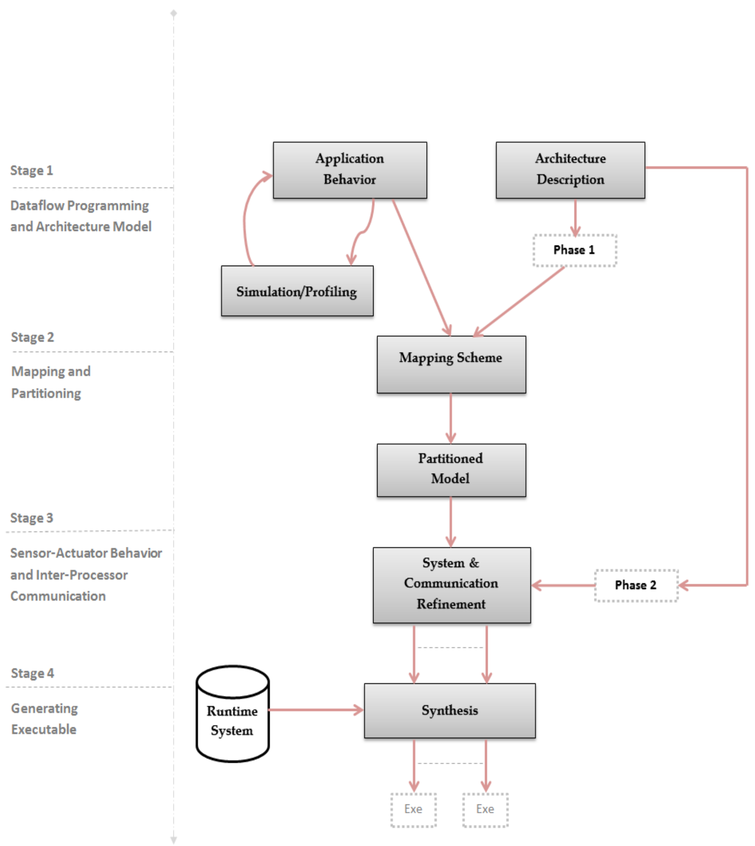
COMPUTER SCIENCE

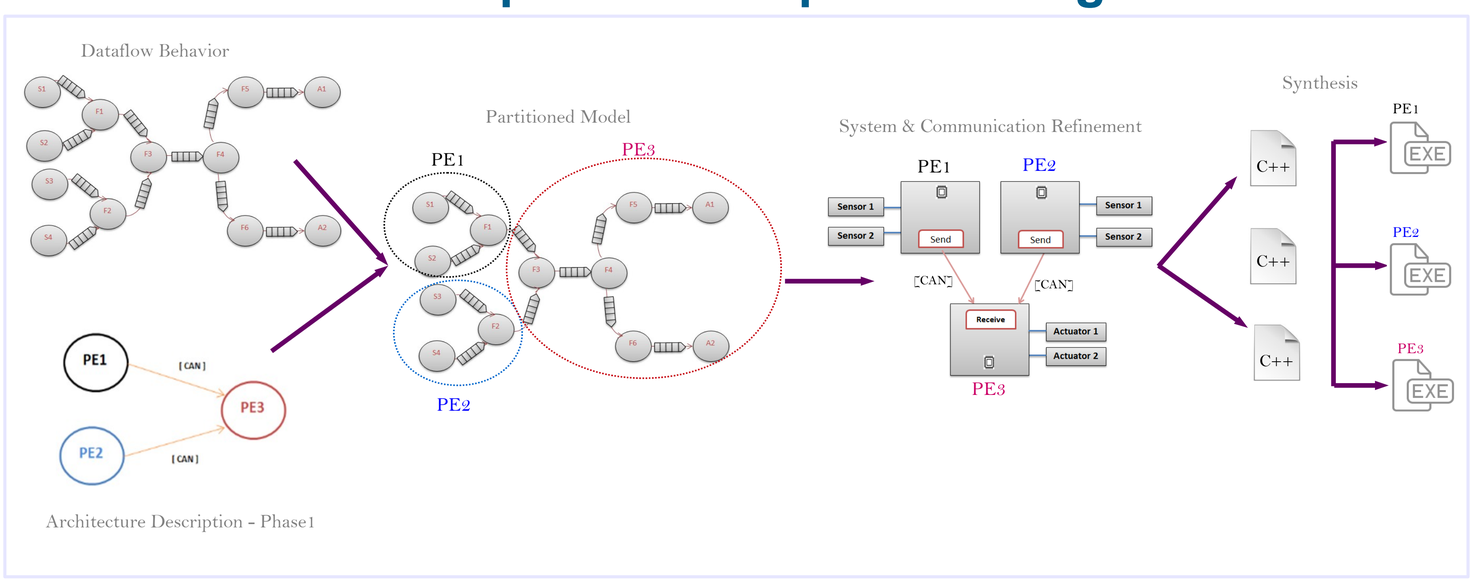
System and Communication Synthesis Design Flow for Model-Based Embedded Systems

The Framework



In general, model-based design flows for embedded systems are developed with abstract models based on different MoCs. The typical design steps like simulation, verification, partitioning and synthesis finally yield a couple of C files. The generated C files are then manually deployed/implemented onto heterogeneous architectures accompanied with software and hardware components. The software part is generally considered as a single embedded processor (with single or multiple cores). Consequently, the synthesis stage generates a ***single executable*** for the embedded processor. Moreover, the manual implementation steps at the deployment stage do not necessarily preserve the properties of the underlying MoC of the model-based design. These non-trivial manual design steps can break all correctness-by-construction guarantees of the model-based design and thus introduce the ***deployment gap***.

