electronics

Article

Realizing Interoperability between MBSE Domains in Aircraft System Development

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

[1] System Simulation and Concept Development, Saab Aeronautics, 581 88 Linköping, Sweden
[2] Division of Fluid and Mechatronic Systems (FLUMES), Linköping University, 581 83 Linköping, Sweden;
[3] Technical Management & Maintenance, Saab Aeronautics, 581 88 Linköping, Sweden
[4] Vehicle Systems and Driving Simulation, Swedish National Road and Transport Research Institute (VTI), 583 30 Linköping, Sweden
* Correspondence: robert.hallqvist@saabgroup.com

Abstract: Establishing interoperability is an essential aspect of the often-pursued shift towards Model-Based Systems Engineering (MBSE) in, for example, aircraft development. 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, a piece to the overall solution could be 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 on two different industrially relevant application examples addressing varying aspects of complexity. 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: modeling and simulation; SSP; FMI; CATIA; Dymola; OMSimulator; OpenModelica

check for updates

Citation: Hällqvist, R.; Munjulury, R.C.; Braun, R.; Eek, M.; Krus, P. Realizing Interoperability between MBSE Domains in Aircraft System Development. *Electronics* **2022**, *11*, 2901. https://doi.org/10.3390/electronics11182901

Academic Editor: Rashid Mehmood

Received: 27 July 2022
Accepted: 5 September 2022
Published: 13 September 2022

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. The latter view on the application of MBSE was selected early on in the Saab *Gripen E/F* [1] project where the best-suited tool for each engineering task was preferred over a single Swiss army knife type tool [2]. Furthermore, 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. Development tools continuously evolve at different rates, and the effort required to maintain non-standardized tool couplings likely increase as a result. This motivates the utilization of standards to ensure continuity and consistency in the digital thread. The use of common and standardized formats for the integration of digital artifacts is highlighted as essential by the International Organization for Standardization (ISO) in achieving interoperability throughout collaborative organizations, see ISO 14258 [3].

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 their constituent 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 [4] 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* [5]. 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 [6–8]. The interdependent processes 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 [9,10]. The process description presented in Figure 1 can be mapped to that of an *enterprise model*. In alignment with the nomenclature of ISO 14258, an enterprise model is an abstraction representing the basic elements, in this case, the different collaborating departments within the overall organization, along with the information required for integrating the basic elements into a functioning enterprise. Furthermore, an enterprise is defined in ISO 14258 as

"A group of processes that one wishes to analyze." [3]

The work presented here does not focus on the development and use of enterprise models. The focus is instead placed on the use of standardized formats and technology for information exchange between elements within an enterprise model, i.e., the different levels of abstraction presented in Figure 1. However, the work may very well contribute to the development of enterprise models by means of motivating the use of specific standards throughout the organization.

The SSP [11] and FMI [12] standards provide standardized formats for establishing interoperability between the different levels of abstraction. See Sections 2.1 and 2.2 for descriptions of the relevant aspects of the respective standards. In short, the FMI standard concerns the export of models for integration in *simulator* [13] 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.

Digital applications at the different levels of abstractions are often developed and managed by different departments throughout the overall organization. FMI and SSP provide a tool agnostic mapping between disciplines that accounts for any differences in their respective processes, i.e., if applying the nomenclature of ISO 14258, the standards enable the transformation of legacy models to the *unified* and reusable state [3]. Such a mapping is specified as essential for achieving interoperability by Zacharewicz et al. in [14].

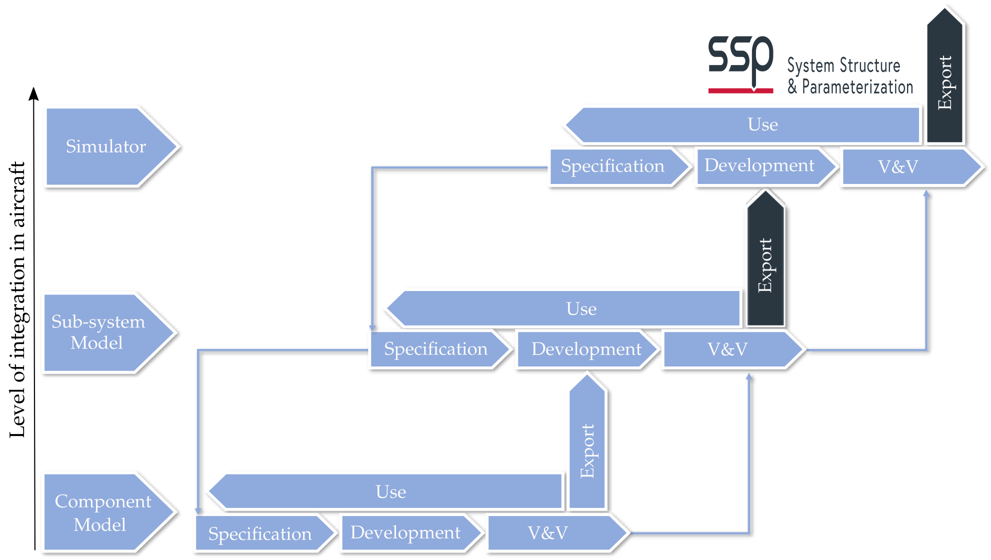


Figure 1. Simulation application development process connecting the Component Model, Sub-system Model, and Simulator levels of abstraction. The term *Model* here refers to mathematical simulation models as specified by Ljung [9] and Fritzon [10]. A simulator here is also seen as a mathematical simulation model composed of several connected models or simulators exported from a lower level of hierarchy [13]. The V&V activities are seen as bottom up, a view that is in line with the *Validation Level per Level* approach [6]. The natural connection between the two standards, FMI and SSP, and the activities of the simulation application development process are highlighted in the figure.

Furthermore, the SSP standard provides a standardized format for specifying the parameters of M&S artifacts representing more than a single realized aircraft system, sub-system, 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.

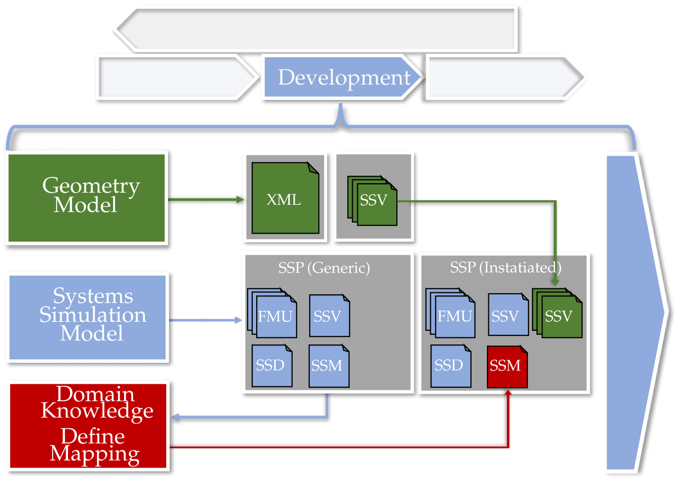


Figure 2. Expansion of the development activity specified in Figure 1. The workflow is designed to be applicable to the three system levels shown in Figure 1: the Component Model, Sub-System Model, and Simulator levels of abstraction. The colors in the figure indicate the source of the information. The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

The reuse of digital artifacts, such as models, is seen as essential in transferring knowledge between development projects. It is therefore important that any introduced methodology, such as model-centric information exchanged using open tools and standards, have a minimal impact on the feasible reuse. A method for automatically appending legacy models with the information necessary to facilitate the developed interoperability is seen as an important result presented herein.

This paper is an extension to the work presented in [15]; particularly focusing on scalability aspects of the previous research efforts. The paper is structured in the following way. In Section 2, the enablers of the work are introduced and related to the presented research. 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 two different application examples are described; the first application example is subjected to a use-case presented in Section 6. The use-case results are presented and discussed in Section 7 along with the results of the second application example. 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 are by no means a new research area, and a plethora of solutions have been proposed in the literature, including published research led by Saab [16].

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 [17]. Similarly, Elmqvist et al. developed a tool for the automatic translation of CATIA multibody models into Modelica code [18]. Remond et al. employ a similar approach to generating Modelica models of thermo-fluid piping networks based on CAD data from CATIA [19].

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 [20]. 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 that are used and what parameters that 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 [16].

Even though it is similar, the focus of the research presented here is instead only on 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 abovementioned problem.

High Level Architecture (HLA) is a standard for establishing interoperability between distributed simulation models, favoring model reuse and tool interoperability [21]. The HLA standard not only considers the assembly of coupled simulation models but also, much like the FMI standard, provides a standardized model format for model export and integration. Even though interoperability between HLA and FMI was shown to be successful by Sievert among others [22], HLA is mentioned merely for context in the present work. The focus is instead placed on the FMI and SSP standards which jointly aim to solve the same model exchange and architecture specification problem as HLA. This delimitation is motivated by the FMI and SSP available and stipulated future tool support. Furthermore, the FMI and SSP are standards that are developed and maintained by the Modelica Association; the Modelica Association is driven by the tool vendors and industrial parties also maintaining and improving on the Modelica modeling language which is frequently used at Saab Aeronautics.

2.1. Functional Mock-Up Interface (FMI)

The FMI standard specifies a format for models exported for use outside of its original development tool. A model exported according to the FMI standard is referred to as an FMU in the specification. Such an exported model is packaged in a *<model>.fmu* file. This file contains the model compiled binaries (or source code for *source code Functional Mock-up Units (FMUs)*) along with an Extensible Markup Language (XML) file denoted as *ModelDescription*. In short, the ModelDescription describes the model interface along with a sub-set of the model properties necessary for simulation execution, according to a standardized XML schema. The FMI standard can, at the time of this writing, be considered mature with over 170 tools formally supporting the various versions and functionality of the standard [23]. There are currently three major versions of FMI available, FMI 1.0 [24], 2.0 [12], and 3.0 [25]. The work presented here is exploiting FMI 2.0 as it is supported by a large number of the tools currently in use at Saab Aeronautics. Any adoption of FMI 3.0 is left for the future once sufficient tool support for this latest version is available to the end users.

2.2. System Structure and Parameterization (SSP)

Three of the SSP XML formats are the focal point of the research presented here: the System Structure Description (SSD), theSystem Structure Parameter Values (SSV), and theSystem 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 models. 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 file is often more volatile as it is 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.

The SSP standard is not developed as FMI exclusive and neither is the approach presented here. This translates to the fact that it is fully possible to populate an SSP file with, for example, Modelica models instead of FMUs. However, as visualized in Figure 1, combining the two standards enables interoperability between all the considered M&S application levels of abstraction. The industrial feasibility and strengths of the FMI and SSP standards have been demonstrated in numerous publications within the aeronautical industry [26–28] and the automotive industry [29]. Furthermore, initiatives related to managing SSP and FMI compliant digital artifacts throughout a product life cycle is ongoing in both industry and academia. See for example Coïc et al. [30] and their work on the *LOTAR* manifest that aims to provide digital artifacts with unique identifiers ensuring traceability throughout the life-cycle stages. Similarly, the SSP Traceability layer upon the SSP standard currently under development aims to specify a tool-independent XML format incorporating life-cycle information required to realize a credible simulation process in terms of quality assurance, traceability, and re-usability [31].

2.3. Use, Exchange, and Manipulation of SSPs

The FMI and SSP standards are both utilized together with *Transmission Line Modeling (TLM)* [32,33] in the simulation environment *OMSimulator* [34]. The OMSimulator is an open-source master simulation tool originally developed during the ITEA3 project *OpenCPS* [35]. It is based on a simulation framework developed by the Swedish bearing manufacturer SKF for connecting models of bearings with models from external tools [34,36]. 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 textual development of simulators. Other tools with similar support are available, and a list of the tools officially supporting the SSP standard is provided at the SSP project web page [37]. The two M&S tools, *FMIComposer* [38] and *FMIGo* [39], are mentioned for context as both provided FMI and SSP support early on.

