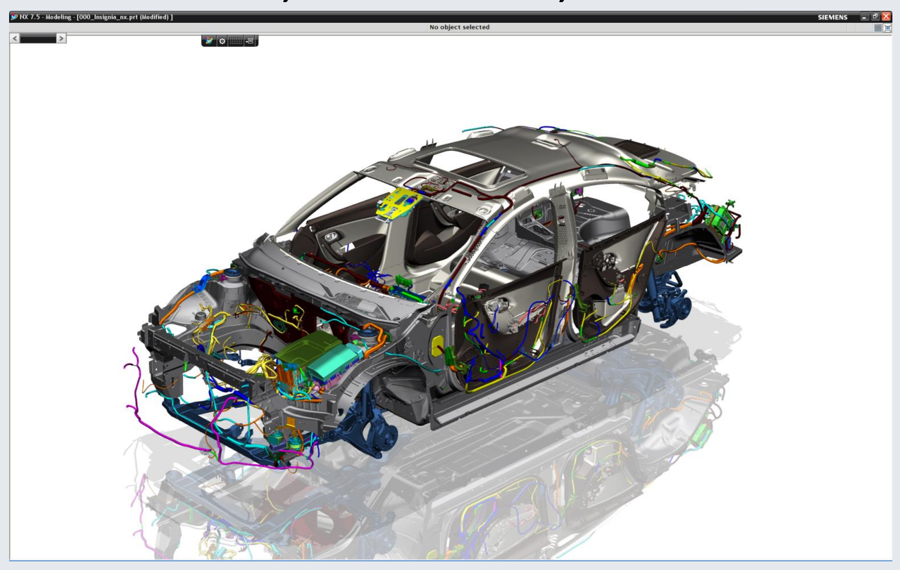
Functional Modeling Compiler for System-Level Design of Automotive Cyber-Physical Systems

Jiang Wan , Prof. Mohammad Abdullah Al Faruque
The Henry Samueli School of Engineering Electrical Engineering & Computer Science
{jiangwan, alfaruque}@uci.edu

Arquimedes Canedo
Siemens Corporation Princeton,
arquimedes.canedo@siemens.com

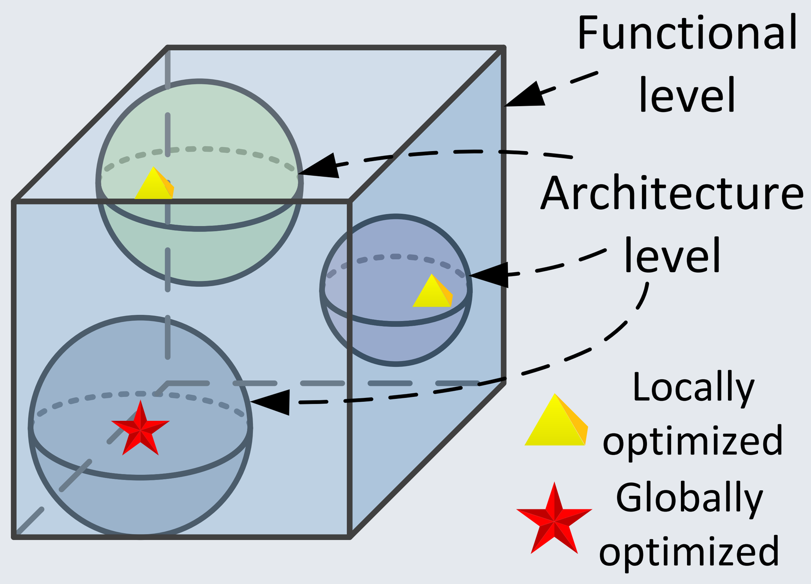
Background and Introduction

- A typical **Cyber-Physical System (CPS)** such as the modern automotive includes not only the physical domains such as mechanics, chemistry, etc., but also cyber domains such as Software, Hardware, control and communication.
  - More than 70 ECUs
  - More than 5 bus systems
  - More than 10,000,000 lines of code
  - 2000 individual functions
  - 40% of the costs due to cyber domain development



