Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 21 (2014) 46 – 51

www.elsevier.com/locate/procedia

24th CIRP Design Conference

Architectural Design Space Exploration of Cyber-Physical Systems using the Functional Modeling Compiler

Arquimedes Canedo[a], Jan H. Richter[b]

[a]*Siemens Corporation, Corporate Technology, 755 College Road East, Princeton, 08540, USA*
[b]*Siemens AG, I IA ATS 43, Gleiwitzer Str. 555, 90475, Nuremberg, Germany*

* Corresponding author. Tel.: +1-609-734-3317; fax: +1-609-734-3583. *E-mail address:* arquimedes.canedo@siemens.com

Abstract

This paper reports our experience in the development of a novel concept design support tool for cyber-physical systems (CPS). We show that the various disciplines in CPS design can be brought together to enhance the communication and requirements negotiation among engineers and organizations, to enable multi-disciplinary simulations to evaluate the system-level impact of domain-specific design decisions, and to reduce the overall design cycle. Our method relies on functional modeling to create a technology-independent description of what the system does, and uses a Functional Modeling Compiler (FMC) to synthesize technology-dependent solutions that can be directly used to perform architectural design space exploration using multi-disciplinary simulations in AMESim and Modelica. We show that our FMC is capable of performing detailed multi-domain design space exploration of realistic automotive architectures.

Keywords: Architecture; Design Space Exploration; Functional Modeling; Cyber-Physical Systems; Simulation

1. Introduction

Cyber-physical system (CPS) applications in automotive, aerospace, energy, and manufacturing are currently facing an unprecedented level of complexity due to the real-time interactions among hundreds to thousands of heterogeneous data processing components and physical processes (e.g. mechanical, electrical, thermal, chemical). Automotive companies (OEMs), for example, are at the forefront of CPS design due to the global competition for new features, heterogeneity of components, quality and development cost targets, and product launch dates. It has been estimated that more than 80% of the innovations in a car come from computer systems [1]. In premium cars, software alone represents an investment of about one billion dollars [2]. To minimize risk, automotive manufacturers develop architectures, or platforms, to allow component reusability across brands and car models [3]. Architecture-based design is an important process because it facilitates the selection of components from catalogues provided by the OEM's suppliers. Current market trends for green transportation are forcing automotive companies to come up with disruptive innovations that may require drastic changes in the existing architectures. Determining the impact of new design alternatives requires a very expensive and long redesign process that is not well supported by state-of-the-art CPS design tools.

Currently, CPS design is siloed into specific disciplines and supported by highly specialized but domain-specific model-based design (MBD) automation tools. For example, mechanical engineering is done in computer-aided design (CAD) and engineering (CAE) tools; electrical engineering is done in electronic design automation (EDA) and wire harness design tools; control engineering is done in Matlab/Simulink and Modelica; and software engineering is done in UML and in-house software development environments. Unfortunately, these detail design tools are often not compatible with each other and the data exchange between them is difficult or sometimes impossible. In order to better support the CPS design process, it is important to recognize that a single design is passed hundreds of times through hundreds of personnel from various organizations from the initial concept through the construction and product test. Therefore, we believe it is essential that the next-generation CPS design tools focus on supporting three key aspects: (i) meaningful evaluation of design choices as early as possible, (ii) system-level, cross-architecture, and multi-disciplinary analysis, and (iii) seamless transition from concept design to detail design.

This paper reports our experience in the development of a novel multi-disciplinary integrated design automation tool for cyber-physical systems. Our tool shows that the various disciplines in CPS design can be brought together to enhance the communication and requirements negotiation among engineers and organizations, enable multi-disciplinary simula-

doi:10.1016/j.procir.2014.03.183

tions to evaluate the system-level impact of domain-specific design decisions, and reduce the overall design cycle. Our concept design method relies on functional modeling to create a technology-independent description of *what the system does*, and uses a Functional Modeling Compiler (FMC) to synthesize technology-dependent solutions that can be directly used in multi-disciplinary simulations for validation and architectural design-space exploration. The contributions of this paper are:

- A new top-down architectural design space exploration approach that supports designers to broaden the design space during the creative phase, and helps them narrow the solution space to feasible solutions.
- A tool chain to allow a realistic and accurate comparison of different architectures and evaluate the system-level effects of architectural parameters.
- A demonstration of the feasibility of our idea using a realistic automotive use-case.

