((AbsInt: Halle 4, Stand 350))
**INDUSTRIECOMPUTER & EMBEDDED SYSTEME**

Modelltransformation und automatische Codegenerierung

# Physik „inside“

Der steinige Weg vom abstrakten Simulationsmodell zum Embedded-Code wurde im Forschungsprojekt „FMI for embedded systems“ zur Schnellstraße ausgebaut. Mit dem neuen eFMI-Standard, der Modelltransformationen und Codegenerierungstechniken nutzt, lässt sich besserer Code mit weniger Aufwand entwickeln.

Von Christoff Bürger, Oliver Lenord, Reinhold Heckmann und Zdeněk Husár

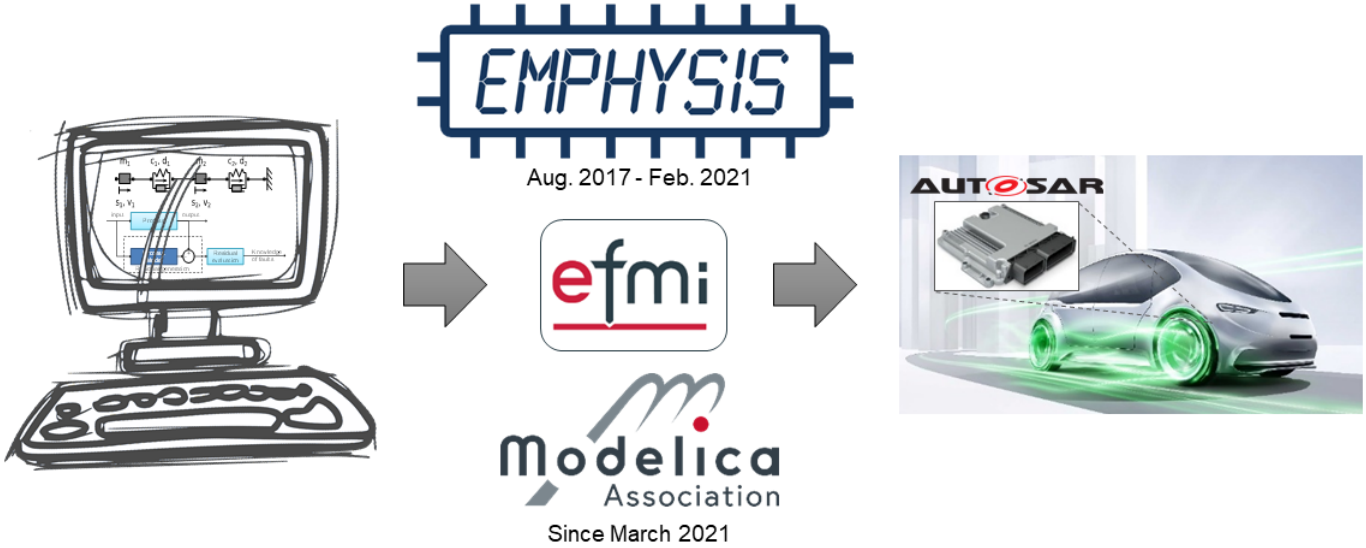


**Bild 1.** *Am europäischen Forschungsprojekt „EMPHYSIS – Embedded systems with physical models in the production code software“, das vom August 2017 bis Februar 2021 lief, beteiligten sich 26 Unternehmen und Forschungseinrichtungen aus Belgien, Kanada, Frankreich, Deutschland und Schweden. Seit März 2021 werden der in EMPHYSIS entwickelte eFMI-Standard und die Werkzeugprototypen von der Modelica Association (MA) im MA Project eFMI (MAP eFMI) zur industriellen Reife und ersten offiziellen Veröffentlichung vervollständigt. (Bild: EMPHYSIS Consortium, MAP eFMI)*

Der Innovationsdruck, für moderne Fahrzeuge effiziente Betriebsstrategien zu entwickeln, ist ungebrochen. In konventionellen Antrieben sind es die steigenden Anforderungen hinsichtlich Emissionsreduktionen, für moderne Elektrofahrzeuge sind es die Kundenerwartungen hinsichtlich maximaler Reichweite bei uneingeschränktem Klimakomfort und minimalen Ladezeiten bei uneingeschränkter Lebensdauer der Batterie, die große Anstrengungen erfordern.

