Report from Dagstuhl Seminar 19502

Future Automotive HW/SW Platform Design

Edited by

Dirk Ziegenbein[1], Selma Saidi[2], Xiaobo Sharon Hu[3], and Sebastian Steinhorst[4]

1 **Robert Bosch GmbH – Stuttgart, DE,** dirk.ziegenbein@de.bosch.com
2 **TU Dortmund, DE,** selma.saidi@tu-dortmund.de
3 **University of Notre Dame, US,** shu@nd.edu
4 **TU München, DE,** sebastian.steinhorst@tum.de

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 19502 "Future Automotive HW/SW Platform Design". The goal of this seminar was to gather researchers and practitioners from academia and industry to discuss key industrial challenges, existing solutions and research directions in the design of future automotive HW/SW platforms, particularly focusing on predictability of systems regarding extra-functional properties, safe integration of hardware and software components and programmability and optimization of emerging heterogeneous platforms.

Seminar December 8–11, 2019 – http://www.dagstuhl.de/19502
2012 ACM Subject Classification Computer systems organization → Embedded software, Theory of computation → Models of computation, Software and its engineering → Real-time systems software
Keywords and phrases automotive, hw/sw platforms, real-time systems, systems design automation
Digital Object Identifier 10.4230/DagRep.9.12.28
Edited in cooperation with Lea Schönberger

1 Executive Summary

Dirk Ziegenbein (Robert Bosch GmbH – Stuttgart, DE)
Selma Saidi (TU Dortmund, DE)
Xiaobo Sharon Hu (University of Notre Dame, US)
Sebastian Steinhorst (TU München, DE)

Driven by new functionality and applications (such as automated driving and vehicle-to-X-connectivity) and fueled by the entry of new players from the IT industry, automotive systems are currently undergoing a radical shift in the way they are designed, implemented, and deployed. The trend towards automation and connectivity imposes an increased complexity and requires unprecedented computing resources, while, at the same time, the demanding requirements regarding cost-efficiency and dependability still need to be fulfilled. One of the most visible changes is the integration of formerly separated function domains onto centralized computing platforms. This leads to a heterogeneous mix of applications with different models of computation (e.g., control, stream processing, and cognition) on heterogeneous, specialized hardware platforms (comprising, e.g., application cores, safety cores, GPUs, deep learning

accelerators) to accommodate advanced functionalities such as automated driving and on-line optimization of operating strategies for electrified powertrains.

The adoption of these novel heterogeneous platforms raises several challenges. In particular, many of their components stem from embedded consumer devices and have never been designed for application in safety-critical real-time systems. Therefore, while their computational capabilities are well understood, there is an increased need to comprehend these platforms from the perspective of extra-functional requirements such as predictability, determinism, and freedom-from-interference. This process deeply impacts the core design aspects of automotive E/E architectures and heavily challenges established methods and methodologies in HW/SW automotive design.

The goal of this Dagstuhl Seminar was to gather researchers and practitioners from academia and industry to discuss key industrial challenges, existing solutions and research directions in the HW/SW design of future automotive platforms. The seminar focussed, in particular, on

- predictability of systems regarding extra-functional properties,
- safe integration of hardware and software components and
- programmability and optimization of emerging heterogeneous platforms.

These inter-dependent challenges require the interaction between multiple disciplines, combining resource-constrained embedded, cyber-physical, and real-time aspects. Another important aspect of the seminar was to provide insight into novel automotive functionalities (such as automated driving, online optimization, or over-the-air-update) and their software architectures and requirements as well as into the HW/SW platforms they are executed on.

The seminar provided a unique opportunity for participants from the automotive industry to present their challenges and constraints and receive feedback and ideas from academia. At the same time, it allowed researchers to confront their own ideas and/or solutions with industrial reality and together identify new research directions in order to make an impact in the automotive industry.

Organization of the seminar

The seminar took place from 8th to 11th December 2019. The seminar started with an overview of current trends and challenges in the design of future automotive HW/SW platforms by the organizers. After that the agenda was structured along the previously mentioned challenges. Monday's talk sessions were focused on dependability and predictability of HW/SW systems. The sessions on Tuesday dealt with the safe integration of heterogeneous software applications covering aspects of software architectures, networks and cyber-physical systems in the automotive domain and touched societal issues as well. On Wednesday, the talks focused on the programmability and optimization of heterogeneous platforms. All talks were restricted to 15 minutes, leaving ample time for discussions as well as breakout sessions on the following topics:

- Modeling hardware and software dependencies
- Weakly hard real-time models
- Machine learning in cyber-physical systems
- HW/SW architecture exchange
- Benchmarking efforts for future HW/SW platforms
- Modularizing control systems
- Automotive software lifecycle
- Programming vs. execution models

More details on breakout sessions are available in a dedicated section of this document, after the overview of the talks given during the seminar.

Outcome

The seminar succeeded in bringing together participants from different communities who were engaged in very intensive, interdisciplinary group discussions. Not surprisingly, many participants stated that they were able to learn a lot from adjacent fields. As many of the industrial challenges at hand require interdisplinary approaches, the organizers consider this a significant success of the seminar. One example that became evident during the course of the seminar was that terms like execution model are quite differently used in e.g. the high performance computing domain and in the embedded systems community. A group formed in one of the breakout sessions intends to write a whitepaper on unifying terminology and formulating a common understanding of the different layers of models used in designing automotive HW/SW systems. A first follow-up meeting already took place in February 2020.

Several industrial presentations gave valuable insights in the industrial state-of-the-practice and outlined challenges for future research. A very good example for this was the breakout session "HW/SW Architecture Exchange" which discussed current architectural patterns and open challenges in the context of designing dependable systems and achieving deterministic behavior on heterogeneous high-performance HW platforms.

Another breakout session provided an overview of current automotive benchmarks and performance models that can be used as a basis for research activities. This session also raised the awareness that industry needs to be more active in providing relevant benchmarks in order to enable researchers to validate the industrial viability of their solutions.

Overall, the feedback of the participants showed that they made a lot of new contacts in academia and industry and a follow-up seminar in about two years was requested by many participants. The seminar inspired several new collaborations including contributions to the Autonomous Systems Design workshop at DATE 2020, ideas for special sessions at DAC 2020 and ESWEEK 2020 and also a student project on automotive HW/SW platform simulation between a students' project group and an industrial partner.

2 Table of Contents

3 Overview of Talks

3.1 Towards a Contract Theory for Physical Systems

Bart Besselink (University of Groningen, NL)

Main reference Bart Besselink, Karl Henrik Johansson, Arjan van der Schaft: "Contracts as specifications for dynamical systems in driving variable form", in Proc. of the 18th European Control Conference, ECC 2019, Naples, Italy, June 25-28, 2019, pp. 263–268, IEEE, 2019.
URL http://dx.doi.org/10.23919/ECC.2019.8795736

Cyber-physical systems such as modern intelligent transportation systems generally comprise a large number of physical components connected through cyber elements for computation and communication. The large-scale, heterogeneous, and multi-disciplinary nature of these cyber-physical systems necessitates a theory for analysis, design, and control that is inherently modular, i.e., that allows for considering components independently. Whereas such theories exist for cyber elements in the form of contract theories, systematic approaches for physical components are lacking.

This talk will present some first results on a contract theory for physical systems by introducing assume/guarantee contracts for differential equation models of dynamical systems. The use of contracts as component specifications will enable modular design while providing guarantees on the behavior of the integrated system. The proposed framework is illustrated by application to an adaptive cruise control system and provides a first step towards a contract theory for cyber-physical systems.

3.2 Predictable Heterogeneous Computing for Next-generation Cyber-Physical Systems

Alessandro Biondi (Sant'Anna School of Advanced Studies – Pisa, IT)

Joint work of Alessandro Biondi, Marco Pagani, Francesco Restuccia, Giorgiomaria Cicero, Biruk Seyoum, Mauro Marinoni, Giorgio Buttazzo

Motivated by the increasing demand of high-performance computational capabilities to implement safety-critical functionality in next-generation automotive systems, this talk presents a series of research findings to enable predictability on heterogeneous computing platforms that integrate both asymmetric multiprocessors and hardware accelerators. The talk makes an opinionated point: FPGA-based system-on-chips are far more prone to predictability than other heterogeneous platforms. Such platforms have been found to particularly prone to enable time-predictable hardware acceleration (thanks to limited fluctuations of execution times and the possibility to control the bus/memory traffic with custom logic) and to design cost- and resource-efficient systems thanks to FPGA area virtualization. Such platforms are also sufficiently open to realize SW-based isolation mechanisms for SW tasks running on multiprocessors.

3.3 The Role of Programming Abstractions in Automotive Software

Jerónimo Castrillón-Mazo (TU Dresden, DE)

Digitalization and the new golden age for computer architecture pose great challenges for automotive software, as ever increasingly complex applications must safely execute on emerging computing systems. This talk argues for building strong semantics whenever possible in software platforms in order to keep this complexity under control. This is in contrast to more mainstream software approaches that, understandingly, focus on quickly providing functionality in a rather ad-hoc manner without a formal underlying execution model.

Formal models of computation represent a promising formalism to soundly frame automotive software design. Dataflow [1], for example, is a proven model that has enabled lots of methodologies for software and hardware synthesis [2]. Recently, methodologies have been extended to cope with the adaptivity required in embedded systems in general, and in automotive software in particular. By formalizing the structure of target parallel architectures and exploiting the semantics and the structure of dataflow programs [3], applications can be remapped dynamically at runtime while displaying superior time predictability [4]. The same formal properties of the model enable runtime decision-making to improve the energy efficiency of the overall system when multiple dataflow applications share resources [5]. All these methodologies and optimizations are possible only thanks to the properties of the underlying model of computation.

Adaptive extensions to dataflow methodologies are however not sufficient for automotive applications. Additional semantics are needed to *react* to external (sensory) inputs and to capture the inherent time dimension of cyber physical systems. Reactors [6] is a novel model that supports both reactive and timed behavior, enabling time determinism in a way similar to System-level LET [7]. We show how the reactor model can be enforced on top of the AUTOSAR Adaptive Platform via APIs, while still adhering to the standard. This way, we remove timing errors in the delivery and processing of frames in a video test application meant for automatic breaking [8]. In future work, we expect to be able to perform similar analysis and optimizations to these kinds of applications as done in the past for dataflow applications. Better language and platform-level support would also improve the programming experience.

Moving forward, extreme heterogeneity in the form of accelerators, novel post-CMOS fabrics and emerging memories will further complicate the process of mapping applications to hardware. Formal models and domain specific languages (DSLs) will become even more important to deal with such future systems, accompanied with software programming stacks that allow passing information up and down the stack [9, 10]. A tensor expression language [11], for example, could be used to pass information about required operations and a runtime resource negotiator could determine whether to provide access to acceleration in a tensor processing unit. Such tensor abstractions, made popular thanks to the machine learning boom, have already proven useful to optimize memory layout for emerging memory architectures in [12].

The golden age of computer architecture brings along a golden age for research on programming abstractions and software platforms. There is still a lot of work ahead, including efforts to refine the models to better match real automotive applications, to seamlessly integrate DSLs, and to improve information exchange within layers of the software platform.

References

1 J. B. Dennis. *First Version of a Data Flow Procedure Language.* In Programming Symposium, pages 362–376. Springer, 1974.

2 J. Castrillon and R. Leupers. *Programming Heterogeneous MPSoCs: Tool Flows to Close the Software Productivity Gap.* Springer, 2014.

