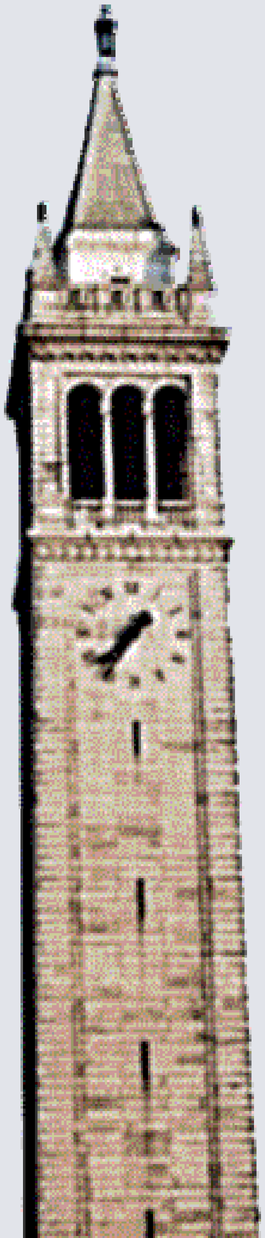
New Vistas on Automotive Embedded Systems

Edited and presented by
Alberto Sangiovanni-Vincentelli
UC Berkeley



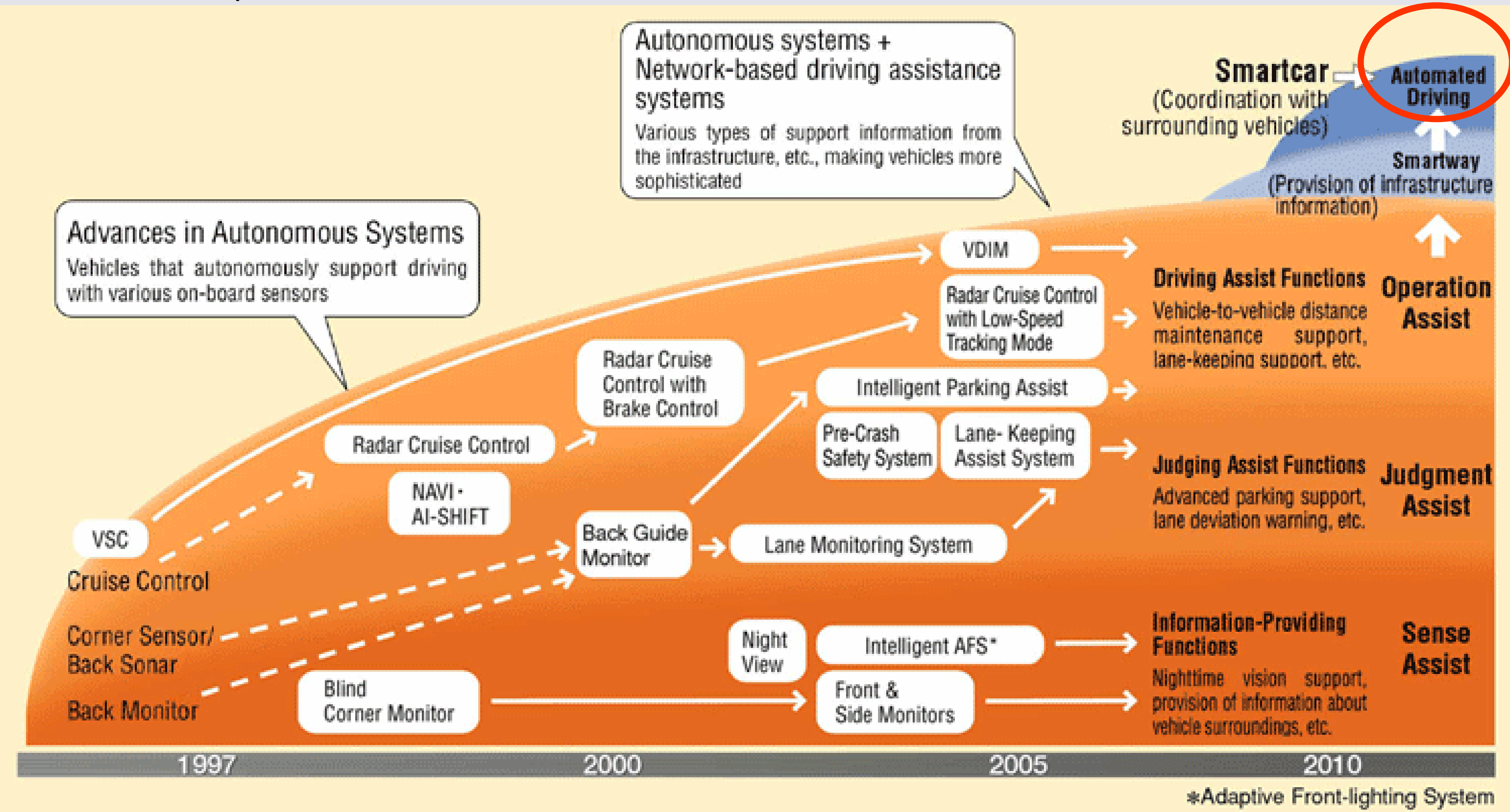
Chess Review
October 4, 2006
Alexandria, VA

Notable Quotes

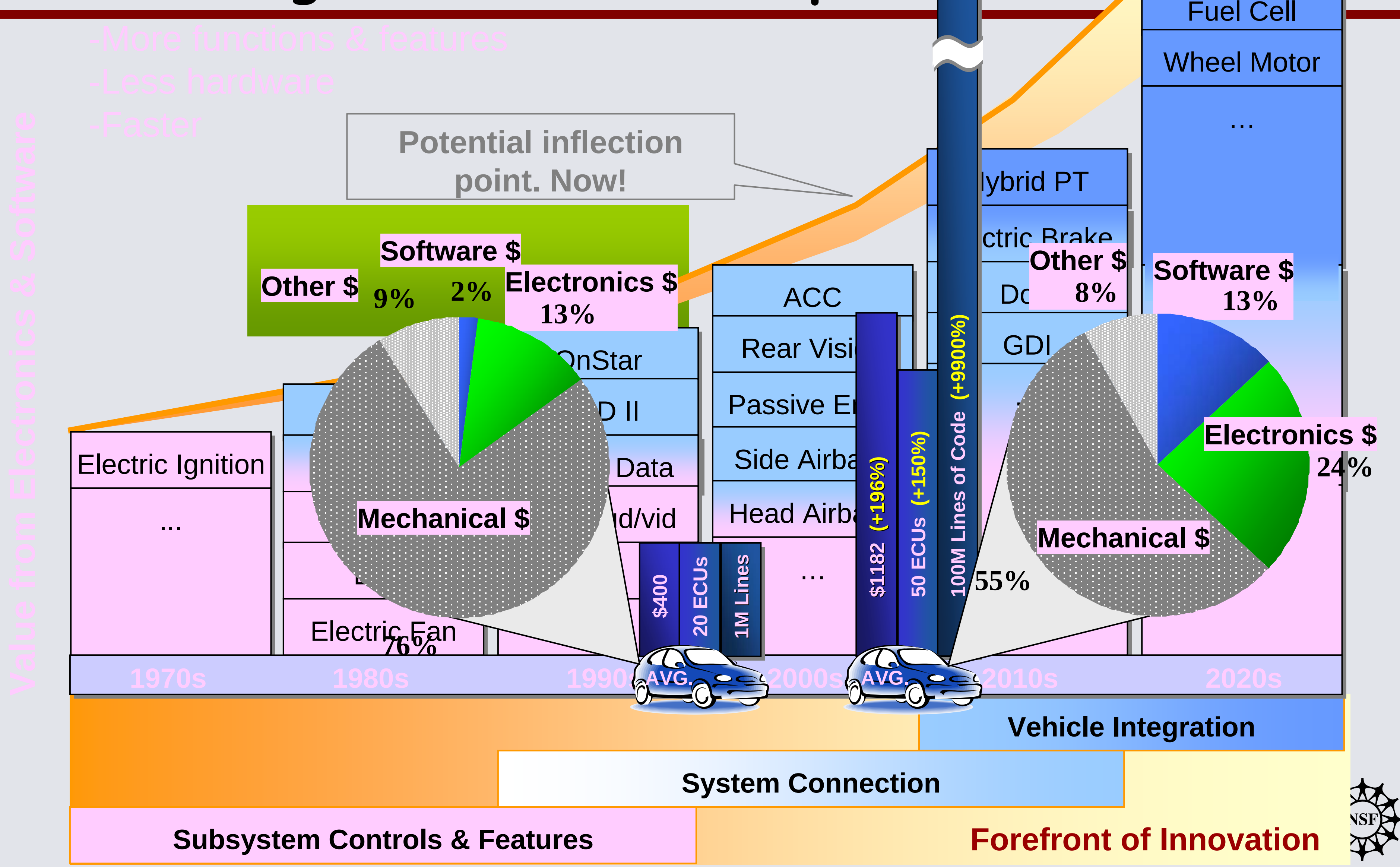
- The Nihon Keizai Shimbun reported that Japan Ministry of Economy, Trade and Industry estimates that Japanese companies spend more than 100 billion yen (USD $903 million) per year developing automotive-related software. And it isn't going to get any cheaper, with some analysts estimating costs escalating to 1 trillion yen (**USD $9.1 billion**) by 2014, according to the daily newspaper.
- So is the industry ultimately **moving toward 'plug-and play'?** Taking the idea of multiplexing to its logical extreme, a carmaker could potentially wait until relatively late in the vehicle's development cycle before committing to specific electronic hardware yet avoid having to delay - or worse, tear up - its electrical architecture in the last minute.

Toyota Autonomous Vehicle Technology Roadmap

Source: Toyota Web site

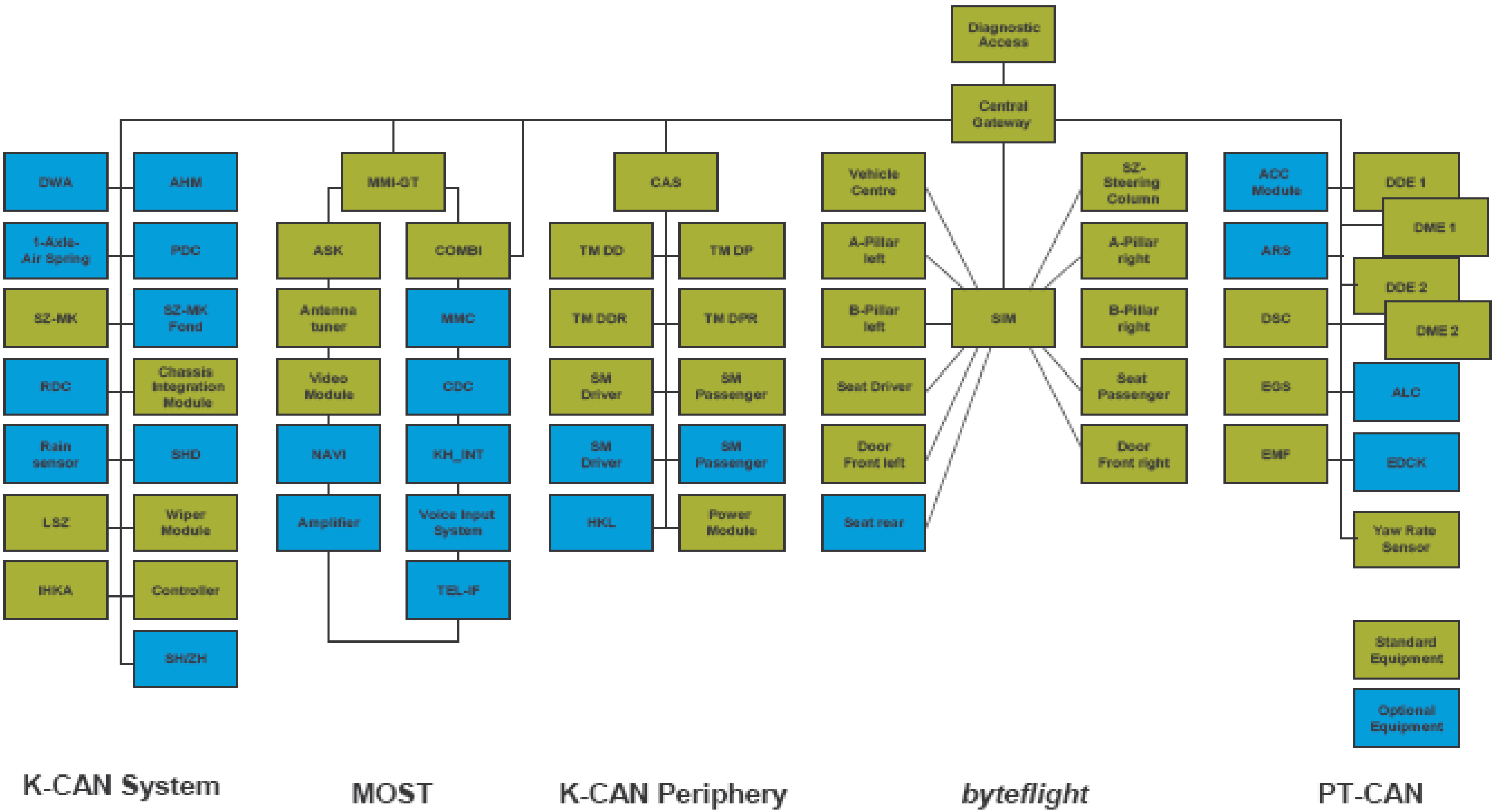


Electronics, Controls & Software
Shifting the Basis of Competition in Vehicles



Source: Matt Tsien, GM

A Typical Car Architecture (BMW)

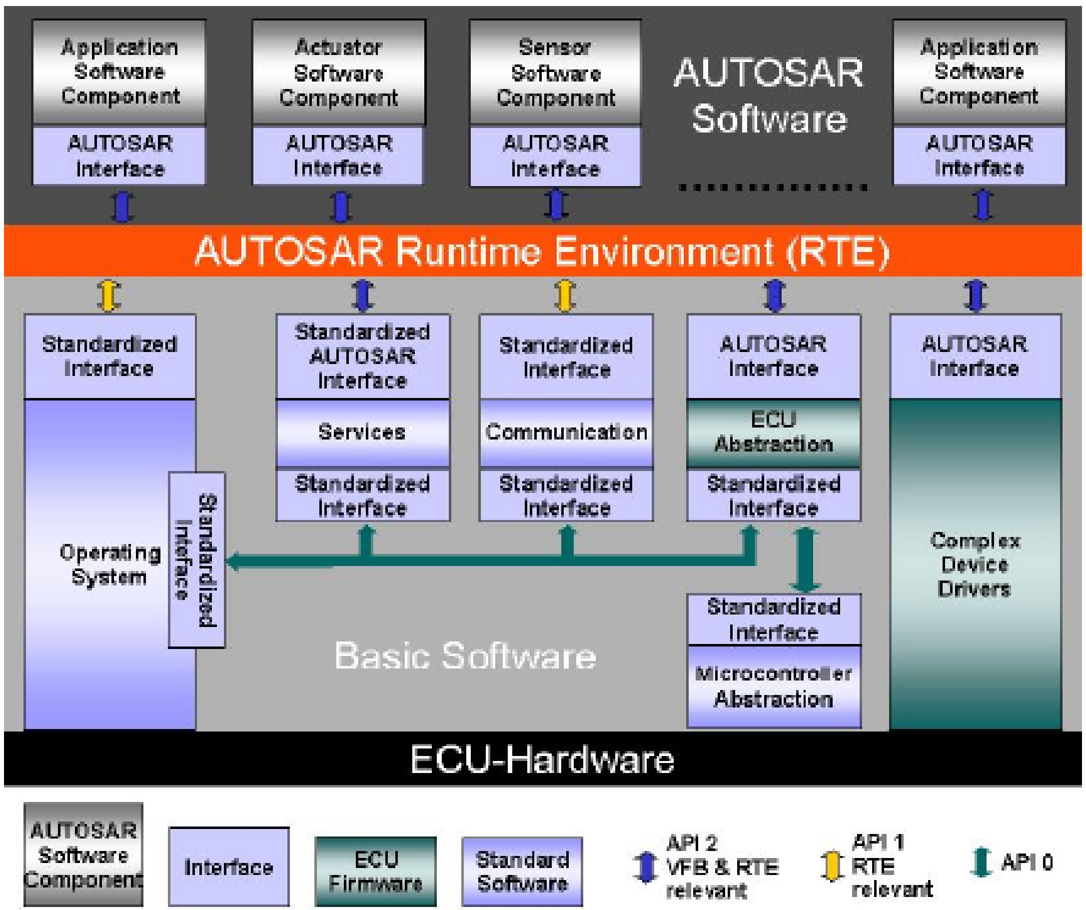


Top Priorities

- System-level architecture design approach
  - To what extent can we decouple the dimensions of architecture (computation, communication, power, etc.)?
  - What are the guiding principles of system-level architecture design?
  - What are the tools to support system-level architecture design, modeling, simulation, and analysis?
- Next-generation architecture strategy
  - What is the long-term architecture vision
    - Independent of (not biased by) today's architecture
    - Not just evolution of Michigan A / Global A.
  - What is the best approach to incrementally transition to the long-term architecture?
  - Is Global A architecture good enough for the long term? How much better is possible?

AUTOSAR

AUTOSAR – ECU Software Architecture

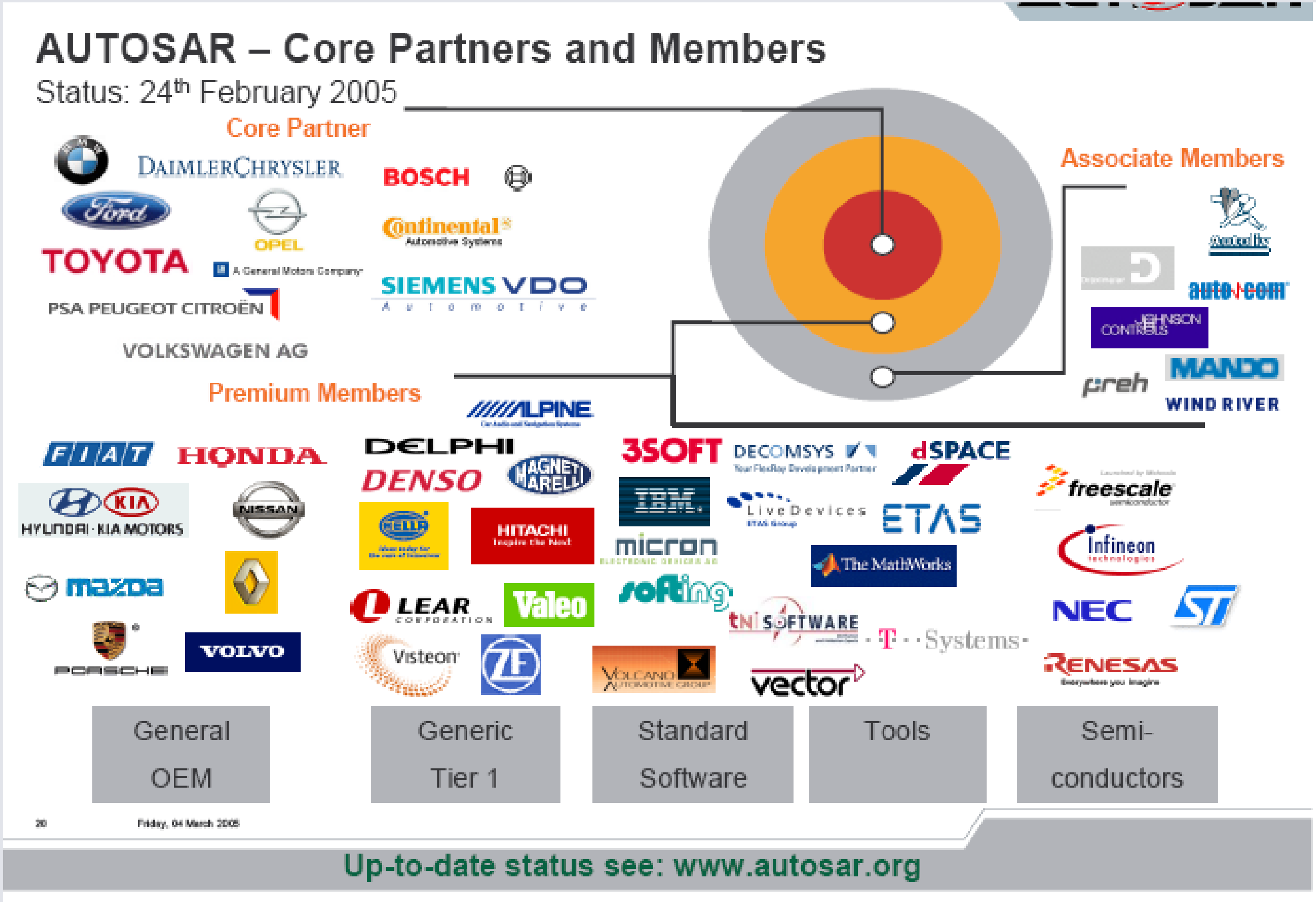


Automotive Open System Architecture (AUTOSAR):