Fortschrittliche Regelstrategien und Diagnosefunktionen unter Ausnutzung des Wissens über das physikalische Verhalten, in Form von mathematischen Modellen, bieten Lösungsansätze, um über die Software ungenutztes Potential zu heben, wo Hardware-Komponenten in ihrer Leistungsfähigkeit bereits weitgehend ausgeschöpft sind – wie z.B. im Absatz „Anwendungsfall Volvo“ betrachtet. Zudem eröffnen sich neue Möglichkeiten für die Zusammenarbeit von Simulationsexperten aus der Systementwicklung mit Softwareentwicklern in der Funktionsentwicklung in Richtung von Modell-, Software- und Hardware-in-the-Loop (MiL, SiL und HiL) Simulationen bis zur Embedded-Software auf Basis eines durchgängigen Modells – wie z.B. im Absatz „Anwendungsfall Mercedes-Benz“ erläutert. Bei alledem stellt sich die Frage, wie gut die generierten Lösungen im Vergleich zum Stand der Technik abschneiden und welche Kosteneinsparungen zu erwarten sind. Soviel vorweg: Erste Performance Assessments mit Werkzeugen des neuen eFMI-Standards liefern sehr ermutigende Ergebnisse, doch dazu mehr im Abschnitt „Studie Bosch“.

## Vom Modell zum Code – Herausforderungen

Ein mathematisches Modell bestehend aus Differentialgleichungen und algebraischen Gleichungen auf einem Steuergerät zu lösen ist nach dem Stand der Technik kein grundsätzliches Problem. Allerdings gilt es eine Vielzahl an Randbedingungen zu beachten, die zwingend einzuhalten sind. Dies gilt insbesondere, wenn es sich um Funktionen für sicherheitskritische Anwendungen handelt.
Wird ein mathematisches Modell eines physikalischen Systems in einem Simulationswerkzeug entwickelt, ist dieses typischerweise auf die Analyse des dynamischen Verhaltens ausgerichtet. Entsprechend stehen leistungsfähige numerische Lösungsverfahren zur Verfügung, die es erlauben, auf einem leistungsstarken PC ohne Echtzeiteinschränkungen – Stichwort offline simulation – eine robuste Lösung zu berechnen. Ist man nun bestrebt ein derartiges Simulationsmodell in Echtzeitanwendungen wie auf Hardware-in-the-Loop (HiL) Prüfständen oder als Bestandteil einer embedded Software einzusetzen, besteht keine Möglichkeit, dies in automatisierter Weise zu tun. Eine Reimplementierung als Embedded-Code, typischerweise als reiner C-Code, ist unausweichlich. Dies erfordert ein umfassendes Expertenwissen, das weit über das physikalische Verständnis und die mathematische Modellierung hinausgeht. Vielmehr bedarf es umfassender Kenntnisse in den Bereichen Numerik und Programmierstandards für Embedded-Software, z.B. MISRA C:2012, um schließlich zu einer Implementierung zu kommen, die den hohen Qualitätsanforderungen, wie z.B. im Automobilbereich, genügt, und auf den vergleichsweise leichtgewichtigen Embedded-Prozessoren robust und zuverlässig läuft. Handelt es sich dabei um nicht-triviale physikalische Sachverhalte, so ist zudem oftmals die notwendige Transformation in eine algorithmische Lösung aufwändig, wenn nicht sogar das Maß des praktisch handhabbaren übersteigend – was im Umkehrschluss bedeutet: Reduktion des physikalischen Modells und seiner Qualität oder komplett anderer Entwurf ohne solches.
Typische Anforderungen eingebetteter, sicherheitskritischer Software sind:

- Minimale Beanspruchung von Speicher für Daten und Code.
- Minimale Beanspruchung der CPU.
- Beschränkung auf Datentypen mit nur 32 Bit statt 64 Bit Gleitkommazahl-Genauigkeit (floating-point precision).
- Beschränkung auf statische Speicherallokation.
- Garantiert ausnahme- und fehlerfreie Ausführung.
- Garantie, dass die Laufzeit im ungünstigsten Fall innerhalb der verfügbaren Grenzen liegt.
- Robuste numerische Lösung bei fester Abtastrate (sampling rate).
- Zuverlässige Behandlung von Operationen mit „Not-a-Number“ (NaN) Rückgabewerten (IEEE 2019-07), teilweise undefinierten mathematischen Funktionen oder mehrdeutigen Lösungen.
- Garantierte Beschränkung von Signalen auf ihren Gültigkeitsbereich.

