<table>
<thead>
<tr>
<th><strong>Deliverable No.</strong></th>
<th>HIFI-ELEMENTS D3.1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deliverable Title</strong></td>
<td>Set of SYNECT add-ons to import all model formats and set of frame model generators</td>
</tr>
<tr>
<td><strong>Deliverable Date</strong></td>
<td>2019-03-27</td>
</tr>
<tr>
<td><strong>Deliverable Type</strong></td>
<td>REPORT</td>
</tr>
<tr>
<td><strong>Dissemination level</strong></td>
<td>Public (PU)</td>
</tr>
<tr>
<td><strong>Written By</strong></td>
<td>Markus Deppe, dSPACE (WP3 leader)</td>
</tr>
<tr>
<td><strong>Checked by</strong></td>
<td>Jens Ewald, FEV (coordinator)</td>
</tr>
<tr>
<td><strong>Approved by</strong></td>
<td>Steering Committee</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Final</td>
</tr>
</tbody>
</table>
H2020-GV7-2017 – 769935 – High Fidelity Electric Modeling and Testing

Acknowledgement

The author(s) would like to thank the partners involved with the project for their valuable comments on previous drafts and for performing the review.

Project partners
1 – FEV – FEV Europe GmbH – DE
2 – RWTH – Rheinisch-Westfälische Technische Hochschule Aachen – DE
3 – RIC – Ricardo UK Limited – UK
4 – TEC – Fundacion TECNALIA Research & Innovation – ES
5 – CID – Fundacion CIDETEC – ES
6 – IVE – IVEco S.p.A. – IT
7 – DEN – Denso Automotive Deutschland GmbH – DE
8 – TOF – TOFAS Turk Otomobil Fabrikasi A. S. – TR
9 – IDI – IDIADA Automotive Technology SA – ES
10 – TNO – Nederlandse Organisatie voor Toegepast-natuurwetenschappelijk Onderzoek – NL
11 – MGE – Mondragon Goi Eskola Politiknikoa J.M.A. S.Coop – ES
13 – VUB – Vrije Universiteit Brussel – BE
14 – MAG – MAGNA Engineering Center Steyr GmbH & Co. KG – AT
15 – YASA – YASA Motors Limited – UK
16 – FORD – Ford Werke GmbH – DE

Disclaimer:
This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 769935.
**Document information**

**Additional author(s) and contributing partners**

<table>
<thead>
<tr>
<th>Name</th>
<th>Organisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pajenkamp, Thomas Max, Meyer</td>
<td>dSPACE RWTH</td>
</tr>
</tbody>
</table>

**Document Change Log**

<table>
<thead>
<tr>
<th>Name</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0.1</td>
<td>2018-12-13</td>
<td>Initial draft</td>
</tr>
<tr>
<td>V1.0</td>
<td>2019-03-27</td>
<td>Final</td>
</tr>
</tbody>
</table>

**Document Distribution Log**

<table>
<thead>
<tr>
<th>Name</th>
<th>Organisation</th>
<th>Distributed to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jens Ewald</td>
<td>FEV</td>
<td></td>
</tr>
</tbody>
</table>

**Verification and approval**

<table>
<thead>
<tr>
<th>Organization</th>
<th>Name</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinator</td>
<td>Jens Ewald</td>
<td>2019-03-07</td>
</tr>
<tr>
<td>Steering Committee</td>
<td></td>
<td>2019-03-13</td>
</tr>
</tbody>
</table>
Executive summary

This deliverable describes the SYNECT Add-Ons to import all model formats and to generate frame models. The SYNECT Add-Ons are part of the Agile Model Development Framework (AMDF) and support the respective model formats such as FMU, XMODEL and SIMULINK. The purpose of the model import is to validate the model interfaces against the architecture specification and to add the models to the SYNECT data management system. With the automatic export of a frame (also called stub, template or shell) which just contains the interface part (inputs and outputs) a new model can be started very quickly from scratch.
Contents

1 Purpose of the Document ............................................................................................................. 7
  1.1 Document Structure .................................................................................................................. 7