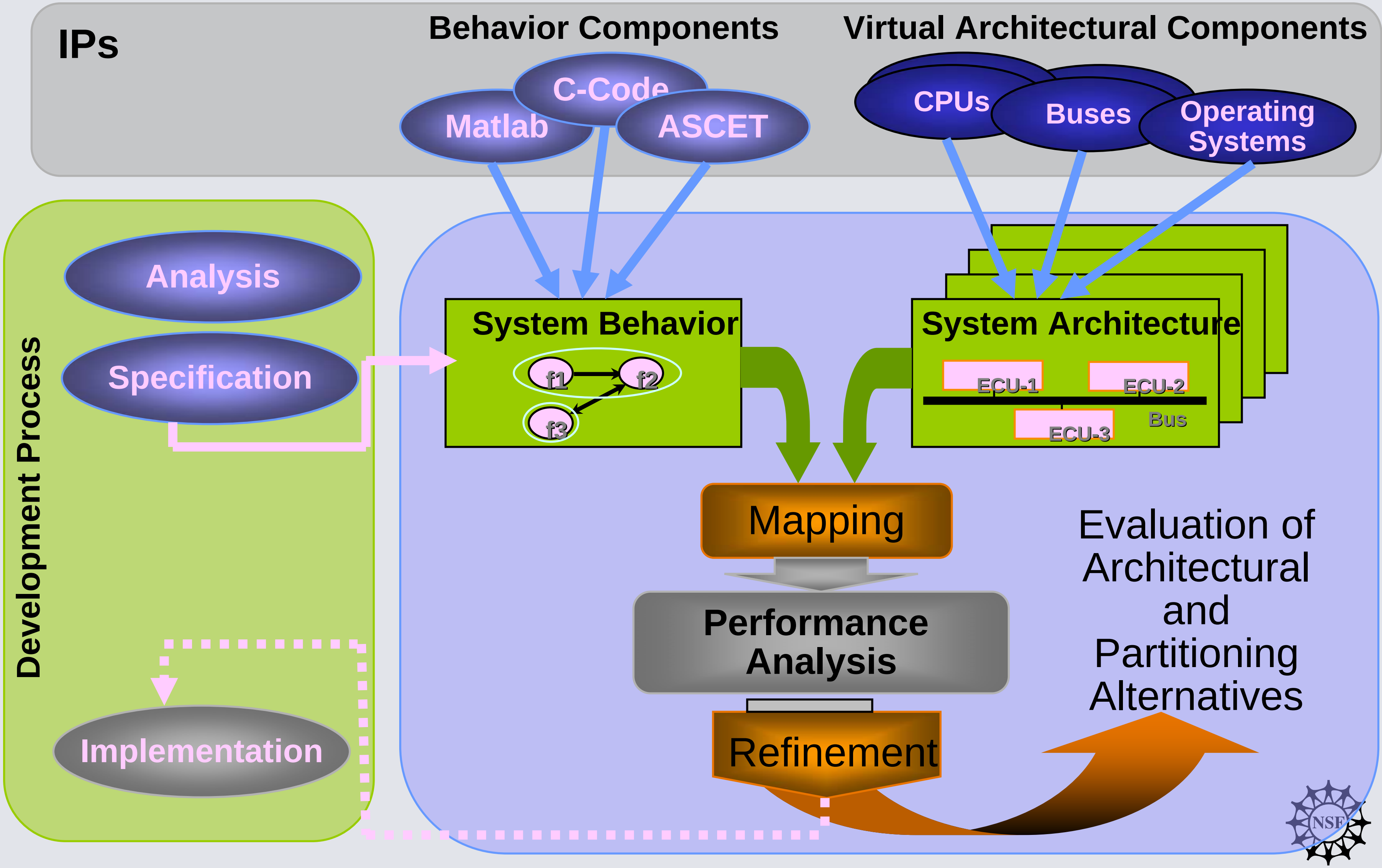
- Standardized, openly disclosed interfaces
- HW independent SW layer
- Transferability of functions
- Redundancy activation

AUTOSAR RTE:
by specifying interfaces and their communication mechanisms, the applications are decoupled from the underlying HW and Basic SW, enabling the realization of Standard Library Functions.

AUTOSAR Organization



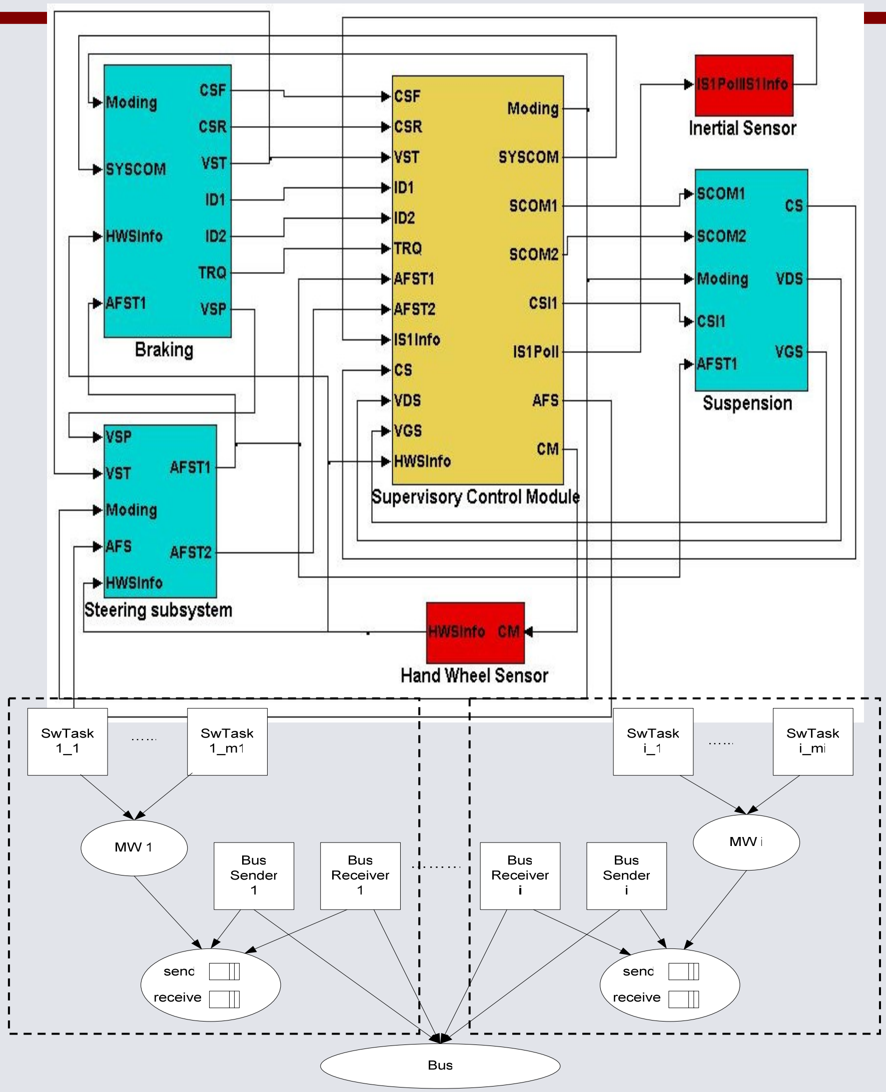
Metro: Separation of Concerns



Design Practice: Mismatch

- Functional Modeling and Code Generation assume uniprocessor implementation….
  - Modeling and stability analysis for control algorithms with Simulink
  - Code generation with RealTime Workshop
- But then code is distributed
- Architectural limitations
  - Shared buffers and clock drift between processors (ECUs)
  - Symptoms: Message loss and duplication
- Current mitigation
  - Limited analysis
  - In-vehicle testing: Expensive, not exhaustive
  - Oversampling: Brute force, too conservative