Code wie er von Simulationswerkzeugen für ausführbare Modelle oder für den Modellaustausch, z.B. FMI [1], generiert wird, genügt diesen Anforderungen bei Weitem nicht.

## Die Technik des eFMI-Standards

Im Rahmen des europäischen öffentlich geförderten Projekts „EMPHYSIS – Embedded systems with physical models in the production code software“ [2] (ITEA, Nr. 15016) wurde mit eFMI (FMI for embedded systems, [3]) ein neuer offener Standard entwickelt, der komplementär zum etablierten FMI-Standard (Functional Mock-up Interface, [1]) ist (**Bild 1**). Ermöglicht FMI den Austausch und die Integration von Simulationsmodellen zwischen mehr als 150 Simulationswerkzeugen, so bietet eFMI eine flexible Container-Architektur, um auf Basis eines targetunabhängigen Zwischenformats optimierten Code für unterschiedliche Zielplattformen und Software-Architekturen zur Verfügung zu stellen.
Für die Repräsentation von mathematischen Modellen als zyklisch aufrufbarer Algorithmus, wurde die neue Sprache GALEC (Guarded Algorithmic Language for Embedded Control) als Bestandteil des eFMI-Standards entwickelt. GALEC ist so konzipiert, dass ein in dieser Sprache formuliertes Berechnungsverfahren auf jeden Fall einen wesentlichen Teil der oben genannten Anforderungen erfüllt. So wird z.B. die statische Allokation von Speicher dadurch ermöglicht, dass in der Deklaration von Vektoren und Matrizen die Dimension nur durch zur Compilierzeit berechenbare statische Ausdrücke definiert sein darf. Indizierungen in multidimensionale arithmetische Ausdrücke und ihre Größen sind statisch auswertbar und prüfbar. Laufvariablen von Schleifen müssen ebenfalls statisch begrenzt sein und eine Rekursion ist ausgeschlossen, so dass Laufzeitschranken garantiert werden können. Implizite Typumwandlung zwischen Integer- und Real-Variablen ist ausgeschlossen und einfache Namensräume ohne Überladung und Polymorphismus forcieren die Einhaltung entsprechender MISRA-C:2012-Regeln. Des Weiteren bietet GALEC ein konsistentes Fehlerbehandlungskonzept mit garantierter NaN-Propagation und kontrollflussintegrierter Fehlerbehandlung; diese erlaubt die verzögerte Behandlung möglicher Fehler, ohne dass solche „verloren“ gehen können. Auf diese Weise kann sichergestellt werden, dass ein mathematisches Modell, das in einem ersten Schritt in GALEC-Code transformiert wurde, in einem zweiten Schritt in Embedded-System-tauglichen und MISRA-konformen C-Code übersetzt werden kann. Für die beiden Schritte sind normalerweise zwei verschiedene Codegeneratoren zuständig, die jeweils auf „ihren“ Schritt spezialisiert sind.

Eine Besonderheit des zweiten Codegenerierungsschritts von eFMI-GALEC nach Produktionscode ist die Möglichkeit, optimierten Code für unterschiedliche Architekturen bzw. unterschiedliche Prozessoren zu generieren. Diese unterschiedlichen Codevarianten können in mehreren, alternativen Production Code Containern innerhalb derselben eFMU (Functional Mock-up Unit for embedded systems) abgelegt werden, ohne die Verknüpfung zum selben Ausgangsmodell zu verlieren (siehe **Bild 2**). Sogenannte Manifest-Dateien in jedem eFMI-Container liefern die dazu erforderlichen Meta-Informationen. Diese ermöglichen dem Productioncode-importierenden Werkzeug – ohne den Code syntaktisch oder semantisch analysieren zu müssen – allein aus den in XML geschriebenen Manifestdateien herauszulesen, welche Funktionsnamen, Funktionssignaturen, Datentypen und Service-Funktionen verwendet werden, um die passende Variante zu identifizieren und in das Embedded-Softwareprojekt zu integrieren. Die XML-Manifeste der verschiedenen eFMU-Container bieten zudem Meta-Informationen bezüglich ihrer Abhängigkeiten sowie Checksummen generierter Artefakte, so dass Rückverfolgbarkeit und automatisches Erkennen veralteter Softwareartefakte möglich sind.
Diese Mechanismen erlauben die Rückverfolgung des C-Codes bis zum Ausgangsmodell sowie die Verifikation des Verhaltens durch Testläufe – durch ebenfalls in der eFMU mitgelieferte Behavioral Model Container, die Referenzszenarien zum automatischen Testen definieren.

## Der eFMI-Prozessablauf

Mit der eFMI-Technik erhalten Entwickler maximale Flexibilität in der Ausgestaltung des Entwicklungsprozesses und – auf Basis des offenen Standards – eine Wahlfreiheit hinsichtlich der verwendeten Werkzeuge. Durch die reichen Meta-Informationen und in den Containern hinterlegte Checksummen eignet sich eFMI auch als Archivierungsformat und als Mittel, um unterschiedliche Code und/oder Model-Sharing-Prozesse in Zusammenarbeit von Herstellern (OEM) und Zulieferern zu organisieren. Der eFMI-Standard bietet eine gute Ausgangsbasis für bedeutungserhaltende Transformationen von physikalischen Modellen zu Embedded-Software.

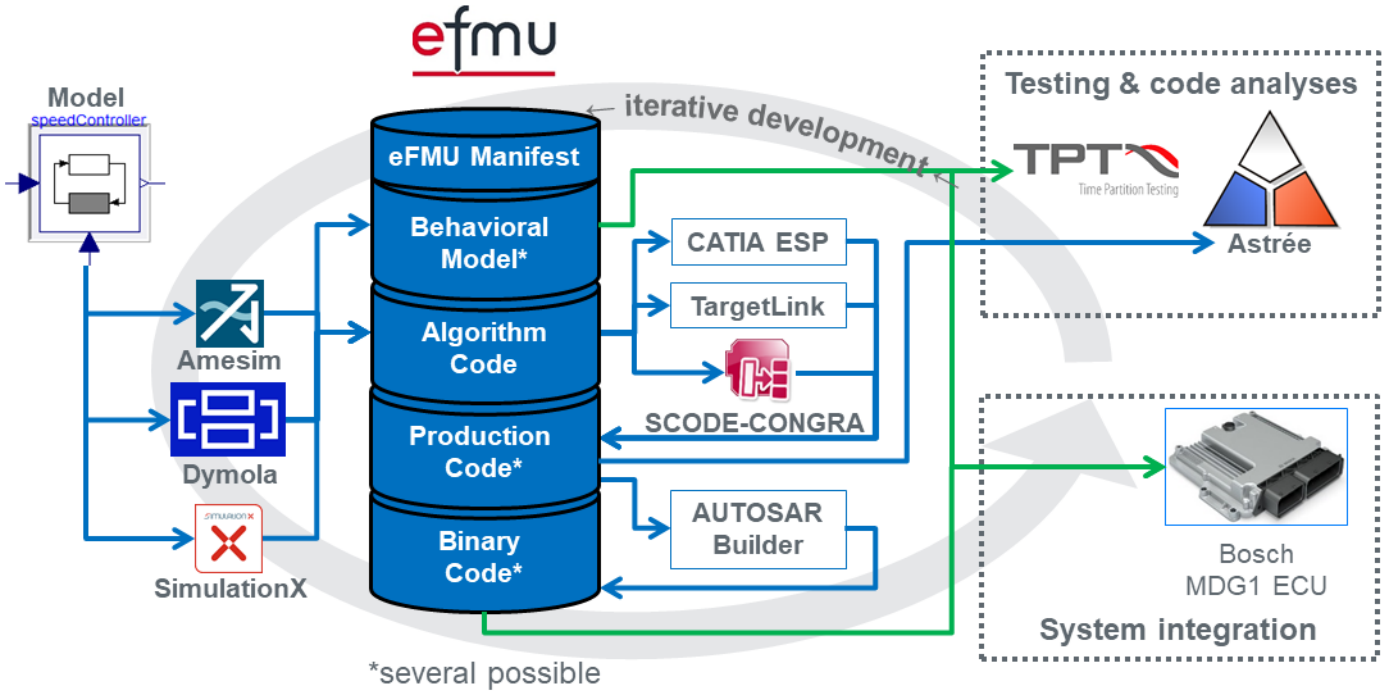


**Bild 2.** *Der eFMI-Entwurfsablauf am Beispiel der im EMPHYSIS-Projekt entwickelten eFMI-fähigen Werkzeugprototypen. Dargestellt sind (v.l.n.r):*