1.2 Deviations from original Description in the Grant Agreement Annex 1 Part A ......................... 7
  1.2.1 Description of work related to deliverable in GA Annex 1 – Part A ........................................ 7
  1.2.2 Time deviations from original planning in GA Annex 1 – Part A ........................................... 7
  1.2.3 Content deviations from original plan in GA Annex 1 – Part A .............................................. 7

2 Introduction .................................................................................................................................. 8
  2.1 Overall Target of the HIFI-ELEMENTS project ...................................................................... 8
  2.2 Target of Work Package 3 (WP3) ............................................................................................... 8

3 Motivation and Overview ............................................................................................................. 10
  3.1 Terminology .............................................................................................................................. 11
  3.1.1 SYNECT Add-On Concept .................................................................................................. 11
  3.1.2 List of SYNECT Add-Ons ................................................................................................... 12

4 Component Model Import ............................................................................................................ 13

5 SIMULINK Add-On ...................................................................................................................... 15
  5.1 Export Simulink Frame Model ................................................................................................. 15
  5.2 Export SIMULINK Test Frame ............................................................................................... 16
  5.3 Validate SIMULINK Model Interface ................................................................................... 17

6 FMI Add-On .................................................................................................................................. 19
  6.1 Export FMU Frame Model ........................................................................................................ 19
  6.2 Export FMU Shell Models ........................................................................................................ 20
  6.3 Validate FMI Model Interface ............................................................................................... 22

7 XMODEL Add-On ........................................................................................................................ 25
  7.1 Build XMODEL Frame Model ............................................................................................... 25
  7.2 Validate XMODEL Interface ............................................................................................... 25

8 Summary ...................................................................................................................................... 28

9 Risk Register ................................................................................................................................ 29
  9.1 References ............................................................................................................................... 29

10 Quality Assurance ...................................................................................................................... 30

11 Acknowledgment ........................................................................................................................ 31

Figure 1-UML class diagram in a HiFi-Elements Model Management Project in SYNECT. .................. 11
Figure 2-Context menu “Create Implementation from File(s)...” ....................................................... 13
Figure 3-Component model format selection .................................................................................. 14
Figure 4-Workflows for Simulink model development .................................................................... 15
Figure 5-Simulink Frame example for a Driver component model. .................................................. 16
Figure 6-Simulink Test Frame example for a Driver component model. ............................................. 17
Figure 7-Context menu “Validate Simulink Interface...” ................................................................... 17
Figure 8-User dialog for Simulink model interface validation. ............................................................ 18
Figure 9-Extract from resulting report for Simulink model interface validation .......................................................... 18
Figure 10-Context menu "Export FMU". .......................................................... 19
Figure 11-User dialog for FMU Type. .......................................................... 20
Figure 12-Resulting FMU archive. .......................................................... 20
Figure 13-Context menu "Export FMU..." .................................................. 21
Figure 14-User dialog for FMU Type. .......................................................... 21
Figure 15-Resulting FMU archives. .......................................................... 22
Figure 16-Context menu "Validate FMI Model Interface..." .......................................................... 23
Figure 17-User dialog for FMI model interface validation .......................................................... 23
Figure 18-Resulting report for FMI model interface validation .......................................................... 24
Figure 19-Build XMODEL frame from Simulink stub with MATLAB R2016b and Morphee 7.0.13 .......................................................... 25
Figure 20-Context menu "Validate XMODEL Interface..." .......................................................... 26
Figure 21-User dialog for XMODEL interface validation .......................................................... 26
Figure 22-Resulting report for XMODEL interface validation .......................................................... 27

Tables
Table 1: D3.1 contribution to HIFI objectives .......................................................... 8
Table 2: Overview of terms used in this document .......................................................... 11
Table 3: Overview of SYNECT add-ons .......................................................... 12
Table 4-Supported import formats .......................................................... 13
1 Purpose of the Document

This document is the first deliverable of WP3. It explains the SYNECT add-ons to import all model formats with the purpose of interface checking and to generate frame models. Each add-on is described in terms of practical usage, features and limitations. The add-on software packet, the user documentation and the installation instructions are provided to the HiFi-Element partners via the central HIFI SVN repository. This document explains the basic capabilities of each add-on but does not replace the user documentation or installation instructions.

1.1 Document Structure

The chapters of the document have the following focus.

Chapter 2: Introduces the goals of the HIFI project and WP3. Clarifies the scope of this deliverable.
Chapter 3: Explains the motivation and connects to the guiding ideas and term used in D1.2 [1] / D1.3 [2].
Chapter 4: Explains the import workflow for component models.
Chapter 5: Describes the capabilities of the SIMULINK Add-On.
Chapter 6: Describes the capabilities of the FMI Add-On.
Chapter 7: Describes the capabilities of the XMODEL Add-On.
Chapter 8: Summarizes the findings.

1.2 Deviations from original Description in the Grant Agreement Annex 1 Part A

1.2.1 Description of work related to deliverable in GA Annex 1 – Part A

Within WP3, the component models (including parameter handling), the model integration process, as well as the test processes will be specified and implemented for offline as well as real-time simulation based on a central management system. Furthermore, the software framework to enable real-time co-simulation (including automated generation of system models) will be refined and extended by a test framework for automatic execution of functional and nonfunctional test cases on component and system level. Finally, the model-based calibration will be performed.

1.2.2 Time deviations from original planning in GA Annex 1 – Part A

There are no deviations with respect to timing of this deliverable.

1.2.3 Content deviations from original plan in GA Annex 1 – Part A

There are no deviations from the Annex 1 – Part A with respect to the content.
2 Introduction

2.1 Overall Target of the HIFI-ELEMENTS project

The HiFi-Elements project aims to reduce the design and validation effort for e-drivetrains by defining interface standards and workflows to combat insufficient model reuse and to ensure model interoperability and scalability. With the development of a structured toolchain it is possible to realize different software (virtual validation software, model data management software) to let them work in symbiosis by following a standardized automated path. Within the developed workflow, automatic parameter identification functions and automatic test case generation will also be implemented. The proposed workflow and interface standards will provide the possibility of earlier system validation and, as a result, it aims to:

- Increase the efficiency of the development process
- Decrease the development, testing and system integration effort to reduce time-to-market
- Increase safety and reliability of EVs

To test the improvement of the developed workflow and toolchain, four different use cases will be used to record the efforts expended for the different tasks in each of them. The recordings will be compared with the assessment of historical documented efforts for similar tasks utilizing the current practice.

2.2 Target of Work Package 3 (WP3)

Within WP3, the component models (including parameter handling), the model integration process, as well as the test processes will be specified and implemented for offline as well as real-time simulation based on a central management system. Furthermore, the software framework to enable real-time co-simulation (including automated generation of system models) will be refined and extended by a test framework for automatic execution of functional and nonfunctional test cases on component and system level.

This deliverable is part of task 3.1 “Management of simulation models across development phases”.

Contribution by partners:
DSPA will develop the new model format importers and frame generators. The configuration management and validation throughout the project will be integrated. Furthermore, all component models supplied by project partners will be imported into central model database (supporting task). Finally, DSPA will develop all agreed consistency and convention checker rules as automation workflows in the data management tool.

The following table lists the HIFI objectives which are supported by Task 3.1.

Table 1: D3.1 contribution to HIFI objectives.

<table>
<thead>
<tr>
<th>HIFI Objective</th>
<th>Description</th>
<th>Task 3.1 / Deliverable D3.1 Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1a</td>
<td>We will publish a recommendation for a standard functional model interface specification for a number of E-drive components.</td>
<td>The standard functional model interface specification is provided as a set of frame and shell model FMUs based on the FMI 2.0 standard [4]. In addition, XMODEL and SIMULINK frame models are supported.</td>
</tr>
<tr>
<td>O2a</td>
<td>Exchange models (within a component class) in a co-simulation tool chain (or physical parts in an XiL test), targeting a 75% reduction in effort.</td>
<td>Model Management in combination with Variant Management (see D3.2) will support a semi-automated model integration process allowing an easy exchange of models. Standardized model interfaces are a basis for this. Automatic validation of model interfaces against the architecture definition allows to detect interface deviations prior to model integration.</td>
</tr>
</tbody>
</table>
We will demonstrate 3-fold to 10-fold increase in test coverage for the same total testing effort using the SYNECT/Morphee workflow (allowing automatic execution of test cases) in combination with automatic test case generation, prioritization and selection, then compared to the current SotA workflow (which is the “manual”, ad-hoc generation and execution of test cases). Standardized model interfaces in combination with frame/shell models allow parallel development of models and tests.
3 Motivation and Overview

During the HiFi-Elements project, component developers create various models for a battery electric vehicle as work product of WP2. Those component models must comply to a common model interface defining the inputs, outputs and parameters. The WP3 toolchain can export interface compliant frame models to be used as starting models or model references by component developers.

