on 2019-11-15).

Remond, Xavier, Thierry Gengler, and Christophe Chapuis (2015-09). "Simulation of Piping 3D Designs Powered by Modelica". In: *Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015*. Linköping University Electronic Press. DOI: 10.3384/ecp15118517.

Roy, Christopher J. and William L. Oberkampf (2011-06). "A comprehensive framework for verification, validation, and uncertainty quantification in scientific computing". In: *Computer methods in applied mechanics and engineering* 200.25-28, pp. 2131–2144. DOI: 10.1016/j.cma.2011.03.016.

Roza, Manfred, Jeroen Voogd, and Derek Sebalj (2012-10). "The Generic Methodology for Verification and Validation to support acceptance of models, simulations and data". In: *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology* 10.4, pp. 347–365. DOI: 10.1177/1548512912459688.

Stokes, Melody (2001). *Managing engineering knowledge : MOKA: methodology for knowledge based engineering applications*. Professional Engineering Publ. ISBN: 1860582958.

The Modelica Association (2019). *Modelica and the Modelica Association*. Accessed: 2018-06-21. URL: https://www.modelica.org/ (visited on 2019-11-15).