

# Tradeoffs in Reactive Systems Design

Jerónimo Castrillón-Mazo<sup>\*1</sup>, Chadlia Jerad<sup>\*2</sup>, Edward A. Lee<sup>\*3</sup>,  
Claire Pagetti<sup>\*4</sup>, and Shaokai Jerry Lin<sup>†5</sup>

1 TU Dresden, DE. [jeronimo.castrillon@tu-dresden.de](mailto:jeronimo.castrillon@tu-dresden.de)

2 University of Manouba, TN. [chadlia.jerad@ensi-uma.tn](mailto:chadlia.jerad@ensi-uma.tn)

3 University of California – Berkeley, US. [eal@berkeley.edu](mailto:eal@berkeley.edu)

4 ONERA – Toulouse, FR. [claire.pagetti@onera.fr](mailto:claire.pagetti@onera.fr)

5 University of California – Berkeley, US. [shaokai@berkeley.edu](mailto:shaokai@berkeley.edu)

---

## Abstract

Reactive systems – software systems that continuously interact with their physical or digital environment – are central to safety-critical domains such as autonomous vehicles, industrial automation, and medical devices. These systems face inherent design tensions: the need to be predictable yet adaptable, timely yet accurate, consistent yet available, and secure yet accessible. Addressing one requirement often undermines another, revealing tradeoffs that are not merely engineering challenges but fundamental limits. This seminar aimed to confront these tradeoffs directly, drawing on insights from real-time systems, distributed computing, formal methods, machine learning, and security. By exploring case studies, formal frameworks, and practical tools, we made progress in the understanding of how to make design decisions when no single solution satisfies all competing goals. Interactive sessions and demos gave participants a tangible sense of the costs and compromises involved. With all this, we achieved the seminar’s goal, namely, to cultivate a shared understanding of how to navigate the complex design space of reactive systems and chart paths toward more robust and principled solutions.

**Seminar** February 23–28, 2025 – <https://www.dagstuhl.de/25091>

**2012 ACM Subject Classification** Computer systems organization → Embedded and cyber-physical systems; Computer systems organization → Dependable and fault-tolerant systems and networks; Computer systems organization → Real-time systems; Computing methodologies → Distributed computing methodologies; Computing methodologies → Concurrent computing methodologies; Mathematics of computing → Mathematical analysis; Security and privacy → Software and application security

**Keywords and phrases** reactive systems, cyber-physical systems, design tradeoffs, real-time systems, distributed computing, predictability, adaptability, timeliness, accuracy, consistency, availability, security, accessibility, machine learning, formal methods, system design, embedded systems, safety-critical systems, tool support, programming models, runtime verification

**Digital Object Identifier** 10.4230/DagRep.15.2.126

---

\* Editor / Organizer

† Editorial Assistant / Collector



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 4.0 International license

Tradeoffs in Reactive Systems Design, *Dagstuhl Reports*, Vol. 15, Issue 2, pp. 126–157

Editors: Jerónimo Castrillón-Mazo, Chadlia Jerad, Edward A. Lee, Claire Pagetti, and Shaokai Jerry Lin



Dagstuhl Reports

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Executive Summary

*Jerónimo Castrillón-Mazo (TU Dresden, DE)*

*Chadlia Jerad (University of Manouba, TN)*

*Edward A. Lee (University of California – Berkeley, US)*

*Claire Pagetti (ONERA – Toulouse, FR)*

License © Creative Commons BY 4.0 International license

© Jerónimo Castrillón-Mazo, Chadlia Jerad, Edward A. Lee, and Claire Pagetti

Reactive systems are software systems that engage in a continual dialogue with their environment. They constitute the software parts of cyber-physical systems where timely reactions are often critical to safety. Applications include autonomous vehicles, electric power systems, industrial automation, healthcare electronics, and robotics. Because the software engages in a continual dialog with its environment, it often has conflicting requirements. It needs to be predictable, but robust to unpredictable events; it needs to react in a timely manner, but this often requires reacting with inconsistent information; it needs to be adaptable, but demonstrably safe; and it needs to be secure, but accessible and available. Many conferences and workshops focus on one of the goals, such as achieving real-time behavior, without explicitly acknowledging the costs and without providing sound strategies for dealing with failures that prevent reaching the goals. The focus of this seminar was on the tradeoffs that are intrinsic in the design of such systems. When you make a system predictable, available, secure, or even demonstrably safe, what have you lost?

This seminar pulled in experts from manifold disciplines, both academic and industrial, to identify and discuss the fundamental limits in reactive systems design that make tradeoffs inevitable. In preparation for the seminar, the co-organizers reached out to leading experts among the participants and invited them to deliver four talks to frame the discussions for each of the first four days. Following this initial outreach, all participants were contacted and invited to contribute through short talks, position statements, and demonstrations of any relevant tools. The seminar format was kept flexible and open to allow space for ideas and key discussion points to emerge organically. In this context, the working groups described in Section 4 arose from the core ideas and challenges identified during the morning sessions. These groups held their discussions during the first half of the afternoon sessions of the first two days. The seminar was organized as described below.

### Day 1: Consistency vs. availability

Consistency is agreement on shared information across a distributed system. Availability is the ability to act on that shared information in a timely way. It has been shown that as latency increases when sharing such information, either consistency or availability or both must be sacrificed. This topic focused on how to manage this tradeoff. It included presentations proposing different ways to formalize the tradeoff, as well as examples of how it arises in software design. Two breakout groups formed on the first day, focusing on the topics “Distributed Music Challenge” and “Can AI Be Used in Critical Systems?”

### Day 2: Timeliness vs. accuracy

Because reactive systems interact continuously with their environment, they need to sense and interpret that environment. Today, many such systems need to include sophisticated vision subsystems, audio information processing, motion sensing, etc. The computation required to interpret sensor data often implies unacceptable delays or impossible energy

requirements. It is not acceptable for an automated vehicle to identify a pedestrian after it has hit the pedestrian. This topic focused on how to manage this tradeoff. Three breakout groups formed on the second day, focusing on the topics “Benchmarks for RT systems”, “Tradeoff of timeliness and accuracy”, and “Orchestration/coordination languages vs reactive languages.”

### **Day 3: Predictability vs. adaptability**

Reactive systems often perform critical tasks. We need for them to behave predictably during normal operation, but also adapt to behave reasonably in abnormal situations. Recent innovations in machine learning promise significant improvements for the latter requirement, but it is unclear how to reconcile the use of ML with the former requirement. This topic focused on how to manage this tradeoff.

### **Day 4: Security vs. accessibility**

When systems are secure, nothing bad happens even when malicious players are present. Achieving the goal that “nothing bad happens,” however, is trivially easy by ensuring that nothing at all happens. Security measures often get in the way of other goals. This topic will focus on how to manage this tradeoff. For example, techniques that offer tiered access to capabilities, taint analysis, or mixtures of encrypted and unencrypted communication might be explored. During the afternoon session, reports from the group discussions were shared, along with brief previews of the software demonstrations (teasers) scheduled for Day 5.

### **Day 5: Tools and Demos**

This topic focused on tools that support analysis and design and make explicit the management of tradeoffs. The key goal of the groups was to feel the pain intrinsic to the tradeoffs that are the theme of the seminar. A total of five software tools tutorials were presented (see Section 5). Hands-on exercises were organized into two sessions, each consisting of parallel tracks. This structure was designed to give participants the opportunity to experiment with two of the five available tools, rather than one. The tools presented were: “Lingua Franca”, “Timed SCCharts”, “QRML”, “Rebecca”, and “HipHop”.

## 2 Table of Contents

### Executive Summary

<i>Jerónimo Castrillón-Mazo, Chadlia Jerad, Edward A. Lee, and Claire Pagetti . . .</i>	2
---	---

### Overview of Talks

Safety and AI Standards for Automobile: An overview <i>Andres Barrilado . . . . .</i>	7
Modeling reconfigurable CPS using UML <i>Grzegorz Bazydło . . . . .</i>	7
Reactive systems: Optimizations and opportunities in domain-specific computing <i>Jerónimo Castrillón-Mazo . . . . .</i>	7
Formalizing Tradeoffs in Reactive Systems Design <i>Samarjit Chakraborty . . . . .</i>	8
Security Efficiency Tradeoffs for Intelligent Systems <i>Anupam Chattopadhyay . . . . .</i>	11
Assurance of Neural Network-based Safety-Critical Avionics with Formal Methods <i>Arthur Clavière . . . . .</i>	11
A component model and modelling language for tradeoffs and multi objective optimization <i>Marc Geilen . . . . .</i>	12
Exploring the memory / execution time tradeoff in dataflow graphs <i>Alain Girault . . . . .</i>	12
Latency and Consistency Tradeoffs in Shared-Memory Systems <i>Andrés Goens Jokisch . . . . .</i>	13
Can we make COTS CPS Ultra-Reliable? <i>Arpan Gujarati . . . . .</i>	13
Necessary Conditions for Model Engineering to Ensure System Correctness <i>Jérôme Hugues . . . . .</i>	14
Introduction to Avionic Certification <i>Victor Jegu . . . . .</i>	14
A Motivating Example: Distributed Music <i>Erling Rennemo Jellum . . . . .</i>	15
Time-Tarot: Toward a Quantitative Approach to Time-Predictability <i>Chadlia Jerad . . . . .</i>	15
Declarative Lifecycle Management in Digital Twins <i>Einar Broch Johnsen . . . . .</i>	16
Security vs. Accessibility Tradeoffs in Reactive Systems <i>Hokeun Kim . . . . .</i>	16
Consistency vs. Availability in Reactive Systems Design <i>Edward A. Lee . . . . .</i>	17
CAL Theorem in Reactive Systems <i>Shaokai Jerry Lin . . . . .</i>	17