- *Die Generierung einer algorithmischen GALEC-Lösung aus einem physikalischen Gleichungsmodell (Algorithm Code Container erzeugt z.B. mit Amesim, Dymola oder SimulationX).*
- *Die Generierung der Referenztestszenarien aus Simulationen des physikalischen Modells (Behavioral Model Container erzeugt z.B. mit Dymola).*
- *Die Generierung des Produktionscodes aus der algorithmischen GALEC-Lösung (Production Code Container erzeugt z.B. mit CATIA ESP, TargetLink oder SCODE-CONGRA).*
- *Die Generierung von ausführbarem Binärcode aus Produktionscode (Binary Code Container für z.B. die AUTOSAR Adaptive Platform erzeugt mit AUTOSAR Builder) und das Analysieren und Testen von Produktions- und Binärcodes, z.B. mit Astrée und TPT, auf einem konkreten Steuergerät, z.B. Bosch MDG1.*

*(Bild: MAP eFMI, Dassault Systèmes)*

Ausgangspunkt des in **Bild 2** dargestellten Prozesses ist die Modellierung eines zu regelnden Systems in einer dedizierten Modellierungsumgebung, die beste Voraussetzung bietet, um auf Basis von komponentenorientierten Modellbibliotheken auch komplexe Systeme mit überschaubarem Aufwand zu modellieren und mit Mitteln der Simulation zu validieren, z.B. Modelica-Werkzeuge wie Dymola von Dassault Systèmes, SimulationX von ESI Group oder Amesim von Siemens.
Reglerstrukturen, vom einfachen PID-Regler bis zum flachheitsbasierten Regler unter Verwendung eines inversen Streckmodells, können direkt in der Simulationsumgebung mittels Model-in-the-Loop-Simulation (MiL) ausgelegt und hinsichtlich Robustheit und Stabilität analysiert werden. Die gefundenen Lösungen können automatisiert in GALEC-Code transformiert und zusammen mit den Referenzlösungen der MiL-Testfälle als Algorithm Code Container bzw. Behavioral Model Container in einer eFMU exportiert werden.
Embedded-Code-generierende Werkzeuge, z.B. TargetLink von dSPACE, SCODE-CONGRA von ETAS oder CATIA ESP von Dassault Systèmes, können diese eFMU einlesen. Anhand konkreter Code-Konfigurationen können targetspezifische Varianten generiert und als separate Production Code Container zu der eFMU hinzugefügt werden. Passend zu jedem Production Code wird zugehöriger FMI-kompatibler Wrapper-Code generiert, der es erlaubt, die gesamte eFMU als FMU zu kapseln und somit in jeder FMI-unterstützenden Simulationsumgebung einen Software-in-the-Loop-Test (SiL) durchführen zu können, der denselben Code ausführt, der auch auf dem Target ausgeführt werden wird.
Optional kann auch der für das Target compilierte Binär-Code in der eFMU als eigener Container abgelegt werden.
Neben Modellierungswerkzeugen und Codegeneratoren wird eFMI auch von Werkzeugen zur statischen Codeanalyse (z.B. Astrée von AbsInt) – beispielsweise für die Analyse von Fehlern bei Feldzugriffen, Überläufen oder Division durch 0 oder MISRA-C:2012-Regelverletzungen – und zum Testen (z.B. TPT von PikeTec) unterstützt, so dass umfassende Überprüfungen erfolgen können, um die Qualität des Codes zu garantieren. Dazu können die umfangreichen Meta-Informationen der entsprechenden Container Manifeste genutzt werden, um erforderliche Informationen und Zusammenhänge zwischen den verschiedenen Transformationsstufen vom Modell zum embedded Produktionscode zusammenzubringen; z.B. um existierende Behavioral Model Container automatisch zum HiL-Testen neuer Produktionscodes zu nutzen.
**Bild 2** fasst den eFMI-Ablauf zwischen den einzelnen Containern des eFMI-Standards am Beispiel der im EMPHYSIS-Projekt entwickelten Prototyp-Werkzeuge zusammen.