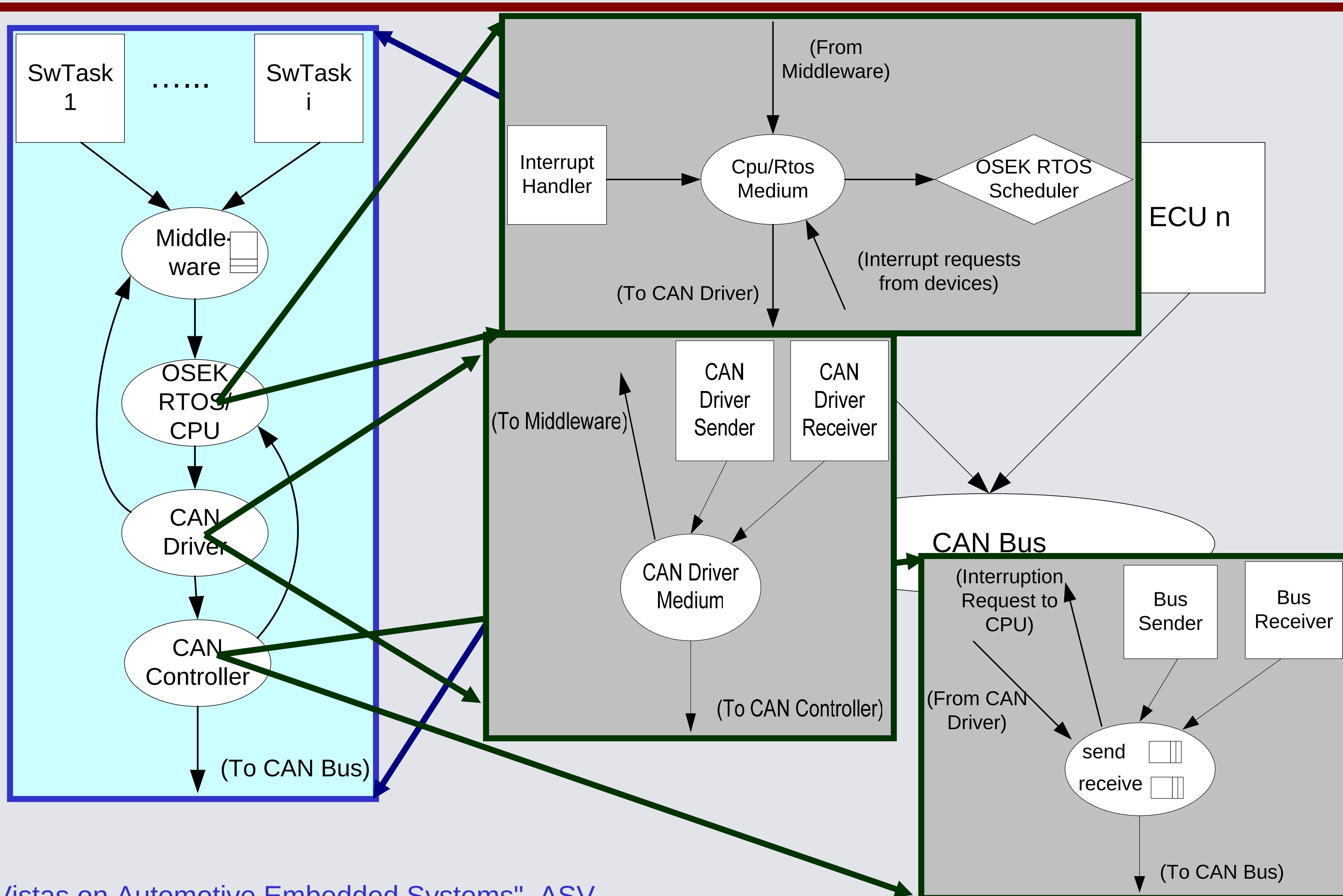
Stabilitrak Case Study with Lossy MoC



- Drive-by-wire application on distributed CAN platform
- System model accurately captures design space
  - Loss and duplication
  - Message latency
  - Priority inversion
- Metropolis library to support lossy MoC

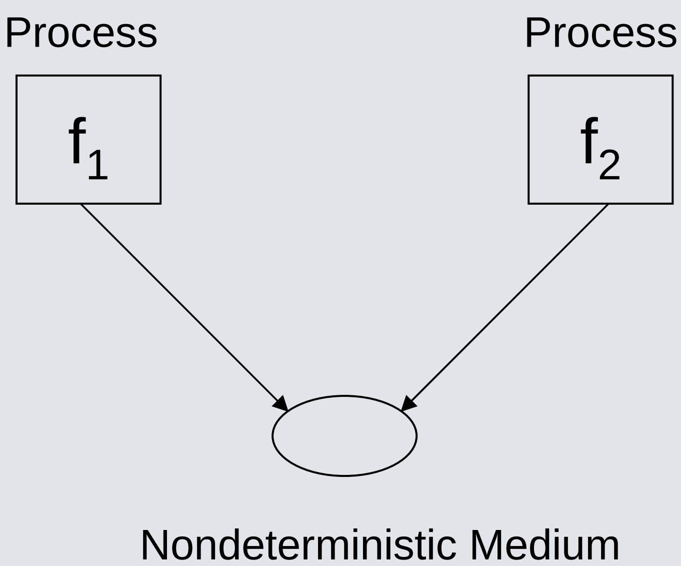
H. Zeng, A. Davare, ASV, S. Sonalkar, S. Kanajan, C. Pinello, “Design Space Exploration of Automotive Platforms in Metropolis,” SAE Cong. 2006.

Architecture Model: Abstraction Levels



Matching Models of Computation

- The functional and architectural models should be described using the same model of computation
- Architecture Characteristics:
  - Network of processes connected by point-to-point FIFOs
  - Non-blocking reads and writes
  - Messages may be lost or duplicated within FIFO
- Functional Model
  - Functional blocks operate concurrently
    - Single rate
    - No synchronization across processes
  - Non-blocking read, non-blocking write communication semantics
- Mapping: intersection of behaviors
  - Before mapping, nondeterministic loss and/or duplication of messages in functional model
  - After mapping, functional loss/duplication follows architecture



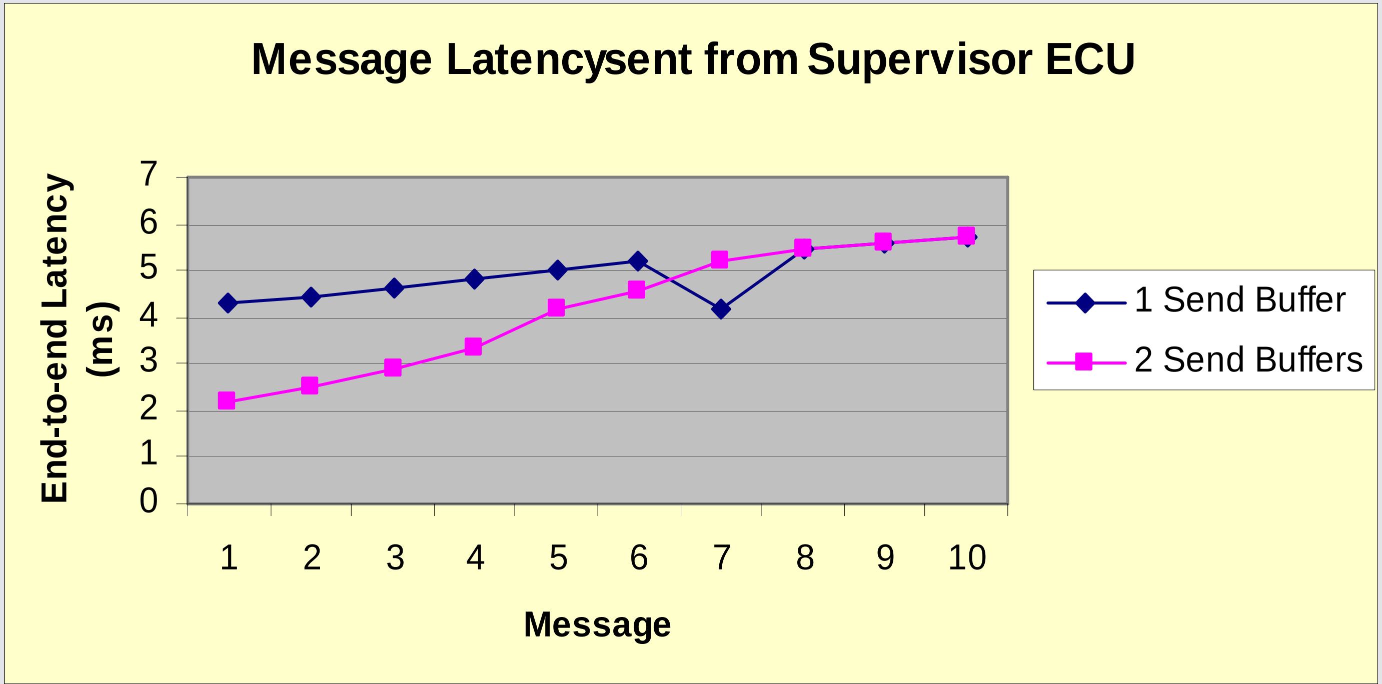
Functional Model

Finding a Compatible MoC

- Two initial options
  - "Handshaking" MoC which guarantees lossless delivery, but with latency overhead
  - "Lossy" MoC which exposes loss and duplication, but with limited functional verification capabilities
    - Point-to-point channels can lose or duplicate data