Additionally, implemented component models are imported to a central database. The component models of this central database are used for vehicle system integration. The model import automatically verifies that a submitted model follows the specified model interface and gives a detailed report to the component developer. If the model interface is violated, the model submission is rejected.

The workflow for component development follows a top-down approach. The interfaces for all components deployed by the use cases are defined by the system architects in collaboration with the involved parties. These model interfaces specify the input and output signals required to interconnect the components. They are a work product of WP1 and are described in chapter 5 of D1.2 [1]. Additionally, the model interfaces include parameters that are shared by multiple components or requested by the use cases. For the project, five different vehicle architectures have been defined. The model interfaces specify which signals and parameters are mandatory in which system architecture. The task of the component developers is to implement model behavior utilizing the defined inputs, outputs and parameters.

During component development and system integration, it is likely that issues with the model interface specification will be discovered. For that reason, the model interface will be adapted in multiple iteration as the need arises. As a result, component developers must update the models to remain compliant with the new specification. This process is described in section 4 of D1.3 [2].

The toolchain supports the creation of frame models in the model formats MATLAB Simulink, FMU and XMODEL. The concept is explained in section 4.3 of D1.2 [1]. In D1.2, Simulink frame models are called Simulink stub models and FMU frame models are called template FMUs. These frame models can be used by component developers to have a direct view on the signals and parameters they must implement in their familiar modeling tool. They are valid models following the specified model interface, but without meaningful model behavior. Another useful approach uses test frames and shell models as described in section 4.3 of D1.2. A test frame is an environment in which the model is embedded for testing purposes. The most general frame model of a component exports a 150% interface. This 150% interface includes all signals and parameters of a component used in any architecture. This way, the frame model can be integrated into all architectures. A component model that does not support all system architectures can omit the superfluous signals and parameters in its implementation.

When importing a component model to the central database, the submitted model file is validated against the specified model interface. The validation can also be performed any time after the initial import, e.g. to check which changes must be made after a revision of the specifications. The component developer states the system architectures a component model is supposed to be compatible with. The validation only considers the selected architectures. If only one architecture is selected, the validation is performed against the 100% model interface that architecture.

HiFi-Elements uses SYNECT by dSPACE as model- and data management tool. SYNECT is the central user interface to access the specified model interfaces and to manage the developed model files. It is an integral part of many workflows, such as frame model export, model interface validation and component model submission. SYNECT is composed of different domains, one of which is SYNECT Model Management. In a Model Management Project, a Model is an abstract concept with Input Ports, Output Ports and Parameter Ports which are connected to Variables via Signal Interfaces or Parameter Interfaces. A Model can be associated with multiple Implementations representing concrete models. An Implementation typically stores the model file location in a content management system like
Figure 1 - UML class diagram in a HiFi-Elements Model Management Project in SYNECT.

SVN and additional meta data of the specific model. Figure 1 shows a simplified class diagram of such a Model Management Project as it is used in HiFi-Elements for component development. A SYNECT Model represents a component entity. The model interface is matched by the Ports, Interfaces and Variables where each signal or parameter with a unique name exists exactly once as a Variable connected with a single Interface. For each actual component model, an Implementation is created that is linked with the appropriate Model.

The design and maintenance of the model interface is performed in Enterprise Architect, a modeling and design tool. The specification is made available to SYNECT by a synchronization mechanism that creates and updates the model interface in the Model Management Project. The data is first exported to Excel spreadsheets from Enterprise Architect and then imported from there into SYNECT.

### 3.1 Terminology

Table 2 lists and explains special terms regularly used in this document.