## Anwendungsfall Volvo

In diesem Anwendungsfall hat Volvo die Verbesserung der Steuerung des Elektromotors eines Elektrofahrzeuges betrachtet. Eine gute Motorsteuerung sollte Ruckler und Vibrationen vermeiden, die nicht nur für die Passagiere unangenehm sind, sondern auch die mechanischen Komponenten des Systems belasten und zu größerem Verschleiß führen. Dafür benötigt die Motorsteuerung möglichst vollständige und zeitnahe Information über den Zustand des gesamten Antriebssystems einschließlich der Antriebswelle und der Räder, insbesondere der Drehmomente und Kräfte, die an den individuellen Komponenten wirken.
Informationen über den Zustand des Antriebssystems können durch Sensoren an den verschiedenen Teilen des Systems geliefert werden. Der Einbau von immer mehr Sensoren scheint zwar auf den ersten Blick die Informationsdichte zu erhöhen, bringt aber Probleme nicht nur durch Mehrkosten und Platzbedarf, sondern auch durch die Gefahr der Überlastung der Kommunikationskanäle, die die gewünschte Verbesserung der Qualität der Information verhindern oder sogar ins Gegenteil verkehren kann. Des Weiteren impliziert sensorbasierte Kommunikation auch immer eine untere Grenze der maximalen Abtastrate, da die Kommunikationskanäle eine Signallaufzeit (delay) haben.
Eine mögliche Alternative besteht darin, ein fortgeschrittenes physikalisches Modell des Antriebssystems als virtuellen Sensor zu benutzen, also eine echtzeitfähige Simulation des kompletten Antriebssystems neben der eigentlichen Kontrollsoftware auf dem Steuergerät einzubetten. Die Simulation auf dem Embedded-System kann dann die gewünschten virtuellen Sensordaten, z.B. Drehmomente an Antriebswelle und Rädern, aus ihrem physikalischen Modell extrahieren und direkt an die Steuersoftware liefern. Die eFMI-Technik hilft dabei, existierende Modelle des physikalischen Systems wiederzuverwenden, die bereits verifiziert und durch Messungen validiert sind.
In dem Anwendungsfall wurde die eFMI-Technik erfolgreich genutzt, um Produktionscode aus dem physikalischen Modelica-Modell eines elektrischen Antriebssystems zu gewinnen und durch Vergleich mit einer Simulation des ursprünglichen Modells zu validieren. Die eFMI-Technik beschleunigte dabei den Übergang vom gleichungsbasierten Modell zu algorithmischem Code und half Fehler zu vermeiden, einerseits dadurch, dass nur ein einziges Modell entwickelt und validiert werden musste, und andererseits durch die Möglichkeit, dieses Modell automatisch in GALEC-Code und von da aus in Produktionscode zu transformieren und diesen durch Simulationen und Tests zu validieren. Volvo sieht es dabei als vorteilhaft an, dass der eFMI-Standard es erlaubt, die verschiedenen Transformations- und Validierungsschritte mit verschiedenen Werkzeugen verschiedener Hersteller durchzuführen.

## Anwendungsfall Mercedes-Benz

Auch in diesem Anwendungsfall ging es um die Entwicklung eines virtuellen Sensors, allerdings im Zusammenhang mit der Überwachung der dynamischen Eigenschaften eines automatischen Doppelkupplungsgetriebes mit zwei miteinander verbundenen Reibkupplungen. Dabei wurde ein eFMU-Container für ein vereinfachtes Getriebe-Systemmodell genutzt. Mit dessen Hilfe wird, aus den bisherigen Regelungssignalen und den Signalen echter Sensoren sowie den bisherigen modellierten Zuständen der Kupplungen, der aktuelle Zustand (Geschwindigkeit, Beschleunigung und Drehmoment) gemäß Modell hergeleitet. Der aktuelle Zustand wird dann wiederum verwendet, um neue Regelungssignale zu erzeugen.

Die dazu verwendeten physikalischen Modelica-Modelle sind keine Neuentwicklungen, sondern seit Jahren bei Mercedes-Benz zur Simulation – bisher rein offline – solcher Kupplungsgetriebe im Einsatz. Derartige physikalische Modelle von Kupplungsgetrieben sind in der Automobilindustrie Standard; die Wiederverwendung derselben zur automatisierten Generierung von Regelungssoftware ist ein echter „game changer".
Die physikalische Modellierung des Doppelkupplungsgetriebes führte zu einem „steifen" Gleichungssystem, das für eine stabile numerische Lösung spezielle hochkomplexe Lösungsverfahren erforderte. Diese speziellen Lösungsverfahren wurden von dem EMPHYSIS-Partner Dassault Systèmes in Dymola entwickelt und als eFMU mit GALEC-Code exportiert. Ein daraus generierter Produktionscode konnte für Software- und Hardware-in-the-Loop-Simulationen transformiert werden. Der durchgängige eFMI-Prozess verkürzt somit den Weg von einem physikalischen Streckenmodell zum ausführbaren Embedded-Code.
Allerdings zeigen die Simulationen noch teilweise unakzeptables Verhalten und es ist noch weitere Arbeit an den eingebetteten Lösungsverfahren notwendig. Doch die bisherigen Ergebnisse sind sehr vielversprechend. Eine erfolgreiche Lösung solcher numerischen Probleme wird einen viel breiteren Einsatz des eFMI-Standards auch in anderen industriellen Anwendungen mit vergleichbar hoher numerischer Komplexität ermöglichen.

