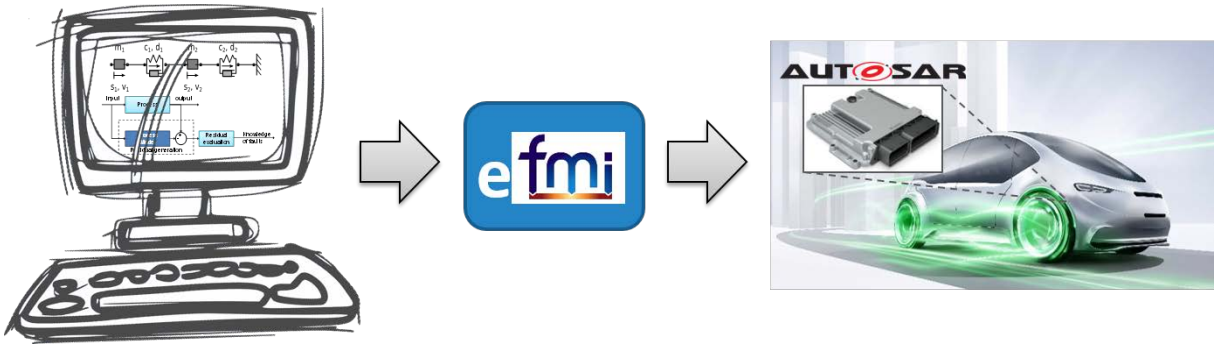
ITEA Office
High Tech Campus 69 - 3
5656 AG Eindhoven
The Netherlands
T + 31 88 003 6136
E info@itea3.org
W www.itea3.org

ITEA 3 is a EUREKA strategic ICT cluster programme

State of the Art Analysis

EMPHYSIS

Embedded systems with physical models
in the production code software



Date: Feb. 15, 2019

Editors: Oliver Lenord, Martin Otter, Reinhold Heckmann, Sascha Ridder, …

Contributors: Bosch, DLR, AbsInt, dSPACE, Siemens, UAntwerp

Reviewer: Yuri Durodie, Bert van Acker, Christian Bertsch

Document version #	Date (yyyy/mm/dd)	Changes
1.0	2019/02/15	Initial version

Contents

1 Introduction

1.1 EMPHYSIS Overview

The major goal of the EMPHYSIS project is to enhance production code of embedded control systems in automotive vehicles in order to improve the performance of the underlying system:

Typical ECU: Bosch MDG1

- faster and safer operation,
- improved driving dynamics,
- driving automation,
- reduced energy consumption,
- reduced emission,
- reduced maintenance costs.

Additionally, cost and time for the software development of these embedded systems shall be reduced. This is achieved by providing *physics-based models* in an *automated* way on *electronic control units* (ECU), micro controllers, or other embedded systems.

A physics-based model consists of a physical model of the system to be controlled or monitored, such as an engine or a powertrain, which is able to predict the behavior of the system in its whole operating region. It is typically described by differential and algebraic equations. The physical model is utilized in observers/virtual sensors, model-based diagnosis, or in advanced control algorithms (e.g., inverse models, non-linear dynamic inversion, model-predictive control) to achieve significantly better vehicle operations. There are several important applications, such as the ESP (Electronic Stability Program) or virtual sensors within the engine control, in which nonlinear physics-based models are already used on an ECU. Typically, this is hand-coded software that was implemented and tested in an elaborate and time-consuming way.

The goal of the EMPHYSIS project is to simplify and automate this process considerably and demonstrate the success by means of industrial automotive use cases. The major expected outcomes are:

- A new, open standard, FMI for Embedded Systems (eFMI), that is an extension of the very successful Functional Mock-up Interface (FMI, https://fmi-standard.org/) standard. It will use the core functionality of FMI, together with new interfaces so that the very special requirements of ECUs and micro controllers can be fulfilled.
- Seamless interoperability of eFMI with the automotive embedded system standard AUTOSAR, in order that (a) tools specialized on physical systems modelling and (b) tools specialized on AUTOSAR can work together using eFMI as the interface between the two very different worlds.
- New code generation techniques so that physics-based models described on a high level can be transformed to low level code fulfilling the requirements of ECU software and hardware.

This will boost the usage of advanced control and diagnosis functions within the ECU software and the usage of modelling and simulation tools for control design.

Results of the EMPHYSIS project will be usable outside of the automotive domain by utilizing eFMI components in non-automotive embedded systems. This requires support of eFMI on the target embedded system and might require small adaptations of the eFMI standard.

1.2 EMPHYSIS Components

Within the scope of the described development workflow for physics-based functions one can identify two different value chains as depicted in Figure 1. As of today these two value chains are disconnected.

Value chain for modelling and simulation

Modelling and simulation tool vendors provide simulation software and model libraries to simulation and control engineers at OEMs and suppliers. Control engineers use the ('plant') models to design controllers and diagnosis functions to fulfil the system requirements. Advanced software functions use physics-based models within the ECU software which can be designed also in the offline simulation tools. Then the second value chain begins:

Value chain for production code ECU-software

OEMs and suppliers implement production code software in specialized graphical programming environments or in C code. Tier-1 suppliers provide the ECU – often as a bundle of hardware, base software and varying fractions of the application software. Physics-based models have to be simplified, discretized and manually re-implemented for the ECU production code. This is done in the typical ECU-software programming environments that are unaware of the special properties and requirements of physics-based models.

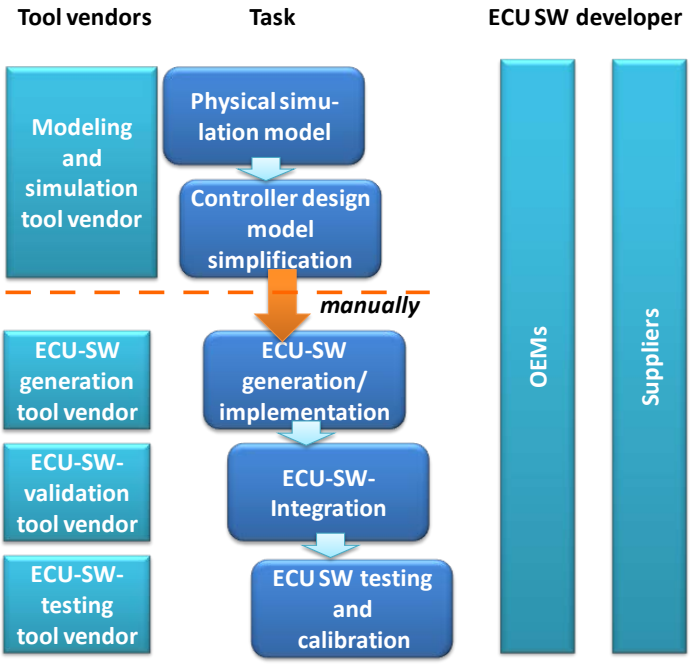


Figure 1: Value market chains for ECU Software development

With the intended eFMI standard those two value chains can be combined providing new business opportunities for modelling and simulation tools in the field of embedded software and new capabilities and a significant efficiency gain for the ECU-software generation.

Based on this value proposition of the eFMU standard this state of the art document is considering the technological developments in the field of

- Modelling and Simulation Software
- Controller Design Technologies and Software
- ECU Software Development and Integration Tools
- ECU Software Validation, Verification & Test Tools

with special emphasize on the interoperability and workflow integration of the corresponding tools.

2 Related Projects

2.1 Related Collaborative Research Projects.

Name, period, (program)	Technical Focus	Relationship
MODELISAR 2008 – 2010 (ITEA) https://itea3.org/project/modelisar.html	Development of the FMI model exchange and co-simulation standard for offline simulation. It was planned to develop FMI for embedded systems as well. This turned out to be too ambitious and no attempt was made to take the requirements of embedded systems into account. Only a feasibility study was performed how FMI could be utilized in AUTOSAR.	EMPHYSIS extends the FMI standard for embedded applications (= eFMI), e.g. introduces an intermediate model behavioral format (target independent, mathematical description) and a generic or target dependent production code format suited for ECUs.
AMALTHEA 2011 - 2014 (ITEA) https://itea3.org/project/amalthea.html	AMALTHEA defined an open platform for embedded multicore systems. It provides a framework, to partition an existing software for multicore systems.	AMALTHEA does not address code generation for physics-based models, but there can be a future usage of AMALTHEA results for the distribution of very large controllers with physics-based models exported as multiple runnables/eFMUs for multicore systems
AMALTHEA	AMALTHEA4public extends the Eclipse	See AMALTHEA; general

Name, period, (program)	Technical Focus	Relationship
4public 2014 - 2017 (ITEA) https://itea3.org/project/amalthea4public.html	IDE for Automotive Software Developers for multicore systems	availability through open source Eclipse plug-in
SAFE 2011 - 2014 (ITEA) https://itea3.org/project/safe.html	SAFE focused on safety-relevant automotive systems, extending the AUTOSAR architecture with a system description with the EAST-ADL language to increase reliability	Safety analysis from SAFE can be applied to safety critical model-based components defined by the planned eFMI standard of EMPHYSIS.
MODRIO 2012 - 2016 (ITEA) https://itea3.org/project/modrio.html	Extends tools for system design to system operation using nonlinear physical models for state estimation and model predictive control. The MODRIO demonstrators are targeted for PC based online applications in power plants and for applications on rapid prototyping hardware.	EMPHYSIS will use MODRIO results and provides the missing technology and tools to fulfil the special requirements of production code of model-based diagnosis components and model-based controllers on electronic control units, micro controllers and other embedded systems.
ACOSAR 2015 - 2018 (ITEA) https://itea3.org/project/acosar.html	ACOSAR defines a distributed co-simulation protocol for high performant co-simulations via networks. It is published in Feb. 2019 as Modelica Association standard DCP and is further developed via the Modelica Association.	No direct connection. ACOSAR extends FMI for soft realtime, distributed co-simulation of the plant model part of HiL, ViL systems, while EMPHYSIS focuses on embedded systems. Both technologies could be combined for the validation of ECUs with included eFMUs.
IMPROVE 2013-2016 FP7	New electric vehicle power and ICT architecture: Multi-level tailored integration of Amesim models in multiple control system ranging from low-end ECU (engine and subsystems controls) to	The link with EMPHYSIS is the return of experience of multiple types of model integration into controls (full rolling vehicle demonstrator) and the issues with

Name, period, (program)	Technical Focus	Relationship
	telematics (eco-driving) and cloud (range prediction). On-board embedded models (ECU, telematics) involved several reduction methods like linearization and networks of response surfaces (complex heat-pump based cooling system combined with HVAC). Lowest target was running below 80Mhz while supporting the more complex model.	control-tailored model integration which should be solved with eFMI.
OPENCPS 2015-2018 ITEA3 https://itea3.org/project/opencps.html	The project focuses on interoperability between the standards Modelica/UML/FMI, improved execution speed of (co-)simulation, and translation validation of Modelica-generated code.	The improved technology for Modelica code-generation and FMI-based simulation (in OpenModelica) will be taken as a basis for the eFMI generation from OpenModelica (T4.6).
Model based force measurements(MoForM) 01/02/2016 – 01/02/2020	Knowledge on (internal and external) dynamic forces and torques is of crucial importance, both during the prototype development phases of mechatronic products, machines and processes, as well as during their operational lifetimes. Measuring forces is a time consuming, error-prone, expensive and often intrusive process. Furthermore, it occurs regularly that force measurements at the desired locations are prohibited due to space limitations or too harsh circumstances. The main goal of the project is to develop a breakthrough force/torque measurement technology by adopting a virtual sensing strategy. This involves the evaluation and development of single (Kalman filter based) and multistep (Moving Horizon Estimation based) estimators that combine high-fidelity physical models and physically inspired grey box models with affordable non-intrusive sensors to retrieve unknown forces in a fast (possibly real-time),	The link with EMPHYSIS is the return of experience on the deployment of high-performant physics-based models on restricted embedded platforms.

Name, period, (program)	Technical Focus	Relationship
	accurate, in-situ and on-line manner. The targeted performance is defined in cooperation with industry and spans from real-time in-situ force estimation with a 10 Hz bandwidth and a 20 dB dynamic range to on-line in-situ force estimation with a 200 Hz bandwidth and an 80 dB dynamic range. The estimation technologies should be able to account for the non-linear dynamic effects as encountered in mechatronic drivetrains and systems.	

2.2 Conclusions

Some of these projects have laid a good foundation to build upon, but none of the projects is or has addressed the specific needs of the intended tool chain for designing physics-based functions for embedded targets.

3 Related Standards

3.1 Automotive Standards

Name	Technical Focus	Relationship
Classic AUTOSAR https://www.autosar.org/standards/classic-platform/	Standardization for automotive development and artefact exchange between different development functions in the development process	A standardized software framework providing a set of specifications for classical static ECU software architectures. Those describe basic software modules, define application interfaces and build a common development methodology based on a standardized exchange format. The standard is supposed to simplify the exchange of development artefacts and hence enhance the cooperation possibilities between OEMs and suppliers. One of the target production code environments of eFMI is AUTOSAR.
Adaptive AUTOSAR https://www.autosar.	Standardization for automotive	A standardized software framework providing a set of specifications for service-oriented

Name	Technical Focus	Relationship
org/standards/adaptive-platform/	development for high performance and non-safety and non-time critical applications and artefact exchange between different development functions in the development process	software architectures (SOA). Those describe basic software modules, define application interfaces and build a common development methodology based on standardized exchange format. The standard is supposed to simplify the exchange of development artefacts and hence enhance the cooperation possibilities between OEMs and suppliers. It allows to dynamically link services and clients during runtime, which is not possible with Classic AUTOSAR. This is no target for the first version of eFMI.
ASAM Association for Standardization of Automation and Measuring Systems https://www.asam.net/	Standardization for Automotive Development	Additional set of automotive standards providing tool-independent descriptions of protocols, data models, file formats and application programming interfaces (APIs) for use in the development and testing of automotive electronic control units. eFMI is planned to be defined in such a way that ASAM standards can be used (for example to describe tables). Since most ASAM standards are not publicly available, they will not be utilized in eFMI which is planned to become an open standard.

3.2 Model Exchange Standards

Name	Technical Focus	Relationship
FMI Functional Mock-up Interface www.fmi-standard.org	Low level, open standard to exchange simulation models for model integration and co-simulation purposes	Basis for the new eFMI standard.
Modelica https://www.modelica.org/	High level, open standard to conveniently model complex physical systems containing, e.g. mechanical,	Modelica is one of the description forms in which the physical models for eFMI are provided. Some eFMI tools developed in the EMPHYSIS project will transform Modelica models to production code in eFMI format.

Name	Technical Focus	Relationship
	electrical, electronic, thermal, control, process-oriented subcomponents	
VHDL-AMS https://standards.ieee.org/standard/1076_1-1999.html	IEEE standard for the description and the simulation of analogue, digital, and mixed-signal systems.	A high level description format for physical models, targeted to the electrical domain. In the EMPHYSIS project there are no tools planned that transform VHDL-AMS models to production code in eFMI.

3.3 Conclusions

None of the existing standards describes the exchange of physics-based models for embedded targets as addressed by the new eFMI standard.

4 Related Technological Development

4.1 Modelling and Simulation Software

Modelling and simulation of physical systems is a large area ranging from system modelling and simulation to finite element and computational fluid dynamics programs. Within EMPHYSIS the focus is on using simplified, linearized and nonlinear physical models from system modelling solutions, addressing 1D to 3D mechanics, engines, electrical, thermal and fluid domains. Modelling and simulation technologies of such systems have reached industrial maturity and several solutions are available as commercial and open source tools: These tools provide a graphical user interface to define physical systems in a convenient way (for example as electric circuit diagrams). Mathematically, the models are described by implicit Differential-Algebraic Equations (DAE), $0 = f(\dot{x}(t), x(t), t)$, that are solved either directly by an implicit integration method, or are first transformed in a pre-processing step to explicit Ordinary Differential Equations (ODE) $\dot{x}(t) = f(x(t), t)$ that are then solved numerically by explicit or implicit integration methods. There are single-domain tools that are primarily designed to model and simulate components from one domain (and most of them can import models from other domains via the FMI standard). These tools utilize typically dedicated numerical integration methods that take into account the special model structure of the particular domain (for example, multi-body system tools determine the first and second derivatives of constraint equations between parts and solve this overdetermined DAE system). Examples:

- Tools to model three-dimensional multi-body mechanics, such as SIMPACK, MSC ADAMS, RecurDyn.
- Tools to model electrical systems, such as PSpice.

- Tools to model fluid systems, such as gPROMS, Flowmaster, Thermoflow

There are multi-domain tools that are capable to model and simulate components natively from more than one domain. Examples are

- Tools based on proprietary modelling formats
  - Simulink (for block diagrams) and Simscape (for physical models).
  - GT-Suite
- Tools based on the VHDL-AMS standard, such as Saber, ANSYS Simplorer.
- Tools based on the Modelica standard, such as, Dymola, MapleSim, OpenModelica, SimulationX, OPTIMICA Compiler Toolkit.
- Tools based on the Bond Graph theory, such as Amesim or 20-Sim.

Modelica-based multi-domain tools typically use symbolic transformation techniques, for example by analytically differentiating constraint equations “sufficiently often” and by statically or dynamically selecting continuous-time states for the ODE description form. With these techniques it is possible, for example, to model multi-body systems (this is not possible with purely numerically based multi-domain approaches, such as VHDL-AMS, or Simscape). Furthermore, it is possible to exchange inputs and outputs of a nonlinear system and handle the resulting DAE system. This requires typically that equations must be analytically differentiated one or more times. Such inverse models can be utilized in nonlinear control systems, see below.

Within the ITEA project MODELISAR[1] the Functional Mockup Interface (FMI)[2] standard was developed that primarily defines physical models described by ordinary differential equations, algebraic and discrete equations on a low level with a combination of C code and XML. Here is a short overview, based on the FMI 2.0 document[3]:

An FMI component (also called FMU - Functional Mockup Unit) is an input/output block, see Figure 2, where either the environment must provide the integrator (= FMI for Model Exchange) or the integration method is already embedded in the FMU (= FMI for Co_Simulation). An FMU is distributed in one zip file. The zip file contains:

- An XML-file containing the definition of all exposed variables and other static information (it is then possible to run the FMU on a target system without this information, in other words with no unnecessary overhead).
- The C sources of the FMU, including the needed run-time libraries used in the model, and/or binaries for one or several target machines, such as Windows dynamic link libraries (.dll) or Linux shared object libraries (.so). The latter solution is especially used if the FMU provider wants to hide the source code to secure the contained know-how or to allow a fully automatic import of the FMU in another simulation environment.
- Additional FMU data (like tables, maps) in FMU specific file formats.

[1] https://itea3.org/project/modelisar.html
[2] https://fmi-standard.org/
[3] Functional Mock-up Interface for Model Exchange and Co-Simulation, version 2.0, July 25, 2014.