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame, Mockup, Stub, Template, Wrapper</td>
<td>A model without meaningful behavior that is formally compliant with the model interface.</td>
</tr>
<tr>
<td>Shell</td>
<td>Two models, where one model provides the inputs of a component as outputs and the other model receives its outputs as inputs.</td>
</tr>
<tr>
<td>Test Frame</td>
<td>An environment model containing shell models connected with the component model.</td>
</tr>
<tr>
<td>100% Interface</td>
<td>The model interface used in a specific system architecture.</td>
</tr>
<tr>
<td>150% Interface</td>
<td>The combination of the model interfaces for all system architectures.</td>
</tr>
</tbody>
</table>

#### 3.1.1 SYNECT Add-On Concept

SYNECT has a built-in add-on mechanism to extend its functionalities. The most common ways to add new features are context menu extensions, ribbon buttons and import/export plug-ins. Context menu extensions add new commands to the context menus of specific SYNECT item types, for example Models or Implementations. To indicate those commands, the menu entries created for the HiFi-Elements project start with a prefix “HIFI:”. Ribbon buttons are added to the ribbon menu on top of the SYNECT Client window and are organized in named tabs. For the HiFi add-ons, the tab for new buttons is called “HIFI”. Both context menu commands and ribbon menu buttons trigger...
special features and workflows implemented in the Python programming language. Import and export plug-ins are used to support new data formats for the built-in import or export mechanism.

### 3.1.2 List of SYNECT Add-Ons

For the HiFi-Elements project, multiple SYNECT add-ons have been developed. Table 3 shows the add-ons relevant to model import and export. The HIFI Core Add-On has some basic features and must be installed for any other HIFI add-on to work correctly. The HIFI Demodata Add-On can be used to create a sandbox to get familiar with SYNECT and the project-specific data model and workspace layout. To synchronize the model file database stored on a central SVN server with the local working folder, the HIFI SVN Adapter offers upload and download commands. The three add-ons “HIFI SIMULINK Add-On”, “HIFI FMI Add-On” and “HIFI XMODEL Add-On” are discussed in more detail in chapters 5, 6 and 7. They include the main functionality to deal with frame models, model import and model interface validation of their respective model format.

**Table 3: Overview of SYNECT add-ons.**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIFI Core Add-On</td>
<td>Contains base features shared by multiple Add-Ons</td>
<td>2.5.5</td>
</tr>
<tr>
<td>HIFI Demodata Add-On</td>
<td>Creates demo data for training, testing and development in local SYNECT databases</td>
<td>2.5.5</td>
</tr>
<tr>
<td>HIFI SVN Adapter Add-On</td>
<td>Supports upload and download commands for SVN</td>
<td>2.5.5</td>
</tr>
<tr>
<td>HIFI SIMULINK Add-On</td>
<td>Supports import and export of MATLAB Simulink files</td>
<td>2.5.5</td>
</tr>
<tr>
<td>HIFI FMI Add-On</td>
<td>Supports import and export of FMU files [4]</td>
<td>2.5.5</td>
</tr>
<tr>
<td>HIFI XMODEL Add-On</td>
<td>Supports import and build of XMODEL files</td>
<td>2.5.5</td>
</tr>
</tbody>
</table>
4 Component Model Import

Table 4-Supported import formats.

<table>
<thead>
<tr>
<th>Model Format</th>
<th>Required Add-On</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATLAB Simulink S-Function (*,mexw64)</td>
<td>-</td>
</tr>
<tr>
<td>MATLAB Simulink Model (*.sli, *.mdl)</td>
<td>HIFI Simulink Add-On</td>
</tr>
<tr>
<td>FMU 2.0 (*,fmu)</td>
<td>HIFI FMI Add-On</td>
</tr>
<tr>
<td>XMODEL (*.xmodel)</td>
<td>HIFI XMODEL Add-On</td>
</tr>
</tbody>
</table>

To submit a component model to the central data base, the SYNECT Core Add-On offers a context-menu command “Create Implementation from File(s)” for Models. The initiation is shown in Figure 2 and Figure 3. This feature validates the submitted model against the model interface of its component entity and creates a new Implementation for it on success. Adding the model files to the central SVN repository is also part of the workflow. The available model formats for the import depend on the installed add-ons. The supported model formats are listed in Table 2. For Simulink S-Function MEXW64 files, the model interface can’t be verified due to the absence of interface meta data, and no validation report is generated. Otherwise, the report is the same as if requested explicitly with the features described in sections 5.3, 6.3 and 7.2.

Figure 2-Context menu “Create Implementation from File(s)...".
Figure 3-Component model format selection.
5 SIMULINK Add-On

The HIFI Simulink Add-On offers
- to create frame models in MATLAB Simulink format,
- to create test frame models in MATLAB Simulink format that enclose the Simulink component model,
- to add an existing Simulink model to the model database (see chapter 4),
- to create a new Implementation with a starting frame model,
- to validate the model interface of a Simulink model.

The add-on requires MATLAB with Simulink to run. Only 64-bit versions of MATLAB are supported. The test frame model creation requires MATLAB R2017b or newer because it makes use of FMU Import blocks that are not available in earlier versions. The officially supported MATLAB versions are further limited by the SYNECT release. On request, the MATLAB versions usable with SYNECT can be extended.

In the SYNECT HIFI add-ons, two workflows support the submission of Simulink model implementations. An overview of these workflows is given in Figure 4. Workflow A creates an Implementation in SYNECT and exports the Simulink frame model to use as basis for component development. It is started from a Model context menu entry in SYNECT labeled “New Simulink Implementation”. The finished model is later committed to the SVN model database. Workflow B starts with a Simulink model that has already been implemented as detailed in chapter 4. A supplementary workflow supports the model update after a new version of the model interface has been finished.

![Workflow Diagram](image)

**Figure 4-Workflows for Simulink model development.**

5.1 Export Simulink Frame Model

A Simulink frame model is a Simulink model that consists of an *Import* block for each specified input signal of the model interface and an *Output* block for each specified output signal. Additionally, if parameters are specified for the component, every parameter is created as a workspace variable and is placed inside a *Constant* block within a *Subsystem* called “*Parameters*”. The *Import* blocks are linked with *Terminator* blocks and the *Output* blocks with *Ground* blocks so that no port is without connection. The “Data type” and “Port dimension” block parameters are set to match the information stored in SYNECT Signal & Parameter Management. The “Unit” block parameter is not set explicitly but can easily be derived from the port name. An example of such a Simulink frame model is shown in Figure 5. If a SYNECT Implementation must only be compatible with a specific selection of system architectures, only signals and parameters required by those architectures are part of the frame model export for that Implementation.

The component developer can implement the model behavior directly into the frame model file by deleting the *Ground* and *Terminator* blocks and drawing lines between the *In-* and *Out-* port blocks and his or her own blocks. The *Constant* blocks representing parameters can be removed once their respective parameter is used elsewhere inside the model blocks. Once every parameter is in use, the “*Parameters*” subsystem block can be safely removed.
A Simulink frame model export is automatically performed when a component developer uses the "New Simulink Implementation" command of the HIFI Simulink Add-On. This command is triggered from a SYNECT Model context menu. It creates both a new SYNECT Implementation for a specific system architecture and generates the frame model. Alternatively, the frame model export can be executed for an already existing SYNECT Implementation by using the HIFI Simulink export plug-in which is part of the HIFI Simulink Add-On.

5.2 Export SIMULINK Test Frame

A Simulink test frame that can be exported with this feature is a Simulink model that combines FMU shell models (see section 6.2) with the Simulink component model. For the shell models, FMU import blocks are used and the component model is referenced using a ModelReference block. The ports of the ModelReference block are connected via lines with the shell models. Figure 6 shows an example of a Simulink test frame. For the export to work, the FMU shell models and the Simulink component model must already exist in the SYNECT Implementation’s working folder. Initially, a Simulink frame model can be used as component model.

For the test frame to work correctly, the ports of the component model must have the same order as the shell models. Basic interface compatibility like the signal dimension is ensured by Simulink, but the matching of port names is not enforced. The test frame export fails if the number of input ports of the component model does not equal the number of output ports of the left-hand side shell FMU. Analogously, the same applies to output ports of the component model and input ports of the shell FMU. The Simulink test frame export is executed as a regular SYNECT Implementation export plug-in.
5.3 Validate SIMULINK Model Interface

A Simulink validation report compares the signal and parameter interface of a model file with the model interface specification of the component it implements. It is a comprehensive list of differences between the expected and actual interface. It covers the names, dimensions and datatypes of signals and parameters.

Figure 7 and Figure 8 show how to validate the model interface of any Simulink model on the local filesystem. The command is run from the context menu of the Model containing the model interface to validate against. The system architectures to include in the validation can be selected individually. This way, the validation can be performed for a 100% interface, 150% interface or somewhere in between. A similar command can be executed from the Implementation of a submitted Simulink component model. This command automatically detects the system architectures the component model has been defined to be compatible with. A sample report is displayed in Figure 9.
Figure 8-User dialog for Simulink model interface validation.

Figure 9-Extract from resulting report for Simulink model interface validation.
6 FMI Add-On

The HIFI FMI Add-On supports workflows concerned with Functional Mock-up Units (FMUs) using the Functional Mock-up Interface (FMI) standard version 2.0 [4]. The add-on features are

- to export an FMU frame model,
- to export FMU shell models,
- to add an existing FMU to the model database (see chapter 4),
- to validate the model interface of an FMU.

The HIFI FMI Add-On generates FMU files without the need of any third-party FMI toolbox. However, the GNU Compiler Collection (GCC) is required to compile the binary C code of the exported FMUs. The system architecture of exported FMUs is always the 150% architecture.

6.1 Export FMU Frame Model

An FMU frame model implements the variables’ datatypes, units, dimensions and descriptions of a specified model interface. The FMU frame model can be executed by a simulator with FMI support, but its outputs will always remain constant. Since FMUs are essentially closed blackbox models, they are not designed to be modified by other modeling tools. Instead, they can be used as standardized model interface format and as placeholders, e.g. in system integration. If the modeling tool of a component developer can import FMUs, the developer can import the FMU frame model and compare its inputs, outputs and parameters with his own model implementation. Figure 10 and Figure 11 demonstrate the first steps to export FMU frame models from SYNECT. Figure 12 shows the content of the resulting FMU container. The FMU frame models created by the add-on contain both C source code and windows binaries (dll) for 64-bit systems.

Figure 10-Context menu “Export FMU”.
6.2 Export FMU Shell Models

FMU shell models can be used in a test frame. They always consist of a pair of two FMUs. The left-hand side FMU mirrors the inputs of the model interface as outputs and the right-hand side FMU mirrors the outputs as inputs. This way, a model that respects the model interface can be placed and connected in between the shell models. To be able to stimulate the component model within such a test frame, the left-hand side shell FMU has an input for every output whose value is simply passed-through. The right-hand side shell FMU has outputs, respectively. Figure 13 and
Figure 14 demonstrate the first steps to export a pair of FMU shell models from SYNECT. Figure 15 displays the content of the resulting FMU containers. Figure 6 shows a pair of shell models within a test frame.
6.3 Validate FMI Model Interface

An FMI validation report compares the signal and parameter interface of a model file with the model interface specification of the component it implements. It is a comprehensive list of differences between the expected and actual interface. It covers the names, dimensions, units and datatypes of signals and parameters.

Figure 16 and Figure 17 show how to validate the model interface of any FMU on the local filesystem. The command is run from the context menu of the Model containing the model interface to validate against. The system architectures to include in the validation can be selected individually. This way, the validation can be performed for a 100% interface, 150% interface or somewhere in between. A similar command can be executed from the Implementation of a submitted FMU component model. This command automatically detects the system architectures the component model has been defined to be compatible with. A sample report is displayed in Figure 18.
Figure 16-Context menu “Validate FMI Model Interface...”.

Figure 17-User dialog for FMI model interface validation.
Figure 18-Resulting report for FMI model interface validation.
7 XMODEL Add-On

With the HIFI XMODEL Add-On, support for the xMOD (see [3]) model format XMODEL is introduced. The add-on offers to validate the model interface of an XMODEL. The creation of XMODEL files is performed by the xMOD-Morphee Target by FEV. It is a build target for MATLAB Simulink and depends on the Simulink Coder. After the build, XMODEL models can be configured and simulated in xMOD.