Results

- Functional Model
  - 14 functional processes
  - 48 signals
- CAN controller configurations:
  - Number of send buffers
- Metric
  - Message End-to-end Latency

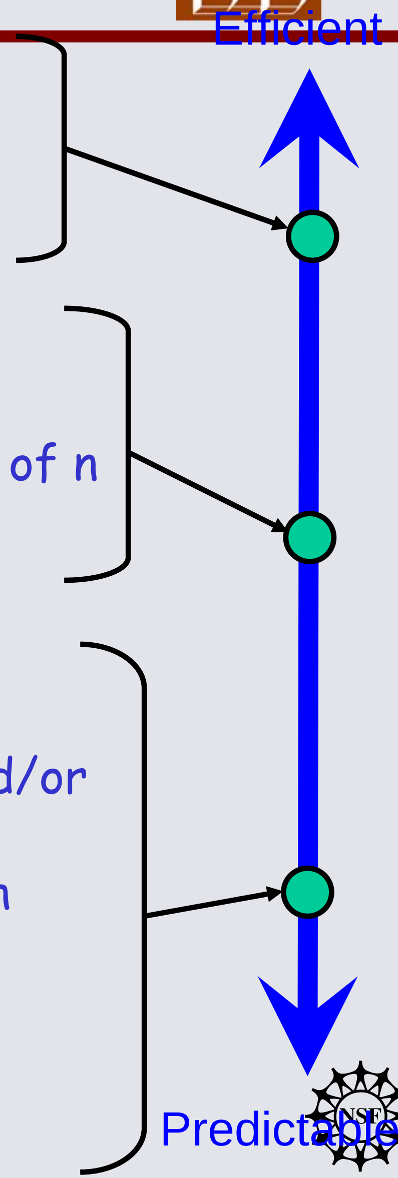


- With 1 send buffer:
  1. Priority inversion: Message 7 < Message 1~6
  2. Average message latency = 4.936ms

- With 2 send buffers:
  1. No priority inversion
  2. Average message latency = 4.165ms

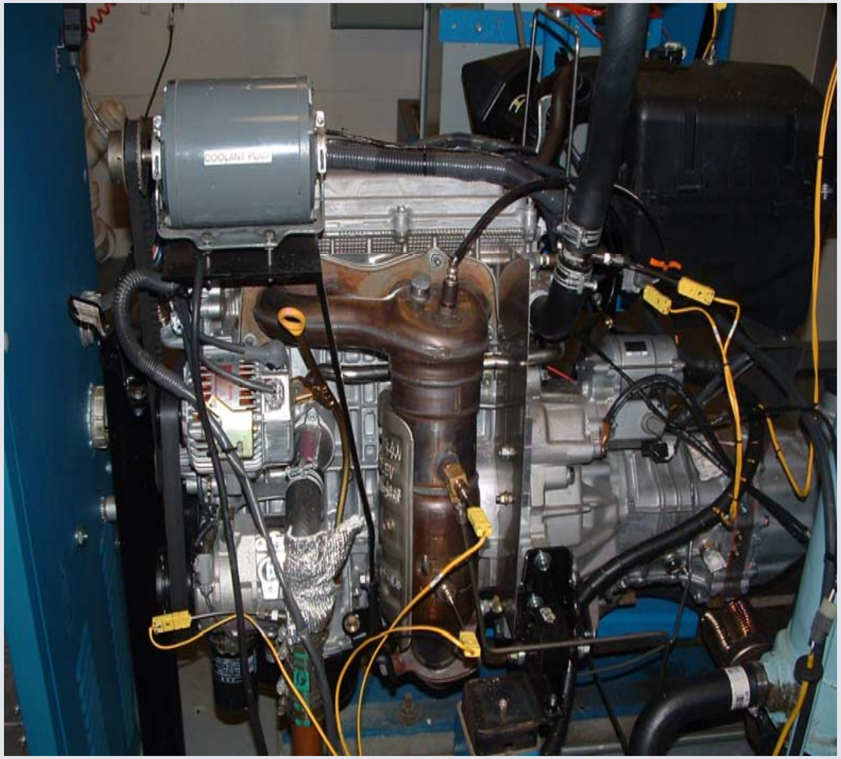
Automotive: Ongoing and Future Work

- Mapping Techniques for lossy MoC
  - Sensitivity criterion for message loss affects mapping decisions
- Alternative MoC that offers slightly stronger analysis capabilities
  - Guarantee that at most one message lost out of sequence of n messages
- Handshaking over unreliable network
  - Synchronous functional modeling
  - Reduce handshaking overhead based on timing analysis and/or allocation of tasks to ECUs
  - A. Davare, K. Lwin, A. Kondratyev, ASV, “The Best of Both Worlds: The Efficient Asynchronous Implementation of Synchronous Specifications,” DAC 2004.



Toyota: Coldstart Engine Controller Design
(C. Zavala and K. Hedrick)

- Objectives:
  - Minimize the HC emissions of cold-start
  - Reduce design-to-implementation controller cycle time.
- Challenges
  - Sensors not active, poor combustion, keep development cost low.
- Strategies
  - Design of AFR and HC observers, use of design of automated tools, use of modern controller design techniques