The paper is organized as follows. Section 2 motivates the use of and architecture-based design for CPS. Section 3 presents our Functional Modeling Compiler and the process on how functional models are allocated to various high-fidelity simulation models of various architectures. Section 4 provides an automotive use-case and demonstrates the main features of the FMC. Section 5 summarizes the paper and provides the direction of our future work.

2. Architectural Design Space Exploration of CPS

At the heart of *architecture* are the functional and physical structures of a CPS, and the allocation relations connecting them [4]. Architecture-based design[1] allows companies to streamline the development process of complex products across different organizations [6]. Automotive architectures (or platforms), for example, are estimated to save billions of dollars annually to companies because they allow reusability of components across different models and brands [3]. *Architectural parameters* are used to provide configurability to an architecture and are also relevant to evaluate its performance [7]. For example, the number of cylinders in an internal combustion engine architecture is an architectural parameter that designers vary to evaluate the performance, fuel consumption, emission rating, and handling of various vehicle architectures.

Architectural design spaces grow exponentially due to the many interacting parameters. Even with efficient simulation software available, performing a complete architectural design space exploration remains an open challenge due to the large number of simulations required to explore the space. In this paper we do not intend to automatically explore the architectural design space, but instead we focus on providing the tool chain to allow a realistic and accurate comparison of architectures and to allow the investigation of how diverse architectural parameters affect the system-level performance. During the creative stage of design, designers broaden the space of options and concepts; evaluation narrows this space down to a few feasible options. Our tool chain supports this creative process by first automating the generation of many options and concepts faster, and then accelerating the evaluations and the decisions made to find few feasible options.

CPS is currently an active area of research and recent publications highlight the idea of architecture-based CPS design. Figure 1(a) shows the multi-domain modeling of CPS using architectural views approach presented in [8]. This approach allows *existing* multi-domain models to be consistently related through an architecture as the unifying framework. The authors use typed graph matching algorithms [9] to create a base architecture from existing heterogeneous models (i.e. Simulink, Modelica, Finite State Process) and therefore we refer to it as *architectural bottom-up approach*. Rather than relying on existing models, our approach uses a *functional top-down approach* aided by a context-based synthesis algorithm [10] to generate multiple architectures and their corresponding simulation models from a unifying functional model as shown in Figure 1(b). A functional model is an embodiment-independent design rationale that describes *what the system does* [11]. Since there are multiple possible embodiments of the same functional model, different architectures can be expressed by the same functional model and this enables discipline-specific engineers to test their designs on multiple different architectural configurations. Coupled with the context-based synthesis, our method enables concurrent design because a high-level change to the functional model gives engineers the ability to generate new physical configurations of the system that can be used for the detailed design of controllers, software and hardware. This is possible because within a functional architecture, several physical configurations can be developed.

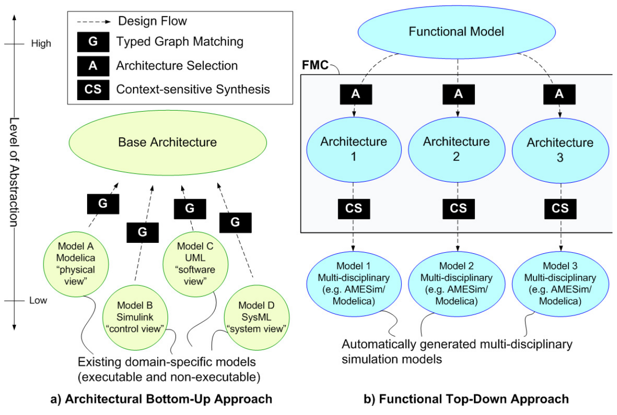
Design Flow
G Typed Graph Matching
A Architecture Selection
CS Context-sensitive Synthesis
High
Low
Level of Abstraction
Base Architecture
G
G
G
G
Model A Modelica "physical view"
Model B Simulink "control view"
Model C UML "software view"
Model D SysML "system view"
Existing domain-specific models (executable and non-executable)
a) Architectural Bottom-Up Approach
Functional Model
FMC
A
A
A
Architecture 1
Architecture 2
Architecture 3
CS
CS
CS
Model 1 Multi-disciplinary (e.g. AMESim/ Modelica)
Model 2 Multi-disciplinary (e.g. AMESim/ Modelica)
Model 3 Multi-disciplinary (e.g. AMESim/ Modelica)
Automatically generated multi-disciplinary simulation models
b) Functional Top-Down Approach

Fig. 1. In practice, the (a) architectural bottom-up approach [8] is suitable for existing workflows that attempt to reuse, as much as possible, the models created for similar products during the detail design phase. Our functional top-down approach (b) is better suited for next-generation workflows that begin during the concept design and allow architectural design space exploration of new designs created from scratch.

3. Functional Modeling Compiler

Our FMC leverages the technological advances in hybrid simulation languages such as AMESim [12] and Modelica [13] to generate multi-disciplinary simulation models to validate the interactions between the physical and the cyber components of a car. The commercial support of simulation component libraries written in these languages allows us to generate high-fidelity functional simulation models with components that have been validated by the vendors and are very close to the behavior of their real-life counterpart. These components include multi-physics components, and ECUs and control algorithms for internal combustion engines, automatic gearboxes, fuel cells, series- and parallel-hybrid cars. Figure 2 shows how the FMC is used to perform a design-space exploration of dif-

[1]This methodology when applied to electronics design is referred to as *platform-based design* [5]

ferent architectures by combining different simulation components (e.g. engines, ECUs, controllers, etc.) stored and classified in an architecture & simulation component repository. The input to the FMC is a functional model and it generates multi-disciplinary simulation models as an output[2].

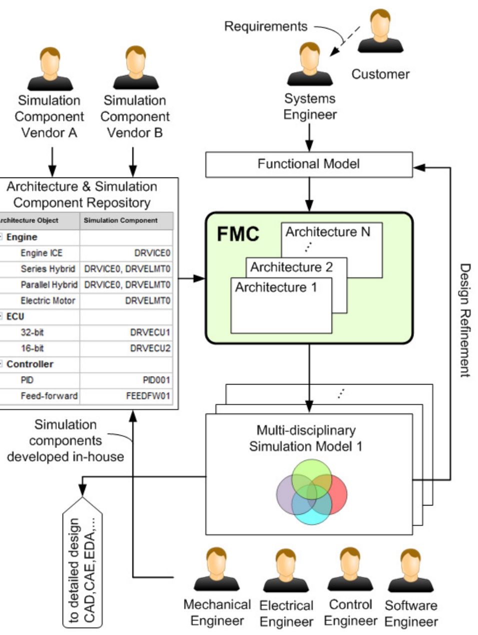

Fig. 2. FMC selects target architectures and allocates simulation components to functions in a functional model. The result are multi-disciplinary simulation models that domain-engineers can use during concept design to validate new components in various architectures.

3.1. Functional Modeling

Functional modeling is a concept design phase activity where the CPS product requirements are formally specified in terms of *functions*. A function describes *what the system does* in terms of energy, material, and signal transformations yet it remains technology independent. Our functional models are written in the Functional Basis [14], a high-level language close to natural language with well defined syntax and semantics to facilitate inter-disciplinary communication among engineers from different domains. For example, the function "`convert chemical energy to rotational mechanical energy`" implies the use of fuel (chemical energy) to generate torque (rotational mechanical energy) but does not specify whether it is a gasoline, gas, or diesel car with a four-stroke or a Wankel cycle. The Functional Basis taxonomy provides 32 elementary functions and 18 flow types that can be used to compose more complex functions.

In the existing practice, functional models are static documents or non-executable models used by the domain engineers to translate requirements into engineering specifications. Unfortunately, it is very difficult for computer-aided design tools from different disciplines to exchange data and take advantage of the model-based design at the system-level. To overcome the limitations of the current siloed CPS development, this paper presents a FMC capable of automating the allocation of functions to components and generating feasible multi-disciplinary simulation models to validate different architectures and various components (e.g. ECUs, transmissions, engines, etc.) at the system-level. Our FMC is intended to be used as a design-space exploration tool at the concept design phase to evaluate how different combinations of CPS components can be leveraged to achieve the functional and non-functional[3] requirements.