Benchmarking Worst-Case Performance of Real-Time Systems <i>Martin Schoeberl, Erling Jellum, Shaokai Jerry Lin, Chadlia Jerad, Emad Jacob Maroun, Marten Lohstroh, and Edward A. Lee</i> . . . . .	18
Certification of ML-based systems <i>Claire Pagetti and Arthur Clavière</i> . . . . .	18
Synergies between Timing Predictability and Microarchitectural Security <i>Jan Reineke</i> . . . . .	19
Observations on Formalizing Reactive Systems <i>Marcus Rossel</i> . . . . .	19
Timing Tradeoffs in Timed Automata with Dynamic Ticks <i>Alexander Schulz-Rosengarten and Reinhard von Hanxleden</i> . . . . .	19
Fail-operational Systems in Autonomous Driving Applications <i>Katharina Sedow</i> . . . . .	20
The Functional, the Imperative, and the Sudoku <i>Manuel Serrano</i> . . . . .	20
Consistency versus Availability in Redundant Controllers – Formal Verification, Test Design, Time Analysis <i>Marjan Sirjani</i> . . . . .	21
Tradeoffs in Accuracy and Timeliness in Transportation Cyber-Physical Systems <i>Jonathan Sprinkle</i> . . . . .	21
Timeliness vs. Accuracy <i>Lothar Thiele</i> . . . . .	22
Fundamental Tradeoffs in Reactive Systems for Smart Agriculture and Pollutants Detection in Resource-Constrained Environments <i>Eric Tutu Tchao</i> . . . . .	22
Utility-Based System Design: Making Sense of Tradeoffs <i>Eugene Yip</i> . . . . .	23

### Working groups

Can AI be used in critical Systems? <i>Jérôme Hugues, Andres Barrilado, Frédéric Boniol, Thomas Carle, Jerónimo Castrillón-Mazo, Samarjit Chakraborty, Arthur Clavière, Victor Jegu, Claire Pagetti, Marcus Rossel, Selma Saidi, Jonathan Sprinkle, Hasna Bouraoui, and Eugene Yip</i> . . . . .	23
Distributed music challenge <i>Grzegorz Bazydło, Anupam Chattopadhyay, Andrés Goens Jokisch, Chadlia Jerad, Erling Jellum, Einar Broch Johnsen, Hokeun Kim, Edward A. Lee, Shaokai Jerry Lin, Jan Reineke, Manuel Serrano, Martin Schoeberl, Lothar Thiele, and Reinhard von Hanxleden</i> . . . . .	24
Orchestration/coordination languages vs reactive languages <i>Jérôme Hugues, Edward A. Lee, Shaokai Jerry Lin, and Manuel Serrano</i> . . . . .	24
Tradeoffs in accuracy and timeliness <i>Andres Barrilado, Hasna Bouraoui, Anupam Chattopadhyay, Jerónimo Castrillón-Mazo, Arpan Gujarati, Hokeun Kim, Lothar Thiele, Alexander Schulz-Rosengarten, Katharina Sedow, and Jonathan Sprinkle</i> . . . . .	25

Benchmarks for real-time systems  
*Thomas Carle, Samarjit Chakraborty, Arthur Clavière, Victor Jegu, Chadlia Jerad, Claire Pagetti, Jan Reineke, and Martin Schoeberl . . . . .* 29

**Software teasers**

Timed Rebeca  
*Marjan Sirjani . . . . .* 30

Lingua Franca  
*Erling Rennemo Jellum and Edward A. Lee . . . . .* 30

SCCharts  
*Alexander Schulz-Rosengarten . . . . .* 30

QRML  
*Marc Geilen . . . . .* 31

HipHop  
*Manuel Serrano . . . . .* 31

**Participants . . . . .** 32

### 3 Overview of Talks

#### 3.1 Safety and AI Standards for Automobile: An overview

*Andres Barrilado (NXP Semiconductors – Toulouse, FR)*

**License** © Creative Commons BY 4.0 International license  
© Andres Barrilado

**Joint work of** Andres Barrilado, Iban Guinebert

Presentation in which a short summary of recently published safety standards for AI in the automotive domain is shared. Some selected extracts are discussed with regards to tradeoffs linked to predictability and adaptability. Additionally, we provide an introduction to the tradeoffs now being observed specifically by semiconductor manufacturers when having to produce functionally safe inference accelerators.

#### 3.2 Modeling reconfigurable CPS using UML

*Grzegorz Bazydło (University of Zielona Góra, PL)*

**License** © Creative Commons BY 4.0 International license  
© Grzegorz Bazydło

The presentation introduces an approach to designing reconfigurable cyber-physical systems (CPS) using state machine diagrams from the Unified Modeling Language (UML). The UML specification is transformed using model-driven development (MDD) techniques into an efficient hardware description language (HDL) program, utilizing a concurrent finite state machine (CFSM) as an intermediate model. The resulting HDL specification can be analyzed, validated, synthesized, and ultimately implemented in field-programmable gate array (FPGA) devices.

Dynamic partial reconfiguration – a feature of modern FPGAs – enables the replacement of parts of the CPS algorithm without powering down the device. However, to leverage this feature, the model must be safe, which, in the proposed approach, means incorporating special idle states where control is transferred during the reconfiguration process. The CFSM model significantly facilitates this task.

The proposed design method provides an efficient graphical modeling approach for the control part of a reconfigurable CPS, along with an automated translation of the behavior model into a synthesizable Verilog description. This Verilog code can be directly implemented in FPGA devices and dynamically reconfigured as needed. A practical example from the field of demand-side management illustrates the successive stages of the proposed method.

#### 3.3 Reactive systems: Optimizations and opportunities in domain-specific computing

*Jerónimo Castrillón-Mazo (TU Dresden, DE)*

**License** © Creative Commons BY 4.0 International license  
© Jerónimo Castrillón-Mazo

This talk discusses optimizations for dataflow and reactive programs. This includes optimizations to the runtime system of Lingua Franca [5], initial work on decoupling timelines to expose more parallelism [6], a system to explore energy-performance tradeoffs [7, 2], and a

methodology to explore tradeoffs between performance and accuracy for ML workloads [3, 4]. The talk closes with an outlook on how domain-specific programming abstractions and domain-specific computer architectures [1] may offer more control over system-level tradeoffs, including performance, energy consumption, accuracy and reliability.

## References

- 1 Asif Ali Khan, Hamid Farzaneh, Karl F. A. Friebel, Lorenzo Chelini, and Jeronimo Castrillon. Cinm (cinnamon): A compilation infrastructure for heterogeneous compute in-memory and compute near-memory paradigms. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'25)*, ASPLOS '25. Association for Computing Machinery, March 2025.
- 2 Robert Khasanov and Jeronimo 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, pages 909–914. IEEE, March 2020.
- 3 Guilherme Korol, Michael Guilherme Jordan, Mateus Beck Rutzig, Jeronimo Castrillon, and Antonio Carlos Schneider Beck. Design space exploration for CNN offloading to FPGAs at the edge. In *2023 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 1–6, Los Alamitos, CA, USA, June 2023. IEEE, IEEE Computer Society.
- 4 Guilherme Korol, Michael Guilherme Jordan, Mateus Beck Rutzig, Jeronimo Castrillon, and Antonio Carlos Schneider Beck. Pruning and early-exit co-optimization for CNN acceleration on FPGAs. In *Proceedings of the 2023 Design, Automation and Test in Europe Conference (DATE)*, DATE'23, pages 1–6. IEEE, April 2023.
- 5 Christian Menard, Marten Lohstroh, Soroush Bateni, Mathhew Chorlian, Arthur Deng, Peter Donovan, Clément Fournier, Shaokai Lin, Felix Suchert, Tassilo Tanneberger, Hokeun Kim, Jeronimo Castrillon, and Edward A. Lee. High-performance deterministic concurrency using Lingua Franca. *ACM Transactions on Architecture and Code Optimization (TACO)*, 20(4):1–29, August 2023.
- 6 Julian Robledo, Christian Menard, Erling Jellum, Edward A. Lee, and Jeronimo Castrillon. Timing enclaves for performance in Lingua Franca. In *2024 Forum for Specification and Design Languages (FDL)*, pages 1–9, September 2024.
- 7 Till Smejkal, Robert Khasanov, Jeronimo Castrillon, and Hermann Härtig. E-Mapper: Energy-efficient resource allocation for traditional operating systems on heterogeneous processors, 2024.

## 3.4 Formalizing Tradeoffs in Reactive Systems Design

*Samarjit Chakraborty (The University of North Carolina (UNC) at Chapel Hill, US)*

License © Creative Commons BY 4.0 International license  
© Samarjit Chakraborty

Today, most complex reactive systems, and especially autonomous systems, are composed of multiple subsystems such as controllers, schedulers, machine learning (ML) components, and security systems. These subsystems are designed independently and are aimed to work to perfection. But in reality they do not work perfectly and the imperfection is assumed to be small enough that it may be ignored. We argue that design approaches should make the imperfections in these components first class citizens, formalize specifying imperfections, and allow tradeoffs in the design play a more prominent role.

Towards this, we first define a flexible notion of system-level safety to admit imperfect behaviors. We then compute which component behaviors would ensure such a notion system safety. This allows more component behaviors compared to those that would be admitted



if the component was to be designed to perfection. These additional behaviors allow more efficient component implementations. Further, they allow exploration of tradeoffs where the deficiency of one component may be compensated by another component, while satisfying the notion of system-level safety and the flexibility it allows. To arrive at a general notion of imperfection, we consider the dynamics of closed-loop control systems. First, we determine the trajectory followed by such systems in their state space, when all the system components behave perfectly, *e.g.*, when all the elements of the system state may be estimated exactly, all machine learning components in the system always return accurate inferences, and the deadlines of all software/control tasks are always met. For any given initial condition, such a system trajectory represents the *ideal* behavior of the system, which might not be practically realizable. To allow imperfect system behaviors, we consider a *safety pipe* around this ideal trajectory. Any system trajectory that lies within this safety pipe is considered to be an acceptable or safe behavior of the system [4].

It is, however, non-trivial to estimate component behaviors from such specifications of safety pipes that capture system-level behaviors. In this talk we illustrated how this may be done for the specific case of imperfect timing behaviors. In particular, we asked: Given a safety pipe specifying allowed system behaviors, what are feasible timing behaviors that a control task may experience? Our procedure to answer this question involves the following two steps: (1) Guess a regular language  $\mathcal{L}$  of timing behaviors that a control task may experience [1]. Such a language is a set of strings over the alphabet  $\Sigma = \{0, 1\}$ . Here, 0 represents the case where the control task in question misses its deadline at the specified sampling period, *i.e.*, the control input is not available at the time of actuation and an old control input is used. On the other hand, 1 denotes the case that the control task successfully completes its execution before the predetermined time of actuation. Hence, a string 101010... denotes that the case where every alternate execution of the control task misses its deadline. Equivalently, it also denotes the case where the control task is purposefully not scheduled in every alternate sampling period to save computation bandwidth.

In the next step: (2) We use approximate reachability analysis over a bounded time horizon to check whether all the trajectories of the given closed loop system stay within the specified safety pipe, when the control task is subjected to any timing behavior that belongs to the language  $\mathcal{L}$ . Here, we use *approximate* reachability to contain the state-space explosion that would otherwise happen. If the language  $\mathcal{L}$  of timing behaviors result in safe closed-loop behaviors (*i.e.*, all the closed-loop trajectories are within the safety pipe) then we add  $\mathcal{L}$  to the set of safe languages, *viz.*,  $\mathcal{L}_{safe} = \mathcal{L}_{safe} \cup \mathcal{L}$ , where  $\mathcal{L}_{safe}$  was initially empty. By iteratively following this procedure, we create a set of timing behaviors  $\mathcal{L}_{safe}$  that results in safe closed-loop system behaviors. Now, since the union of regular languages is regular,  $\mathcal{L}_{safe}$  may be represented as a finite state automata. We can use this procedure to synthesize safe schedules for a set of control tasks. For this, we rely on automata-theoretic techniques [5, 9] to combine the  $\mathcal{L}_{safe}$ s of all the controllers to synthesize schedules where the tasks of each controller is not always scheduled for execution (*i.e.*, occasionally misses its deadline), but the safety of all the closed-loop systems is guaranteed [10]. Such schedules account for the dynamics of the closed-loop system [2, 3, 6] and cannot be reproduced by standard scheduling policies like fixed priority or earliest deadline first (EDF) [7]. This approach of using a flexible notion of system safety to allow multiple (imperfect) component behaviors allows the exploration of different tradeoffs between the performance of different system components. For example, the work in [11] uses the size of the reachable set as a measure of safety and shows how the size of a neural network – and therefore its inference quality – for state estimation in a closed-loop control system may be traded off with the

magnitude of the control input. This is used to partition a deep neural network (DNN) between edge and cloud resources. A similar approach was also used [8] to purposefully choose DNNs with less than optimal size and inference accuracy, to fit multiple DNNs on the same shared graphics processing unit (GPU), while ensuring that the dynamics of the closed-loop systems using these DNN inferences remain safe.

While a safety pipe around an ideal system trajectory, or the size of the reachable set, are two different notions of system safety, there are many other possibilities. For example, not all trajectories within a safety pipe might be permissible. Most work on formal methods till date have focused on specifying perfect or ideal system behaviors (*e.g.*, all deadlines are met). Specifying what kind of imperfect behaviors might be acceptable requires domain knowledge, and those with such knowledge are usually not sufficiently conversant with formal methods to be able to write formal specifications. Hence, new avenues for involving domain experts – who are not specialists in formal methods – to write formal specifications, perhaps using tools from machine learning will be needed.

## References

- 1 Bineet Ghosh, Clara Hobbs, Shengjie Xu, F. Donelson Smith, James H. Anderson, P. S. Thiagarajan, Benjamin Berg, Parasara Sridhar Duggirala, and Samarjit Chakraborty. Statistical verification of autonomous system controllers under timing uncertainties. *Real Time Syst.*, 60(1):108–149, 2024.
- 2 Dip Goswami, Reinhard Schneider, and Samarjit Chakraborty. Co-design of cyber-physical systems via controllers with flexible delay constraints. In *16th Asia South Pacific Design Automation Conference (ASP-DAC)*, 2011.
- 3 Dip Goswami, Reinhard Schneider, and Samarjit Chakraborty. Re-engineering cyber-physical control applications for hybrid communication protocols. In *Design, Automation and Test in Europe (DATE)*, 2011.
- 4 Clara Hobbs, Bineet Ghosh, Shengjie Xu, Parasara Sridhar Duggirala, and Samarjit Chakraborty. Safety analysis of embedded controllers under implementation platform timing uncertainties. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 41(11):4016–4027, 2022.
- 5 Clara Hobbs, Shengjie Xu, Bineet Ghosh, Enrico Fraccaroli, Parasara Sridhar Duggirala, and Samarjit Chakraborty. Quantitative safety-driven co-synthesis of cyber-physical system implementations. In *15th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPs)*, pages 99–110. IEEE, 2024.
- 6 Martin Lukasiewicz, Reinhard Schneider, Dip Goswami, and Samarjit Chakraborty. Modular scheduling of distributed heterogeneous time-triggered automotive systems. In *17th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2012.
- 7 Florian Sagstetter, Sidharta Andalām, Peter Waszecki, Martin Lukasiewicz, Hauke Stähle, Samarjit Chakraborty, and Alois C. Knoll. Schedule integration framework for time-triggered automotive architectures. In *The 51st Annual Design Automation Conference (DAC)*, 2014.
- 8 Shengjie Xu et al. GPU partitioning & neural architecture sizing for safety-driven sensing in autonomous systems. In *IEEE International Conference on Assured Autonomy (ICAA)*, 2024.
- 9 Shengjie Xu, Bineet Ghosh, Clara Hobbs, P. S. Thiagarajan, and Samarjit Chakraborty. Safety-aware flexible schedule synthesis for cyber-physical systems using weakly-hard constraints. In Atsushi Takahashi, editor, *28th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2023.
- 10 Anand Yeolekar, Ravindra Metta, Clara Hobbs, and Samarjit Chakraborty. Checking scheduling-induced violations of control safety properties. In *20th International Symposium on Automated Technology for Verification and Analysis (ATVA)*, volume 13505 of *Lecture Notes in Computer Science*. Springer, 2022.

- 11 Tingan Zhu, Prateek Ganguli, Arkaprava Gupta, Shengjie Xu, Luigi Capogrosso, Enrico Fraccaroli, Marco Cristani, Franco Fummi, and Samarjit Chakraborty. Controllers for edge-cloud cyber-physical systems. In *17th IEEE International Conference on COMmunication Systems and NETworks (COMSNETS)*, 2025.

### 3.5 Security Efficiency Tradeoffs for Intelligent Systems

*Anupam Chattopadhyay (Nanyang TU – Singapore, SG)*

**License** © Creative Commons BY 4.0 International license  
© Anupam Chattopadhyay

**Joint work of** Anupam Chattopadhyay, Prasanna Ravi, Sourav Sen Gupta, Mustafa Khairallah, Zakaria Najm, Shivam Bhasin, Arpan Jati, Naina Gupta, Anh Tu Ngo

Security and privacy are growing concerns across all forms of digital systems, including Cyber-Physical Systems (CPS). Embedding intelligence in such systems are done to enhance performance and autonomy. Such an inclusion compromises security, which is often fixed as an afterthought. The talk highlights several instances across the stack of digital systems, from machine learning, general purpose processors to hardened cryptographic implementations, which showcase that security and efficiency present contrasting choices for a system designer. This situation is exacerbated for autonomy, where Artificial Intelligence (AI) is invoked. Lack of explanation in many AI systems directly falls into the hands of a malicious actor, who can compromise the system in many subtle ways.

### 3.6 Assurance of Neural Network-based Safety-Critical Avionics with Formal Methods

*Arthur Clavière (Collins Aerospace – Blagnac, FR)*


**License** © Creative Commons BY 4.0 International license  
© Arthur Clavière

**Joint work of** Arthur Clavière, Dmitrii Kirov, Darren Cofer

We present a process for the Verification of neural networks using formal methods, developed to support the needs of Collins engineers working on neural networks for aircraft. The process focuses on low-complexity neural networks (having hundreds or thousands of learnable parameters) with simple input domains (less than 10 input features). It is especially applicable when the neural network is used to approximate a given target function (such as a complex optimization or large look-up table) at a lower computational cost. It compares the behavior of the neural network to the target function by exhaustively verifying the entire input space. Such an exhaustive verification, made possible by using formal methods, analyzes how the neural network would respond not only on the training or test dataset but also to any unseen data in its input range. We anticipate that this approach will be useful for high-criticality neural network-based components (DAL-C and above). It can be used as a means of compliance with certification objectives e.g., stability analysis and generalization capability assessment as defined by the European Union Aviation Safety Agency (EASA). The process is illustrated through a Collins use case, namely the Recommended Cruise Level (RCL) function, which can be used to provide pilots with a suggested altitude to reduce the operational cost of the flight. The process is used to verify a neural network implementation of the RCL, that significantly reduces the computational cost and memory footprint of a traditional implementation.

### 3.7 A component model and modelling language for tradeoffs and multi objective optimization

*Marc Geilen (TU Eindhoven, NL)*

License  Creative Commons BY 4.0 International license

© Marc Geilen

Joint work of Marc Geilen, Twan Basten

Tradeoffs represent choices between multiple objectives that are determined by choices between different configurations. This presentation introduces a mathematical framework, a component model and a domain-specific language. The mathematical framework captures the essential operations and requirements to support compositional computation of Pareto optimal configurations of a system from descriptions of the (optimal) configurations of components. It defines a number of relations on configurations, the sets of possible configurations of components or systems to characterize which ones are considered better, worse, or equivalent. The framework provides insight in requirements on the models and operations to work effectively, preserve optimality, etcetera. We further introduce a component model for Quality and Resource Management that builds on Pareto Algebra. It aims to describe application and platform components in a virtualized platform setting. Components are described in terms of input and out qualities, their required and provided budgets, their optimization objectives and the parameters used to configure the component. The resulting system semantics is a set of (Pareto optimal) configurations of the overall system. This set can be implicitly captured as a set of equations in terms of the combined system parameters. We have defined a Domain-Specific Language, called QRML, to specify such component models generated different types of visualization and generate the set of constraints to be checked for satisfiability, or for optimization, by a constraint solver, such as the Z3 SMT solver. The tools are available through the web page <https://www.qrml.org>.

### 3.8 Exploring the memory / execution time tradeoff in dataflow graphs

*Alain Girault (INRIA – Grenoble, FR)*

License  Creative Commons BY 4.0 International license

© Alain Girault

Joint work of Alain Girault, Pascal Fradet, Alexandre Honorat

Many computing systems are constrained by a fixed amount of available shared memory. Modeling applications with task graphs makes it possible to analyze and optimize their memory usage. The NP-complete problem studied here is finding a parallel schedule of a given task graph that minimizes its memory peak.

Our first contribution is an algorithm that finds optimal sequential schedules for a dataflow task graph. This algorithm is based on graph transformation rules. On a large class of graphs, it is able to compress the graph into a single node which contains a sequential schedule optimal w.r.t. the memory peak. The approach also applies to SDF graphs after converting them into task graphs. However, since that conversion may produce very large graphs, we also propose a new suboptimal method, similar to Partial Expansion Graphs, to reduce the problem size. We evaluate our approach on classic benchmarks, on which we always outperform the state-of-the-art.

From this optimal sequential schedule, our second contribution is a dynamic parallel schedule, which consists of a ready list scheduling that we adapt to take into account memory requirements. Our approach always produces a parallel schedule that meets the constraints

and enjoys very good speedups. It can also be applied to less harsh memory constraints, leading to more substantial speedups. We compare with alternative approaches on multiple applications expressed as task graphs (scientific workflows, signal processing filters). When memory constraints are close to their minimum, our approach always succeeds in finding a parallel schedule meeting the given constraints, whereas the other approaches mostly fail. When memory constraints are significantly higher, our results are comparable to others for speedup. Furthermore, our approach is faster and can deal with very large task graphs (up to 50,000 nodes) using a naive Python implementation.

### 3.9 Latency and Consistency Tradeoffs in Shared-Memory Systems

*Andrés Goens Jokisch (University of Amsterdam, NL)*

License  Creative Commons BY 4.0 International license  
© Andrés Goens Jokisch

Shared-memory systems are a reality of today’s hardware, and current trends are to increase their scale and complexity, with heterogeneous shared-memory architectures and hierarchical systems with protocols like CXL. These systems, however, have complex behaviors that trade consistency with latency, through out-of-order execution and weak guarantees in the protocols, resulting in weak memory consistency models, i.e. models weaker than sequential consistency. In this talk, I present some of the architectural guarantees, and open questions in this context, which give us primitives to express and control tradeoffs in consistency and latency. I argue we should build higher-level abstractions on top of these primitives to reason about these tradeoffs at a higher level of abstraction in the design process, e.g. programming languages or runtime systems.

### 3.10 Can we make COTS CPS Ultra-Reliable?


*Arpan Gujarati (University of British Columbia – Vancouver, CA)*

License  Creative Commons BY 4.0 International license  
© Arpan Gujarati

In the avionics domain, “ultra-reliability” refers to the practice of ensuring quantifiably negligible residual failure rates in the presence of transient and permanent hardware faults. In this talk, I discuss the need for new mechanisms that can help us make contemporary and next-generation CPS, which use inexpensive, relatively unreliable off-the-shelf components, more reliable and, if possible, ultra-reliable (as airplanes). I will frame the discussion around the problem of Byzantine fault tolerance (BFT). Specifically, I will briefly present our RTSS 2022 work, where we proposed a novel BFT timed key-value store, called Achal, designed specifically for Ethernet-based distributed real-time control applications. I will then describe our current work on understanding two key limitations of Achal-like systems. First, can they be seamlessly ported to real, complex CPS applications? Second, can precision time synchronization be considered a reliable primitive? Finally, I wrap up with my key takeaway, that we must focus on systems / middleware / tools that make it easier to engineer reliable systems, as opposed to designing new fault-tolerance protocols from scratch.

### 3.11 Necessary Conditions for Model Engineering to Ensure System Correctness

*Jérôme Hugues (Carnegie Mellon University – Pittsburgh, US)*

License  Creative Commons BY 4.0 International license  
© Jérôme Hugues

Engineering Reactive Systems primarily involves evaluating the accuracy of computations, as well as the system's quality attributes (such as safety, security), or its functions or data (such as availability, consistency). This list is not exhaustive, and neither is this mapping. These properties can be assessed by testing the final system. A more timely and cost-effective approach is to rely on models and verification and validation (V&V) techniques such as simulation or model checking. Evaluating these quality attributes necessitates the construction of an additional system: a model and a demonstration that the model is sufficiently accurate to both represent the system and support a verification objective.

In this presentation, I propose some considerations in the scope of Dagstuhl Seminar 25091. Firstly, when considering a model as a system, it is crucial to define its requirements. The term “capturing a system behavior” is too ambiguous. A model should support an engineering goal, such as evaluating a system attribute. Therefore, the selection of a modeling language, the creation of the model, and the property to be evaluated cannot be separated. It is essential to ensure that the evaluation of an attribute or a tradeoff analysis among multiple attributes can be initially expressed in a model and then analyzed by some techniques. This can only be achieved if the modeling language is sound, the model captures a domain taxonomy, and, of course, it is correct.

### 3.12 Introduction to Avionic Certification

*Victor Jegu (Airbus S.A.S. – Toulouse, FR)*

License  Creative Commons BY 4.0 International license  
© Victor Jegu

Before aircrafts are approved to carry passengers, the industry has to provide evidence to the aviation authorities of the safety and conformity of their products. This approval is called “certification”. And is achieved by conducting a very thoroughly documented process of addressing every foreseeable failure with safety impact. This process is particularly stringent with failures of catastrophic or hazardous consequences. This talk briefly describes this process for avionic equipment (computers). For functions with minor safety impact, COTS or COTS-like “Design Assurance Level” (DAL) could be considered. At such a level, low energy (edge) AI technology could already be considered. But for higher Assurance Level, achieving this objective may require a thorough understanding of the technology, exhaustive (including formal) design checking, use of diverse strategies for error detection, and resource redundancy. The purpose of this talk is to give academics with intent to assist industry by providing tools, methods, languages and IP in general, an understanding of the level and nature of information (documentation and support) the industry needs to adopt their proposals.

### 3.13 A Motivating Example: Distributed Music

*Erling Rennemo Jellum (University of California – Berkeley, US)*

**License** © Creative Commons BY 4.0 International license  
© Erling Rennemo Jellum

**Joint work of** Edward A. Lee, Erling Jellum

This talk will introduce the Distributed Rhythm Generator, a distributed real-time program written in Lingua Franca. Distributed real-time systems are intriguingly challenging systems, as they must balance two opposing requirements. First, they must often only actuate based on a consistent view of the shared variables of the overall system, and second, they must actuate in a timely manner. Without time-predictable networks, reconciling these requirements might not be possible and a tradeoff must be made. We will show how the reactor-oriented programming paradigm introduced by Lingua Franca, enables an elegant specification of such consistency-available tradeoffs.

### 3.14 Time-Tarot: Toward a Quantitative Approach to Time-Predictability

*Chadlia Jerad (University of Manouba, TN)*

**License** © Creative Commons BY 4.0 International license  
© Chadlia Jerad

**Joint work of** Chadlia Jerad, Martin Schoeberl, Emad Jacob Maroun, Edward A. Lee, Shaokai Lin, Erling Jellum

The current trends in hardware design aim at improving the average performance. Together with the increase in software complexity, this trajectory runs contrary to time-predictability, especially when designing safety-critical and real-time systems, as timing requirements need to be guaranteed a priori. Literature is rich with valuable attempts to define and measure time-predictability. Still, a quantitative approach that covers all the different features at the different abstraction levels is missing. This talk suggests the design of a concept map for time-predictability levels. It is based on results from different research papers and is meant to be extensible to accommodate future inventions. It is also derived from expert knowledge in the different fields. The concept map, when combined with multi-criteria decision techniques, can formalize the choice among two or more designs. The techniques to be used are the Brown-Gibson Model and the Analytic Hierarchy Process. The former has the advantage of combining objective and subjective factors that immediately derive from the concept map. Because weighted scoring is required in this method, we use the Analytic Hierarchy Process to weigh the features among each other. A challenge though is about the support of multi-processor systems with different architectures. In addition, the use of the worst-case performance can be decided to be used either as a validation tool or to perform fusion of the results.

### 3.15 Declarative Lifecycle Management in Digital Twins

Einar Broch Johnsen (*University of Oslo, NO*)

**License** © Creative Commons BY 4.0 International license  
© Einar Broch Johnsen

**Joint work of** Einar Broch Johnsen, Eduard Kamburjan, Nelly Bencomo, Silvia Lizeth Tapia Tarifa

Together, a digital twin and its physical counterpart can be seen as a self-adaptive system: the digital twin monitors the physical system, updates its own internal model of the physical system, and adjusts the physical system by means of controllers in order to maintain given requirements. As the physical system shifts between different stages in its lifecycle, these requirements, as well as the associated analyzers and controllers, may need to change. The exact triggers for such shifts in a physical system are often hard to predict, as they may be difficult to describe or even unknown; however, they can generally be observed once they have occurred, in terms of changes in the system behavior. This talk presents an automated method for self-adaptation in digital twins to address shifts between lifecycle stages in a physical system, based on recent work [1]. Our method is based on declarative descriptions of lifecycle stages for different physical assets and their associated digital twin components. Declarative lifecycle management provides a high-level, flexible method for self-adaptation of the digital twin to reflect disruptive shifts between stages in a physical system.

#### References

- 1 Eduard Kamburjan, Nelly Bencomo, Silvia Lizeth Tapia Tarifa, and Einar Broch Johnsen. Declarative lifecycle management in digital twins. In *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems, MODELS Companion '24*, page 353–363, New York, NY, USA, 2024. Association for Computing Machinery.

### 3.16 Security vs. Accessibility Tradeoffs in Reactive Systems

Hokeun Kim (*Arizona State University – Tempe, US*)

**License** © Creative Commons BY 4.0 International license  
© Hokeun Kim

In this framing discussion, I explore the fundamental tradeoffs between security and accessibility in designing reactive and cyber-physical systems (CPS). This discussion examines how secure access mechanisms – such as authentication, encryption, detection, and mitigation – must balance with the need for accessibility. Through real-world examples and case studies, I highlight the challenges of designing secure reactive systems and CPS while managing the tradeoffs between security and accessibility, considering threat models, costs, and security requirements of the target systems. Various design decisions, including connectivity, encryption strategies, and the choice between prevention and detection of cyber threats, are analyzed in the context of modern threat models. This framing discussion also presents a methodology for enforcing security without sacrificing essential accessibility, ensuring the domain-specific reactive systems and CPS are secure and protected yet accessible.



### 3.17 Consistency vs. Availability in Reactive Systems Design

*Edward A. Lee (University of California – Berkeley, US)*

License  Creative Commons BY 4.0 International license  
© Edward A. Lee

Joint work of Edward A. Lee, Ravi Akella, Soroush Bateni, Shaokai Lin, Marten Lohstroh, Christian Menard

Distributed software systems often require consistent shared information. For example, connected vehicles require agreement on access to an intersection before entering the intersection. It is far from trivial, however, how to achieve consistency, or even how to define it rigorously enough to know when it has been achieved. In this talk, I will show how strong and weak forms of consistency can be defined, how software infrastructure can provide reasonable guarantees and efficient implementations, and what are the fundamental costs of achieving consistency that no software system can avoid. Specifically, I will outline the CAL theorem, which quantifies consistency, availability, and latency, and gives an algebraic relation that shows that as latency increases, either availability or consistency or both must decrease. I will describe a coordination language called Lingua Franca that enables programmers to explicitly work with the tradeoffs between these three quantities.

### 3.18 CAL Theorem in Reactive Systems

*Shaokai Jerry Lin (University of California – Berkeley, US)*

License  Creative Commons BY 4.0 International license  
© Shaokai Jerry Lin

Joint work of Shaokai Jerry Lin, Edward A. Lee

In 2000, Eric Brewer (UC Berkeley) famously stated the CAP theorem during the PODC keynote. However, the tradeoffs mentioned in the theorem was not formally specified. Lee et al. [1] later proposed the CAL theorem, which establishes a mathematical relationship among the three properties: Consistency, Availability, and Latency (extending Partitioning in the CAP theorem). In this talk, we present a work-in-progress formalism based on the CAL theorem, which aims to 1) be easily applicable to programming frameworks, and 2) capture the behavior of a reactive system over time. We demonstrate the formalism on a simple stock exchange application written in the Lingua Franca (LF) coordination language. We derive timing constraints from the semantics of the LF language and the application, formulating a system of equations calculating earliest firing times of reactions from previous firing times and asynchronous inputs. We represent the system of equations into a linear difference equation using the max-plus algebra and show that the eigenvalue of a coefficient matrix represents the threshold of the system's responsiveness.

#### References

- 1 Edward A. Lee, Ravi Akella, Soroush Bateni, Shaokai Lin, Marten Lohstroh, and Christian Menard. Consistency vs. availability in distributed cyber-physical systems. *ACM Trans. Embed. Comput. Syst.*, 22(5s), September 2023.

### 3.19 Benchmarking Worst-Case Performance of Real-Time Systems

*Martin Schoeberl (Technical University of Denmark – Lyngby, DK), Erling Jellum (University of California – Berkeley, US), Shaokai Lin (University of California – Berkeley, US), Chadlia Jerad (University of Manouba, TN), Emad Jacob Maroun (Technical University of Denmark – Lyngby, DK), Marten Lohstroh (University of California – Berkeley, US), and Edward A. Lee (University of California – Berkeley, US)*

**License** © Creative Commons BY 4.0 International license

© Martin Schoeberl, Erling Jellum, Shaokai Jerry Lin, Chadlia Jerad, Emad Jacob Maroun, Marten Lohstroh, and Edward A. Lee

Real-time systems rely on tasks with well-defined worst-case execution times (WCET) to ensure predictable behavior. To facilitate accurate WCET estimation, some researchers advocate for processor architectures that simplify timing analysis, known as precision-timed or time-predictable architectures. However, the precise definition and quantification of time predictability remain open questions. This talk examines the concept of time predictability and the challenges in measuring it. Instead of focusing solely on architectural properties, we argue that worst-case performance should be evaluated as a combined property of the processor, compiler, and WCET analysis tool. To enable systematic evaluation, we propose to standardize on a benchmark suite for assessing time-predictable processors, compilers, and WCET analysis tools. We define worst-case performance as the geometric mean of WCET bounds across this benchmark set.

### 3.20 Certification of ML-based systems

*Claire Pagetti (ONERA – Toulouse, FR) and Arthur Clavière (Collins Aerospace – Blagnac, FR)*


**License** © Creative Commons BY 4.0 International license

© Claire Pagetti and Arthur Clavière

The purpose of the talks was to present some of the results reached for introducing ML (Machine Learning) algorithms in Airborne systems. First a brief introduction on certification and on current drafted standards for ML was done. Indeed, EASA is writing roadmaps and preliminary guidance. SAE/EUROCAE working group is in parallel writing a future ED/ARP document. The presentation then has dug to two use cases. The first was the simplest: how to replace a set of LUT (look-up tables) – here for the avoidance collision system ACAS Xu – with neural networks while preserving the intended function. For that, formal verification was successfully applied. These kinds of surrogate models to compress existing code is a way forward. The second case is exploratory: vision-based perception algorithm to detect runway during an aircraft landing. Again formal verification was successfully applied to check robustness against some foreseeable perturbations. These preliminary results open the path for certification.

### 3.21 Synergies between Timing Predictability and Microarchitectural Security

*Jan Reineke (Universität des Saarlandes – Saarbrücken, DE)*

**License**  Creative Commons BY 4.0 International license


© Jan Reineke

**Joint work of** Marco Guarnieri, Gideon Mohr, Zilong Wang, Klaus v. Gleissenthall, Jan Reineke

Timing attacks exploit variations in execution time to leak sensitive information. Hardware-software leakage contracts are a new security abstraction that augments the instruction-set-architecture (ISA) to capture microarchitectural leakage at software level. In recent work we have shown how to verify that open-source RISC-V processors satisfy particular contracts and how to synthesize precise leakage contracts for a given processor. In this talk, I motivate and introduce leakage contracts and I discuss two equivalent notions of contract satisfaction that suggest synergies between research on timing predictability and microarchitectural security.

### 3.22 Observations on Formalizing Reactive Systems

*Marcus Rossel (Barkhausen Institut – Dresden, DE)*

**License**  Creative Commons BY 4.0 International license

© Marcus Rossel

Reactive systems have a rich history of rigorous modeling and verification techniques. As modern verification efforts become more ambitious, they require the composition of a growing number of techniques. This composition necessitates interfacing between different formalisms and tools, which can lead to error-prone discontinuities in the verification pipeline. While there exist systems for coherently verifying reactive systems, they are limited in expressivity. Interactive theorem provers (ITPs) provide a means of verification with virtually unbounded expressiveness, but do not natively capture the semantics of notions like concurrency and mutability. We discuss the problems encountered when attempting gap-free verification of reactive systems in ITPs.

### 3.23 Timing Tradeoffs in Timed Automata with Dynamic Ticks

*Alexander Schulz-Rosengarten (Universität Kiel, DE) and Reinhard von Hanxleden (Universität Kiel, DE)*

**License**  Creative Commons BY 4.0 International license

© Alexander Schulz-Rosengarten and Reinhard von Hanxleden

Synchronous languages build on an abstraction from physical execution time by dividing the execution into logical ticks. However, they say little about when to execute the ticks and traditionally lack built-in support for physical time. This makes it rather cumbersome to express things like time-outs or periodic executions. We present Timed SCCharts that use a timed automata formalism and combine it with the concept of dynamic ticks. This enables specifying time-dependent behavior while addressing tradeoffs in timing in the face of imperfections of execution with physical time.

One application area we plan to explore is the railway domain, as part of the REAKT initiative. This initiative aims to revitalize rural rail lines, using the line Bad Malente – Lütjenburg as real world laboratory. As an aside, in Germany alone, about 5000 km of rural train lines have been put out of service since the 1990s. This corresponds to about 30% of the total network, and it directly affects more than 3 million citizens.

### 3.24 Fail-operational Systems in Autonomous Driving Applications

*Katharina Sedow (Saneon GmbH – Ismaning, DE)*

License © Creative Commons BY 4.0 International license  
© Katharina Sedow

As the automotive research and development field approaches automated driving levels SAE L4 and L5, the introduction of the requirement for the driving function to continue operation when an abnormal system behavior occurs becomes crucial. Highly automated and autonomous systems cannot use a human driver as the fallback decision-maker and actuator, and, in order to be safe, they must possess the ability to function until the vehicle reaches a safe state. This talk briefly introduces the concept behind fail-operational systems. First, we introduce the advances of such systems when compared to the fail-safe implementations. We further provide an example of a fail-operational architecture. In the last part of the talk, we discuss the challenges the automotive industry faces when developing fail-operational driving functionality. They include the assurance of independent redundancy, tradeoffs in performance and safety, as well as the development costs. Finally, we address the approaches intended to handle those issues, such as redundant system design, the introduction of monitoring safety components, and the handling of the high safety integrity levels.

### 3.25 The Functional, the Imperative, and the Sudoku

*Manuel Serrano (INRIA – Sophia Antipolis, FR)*


License © Creative Commons BY 4.0 International license  
© Manuel Serrano  
Joint work of Manuel Serrano, Robert Bruce Findler

Conventional wisdom suggests that the benefits of functional programming no longer apply in the presence of even a small amount of imperative code, as if the addition of imperative code effectively subtracts. And yet, as we show in this talk, combining functional programming with the special imperative language Esterel provides a multiplicative improvement to the benefits of functional programming.

To illustrate these benefits, the bulk of this talk consists of an in-depth exploration of HipHop code (a mashup of JavaScript and Esterel) that implements a Sudoku solver, showing how it is possible to write code that is as easy to understand as if it were written in a pure functional programming style, even though it uses multiple threads, mutable state, thread preemption, and even thread abortion. Even better, concurrent composition and task canceling provide significant program structuring benefits that allow a clean decomposition and task separation in the solver.

### 3.26 Consistency versus Availability in Redundant Controllers – Formal Verification, Test Design, Time Analysis


*Marjan Sirjani (Mälardalen University – Västerås, SE)*

License  Creative Commons BY 4.0 International license  
© Marjan Sirjani

A potential problem that may arise in the domain of distributed control systems is the existence of more than one primary controller in redundancy plans which may lead to inconsistency. We worked on an algorithm called NRP FD (Network Reference Point Failure Detection), proposed by industry, to solve this issue by prioritizing consistency over availability. I explain how by using modeling and formal verification, we discovered an issue in NRP FD where we may have two primary controllers at the same time. We then provide a solution to mitigate the identified issue, thereby enhancing the robustness and reliability of such systems. In the same context, I will also show how we used model checking for making informed decisions in designing test cases for fault-tolerant systems. I will add a discussion on how this approach may be generalized in different contexts.

### 3.27 Tradeoffs in Accuracy and Timeliness in Transportation Cyber-Physical Systems

*Jonathan Sprinkle (Vanderbilt University – Nashville, US)*

License  Creative Commons BY 4.0 International license  
© Jonathan Sprinkle

Joint work of Jonathan Sprinkle, George Gunter, Daniel B. Work, the CIRCLES Consortium


This talk explores the timing and accuracy tradeoffs from application problems in transportation cyber-physical systems. The talk demonstrates previous proof of dampening traffic waves in closed road scenarios, and evidence that existing adaptive cruise controllers amplify (rather than reduce) traffic waves. To address these issues, the work discusses the use of hierarchical high-latency and low-latency controllers, which were shown to be successful. These approaches utilize shared information of the road network that may be gathered over recent minutes, for the purposes of reducing stop and go traffic waves. Given the dynamics of freeway traffic, where speeds of up to 35 m/s regularly encounter stop and go waves traveling at -4 m/s (i.e., against the flow), the lookahead of the on-vehicle sensors is insufficient to dampen these waves, and external information is needed. Using only local sensors it is possible to avoid collisions, but it is not trivial to also dampen the traffic waves. Thus it is important to understand how to share data on downstream state information, with timing information to help understand how to interpret those data. The talk transitions to discussion on how to design and utilize safety envelope controllers for future experiments, in which AI or other untrusted controllers are fielded in open road scenarios. While these controllers can be theoretically shown to be safe, the theory requires information from vehicle sensors (such as derivatives of velocity or acceleration) which are difficult to obtain, and for which filtering delays the availability of the data – requiring an additional following gap for safety that may reduce the effectiveness of the approach. The conclusion calls for continued exploration in determining the accuracy of these distributed systems, which are loosely coordinated, when it is impossible to recreate the situations in which they were fielded. Further, it calls for

understanding how one can safely field AI-based controllers to gather field training data, when there may be significant limitations in understanding how safe the safety envelopes are for general use.

This work is supported by the US National Science Foundation, and the US Department of Energy.

### 3.28 Timeliness vs. Accuracy

*Lothar Thiele (ETH Zürich, CH)*

License  Creative Commons BY 4.0 International license  
© Lothar Thiele

This talk surveys the research landscape surrounding the fundamental tradeoff between timeliness and accuracy in complex systems. It explores formal and informal approaches to managing this tradeoff from multiple perspectives, including neuroscience and psychology, scheduling theory, algorithm design, system architecture, and multi-agent systems. Connecting research from the late 1980s to cutting-edge developments, the talk examines how current abstractions, particularly those related to accuracy (e.g., algorithmic precision) and timeliness (e.g., system timing requirements), hold up in the face of increasing system complexity. A core question explored is whether these abstractions remain adequate, or if we need fundamentally new approaches beyond reliance on testing and benchmarking in real-world environments, to effectively address the timeliness-accuracy tradeoff across algorithms, runtime systems (including hardware), and requirements. Finally, the talk identifies key open research questions in this area.

### 3.29 Fundamental Tradeoffs in Reactive Systems for Smart Agriculture and Pollutants Detection in Resource-Constrained Environments

*Eric Tutu Tchao (Kwame Nkrumah University of Science and Technology, GH)*

License  Creative Commons BY 4.0 International license  
© Eric Tutu Tchao

In resource-constrained environments like rural Ghana, designing reactive systems for critical applications, such as smart agriculture and detecting pollutants from illegal mining activities, demands confronting unavoidable tradeoffs. These systems must balance competing priorities shaped by limited energy, connectivity, and financial resources. This has forced researchers to examine how fundamental engineering limits force compromises between accuracy, timeliness, security, and scalability. Researchers cannot eliminate compromises but formalize them, embedding transparency into system architecture. By doing so, they create solutions that are survivable, equitable, and aligned with the pragmatic realities of communities like Ghana's – where “good enough” is not a concession but a necessity. My presentation at Dagstuhl Seminar 25091 illustrates a broader truth that in resource-constrained environments, tradeoffs are not design failures but fundamental constraints.

### 3.30 Utility-Based System Design: Making Sense of Tradeoffs

*Eugene Yip (GLIWA – Weilheim, DE)*

License  Creative Commons BY 4.0 International license  
© Eugene Yip


Joint work of Eugene Yip, Gerald Lüttgen

Selecting software components to reuse as a means to reduce development time and costs can be challenging, especially when multiple components provide the same functionality but offer different non-functional properties, e.g., numerical accuracy, memory footprint, power rating, timeliness, robustness, security, or financial cost. In this talk, we seek to model the benefit that a system gains from a component’s non-functional properties using so-called “Property-Utility Functions” (PUFs) as a means to formally analyse tradeoffs in component selection. We will discuss how a PUF can model the incremental benefit (utility) that a system would receive for each additional unit of resource that a component consumes. We will explore open challenges in how the utility of a datum or event is transformed by the PUFs of the components they flow through, how to tradeoff one component for another based on their qualities, how to compose and decompose PUFs for bottom-up and top-down system design, and how to diagnose violations in expected utilities.

## 4 Working groups

### 4.1 Can AI be used in critical Systems?

*Jérôme Hugues (Carnegie Mellon University – Pittsburgh, US), Andres Barrilado (NXP Semiconductors – Toulouse, FR), Frédéric Boniol (ONERA – Toulouse, FR), Thomas Carle (Toulouse University, FR), Jerónimo Castrillón-Mazo (TU Dresden, DE), Samarjit Chakraborty (University of North Carolina at Chapel Hill, US), Arthur Clavière (Collins Aerospace – Blagnac, FR), Victor Jegu (Airbus S.A.S. – Toulouse, FR), Claire Pagetti (ONERA – Toulouse, FR), Marcus Rossel (Barkhausen Institut – Dresden, DE), Selma Saidi (TU Braunschweig, DE), Jonathan Sprinkle (Vanderbilt University – Nashville, US), Hasna Bouraoui (TU Dresden, DE) and Eugene Yip (GLIWA – Weilheim, DE)*

License  Creative Commons BY 4.0 International license  
© Jérôme Hugues, Andres Barrilado, Frédéric Boniol, Thomas Carle, Jerónimo Castrillón-Mazo, Samarjit Chakraborty, Arthur Clavière, Victor Jegu, Claire Pagetti, Marcus Rossel, Selma Saidi, Jonathan Sprinkle, Hasna Bouraoui, and Eugene Yip

The working group concluded that deploying AI in safety-critical systems hinges on balancing two tightly linked factors – accuracy and timeliness – because longer training or inference windows can enhance correctness but may violate real-time constraints. Across the four domains discussed (avionics, automotive, medical devices such as pacemakers, and rail), regulators are cautiously expanding standards: avionics currently only certifies off-line-trained ML components, automotive guidelines are rapidly evolving, medical devices show promising results under tight safeguards, and rail is changing slowly. Participants agreed AI is most defensible when it supplements rather than replaces human judgment or expensive hardware (e.g., monocular camera + AI distance estimation versus LiDAR) and when robust mitigation layers – observers, simplex architectures, safety “nets,” or treating AI like an imperfect sensor – contain the impact of inevitable errors. Defining what constitutes a “good” AI therefore means specifying clear functional and safety requirements, acknowledging residual

failure probabilities, and ensuring retraining or adaptation does not void certification credits. Finally, the group underscored that verification strategies must scale with AI complexity: small-state reinforcement learning can leverage formal methods, whereas high-dimensional perception models require new notions of coverage, traceability, and explainability, reinforcing that full autonomy remains out of reach today while incremental, supervised uses continue to gain traction.

## 4.2 Distributed music challenge

*Grzegorz Bazydło (University of Zielona Góra, PL), Anupam Chattopadhyay (Nanyang TU – Singapore, SG), Andrés Goens Jokisch (University of Amsterdam, NL), Chadlia Jerad (University of Manouba, TN), Erling Rennemo Jellum (University of California – Berkeley, US), Einar Broch Johnsen (University of Oslo, NO), Hokeun Kim (Arizona State University – Tempe, US), Edward A. Lee (University of California – Berkeley, US), Shaokai Jerry Lin (University of California – Berkeley, US), Jan Reineke (Universität des Saarlandes – Saarbrücken, DE), Manuel Serrano (INRIA – Sophia Antipolis, FR), Martin Schoeberl (Technical University of Denmark – Lyngby, DK), Lothar Thiele (ETH Zürich, CH) and Reinhard von Hanxleden (Universität Kiel, DE)*

**License** © Creative Commons BY 4.0 International license

© Grzegorz Bazydło, Anupam Chattopadhyay, Andrés Goens Jokisch, Chadlia Jerad, Erling Jellum, Einar Broch Johnsen, Hokeun Kim, Edward A. Lee, Shaokai Jerry Lin, Jan Reineke, Manuel Serrano, Martin Schoeberl, Lothar Thiele, and Reinhard von Hanxleden

In the breakout group session titled “Distributed Music Challenge,” we explored the practical aspects of the Distributed Rhythm Generator introduced in the preceding talk. This hands-on activity involved running a distributed real-time program written in Lingua Franca across two laptops, simulating the behavior of a networked rhythm generator. The session provided a concrete illustration of the tradeoffs between consistency and availability in distributed real-time systems. Participants experimented with three distinct conditions: normal network operation, a scenario with network overload, and one with disrupted clock synchronization. Through these scenarios, the group observed how the Lingua Franca runtime handled timing and consistency under different constraints, deepening the discussion on how reactor-oriented programming can be used to model and manage such tradeoffs elegantly in distributed systems.