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")
6) oms.reduceSSV("<model>","<submodel>.ssv","<submodel>.ssm")
```

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 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 an 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. The final listed API command, *oms.reduceSSV*, is particularly relevant for generated SSV files with many unused parameter values. The command removes all elements of the SSV file that are not bound to any parameter in the executable via a function input SSM file. A complete description of all the available OMSimulator API functions is provided in the user manual [40].

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 [41]. 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 geometry model components in the two application examples, see Section 3, can be modified alongside any changes in the respective system specifications. The *User Defined Features (UDF)* created in CATIA encapsulates the design intent and design automation is applied 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 [16,42,43], Visual Basic for Applications (VBA) is exploited to extract parameter information from CATIA via a set of macros. 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 mappings, as described in Section 3.2, are applied to the intermediate XML visualized in Figure 2.

3.1. Parameter Extraction and SSP Support

An approach similar to that of Munjulury et. al. [42], 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. These identifiers are only specified once for each parameter set of interest.
- The 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.

3.3. Adaptation of Legacy Models

A fundamental requirement of the work presented is that any introduced changes to the overall M&S methodology should have minimal impact on the modeling methods applied in each individual engineering discipline. Tool support to prepare legacy geometry models is as a result a focus area.

The following semi-automatic procedure describes the methodology that is created for retrieving and expressing the parameters' values in an SSV file; VBA programming in EXCEL is used for the processes.

1. First, the *main assembly* is searched for the required Parts/Products in ENOVIA VPM. The *main assembly* could contain several sub-assemblies or Parts, and it can be browsed using the option "In Context, With Children at all Levels" in the *Open Modes* dialog. This is performed to open all the parts in the respective assemblies and sub-assemblies for both the investigated legacy and non-legacy models.
2. In the next step, all the names of all the opened Parts/Products are read into an EXCEL worksheet. Here, a manual cross-check should be performed. Such a cross-check ensures that all the Parts/Products inside the *main assembly* that are needed in the next steps are available and opened in Context. If the *main assembly* is not opened in Context, then only the name of the Products in a few assembly levels are read into the EXCEL worksheet.

3. The above names are copied to another worksheet for manual input of all the respective connections in the Simulation model. This is performed in order to sort and group two or more parts. A corresponding name in the Simulation model is added next to the Parts/Products that are represented in CATIA.
4. A sorting operation is performed, and only the Parts/Products that have a corresponding simulation name are copied to a new Worksheet. This is used for copying only the necessary parts, assemblies, and sub-assemblies. It also helps to narrow down the number of Parts/Products that are to be copied.
5. The Parts that have pre-defined names of interest, for example, *Pipe* or *Hose*, are identified in the opened *main assembly*. They are copied and pasted with the Break link option in a new Assembly. By using the Break link option upon creation in a new assembly, the legacy model is left unchanged whereas the adapted model is available for further manipulation. Both these model versions should be placed under stringent configuration control to ensure traceability.
6. Each part of the adapted model is opened in a new window in CATIA ,and a semi-automatic operation is performed. This step is performed in two operations.
   (a) Manual operation is needed to organize the points and curves that are used to create the *Pipes* or *Hoses*. The points are then used to compute the angle of the bends, and the curve is used to obtain the overall length of the *Pipe* or *Hose*.
   (b) The macro program is used to perform the following tasks.
      - To add the required input and output Parameters,
      - Create formulas by connecting the Parameters to diameters of the *Pipe* or *Hose*,
      - Create the coordinates of all the points that were organized as parameters,
      - Create the bend angles, depending on the number of points in the manual operation,
      - Finally, the overall details of the *Hose* or *Pipe*, such as length, diameters, number of bends, and the pressure drop coefficient.

The geometry model, including all the necessary meta-information, is available once the above steps have been completed. This updated geometry model can be used to express parameter values in the SSV format using the functionality described in Sections 3.1 and 3.2. The preparation described in this section needs to be executed once for each legacy geometry model of interest.