A schematic view of an FMU is shown in the next figure:

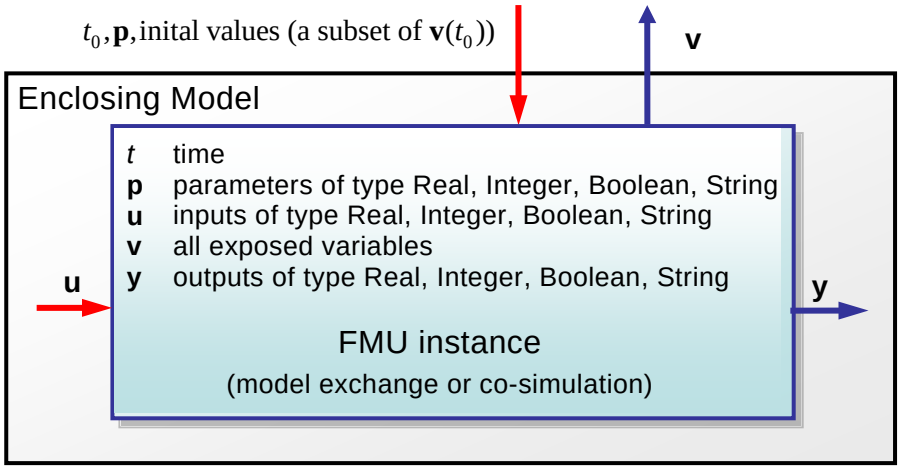


Figure 2: Data flow between the environment and an FMU.
***Blue** arrows: Information provided by the FMU.*
***Red** arrows: Information provided to the FMU.*

As of December 2018, more than 120 tools support the FMI standard (https://www.fmi-standard.org/tools). The FMI standard is used as basis for the EMPHYSIS project.

4.2 Controller Design Technologies and Software

The key technologies for EMPHYSIS are advanced controllers and monitoring devices utilizing knowledge about the plants via physics-based models. Most interesting are physics-based models that are able to describe the complete operating region. Typically, this is only possible by using sets of linearized models and/or nonlinear physics-based models. The technology for using such models within control systems is well-known in principle, such as

- utilizing nonlinear inverse models in the feedforward or feedback path (for example feedback linearization, nonlinear dynamic inversion, flat systems),
- optimization based controllers such as linear or nonlinear Model Predictive Controllers (= in every sample instant a trajectory optimization problem is solved where the control output is determined such that the system output follows a reference trajectory as best as possible, taking into account actuator limits),
- nonlinear state estimation (such as extended Kalman filter, unscented Kalman filter, moving horizon estimator),
- as well as model-based diagnosis.

Besides the large number of software tools available for modelling and simulation, there is currently no general purpose software available to utilize nonlinear physics-based models in embedded control systems. Within the ITEA project MODRIO[4] available modelling software is

[4] https://itea3.org/project/modrio.html

extended to utilize physics-based models in online applications. Targets are PC based online applications in power plants[5] and for applications on rapid prototyping hardware.

4.3 ECU Software Development and Integration Tools

The ECU software can be split into Application Software and Basic Software according to the AUTOSAR software architecture, see *Figure 3*:

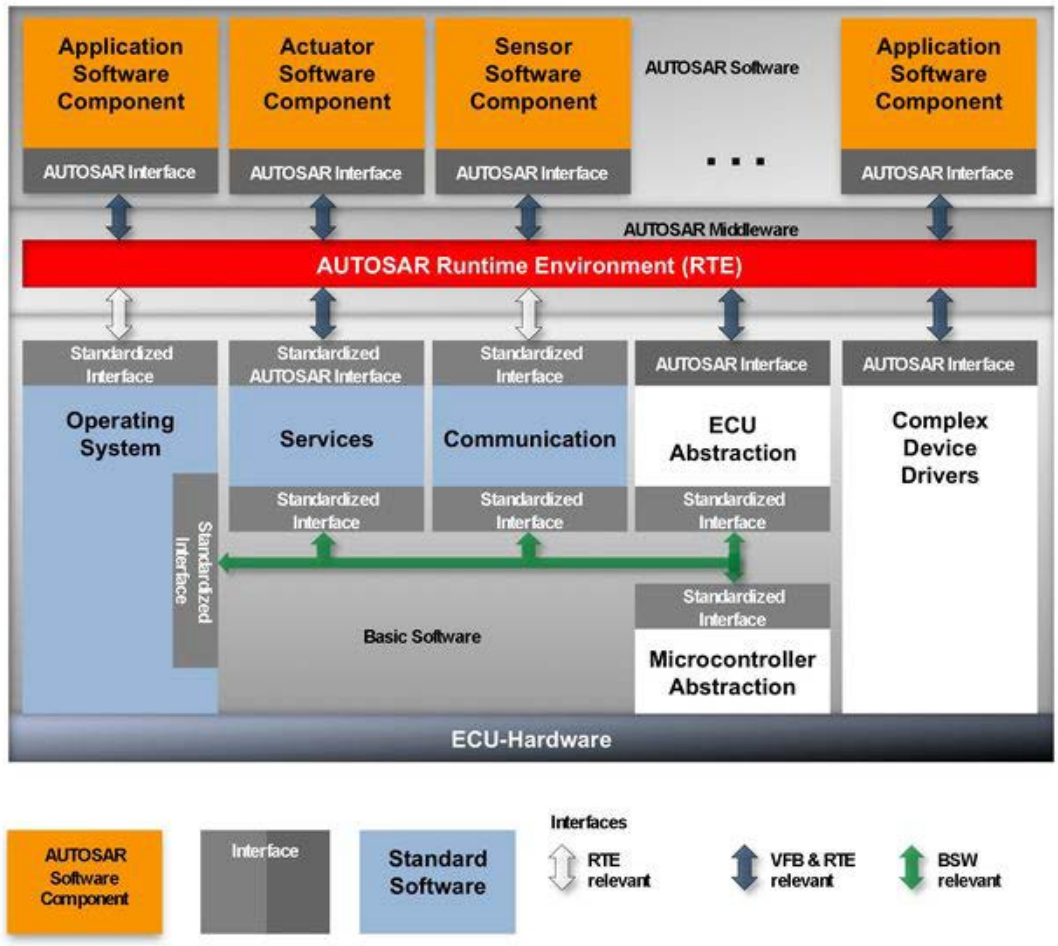


Figure 3: AUTOSAR architecture. Source: www.autosar.org

AUTOSAR is used in more and more automotive ECU software projects and partially replaces proprietary formats for ECU software components.

Application Software components realize “functions” of the system, for example, a controller calculating actuator signals from sensor measurements. They can be implemented either

- Directly in a programming language such as C.
- Generated from a graphical programming environment:

[5] Rüdiger Franke, Marcus Walther, Niklas Worschech, Willi Braun and Bernhard Bachmann: Model-based control with FMI and a C++ runtime for Modelica. Modelica Conference 2015, Versailles.

- Multi-purpose tools such as MATLAB/Simulink. Simulink is the most common tool for model design and simulation purposes in the automotive industry. With additional toolboxes such as the Embedded Coder by The Mathworks or dSPACE TargetLink ECU-compatible C-code can be generated, where coding and automotive standards, e.g. MISRA-C and AUTOSAR Standard are also supported. In both cases one has to restrict the usage of blocks either to a very limited subset of blocks and typically uses special in-house libraries or the mentioned 3rd party block sets to enable ECU-capable code generation.
- Specialized tools for ECU-Software such as ETAS ASCET
- UML Statechart tools from which C code can be generated. Those tools are used in industry, however, the scope of these is often limited to specific use cases.

Basic Software components comprise, e.g., hardware drivers and AUTOSAR service libraries providing e.g. basic numerical functions such as linear interpolation or trigonometric functions. The Basic Software is implemented typically in the C programming language or in Assembler.

The development of complex ECU software is often done in a distributed way by the collaboration of an OEM, an ECU software and hardware supplier and possibly additional suppliers. Depending on the form of the software exchange this is called model sharing (here model refers to a model of the software, e.g. a block diagram in MATLAB/Simulink), source code sharing (e.g. C-code) or object code sharing. The prerequisites for all of these possibilities are very high:

- to use exactly the same modelling tool by all partners for model sharing,
- to use the same coding guidelines and comply with all requirements of the importing system (e.g., AUTOSAR) for source code sharing,
- to use the same object code generation tool chain so that it fits together with the rest of the software for object code sharing.

The generation of a complete ECU-Software from the different components is performed with specialized tool chains including code checking for coding guidelines for critical systems, such as MISRA-C, with tools such as Astrée from AbsInt, as well as compilation with specialized compilers and specialized linkers. For the generation of AUTOSAR ECU software, specialized tools are used, e.g. for the creation of the AUTOSAR architecture and basic software configuration and the creation of the run-time environment (RTE). Examples are

- the AUTOSAR Builder from Dassault Systèmes SE,
- the ISOLAR tool suite from ETAS.

Future ECUs will have more and more cores. For example the current Bosch ECU MDG1 has up to six cores[6]. Parallelization of ECU-Software on different cores is a very difficult task addressed by several research projects listed in section 2.1.

Software is growing and becoming more complex. Tools like Simcenter ESD help in further decompose the (AUTOSAR) software architecture and evaluate the consistency and

[6] Rüger, J.-J., Wernet, A., Kececi, H.-F., Thiel, T., MDG1: The New, Scalable, and Powerful ECU Platform from Bosch, Proceedings of the FISITA 2012 World Automotive Congress - Volume 6: Vehicle Electronics, Springer, 2014

completeness of the requirements and the software architecture. This will improve the integration of the software as it will be divided over different internal and external suppliers. For this, understanding the impact that the hardware properties will have on the (physics-based) model will become crucial to provide meaningful requirements to both the controls engineers and the software engineers early in the development process. The eFMU standard would enable to study this in more detail.

4.4 ECU Software Validation, Verification & Test Tools

The release of a complete ECU-Software involves validating that it has the required properties. This can be done in part by testing, in part by employing static program analyzers that check properties of programs without actually executing them.

The Astrée tool is such a static program analyzer. It has been licensed from École Normale Supérieure and is being extended and marketed by AbsInt. Astrée is a sound static analyzer that finds potential run-time errors in C code (e.g. invalid pointer accesses and manipulations, index out of bound, division by zero, arithmetic overflows, etc.), or proves that such errors cannot occur. The tool can also check compliance to coding guidelines such as MISRA-C.

For some of the proofs the C code alone is not sufficient and more information has to be provided. For example, in order to prove that no overflow can occur, bounds for all parameters and input signals are needed. Such additional information can be provided by the user by means of special Astrée directives. To avoid this manual overhead, Astrée has already been integrated with the TargetLink code generator from dSPACE so that it can extract the necessary information from TargetLink's data dictionary. However, Astrée does not provide any support for analyzing FMI or eFMI code so far. Extending Astrée to support the analysis of code implementing eFMI components is one of the goals of the EMPHYSIS project.

Physics based models are currently used in the software architectures, but there is no framework for testing the validity of the model when they are adapted to serve the controllers need. This allows the software engineer to change the model in such a way it might not represent the physics anymore. eFMI could provide better boundaries in which the software engineers have space to adapt the model to the hardware needs and raise possible problems. This would prevent errors that would only show up at closed loop SIL or even worse will only show up during real testing. Tools that support open and closed loop testing of the software like Simcenter ESD could benefit from the eFMI framework to prevent errors early in the integration.

4.5 Conclusions

None of these current tools and tool chains covers the use case of porting a physics-based model in a tool-independent way from a given modelling and simulation tool unaware of ECU-specifics to an ECU.

5 Summary

The major goal of the EMPHYSIS project to develop the eFMI standard in order to enable a seamless end to end workflow from physical models to ECU software is not addressed by any other research project or technical development. The expected benefits of future tool chains

based on the new eFMI standard can be confirmed. A significant gain in productivity in the collaborative development of advanced ECU software functions can be expected enabling future innovations in the modeling & simulation software market, ECU software market and the automotive market.

According to this state of the art survey no other technology is foreseen to compromise or negate the objectives of the EMPHYSIS project.