## 4.3 Orchestration/coordination languages vs reactive languages

*Jérôme Hugues (Carnegie Mellon University – Pittsburgh, US), Edward A. Lee (University of California – Berkeley, US), Shaokai Jerry Lin (University of California – Berkeley, US), and Manuel Serrano (INRIA – Sophia Antipolis, FR)*

**License** © Creative Commons BY 4.0 International license

© Jérôme Hugues, Edward A. Lee, Shaokai Jerry Lin, and Manuel Serrano

The breakout group discussed the relationship between coordination languages, synchronous reactive languages, and choreographic languages. First, we started by reviewing examples of each language category, namely Reo, Choral, Rebeca, Esterel, and Lingua Franca. The group recognized that all these languages do some form of coordination. Some also have



the capability to support the full implementation either within their own language such as Esterel, or through the integration with other programming languages like Lingua Franca. In other cases, the functional code is considered opaque, and the language only defines the interface of these containers.

The group noted that these languages define how the synchronization of the execution of some containers happen. More specifically, their semantics define when (time), why (causality), and which action is executed. These containers are stateful, and they react to multiple, possibly simultaneous, inputs. Hence, it is critical to address when these events are sent, how they are processed, and how to change the container's state to guarantee determinism. These questions become more complicated when considering an implementation on resource-constrained embedded targets, as opposed to larger machines.

Finally, the group noted that this reasoning can be extended to not just mathematical formalisms such as communicating sequential processes (CSP) or Kahn process networks (KPN), but also synchronization/communication APIs such as POSIX or MPI. They also define some notion of stateful containers. However, in the case of API-based solutions, the actual boundary of the container and its interface is not a first-class citizen: asserting the correctness or determinism of a concurrent system, e.g., built on top of POSIX APIs, is more complex than a similar system expressed in a dedicated coordination language.

#### 4.4 Tradeoffs in accuracy and timeliness

*Andres Barrilado (NXP Semiconductors – Toulouse, FR), Hasna Bouraoui (TU Dresden, DE), Anupam Chattopadhyay (Nanyang TU – Singapore, SG), Jerónimo Castrillón-Mazo (TU Dresden, DE), Arpan Gujarati (University of British Columbia – Vancouver, CA), Hokeun Kim (Arizona State University – Tempe, US), Lothar Thiele (ETH Zürich, CH), Alexander Schulz-Rosengarten (Universität Kiel, DE), Katharina Sedow (Saneon GmbH – Ismaning, DE) and Jonathan Sprinkle (Vanderbilt University – Nashville, US)*

**License** © Creative Commons BY 4.0 International license

© Andres Barrilado, Hasna Bouraoui, Anupam Chattopadhyay, Jerónimo Castrillón-Mazo, Arpan Gujarati, Hokeun Kim, Lothar Thiele, Alexander Schulz-Rosengarten, Katharina Sedow, and Jonathan Sprinkle

This breakout session addressed the challenges, methodologies, and future directions involved in managing tradeoffs between accuracy and timeliness in computational systems. The central theme built on Lothar Thiele's earlier seminar presentation and drew parallels with Thinking, Fast and Slow [1], asking how systems could dynamically respond with appropriately timed decisions and varying confidence levels. The key questions and themes were:

- What methodologies can be used to manage accuracy vs. timeliness tradeoffs?
- What are the real costs of missing a timeline in industry today?
- How do real-time systems currently handle uncertainty or degraded performance?
- What is the role of resource scaling (e.g., parallelism) in trading accuracy for time?

During the discussion, Andr s Goens emphasized the benefits of early-exit strategies and iterative answer refinement for real-time constrained systems, raising concerns about how uncertainty compounds in extended predictions – illustrated with a motor motion example where the sine wave frequency is unknown. He suggested connecting these ideas to computationally aware model predictive control (MPC), citing Jonathan's earlier work, and posed the question: "Getting something quick converges eventually, but what issues expand with extended predictions?". Anupam Chattopadhyay contributed by stressing the

importance of leveraging parallel resources to improve accuracy under time constraints, especially in embarrassingly parallel applications. He framed timeliness and accuracy as just two among many competing objectives, and discussed the challenge of managing out-of-order answers in anytime systems, noting that “getting more accuracy with unbounded time might not be worth it.”

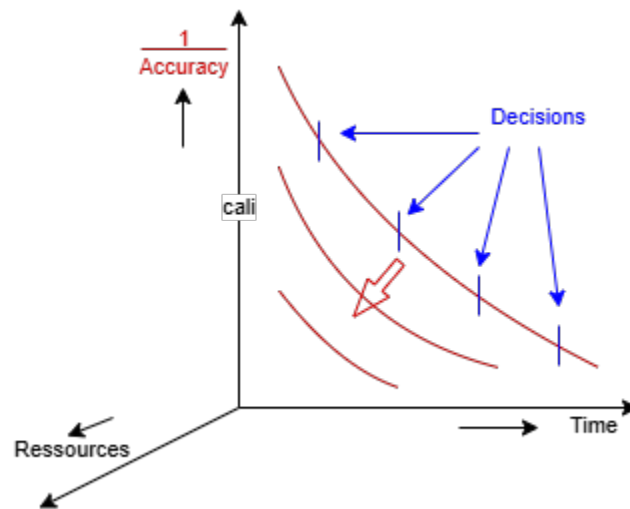
The discussion on use case and domain examples highlighted challenges in automotive and rail systems, particularly regarding decision timelines, fallback mechanisms, and uncertainty handling. In autonomous vehicles across L2 to L5 levels, issues such as classification errors – like braking for an image of a person rather than a real one – underscored the difficulty of making safe decisions when initial data is ambiguous. Real-world cases included Waymo’s geofenced operations in Phoenix and the contrast between fast emergency fallback systems and more complex uncertainty mitigation strategies in advanced vehicles. In rail systems, the focus shifted to multi-modal data fusion from LiDAR and cameras, emphasizing the challenge of resolving inconsistencies, especially when distinguishing between humans and other objects on the tracks. The layered processing approach – comprising rapid object detection, slower identification, and subsequent motion tracking – was presented as a method to incrementally refine understanding while balancing timeliness and accuracy.

Human-centered analogies highlighted how people naturally adjust processing time based on perceived threat and proximity, serving as an implicit model for soft real-time systems. For example, humans allocate more cognitive resources to assessment when a potential hazard appears distant, whereas immediate threats like the need to brake demand rapid reaction. Additionally, the way distractions – such as animals – can temporarily obscure more critical upcoming stimuli like vehicles illustrates how prioritization and attention shifts occur dynamically, a concept relevant to designing systems that must manage changing real-time demands.

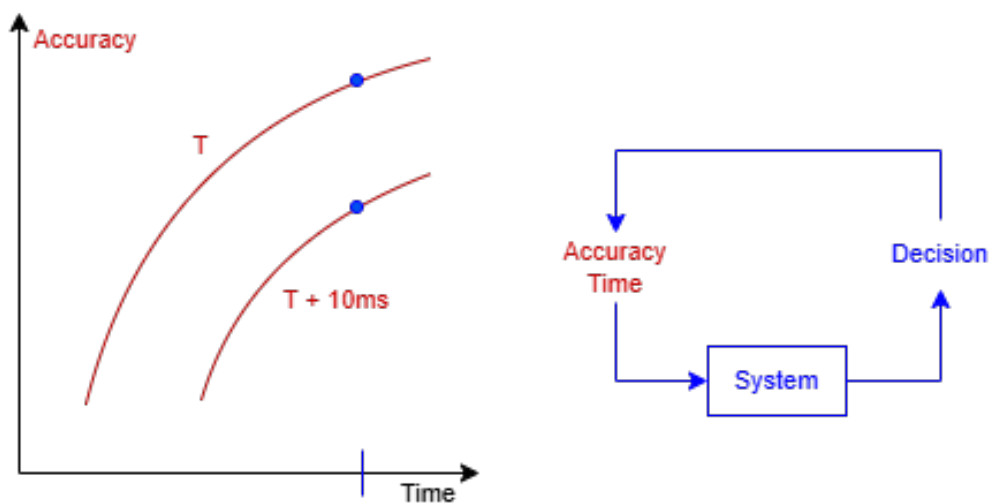
Key technical considerations included debates on online training and distribution drift, particularly whether it is preferable to adapt models dynamically at runtime or to preserve memory of rare edge cases that could be critical. Shadow mode simulation emerged as a strategy for evaluating new algorithms alongside active systems, especially within virtual or “metaverse” environments, offering a non-intrusive way to test improvements. Additionally, sensor cascades were discussed, where secondary sensors are activated based on outputs from primary sensors to refine decision accuracy, enabling more responsive and adaptive perception pipelines.

To visually support and extend the conceptual discussions, several figures (Figure 1, Figure 2, Figure 3) were sketched throughout the session. These illustrations capture the tradeoffs between accuracy, time, and computational effort in real-time and soft real-time systems. They also highlight architectures that mirror human-inspired decision strategies and progressive refinement models.

A key focus of the discussion was on the complex challenges of designing systems that must operate across multiple timelines while adapting to shifting, context-sensitive deadlines. Participants explored how systems can meaningfully react to outdated but potentially valuable information, and the risks associated with making premature decisions based on incomplete data. A recurring theme was the importance of metadata – such as timestamps and confidence scores – to inform decision-making under uncertainty. Additionally, the need to establish acceptance criteria for degraded or “poisoned” data, and the integration of error-correcting mechanisms in fault-tolerant designs, was emphasized. These challenges raised several open questions about the future of dynamic system design.



■ **Figure 1** Anytime Algorithm Convergence: Anytime algorithms will converge on a specific accuracy, which improves over time. Understanding any mid-execution accuracy (decision points) can help approximate what the final accuracy may be. Additional resources for parallel activity may further improve the convergence time. Axes: Y-axis: accuracy, X-axis: time, Z-axis: resources required to obtain the particular decision points).



■ **Figure 2** Frame-based Decision Enhancement: Taking into account regular processing of frame rates: enabling parallel access with in-process updates (not early exit) could open the pathway to taking 2 data values from two frames, and using the combined information to make a more informed decision (Axes: Y-axis: accuracy, X-axis: time).