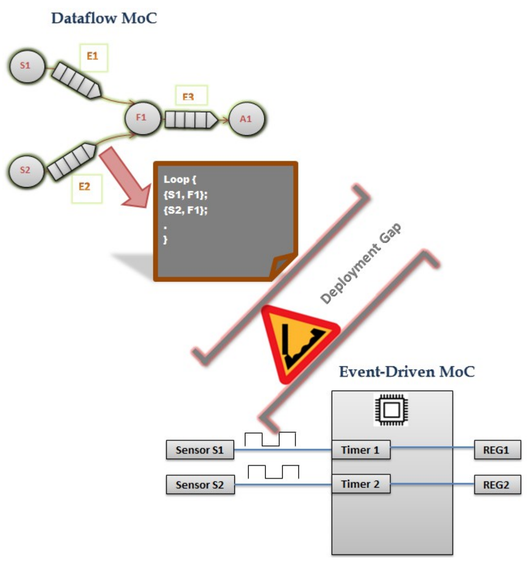
Example: The Proposed Design Flow



Example: The Deployment Gap and the MoC Drivers

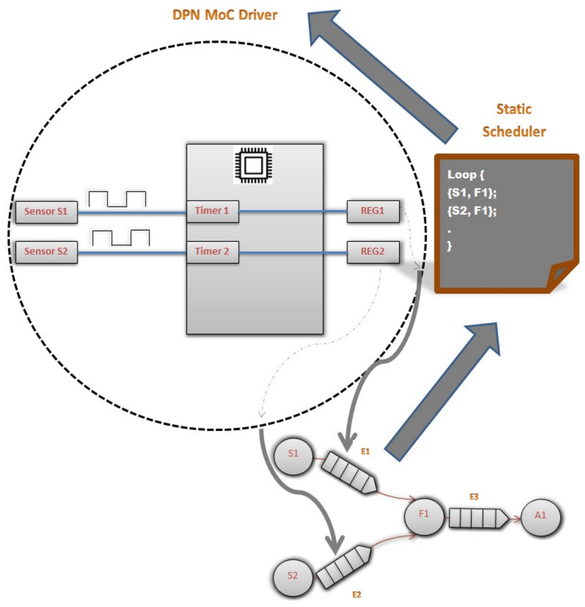
The Deployment Gap

The deployment gap between an application modeled with DPNs and the underlying target architecture using an event-driven computation



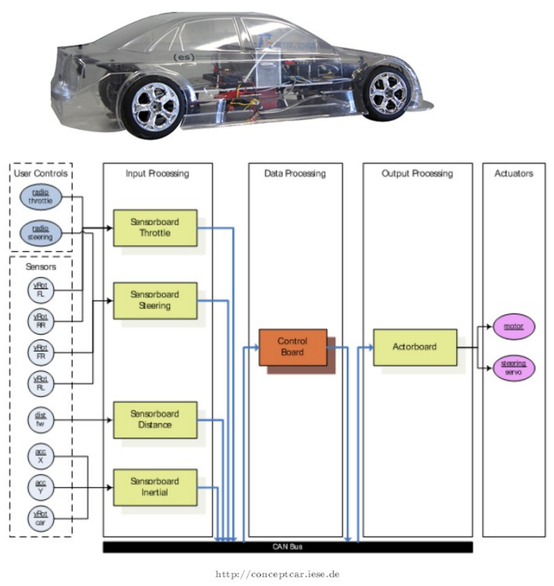
The MoC Drivers

The MoC driver wraps devices of the target architecture in an abstract behavior that makes use of the MoC of the used model-based design



The Test Case – Concept Car

The ConceptCar is an experimental vehicle with the objective of testing and verifying modern future car features by deploying different classes of applications



Model-Based Design Flows

- different MoCs for abstract application behaviors
- target architecture exposed via architecture description languages
- behavioral simulation for early design refinements
- automatic partitioning and mapping of abstract behavior onto the target architecture
- communication refinement for inter-processor communication

Research

- **hardware-agnostic application behavior**
  - CAL actor language for defining DPNs with different variants (KPN, SDF, cyclo-static)
  - characterized with simple input-process-output semantics
- **architecture-description language**
  - allows integrating event-driven behaviors of devices
  - provides standard templates to device vendors to abstract the behavior at the MoC level
- **heterogeneous distributed architectures**
  - consideration of heterogeneous architectures with single/multiple core processors
- **system synthesis**
  - generating multiple executable for target architectures
  - automatically bridging the deployment gap at the MoC level
- **communication synthesis**
  - abstract representations for inter-processor communication synthesized to specific interfaces

M.Eng. Omair Rafique
Chair for Embedded Systems
Department of Computer Science
University of Kaiserslautern

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

https://es.cs.uni-kl.de