For the methodology to work all the time, there needs to be consistency in the naming of the Parts/Products. This will eradicate cumbersome programming and make it easy to search the required names of Parts, Products, Parameter sets, or Geometrical sets. The names of the abovementioned entities are the identifiers that need to be searched to retrieve the required Parameters. In the case of the Legacy model, some geometrical entities such as points and curves are not organized, If all the geometry is organized from the beginning, creating the required parameters for the simulation can be fully automated.

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 here is 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, a sub-system model, or a 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. The third-party Modelica Association (MA) Modelica library *ExternData* [44] offers the possibility of incorporating parameter values in SSV files into Modelica models. Even though most relevant, the library has not been exploited in the presented research.

5. Application Examples

Two different application examples are used to specify and develop the interoperability aspects in focus herein. Both the two application examples incorporate the targeted engineering domains and M&S tools.

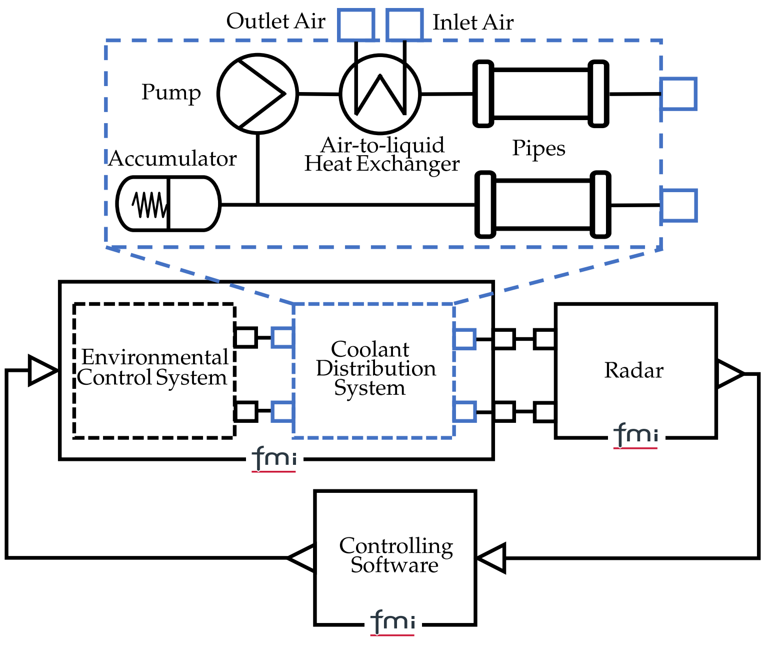
- The first described example is primarily used to develop the needed functionality. It is kept fully open and made available to all the stakeholders, including developers of both the relevant open-source and commercial tools. This example is from hereon referred to as *application example one*, and it is made publicly available in the OMSimulator test suite [45].
- The second application example is primarily used for evaluating the scalability of the proposed methodology along with the corresponding functionality. This example is a model of a proprietary coolant distribution system, and it is therefore not described in as much detail. This example is from hereon referred to as *application example two*.

Application example one is subjected to the use-case presented in Section 6.1, demonstrating the overall use of M&S as presented in Figure 1. The use-case is not applied in its entirety to application example two in the presented research. It is merely used to export the relevant parameter values from the geometry domain such that they can be used during simulation execution.

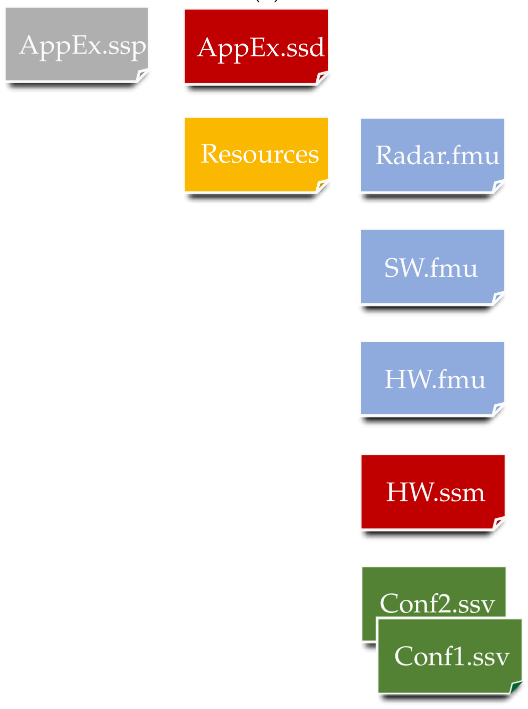
That being said, a schematic overview representing both application examples is provided in Figure 3. The first application example is separated into three different main parts; each of these parts is exported as an FMU from its original development tool. The modeled *Coolant Distribution System* is highlighted as dashed and blue in Figure 3a as it here is the target of the established interoperability between systems simulation and geometry modeling. Proprietary application example two has a similar architecture; however, the coolant distribution system modeled is expressed in significantly more detail.

The FMUs of the first application example are packaged in an 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*.

The second application example is also packaged in an SSP file; this SSP contains a single FMU representing the coolant distribution system. It does, however, include the SSM and SSV files needed to incorporate parameter values extracted from the corresponding geometry model. The difference in scale of the two application example coolant distribution Modelica models are summarized in Table 1. Even though application example one is developed to capture the relevant requirements on methodology and tool support, application example two is needed to verify the industrial scalability of the research.


Outlet Air
Inlet Air
Pump
Air-to-liquid Heat Exchanger
Pipes
Accumulator
Environmental Control System
Coolant Distribution System
Radar
fmi
Controlling Software

(**a**)



(**b**)

Figure 3. Application example description. A schematic description representing both the application example architectures is presented in (**a**). The structure of the application example one resulting SSP file is provided in (**b**). (**a**) Schematic description capturing both application examples. 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 one 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. The SSP of the second application example merely describes a single configuration including one FMU, one SSV, and one SSM.

Table 1. Summary of the two application examples coolant distribution system Modelica implementations.

Property	Application Example One	Application Example Two
No. of equations	297	2064
No. of variables	708	5337
No. of components	46	155

5.1. Mathematical Modeling

Two of the three constituent parts of both the application examples are individual models of the mathematical sort. At this point, the two application examples start to differ. In application example one, the 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. In application example two, all constituent parts are represented as individual models. They are all detailed enough for it to be beneficial to keep them separated in terms of reuse and configuration management.

The development of these aircraft sub-systems is typically conducted by different departments at Saab or by suppliers, and this partitioning of both examples reflects a likely situation during development. All three sub-system models of hardware are developed using components from the Saab Aeronautics in-house Modelica library *Modelica Fluid Light* [46] and the *Modelica Standard Library* [47]. The mathematical modeling is conducted in the Dymola and OpenModelica SW.

5.1.1. Environmental Control System (ECS)

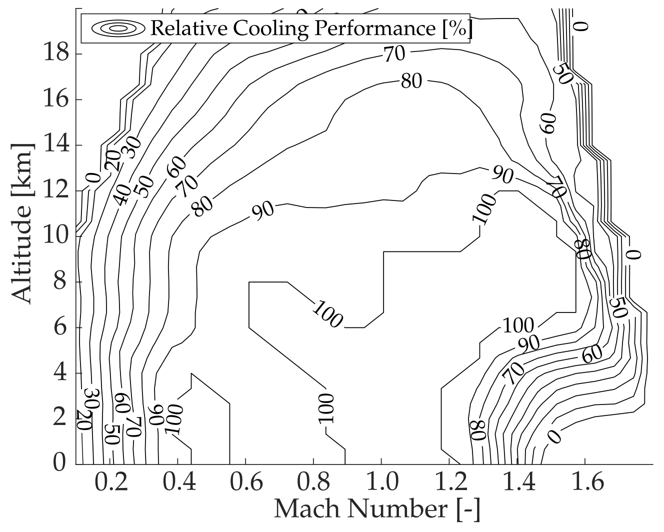
A simulator enabling detailed studies of pilot thermal comfort was presented in [26]. The ECS incorporated in application example one is based on the aircraft cooling system of that simulator. The ECS model is intended to provide a connection between the operating conditions and the cooling power available 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 4.

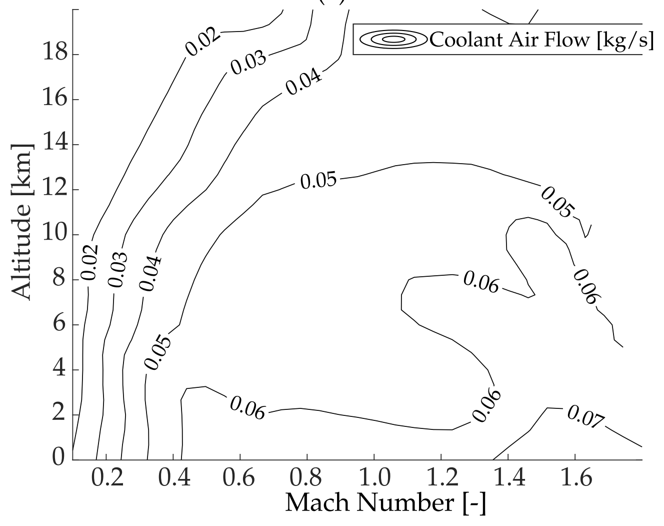
Provided the characteristics of Figure 4, the minimum possible cold air temperature can be calculated as

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

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 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.



(**a**)



(**b**)

Figure 4. Steady-state characteristics of the application example ECS: (**a**) ECS available relative cooling power ($\dot{Q}_{rel}$) as function of altitude and Mach number; (**b**) ECS available coolant flow ($\dot{m}$) as a function of altitude and Mach number.

5.1.2. Coolant Distribution System

The two application examples coolant distribution systems are 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 examples geometry models, are presented in Table 2. A sub-set of the included model component equations is described in detail in the following paragraphs to highlight how the selected parameters impact the characteristics of the coolant distribution system model.

Table 2. Summary of the parameters that are exchanged in the application examples. 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}.

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

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 [48]. 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 2 also impact 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:

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

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

$$p = \frac{p_f - p_e}{V_{acc}}\left[\sum_{i=1}^{N}(V_i(T) - V_i(T_0)) + V_{acc}(T_0)\right] + p_e \tag{3}$$

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 is 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 sides. 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 [49].

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 the fact that the gas temperature is constant perpendicular to the direction of the flow. The hot and cold side are $i = h$ and $i = c$, respectively, and heat transfer rates can be expressed as

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

where A_i is the total heat transfer area of side i. The Stanton number S_t in Equation (4), named after Thomas Stanton, is a heat transfer modulus that is used to characterize heat transfer [50]. 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

$$R = \sum_{i=c}^{h} \frac{1}{h_i A_i} \tag{5}$$

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

$$\epsilon = 1 - e^{-\left[1 - e^{-N_{tu} \frac{C_{min}}{C_{max}}}\right] \frac{C_{max}}{C_{min}}} \tag{6}$$

or

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

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 (6) is applicable, and if $C_{min} = C_h = \dot{m}_h C_{p_h}$ then Equation 7 is applicable. The thermal resistance influences the efficiency via the heat transfer parameter

$$N_{tu} = \frac{1}{RC_{min}} \tag{8}$$

which connects the geometrical parameters to the efficiency. In the application examples LHEX models $l_h = 2 \cdot b$, 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 a function of the efficiency

$$T_h^{out} = T_h^{in} - \epsilon (T_h^{in} - T_c^{in}) \tag{9}$$

where the superscript indicates inlet or outlet temperature.

5.2. Geometrical Representations of Application Example One

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.

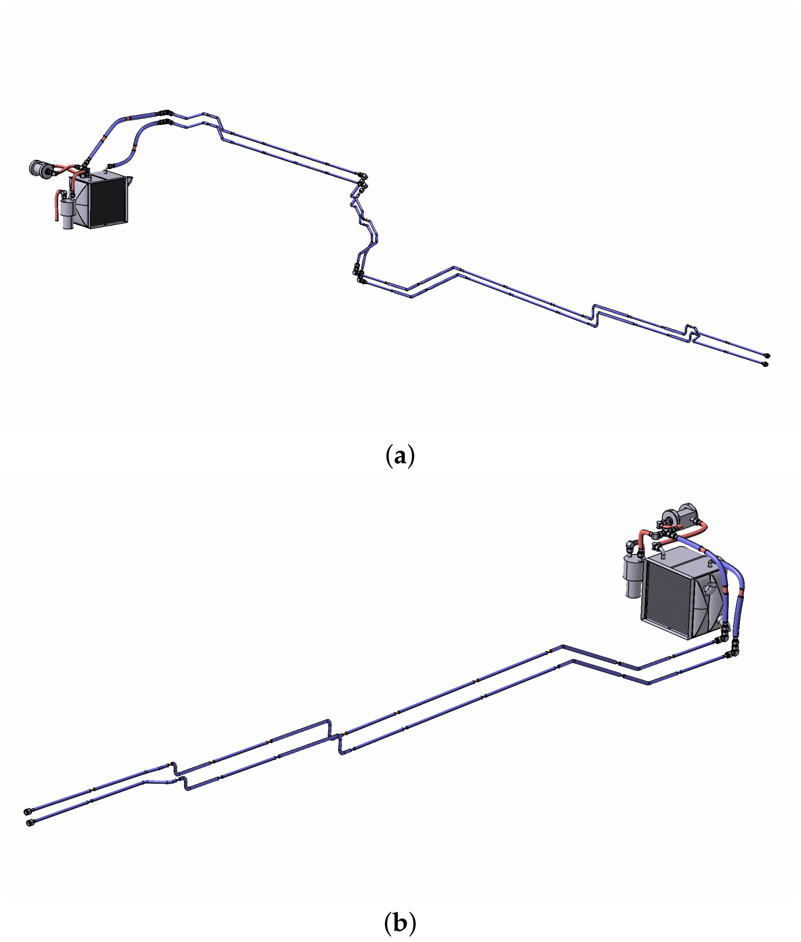


Figure 5. Use-case geometry models representing the two different routing options under investigation. The piping reaches from the LHEX to the front of the aircraft where the radar is located. The radar is not included in either (**a**) or (**b**): (**a**) geometry model of cooling power distribution 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 one (*Configuration 1*). The ECS is located in the aft of the aircraft.

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 lies 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 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) [6] mission, along with a sub-system requirement posed by the hypothetical developer of the application example one radar, together compose the use-case requirements on the coolant distribution system.

6.1. Prerequisites

Application example one described in Section 5 naturally serves as the primary use-case prerequisite. Here, application example one 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 that 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.

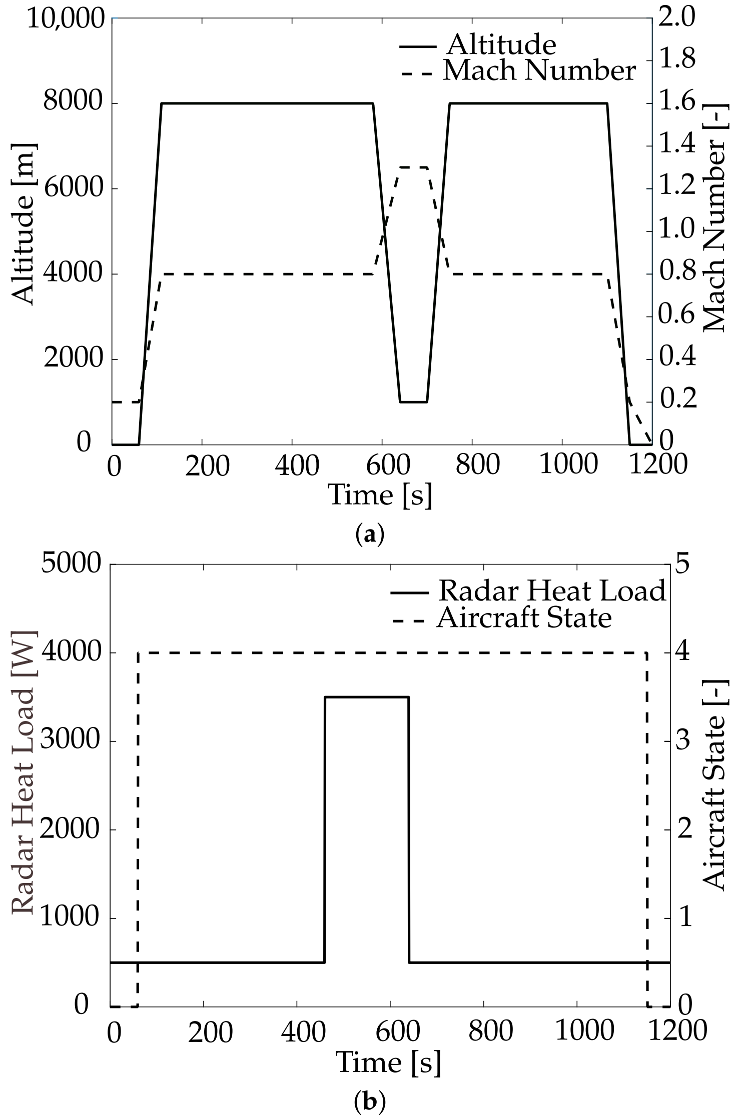


Figure 6. Boundary conditions corresponding to the specified OpsCon mission profile: (**a**) altitude and Mach number of the OpsCon mission profile; (**b**) heat load exerted by the application example radar component during the OpsCon mission. The radar here can operate in two different discrete modes: a stand-by mode corresponding to 500 W of power and an 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).

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 sub-system requirement, *the difference in radar coolant inlet and outlet temperature shall not exceed* 13 °C, is fulfilled throughout the OpsCon mission.

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 proposes 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.