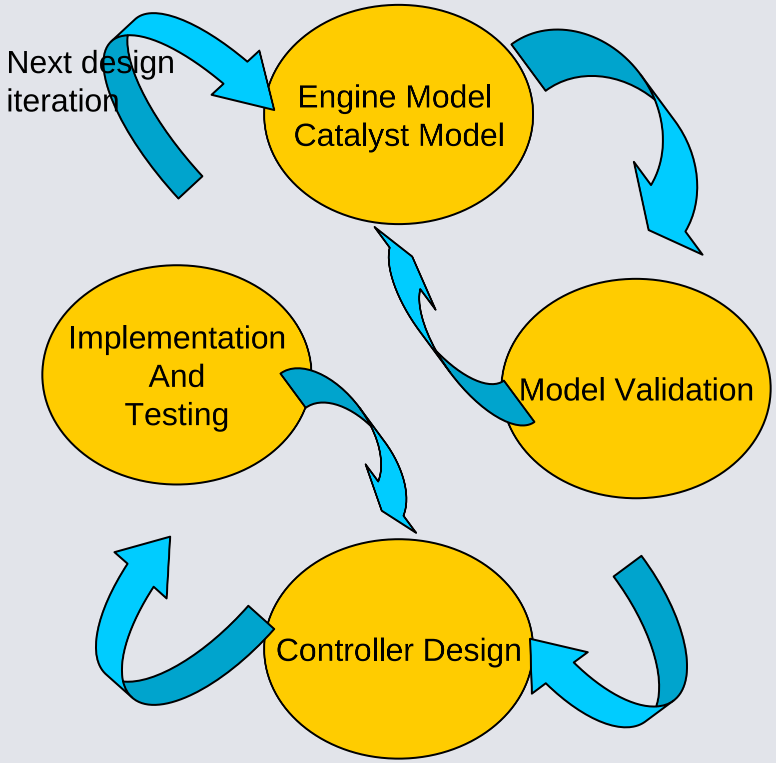
Experimental facilities

Coldstart Engine Modeling and Control

Karl Hedrick, Pannag Sanketi,
Mark Wilcuts, Tomoyuki Kaga,
Carlos Zavala

- Goals:
  - –Minimize the HC emissions of cold-start
  - –Reduce design-to-implementation controller cycle time.
- Requirements
  - – driveability: no noise or vibration, robustness to uncertain external conditions, low calibration effort, reliability in validation.
- Strategies
  - –Design of AFR and HC observers, use of design of automated tools, use of modern controller design techniques

Model Based Strategy



Transmission Control

Goal:

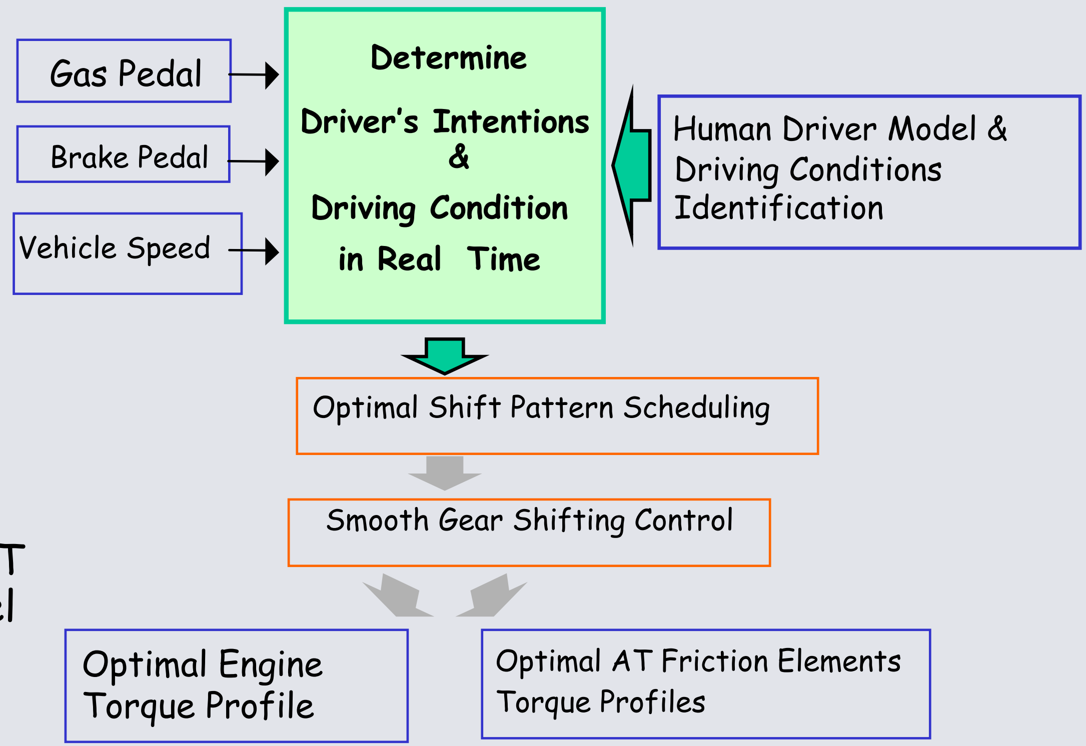
- Improve drivability and fuel efficiency by automotive control.

Approach:

- Utilize dynamical model-based analysis and controller design.

Control Strategy:

- Multi-tiered approach to achieve shock-free gear shifting by smooth gear shifting control with engine/AT collaboration balancing between fuel economy & performance by optimal shift pattern scheduling

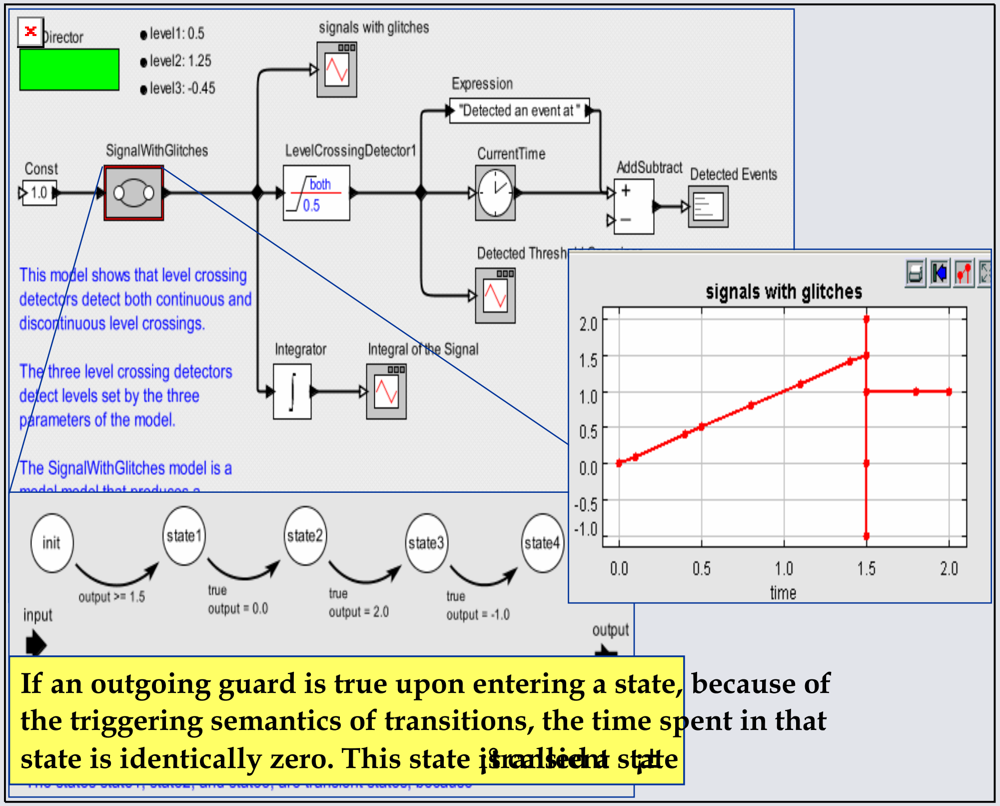


Prospected control structure for intelligent shifting

Hybrid Systems Modeling

Objectives

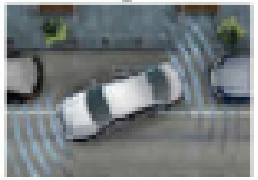
- Hybrid System Analysis: study of a general semantics for simulator engines to execute hybrid system models.
- Study of representations of discontinuities and interactions between continuous-time dynamics and simultaneous discrete events
- The code generation project aims to produce application code automatically from graphical models in Ptolemy II