An example of CPS: a modern automotive

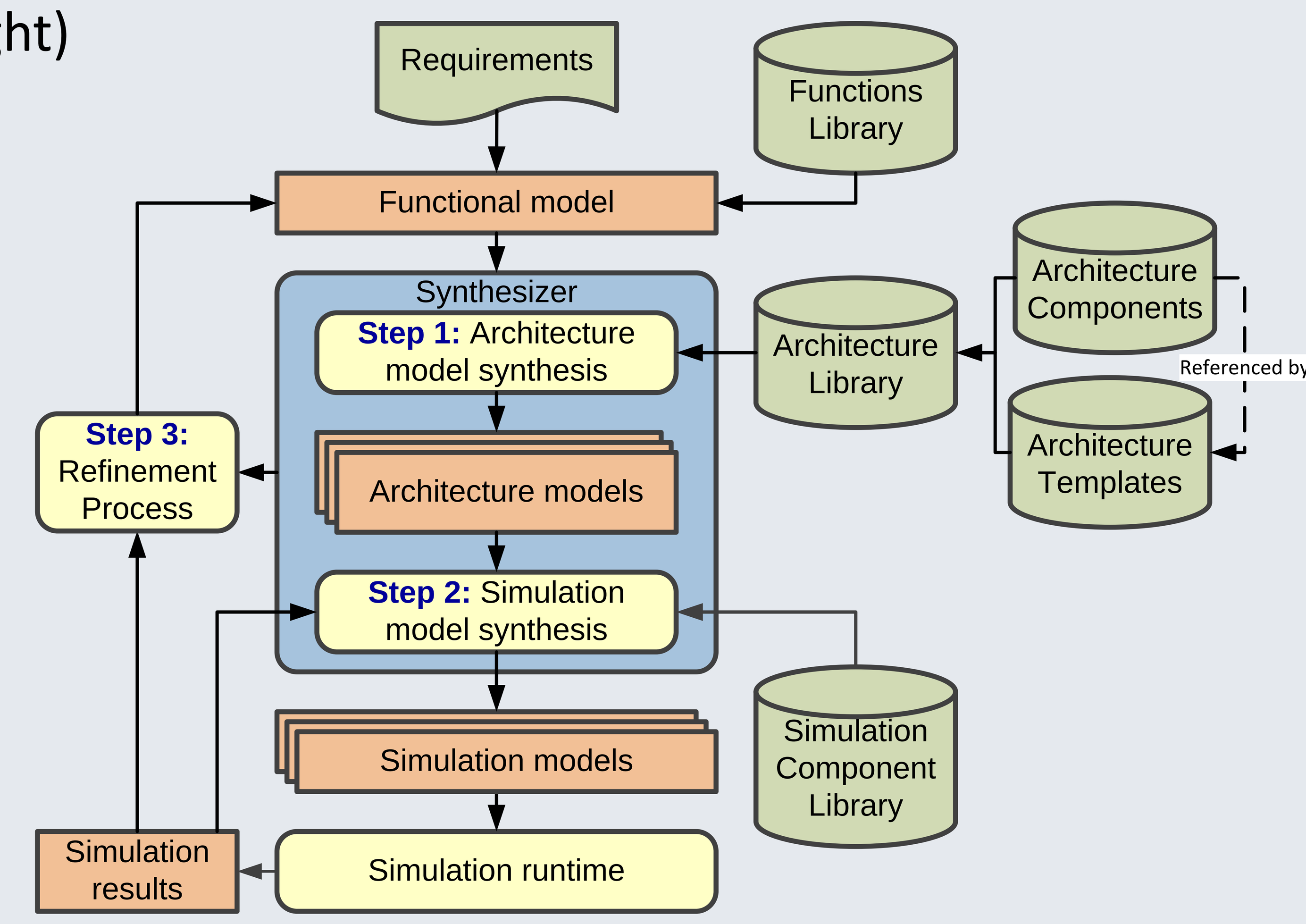
- Current CPS design methods start directly from the **architecture models** or **simulation models**
- Locally optimized solution
- We propose to start from the **functional models** to automatically explore the globally optimized solution



Design space at functional & architecture level

Functional Model Compiler (FMC)