```xml
<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.

```xml
<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 the 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 results of the research are presented and discussed in this section. The results are separated according to the two incorporated examples: *application example one* and *application example two*. The results corresponding to the foremost are a result of traversing every aspect of the presented use case, see Section 6. The results of the latter are presented to the degree necessary to further demonstrate industry-grade feasibility and scalability of the interoperability established herein.

7.1. Application Example One

Application example one is simulated for the mission profile described in Section 6.1. The geometry settings of the two different modeled configurations are summarized in Tables 3 and 4.

Table 3. Summary of parameter values that differ between the two configurations of application example one. The cooling distribution system routing is subject to modification. The remaining coolant distribution system components, along with their constituent parameters, remain unchanged.

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

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

The simulation results used to assess the feasibility of the two different configurations, with respect to the requirements are shown in Figures 7 and 8. Figure 7 quantifies the performance limits during the OpsCon mission of the included ECS model. The available cooling power is shown solid in the figure, and the available coolant mass flow is 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), which are presented in Table 3.

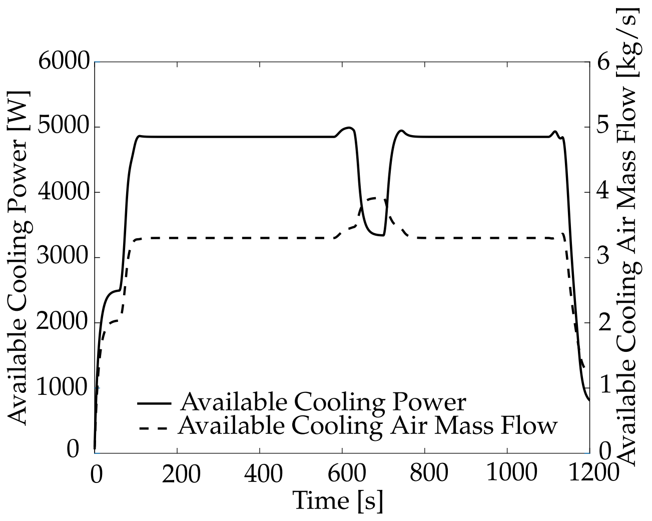
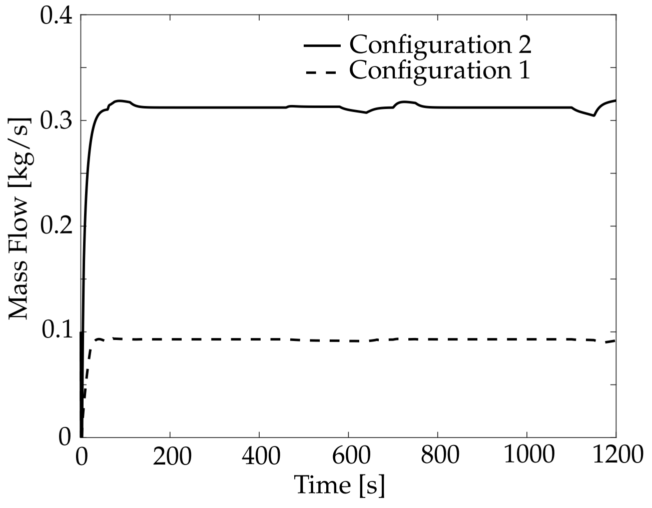
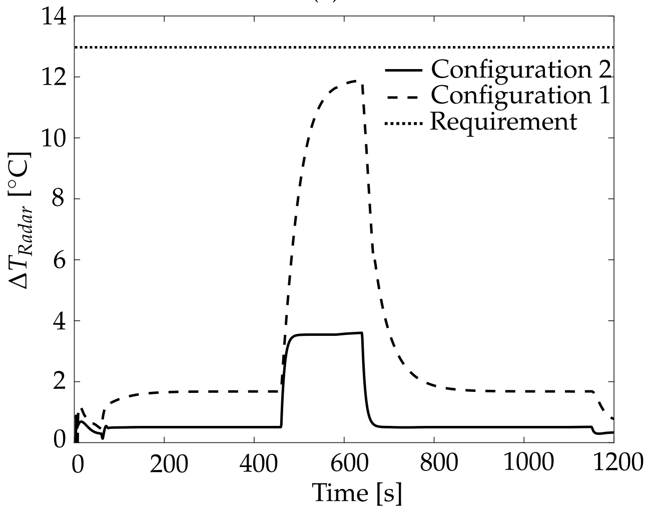


Figure 7. ECS model available cooling power (solid) and the available coolant mass flow as (dashed) for the OpsCon simulations.



(**a**)



(**b**)

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; (**a**) coolant distribution system mass flow during the two OpsCon simulations; (**b**) temperature increase over the radar during the two OpsCon simulations.

7.2. Application Example Two

The results related to the second application example are best summarized by the fact that an SSV file has been created from an industry-grade legacy geometry model which was successfully adapted by means of exploiting the tool functionality presented in Section 3.3. The authors would like to highlight that the preparation of the legacy model, using the developed tool functionality, is completed in minutes not hours. This is an essential result in terms of technology adoption in the industry. The properties of the application example SSV files are summarized in Table 5 which quantifies the evaluated method scalability. It is to be noted that only parameters of parts such as *Hoses* and *Pipes* are saved to an SSV file. Other parts such as Insulation, connectors between *Pipes* or *Hoses*, Clamps, or any other supporting parts are not considered at this moment. Adding such parameters affect the scale of the resulting SSV file; however, they are not deemed to affect the overall conclusions, and the implementation is as a result left for future work.

Table 5. Summary of the two application example SSV files exported from the respective geometry models.

Property	Application Example One		Application Example Two
	Configuration 1	*Configuration 2*	
Total Parts/Products	37	54	1686
Pipe or Hose Parts	8	12	81
Total Parameters	510	886	3763
File size SSV (KB)	115	200	1200

8. Conclusions

There is much to gain 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 suitable open tools, and standards. The feasibility of the work has been demonstrated by means of simulating an OpsCon mission where two different aircraft coolant distribution system designs are evaluated against a formulated requirement, see Section 7.

The method has been developed while keeping the aim of minimizing the impact on the modeling methodologies, mathematical or geometrical, in mind. The application examples 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 put into the context of the simulation model development and maintenance processes currently deployed at Saab Aeronautics by means of two different application examples. 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.

The presented research has provided feedback to the SSP design group for upcoming standard development in terms of round-trip engineering [51,52]. That being said, the research has been realized successfully on industry-grade examples using SSP 1.0. Any improvements to the standard identified in relation to the presented work are related to exploitation during design where SSP files are evolved from incomplete to fully executable.

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 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.

Author Contributions: R.H. and R.C.M. should be seen as the main authors of the presented work but all listed authors have contributed substantially. Conceptualization, R.H., R.C.M. and M.E.; methodology, R.H. and R.C.M.; software, R.H. and R.C.M.; validation, R.H.; writing—original draft preparation, R.H. and R.C.M.; writing—review and editing, R.B., M.E. and P.K.; supervision, R.B., M.E. and P.K.; project administration, R.H., M.E. and P.K.; funding acquisition, R.H., M.E. and P.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Vinnova and Saab Aeronautics under grant numbers 10.13039/501100001858 and 2019-02386.

Acknowledgments: 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 for reviewing the manuscript.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

CAD	Computer Aided Design
DOM	Document Object Model
ECS	Environmental Control System
FMI	Functional Mock-up Interface
FMU	Functional Mock-up Unit
HLA	High Level Architecture
KBE	Knowledge-Based Engineering
LHEX	Air-to-Liquid Heat Exchanger
MBSE	Model-Based Systems Engineering
M&S	Modeling & Simulation
OpsCon	Operational Concept
OSMC	Open Source Modelica Consortium
RBF	Radial Basis Function
SSD	System Structure Description
SSP	System Structure and Parameterization
SSM	System Structure Parameter Mapping
SSV	System Structure Parameter Values
UDF	User Defined Features
UI	User Interface
VBA	Visual Basic for Applications
V&V	Verification & Validation
XML	Extensible Markup Language

References

1. SAAB AB . Gripen E-Series, Sales and Marketing E-Series. Available online: https://www.saab.com/products/gripen-e-series (accessed on 26 July 2022).
2. Walle'n Axehill, J.; Herzog, E. Don't Mix the Tenses: Managing the Present and the Future in an MBSE Context. In Proceedings of the 32nd INCOSE International Symposium, Detroit, MI, USA, 25–30 June 2022.
3. International Organization for Standardization (ISO). ISO 14258:1998/COR 1:2000 Industrial Automation Systems—Concepts and Rules for Enterprise Models—Technical Corrigendum 1. Available online: https://www.iso.org/standard/33536.html (accessed on 18 August 2022).
4. NASA. *Handbook for Models and Simulations: An Implementation Guide for NASA-STD-7009*; Technical Report, NASA-HDBK-7009; National Aeronautics and Space Administration: Washington, DC, USA, 2013.
5. Andersson, H.; Carlsson, M. *Saab Aeronautics Handbook for Development of Simulation Models: Public Variant*; Technical Report 12/00159; Machine Design; Linköping University: Linköping, Sweden, 2012.
6. Walden, D.D.; Roedler, G.J.; Forsberg, K.; Hamelin, R.D.; Shortell, T.M. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, 4th ed.; John Wiley & Sons, Inc.: San Diego, CA, USA, 2015.
7. Roza, M.; Voogd, J.; Sebalj, D. The Generic Methodology for Verification and Validation to support acceptance of models, simulations and data. *J. Def. Model. Simul. Appl. Methodol. Technol.* **2012**, *10*, 347–365. [CrossRef]
8. Roy, C.J.; Oberkampf, W.L. A comprehensive framework for verification, validation, and uncertainty quantification in scientific computing. *Comput. Methods Appl. Mech. Eng.* **2011**, *200*, 2131–2144. [CrossRef]
9. Ljung, L.; Glad, T. *Modellbygge Och Simulering*; Studentlitteratur: Lund, Sweden, 2004; Volume 2.
10. Fritzson, P. *Principles of Object Oriented Modeling and Simulation with Modelica 2.1*; Wiley-IEEE Press: Hoboken, NJ, USA, 2004. [CrossRef]
11. Modelica Association. System Structure and Parameterization, Report 1.0. 2019. Available online: https://ssp-standard.org/publications/SSP10RC1/SystemStructureAndParameterization10RC1.pdf (accessed on 5 March 2019).
12. FMI Development Group. Functional Mock-Up Interface for Model Exchange and Co-Simulation, Report 2.0.3. Available online: https://fmi-standard.org/downloads/ (accessed on 15 November 2021).
13. Hällqvist, R. On Standardized Model Integration: Automated Validation in Aircraft System Simulation. Licentiate's Thesis, Linköping Studies in Science and Technology, Linköping University, Linköping, Sweden, 2019; Volume 1866. [CrossRef]
14. Zacharewicz, G.; Daclin, N.; Doumeingts, G.; Haidar, H. Model Driven Interoperability for System Engineering. *Modelling* **2020**, *1*, 94–121. [CrossRef]
15. Hällqvist, R.; Munjulury, R.C.; Braun, R.; Eek, M.; Krus, P. Engineering Domain Interoperability Using the System Structure and Parameterization (SSP) Standard. In Proceedings of the 14th Modelica Conference, Linköping, Sweden, 20–24 September 2021; pp. 37–48. [CrossRef]
16. Lind, I.; Oprea, A. Detailed geometrical information of aircraft fuel tanks incorporated into fuel system simulation models. In Proceedings of the 9th International MODELICA Conference, Munich, Germany, 3–5 September 2012. [CrossRef]
17. Engelson, V.; Larsson, H.; Fritzson, P. A design, simulation and visualization environment for object-oriented mechanical and multi-domain models in Modelica. In Proceedings of the 1999 IEEE International Conference on Information Visualization (Cat. No. PR00210), London, UK, 14–16 July 1999; pp. 188–193. [CrossRef]
18. Elmqvist, H.; Mattsson, S.E.; Chapuis, C. Redundancies in Multibody Systems and Automatic Coupling of CATIA and Modelica. In Proceedings of the 7th International Modelica Conference, Como, Italy, 20–22 September 2009. [CrossRef]
19. Remond, X.; Gengler, T.; Chapuis, C. Simulation of Piping 3D Designs Powered by Modelica. In Proceedings of the 11th International Modelica Conference, Versailles, France, 21–23 September 2015. [CrossRef]
20. Baumgartner, D.; Pfeiffer, A. Automated Modelica Package Generation of Parameterized Multibody Systems in CATIA. In Proceedings of the 10th International Modelica Conference, Lund, Sweden, 10–12 March 2014. [CrossRef]
21. Dahmann, J.S.; Fujimoto, R.M.; Weatherly, R.M. The department of defense high level architecture. In Proceedings of the 29th Conference on Winter Simulation, Atlanta, GA, USA, 7–10 December 1997; pp. 142–149. [CrossRef]
22. Sievert, N. *Modelica Models in a Distributed Environment Using FMI and HLA*; Department of Computer and Information Science, Software and Systems, Linköping University: Linköping, Sweden, 2016.
23. FMI Development Group. FMI Tools. Available online: https://fmi-standard.org/tools/ (accessed on 18 August 2022).
24. MODELICA Association Project FMI. Functional Mock-Up Interface for Model Exchange, Report 1.0.1. Available online: https://fmi-standard.org/downloads/ (accessed on 10 September 2017).
