EMPHYSIS – the missing link between digital simulation and embedded software

Bosch Research Blog | Post by Oliver Lenord, 2021-09-14

This story is part of “Bosch Research Blog”
Discover the whole series

Developing a new function is fun. Testing it in your development environment is natural. Encoding it according to the terms and conditions of safety critical software is tedious. Getting it flashed onto a device that has less memory than you can imagine is a challenge. And executing it under hard real-time constraints on a microcontroller that has a CPU vastly weaker than your phone might be the final straw for your excellent initial idea to find a smarter way to operate and control your product.

This is the tragedy for so many engineers who learned the hard way that their outstanding knowledge of product physics is far from sufficient to develop a function for an embedded target. Building a sophisticated simulation model to optimize the behavior of their product is one thing, but satisfying the hard requirements of safety critical software is entirely different. This is especially true under the constraints of the very limited computation power and memory of even the latest and greatest Electronic Control Unit (ECU) used, for instance, in automotive engine control.

With Bosch striving for better solutions and new software innovations to make vehicles and products safer, more efficient, more enduring and reliable, the public research project EMPHYSIS (Embedded Systems with Physical Models in the Production Code Software) was initiated with Bosch as consortium leader. The aim: bringing together market leading industrial partners and top researchers from Europe, Canada and Japan to jointly design a new open standard that lays the foundation for developing new innovative tools that enable model-based functions to be realized directly in embedded software with better code and less effort!

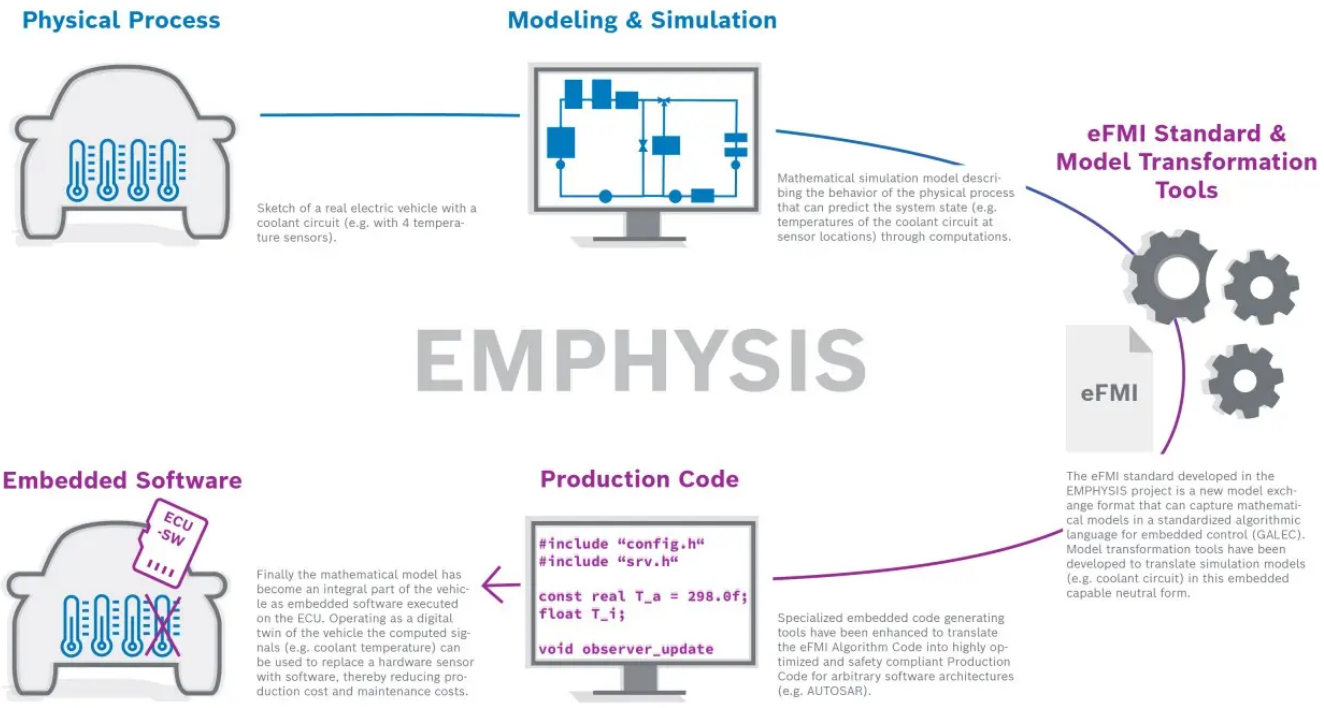

Figure1: The mathematical model of a physical process becomes part of the embedded software in order to e.g. replace a real sensor by a virtual sensor.

The use case

You may ask yourself what kind of function this is. Let me give you an example.

Imagine the coolant circuit of an electric vehicle. The purpose of the fan and the coolant pump is to make sure that all components are operated under optimal conditions and, most importantly, never overheat. A straightforward, but very cost intensive approach to this is a hardware temperature sensor at each component, so that a controller can regulate the system properly under all possible operation conditions and load cases.

Now imagine there was a clever engineer who knows exactly how the power losses are transferred as heat through the system. Maybe they have used that knowledge to create a simulation model and find proper dimensions of the components. Now they realize that, instead of measuring the temperature via the sensor, they could compute the same by utilizing the torque and speed signal and solving a set of differential equations to infer the heat flow through the system. This would allow the hardware sensor to be replaced with a piece of software, not only reducing the total cost but also giving Bosch a business opportunity to sell an additional function to the OEMs.