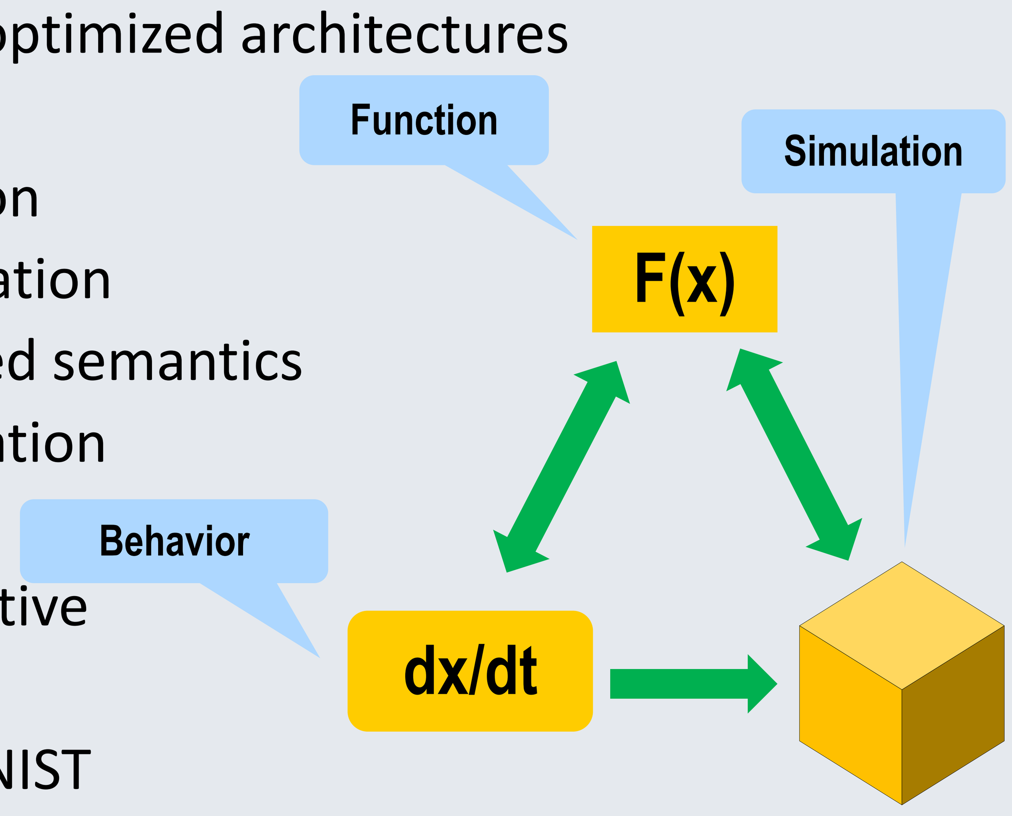
- **Objective:** identify all potential engineering solutions for an automotive CPS design, and down-select the ones that fulfill both the requirements (e.g. regulatory) and the constraints (e.g. weight)



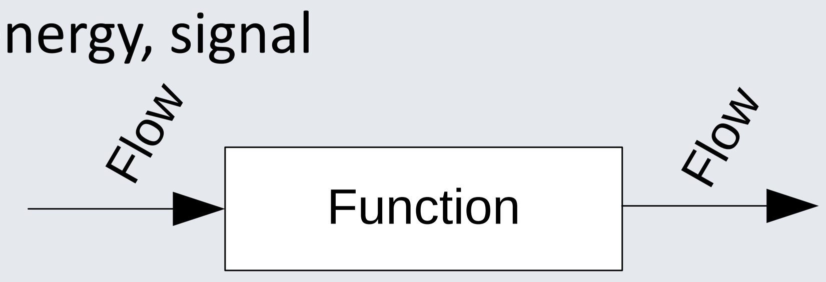
- **Step 1:** Synthesize functional model into architecture models
- **Step 2:** Synthesize architecture models into simulation models
- **Step 3:** Refinement of both architecture and functional models

Functional Model

- **Functional models** provide a higher-level of abstraction to specify the systems dynamics systematically to perform a broad design space exploration to select various optimized architectures
  - What the system does?
  - Unified system representation
  - Inter-disciplinary communication
  - Visual syntax and well defined semantics
  - Multi-disciplinary representation
  - Functions and flows
  - Standard practice in automotive