## Studie Bosch

Im Rahmen des EMPHYSIS-Projekts hat sich Bosch die Aufgabe gestellt, zu evaluieren, inwieweit die mit der vorgestellten eFMI-Werkzeugkette weitgehend automatisiert generierten Lösungen die hohen Qualitätsanforderungen und Gütekriterien für sicherheitskritische Software von Bosch erfüllen. Des Weiteren galt es zu quantifizieren, ob die zu erwartenden Performanzverluste des generierten Codes gegenüber händischen Implementierungen in engen Grenzen von max. +20 % liegen und durch eine signifikante Einsparung bei den Entwicklungsaufwänden gerechtfertigt werden können.
Dazu hat Bosch aus seiner langjährigen Erfahrung Probleme formuliert, die bekanntermaßen für Code-generierende Werkzeuge herausfordernd sind, um robuste, recheneffiziente und kompakte Implementierungen zu finden. Auf Basis dieser Probleme wurden sechs Testfälle definiert, die von erfahrenen Embedded-Softwareentwicklern händisch implementiert wurden, um als Benchmark für die durch die eFMI-Werkzeugkette generierten Lösungen zu dienen. Umfassende Messungen von Laufzeit und Speicherbedarf auf einem Bosch-MDG1-Steuergerät wurden durchgeführt. Zudem erfolgten die gängigen Code-Quality-Checks.
Für die Durchführung des gesamten Prozesses, von der Erstellung des Modells, dessen Validierung, Implementierung und Test auf dem Steuergerät wurden die Arbeitsstunden erfasst.
Das Ergebnis dieser Studie war, dass in allen Fällen die gesetzte Grenze von 20 % Performanzverlust eingehalten wurde. Erstaunlicherweise wurde sogar in vier von sechs Fällen die händische Implementierung übertroffen. Dies war dadurch zu erklären, dass die symbolischen Modelltransformationen durch die Wahl der Zustandsvariablen und durch symbolische Vereinfachung der Gleichungssysteme häufig bessere Lösungen gefunden haben, als es durch die händische Herleitung gelungen ist.
Auch die C-Code-Generatoren hatten teilweise weitere Optimierungen gefunden, die bis dahin unentdeckt geblieben waren. Sicherlich wäre es möglich, all diese Verbesserungen in die händische Implementierung zu übernehmen und dann die generierte Lösung zu übertreffen, dennoch konnte festgestellt werden, dass im Mittel mit eFMI besserer Code mit deutlich geringerem Aufwand und 50–90 % Produktivitätssteigerung erzeugt werden konnte als mit einer händischen Entwicklungsmethodik nach dem Stand der Technik.

## Status und Weiterentwicklung des eFMI-Standards

Die eFMI-Spezifikation wurde nach Abschluss des EMPHYSIS Projekts an die Modelica Association [4] zur Weiterentwicklung und Standardisierung übergeben. Im März 2021 wurde das Modelica Association Project eFMI (MAP eFMI) gegründet, das seitdem an der Weiterentwicklung arbeitet.
Eine ausgereifte Vorabversion (pre-release draft) des eFMI-Standards ist verfügbar [3]. Eine gute Übersicht des eFMI-Standards bieten [5] und [6]. Die industriellen Anwendungsfälle des EMPHYSIS-Projekts, sowie die BOSCH-Assessment-Studien, sind in einem öffentlichen technischen Bericht [7] beschrieben.
Die im Rahmen des EMPHYSIS-Projekts entstandenen 13 Werkzeug-Prototypen wurden anhand der veröffentlichen Modelica-Test-Bibliothek [8] kontinuierlich verbessert. Erste kommerziell verfügbare Produkte werden mit den in 2022 anstehenden Produkt-Releases erwartet. *hs*