3.2. Architecture Selection

Architectures are often associated to mature products and architecture-based design is considered to be more appropriate for evolutionary designs. In [15], the authors argue that genuine product innovation is often associated with design from *first principles* and not with modular design methods such as architecture-based design. While this is a valid argument, we believe that architecture-based design is now and will continue to be the design methodology of choice for complex CPS. For this reason, we believe it is important to first develop new tools that support the development of products with *existing* architectures. Thus, in this paper we assume that at least one architecture is available for the FMC to map functions to simulation components.

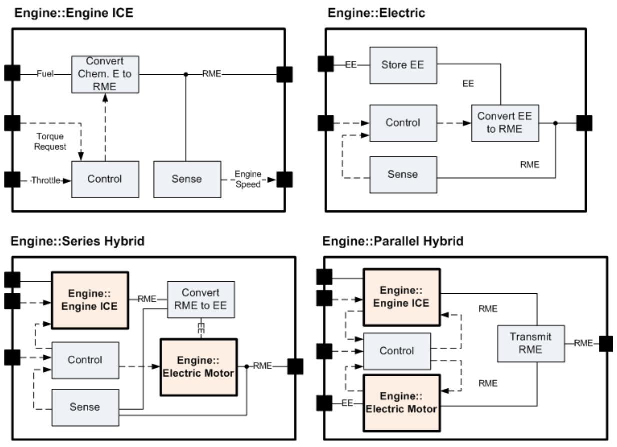

Fig. 3. Architecture description using a visual-based architecture description language (ADL). Architecture objects have well defined typed interfaces and are defined in terms of functions (blocks) and flows (arrows). Architecture object reusability is possible as shown in the hybrid architecture objects.

Architectures are typically defined using architecture description languages (ADL) [16]. In our system, the architecture information is stored in the Architecture and Simulation Component Repository. For example, Figure 2 shows that simulation components are organized according to root architecture objects including the `Engine`, `ECU`, and `Controller`. Architecture variability is expressed in each of the root architecture objects. For example, Figure 3 shows the four architecture alternatives for the `Engine` using a visual architecture description language. These objects inherit the properties of the architecture object `Engine` but each specializes a particular architectural choice such as internal combustion engine, electric motor, and series-hybrid and parallel-hybrid implementations of a car. Notice that architecture objects have typed ports and contain fragments of functional models with functions in-

[2]The details of our mapping algorithm that allocates components to functions according to a context-sensitive analysis can be found in [10].

[3]Non-functional requirements refer to performance characteristics of a car such as acceleration, noise-vibration-harshness (NVH), etc.

terconnected by flows. For example, the `Engine::Engine ICE` object contains "`Convert Chemical Energy to Rotational Mechanical Energy`", "`Control`" and "`Sense`" functions. Similarly, the `Engine::Electric Motor` object describes the design intent of having "`Convert EE to RME`" and "`Store EE`" functions. Our object oriented ADL is composable and it allows us to reuse architecture objects as seen in `Engine::Series Hybrid` and `Engine::Parallel Hybrid`, where each references the `Engine::Engine ICE` and `Engine::Electric Motor` components.

FMC selects the target architectures for which to generate simulation models by matching the functions in the input functional model to functions contained in the architecture objects. For example, an input functional model that describes a car with "`Convert Chemical Energy to Rotational Mechanical Energy`" matches three of the four architectures in Figure 3, namely `Engine::Engine ICE`, `Engine::Series Hybrid`, and `Engine::Parallel Hybrid`. Thus, FMC generates three simulation models as outputs corresponding to each one of the matched architectures. Generating simulation models for multiple architectures is useful for performing cross-architectural studies related to architectural parameters such as weight, fuel consumption, emissions, etc.

3.3. System-level Simulation Generation

Simulation has become standard practice during the development of complex systems because it is an economical and effective way to design, validate, and test the CPS early. Systems can be *virtually* explored and analyzed without the need for physical prototypes because software is used to estimate the dynamic behavior of the system under a large variety of conditions. This, in principle, enables the early identification of system-level problems. In practice, however, simulation has evolved independently on the different engineering domains and coupling different tools for a holistic analysis and multi-domain simulation is very complicated and most of the time not even possible.

CPS design requires a general cross-disciplinary simulation approach capable of characterizing how the design changes in one domain affect the rest of the domains in the system and the system-level behavior. When developing a heterogeneous complex system, coupling models and simulation of different engineering disciplines (e.g. mechanical, electrical, software) is necessary because it speeds up the development time, increases the understanding and communication, facilitates design and optimization, and enables virtual prototyping and verification. *Physical modeling* languages such as bond graphs [17], Modelica [13], and Simscape [18] have been developed for inter-disciplinary modeling and simulation of complex heterogeneous systems that describe the structure and the behavior of physical, control, and software systems. Physical modeling relies on the interconnection between physical *components* (i.e. resistor, pump, gear box) that encapsulate behavioral descriptions (mathematical models). Components interact through physical *ports* while honoring the energy conservation principles using the effort-flow variables such as voltage-current in electrical, temperature-heat flow in thermal, and angular velocity-torque in rotational mechanics domain. This modeling paradigm based on energy conservation principles couples various disciplines, allows the integration of user-defined disciplines, and facilitates model reusability. Because each component defines a behavioral model using energy conservation principles, physical models can be translated into compatible mathematical models and numerically solved in a holistic simulation.

Commercially available compilers and code generators transform the system-level simulation models into different executable models used for CPS validation and verification. For example, in *model-in-the-loop* simulation (MILS), the controller and the physical system are simulated sequentially in an interpreter that allows early validation of the control logic and the plant. Similarly, *software-in-the-loop* simulation (SILS) compiles the models and allows the validation of the behavior of the low level code executed in the controller (typically embedded C code). *Hardware-in-the-loop* simulation compiles the models and the controller code is executed in the actual embedded processor for timing and behavior validation.

Despite the available support tools, system-level simulation models are still manually created and maintained by experts making it a lengthy and error-prone process. In this paper, we show how functional models can be used to automatically generate system-level simulation models according to the latest engineering specifications. This not only eliminates the manual effort, but creates a more efficient concept design phase where engineers are allowed to try alternative designs at the functional level and obtain the corresponding system-level simulation models automatically. Although we target AMESim and Modelica language, our method is general to functional modeling and system simulation and can be easily adapted to other physical modeling languages.

Obtaining a system-level simulation from a functional model requires concrete architectural decisions that specify how functions are mapped to components [7]. In the FMC, these design guidelines are obtained from the architecture descriptions and the design intent provided in the functional model. We define *synthesis* as the *process by which a functional model describing what the system does is transformed into a concrete design implementation describing its structure (architecture) and its behavior (simulation model).* The most important aspect of our synthesis method is building a context in order to eliminate the ambiguity and variability associated to functional modeling. Functions-to-components are many-to-many relationships. For example, the function "`convert RME to TME`" in the second-level decomposition in Figure 4 can be mapped to two or four wheels depending on whether the architecture specifies whether the car is rear/front-wheel drive or four wheel drive.

It is important to note that the current implementation of the FMC focuses on generating simulation models that are compliant with the *intended behavior* expressed in the functional model. *Unintended behavior* is partially addressed; some AMESim components model unexpected and unintended behavior in the form of energy losses that are detrimental to their performance. This information is propagated back by the FMC in the form of *emergent functions* that create *refinements* of the original functional model in the following design cycle. These emergent functions serve two purposes: (1) inform the user about unexpected, unintended, or unfulfilled functions, and (2) refine the functional models. However, all phenomena cannot be represented and the need for verification [4] remains but is beyond the scope of automatically generated simulations.

4. Automotive Use-Case

A functional model of a car consisting of a two-level functional decomposition is shown in Figure 4. The top-level function "`transport people`" describes the main function of the "Car". It inputs three material flows (bold arrows for Air, People, and Fuel) and outputs two material flows (People and

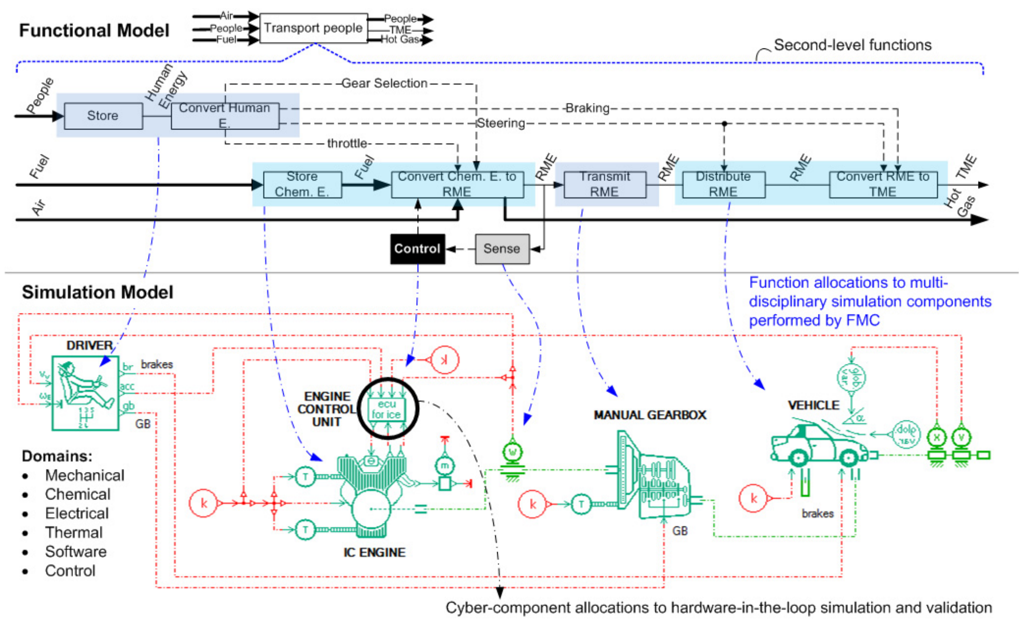

Fig. 4. Functional model associated with an internal combustion engine architecture and its corresponding simulation model generated by the FMC. This functional decomposition shows two levels and illustrates the embodiment-independent view of a system. Using this internal combustion engine architecture simulation model, designers can validate architectural parameters such as fuel consumption, and emissions.

Hot Gas) and one energy flow (arrow representing translational mechanical energy or TME). Notice that the second-level functions and flows that can be naturally mapped to the main subsystems of a car represented by architectural objects. For example, the function "convert chemical energy to rotational mechanical energy" function can be mapped to an Engine::Engine ICE, Engine::Series Hybrid, and Engine::Parallel Hybrid architecture objects as discussed in Section 3.2. Similarly, the function "transmit rotational mechanical energy" is mapped to a "Transmission::Manual" and not to a "Transmission::Automatic" because there is a "Gear Selection" control flow flowing from "convert human energy" function to "convert chemical energy to rotational mechanical energy" function. The shaded areas enclosing the functions in Figure 4 show how functions and flows are mapped to architectural objects, and how these architectural objects are allocated to AMESim simulation components to produce a simulation model of an internal combustion engine architecture. Figure 5 shows the generated simulation models after allocating functions to the Engine::Series Hybrid and Engine::Parallel Hybrid architecture objects.

Using functional models, the design intent remains technology-independent and allow non-experts to model the environment to design their subsystems. For example, this particular functional model was created by a software engineer with marginal understanding of mechanical engineering principles who wanted to design the engine control system (ECU and its controller software) represented by the "Sense" function (gray box) and the "Control" function (black-box). FMC simplifies the design process by allocating functions to high-fidelity simulation components with the help of architecture objects and creating a multi-disciplinary simulation models that domain engineers can use to design and validate their subsystems in software- and hardware-in-the-loop simulations (SILS, HILS). In this example, as it is shown in Figure 6, the effects

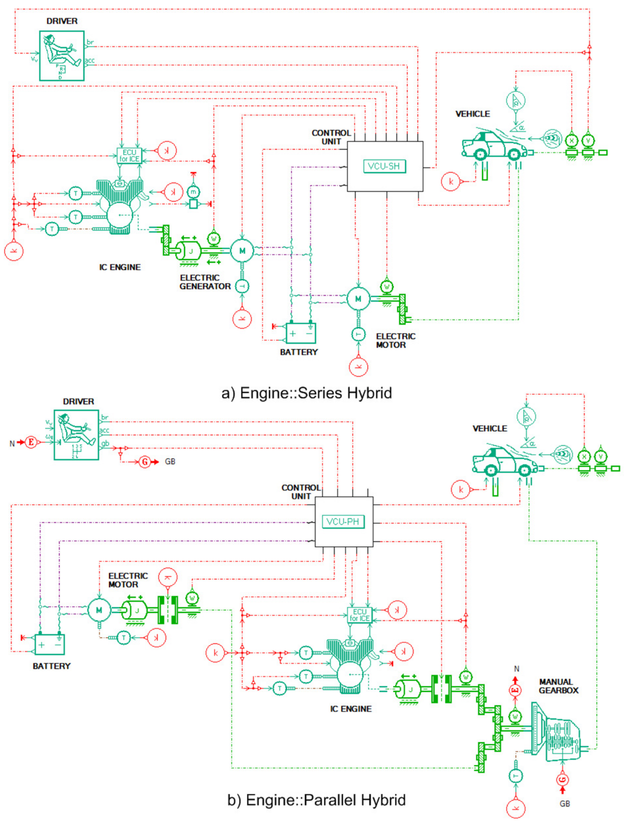

Fig. 5. Synthesized hybrid vehicle architectures from a functional model using the FMC and AMESim components.

of the ECU on fuel consumption, CO, HC, and NOx emissions can be tested and validated for the internal combustion engine architecture. Similar results can be obtained for the hybrid architectures.

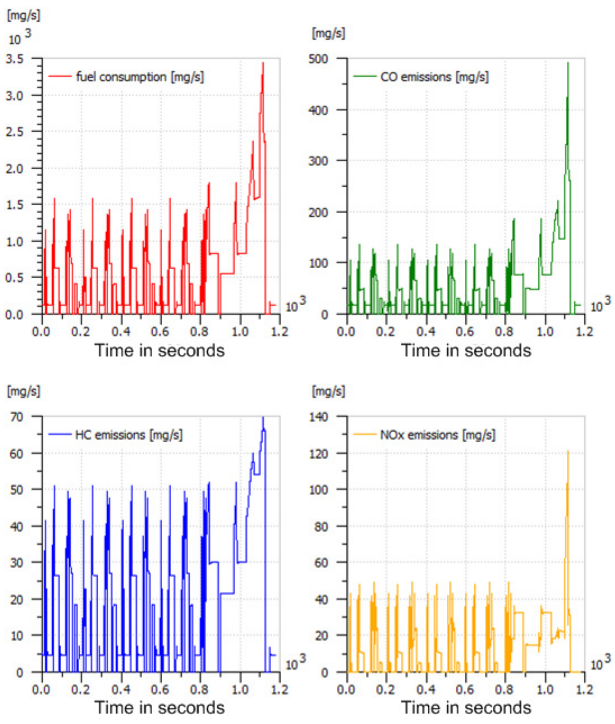

Fig. 6. Simulation results used to validate the effects of an ECU on fuel consumption, CO, HC, and NOx emissions of an internal combustion engine architecture.

Notice that with the help of the FMC, a software engineer can simply change the “Fuel” flow for an “Electrical Energy” flow in the functional model and automatically synthesize an electric vehicle simulation – as shown in Figure 7 – that, instead of a combustion engine, it includes a battery, an electric motor, and a suitable ECU to control these electric components. Figure 7 shows the validation of the ECU’s “torque command” control signals sent to the electric motor, and the motor behavior or “motor torque” accomplished with this controller. Currently, our FMC supports functional models written using the NIST Functional Basis [14] and synthesizes multi-disciplinary simulation models using AMESim and Modelica libraries.

5. Summary and Future Work

This paper highlights our research activities in developing the Functional Modeling Compiler, a new model-based design automation tool for the concept design of cyber-physical systems. The FMC synthesizes multi-disciplinary simulation models in AMESim and Modelica from functional information and architecture descriptions. Using an automotive use-case, we show that FMC enables embedded software engineers to automatically synthesize multiple vehicle architectures ready for SILS and HILS and validate new ECUs and control strategies for existing and new architectures (e.g. hybrid and electric vehicles). Because embedded software may be auto-generated from such simulation models using existing tools, we believe that high-level synthesis is a promising technology for the design of future complex CPS. In the future, we intend to extend our architectural design space exploration tool to explore and evaluate innovative designs of *new products not built upon existing architectures*. The open challenge is not only to synthesize simulation models but also to synthesize innovative architectures.

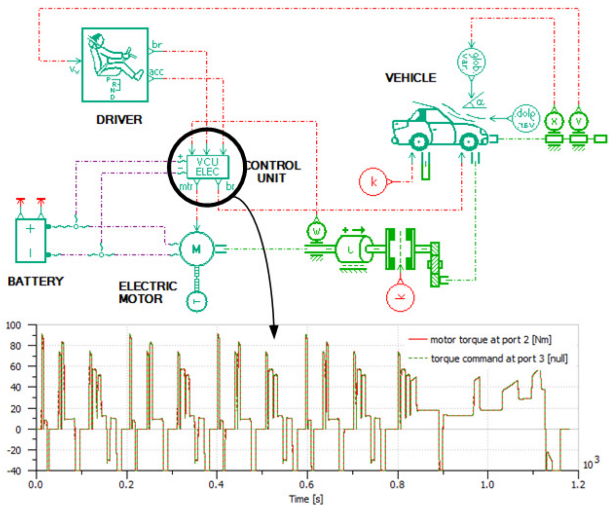

Fig. 7. Synthesized electric vehicle architecture simulation model from a functional model using the FMC and AMESim components.

References

[1] Broy, M., Krüger, I.H., Pretschner, A., Salzmann, C.. Engineering automotive software. Proc of the IEEE 2007;95(2).
[2] Charette, R.N.. This car runs on code. In: IEEE Spectrum. 2009,.
[3] Dahmus, J.B., Gonzalez-Zugasti, J.P., Otto, K.N.. Modular product architecture. Design Studies 2001;22(5):409 – 424.
[4] Kossiakoff, A., Sweet, W., Seymour, S., Biemer, S.. Systems Engineering Principles and Practice. 2nd ed.; Wiley; 2011.
[5] Sangiovanni-Vicentelli, A.. Defining platform-based design. In: EEDesign of EETimes. 2002,.
[6] Broy, M., Gleirscher, M., Kluge, P., Krezner, W., Merenda, S., Wild, D.. Automotive architecture framework: Towards a holistic and standardised system architecture description. Tech. Rep.; TUM; 2009.
[7] Bonnema, G.M.. Funkey architecting : an integrated approach to system architecting using functions, key drivers and system budgets. Ph.D. thesis; 2008.
[8] Bhave, A., Garlan, D., Krogh, B., Schmerl, B.. Multi-domain modeling of cyber-physical systems using architectural views. In: Proc. of the Embedded Real Time Software and Systems Conf. (ERTS 2010). 2010,.
[9] Bhave, A., Krogh, B., Garlan, D., Schmerl, B.. View consistency in architectures for cyber-physical systems. In: 2011 IEEE/ACM Intl. Conf. on Cyber-Physical Systems (ICCPS). 2011, p. 151 –160.
[10] Canedo, A., Schwarzenbach, E., Al Faruque, M.A.. Context-sensitive synthesis of executable functional models of cyber-physical systems. In: Proc. of the ACM/IEEE 4th Intl. Conf. on Cyber-Physical Systems. ICCPS ’13; 2013, p. 99–108.
[11] Erden, M., Komoto, H., Beek, T.V., V.D’Amelio, , Echavarria, E., Tomiyama, T.. A review of function modeling: approaches and applications. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 2008;22:147–169.
[12] LMS. AMESim. `http://www.lmsintl.com/`; 2013.
[13] Modelica. Modelica. `https://modelica.org/`; 2013.
[14] Hirtz, J., Stone, R.B., Szykman, S., McAdams, D.A., Wood, K.L.. A functional basis for engineering design: Reconciling and evolving previous efforts. Tech. Rep.; NIST; 2002.
[15] Barker, T., Kokotovich, V.. The impact of modular product design on innovation compared with design from first principles. In: Design Research Society (DRS) 2010. 2010,.
[16] EAST. EAST-ADL. `http://www.east-adl.info/`; 2013.
[17] Cellier, F.E.. Continuous System Modeling. Springer-Verlag; 1991.
[18] MathWorks, . Simscape. `http://www.mathworks.com/`; 2013.