Their boss is probably very excited at first, but then might start asking uncomfortable questions like: are you sure that you can solve the differential equations at a fixed step size of 10ms? Will the solution be robust in single precision too? How much memory will be required on the stack? Will you be able to prove that it works on a real ECU? What is the risk that your approach might fail?

Using the simulation model, they should be able to answer the first question, but most likely will not have the time to translate the entire mathematical simulation model into plain C code and integrate it into an ECU.

So, what could the solution be? Just assume they could use the simulation environment not only to prove that the simulation works in principle, but also to transform the differential equations into a code-based algorithm and export it into an eFMU (Functional Mock-up Unit for embedded systems), which can be processed by the Bosch Build Tool Chain including embedded code generators for Bosch ECU targets. This would reduce the time and effort to realize this idea from weeks to just days and hours.

The project

That is why we've been working hard for 3.5 years in the European publicly funded project EMPHYSIS together with modeling and simulation tool vendors like Dassault Systèmes, ESI-ITI, Maplesoft, Siemens PLM, and leading companies in the field of embedded software like dSPACE and ETAS, to come up with something new that could join the opposite ends of a long development process. We did so by allowing all partners to utilize a highly automated process to generate code that lives up to the high quality standards of Bosch at 20-90 percent less effort, as demonstrated in our performance benchmarks and productivity assessment.

The technical deep-dive

Technically speaking, what makes eFMI special is not so much that code is generated automatically from a model. The really exciting and new thing about eFMI is that it provides a target-independent intermediate format defined by an entirely new language, which is the Guarded Algorithmic Language for Embedded Control: GALEC. This new programming language is able to guarantee that an algorithm described in this language can be translated into code that:

has static worst case execution time,
has static a-priori known memory demand,
can be statically proven to have no illegal memory access,

in other words, code that satisfies the hard requirements of automotive safety critical embedded software, which also makes it applicable to many other, less restricted domains like robotics, industrial applications and consumer goods.

With eFMI and GALEC, it is now possible to capture the computational essence of a model in a target independent form that provides a solid foundation for any code generator to produce highly optimized code for arbitrary runtime environments and software architectures. All this is built into a traceable, extendable and verifiable container architecture that goes far beyond a simple exchange format.

The biggest benefits of this solution and the associated eFMI workflow are:

accelerated development time

better use of domain experts by separating the concerns of physical modeling and embedded implementation issues
overcoming vendor lock-in through eFMI being published as an open standard on GitHub
enabling new ways of OEM supplier collaborations

Aiming to leverage the full potential of this technology beyond automotive, we're currently working with B/S/H and Bosch Rexroth to also apply eFMI to consumer goods and industrial applications.

The glimpse behind the scenes ?

Developing eFMI and the GALEC language was a big achievement and, at the same time, the biggest challenge of the EMPHYSIS project; it required computer scientists, simulation engineers, control engineers, mathematicians and embedded software developers to tear down the domain borders, learn from each other and reinvent themselves to finally agree on something that is so much more than the sum of its pieces. Doing this in the secure environment of a publicly funded project with experts from all across Europe and Canada was an exciting experience to the benefit of all partners and the global community.

What makes us proud as a team is that this success has been recognized not only by our business partners and the OEM Advisory Board representing the voice of the customer, but also by the ITEA organization. With excellent ratings in all three categories - innovation, business impact and standardization - the EMPHYSIS project will be honored with the ITEA Vice-Chairman Award of Excellence at the award ceremony in September 2021.

But there is more to come: In March 2021, just after the official end of the EMPHYSIS project, a new project was created under the umbrella of the non-profit Modelica Association. With Bosch, Dassault Systèmes, dSPACE and ETAS as initial members of the steering committee and Mercedes-Benz Passenger Cars having recently joined, the first official release of eFMI 1.0 is targeted within 2021.

With 14 tools developed within EMPHYSIS, eFMI already has a broad foundation even before its official release. But we're looking forward to more tool vendors and users joining this endeavor. Together we will establish this new open standard and revolutionize the model-based development of embedded software.

We’re looking forward to welcoming you to the eFMI community at https://eFMI-standard.org

(eFMI specification, test cases and OS tools accessible under https://github.com/modelica/).

Acknowledgements

This work has been funded in the context of the ITEA3 project 15016 EMPHYSIS by the German Federal Ministry of Education and Research (BMBF) under the grant number FKZ 01|S17023.

What are your thoughts on this topic?

Please feel free to share them or to contact me directly.

Author: Oliver Lenord

Oliver Lenord is research engineer and leader of publicly funded projects at Bosch Research. He earned his doctorate at the Mechatronics Institute of the University of Duisburg, Germany for his work on virtual prototyping and control of legged robots. Since 2016 he is affiliated with Robert Bosch - Corporate Research working in the field of model-based systems engineering. Formerly he led the simulation software development at Bosch Rexroth and worked as product manager for Siemens PLM in California, USA.

Recently he led the European ITEA3 project EMPHYSIS that delivered the new eFMI standard for embedded systems. He initiated the ongoing German research project PHyMoS concerned with hybrid (data and physics-based) methods for generating proper models for mechatronic systems. He is well connected within the Modelica community, represents Bosch in the Modelica Association project eFMI and is the vice-chairman of the Open Source Modelica Consortium (OSMC).

Oliver on Researchgate
Oliver on Xing

Share this on: