<table>
<thead>
<tr>
<th>Deliverable No.</th>
<th>HIFI-ELEMENTS D1.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deliverable Title</td>
<td>Document describing the workflow architecture</td>
</tr>
<tr>
<td>Deliverable Date</td>
<td>2018-05-30</td>
</tr>
<tr>
<td>Deliverable Type</td>
<td>REPORT</td>
</tr>
<tr>
<td>Dissemination level</td>
<td>Public (PU)</td>
</tr>
<tr>
<td>Written By</td>
<td>Markus Deppe, dSPACE</td>
</tr>
<tr>
<td>Checked by</td>
<td>Jens Ewald, FEV (coordinator)</td>
</tr>
<tr>
<td>Approved by</td>
<td>Steering Committee</td>
</tr>
<tr>
<td>Status</td>
<td>Final</td>
</tr>
<tr>
<td>Status</td>
<td>Final</td>
</tr>
<tr>
<td>Dissemination level</td>
<td>Public (PU)</td>
</tr>
<tr>
<td>Written By</td>
<td>Markus Deppe, dSPACE</td>
</tr>
<tr>
<td>Checked by</td>
<td>Jens Ewald, FEV (coordinator)</td>
</tr>
<tr>
<td>Approved by</td>
<td>Steering Committee</td>
</tr>
<tr>
<td>Status</td>
<td>Final</td>
</tr>
</tbody>
</table>
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 – MGEP – Mondragon Goi Eskola Politeknikoa 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>Granrath, Christian</td>
<td>RWTH Aachen University</td>
</tr>
<tr>
<td>Aitor, Arrieta</td>
<td>Mondragon University</td>
</tr>
<tr>
<td>Carl, Hettig</td>
<td>FEV</td>
</tr>
<tr>
<td>Hassen, Hadj-Amor</td>
<td>FEV SA</td>
</tr>
<tr>
<td>Rene, Scheer</td>
<td>RWTH Aachen University</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>2018-03-04</td>
<td>Initial draft</td>
</tr>
<tr>
<td>V0.2</td>
<td>2018-05-07</td>
<td>Refinement of document</td>
</tr>
<tr>
<td>V1.0</td>
<td>2018-05-29</td>
<td>Review comments considered; 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>V0.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V0.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V0.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V1.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Verification and approval

<table>
<thead>
<tr>
<th>Name</th>
<th>Organisation</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Granrath, Christian</td>
<td>RWTH Aachen University</td>
<td>2018-05-17</td>
</tr>
</tbody>
</table>
Executive summary

This deliverable describes documents the workflow and the architecture of the planned toolchain (Agile Model Development Framework).

Deliverable review process

The WP leaders are responsible for a first review of the deliverables. The WP leader must ensure that the deliverable has been completed according to the agreed work plan within the respective WP. WP leaders are responsible for the delivery of all WP related deliverables to the Steering Committee at least 20 working days ahead of the EC deadline. The Steering Committee must ensure that the result fits into the overall workplan. Members of the Steering Committee can provide feedback in writing to the author. The approval by SC members can be done tacitly – an email is not necessary. The SC will be kept up-to-date during the monthly meetings on the deliverables status.

The Project Coordinator FEV has 10 days to review deliverables and request necessary amendments to the Work Package leader and the author of the deliverable.

The deliverable is submitted to the EC by Uniresearch once the final document has been checked by Uniresearch on form, not on content.
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 HI-FI ELEMENTS project .............................................................. 8
  2.2 Target of Work Package 1 (WP1) ..................................................................................... 8
3 AMDF Architecture ................................................................................................................ 9
  3.1 Standards ........................................................................................................................... 9
    3.1.1 ASAM XIL API 2.0 ....................................................................................................... 9
    3.1.2 ASAM MDF 4.0 ......................................................................................................... 9
    3.1.3 ASAM CDF 2.0 ....................................................................................................... 10
    3.1.4 FMI 2.0 .................................................................................................................. 10
  3.2 Tools .................................................................................................................................. 10
    3.2.1 Enterprise Architect ................................................................................................ 10
    3.2.2 SYNECT ................................................................................................................ 10
    3.2.3 xMOD ................................................................................................................... 11
    3.2.4 TEVET: Tool for Electrical Vehicle Testing .............................................................. 11
    3.2.5 Calibration Tooling .................................................................................................. 11
  3.3 Toolchain ............................................................................................................................ 11
    3.3.1 Overview .................................................................................................................. 11
    3.3.2 Systems Engineering................................................................................................. 13
    3.3.3 Model/System Integration ....................................................................................... 14
    3.3.4 Simulation, Calibration and Testing ........................................................................ 15
    3.3.5 Testing Framework .................................................................................................. 16
4 AMDF Workflow ..................................................................................................................... 18
  4.1 Continuous Integration ...................................................................................................... 18
  4.2 Roles .................................................................................................................................. 19
  4.3 Workflow Step I: Architecture and Requirements Definition ........................................ 20
    4.3.1 Formal Requirements ............................................................................................... 20
  4.4 Workflow Step II: Component Model Development .......................................................... 21
    4.4.1 Parameter Optimization Workflow ........................................................................... 21
  4.5 Workflow Step III: Component Model Testing ................................................................. 23
    4.5.1 Automatically generated Test cases ......................................................................... 24
    4.5.2 Manually created Test cases .................................................................................... 24
4.6 Workflow Step IV: System Integration ........................................................................... 24
  4.6.1 System Integration Details ....................................................................................... 25
  4.6.2 Co-Simulation with xMOD ....................................................................................... 26
  4.6.3 Integration in SIMULINK ......................................................................................... 28
4.7 Workflow Step V: System Testing ................................................................................. 28

5 Risk Register ....................................................................................................................... 30
  5.1 References ..................................................................................................................... 30

6 Quality Assurance ............................................................................................................. 32

7 Acknowledgment ................................................................................................................. 33

Appendix ................................................................................................................................ 34

Figure 1-The overall AMDF architecture. ............................................................................. 13
Figure 2-System Layers and Tools ....................................................................................... 14
Figure 3-xMOD Target .......................................................................................................... 15
Figure 4-Model Access with XIL API MA Port .................................................................... 15
Figure 5-An STI compliant test specification. ..................................................................... 16
Figure 6-Separation of execute (server) and evaluation (client) parts for Testing ................. 17
Figure 7-Overview of the AMDF workflow ....................................................................... 18
Figure 8-Continuous integration workflow ......................................................................... 19
Figure 9-Overview of the roles in the AMDF workflow ....................................................... 20
Figure 10-Architecture and Requirements Definition ............................................................ 21
Figure 11-Component Model Development Workflow .......................................................... 22
Figure 12-Parameter Optimization Procedure ..................................................................... 23
Figure 13-Parameter Optimization with SYNECT and XIL API MA Port ............................ 23
Figure 14-Component Model Testing Workflow .................................................................. 24
Figure 15-System Integration Workflow ............................................................................. 25
Figure 16-The System Model ................................................................................................ 25
Figure 17-Integration Steps ................................................................................................... 26
Figure 18-Integration Steps ................................................................................................... 26
Figure 19-Component Model Development Workflow .......................................................... 27
Figure 20-MIPS example ..................................................................................................... 27
Figure 21-Example of a Co-simulation with dedicated dashboards in xMOD ....................... 28
Figure 22-System Testing Workflow .................................................................................... 29

Tables
Table 1-Elements of the AMDF tool chain. ......................................................................... 11
1 Purpose of the Document

This document is the third deliverable of WP 3. It documents the workflow and the architecture of the AMDF (Agile Model Development Framework) toolchain to be developed in WP3.

1.1 Document Structure

The chapters of the document focus on:

- Chapter 2: Introduction to the targets of WP1
- Chapter 3: Description of the AMDF (Agile Model Development Framework) SW architecture
- Chapter 4: Overview of the workflow when using AMDF

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

WP1 defines the system architecture and requirements to achieve interoperability between the modules and the tool chain. It defines the module interfaces and structure; component boundaries and interfaces to ensure their exchangeability and their upgradeability. Additionally, the multi-modelling and scalability definition and approach will be specified, including maintainability and variant handling.

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 HI-FI ELEMENTS project

The HiFi Elements project aims to reduce the design and validation effort for e-drivetrain through the definition of interface standards and workflow to combat insufficient model reuse and to ensure model interoperability and scalability. With the developing of a structured toolchain it is possible to interface 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 it will be implemented also automatic parameter identification functions and automatic test case generation. The proposed workflow and interface standards will provide the ability for earlier system validation and, as a result, it aims to:

- Increase the efficiency of the development process
- Reduce the development and testing 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 it and will be compared with the assessment of historical documented efforts for similar tasks when using the current practice.

2.2 Target of Work Package 1 (WP1)

The objective of WP1 is to define the requirements and standardization criteria that will ensure the seamless information exchange between modules to be developed in WP2, as well as the interfaces necessary for the communication between the modules and the toolchain development WP3 respectively. Specifically, the WP objective are:

- Enable exchangeability, interoperability and replaceability of models in real time co-simulation by specification of system architecture, model interfaces and co-simulation environment.
- Allow real time co-simulation for all models in different stages of development by specification of system architecture, model interfaces, SW tools & HW requirements.

The initial deliverable D1.1 covers the assessment of the safety requirements and the modeling guidelines. The second deliverable (D1.2) will focus on the standardized interfaces of the component models and the system architecture. This third deliverable (D1.3) documents the workflow and the architecture of the AMDF (Agile Model Development Framework) toolchain to be developed by WP3.
3 AMDF Architecture

The following sections describe the tools, the used standards and the architecture of the Agile Model Development Framework (AMDF).