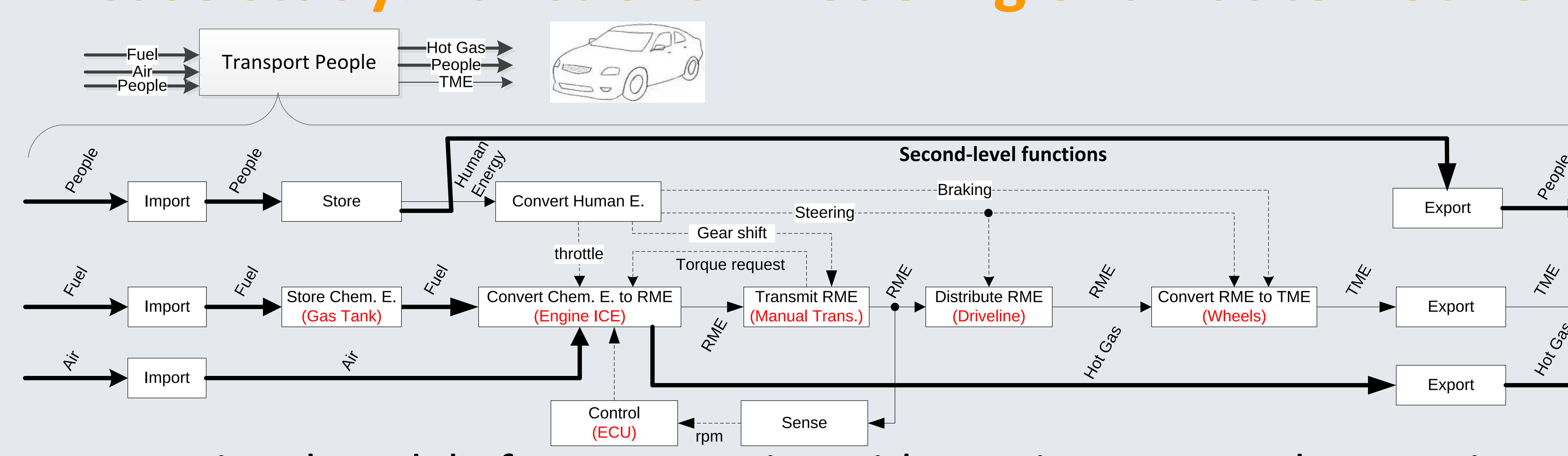


- **Functional Basis language**
  - Well defined vocabulary by NIST
    - Three flow categories: Material, energy, signal
    - 18 flow types
    - 32 elementary functions
  - Execution flows from left-to-right



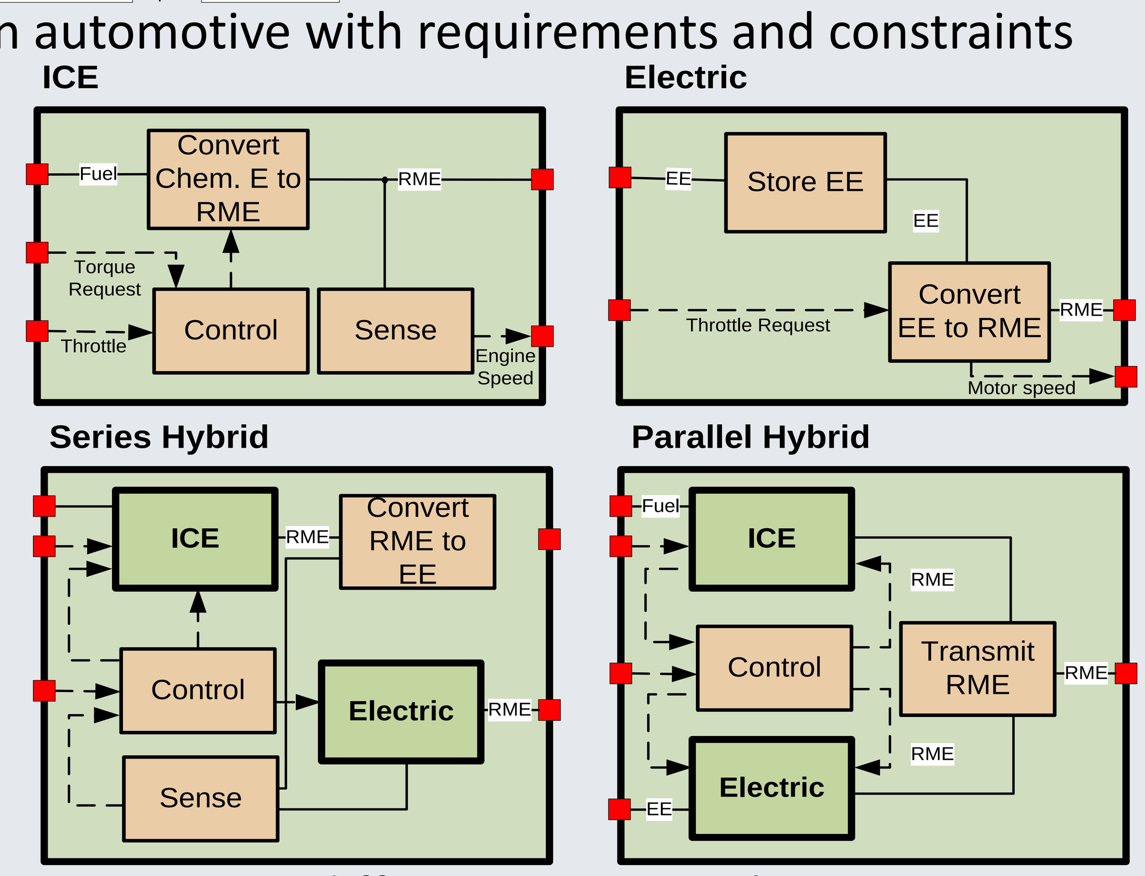
Functional Basis language example

Case study: Functional Modeling of an automotive

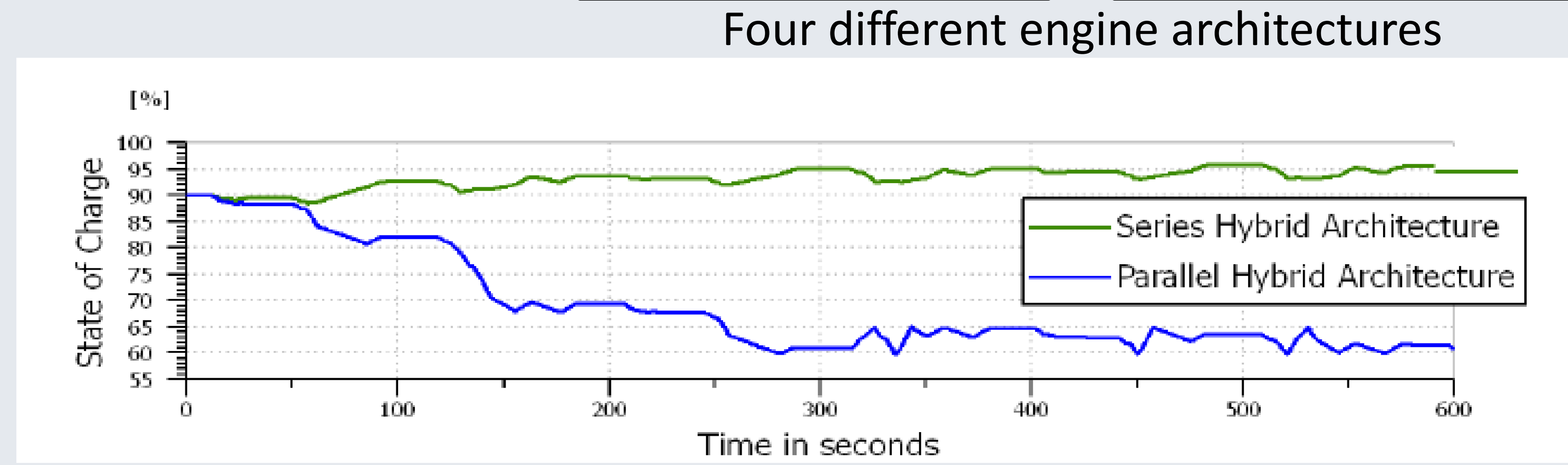


Functional model of an automotive with requirements and constraints

- **Step 1:** Four engine architecture models are synthesized
- **Step 2:** Series Hybrid and Parallel Hybrid simulation models are synthesized
- **Step 3:** Provide suggestions to refine the functional model



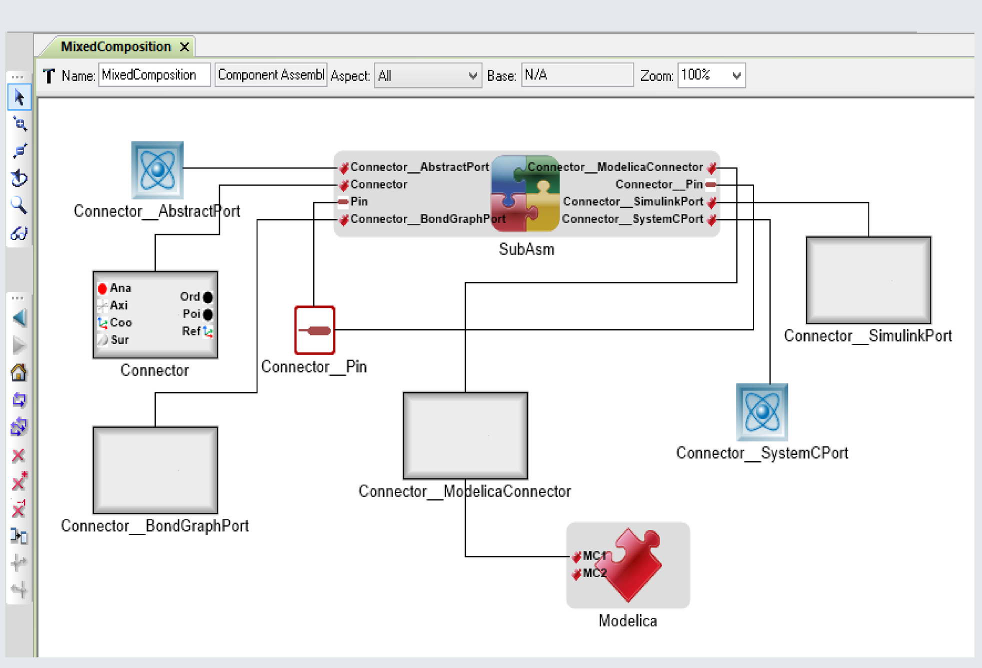
Four different engine architectures



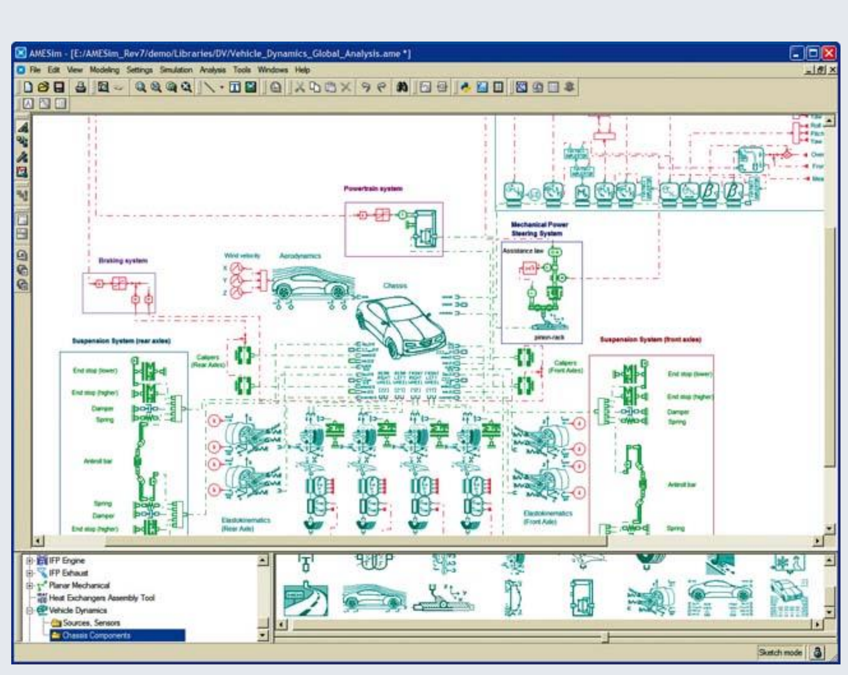
One of the simulation results: Battery's state of charge comparison between the Series Hybrid and Parallel Hybrid architectures

Architecture models and simulation models

- **Architecture model:** Architecture models define the allocation of physical structures to functionality.
- State-of-the-art architecture modeling techniques: CyPhyML; Ptolemy; AADL; Acme; etc.
- **Simulation model:** Simulation models are component-based executable multi-domain models
- State-of-the-art multi-domain simulation tools: Simulink; AMESim; etc.



Architecture Modeling using CyPhyML



An AMESim model of the automotive

Our related publications

- A. Canedo, M. A. Al Faruque, "Context-sensitive synthesis of executable functional models of cyber-physical systems", IEEE/ACM ICCPS 2013
- A. Canedo, M. A. Al Faruque, " Multi-disciplinary integrated design automation tool for automotive cyber-physical systems ", IEEE/ACM DATE 2014

Related work

- E. A. Lee, et. al, "Taming heterogeneity-the Ptolemy approach" Proceedings of the IEEE vol:91(1) 2003
- J. Sztipanovits, et al, "Specification of Cyber-Physical Components with Formal Semantics–Integration and Composition“, In Model-Driven Engineering Languages and Systems 2013
- M. Broy, "Challenges in automotive software engineering" international conference on Software engineering 2006

Presented @ 2014 51st ACM/EDAC/IEEE Design Automation Conference