7.1 Build XMODEL Frame Model

An XMODEL frame can be used as a placeholder in xMOD. As any XMODEL, it consists of three files: a binary file (dll), a model description (xmodel) and a parameter file (xml). An XMODEL frame model is created by building a Simulink frame model, as described in section 5.1, with the Morphee-xMOD Real-Time Target for Simulink. Figure 19 shows how to configure the code generation settings to build such an XMODEL frame. While the built process is entirely performed by MATLAB Simulink and the xMOD-Morphee Target, it can be automated by remote-controlling MATLAB via M-scripts and console commands.

![Figure 19-Build XMODEL frame from Simulink stub with MATLAB R2016b and Morphee 7.0.13.](image)

7.2 Validate XMODEL Interface

An XMODEL validation report compares the signal and parameter interface of a model file with the model interface specification of the component it implements. It is a comprehensive list of differences between the expected and actual interface. It covers the names, dimensions and datatypes of signals and parameters.

Figure 20 and Figure 21 show how to validate the model interface of any XMODEL on the local filesystem. The command is run from the context menu of the Model containing the model interface to validate against. The system
architectures to include in the validation can be selected individually. This way, the validation can be performed for a 100% interface, 150% interface or somewhere in between. A similar command can be executed from the Implementation of a submitted XMODEL component model. This command automatically detects the system architectures the component model has been defined to be compatible with. A sample report is displayed in Figure 22.

![Figure 20-Context menu “Validate XMODEL Interface...”](image)

![Figure 21-User dialog for XMODEL interface validation.](image)
Figure 22-Resulting report for XMODEL interface validation.
8 Summary

For the HiFi-Element project, component interfaces have been defined as formal description of component models. Due to the model interface definitions, multiple component implementations following the same interface can to some degree be used interchangeably in system integration models consisting of multiple components. SYNECT as data management and model management platform, in conjunction with SVN for file storage, is used to collect all component models and system models developed for the project.

Frame models are formally valid models without meaningful model behavior. They comply with the specified component interface and can be used as placeholder for an actual component implementation in a simulation. Additionally, they serve as reference for the model interface and can, in the case of MATLAB Simulink frame models, be used as basis for the model implementation.

Shell models serve as test frames. They always consist of a pair of two FMUs. The left-hand side FMU mirrors the inputs of the model interface as outputs and the right-hand side FMU mirrors the outputs as inputs. This way, a model that respects the model interface can be placed and connected in between the shell models. By this interface mismatches become visible in SIMULINK.

SYNECT has been extended by add-ons to support frame/shell model generation and component model import. The supported model formats are MATLAB Simulink, FMI 2.0 [4] and XMODEL for xMOD [3]. The component model import checks if a model is compliant to the defined model interface and integrates valid models into the database so that system integrators can use them.

By this features the objectives O1a, O2a and O4 are supported (see also section 2.2) since the formalization and automatic validation of model interfaces against the architecture definition allows to detect interface deviations prior to model integration and the standardized model interfaces in combination with frame/shell models allow parallel development of models and tests.
9 Risk Register

<table>
<thead>
<tr>
<th>Risk number</th>
<th>Description of Risk</th>
<th>Proposed Risk Mitigation Measure</th>
<th>Probability / effect(^1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(^1\) Level of probability / effect: 1 = high, 2 = medium, 3 = low

9.1 References

[1] Deppe, M. et al, *HIFI-ELEMENTS Deliverable D1.2; Document describing the SYNECT generated component model reports, the component interfaces and the system architecture*, 2018


10 Quality Assurance

The Steering Committee is the body for quality assurance. The procedure for review and approval of deliverables is described in the deliverable report D8.1 – “Project Handbook”. The quality will be ensured by checks and approvals of WP Leaders as part of the steering committee (see front pages of all deliverables).
11 Acknowledgment

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under Grant Agreement no. 769935.

Disclaimer
This document reflects the views of the author(s) and does not necessarily reflect the views or policy of the European Commission. Whilst efforts have been made to ensure the accuracy and completeness of this document, the HIFI-ELEMENTS consortium shall not be liable for any errors or omissions, however caused.