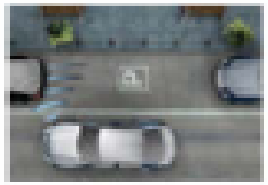
Product and Technology Overview

**Ultra-
Sonic**

Standard Parking



Parking Space Measurement

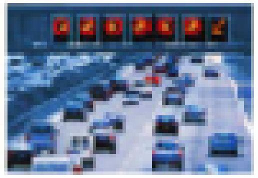


Semi-autonomous Parking Assistant

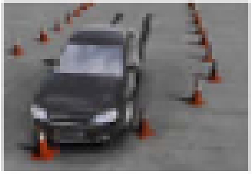
**Long
Range
Radar**

ACC > 30km/h

ACC plus 0 to ~200km/h



Predictive Safety Systems (PSS)



**Integr
Vision
Syste**

Night Vision Support

Lane Departure Warning

Video Park Pilot

Connected Drive

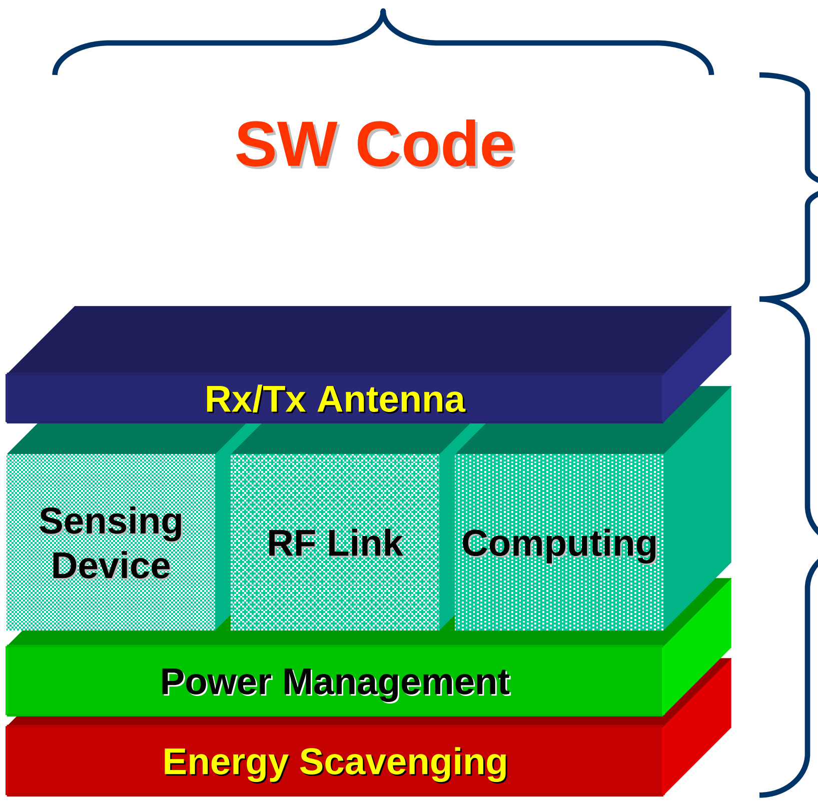
Connected Car-to-Car

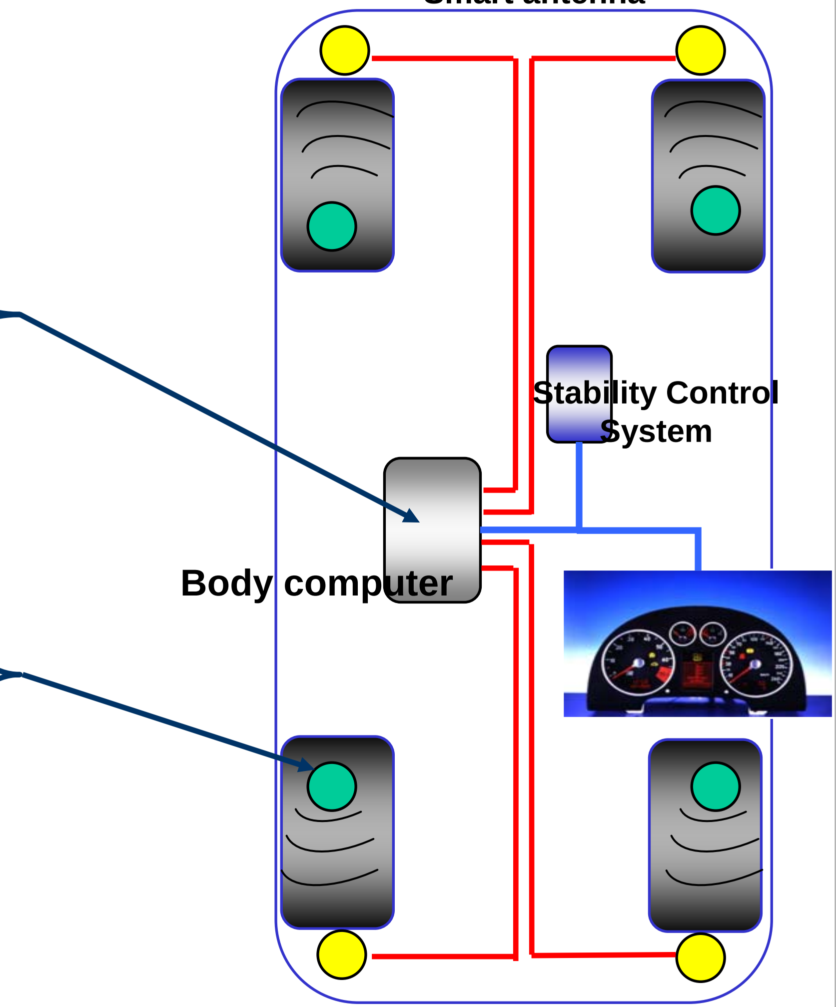
Implication on availability of a spontaneous car-to-car connectivity as part of the connected services concept

Today, there are no vehicle functions (beyond advanced sensors) based on spontaneous car-to-car connectivity

- Opportunities in advanced / new functionalities
- Implications on architectures
- Methods and tools for the design of such systems

Tyre to Vehicle





System Content

L_3

Software Code to Compute

$$S_1 = g_1(E_1, E_2, \ldots, x_1, x_2, \ldots, p_1, \ldots)$$

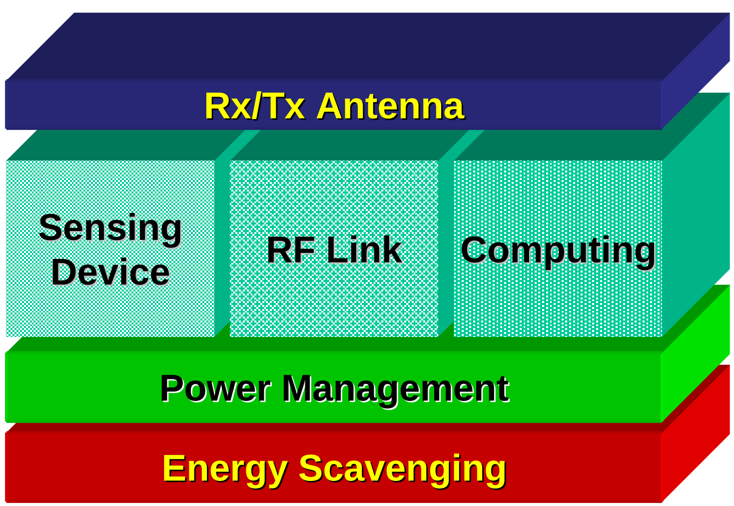
L_2

Software Code to Compute

$$E_1 = g_1(x_1, x_2, \ldots, p_1, \ldots)$$

L_1

Hardware



- 3x3x3, (3 mm^3, 3 grams, 3 $)
- Tyre compatible packaging
- Use of existing vehicle infrastructure
- No de-standardization