3 A. Goens, S. Siccha, and J. Castrillon. *Symmetry in Software Synthesis.* ACM Transactions on Architecture and Code Optimization (TACO), 14(2):20:1–20:26, July 2017.

4 A. Goens, R. Khasanov, M. Hähnel, T. Smejkal, H. Härtig, and J. Castrillon. *Tetris: A Multi-application Run-time System for Predictable Execution of Static Mappings.* In Proceedings of the 20th International Workshop on Software and Compilers for Embedded Systems (SCOPES'17), SCOPES '17, pages 11–20, New York, NY, USA, June 2017. ACM.

5 R. Khasanov and J. Castrillon. *Energy-efficient Runtime Resource Management for Adaptable Multi-application Mapping.* In Proceedings of the 2020 Design, Automation and Test in Europe Conference (DATE), DATE '20. EDA Consortium, March 2020.

6 M. Lohstroh, Í. Romero, A. Goens, P. Derler, J. Castrillon, E. A. Lee, and A. Sangiovanni-Vincentelli. *Reactors: A Deterministic Model for Composable Reactive Systems.* In Proceedings of the 9th Workshop on Design, Modeling and Evaluation of Cyber Physical Systems (CyPhy 2019) and the Workshop on Embedded and Cyber-Physical Systems Education (WESE 2019), page 26pp, October 2019.

7 R. Ernst, L. Ahrendts, and K.-B. Gemlau. *System Level LET: Mastering Cause-effect Chains in Distributed Systems.* In IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society, pages 4084–4089. IEEE, 2018.

8 C. Menard, A. Goens, M. Lohstroh, and J. Castrillon. *Achieving Determinism in Adaptive AUTOSAR.* In Proceedings of the 2020 Design, Automation and Test in Europe Conference (DATE), DATE '20. EDA Consortium, March 2020.

9 J. Castrillon, M. Lieber, S. Klüppelholz, M. Völp, N. Asmussen, U. Assmann, F. Baader, C. Baier, G. Fettweis, J. Fröhlich, A. Goens, S. Haas, D. Habich, H. Härtig, M. Hasler, I. Huismann, T. Karnagel, S. Karol, A. Kumar, W. Lehner, L. Leuschner, S. Ling, S. Märcker, C. Menard, J. Mey, W. Nagel, B. Nöthen, R. Peñaloza, M. Raitza, J. Stiller, A. Ungethüm, A. Voigt, and S. Wunderlich. *A Hardware/Software Stack for Heterogeneous Systems.* IEEE Transactions on Multi-Scale Computing Systems, 4(3):243–259, July 2018.

10 G. Fettweis, M. Dörpinghaus, J. Castrillon, A. Kumar, C. Baier, K. Bock, F. Ellinger, A. Fery, F. H. P. Fitzek, H. Härtig, K. Jamshidi, T. Kissinger, W. Lehner, M. Mertig, W. E. Nagel, G. T. Nguyen, D. Plettemeier, M. Schröter, and T. Strufe. *Architecture and Advanced Electronics Pathways Towards Highly Adaptive Energy-efficient Computing.* Proceedings of the IEEE, 107(1):204–231, January 2019.

11 N. A. Rink, I. Huismann, A. Susungi, J. Castrillon, J. Stiller, J. Fröhlich, and C. Tadonki. *CFDlang: High-level Code Generation for High-order Methods in Fluid Dynamics.* In Proceedings of the 3rd International Workshop on Real World Domain Specific Languages (RWDSL 2018), RWDSL2018, pages 5:1–5:10, New York, NY, USA, February 2018. ACM.

12 A. A. Khan, N. A. Rink, F. Hameed, and J. Castrillon. *Optimizing Tensor Contractions for Embedded Devices with Racetrack Memory Scratch-pads.* In Proceedings of the 20th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES), LCTES 2019, pages 5–18, New York, NY, USA, June 2019. ACM.

3.4 Mixed Criticality Communication in Future In-Vehicle Architectures

Lulu Chan (NXP Semiconductors – Eindhoven, NL)

The autonomy and connectivity trends in the automotive industry lead to an increase in the number of functions and the amount of associated data in a vehicle. This talk presents an overview of the resulting challenges for automotive in-vehicle networks to help address these industry trends. This overview includes among others different architecture concepts, function mappings, and sharing of resources. Especially given the latter, there are challenges regarding how to ensure that the in-vehicle data communication is fast, prioritized, service oriented, safe and secure.

We review several solution approaches being discussed, such as recent TSN standards and the adoption of service oriented architectures.

The purpose of this talk is to trigger discussions on what the challenges and current solutions mean for future research directions.

3.5 Predictable and Reliable Automated Transportation Systems

Thidapat Chantem (Virginia Polytechnic Institute – Arlington, US)

Joint work of Thidapat Chantem, Ali Al-Hashimi, Pratham Oza, Thidapat Chantem, Ryan Gerdes
Main reference Ali Al-Hashimi, Pratham Oza, Ryan M. Gerdes, Thidapat Chantem: "The Disbanding Attack: Exploiting Human-in-the-Loop Control in Vehicular Platooning", in Proc. of the Security and Privacy in Communication Networks – 15th EAI International Conference, SecureComm 2019, Orlando, FL, USA, October 23-25, 2019, Proceedings, Part II, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol. 305, pp. 163–183, Springer, 2019.
URL http://dx.doi.org/10.1007/978-3-030-37231-6_9

As self-driving cars are becoming a reality, the dependability and reliability concerns for such systems lie at the forefront and must be addressed before a large-scale adoption can be expected. Up to now, we have relied on human operators to act as a "fail-safe" option, i.e., using human intervention to make critical decisions, when possible. However, systems have become so advanced and sophisticated that human intervention may in fact be detrimental. We will discuss such a case study in this talk, and to brainstorm ideas on design principles for the realization of safe systems of the future.

3.6 Specification-driven Design and Analysis for Perception, Decision-Making and Control in Autonomous Systems

Jyotirmoy Deshmukh (USC – Los Angeles, US)

Joint work of Adel Dokhanchi, Heni Ben Amor, Georgios Fainekos, Anand Balakrishnan, Aniruddh G. Puranic, Xin Qin

We argue that we can use partial specifications for components in self-driving cars to help reason about correctness and safety. We review the use of Signal Temporal Logic (STL) to express control-theoretic properties. We show how STL can be extended to a new logic called Timed Quality Temporal Logic (TQTL) that allows capturing sanity conditions on the results of perception. Such sanity conditions essentially encode geometric relations between objects detected in individual perception frames as well as physics-based laws governing how the representations of detected objects are temporally related across frames. We show [1, 2] how we can monitor such conditions on the outputs of state-of-the-art convolutional neural networks used in self-driving applications and discuss some results showing violations of such sanity conditions by such object detection CNNs.

References

1 A. Dokhanchi, H. B. Amor, J. V. Deshmukh, and G. Fainekos. *Evaluating perception systems for autonomous vehicles using quality temporal logic.* In International Conference on Runtime Verification, pp. 409-416. Springer, Cham, 2018.
2 A. Balakrishnan, A. G. Puranic, X. Qin, A. Dokhanchi, J. V. Deshmukh, G. Fainekos, *Specifying and Evaluating Quality Metrics for Vision-based Perception Systems.* In 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE) (pp. 1433-1438). IEEE.

3.7 Predictable Low-latency Data Services for Critical Applications – Challenges and Concepts

Rolf Ernst (TU Braunschweig, DE)

Joint work of Rolf Ernst, Leonie Köhler, Mischa Möstl, Kai Gemlau, Jonas Peeck
Main reference Rolf Ernst, Leonie Ahrendts, Kai Bjorn Gemlau: "System Level LET: Mastering Cause-Effect Chains in Distributed Systems", in Proc. of the IECON 2018 – 44th Annual Conference of the IEEE Industrial Electronics Society, Washington, DC, USA, October 21-23, 2018, pp. 4084–4089, IEEE, 2018.
URL http://dx.doi.org/10.1109/IECON.2018.8591550

Given the recent trend towards service oriented, data centric architectures at higher safety and real-time requirements, it is about time to re-evaluate the efficiency and appropriateness of the current in-vehicle communication. Using DDS as an example, the talk will highlight the upcoming challenges and discuss new concepts which combine all levels of a communication stack for superior efficiency. It will introduce a timed distributed programming paradigm that enables low-latency predictable IP communication stacks. The approach can be combined with application-level error handling to derive protected low latency object communication. The talk gave results and runtime measurements and outlined possible approaches to worst case end-to-end response time analysis.

3.8 Security and Correctness in the Face of Self-Adaptive Learning Automotive Systems

Sabine Glesner (TU Berlin, DE)

Automotive systems have changed in many ways during the last years. Not only their size has increased dramatically, also their interconnectedness has grown rapidly. Moreover, learning and adaptive components are finding their way into automotive systems. While it is well-known that safety is important in the design and implementation of automotive systems, nowadays also security has become a major issue. Since components are connected with each other and also with networks outside the car, intruders might find ways to corrupt the correct functioning of automotive systems. On top of this, learning and adaptive systems might change the functionality of the overall systems in unexpected ways. It is an open question what correctness means in the face of self-adaptive learning systems and how security can be ensured. In this talk, I will discuss these questions.

3.9 Aggregation and Integration of Next-generation Vehicle Computing and the OS Technologies

Masaki Gondo (eSOL – Tokyo, JP)

As the C.A.S.E, namely Connected, Autonomous, Sharing/Service, and Electric have become the driving force of the vehicle design, the E/E architecture design will also inevitably change. The design shift challenges the current highly distributed vehicle computing architecture, as the software needs to be aggregated and integrated to the level that is never imagined before. The software platform such as AUTOSAR or ROS with its underlying OS will be the key to realize a high-performance but energy-efficient, deterministic, cost-effective safe system. This talk will provide insights into the software challenges and the role OS plays to tackle the E/E architecture design shift.

3.10 Automotive Edge Computing Use-cases Inspired by Societal Problems

Baik Hoh (Toyota Motors North America – Mountain View, US) and Seyhan Uçar (Toyota Motors North America – Mountain View, US)

Joint work of Baik Hoh, Seyhan Uçar, K. Oguchi
Main reference Seyhan Uçar, Baik Hoh, K. Oguchi: "Management of Anomalous Driving Behavior". 2019 IEEE Vehicular Networking Conference, Los Angeles, USA, December 4-6, 2019.

Automotive HW/SW platform will soon undergo a radical shift to support future mobility trends, being connected, shared, electrified, and automated. Specifically, automakers have put an attention on connected cars for a sizable amount of time but surprisingly its definition

still remains vague. Customers simply put CarPlay (or Android Auto) to a connected car while automotive industry engineers have struggled in realizing vehicles being connected to nearby vehicles, roadside unit, cloud, and nowadays edge. What makes it even worse is that every stakeholder perceives "edge" in a different way.

All these confusions must have been avoided if not a very product-oriented mindset (i.e., what technology can do) but a customer-centric approach (i.e., what future mobility and urban dwellers would need) called upon a connectivity and edge computing. Being motivated by key societal problems in urban mobility, Dr. Hoh and Dr. Uçar will share a few compelling use cases and insights in the first half. The second half will highlight the roles of emerging edge components to support the use cases. The industry inspired use case presentation would justify why vehicles should be connected to edge components. Also, the presented novel automotive functionalities would motivate an academic research community to include the emerging edge components in the equation of system verification and validation.

3.11 Paving the Way Towards Predictable Performance in Multi-heterogeneous SoC, Industrial Problems and Directions

Ignacio Sañudo (University of Modena, IT)

Joint work of Ignacio Sañudo, Marco Solieri, Micaela Verucchi, Marko Bertogna

Modern embedded platforms designed for automotive applications feature high-performance multi-core CPUs tightly integrated with compute accelerators. The spectrum of variety for such accelerators ranges from re-configurable devices and Graphic Processor Units for general purpose computing to Application Specific Integrated Circuits.

Many challenges have emerged with the integration of such technologies in the automotive domain. In this talk, novel solutions for improving the predictability of the future automotive hardware and software have been provided. For instance: (i) obscure scheduling mechanisms employed on GPUs impede deriving a proper timing analysis for applications. Understanding the scheduling decisions of modern GPUs is necessary to allow real-time engineers to safely model the performance of these accelerators – (paper under revision process) -. (ii) shared resources (such as caches and system RAM) allow latency depends on contention. Accurate memory-centric scheduling mechanisms for guaranteeing prioritized memory accesses to Real-Time components of the system might be a plausible solution to solve such problem [1]. (iii) engine heterogeneity brings another degree of complexity in the task partitioning problem. The Directed Acyclic Graph (DAG) model is appropriate to express the complexity and the parallelism of these tasks. Different aspects can be considered to convert a multi-rate DAG task-set with timing constraints into a single-rate DAG that optimizes schedulability, age and reaction latency, by inserting suitable synchronization constructs – (paper under revision process) –.

References

1 T. Kloda, M. Solieri, R. Mancuso, N. Capodieci, P. Valente, and M. Bertogna. *Deterministic Memory Hierarchy and Virtualization for Modern Multi-Core Embedded Systems.* In 2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS).

3.12 Possibilities Using FMI-based Co-simulation for the Validation of Cyber-Physical Systems

Peter Gorm Larsen (Aarhus University, DK)

Joint work of Peter Gorm Larsen, John Fitzgerald
Main reference Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, Hans Vangheluwe: "Co-Simulation: A Survey", ACM Comput. Surv., Vol. 51(3), pp. 49:1–49:33, 2018.
URL http://dx.doi.org/10.1145/3179993
Main reference John S. Fitzgerald, Peter Gorm Larsen, Ken G. Pierce: "Multi-modelling and Co-simulation in the Engineering of Cyber-Physical Systems: Towards the Digital Twin", in Proc. of the From Software Engineering to Formal Methods and Tools, and Back – Essays Dedicated to Stefania Gnesi on the Occasion of Her 65th Birthday, Lecture Notes in Computer Science, Vol. 11865, pp. 40–55, Springer, 2019.
URL http://dx.doi.org/10.1007/978-3-030-30985-5_4
Main reference Casper Thule, Kenneth Lausdahl, Cláudio Gomes, Gerd Meisl, Peter Gorm Larsen: "Maestro: The INTO-CPS co-simulation framework", Simul. Model. Pract. Theory, Vol. 92, pp. 45–61, 2019.
URL http://dx.doi.org/10.1016/j.simpat.2018.12.005

The Functional Mockup Interface (FMI) has been invented in order to enable an open way to combine the simulation of different independent simulation units (FMUs) representing both cyber as well as physical parts of a Cyber-Physical System (CPS). The kay characteristics is that FMUs can be used in a black-box context such that suppliers are not revealing the IP captured in the model such FMUs are generated from. This presentation will demonstrate an open FMI-based tool chain called INTO-CPS that currently is supporting version 2.0 of the FMI standard, but where the team behind this also are heavily involved in the new version(s) of FMI attempting to ensure an even better support in the future.

3.13 Domain Controllers, Autonomous Driving and Functional Safety, Oh My!

Mark Lawford (McMaster University – Hamilton, CA)

Joint work of Victor Bandur, Vera Pantelic, Gehan Selim, Zinoviy Diskin, Nick Annable, Aalan Wassyng, Tom Maibaum, Asim Shah, and Spencer Deevy

With the advent of hybrid electric drive trains, advanced driver assistance (ADAS) and autonomous driving features being enabled by software, there was been a proliferation of Electronic Control Units and their software in modern vehicles. These new features together with existing automotive requirements for extensive product lines and regional requirements has resulted in an unsustainable growth of software development and safety costs [1]. We have seen in our research that OEMs are at the point where software development groups are required to produce a new release almost every workday to meet these needs.

How could this have happened? Let's consider a contrived example. In a hybrid electric vehicle you might currently have multiple Electronic Control Units (ECUs) connected via a Controller Area Network bus (CANbus) to implement the hybrid electric drive train functionality: the Engine Controller for the internal combustion engine, the Transmission Controller, possibly multiple (electric) motor controllers, a battery starter generator, a battery management unit, the body controller providing driver interfacing and a hybrid controller that orchestrates and optimizes the functioning of the power train. In the United States there is a regulatory requirement that the vehicle shall not roll away when keyed off

and requiring the brake to be depressed before being able to move the vehicle out of park [2]. An OEM has a hybrid electric vehicle that meets this standard but now wants to sell the vehicle in South Korea where there a automated parking garages when unattended vehicles are moved around by automation so perhaps there is a market requirement that vehicle can be shut off and the keyed remove when in neutral so that the vehicle can be rolled for parking garages. Which ECUs require software change for our South Korean variant of the hybrid powertrain? Most likely the body control module that deals with the shifter and key interface and perhaps the transmission controller. We might also have some follow on functional safety change to the battery management unit to allow the car to be charged when the transmission is in neutral. There may also be some changes to software for the electric motor controls or battery starter generator. Thus a simple requirements change possibly requires 3 or more software releases (one for each affected controller).

One solution to this release proliferation problem is to go to a centralized architecture with new high speed networking technologies. In this case the hybrid controller implements the main power train functionality and the other systems effectively become smart actuators that respond to commands from the hybrid controller (eg. “up shift now” to the transmission controller, “provide this much torque” to electric motor controller. The hope is that when there is a requirements change only the hybrid controller as the power train domain controller would require a software update while the smart actuator ECUs would be left untouched.

Such a change in the electrical and electronic architecture of the vehicle raises some interesting questions:

- What are the functional safety implications of moving to a centralized architecture?
- How might the development and safety processes change to deal with them?
- What research problems must be solved to make these systems safe and cost effective, especially when the systems integrate machine learning components?
- How can virtualization be used to meet the requirements of running mixed criticallity software on a single powerful ECU?

In [3] we provide a survey of centralized automotive architecture research and development and discuss these research questions.

A large part of the cost of software intensive automotive system development is the effort required for functional safety and safety of the intended function. With so many vehicle variants across at typical OEM’s product line it is impossible to create a separate complete safety case for each version of a vehicle. Improper reuse of safety related design artifacts can result in a vehicle recall so safety engineers currently are faced with the difficult and labour intensive task of determining what design artifacts need to be revised for a new vehicle design. Preliminary work on the use of model management techniques on explicit safety cases [4] has been proposed as a a way to help automate this process. The downfall of this technique is that assurance case formalisms such as Goal Structured Notation (GSN) lack sufficient detail of the dependencies of the evidence and how it is generated. In [5] and [6] we propose explicit modeling of the development and safety processes to provide fine grained traceability of safety artifacts that can allow model based incremental safety analysis that takes into account not only the safety artifacts but how they are produced. Some sort of rigorous, tool supported model based incremental safety analysis is going to be required for safe cost effective software development as systems become more complex and incorporate autonomous driving features. In addition standardized safety case templates for a given system is going to be required.

Autonomous vehicle architectures have typically implemented some form of the sense-plan-do framework with safety elements often integrated into the main architecture (e.g. [7]). While there have been works advocating for separation of safety and control [8], such works have not utilized well accepted safety patterns such as the 3-level safety of the eGas standard for throttle by wire [9]. We discuss our exploration of implementing this design pattern in autonomous vehicle safety architecture in [10], and in [11] provides a detailed exploration of safety architectures and associated assurance case templates for Level 5 autonomous vehicle.

References

1. R. Lancot. *Software is eating the auto industry.* Strategy CI Blog, Strategy Analytics, August 17, 2017. https://www.strategyanalytics.com/strategy-analytics/blogs/devices/strategic-ci/infotainment-telematics/2017/08/25/software-is-eating-the-auto-industry
2. NHTSA. Federal Motor Vehicle Safety Standard No. 114, Theft Protection and Rollaway Prevention, 74 FR 42837, National Highway Traffic Safety Administration, 2009.
3. V. Bandur, V. Pantelic, G. Selim, M. Lawford. *Towards Centralized E/E Vehicular Architectures.* Technical Report, McMaster Centre for Software Certification, (In preparation.).
4. S. Kokaly, R. Salay, M. Chechik, M. Lawford, and T. Maibaum. *Safety Case Impact Assessment in Automotive Software Systems: An Improved Model-based Approach.* In International Conference on Computer Safety, Reliability, and Security, pp. 69-85. Springer, Cham, 2017.
5. Z. Diskin, T. Maibaum, A. Wassyng, S. Wynn-Williams, and M. Lawford. *Assurance via Model Transformations and Their Hierarchical Refinement.* In Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, pages 426-436, ACM, 2018.
6. Z. Diskin, N. Annable, A. Wassyng, and M. Lawford. *Assurance via Workflow+ Modelling and Conformance.* arXiv preprint arXiv:1912.09912, 2019.
7. K. P. Divakarla, A. Emadi, and S. Razavi. *A Cognitive Advanced Driver Assistance Systems Architecture for Autonomous-capable Electrified Vehicles.* In IEEE Transactions on Transportation Electrification **5**(1), pages 48–58, Mar. 2019.
8. C. B. Santos Tancredi Molina et al. *Assuring Fully Autonomous Vehicles Safety by Design: The Autonomous Vehicle Control (AVC) Module Strategy.* In 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W). Denver, CO, USA, pages 16-21, June 2017.
9. E-GAS Workgroup. Standardized E-Gas Monitoring Concept for Gasoline and Diesel Engine Control Units. Tech. rep. Version 6.0. July 2015.
10. S. A. Shah. *Safe-AV: A Fault Tolerant Safety Architecture for Autonomous Vehicles*, M.A.Sc. Thesis, Department of Computing and Software, McMaster University, Hamilton, ON, Canada, 2019.
11. S. Deevy. *Sentinel: Fault-Tolerant Software Architecture for Safe Artificial Intelligence in SAE J3016 Level 5 Autonomous Vehicles*, M.A.Sc. Thesis, Department of Computing and Software, McMaster, 2019.

3.14 Formal Verification on Finite-State Machines with Weakly-Hard Fault Models

Chung-Wei Lin (National Taiwan University – Taipei, TW)

Joint work of Chung-Wei Lin, Kai-Chieh Chang, Chao Huang, Shih-Lun Wu, and Qi Zhu

A weakly-hard fault model can be captured by an (m, k) constraint, where $0 \leq m \leq$ k, meaning that there are at most m bad events (faults) among any k consecutive events. We use a weakly-hard fault model to constrain the occurrences of faults. Given a constant K, we verify the system safety for all possible pairs of (m, k), where $k \leq K$, in an exact and efficient manner. By verifying all possible pairs of (m, k), we can analyze the optimal strategy to maximize resource saving (for system designers) or minimize attacking cost (for attackers). We believe that this is just an initial step, and many open problems will follow.

3.15 Sorry Software – Hardware Matters for Dependability

Albrecht Mayer (Infineon Technologies – München, DE)

Automated driving demands new levels of dependability (e.g. fail operational) and compute performance. Software complexity is traditionally addressed with approaches resulting in more memory and performance demands. But many software transparent hardware performance features contradict with predictability, dependability, security and hard real-time. High performance and high dependability at the same time can only be achieved with a carefully chosen system architecture, considering hardware failure rates, detection rates and recovery capabilities. The same is true for software which is usually even dominant in terms of failure rates. In this talk different compute components and system architectures will be assessed concerning their capabilities to deliver high dependable performance for hard real time systems.

3.16 Safe and Secure Software Platforms for Autonomous Driving

Philipp Mundhenk (Autonomous Intelligent Driving GmbH – München, DE)

URL http://www.aid-driving.eu

The vastly increasing amount of software in vehicles, its variability, as well as the computational requirements, especially for those built with autonomous driving in mind, require new approaches to the structure and integration of software. The traditional approaches of single-purpose embedded devices with integrated software are no longer a suitable choice. New architectures introduce general purpose compute devices, capable of high-performance computation, as well as high variability of software. Managing the increasing complexity, also at runtime, in a safe and secure manner, are open challenges. Solving these challenges is a high-complexity development and integration effort requiring design-time and runtime

configuration, approaches to communication middleware, operating system configuration, such as task scheduling, monitoring, tight integration of security and safety, and, especially in the case of autonomous driving, concepts for dynamic adaption of the system to the situation, e.g., fail-operational concepts.

3.17 Design-For-Safety for Automotive IC Design: Challenges And Opportunities

Alessandra Nardi (Cadence – San Jose, US)

The autonomous driving revolution is introducing a whole new level of complexity in semiconductor design, not only in terms of computational/feature requirements but also because safety critical applications have very stringent demands on Functional Safety. However, the full automation supported in EDA (Electronic Design Automation) tools for traditional metrics has not yet reached maturity for the new safety metrics and is a green field of innovation. Additionally, the methodology and the criteria accepted for safety sign-off are still in evolution and in discussion in the industry. Even the language for capturing and exchanging the safety intent is yet to be fully defined. In this talk, we discuss some requirements to Design-For-Safety and some of the challenges and opportunities for flow automation that are presented to the semiconductor/systems community.

3.18 Dynamic Aspects of Centralized Automotive Software and System Architectures

Philipp Obergfell (BMW AG – München, DE)

Joint work of Philipp Obergfell, Jürgen Becker, Florian Ozwald, and Florian Sax
Main reference Simon Fürst, Markus Bechter: "AUTOSAR for Connected and Autonomous Vehicles: The AUTOSAR Adaptive Platform", in Proc. of the 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops, DSN Workshops 2016, Toulouse, France, June 28 – July 1, 2016, pp. 215–217, IEEE Computer Society, 2016.
URL http://dx.doi.org/10.1109/DSN-W.2016.24

Today's cars offer various control applications that may include the powertrain, the driving dynamics, or the characteristics of its body. Each of these control applications is realized by a series of electronic control units (ECUs) that interact on the basis of exchanged signals. Although these signals might be only relevant to the application layer of the ECU's software stack, also the low-level software needs to be defined statically whenever a new signal has to be either sent or received by an application.

From the perspective of OEMs, this development paradigm is no longer considered as reasonable. The main arguments in this respect are the increased set of signals (15.000 interactions among approximately 100 distributed ECUs) as well as the architecture's static definition which lacks in the support of light-weight updates.

One possible solution to overcome the mentioned disadvantages in current architectures are computing platforms which pursue three main goals: (1) The centralization of software applications from different existing ECUs, (2) the separation of applications and the low-level software by virtualization concepts, (3) and the interaction between applications on the basis of services.

As one precondition for their rollout, we consider two innovative modeling and model assessment methods. Both of them are related to the execution of software applications on component and system level and wrapped by the following questions:

1. How does the transition from statically defined software tasks in line with the AUTOSAR Classic Platform to POSIX processes and threads affect the design of computing platforms?
2. Which impact is illustrated by network paths between distributed software applications that are dynamically generated in line with the concept of runtime reconfiguration?

References

1 F. Oszwald, J. Becker, P. Obergfell, and M. Traub. *Dynamic Reconfiguration for Real-Time Automotive Embedded Systems in Fail-Operational Context.* IEEE International Parallel and Distributed Processing Symposium Workshops. 2018.

2 F. Oszwald, P. Obergfell, M. Traub, and J. Becker. *Using Simulation Techniques within the Design of a Reconfigurable Architecture for Fail-Operational Real-Time Automotive Embedded Systems.* IEEE International Symposium on Systems Engineering. 2018.

3 F. Oszwald,P. Obergfell, M. Traub, and J. Becker. *Reliable Fail-Operational Automotive E/E-Architectures by Dynamic Redundancy and Reconfiguration.* IEEE International System-on-Chip Conference. 2019.

4 P. Obergfell, S. Kugele, and E. Sax. *Model-Based Resource Analysis and Synthesis of Service-Oriented Automotive Software Architectures.* ACM/IEEE International Conference on Model Driven Engineering Languages and Systems. 2019.

5 P. Obergfell, F. Oszwald, M. Traub, and E. Sax. *Viewpoint-Based Methodology for Adaption of Automotive E/E-Architectures.* IEEE International Conference on Software Architecture Companion, ICSA Companion. 2018.

3.19 Metric-driven, System-level Testing: Release Autonomous Systems with Confidence

Maximilian Odendahl (Silexica – Köln, DE)

At Silexica, we're passionate about complex software solutions and solving the challenges associated with developing such systems. Today, we see that it is more and more difficult to keep the quality of the software stacks running. In the future, software complexity will continue to increase. Especially if you look at mobile systems like automated driving or robotics which have a high likelihood of doing harm, it is paramount that the software stacks within such systems are developed with rigor and thoroughness. Nobody wants to ship faulty products, right?

After investigating how modern software systems are developed and tested it became clear to us that there's a gap that needs filling. With the inherent complexity of today's software systems (and future systems even more so) traditional approaches of debugging and testing are not adequate to capture all defects. There are so many software components involved that when an error occurs, developers often spend several days or even weeks to figure out the root cause. Further, there is often no overview available which shows the complete system state including relevant metrics so that you can draw conclusions about where the root-cause might lie or determine if there's a new defect forming in the system.

With SLX Analytics, Silexica has created a new disruptive testing and integration platform that helps solving those challenges by providing automated testing of non-functional system-level metrics. It offers mult-level insights into all relevant Software Layers at once (from application, through middleware down to OS layer), allowing you to drill-down into the system state at the failing point in time. This helps to minimize the time spent on root-cause analysis and to detect (hidden) system defects in your products.

3.20 Parallel Programming Models for Critical Real-time Embedded Systems

Eduardo Quinones (Barcelona Supercomputing Center, ES)

Joint work of Eduardo Quinones, Sara Royuela, Maria A. Serrano

There is a visible trend in industry to adopt low-power hardware architectures with increasingly parallel execution capabilities supported by a variety of heterogeneous acceleration devices such as many-cores, and DSP fabrics, GPUs, SoC FPGAs. These architectures require the use of advanced parallel programming models for an efficient exploitation of their parallel capabilities. Parallel programming models however, cannot preclude the use of model-driven engineering, a very common practice in the software design as it facilitates the cyber/physical interaction description, abstracts the software/hardware architecture complexity and allows for early verification and validation performed on the models. This is the case of AUTOSAR, which is based on the concept of platform hardware independence, i.e., components are developed independently from their placement in the targeted processor architecture. Therefore, there is a need to bridge this gap by developing tools capable of generating parallel source code optimized for parallel heterogeneous platform, and run-time frameworks capable of respecting the guarantees devised at analysis time, while dynamically optimising the parallel execution to changing execution conditions.

3.21 Automotive System Design: Challenges of the Anthropocene

Sophie Quinton (INRIA – Grenoble, FR)

The current trend in the automotive industry towards automation and connectivity is raising a number of challenges in terms of cost-efficiency and dependability. At the same time, the conversation around the ongoing environmental crisis (in particular the climate emergency) makes it clear that we must either dramatically reduce our global resource consumption and pollution now, or face dire consequences in the near future. How do these two visions of the future relate?

The International Panel on Climate Change (IPCC) declared in 2018 that we must cut our CO2 emissions by 50% by 2030, reaching net zero around 2050, if we wanted to keep the global temperature in 2100 within 1.5°C over pre-industrial levels [1]. “Business as usual” scenarios (which we have followed since) lead to a temperature increase that could reach 5°C

or even more by the end of the century[1]. In comparison, the average global temperature around 21,000 years ago (during the last glacial period) was 6°C colder than today's, and at the time a large part of Northern Europe was covered by several kilometers of ice [3, 4]. The impact on nature of such a huge change in a such a short time is hard to anticipate, and the situation is unprecedented.

Beyond climate change, the Earth is a complex system with many interdependent feedback loops [5]. For example, the devastating fires in Australia at the end of 2019, which have been caused by a severe drought, have released an estimated 250 million tons of CO2 in the atmosphere [6], which will in turn contribute to global warming. A number of planetary boundaries [5] have already been transgressed (e.g., those related to biodiversity and the cycle of nitrogen), meaning that non-linear, abrupt changes at the continental or planetary level cannot be excluded anymore (and this, independently from the direct impact of human activity) because tipping points may have been reached [7]. In addition, some changes are already unavoidable due to the latency in some natural phenomena such as the rising level of oceans [8]. Whether we choose to act or not, we will have to deal with increasingly degrading conditions, the scale of which partly depends on our actions.

In this context, the drivers for action at all levels of society (individuals, companies, etc.) are of different natures. First, the fact that parts of the world will become uninhabitable [9] provides a powerful ethical motivation. A second, more personal reason would be the understanding that climate change will impact the entire population, either directly (e.g. hydric stress and an increased frequency and intensity of extreme meteorological episodes [1]) and indirectly (massive migrations are expected[2]). Finally, a practical reason for companies to address the environmental crisis is the fact that there is a good chance that drastic regulations will come into place – and, given the situation, we should all welcome and prepare for such measures[3].

So, what role does the automotive industry play in this? If we focus on greenhouse gas (GHG) emissions, it appears that about 14% of GHG emissions were directly due to the transportation sector in 2010, a major part of it due to cars [13]. Besides, electricity/heat production itself represents about 25% of all emissions. This underlines that the challenge for the automotive industry cannot be reduced to the shift from fossil fuel propulsion to electric propulsion: electricity itself is not "clean". Whether it is possible to decarbonize energy production is an open problem because of the enormous amount of mineral resources we would need to extract for this (based on current technology for renewable energy production) [14].

Dramatically reducing the overall energy consumption of the automotive sector is therefore an urgent necessity. Autonomous driving is often seen as a potential enabler for this reduction, as it increases driving efficiency and could help reducing traffic congestion. For example, the following claim can be found on the website of Synopsys [15]: *"The real promise of autonomous cars is the potential for dramatically lowering CO2 emissions. In a recent study, experts identified three trends that, if adopted concurrently, would unleash the full potential of autonomous cars: vehicle automation, vehicle electrification, and ridesharing."*

Interestingly, the quoted study [16] also states that "Ride-sharing and renewable energy sources [are] critical to its success." Further [17]: "Our central finding is that while vehicle electrification and automation may produce potentially important benefits, without a corresponding shift toward shared mobility and greater use of transit and active transport, these

[1] More recent publications are even more pessimistic [2].
[2] The most widely accepted figure is 200 million climate migrants by 2050 [11].
[3] Interestingly, carmakers are among key opponents of climate action [12].

two revolutions could significantly increase congestion and urban sprawl, while also increasing the likelihood of missing climate change targets." The reason for that is the so-called rebound effect [18]: The energy benefits of increased driving efficiency offered by autonomous cars would be offset by the steep increase in their use because of the much lower time costs they enable (possibility to work in the car, no need to park, etc.). Such effects are substantial even for ridesharing where modal shifts from public transit and active modes to private cars, longer traveling distances, and increase of urban sprawl are also expected [19].

In conclusion, scenarios in which autonomous driving does not lead to an increase in global CO2 emissions are based on very strong assumptions. In other words, the risk of backfire (i.e., the introduction of autonomous driving actually increases CO2 emissions) is high. Moreover, the contribution of autonomous driving in successful scenarios appears to be limited compared to simple measures such as vehicle right-sizing or de-emphasized performance. Finally, the entire lifecycle of the vehicules (production, usage, end of life) should be covered, with issues arising related to batteries and reparability.

The one thing for which a consensus seems to exist is that we need to rethink the role of cars in our societies – that may be the largest challenge faced by the automotive industry.

Post-scriptum. There was a breakout session on "Cars and Climate Change" at the Dagstuhl seminar following this presentation. Ten people participated. Discussions centered around regulations coming into effect in various countries and their blind spots, e.g. regarding the impact of production and the need to tackle the rebound effect. Questions about the reparability of autonomous cars and a possibility of software obsolescence were also raised.

References

1 Special report of the Intergovernmental Panel on Climate Change (IPCC) on global warming of 1.5°C. https://www.ipcc.ch/sr15/
2 Two French Climate Models Consistently Predict a Pronounced Global Warming. CNRS Press release. September 2019.
3 https://en.wikipedia.org/wiki/Last_Glacial_Period
4 https://en.wikipedia.org/wiki/Quaternary_glaciation
5 W. Steffen et al. *Planetary Boundaries: Guiding Human Development on a Changing Planet*. Science Vol. 347, Issue 6223, 1259855.
6 Australia's Bushfires Have Emitted 250m Tonnes of CO2, Almost Half of Country's Annual Emissions. The Guardian, 13 December 2019.
7 T. M. Lenton, J. Rockström, O. Gaffney, S. Rahmstorf, K. Richardson, W. Steffen, and H. J. Schellnhuber. *Climate Tipping Points – Too Risky to Bet Against*. Nature 575, 592-595 (2019).
8 Special report of the IPCC on the ocean and cryosphere in a changing climate. https://www.ipcc.ch/srocc/
9 C. Mora et al. *Global Risk of Deadly Heat*. Nature Climate Change volume 7, pages 501-506 (2017).
10 *Why Does Europe Need to Limit Climate Change and Adapt to Its Impacts?* European Environment Agency, February 2020. https://www.eea.europa.eu/highlights/why-does-europe-need-to
11 Migration and Climate Change. International Organization for Migration (IOM). IOM Migration Research Series, vol. 31.
12 Carmakers Among Key Opponents of Climate Action. The Guardian, 10 October 2019.
13 https://www.epa.gov/ghgemissions/global-greenhouse-gas-emissions-data#Sector
14 O. Vidal, B. Goffé, and N. Arndt. *Metals for a Low-carbon Society*. Nature Geoscience volume 6, pages 894–896 (2013).

15 https://www.synopsys.com/automotive/what-is-autonomous-car.html
16 https://www.itdp.org/2017/05/03/3rs-in-urban-transport/
17 L. Fulton, J. Mason, and D. Meroux. *Three Revolutions in Urban Transportation*. ITDP and UC Davis research report.
18 M. Taiebat, S. Stolper and M. Xu. *Forecasting the Impact of Connected and Automated Vehicles on Energy Use: A Microeconomic Study of Induced Travel and Energy Rebound*. Applied Energy Volume 247, 1 August 2019, Pages 297–308. https://doi.org/10.1016/j.apenergy.2019.03.174
19 N. Coulombel, V. Boutueila, L. Liu , V. Viguié and B. Yind. *Substantial Rebound Effects in Urban Ridesharing: Simulating Travel Decisions in Paris, France*. Transportation Research Part D: Transport and Environment Volume 71, June 2019, Pages 110–126. https://www.sciencedirect.com/science/article/pii/S1361920918303201

3.22 AnyDSL: A Partial Evaluation Framework for Programming High-Performance Heterogeneous Systems

Roland Leißa (Universität des Saarlandes – Saarbrücken, DE)

Joint work of Roland Leißa, Klaas Boesche, Sebastian Hack, Arsène Pérard-Gayot, Richard Membarth, Philipp Slusallek, André Müller, Bertil Schmidt
Main reference Roland Leißa, Klaas Boesche, Sebastian Hack, Arsène Pérard-Gayot, Richard Membarth, Philipp Slusallek, André Müller, Bertil Schmidt: "AnyDSL: a partial evaluation framework for programming high-performance libraries", PACMPL, Vol. 2(OOPSLA), pp. 119:1–119:30, 2018.
URL http://dx.doi.org/10.1145/3276489

AnyDSL advocates programming high-performance code using partial evaluation. We present a clean-slate programming system with a simple, filter-based, online partial evaluator that operates on a CPS intermediate representation. Our system exposes code generation for accelerators (vectorization/parallelization for CPUs, GPUs, and FPGAs) via compiler-known higher-order functions that can be subjected to partial evaluation. This way, generic implementations can be instantiated with target-specific code at compile time.

In our experimental evaluation we present three extensive case studies from image processing, ray tracing, and genome sequence alignment. We demonstrate that using partial evaluation, we obtain high-performance implementations for CPUs and GPUs from *one* language and *one* code base in a *generic way*. The performance of our codes is mostly within 10%, often closer to the performance of multi man-year, industry-grade, manually-optimized expert codes that are considered to be among the top contenders in their fields.

3.23 DAPHNE – An Automotive Benchmark Suite for Parallel Programming Models on Embedded Heterogeneous Platforms

Lukas Sommer (TU Darmstadt, DE)

Joint work of Lukas Sommer, Florian Stock, Leonardo Solis-Vasquez, Andreas Koch
Main reference Lukas Sommer, Florian Stock, Leonardo Solis-Vasquez, Andreas Koch: "DAPHNE – An automotive benchmark suite for parallel programming models on embedded heterogeneous platforms: work-in-progress", in Proc. of the International Conference on Embedded Software Companion, New York, NY, USA, October 13-18, 2019, p. 4, ACM, 2019.
URL http://dx.doi.org/10.1145/3349568.3351547

Due to the ever-increasing computational demand of automotive applications, and in particular autonomous driving functionalities, the automotive industry and supply vendors are starting to adopt parallel and heterogeneous embedded platforms for their products.

However, C and C++, the currently dominating programming languages in this industry, do not provide sufficient mechanisms to target such platforms efficiently. Established parallel programming models such as OpenMP and OpenCL on the other hand are tailored towards HPC systems.

To assess the applicability of established parallel programming models to automotive workloads on heterogeneous platforms, we present the DAPHNE benchmark suite [1]. The suite contains typical automotive workloads extracted from the open-source framework Autoware and implementations for each kernel in OpenCL, CUDA and OpenMP. Next to details on the benchmark suite, we share lessons learned during the implementation in [2], also with regard to non-functional aspects such as programmer productivity and maintainability, and give an outlook on potentially interesting developments.

References

1 L. Sommer, F. Stock, L. Solis-Vasquez, and A. Koch. *Daphne – An Automotive Benchmark Suite for Parallel Programming Models on Embedded Heterogeneous Platforms: Work-in-progress* in Proceedings of the International Conference on Embedded Software Companion, ser. EMSOFT '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3349568.3351547.

2 L. Sommer, F. Stock, L. Solis-Vasquez, and A. Koch, *FAT-Schriftenreihe 317 – EPHoS: Evaluation of Programming-Models for Heterogeneous Systems*, Tech. Rep., 2019. [Online]. Available: https://www.vda.de/de/services/Publikationen/fat-schriftenreihe-317.html.

3.24 The Role of Synchronized Time for Safe Integration of Heterogeneous Software Applications

Wilfried Steiner (TTTech Computertechnik – Wien, AT)

The use of synchronized time to establish system-wide real-time and safety properties becomes more and more mainstream. Today, we find this basic idea in the form of "time-aware shaping" [1] as part of the IEEE 802.1 TSN standards, as "time-coordinated computing" as technology brand from a large semiconductor vendor, as well as the "time-triggered architecture" [2] in the academic literature. Indeed, synchronized time combined with offline task and message scheduling is also the core paradigm of the safe execution platform MotionWise that is

deployed in automotive series production. In this talk I will review the core concepts of time synchronization and offline scheduling and give a quick overview of MotionWise. Furthermore, I will discuss current developments and potential research challenges in this domain.

References

1 W. Steiner, S. Craciunas, and R. Serna Oliver. *Traffic planning for time-sensitive communication*. IEEE Communications Standards Magazin, vol. 2, no. 2, pp. 42-47, 2018.
2 H. Kopetz and G. Bauer, *The Time-Triggered Architecture*. Proceedings of the IEEE, vol. 91, no. 1, pp. 112-126, 2003.

3.25 Taming Unpredictability: Leveraging Weakly-hard Constraints in Design and Adaptation

Qi Zhu (Northwestern University – Evanston, US)

Joint work of Qi Zhu, Rolf Ernst, Chung-Wei Lin, Wenchao Li, Samarjit Chakraborty, Hyoseung Kim
Main reference Chao Huang, Wenchao Li, Qi Zhu: "Formal verification of weakly-hard systems", in Proc. of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019, Montreal, QC, Canada, April 16-18, 2019, pp. 197–207, ACM, 2019.
URL http://dx.doi.org/10.1145/3302504.3311811
Main reference Hengyi Liang, Zhilu Wang, Debayan Roy, Soumyajit Dey, Samarjit Chakraborty, Qi Zhu: "Security-Driven Codesign with Weakly-Hard Constraints for Real-Time Embedded Systems", in Proc. of the 37th IEEE International Conference on Computer Design, ICCD 2019, Abu Dhabi, United Arab Emirates, November 17-20, 2019, pp. 217–226, IEEE, 2019.
URL http://dx.doi.org/10.1109/ICCD46524.2019.00035
Main reference Chao Huang, Kacper Wardega, Wenchao Li, Qi Zhu: "Exploring weakly-hard paradigm for networked systems", in Proc. of the Workshop on Design Automation for CPS and IoT, DESTION@CPSIoTWeek 2019, Montreal, QC, Canada, April 15, 2019, pp. 51–59, ACM, 2019.
URL http://dx.doi.org/10.1145/3313151.3313165
Main reference Hyunjong Choi, Hyoseung Kim, Qi Zhu: "Job-Class-Level Fixed Priority Scheduling of Weakly-Hard Real-Time Systems", in Proc. of the 25th IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS 2019, Montreal, QC, Canada, April 16-18, 2019, pp. 241–253, IEEE, 2019.
URL http://dx.doi.org/10.1109/RTAS.2019.00028

In automotive systems, limited resources and hard timing constraints often make it difficult to apply design changes for adding new functionality or correcting existing ones, and to adapt the systems under changing and challenging environment. In this talk, I will present some of our initial thoughts in leveraging weakly-hard timing constraints, where operations are allowed to occasionally miss deadlines in a bounded manner, to improve systems' capability in accommodating changes at design-time and runtime. I will introduce our preliminary works in 1) formally verifying the safety of control functions under weakly-hard constraints at the functional layer, 2) exploring the addition of security monitoring tasks under given weakly-hard constraints at the software layer, and 3) setting weakly-hard constraints cross functional and software layers to facilitate system retrofitting and adaptation.

3.26 Breaking Automotive Traditions

Dirk Ziegenbein (Robert Bosch GmbH – Stuttgart, DE)

Joint work of Selma Saidi, Sebastian Steinhorst, Arne Hamann, Dirk Ziegenbein
Main reference Selma Saidi, Sebastian Steinhorst, Arne Hamann, Dirk Ziegenbein, Marko Wolf: "Future automotive systems design: research challenges and opportunities: special session", in Proc. of the International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS 2018, part of ESWEEK 2018, Torino, Italy, September 30 – October 5, 2018, p. 2, IEEE / ACM, 2018.
URL http://dx.doi.org/10.1109/CODESISSS.2018.8525873

The landscape in the automotive industry is rapidly changing, driven by new automotive functionalities and applications such as autonomous driving and fueled by the entry of new players from the IT industry. Consequently, we witness the emergence of new trends in the automotive field imposing new challenges at the hardware and software level and at the way future automotive systems will be designed and developed. This includes the integration of formerly separated function domains onto centralized computing platforms, leading to a heterogeneous mix of applications with different models of computation, and the diversification and specialization of hardware platforms (comprising e.g., application cores, safety cores, deep learning accelerators, issued across several control units and connected by networks). Such platforms are required to meet the tremendous increase of computing power enforced by functionalities such as automated driving or on-line optimization of operating strategies for electrified powertrains. Furthermore, there is an increase of the connectivity beyond vehicle boundaries, such as vehicle-to-vehicle and vehicle-to-infrastructure communication (V2X) leading to systems-of-systems integration problems including function deployment from control unit over edge to cloud infrastructure as well as questions of how to guarantee sufficient availability of communication links and security on these connections.

These new trends constitute a real paradigm shift in the way classical automotive systems are viewed, thereby heavily challenging established methods and methodologies in automotive systems design. This provides a great opportunity for practitioners from both academia and industry to rethink the design of future automotive systems in order to accommodate high efficiency, safety and security requirements.

This talk first discusses the main trends that are currently driving innovation in the automotive industry. Then an overview of the emerging challenges and application constraints necessary to comply with the increased requirements with respect to cost, variability, support of legacy software and dependability is given. The goal is to raise awareness about emerging needs and application constraints in automotive.

4 Working groups

4.1 Modeling Hardware Dependencies and Software Dependencies

Jerónimo Castrillón-Mazo (TU Dresden, DE), Lulu Chan (NXP Semiconductors – Eindhoven, NL), Oliver Kopp (Mercedes Benz AG – Stuttgart, DE), Roland Leißa (Universität des Saarlandes – Saarbrücken, DE), Albrecht Mayer (Infineon Technologies – München, DE), Philipp Mundhenk (Autonomous Intelligent Driving GmbH – München, DE), Philipp Obergfell (BMW AG – München, DE), and Eduardo Quinones (Barcelona Supercomputing Center, ES)

4.1.1 Introduction

This discussion group covered the topics of capturing components and their dependencies. For instance, software A may run on electronic control unit (ECU) E only, whereas software B "just" requires CPU architecture M and operating system O [1].

We saw following aspects in the context of dependencies:

- Dependencies between software components
- Dependencies from software to hardware (platforms), for instance "high performance"/ accelerators, CPU, GPU, timing constraints, i.e., requirements of software to hardware
- Describing hardware capabilities (both in terms of dependencies as required by software, or as assignable resource for allocation)
- Type system of described data
- Non-functional requirements of software & hardware components and communicated data
- Deployment of software to hardware components

We identified function developers, application developers, and system integrators as roles.

- A function developer delivers functionalities of ECUs.
- An application developer delivers applications usable by customers.
- An application developer uses functions delivered by function developers.
- A system integrator combines functionality and application into a system.

We identified the following scenarios involving the aspects of HW/SW dependencies and the roles described above:

- S1: Developers need to describe their required environment.
- S2: Developers need to be aware of the provided environment. It may be the case that developers are unaware of certain properties of their environment. These properties need to be made explicit.
- S3: A system integrator can use information such as timing constraints or timing boundaries for developing the final distribution of software. In some instances, domain-specific knowledge is required for an optimal placement of functionalities. For example, the assignment of accelerated code is not straight-forward.
- S4: Leveraging the information available during runtime to, (for instance, triggering adoption of configuration, movement of execution, redeployment of functionalities)
- S5: Deduction of execution times based on the function in software, its mapping to hardware, and available/used resources
- S6: Configure system from a high-level API, e.g.:
  - a) Middleware configures underlying TSN-layer (based on offered options).
  - b) Migrate from ECU to HPU (accelerators)

4.1.2 Possible Solutions

We briefly discussed possible solutions to capture the HW/SW dependencies:

- High-level programming models, where an abstract model of and estimations (in terms of latency and energy consumption) for the application can be passed to the runtime. This can enable model-based runtime decision making according to available resources and requirements (cf. [5, 6]).
- Architecture description languages such as UML (cf. [4]).
- Languages supporting the description of composite cloud applications. In that field, dependencies and resource management are also a discussed topic ([3, 2]).
- With respect to the hardware description for software tools the IEEE standard 2804-2019[7] is a promising standard.

Potentially, a single solution is insufficient, if a large number of developers and engineers from different backgrounds are to use the system to work together (e.g., embedded SW development vs. high-performance SW development vs. system integration). Different users might have different requirements in terms of representation of data. Translations between formats or different representations would be required.

4.1.3 Next Steps

In the group discussion, we identified the following next steps: First, we are going to use the above scenarios to deduct concrete requirements. Second, we craft end-to-end application scenarios to enable evaluations.

References

1 M. Staron, *Automotive Software Architectures – An Introduction*. Springer Internantional Publishing AG, doi:10.1007/978-3-319-58610-6, 2017.

2 U. Breitenbücher, C. Endres, K. Képes, O. Kopp, F. Leymann, S. Wagner, J. Wettinger, and M. Zimmermann. *The OpenTOSCA Ecosystem – Concepts & Tools*. European Space project on Smart Systems, Big Data, Future Internet – Towards Serving the Grand Societal Challenges, Volume 1: EPS Rome, SciTePress, 2016.

3 OASIS. *Topology and Orchestration Specification for Cloud Applications (TOSCA)*. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca, 2020.

4 P. Obergfell, S. Kugele, and E. Sax. *Model-Based Resource Analysis and Synthesis of Service-Oriented Automotive Software Architectures*, 22nd International Conference on Model Driven Engineering Languages and Systems, IEEE/ACM, 2019.

5 A. Goens, R. Khasanov, M. Hähnel, T. Smejkal, H. Härtig, and J. Castrillon. *TETRiS: a Multi-Application Run-Time System for Predictable Execution of Static Mappings*. Proceedings of the 20th International Workshop on Software and Compilers for Embedded Systems, ACM, doi:10.1145/3078659.3078663, 2017.

6 R. Khasanov and J. Castrillon. *Energy-efficient Runtime Resource Management for Adaptable Multi-application Mapping*. Proceedings of the 2020 Design, Automation and Test in Europe Conference, Grenoble, 2020.

7 IEEE. *2804-2019 – IEEE Standard for Software-Hardware Interface for Multi-Many-Core*. 2020.

4.2 Weakly Hard Real-Time Models

Martina Maggio (Lund University, SE), Jerónimo Castrillón-Mazo (TU Dresden, DE), Lulu Chan (NXP Semiconductors – Eindhoven, NL), Rolf Ernst (TU Braunschweig, DE), Chung-Wei Lin (National Taiwan University – Taipei, TW), Zhu Qi (Northwestern University – Evanston, US), Eduardo Quinones (Barcelona Supercomputing Center, ES), Sophie Quinton (INRIA – Grenoble, FR), and Selma Saidi (TU Dortmund, DE)

We discussed the weakly hard model, its application domains, and what can be achieved using it. We concluded that the model itself is useful when, for example, safety-critical functions are executed on systems that are resource-constrained and evolve over time. For example, when an update or upgrade happens, functionalities are added and this can disrupt the behavior of the real-time systems as it was designed in the first place. Another important use of the model is handling faults and identifying risky situations. A system that can withstand two deadline misses in a window of four could be put in an alarm state after one deadline miss occurs, monitors can be activated, and countermeasures can be taken to ensure the correct behavior despite misses.

Application domains that were identified include: (1) security attacks, (2) control tasks and robotics, (3) image transmission. For security attacks, we talked about protocols in which a given sequence of messages has to be delivered and disrupting the transmission of some of these messages could lead to damage and problems. One of the main application domains would be vehicle to vehicle communication with wireless communication. For control tasks, there is a lot of research in trying to connect weakly hard guarantees and functional properties (e.g., despite missing deadlines, the control system is still stable and has good performance in terms of speed of convergence to the objective and integral of the squared error). We discussed if this is the right model to use for this case, without any conclusive result. It seems that the weakly hard model is a good match for performance handling, but less good for stability analysis, i.e., the analysis is mathematically involved and it is hard to make it scale and generalize it. For image transmission, we discussed cases in which images composed of multiple pixels have to be transmitted in a way such that the content is recognizable despite some pixels not being transmitted correctly.

Generally speaking, these different application models would have very different m-K parameters. For control systems, m and K would be close to one another, while for image transmission m and K would be quite far from each other (an image has very many pixels, thus a large K, and only a few of them can be erroneous, thus m is small). We then briefly discussed how to obtain these parameters. One example is using truncated probability distributions and estimating the parameters from previous runs of the applications. We noted that it might be quite difficult to derive the parameters and also to analyze the corresponding models.

Another point that we addressed is what to do with these models. Two ideas emerged: the first one is to understand when it is important to switch to some safety mode; while the second one is to determine how to handle slack time reclamation (if two tasks can be given some additional time to execute, maybe it is better to prioritize based on how many deadlines can they miss in their future executions based on their window sizes, rather than looking at one single instance of the task execution).

4.3 Machine Learning in Cyber-Physical Systems

Frank Mueller (North Carolina State University – Raleigh, US), Bart Besselink (University of Groningen, NL), Alessandro Biondi (Sant'Anna School of Advanced Studies – Pisa, IT), Lulu Chan (NXP Semiconductors – Eindhoven, NL), Jyotirmoy Deshmukh (USC – Los Angeles, US), Masaki Gondo (eSOL – Tokyo, JP), Baik Hoh (Toyota Motors North America – Mountain View, US), Peter Gorm Larsen (Aarhus University, DK), Mark Lawford (McMaster University – Hamilton, CA), Martina Maggio (Lund University, SE), Albrecht Mayer (Infineon Technologies – München, DE), Zhu Qi (Northwestern University – Evanston, US), Lukas Sommer (TU Darmstadt, DE), Wilfried Steiner (TTTech Computertechnik – Wien, AT), and Seyhan Uçar (Toyota Motors North America – Mountain View, US)

This is the recording of a working group on machine learning for cyber-physical systems, in which more people than recorded participated.

We started the discussion with a challenging example. Suppose that a machine learning system, used for vision and object recognition in an autonomous car, is not able to detect a stop sign. Using some ground truth information (e.g., information received with other sensors), the stop sign is identified, and the learning system is able to rewind and go back to the frame where the stop sign could have been detected. This enables for continuous on-vehicle learning. We asked ourselves how is this possible to achieve. Some aspect of this problem involve the cloud and the ability to "learn from mistakes", autonomously, and possibly cooperatively (with other vehicles).

Generally speaking we moved the discussion on the topics of how it is possible to detect and recognize bad situations, and how it is possible to avoid overfitting in these cases. We discussed the need for going back to supervised learning for these cases and what the update of the learning procedures will mean. However, we noted many open problems, like the presence of dynamically moving objects. We need to analyze the root cause for the error (the missed detection) and to understand how to correct it on the fly.

We then started talking about uncertainty and the guarantees that can be provided either with machine learning or with other techniques, linking this to model order reduction to reduce the complexity of the models that are used for control (e.g., trajectory planning). We had a brief discussion on the relation between optimal control and reinforcement learning. We noted that a recently published paper by Benjamin Recht tries to link the two [1].

References

1 B. Recht. *A Tour of Reinforcement Learning: The View from Continuous Control.* Annual Review of Control, Robotics, and Autonomous Systems, 2(1): 253–279, 2019

4.4 HW/SW Architecture Exchange

Philipp Mundhenk (Autonomous Intelligent Driving GmbH – München, DE)

Joint work of all participants of Dagstuhl Seminar 19502

The breakout session was set up as an interactive session to identify upcoming trends in automotive architectures and to discuss which of the challenges are already being addressed by which solutions. The majority of participants of Dagstuhl Seminar 19502 took part in this session.

The focus of the session was interactively set on determinism for new architectures. Ensuring such determinism is non-trivial with the emergence of new technologies in safety-critical systems, such as Ethernet-based, service-oriented communication, microprocessors, and additional abstraction layers in software, adding complexity [2].

4.4.1 Background

New software platforms add additional layers, supporting developers, simplifying application development and increasing speed of innovation [1]. However, each layer being introduced into a system adds complexity and variability. Ensuring correct and deterministic behavior in such a system is not trivial. For example, scheduling problems arise, e.g., spatial or time-based scheduling of computation on hardware accelerators. Often, this is made worse due to the proprietary nature of many of the hardware components, especially accelerators, such as GPUs. Typically, not much information is known about these, making achieving determinism more difficult. Another example is the mapping of scheduling algorithms and schedulable units on the layer of software platform, operating system, hypervisor, device, etc.

4.4.2 Approaches

Due to the large nature of the problem there is no single solution. Different approaches need to be combined. The following approaches, addressing parts of the problem, have been discussed in this breakout session:

- Limiting every layer in its potentially unlimited freedom could help reduce the overall uncertainty in the system. E.g., timing restrictions can be imposed on the hypervisor, the operating system, etc., and not just on the level of application. The same goes for memory bounds, etc.
- Design time checks will be required wherever possible. The developers and integrators should not be allowed to perform certain actions, if these are already known to be breaking the system. E.g., logically connecting an ASIL-D input of one component to an ASIL-A output of another component can already be detected at design time, even if the system uses service-oriented communication and sets up connections at runtime only.
- The behavior defined at design time needs to be enforced at runtime. E.g., through configuration of the operating system, or the software platform. Furthermore, the correct behavior needs to be monitored at runtime, such that failure reactions can be triggered when abnormal behavior, compared to the design time specification, is detected.
- The execution behavior of microprocessors today is inherently nondeterministic. Due to the use of multi-level caches, virtual memory, etc. it is not reasonably possible to achieve deterministic behavior on microprocessors. This might, however, not hold forever. Chip

designers are working on making their processors more predictable. This would be a long-term approach and not an immediate solution.
- Another layer of protection can be achieved through introduction of a parallel computation path with plausibility computations. These parallel paths do not need to perform the full set of computations, but can be more lightweight, e.g., computing boundaries, and can be implemented on a more deterministic system, such as a microcontroller.
- Such a reliable path, if computation is accurate enough, could also be used as a fallback path, in case the main path fails during operation.
- In case a redundant, deterministic path has less computational performance, the environment of the autonomous system can also be adapted. E.g., the speed of a vehicle could be reduced, allowing longer reaction times, thus allowing computation to be slower and less powerful computation units to be sufficient.

4.4.3 Open Challenges

Despite the numerous approaches discussed in this session, a number of open challenges have been identified, e.g.,:
- Is achieving reliability of the complete system, including microprocessor, GPU, hardware accelerators, operating system, middleware, etc. a realistic goal in the long-term? There is no definitive answer to this question, yet.
- How exactly can redundant paths be designed? Is a reliable path always required? Or could e.g., an unreliable path, plus reliable plausibility checkers be sufficient? Could multiple unreliable paths and reliable voters be used? Are there other approaches?
- When using design time checks, how is illicit behavior defined? How can it be ensured that the complete set of illicit behavior is covered? How can this be integrated into the many different approaches (e.g., development frameworks, continuous integration systems, etc.) at design time? How to enforce this correct behavior at runtime in an efficient and reliable manner?
- When using partial checks in the individual layers of the system, how can these be enforced? What are reasonable checks and bounds (e.g., for the number of obstacles detected)? How can the boundaries be monitored in an efficient manner?

- When adapting to the environment, e.g., through slowing down the vehicle, it needs to be ensured that the adjustment is sufficient. Depending on the available reliable compute performance and the complexity of required computation, it might e.g., still not be sufficient to slow down the vehicle, even to walking speed.
- How to deal with input to the system that is inherently indeterministic? The interpretation of sensor inputs e.g., will likely always introduce indeterminism to the system. How can such input, which is essential to the functionality of the system, be processed in a deterministic system?

4.4.4 Conclusion

This session identified a large number of challenges to and a number of solutions for dependability in the area of highly complex mixed-criticality systems on heterogeneous architectures. A large amount of research is required on all aspects of the system, from hardware over different software layers to processes and legislation in order to address these challenges.

References

1 P. Mundhenk, E. Parodi, and R. Schabenberger. *Fusion: A Safe and Secure Software Platform for Autonomous Driving*. In Proceedings of the 2nd Workshop on Autonomous Systems Design (ASD 2020). Grenoble, France, 2020.
2 L. Bauer, M. Damschen, D. Ziegenbein, A. Hamann, A. Biondi, G. Buttazzo, and J. Henkel. *Analyses and Architectures for Mixed-critical Systems: Industry Trends and Research Perspective*. In Proceedings of the International Conference on Embedded Software Companion (EMSOFT '19). New York, USA, Article 13, 1–2.

4.5 On the Relation between Programming Models, Computational/Execution Models and Software Platforms in Automotive

Selma Saidi (TU Dortmund, DE), Alessandro Biondi (Sant'Anna School of Advanced Studies – Pisa, IT), Rolf Ernst (TU Braunschweig, DE), Sabine Glesner (TU Berlin, DE), Oliver Kopp (Mercedes Benz AG – Stuttgart, DE), Roland Leißa (Universität des Saarlandes – Saarbrücken, DE), Philipp Mundhenk (Autonomous Intelligent Driving GmbH – München, DE), Philipp Obergfell (BMW AG – München, DE), Eduardo Quinones (Barcelona Supercomputing Center, ES), Lukas Sommer (TU Darmstadt, DE), and Dirk Ziegenbein (Robert Bosch GmbH – Stuttgart, DE)

The breakout session discussed the relationship between (parallel) programming models, software platforms and models of execution.

Software platforms in automotive integrate multiple software components including applications, middleware, operating systems, from different software providers and are executed on a heterogeneous collection of hardware including microcontrollers, CPUs and GPUs. Pthread is often used as an execution model (e.g., in AUTOSAR Adaptive) to control multiple workflows that overlap over time.

With the adoption of high-performance platforms in automotive, parallel programming models are becoming prevalent, targeting heterogenous hardware platforms composed often of a combination of CPUs and GPUs. Several programming models like CUDA and OpenMP, implement different models of execution (i.e., execution semantics) in terms of data communication (e.g. shared memory or message passing) and synchronization (e.g., implicit or explicit). Unlike HPC systems, automotive systems have different/additional requirements in terms of heterogeneity, timing predictability, data versioning, etc. Therefore, a computational model (can also be referred to as System Execution Model) is further needed to coordinate the execution of different software components. The role of computational models is to guarantee the integration of different software layers and orchestrate their "correct" execution. Correct execution requires predictable or defined execution and communication timings, data consistency, data versioning, etc. The Logical Execution Time (LET) Model is an example of computational models. Therefore, the programming model is different from the computational model and this latter is often much more complicated as it must as well guarantee non-functional properties and a seamless integration between software layers.

(During the discussion at Dagstuhl, the term execution model was interchangeably used to designate programming models, execution semantics and system-level computational execution models. However, a clear distinction between the two is needed).

It remains however unclear how programming models, used by software programmers to implement a given function, interface with computational models, used by system integrators to guarantee a system-level correct execution. Physical timing (e.g. period, timing constraints, end-to-end latencies) for instance define properties of the software and should be part of the programming model. These properties are later further propagated from the programming model to the computational model responsible for their implementation by structuring the execution and synchronization in the software architecture. It seems that a clear unified software platform view integrating both programming models and computational models is missing. How can the programming model and computational model be integrated in the automotive software stack? Regarding hardware resources allocation, are all hardware resources allocated by the computational model? What's the role/requirements derived from the programming model regarding tasks partitioning, resources allocation and scheduling? Which level of abstraction should be used for scheduling, communication and synchronization? These are all important open questions to be addressed.

4.6 Benchmarking Efforts for Future HW/SW Platforms

Lukas Sommer (TU Darmstadt, DE), Bart Besselink (University of Groningen, NL), Alessandro Biondi (Sant'Anna School of Advanced Studies – Pisa, IT), Jerónimo Castrillón-Mazo (TU Dresden, DE), Lulu Chan (NXP Semiconductors – Eindhoven, NL), Wanli Chang (University of York, GB), Thidapat Chantem (Virginia Polytechnic Institute – Arlington, US), Rolf Ernst (TU Braunschweig, DE), Oliver Kopp (Mercedes Benz AG – Stuttgart, DE), Mark Lawford (McMaster University – Hamilton, CA), Roland Leißa (Universität des Saarlandes – Saarbrücken, DE), Martina Maggio (Lund University, SE), Philipp Mundhenk (Autonomous Intelligent Driving GmbH – München, DE), Philipp Obergfell (BMW AG – München, DE), Eduardo Quinones (Barcelona Supercomputing Center, ES), Selma Saidi (TU Dortmund, DE), Lea Schönberger (TU Dortmund, DE), and Dirk Ziegenbein (Robert Bosch GmbH – Stuttgart, DE)

The closing session of the seminar also saw a discussion about benchmarking efforts for future automotive HW/SW platforms. The participants agreed that the availability of representative benchmarks is critical for the development of central components, such as programming models, compilers, runtime systems and the evaluation of hardware platforms. Therefore, academia and industry should collaborate to provide open benchmarks for use in research and development.

One such approach is the open-source DAPHNE benchmark suite [4], which was created with the intent to provide a number of representative automotive workloads under the permissive Apache v2 license. The suite contains typical automotive workloads extracted from the open-source framework Autoware and implementations for each kernel in OpenCL, CUDA and OpenMP.

In order to extend this open benchmark suite, the original authors of the benchmark seek for contributions to the benchmark. The form of the contribution can be manifold, for example:

- Hints to relevant algorithms used in real-world automotive applications to be implemented as benchmark kernel.
- Implementations of automotive algorithms in any programming language to be added to the benchmark and ported to the different programming models.
- Representative data-sets, e.g. as captured by sensors and cameras, for use with the benchmark algorithms.

We invite everyone to contribute to this open benchmarking approach through the Github-page of the DAPHNE project [5] or via e-mail to sommer@esa.tu-darmstadt.de

Another source for benchmarks is the community forum of the WATERS workshop [1] where the models of the WATERS industrial challenges are stored. The purpose of these challenge is to share ideas, experiences and solutions to concrete timing verification problems issued from real industrial case studies. It also aims at promoting discussions, closer interactions, cross fertilization of ideas and synergies across the breadth of the real-time research community as well as industrial practitioners from different domains. Here the focus is on non-functional performance models of real-world systems, ranging from engine control [2] to automated driving [3].

References

1 WATERS @ ECRTS Community Forum. *Industrial Challenge Models.* https://www.ecrts.org/forum/ .

2 S. Kramer, D. Ziegenbein, and A. Hamann. *Real World Automotive Benchmarks For Free.* WATERS 2019, Lund, 2019, https://www.ecrts.org/forum/download/file.php?id=9&sid=d5f57b4d87f08f919fd0a366b31bab49.

3 F. Rehm, D. Dasari et al. *System Performance Modelling of Heterogeneous HW Platforms: An Automated Driving Case Study.* 22nd Euromicro Conference on Digital System Design (DSD), Dubrovnik, 2019. https://doi.org/10.1109/DSD.2019.00060.

4 L. Sommer, F. Stock, L. Solis-Vasquez, and A. Koch, "Daphne – An Automotive Benchmark Suite for Parallel Programming Models on Embedded Heterogeneous Platforms: Work-in-progress," in *Proceedings of the International Conference on Embedded Software Companion*, ser. EMSOFT '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3349568.3351547.

5 L. Sommer, F. Stock, L. Solis-Vasquez, and A. Koch, "Darmstadt Automotive Parallel HeterogeNEous (DAPHNE) Benchmark-Suite," https://github.com/esa-tu-darmstadt/daphne-benchmark, 2019.

4.7 Modularizing Control Systems

Dirk Ziegenbein (Robert Bosch GmbH – Stuttgart, DE), Bart Besselink (University of Groningen, NL), Peter Gorm Larsen (Aarhus University, DK), and Mark Lawford (McMaster University – Hamilton, CA)

Control systems are defined by their input-output behavior and their end-to-end cause and effect chains. A typical way of decomposing control software is however according to functional subsystems, which is also reflected in the predominant modeling style which typically features hierarchical dataflow and control flow blocks as, e.g., in Simulink. When maintaining and extending such models, changes or additional features usually require modifications to more than one of these subsystems and thus the respective software modules need to be changed in order to implement them. These modifications of multiple subsystems to handle different cases often results in a multitude of distributed "balconies" (nested if-then-else structures) often with state feedback through unit delays effectively resulting in distributed implicit state machines which are hard to understand and maintain.

In this breakout session, the participants discussed whether there are better ways to structure control systems in order to improve encapsulation and information hiding principles from classical software engineering [6] in control systems models. Finding such modularization approaches are of great importance considering the current automotive trends of increasing integration complexity as well as the update of vehicle functions after the vehicle's start of production (feature-over-the-air-update). Two approaches were discussed in more detail.

Simulink is widely used to develop control systems software in the automotive domain but the engineers using it often focus solely on the dataflow aspects of the problem which has been shown in general software engineering to result in systems with poor modularization [6]. While Simulink constructs such subsystems, libraries and model references can all be used to help structure Simulink models, all of these constructs have limitations when trying to perform system modularization based upon Parnas' information hiding principles [3]. Recently Mathworks has added a new language construct, Simulink functions, which can be used to help implement modular Simulink designs on an information hiding basis similar to

modules in the C programming language [3]. Further study is needed to see if structuring Simulink models using the proposed information hiding principles leads to designs that are easier to maintain and comprehensible by the control domain engineers.

At Bosch, a method called SCODE (System Co-Design) Essential Analysis [2] has been developed which decomposes a control system in terms of discrete situations in the system context, the so-called functional invariants or modes. In these functional invariants the control system shows a coherent input-output behavior. The SCODE Essential Analysis is based on the Essential Systems Analysis [4], developed by McMenamin and Palmer originally for IT systems, and extends and modifies it to enable application for physically dominated systems.

The analysis follows a three-step-approach. In the first step, the problem space of the control system is defined by a Zwicky-Box in terms of input and output dimensions (aspects of the system or its context that cause or represent different system behaviors) and alternatives (possible values or value ranges of a dimension). In the second step, the problem space is partitioned into valid and invalid input and output sets, and modes are defined by assigning input sets to output sets. In the third step, the mode transitions are defined, i.e. it is specified which context changes cause a transition between system modes. The analysis is supported by a tool [5], which encodes the problem space as binary decision diagrams and supports the developer by guiding the analysis (e.g., to discover not yet covered parts of the problem space) and to guarantee properties of the system design (e.g. consistency and completeness of mode specifications as well as uniqueness and liveliness of mode transitions). Based on the analysis results, mode switching logic (in, e.g., Simulink, ASCET or C), test cases as well as documentation and reports can be generated automatically.

In related work, [1] attempts to simplify mode switching logic implemented in Simulink truth tables by identifying common behaviors and merging the resulting modes. Still within the Simulink setting, [7] discusses how Simulink models implicitly encoding state machines can be converted into StateFlow models effectively making the mode switching logic in data flow models explicit.

Given the above related works it seems that mode switching logic oriented functions are primary candidates for refactoring and alternative modularizations in order to improve control software designs.

References

1 M. Bialy, M. Lawford, V. Pantelic, and A. Wassyng. *A methodology for the simplification of tabular designs in model-based development.* In 2015 IEEE/ACM 3rd FME Workshop on Formal Methods in Software Engineering, pp. 47-53. IEEE, 2015.

2 M. Bitzer, M. Herrmann, and E. Mayer-John. *System Co-Design (SCODE): Methodology for the Analysis of Hybrid Systems.* at-Automatisierungstechnik, 67(9), 2019, https://doi.org/10.1515/auto-2019-0003.

3 M. Jaskolka. *A Comparison of Componentization Constructs for Supporting Modularity in Simulink*, SAE WCX 2020 World Congress Experience, April 2020.

4 S. McMenamin and J. Palmer. *Essential Systems Analysis.* Prentice-Hall, 1984, https://archive.org/details/essentialsystems00mcme.

5 ETAS GmbH. *SCODE Essential Analysis – Describe and Verify Complex Control Systems.* Whitepaper, 2016, https://www.etas.com/en/downloadcenter/24029.php.

6 Da. L. Parnas. *On the Criteria To Be Used in Decomposing Systems into Modules* (PDF). Communications of the ACM. **15** (12): 1053-1058, 1972, doi:10.1145/361598.361623.

7 S. Wynn-Williams, Z. Diskin, V. Pantelic, M. Lawford, G. Selim, C. Milo, M. Diab, and F. Weslati. *SL2SF: Refactoring Simulink to Stateflow.* In International Conference on Fundamental Approaches to Software Engineering, pp. 264-281. Springer, Cham, 2019.

4.8 SW Lifecycle of Dependable Evolving Systems

Dirk Ziegenbein (Robert Bosch GmbH – Stuttgart, DE), Rolf Ernst (TU Braunschweig, DE), Moritz Neukirchner (Elektrobit Automotive – Erlangen, DE), and Selma Saidi (TU Dortmund, DE)

Traditionally, automotive software was developed, tested and shipped bundled together with the hardware as an electronic control unit to the customer. Updates due to planned maintenance or bug removal were rather an exception and were performed by (re-)flashing the complete electronic control unit in the repair shop. Due to the increasing over-the-air connectivity beyond vehicle boundaries which has shown to create a new security attack surface which mandates fast fixing of security issues as well as to emerging business models of selling feature updates during the vehicle lifecycle, the frequency of such updates will increase significantly in the foreseeable future. Furthermore, due to the integration of several functionalities on so-called vehicle integration platforms, the updates will also become modular, exchanging or adding only parts of the software at a time. These two trends and the challenges they are causing were discussed in this breakout session.

With the increased frequency of updates, the whole release process will need to be highly automated. Note that the release process comprises not just the test and verification but also the fulfillment of all legal requirements and norms, like homologation or the proof of functional safety. While the test automation is already well on its way (even though the automated regression testing of cyber-physical systems still has a lot of challenges), the latter procedures are still dominated by a lot of manual work. Here, being able to achieve modularity, in the sense, that local changes in the system will also only have a clearly identifiable local impact on e.g. the safety case, is seen as a huge advantage over the current state-of-practice. In fact, the automotive standard for functional safety ISO 26262 is very inconclusive about requirements for maintenance such that more work is needed here.

The fact that maintenance will have to be offered over the whole vehicle life cycle poses additional challenges. The typical support and production cycles for microprocessors is much lower than the vehicle life cycle. Thus, one has to assume that during the end of the vehicle life cycle, the software might even need to be ported to new hardware generations as part of the maintenance process (a prominent example is Tesla, which is already offering upgrades to new vehicle computer generations to its customers). Similarly, the rapid development of automotive software platforms such as AUTOSAR Adaptive will lead to questions of whether keeping a system on its original SW platform version or continuously updating the platform version. Both strategies have their specific advantages and disadvantages. While keeping the platform version stable limits changes and reduces risks, it also requires the SW platform vendor to fix issues in all versions that are in the field. This long-term maintenance needs to be considered in the business model right from project start as it tremendously influences the cost of updates. Here, the automotive industry might look e.g. at consumer electronics and the update/maintenance models of SW platforms such as Android or iOS.

The participants of this breakout session agreed that this SW lifcycle topic for dependable evolving systems should be investigated in more detail. The organization of a special session at a top embedded systems conference is envisioned.

Participants

- Bart Besselink
University of Groningen, NL
- Alessandro Biondi
Sant'Anna School of Advanced Studies – Pisa, IT
- Jerónimo Castrillón-Mazo
TU Dresden, DE
- Lulu Chan
NXP Semiconductors – Eindhoven, NL
- Wanli Chang
University of York, GB
- Thidapat Chantem
Virginia Polytechnic Institute – Arlington, US
- Jyotirmoy Deshmukh
USC – Los Angeles, US
- Rolf Ernst
TU Braunschweig, DE
- Sabine Glesner
TU Berlin, DE
- Masaki Gondo
eSOL – Tokyo, JP
- Baik Hoh
Toyota Motors North America – Mountain View, US
- Oliver Kopp
Mercedes Benz AG – Stuttgart, DE
- Peter Gorm Larsen
Aarhus University, DK
- Mark Lawford
McMaster University – Hamilton, CA
- Roland Leißa
Universität des Saarlandes – Saarbrücken, DE
- Chung-Wei Lin
National Taiwan University – Taipei, TW
- Martina Maggio
Lund University, SE
- Albrecht Mayer
Infineon Technologies – München, DE
- Frank Mueller
North Carolina State University – Raleigh, US
- Philipp Mundhenk
Autonomous Intelligent Driving GmbH – München, DE
- Alessandra Nardi
Cadence – San Jose, US
- Moritz Neukirchner
Elektrobit Automotive – Erlangen, DE
- Philipp Obergfell
BMW AG – München, DE
- Maximilian Odendahl
Silexica – Köln, DE
- Zhu Qi
Northwestern University – Evanston, US
- Eduardo Quinones
Barcelona Supercomputing Center, ES
- Sophie Quinton
INRIA – Grenoble, FR
- Selma Saidi
TU Dortmund, DE
- Ignacio Sañudo Olmedo
University of Modena, IT
- Lea Schönberger
TU Dortmund, DE
- Lukas Sommer
TU Darmstadt, DE
- Wilfried Steiner
TTTech Computertechnik – Wien, AT
- Seyhan Uçar
Toyota Motors North America – Mountain View, US
- Dirk Ziegenbein
Robert Bosch GmbH – Stuttgart, DE