■ **Figure 3** A “Thinking Fast, Thinking slow” explicit example: Component 1 can arrive with something above the accuracy threshold, but will not maximize accuracy quickly. The result from its quick response can trigger the next component, which should (with initial criteria) arrive at a better solution, faster than starting without that information. Axes: Y-axis: accuracy, X-axis: time).

#### Key System Design Challenges:

- Managing multiple, context-sensitive timelines and deadlines
- Handling outdated but potentially valuable data
- Mitigating the cost of incorrect early decisions
- Timestamping decisions and including confidence metadata
- Defining acceptance criteria for incomplete or degraded input
- Building fault-tolerant systems that can still reason under uncertainty

#### Open Questions:

- How can contextual deadlines be defined in reactive systems?
- What are effective methods to systematically integrate fault models and fallback strategies?
- Beyond automotive and rail, which domains are best suited for designs emphasizing dynamic tradeoffs in accuracy and timeliness?

As a closing summary, the group emphasized that:

- Timeliness vs. Accuracy must be treated not as static tradeoffs but as part of a system’s runtime behavior.
- Current systems rarely pay a high cost for delayed decisions, but future high-stakes autonomous platforms (e.g., L5 vehicles, medical robotics) will.
- There is opportunity in revisiting layered decision systems, sensor fusion under uncertainty, and the reuse of older data in dynamic ways.
- Jerónimo Castrillón-Mazo’s Debriefing Thought: “When the trigger changes your deadline, you are in a new kind of reactive system.”

#### References

- 1 Daniel Kahneman. *Thinking, fast and slow*. macmillan, 2011.

## 4.5 Benchmarks for real-time systems

*Thomas Carle (Toulouse University, FR), Samarjit Chakraborty (The University of North Carolina (UNC) at Chapel Hill, US), Arthur Clavière (Collins Aerospace – Blagnac, FR), Victor Jegu (Airbus S.A.S. – Toulouse, FR), Chadlia Jerad (University of Manouba, TN), Claire Pagetti (ONERA – Toulouse, FR), Jan Reineke (Universität des Saarlandes – Saarbrücken, DE) and Martin Schoeberl (Technical University of Denmark – Lyngby, DK)*

**License** © Creative Commons BY 4.0 International license

© Thomas Carle, Samarjit Chakraborty, Arthur Clavière, Victor Jegu, Chadlia Jerad, Claire Pagetti, Jan Reineke, and Martin Schoeberl

The work group on “Benchmarks for Real-Time Systems” discussed the shortcomings of existing benchmark suites like TACLeBench and explored possible directions for developing more representative and useful benchmarks. TACLeBench was criticized for being largely composed of small, non-real-time benchmarks, lacking multi-threaded code, and using only fixed inputs. Furthermore, it does not offer standardized licensing, nor does it provide clear guidance on the context in which benchmarks should be executed or evaluated. These limitations make it ill-suited for evaluating modern real-time systems, which are increasingly complex and heterogeneous.

Participants noted that existing real-time benchmarks are used for a range of evaluation purposes – such as analyzing the worst-case performance and/or predictability of microarchitectures, assessing WCET (Worst-Case Execution Time) analysis tools, and evaluating scheduling algorithms. Depending on the benchmarks’ purpose they could consist of individual tasks, full applications, or even complete systems where specific software and hardware combinations are considered together.

A key theme in the discussion was the importance of designing benchmarks that reflect actual real-time workloads and applications. Real-time systems span diverse domains and benchmarks should reflect this diversity. Instead of relying exclusively on small code fragments, participants suggested a potential shift towards specification-based benchmarks: defining the intended behavior rather than a fixed implementation, with the goal of deriving implementations that exhibit predictable worst-case performance.

Domains that are particularly representative of modern real-time systems include vision and perception (e.g., camera or LIDAR input), sensor fusion, control algorithms such as model-predictive control, and mixed workloads running on heterogeneous architectures (e.g., involving both CPUs and GPUs). These domains may feature execution time variation depending on the nature of the input data, a characteristic that existing benchmark suites largely fail to capture.

Several participants pointed to tools like Simulink and Lustre as valuable sources for generating realistic benchmarks. Simulink, in particular, includes many example controllers that are already central to real-world safety-critical systems. These models can be used to generate C code, with support for different code generation configurations, including full closed-loop systems that simulate both controllers and plants. Similarly, code generated from Lustre, or real-world code bases like Autoware, could serve as useful starting points for building a more diverse and representative benchmark suite.

Another important aspect raised was the need to define reference contexts for benchmarks. This includes specifying the hardware architecture and microarchitecture, selecting reasonable parameter values, and ideally providing baseline results obtained using known analysis tools. Benchmarks should also include annotations – such as loop bounds – to ensure they are compatible with WCET analysis tools like AbsInt.

## 5 Software teasers

In this session, various seminar participants demonstrate programming languages and tools they are developing.

### 5.1 Timed Rebeca

*Marjan Sirjani (Mälardalen University – Västerås, SE)*

**License** © Creative Commons BY 4.0 International license  
© Marjan Sirjani

Timed Rebeca is an actor-based modeling language equipped with a model-checking tool. It is designed to be both accessible and usable for software engineers and grounded in formal semantics, ensuring formal verification support. A cyber-physical system is represented as a set of communicating actors, and safety and liveness properties can be verified using the model-checking tool, Afra. Rebeca homepage can be accessed here: <https://rebeca-lang.org/>.

### 5.2 Lingua Franca

*Erling Rennemo Jellum (University of California – Berkeley, US), Edward A. Lee (University of California – Berkeley, US)*

**License** © Creative Commons BY 4.0 International license  
© Erling Rennemo Jellum and Edward A. Lee

Lingua Franca is a reactor-oriented coordination language for implementing cyber-physical systems. Due to its declarative syntax, automatic diagramming is possible. By separating the concerns of coordination and timing from the computation, it is a promising target for code generated by large language models. Lingua Franca homepage can be accessed here: <https://www.lf-lang.org/>.

### 5.3 SCCharts


*Alexander Schulz-Rosengarten (Universität Kiel, DE)*

**License** © Creative Commons BY 4.0 International license  
© Alexander Schulz-Rosengarten

SCCharts is a reactive synchronous programming language with sequentially constructive semantics. It supports dynamic ticks, which are driven by a minimal interface to the environment. As such, it may be integrated with other runtimes, such as Lingua Franca, to provide its ticks. SCCharts rely on several model-to-model transformations to turn an SCChart model into code. SCCharts has sophisticated diagram generation, the same that is used in Lingua Franca. Each compilation step can also be inspected graphically, making the compiler transparent and easing development. Links to SCCharts can be accesses here: <https://github.com/kieler/semantics/wiki/Quick-Overview>.

## 5.4 QRML

*Marc Geilen (TU Eindhoven, NL)*

License  Creative Commons BY 4.0 International license  
© Marc Geilen

QRML is a domain-specific language for formulating multi-objective optimization problems for cyber-physical systems. The problems can be exported to several formats and solved by tools like Z3. QRML also runs in the browser. QRML tool can be accessed here: <https://qrml.org/qrml>.

## 5.5 HipHop

*Manuel Serrano (INRIA – Sophia Antipolis, FR)*

License  Creative Commons BY 4.0 International license  
© Manuel Serrano

HipHop is a synchronous-reactive language embedded in Javascript. It connects the asynchronous world of Javascript with the synchronous world of Esterel. The HipHop program describes a deterministic and concurrent response to external inputs, and the embedding Javascript decides when to trigger the HipHop program. HipHop can be downloaded from GitHub (<https://github.com/manuel-serrano/hiphop>).

## Participants

- Andres Barrilado  
NXP Semiconductors –  
Toulouse, FR
- Grzegorz Bazydło  
University of Zielona Gora, PL
- Frédéric Boniol  
ONERA – Toulouse, FR
- Hasna Bouraoui  
TU Dresden, DE
- Thomas Carle  
Toulouse University, FR
- Jerónimo Castrillón-Mazo  
TU Dresden, DE
- Samarjit Chakraborty  
University of North Carolina at  
Chapel Hill, US
- Anupam Chattopadhyay  
Nanyang TU – Singapore, SG
- Arthur Clavière  
Collins Aerospace – Blagnac, FR
- Marc Geilen  
TU Eindhoven, NL
- Alain Girault  
INRIA – Grenoble, FR
- Andrés Goens Jokisch  
University of Amsterdam, NL
- Arpan Gujarati  
University of British Columbia –  
Vancouver, CA
- Jérôme Hugues  
Carnegie Mellon University –  
Pittsburgh, US
- Victor Jegu  
Airbus S.A.S. – Toulouse, FR
- Erling Rennemo Jellum  
University of California –  
Berkeley, US
- Chadlia Jerad  
University of Manouba, TN
- Einar Broch Johnsen  
University of Oslo, NO
- Hokeun Kim  
Arizona State University –  
Tempe, US
- Edward A. Lee  
University of California –  
Berkeley, US
- Shaokai Jerry Lin  
University of California –  
Berkeley, US
- Claire Pagetti  
ONERA – Toulouse, FR
- Jan Reineke  
Universität des Saarlandes –  
Saarbrücken, DE
- Marcus Rossel  
Barkhausen Institut –  
Dresden, DE
- Selma Saidi  
TU Braunschweig, DE
- Klaus Schneider  
RPTU  
Kaiserslautern-Landau, DE
- Martin Schoeberl  
Technical University of Denmark  
– Lyngby, DK
- Alexander Schulz-Rosengarten  
Universität Kiel, DE
- Katharina Sedow  
Saneon GmbH – Ismaning, DE
- Manuel Serrano  
INRIA – Sophia Antipolis, FR
- Marjan Sirjani  
Mälardalen University –  
Västerås, SE
- Jonathan Sprinkle  
Vanderbilt University –  
Nashville, US
- Eric Tutu Tchao  
Kwame Nkrumah University of  
Science and Technology, GH
- Lothar Thiele  
ETH Zürich, CH
- Reinhard von Hanxleden  
Universität Kiel, DE
- Eugene Yip  
GLIWA – Weilheim, DE