25. FMI Development Group. Functional Mock-Up Interface Specification, FMI for Model Exchange, Co-Simulation, and Scheduled Execution, Report 3.0. Available online: https://fmi-standard.org/downloads/ (accessed on 10 May 2022).
26. Hällqvist, R.; Schminder, J.; Eek, M.; Braun, R.; Gårdhagen, R.; Krus, P. 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, Belo Horizonte, Brazil, 9–14 September 2018.
27. Oprea, A.; Hällqvist, R.; Knöös Franzén, L.; Eek, M.; Staack, I.; Gavel, H. Connecting System Simulation to Aircraft Concept Development. In Proceedings of the 32st Congress of the International Council of the Aeronautical Sciences, Pudong Shangri-La, Shanghai, China, 6–10 September 2021.

28. Schminder, J.; Hällqvist, R.; Eek, M.; Gårdhagen, R. Pilot Performance and Heat Stress Assessment Support Using a Cockpit Thermoregulatory Simulation Model. In Proceedings of the 31st Congress of the International Council of the Aeronautical Sciences, Belo Horizonte, Brazil, 9–14 September 2018.
29. Köler, J.; Heinkel, H.M.; Mai, P.; Krasser, J.; Deppe, M.; Nagasawa, M. Modelica-Association-Project "System Structure and Parameterization"—Early Insights. In Proceedings of the 1st Japanese Modelica Conference, Tokyo, Japan, 23–24 May 2016.
30. Coïc, C.; Murton, A.; Mendo, J.C.; Williams, M.; Tummescheit, H.; Woodham, K. Modelica, FMI and SSP for LOTAR of Analytical mBSE models: First Implementation and Feedback. In Proceedings of the 14th Modelica Conference, Linköping, Sweden, 20–24 September 2021; pp. 37–48. [CrossRef]
31. SSP Traceability Development Group. SSP Traceability Specification, Version 1.0.0-Beta2, Unreleased. Available online: https://pmsfit.github.io/SSPTraceability/master/ (accessed on 17 August 2022).
32. Auslander, D.M. Distributed System Simulation With Bilateral Delay-Line Models. *J. Basic Eng.* **1968**, *90*, 195–200. [CrossRef]
33. Krus, P.; Jansson, A.; Palmberg, J.O.; Weddfeldt, K. Distributed Simulation of Hydromechanical Systems. In Proceedings of the Third Bath International Fluid Power Workshop, Bath, UK, 12–13 September 1990.
34. Ochel, L.; Braun, R.; Thiele, B.; Asghar, A.; Buffoni, L.; Eek, M.; Fritzson, P.; Fritzson, D.; Horkeby, S.; Hällquist, R.; et al. OMSimulator—Integrated FMI and TLM-based Co-simulation with Composite Model Editing and SSP. In Proceedings of the 13th International Modelica Conference, Regensburg, Germany, 4–6 March 2019. [CrossRef]
35. OpenCPS Project Partners. Project 14018 : Open Cyber-Physical System Model-Driven Certified Development. Available online: https://www.opencps.eu/ (accessed on 21 June 2018).
36. Fritzson, P.; Pop, A.; Abdelhak, K.; Ashgar, A.; Bachmann, B.; Braun, W.; Bouskela, D.; Braun, R.; Buffoni, L.; Casella, F.; et al. The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development. *Model. Identif. Control. A Nor. Res. Bull.* **2020**, *41*, 241–295. [CrossRef]
37. Modelica Association Project. System Structure and Parameterization. Available online: https://ssp-standard.org (accessed on 8 May 2021).
38. Modelon. FMI Composer User's Guide 1.0. Available online: https://www.modelon.com/products-services/modelon-deployment-suite/ (accessed on 10 November 2017).
39. Lacoursière, C.; Härdin, T. FMI Go! A simulation runtime environment with a client server architecture over multiple protocols. In Proceedings of the 12th International Modelica Conference, Prague, Czech Republic, 15–17 May 2017. [CrossRef]
40. Ochel, L. OMSimulator's Documentation. Available online: https://openmodelica.org/doc/OMSimulator/master/html/ (accessed on 18 February 2021).
41. Stokes, M. *Managing Engineering Knowledge : MOKA: Methodology for Knowledge Based Engineering Applications*, 1st ed.; John Wiley and Sons: Hoboken, NJ, USA, 2001; p. 310.
42. Munjulury, R.C.; Staack, I.; Berry, P.; Krus, P. A knowledge-based integrated aircraft conceptual design framework. *CEAS Aeronaut. J.* **2016**, *7*, 95–105. [CrossRef]
43. Munjulury, R.C. Knowledge-Based Integrated Aircraft Design: An Applied Approach from Design to Concept Demonstration. Ph.D. Thesis, Linköping Studies in Science and Technology, Linköping University, Linköping, Sweden, 2017; p. 72. [CrossRef]
44. Beutlich, T.; Winkler, D. Efficient Parameterization of Modelica Models. In Proceedings of the 14th Modelica Conference, Linköping, Sweden, 20–24 September 2021; pp. 141–146. [CrossRef]
45. Open Source Modelica Consortium. OpenModelica/OMSimulator. Available online: https://github.com/OpenModelica/OMSimulator/tree/master/testsuite (accessed on 18 August 2022).
46. Eek, M.; Gavel, H.; Ölvander, J. Definition and Implementation of a Method for Uncertainty Aggregation in Component-Based System Simulation Models. *J. Verif. Valid. Uncertain. Quantif.* **2017**, *2*, 011006. [CrossRef]
47. The Modelica Association. Modelica and the Modelica Association. Available online: https://www.modelica.org/ (accessed on 21 June 2018).
48. Miller, D. *Internal Flow Systems*; BHRA Information Services: Cranfield, Bedford, UK, 1990.
49. Kays, W.M.; London, A.L. *Compact Heat Exchangers*; Krieger: Malabar, FL, USA, 1984.
50. Ackroyd, J.A.D. The Victoria University of Manchester's contributions to the development of aeronautics. *Aeronaut. J. 1968* **2007**, *111*, 473–493. [CrossRef]
51. Rosca, D.; Domingues, L. A Systematic Comparison of Roundtrip Software Engineering Approaches applied to UML Class Diagram. *Procedia Comput. Sci.* **2021**, *181*, 861–868. [CrossRef]
52. Sendall, S.; Küster, J. Taming model round-trip engineering. In Proceedings of the Workshop on Best Practices for Model-Driven Software Development (Satellite Event of the 19th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2004)), Vancouver, BC, Canada, 24–28 October 2004.