**Literatur**


**[1] Functional Mock-up Interface. Modelica Association, Website, https://fmi-standard.org**

**[2] EMPHYSIS – Embedded systems with physical models in the production code software. ITEA 4 – the Eureka Cluster on software innovation, Website, https://itea4.org/project/emphysis.html**

**[3] ITEA Project EMPHYSIS. EMPHYSIS Consortium / MAP eFMI, Website, https://emphysis.github.io**

**[4] Modelica Association, Website, https://modelica.org**

**[5]** *Lenord, O; et al.:* **eFMI: An open standard for physical models in embedded software. 14th Modelica Conference 2021, 20.–24. September 2021, Konferenzband, S. 57–71, https://doi.org/10.3384/ecp2118157**

**[6]** *Bürger, C.:* **eFMI Status and Outlook. International Modelica Conference 2021, 23. September 2021, Videopräsentation, https://emphysis.github.io/pages/downloads/Modelica-Conference-2021-MAP-eFMI.mp4**

**[7] EMPHYSIS – D7.9 eFMI for physics-based ECU controllers. ITEA, öffentlicher Bericht, Version 1, September 2021, https://emphysis.github.io/pages/downloads/emphysis-public-demonstrator-summary.pdf**

**[8] modelica / efmi-testcases. GitHub, Website, https://github.com/modelica/efmi-testcases**

## Dipl.-Inf. Christoff Bürger

hat umfangreiche Erfahrungen im Übersetzerbau und Design formaler Sprachen. Er arbeitet seit sechs Jahren bei Dassault Systemès am Compiler-Kern der Dymola-Modelica-Entwicklungs- und Simulationsumgebung. Er ist Hauptentwickler von eFMI-GALEC und Hauptautor der GALEC- und Algorithm-Code-Representation-Kapitel des eFMI-Standards sowie Mitautor der Behavioral-Model-Representation-Spezifikation. Als Projektleiter von MAP eFMI ist er Steering-Committee-Mitglied der Modelica Association und stellvertretender library officer des Modelica.Clocked Pakets der Modelica Standard Bibliothek (MSL).

Christoff.Buerger@3ds.com

## Dr.-Ing. Oliver Lenord

promovierte am Lehrstuhl für Mechatronik der Universität Duisburg-Essen. Seit 2016 arbeitet er als Forschungsingenieur und Leiter öffentlich geförderter Projekte am Robert Bosch Campus in Renningen bei Stuttgart. Seinen beruflichen Einstieg hatte er 2004 in der Vorausentwicklung der Bosch Rexroth AG, deren Simulationssoftware-Entwicklungsgruppe er später leitete, bevor er 2011-2014 zu Siemens PLM nach Kalifornien wechselte. Dort war er im Innovationsfeld Systems Engineering und später als Produktmanager tätig. Bis 2021 leitete Oliver das europäische ITEA3 Projekt EMPHYSIS, in dessen Rahmen der neue eFMI-Standard entwickelt wurde. Derzeit leitet er das von Ihm initiierte BMWi geförderte Projekt PHyMoS und hält den stellv. Vorsitz des Open Source Modelica Consortiums (OSMC).

Oliver.Lenord@de.bosch.com

## Dr. Reinhold Heckmann

arbeitet seit über 20 Jahren bei AbsInt Angewandte Informatik GmbH als Senior Researcher im Bereich der statischen Programmanalyse und ist Mitautor zahlreicher wissenschaftlicher Veröffentlichungen zu diesem Thema. Zu seinen Aufgaben zählt die Vertretung von AbsInt bei der Beantragung und Durchführung von Forschungsprojekten wie z.B. EMPHYSIS.

Heckmann@absint.com

## Ing. Zdeněk Husár, PhD

arbeitet seit 2005 als Berechnungsingenieur in der Powertrainentwicklung bei Mercedes-Benz AG. In dieser Zeit hat er umfangreiche Erfahrungen und experten Kenntnisse auf dem Gebiet der Software- und Hardware-in-the-Loop Gesamtfahrzeug-Simulation mit konventionellen und elektrifizierten Antrieben gewonnen. Seine Erfahrungen aus der Getriebestreckenmodellierung sind in den Anwendungsfall Mercedes-Benz eingeflossen und haben die Teilnahme der Mercedes-Benz AG in dem EMPHYSIS-Projekt, bzw. in MAP eFMI motiviert. Zdeněk repräsentiert die Mercedes-Benz AG als Steering-Committee-Mitglied in MAP eFMI.

Zdenek.Husar@mercedes-benz.com