3.1 Standards

3.1.1 ASAM XIL API 2.0

“ASAM XIL is an API standard for the communication between test automation tools and test benches. The standard supports test benches at all stages of the development and testing process – most prominently Model-in-the-Loop (MIL), Software-in-the-Loop (SIL) and Hardware-in-the-Loop (HIL). The notation "XIL" indicates that the standard can be used for all "in-the-loop" systems. This has the advantage that it enables users to freely choose testing products according to their requirements and integrate them with little effort. Using ASAM XIL-compliant products allows users of test systems to mix and match the best components from different suppliers without having costly integration efforts.

The major goal of the ASAM XIL standardization effort is to allow for more reuse in test cases and to decouple test automation software from test hardware. Therefore, the reuse of test cases within the same test automation software on different test hardware systems is achieved. This leads to a significant reduction of efforts for test hardware integration into test automation software.

Some areas of the XIL standard are not HIL-specific. The simulation model access port, for example, can also be used to adapt simulation tools. This allows engineers to develop test cases in very early stages and in different domains to reuse them in later stages at a real HIL simulator using XIL.

The API provides vendor independent access to the functionalities of a XIL simulator via port interface definitions for the different kinds of ports. Each tool vendor can provide an implementation of these interfaces, which is specific for his tool set. Thus, the user of the XIL Testbench API gains standardized access to the tools of different vendors.”

(taken from ASAM XIL Website [3])

The ASAM XIL Testbench API covers the functional areas:

- Model access (MA port)
- ECU access (ECUC and ECUM port)
- Network access (Network Port),
- Diagnostics access (Diag Port)
- Electrical error simulation (EES port)

For HIFI the Model Access port will be used for calibration and testing.

3.1.2 ASAM MDF 4.0

“MDF (Measurement Data Format) is a binary file format to store recorded or calculated data for post-measurement processing, off-line evaluation or long-term storage. The format has become a de-facto standard for measurement & calibration systems (MC-systems), but is also used in many other application areas. […]

As a compact binary format, ASAM MDF offers efficient and high-performance storage of huge amounts of measurement data. MDF is organized in loosely coupled binary blocks for flexible and high-performance writing and reading.

MDF was originally developed as a proprietary file format in the 90’s by Vector Informatik GmbH and Robert Bosch GmbH. The first public version was MDF 2.0 in 1991; MDF 3.0 was released in 2002. Over the years only gradual extensions were added. The current non-ASAM version MDF 3.3 is still fully backward compatible to all MDF 2.x and 3.x versions.

However, the file size of the proprietary MDF 3.x format was limited to 4 GB due to the usage of a 32-bit data type for file internal links. The rapid growth of measured data and thus file sizes was not foreseeable in 1991: After the
turn of the century, this caused more and more requests for an update of MDF to support larger file sizes and to meet nowadays requirements. In addition, major OEMs expressed their desire for transferring MDF to an official industry standard.

As a result, a newly founded ASAM working group started in 2008 with the revision and standardization of MDF. This led to the release of ASAM MDF 4.0 in 2009. ASAM MDF 4.0 overcomes the size restrictions of the previous MDF 3.x version and offers a range of new features like flexible extensibility via XML, custom signal grouping, events or attachments. However, due to fundamental changes like a 64-bit data type for links, MDF 4.x is not compatible to MDF 3.x any longer.” (taken from ASAM MDF Website [1])

### 3.1.3 ASAM CDF 2.0

“ASAM CDF defines the data description format for storing the ECU parameter values and the associated meta data. [...] ASAM CDF provides a file format that defines and stores this information in a machine-readable format. Specifically, ASAM CDF allows the storing of scalar values, strings, arrays, curves, maps, structures and arrays of supported data types. [...] The file format is typically used in measurement & calibration tools (MC-tools) and by calibration data management tools (CDM tools). The format allows easy import and export, comparison and merge of data from various sources. The file extension of ASAM CDF compliant description files is either "cdfx" or "xml". The internal format of cdfx-files is based upon XML notation. The standard includes a schema definition file (.xsd) and a document type definition file (.dtd) for formal file validation.” (taken from ASAM CDF Website [2])

### 3.1.4 FMI 2.0

“Functional Mock-up Interface (FMI) is a tool independent standard to support both model exchange and co-simulation of dynamic models using a combination of xml-files and compiled C-code.

The first version, FMI 1.0, was published in 2010, followed by FMI 2.0 in July 2014. The FMI development was initiated by Daimler AG with the goal to improve the exchange of simulation models between suppliers and OEMs. As of today, development of the standard continues through the participation of 16 companies and research institutes under the roof of the Modelica Association as a Modelica Association Project. FMI is supported by over 108 tools and is used by automotive and non-automotive organizations throughout Europe, Asia and North America.” (taken from FMI Website [9])

A simulation model following the FMI standard is called a Functional Mock-up Unit (FMU).

### 3.2 Tools

#### 3.2.1 Enterprise Architect

Enterprise Architect [12] is a visual platform for designing and constructing software systems, for business process modeling, and for more generalized modeling purposes. It is based on UML (www.omg.org) and supports all 14 diagram types of UML 2. In HIFI the architecture definition is based on model-based systems engineering methods. High quality SysML models are used to define the interfaces, use cases for each component model or the system architecture by means of visual class-, activity-, block- or context-diagrams. For details and examples please refer to [4].

#### 3.2.2 SYNECT

“SYNECT is a data management and collaboration software tool with a special focus on model based development and ECU testing. The software is designed to help you manage data throughout the entire development process. This data can include models, signals, parameters, tests, test results, and more. SYNECT also handles data dependencies, versions and variants, as well as links to the underlying requirements. One key aspect of SYNECT is direct connection to engineering tools, e.g., MATLAB®, Simulink®, TargetLink®, or AutomationDesk, and application/product lifecycle management systems (ALM/PLM) so that you can work with your preferred tools and seamlessly exchange data.” (taken from dSPACE SYNECT Product Information [5])
SYNECT is highly customizable and for the HIFI project so called Add-Ons are developed to support the entire AMDF workflow.

### 3.2.3 xMOD

“xMOD is a co-simulation and a virtual experimentation laboratory platform. The over-arching concept for xMOD is that it is intended to connect heterogeneous models (e.g. Simulink®, GT-SUITE®, AMESim® ...etc.) through a versatile and efficient coupling process. This is achieved by optimizing model execution through multi-cored and multi-solver processing with multi-rate, and by providing a standalone model execution capability. However, xMOD also provides some additional benefits, including compatibility with emerging standards such as Functional Mock-up Interface (FMI), as well as continuity in the transition from Software-in-Loop (SiL) to Hardware-in-Loop (HiL). Finally, xMOD allows customization of graphical user interfaces which can be essential when facilitating ‘non-expert’ end user access.” (taken from xMOD Product Information: [6])

### 3.2.4 TEVET: Tool for Electrical Vehicle Testing

The university of Mondragon will develop a tool named TEVET (Tool for Electrical Vehicle Testing), which will include an editor for formal requirements specification, a test specification generator and a test execution tool for signal-based testing. These test cases will be automatically generated by considering functional requirements. To specify requirements, a Domain Specific Language (DSL) will be developed, which will contain all the necessary features to automatically generate signal-based test cases. A DSL is a language specifically dedicated to solve a particular problem from a specific domain. The test generator will parse each requirement and will automatically generate from 1 to N test cases complying the ASAM-XiL standard. The test cases will be concretized in a *.sti file. The test execution involves a source that will read the automatically generated *.sti file and will convert these test cases into stimulation signals to test the system.

### 3.2.5 Calibration Tooling

The RWTH Aachen University will develop a tool to calibrate various model structures. The main aim of the tool is to optimize the accuracy of models by using a parameter variation. It will handle with existing model structures and will compare measurement values delivered by a device under test with those out of the calibration process. The tool will not support the selection of suitable model structures. The model structures can be divided in three types. It is customary to use a color code in shades of grey, according to what type of prior knowledge exists. White-box models are perfectly known and grey-box models contain unknown parts. The Black-box models does not use any particular prior knowledge of the character or physics of the relationships involved. The tool will especially deal with black-box models for dynamical systems. The tool will fit black box models to measured input-output data by an optimization problem. After several iterative executions the optimized model parameters will saved in the ASAM CDF standard.

### 3.3 Toolchain

#### 3.3.1 Overview

This section provides a tabular overview of the basic elements used in the AMDF toolchain followed by a graphical overview. The fundamental design decision was that the xMOD simulator will act as a XIL API MA server [3]. This allows all kind of XIL API MA client applications to interact with xMOD in the same standardized way. In addition, the use of standardized file formats for parameter values, simulation results etc. is a design principle.

<table>
<thead>
<tr>
<th>Table 1-Elements of the AMDF tool chain.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td>Enterprise Architect</td>
</tr>
<tr>
<td>SYNECT Server</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>SYNECT RM</td>
</tr>
<tr>
<td>SYNECT TM</td>
</tr>
<tr>
<td>SYNECT SPM</td>
</tr>
<tr>
<td>SYNECT MM</td>
</tr>
<tr>
<td>SYNECT WFM</td>
</tr>
<tr>
<td>SYNECT VM</td>
</tr>
<tr>
<td>Calibration</td>
</tr>
<tr>
<td>Test Spec Generator</td>
</tr>
<tr>
<td>Testing</td>
</tr>
<tr>
<td>xMOD Editor</td>
</tr>
<tr>
<td>xMOD Simulator</td>
</tr>
<tr>
<td>SVN</td>
</tr>
</tbody>
</table>

The graphical overview given in Figure 1 shows the dependencies among the different tools and the most important file formats for calibration, testing and model integration/simulation.
3.3.2 Systems Engineering

For the project, it has been decided to start modeling at system and function layer by using SysML in Enterprise Architect. At architecture layer, SYNECT is used to describe the system architecture. For the synchronization of SysML with the architecture layer in SYNECT it is possible to export an Excel (XLSX) file containing the component interface specifications from Enterprise Architect and to import this file in SYNECT.

To ease the transition from the architecture layer to the implementation layer, it has been decided to use “Component Model Reports” in the form of FMUs in format version 2.0. The idea is to generate template FMUs out of SYNECT which formally reflect the component interfaces but do not have an internal behavior. These FMUs can be imported in various modeling tools on the market. For details please refer to Deliverable 1.2 [4].
3.3.3 Model/System Integration

xMOD consists of two complementary parts:

The Editor with three parts:
- "Configuration" to configure Calculation Group and CAN boards
- "Libraries" which contains the models to be integrated in xMOD
- "Simulation" which enables to build the simulation environment (models connection, GUI customization, automation, choose of the solvers and their step sizes...etc.)

The Runtime which executes the "Simulation" in a separate process than the editor and without needing a complementary license.

xMOD offers the possibility to connect graphically (manually or automatically) models coming from different tools. This graphical description is based on block diagram that represent different parts of a system and signal lines to define the relationship between the blocks. xMOD is able to export and import a global co-simulation using a ZXMOD file which is a specific container for several elements necessary for xMOD runtime.

xMOD provides 3 different methods to import a model:

1. Standalone co-simulation
   Standalone co-simulation is the integration of models in binary format (.DLL, .RTDLL). The process of models generation from the original modeling tool is based on a specific toolchain. xMOD provides a specific target based on SimulinkCoder (formerly known as Real-Time Workshop) to generate binary models for xMOD. A model generated by xMOD is composed by 3 elements: a binary file (.DLL or .RTDLL), a .XMODEL file for model's description and a third XML file for parameters description. The integration and execution of these models is completely standalone and does not require a third party license for its execution.
In this context, xMOD is acting as a co-simulation bus to synchronize data exchange between the simulation tools. Indeed, the models execution is ensured by the original simulation tool. xMOD is also synchronizing the clocks of all the simulation tools. The time of the global simulation is handled by xMOD kernel.

3. FMI integration

xMOD is compatible with the FMI standard. It can import models from both types of specifications (ModelExchange and Co-Simulation). FMI 1.0 and 2.0 specifications are supported.

A co-simulation in xMOD can use the 3 types of models integration in the same time while the xMOD Kernel will synchronize all these methods.

In the context of the HIFI-ELEMENTS project, an XML file describing interconnected components will be developed. This file will have the extension .xmips. The XMIPS file has to be generated by SYNECT or a third party tool to define how xMOD models are interconnected.

3.3.4 Simulation, Calibration and Testing

The XIL API MA standard covers several methods to access the simulation model such as controlling a simulation, reading/writing of variable values, capturing of variable traces and the stimulation of model variables. It defines the AMDF core architecture for simulation, calibration and testing.
3.3.5 Testing Framework

In the context of the HIFI-ELEMENTS project, test cases are a set of signals that stimulate a system/component and a set of reference signals for the expected outputs of these stimulation signals. Both, the stimulation data, as well as the expected outputs (i.e., reference data), are specified in files with *.sti extension, which are xml-like files.

**Test Specification (Example)**

- **Variables**

<table>
<thead>
<tr>
<th>Name</th>
<th>Unit</th>
<th>Tolerance</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POS_BRK</td>
<td>%</td>
<td>± 3</td>
<td>Brake pedal position</td>
</tr>
<tr>
<td>POS_TTL</td>
<td>%</td>
<td>± 3</td>
<td>Throttle control pedal position</td>
</tr>
<tr>
<td>VEL_MEAS</td>
<td>m/s</td>
<td>± 1 m/s</td>
<td>Measured vehicle velocity</td>
</tr>
</tbody>
</table>

- **Conditions/Events**

<table>
<thead>
<tr>
<th>Name</th>
<th>Predecessor</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ManeuverStart</td>
<td></td>
<td>abs(VEL_MEAS) != 0</td>
</tr>
<tr>
<td>GainedSpeed</td>
<td>ManeuverStart</td>
<td>VEL_MEAS &gt; 1.4</td>
</tr>
<tr>
<td>ReachedTarget</td>
<td>GainedSpeed</td>
<td>posedge(VEL_MEAS, 13.9)</td>
</tr>
</tbody>
</table>

- **Segments**

<table>
<thead>
<tr>
<th>Name</th>
<th>StartEvent</th>
<th>EndEvent</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Driveaway</td>
<td>ManeuverStart</td>
<td>GainedSpeed</td>
<td>POS_BRK == 0</td>
</tr>
<tr>
<td>Acceleration</td>
<td>GainedSpeed</td>
<td>ReachedTarget</td>
<td>POS_BRK == 0 &amp;&amp; POS_TTL &lt; 70</td>
</tr>
</tbody>
</table>

**Figure 5-An STI compliant test specification.**

These test cases are automatically generated by using a test specification generator. The generated test cases are compliant with system requirements, which are specified by using a DSL. This DSL contains all the necessary features to generate ASAM-XiL compliant test cases, following the format required in the *.sti files. These *.sti files are later parsed by the testing tool to execute test cases.

The execution of a test is split into two parts: The “execution” and the “evaluation”:

- **Execution (EXEC):** A simulation run including stimulation and capturing of data performed by the XIL API MA server. The stimulation definition contains condition watchers defined by means of the ASAM GES (General Expression Syntax) allowing to dynamically react on variables of the system under test (SUT, represented as FMU or XMODEL).
- **Evaluation (EVAL):** Verdict calculation done after the execution based on the comparison of the captured data (MDF file) and the reference values (STI file) done by the “testing tool” (XIL API MA client).
Figure 6-Separation of execute (server) and evaluation (client) parts for Testing.
4 AMDF Workflow

The AMDF workflow consists of five major elements (Figure 7) which are explained in detail in the following sections. For the definition of the underlying planned SW architecture please refer to section 3 of this document.

The AMDF needs to fulfill these high-level requirements for the workflows:
- Support a Top-Down Approach using Systems Engineering methodologies
- Provide IP protection for models
- Allow efficient testing by deploying formalized requirements for automatic test case generation
- Use highly automated build and test processes (continuous integration)
- Consider a component oriented exchange and integration of models from different sources/tools
- Keep track of all data by a data management-oriented approach

Figure 7-Overview of the AMDF workflow

4.1 Continuous Integration

Continuous Integration (CI) is the concept to frequently execute the integration process to detect integration issues as early as possible in the development process. The Integration process can be triggered once a day (e.g., nightly build approach) or each time a new content is added to the project repository. CI gives the developers direct feedback how their changes on components affect the overall system. Test routines on unit and system level can also be added to the integration workflow.

In the HIFI project CI functionalities will be implemented to ease the development work of the model developers. All the steps of the system integration process will be identified and be aggregated to action sets. The actions sets will be implemented into SYNECT workflow management and can be arranged for different workflow applications flexibly.

To test the successful integration of the component models, the models have to be exported from the data management tool SYNECT to xMOD, which serves as the CO-Simulation platform. All associated metadata will be created for xMOD on the fly, which describes the interconnection between the components (*.xmips). Once all data is provided to xMOD, the seamless integration of the models and the successful initialization of the Co-Simulation will be triggered and checked. The result of the integration test will be logged and reported back to SYNECT.
To trigger the integration test manually SYNECT will be used as main user interface. Functions will be developed to schedule the integration test or to link the test with the SVN Commit process. In addition, concepts to automatically execute the test cases stored in SYNECT during the described integration test shall be investigated.

Figure 8-Continuous integration workflow

4.2 Roles

During the different workflow phases there are different users acting in different roles. Figure 9 shows the roles we defined including the main assignment to the workflow phases (phases not shown in order).

The Stakeholder is a kind of customer, who wants to get a well-tested system simulation for his specific use case. Together with the Architect/Integrator and other experts he specifies his requirements. The Architect/Integrator acts both in the initial Requirements & Architecture Definition phase and in the System Integration phase. He is responsible for the definition and the change management of the component interfaces and the system architecture. He has the knowledge about the vehicle topologies and variants.

The component developer takes functional requirements and an interface definition from the Architect and develops a white-box model. Additionally, the white-box model is to be transferred to a black-box model (FMU or XMODEL) to ensure IP protection and simulation platform independency. This black-box model is to be tested by the component tester against the functional requirements (formal or non-formal). The tests can be interactive or automated and have to be developed by the Component Tester.

During system integration, the Architect/Integrator selects a vehicle topology/variant to be integrated and collects the required black-box component models from the database (SYNECT+SVN). Then, he interconnects the component models, configures and runs a Co-Simulation as integration test.

Finally, the System Tester takes the integrated and pretested Co-Simulation configuration and runs the tests he has developed for this specific system (vehicle topology).
The architecture and requirement definition is based on model-based systems engineering methods. High quality SysML models as well as requirements will reduce communication efforts, inconsistencies between requirements and reduce the amount of defects, which would otherwise only be detected on later verification phases. An improved communication is especially important in the context of large project teams, interdisciplinary exchange and collaborations of many partners in a big consortium.

The System Modeling Language (SysML) was introduced in the 1990s and is derived from the Unified Modelling Language (UML). It provides a large set of structural and behavioral diagram languages to specify systems from different viewpoints on different abstraction levels, but does not provide a concrete process or detailed guidelines which diagram types are to be used in which order or for which level of abstractions [11].

In HIFI-ELEMENTS a systematic step-by-step model-based requirement and function specification due to four abstraction layers will be used for architecture and requirement specification of simulation models for electric vehicles (see Figure 10). A systematic vertical refinement approach from layer to layer is combined with a horizontal hierarchical decomposition from the context of the system model to specific modules (see Figure 1 in Deliverable D1.1). From layer to layer additional requirements are identified and derived from higher level requirements. In consequence, on each level the provided requirements and concepts are reevaluated and detailed, which - in terms of frontloading - ensures a very early verification and validation on the ongoing activities. These steps are quite time-intensive but provide also high quality SysML artifacts as e.g. use case diagrams or activity diagrams, which can be used to significantly speed up the overall development process. The mentioned abstraction layers are applied for the structured documentation whereas the applied process is meant to be agile. The structured decomposition of the system models is intended to frontload and reduce the effort of model integration in later project phases by system based development [11].

The approach for model-based architecture and requirement specification involves the stakeholders of the models, function and software developers (component developers) as well as system architects.

Figure 9-Overview of the roles in the AMDF workflow
4.3.1 Formal Requirements

The requirements are formalized by using a developed DSL. This is necessary to avoid the ambiguity related to requirements specified in natural language, to better trace requirements with model inputs/outputs and to be easily readable by computers. The typical workflow for specifying requirements would be as follows:

1. Specify requirements in natural language with SYNECT
2. Specify model interface involved in the testing process in the DSL
3. “Translate” the requirements specified in natural language into formal requirements by using the developed DSL, linking them to SYNECT
4. Generate test cases for each specified requirement in the DSL

4.4 Workflow Step II: Component Model Development

As described in Section 2.2 one target of work package 1 is to define the interface standard for the electric powertrain system, to ensure a seamless integration of the single components. To ease the start of modelling for the component developers, shell FMUs will be provided and stored in SYNECT. The concept of shell FMUs are explained in D1.2/4.3.2.

The component developers can import the FMUs into their modeling tool and start implementing the internal behavior of the component in a white-box model (*.mdl/*.slx, *.ame, *.mo, *.gtm, ...) using the provided inputs and outputs. Simulink 2017b as recommended modelling tool comes already with a FMU importer. Once the internal functionality for a dedicated variant is implemented the component developer can compile and export the component as a black-box model (.fmu, .xmodel) and store it into SYNECT. The model will then be tested. In an iterative process the test report is handed back to the component developer, who can fix occurring issues.
Figure 11 - Component Model Development Workflow

4.4.1 Parameter Optimization Workflow

The optimization of model parameters is to fit a model within a given structure by using a parameter estimation. The main topic of the parameter optimization workflow is to model the input-output dynamics of a system only by its input-output data, also called black-box modeling. Figure 12 shows the parameter optimization procedure in HiFi-Elements. The procedure starts with the initialized calibration parameters and the reference data as measurement values delivered by a device under test. Both data sets are provided by dSPACE SYNECT. In the first iteration the defined model is executed with initial parameters and the reference input by xMOD simulator. In the next step the model error is calculated by the comparison of the calculated outcome and the reference output. To calculate the model error the least mean square method can be used for instance. If the model error achieves the predefined residuum, the recursive loop is interrupted and the current parameters will be saved in SYNECT. Besides the model error estimation itself is the identification of the relation between the model error and the parameter variation a planning target. If the model error is higher than defined, the modified parameters will be executed by xMOD, again.
Figure 12-Parameter Optimization Procedure.

The data exchange of parameter values are moved by ASAM CDF files. To export these files from SYNECT and to import the optimized parameters back in SYNECT, dSPACE provides the CDF Helper tool runnable by Python and MATLAB Mathworks. Figure 13 shows the integration of the toolchain in the parameter optimization procedure. Besides the conversion to couple xMOD, SYNECT and MATLAB the XIL API MA is used. The selected model and parameters are transmitted and executed from MATLAB to xMOD. The calculated outcome for evaluating the parameter accuracy is also transmitted via XIL API MA.

Figure 13-Parameter Optimization with SYNECT and XIL API MA Port.

4.5 Workflow Step III: Component Model Testing

To generate test cases for a specific component, its requirements need to be selected once these have been defined in the DSL. From these requirements, test cases are generated by automatically parsing the information of the DSL.
To generate test cases, the test engineer will choose which requirements need to be tested, and what kind of test cases are to be generated (e.g., robustness test cases, long test cases, etc.). It is noteworthy that test cases can also be manually generated. This allows for generating test scenarios that have not been considered by the automatically generated test cases, or because the test engineer needs to test a specific scenario. Once all the test cases are generated, these are automatically run by using the xMOD simulation. Test cases include both, stimulation signals to the inputs of the models, and expected outputs. Test cases are executed one by one and the test tool compares whether the executed inputs are in accordance with the expected outputs. This permits obtaining the test verdict (i.e., whether a test case has passed or failed), and thus, reporting the test execution result. The simulation can be either, MiL (if there component is a white-box model) or SiL (the model is an executable black-box model).

![Component Model Testing Workflow](image)

**Figure 14-Component Model Testing Workflow**

4.5.1 **Automatically generated Test cases**

Test cases can be automatically generated from a functional perspective. To this end, the requirements of the components are specified in a DSL and for each of the specified requirements, the test generation tool generates one or more test cases considering different possible scenarios.

4.5.2 **Manually created Test cases**

There might be test cases that will be manually needed to be generated. This could be for different reasons (e.g., when a requirement is too complex to be specified in the DSL or when the engineers wants to test a specific scenario). To this end, there is a possibility to manually generate the STI file containing the test case information, or use a classic test tool (e.g., AutomationDesk) to graphically generate the test case and export its STI format.

4.6 **Workflow Step IV: System Integration**

For test and simulation purposes in the use cases, there is the requirement to integrate the components of the electric drive into an overall system. The components are available in different versions for the different vehicle topologies in the SYNECT system. The topologies can be defined using the SYNECT Variant Management function. To
integrate a topology, it must be selected in SYNECT. Subsequently, the corresponding models as *.FMU or *.XMODEL are compiled and exported to the simulation platform xMod. SYNECT also generates a *.XMIPS file that contains the metadata of the topology-related models for integration in xMOD. The models and configuration data are then imported into xMOD and the simulation is initialized. The model inputs can be manipulated during the runtime of the simulation via controls on the screen.

**Figure 15-System Integration Workflow**

### 4.6.1 System Integration Details

To support the abstract definition of a system model architecture, SYNECT Model Management provides a way to define a so called “System Model”. It contains the interface definitions and the interconnection of the blocks (components). It is called abstract since it does not refer to concrete model implementations (FMUs, XMODELS or even SIMULINK models). One System Model can be the starting point for numerous different Co-Simulation integrations. Different vehicle topologies might require different System Models. Figure 16 shows just an arbitrary example.

**Figure 16-The System Model**
For each block (component) in the System Model there are normally many different interface compatible alternative implementations of a component (variants of components). E.g. M1,1 and M1,2 for the abstract model M1 in Figure 17. To integrate a System Model, the user selects one of the alternative component models for each abstract model. By this, many integrated System Models can be created. This step is done in SYNECT Model Management. Before it comes to simulation (or test), alternative parameter sets can be selected for the integrated system model. Finally, the fully integrated system is exported to a specific Co-Simulation environment (e.g. xMOD).

Figure 17 - Integration Steps

4.6.2 Co-Simulation with xMOD

The various steps in creating a co-simulation that can be run in xMOD are the following:

- Generating a model
- Integrating models as MIPS (a graphical description of interconnected models)
- Constructing the simulation
- Running the co-simulation

Figure 18 - Integration Steps
The following dataflow represents the process of creation of a co-simulation in xMOD.

**Figure 19-Component Model Development Workflow**

A MIPS view is shown on the following picture. It shows 3 models connected and interconnected representing 3 components (Driver model, Control model and a Motor model).

**Figure 20-MIPS example**

The model structure is available by opening the model in the MIPS view.

It is possible to add specific dashboards for any MIPS created in xMOD. The xMOD editor provides a set of tools to help the user building these specific screens. The following example shows an example of dashboard application.
4.6.3 Integration in SIMULINK

Instead of using a dedicated Co-Simulation platform such as xMOD some users might want to import black-box models from other partners in SIMULINK and connect these external components with their own white-box (native SIMULINK blocks) parts to run MIL/SIL simulation. There are different possible approaches to support this use case. An important requirement is that the external models must be IP protected. In the following the existing possibilities to integrate the external component models in SIMULINK are listed:

1. SIMULINK S-Function Block (MATHWORKS)
2. FMI Import Block (MATHWORKS, MODELON, etc.)
3. SIMULINK Protected Models (MATHWORKS) [10]

In any case the block representing the external component model will be treated as an atomic unit in SIMULINK. This can change the evaluation order of model equations and lead to (artificial) algebraic loops or changed numerical simulation results.

4.7 Workflow Step V: System Testing

To automatically test a specific system, the first step is to formally define its requirements in the developed DSL. Furthermore, it is noteworthy that in the DSL the interface between the test tool and the system needs to be specified. This step is important as in the case of system testing, different components will be integrated. However, some parts of the system will be interfaced with the test execution tool (e.g., Throttle request pedal, Brake request pedal, etc.). This is important to be considered from the beginning of the system’s lifecycle, otherwise there might be no chance to stimulate the system. After defining the requirements of the system, the test engineer will choose which requirements need to be tested in addition to the type of test cases that are wanted to be generated (e.g., long test cases, robustness test cases, etc.). For each of the selected requirements, one or more test cases will be generated. In this context, a test case will include a set of stimuli signals as well as the expected outputs for this stimulation signals. With this information, the following step will be to execute each test case one by one; with the generated test cases it is possible to compare the simulation signals with the expected outputs, and thus generate a specific verdict (e.g., test case pass or fail). With this verdict, the outcome of the test is reported.
Figure 22 - System Testing Workflow
5 Risk Register

A Risk Management Plan has been developed by the coordination team. It was approved by the first General Assembly (GA01). The plan identifies risks and contingency plans, and risk management protocols. It will be monitored by the Steering Committee, and be updated accordingly at six monthly intervals. More information on the risk management procedures can be found in D8.1 “Project Handbook”.

<table>
<thead>
<tr>
<th>Risk number</th>
<th>Description of Risk</th>
<th>Proposed Risk Mitigation Measure</th>
<th>Probability / effect¹</th>
</tr>
</thead>
</table>

¹ Level of probability / effect: 1 = high, 2 = medium, 3 = low

5.1 References


[4] 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


[8] Santaroni, L. et al, HIFI-ELEMENTS Deliverable D1.1; Document describing the safety requirements and modelling guidelines, 2018


6 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).
7 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.
Appendix