Science of System Integration for CPS
NSF CPS Program

Vanderbilt University, ISIS; University of Notre Dame, EE/ME

University of Maryland, ISR; General Motors Research

Project Overview
Janos Sztipanovits

CPS PI Meeting

1 August, 2011

Washington DC

Project Investigators

Panos J. Antsaklis (ND)

John S. Baras (UMD)

Bill Goodwine (ND)

Vijay Gupta (ND)

Gabor Karsai (VU)

Xenofon Koutsoukos (VU)

Janos Sztipanovits (VU)

Shige Wang (GMR)

Project Motivation and Goals

- **System integration**: implemented components are connected and system-level properties are tested
    - High risk – many fundamental problems surface during system integration
    - Ad-hoc – ‘making it work somehow’ attitude
    - Fundamental cause – limited composability and compositionality in heterogeneous systems lead to lack of constructivity in system design
- **System integration is like the Weather**: “*Everybody talks about system integration , but nobody does anything about it*” *(after Mark Twain)*

System Integration in Automotive Industry

- OEMs purchase vehicle control system components from different suppliers and integrate them into products.
- Their strategies, engineering processes and tools are different
- Smart sensors and actuators, wireless networks, and multi-core processors make the vehicle control integration challenge far worse

Needs:

- Design, analysis, and verification of components on various levels of abstraction (system, software, platform) but subject to the constraints from other levels
- Composing and analyzing the interactions between vehicle control and physical systems (e.g. engine, transmission, brake, suspension, etc.) to ensure system-level properties (e.g. stability, safety, performance, cost)
- System-level behavior simulation with incomplete or limited details

Know-how in system integration is increasingly the differentiator in competitiveness in automotive as well as other major industrial sectors such as aerospace, health-care, and defense

Project Goals

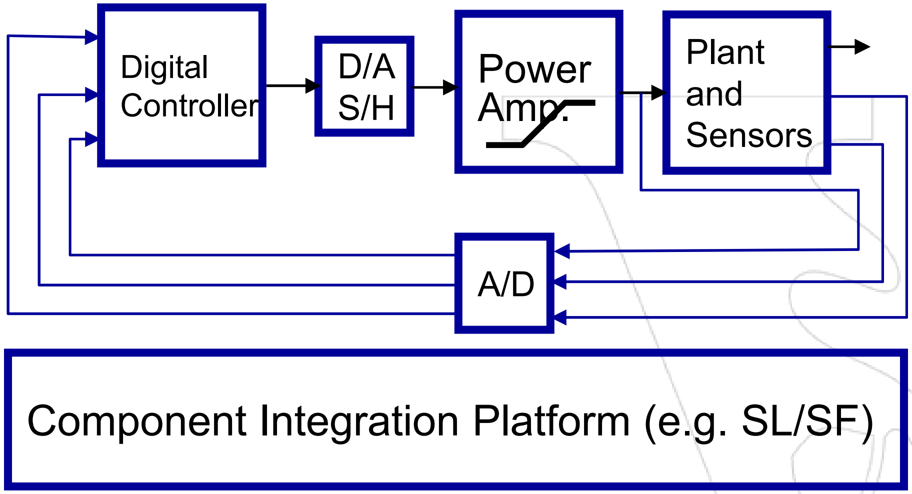
1. Composition and compositionality of heterogeneous systems to achieve constructivity and predictability in CPS integration
2. Composition of tool chains for CPS integration based on semantically rigorous methods to define and compose of heterogeneous modeling languages.
3. Experimental validation of the ideas in automotive and avionics application

Problem Categories in CPS Integration

- Components (horizontal)
- Layers (vertical)
- System of Systems

Component Integration

Functional: E.g.: Dynamics



Digital Controller

D/A S/H

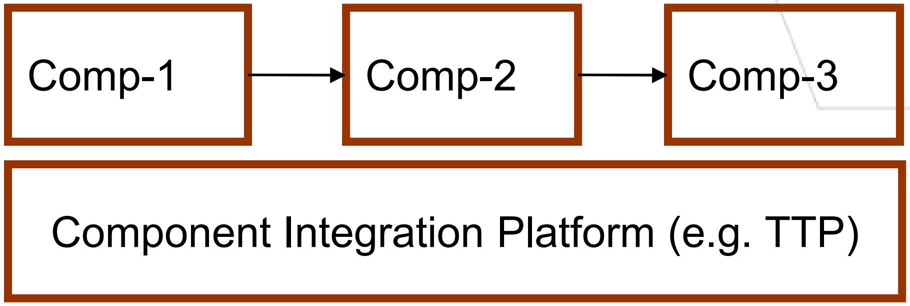
Power Amp.

Plant and Sensors

A/D

Component Integration Platform (e.g. SL/SF)

Software: E.g. Timing



Comp-1

Comp-2

Comp-3

Component Integration Platform (e.g. TTP)

- Composability and compositionality are key concepts
- Defined for carefully selected properties (stability, latency, power,..)
- Decomposed into *structure*, *interaction*, and *behavior*
- ***Challenges***:
  - ***composition frameworks* providing *constructivity* for essential properties**

Layered Systems: Vertical Integration

Roles

Cognitive processes
Social interaction
Command and control

Coordination
Data distribution

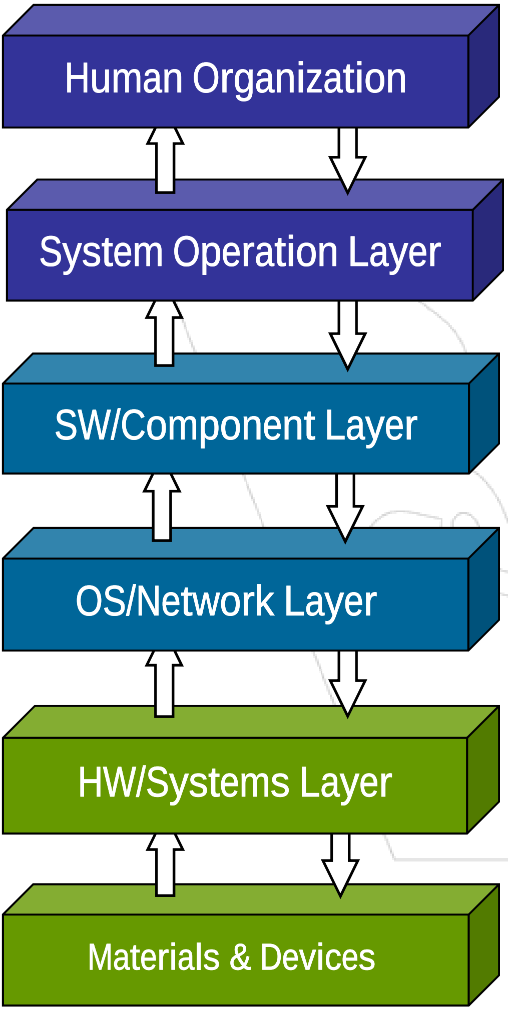
Component interactions
Component behaviors
Architecture

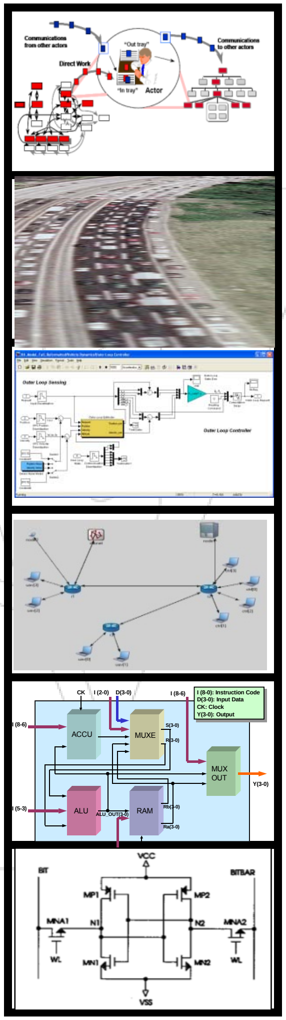
Resource management
Scheduling
Separation

Timing/performance
Fault management
Power management

Heat dissipation
Crossover
Radiation effects

Layers

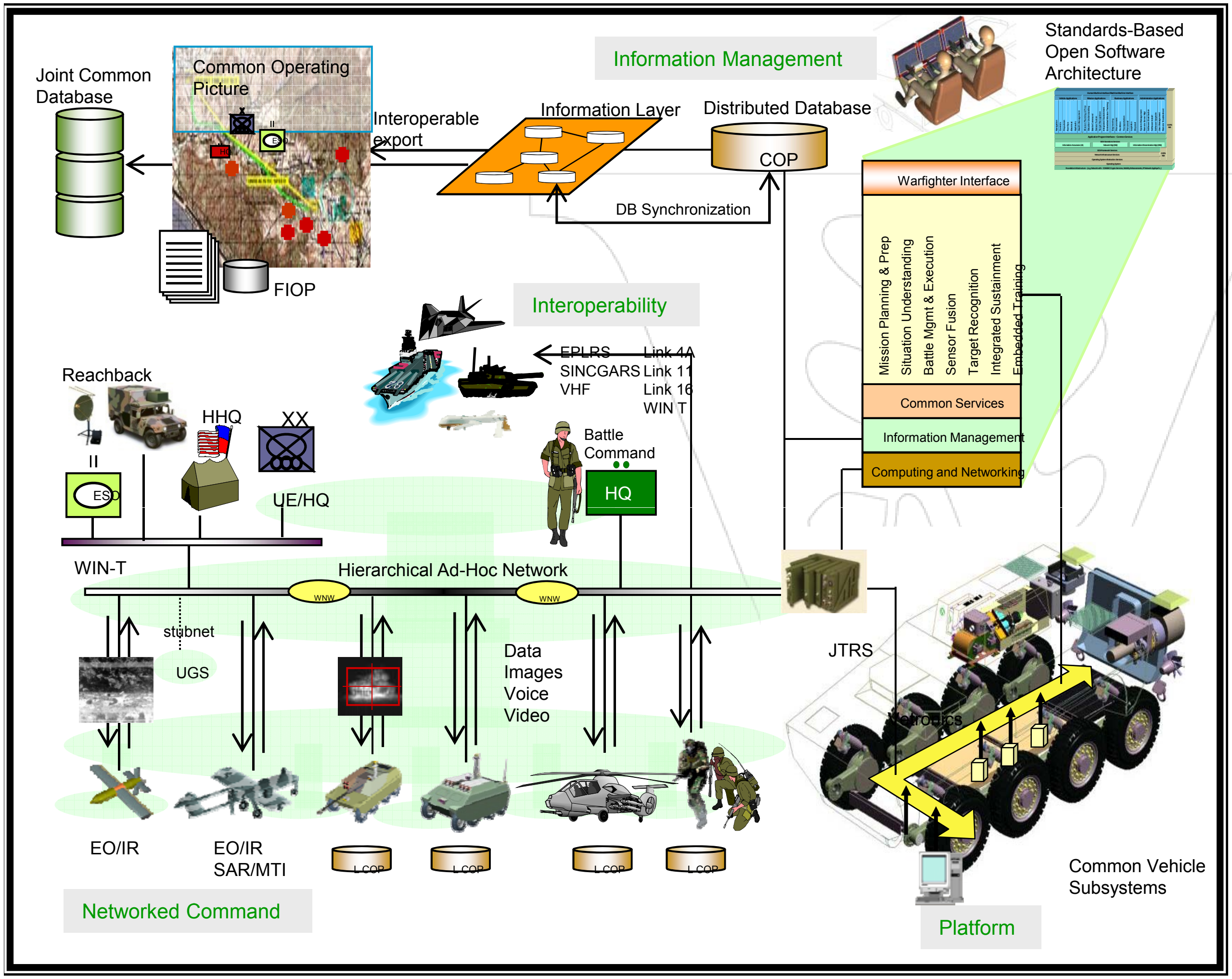




Characteristics

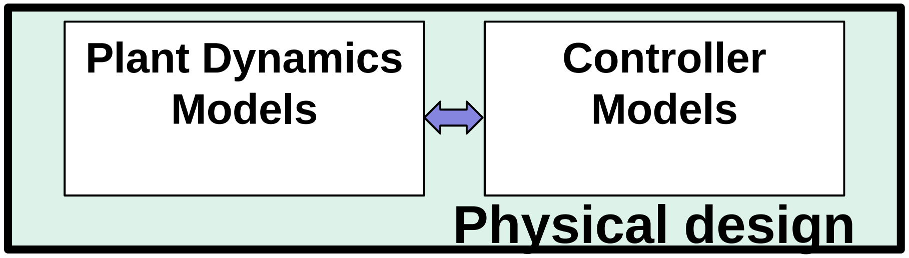
- Inter-layer interactions
- Effects propagate across the layers
- Efficiency and optimization drives toward intractability
- Inter-layer relationship:
  - mapping
  - refinement
  - synthesis
- ***Challenges*:**
  - **modeling,**
  - **constraining**
  - **composing**

System of System Integration



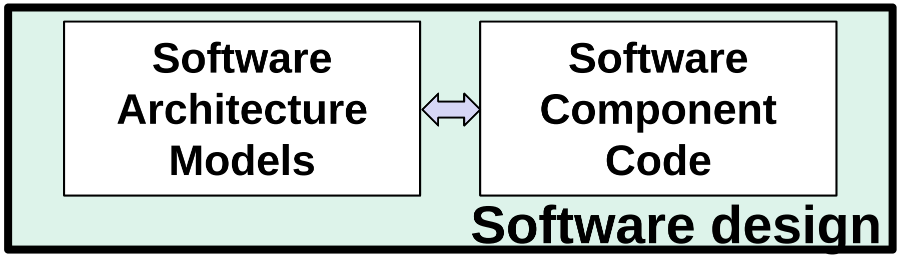
- Heterogeneous
- Open Dynamic Architecture
  - heterogeneous networking
  - heterogeneous components
- Very high degree of concurrency with complex interactions
- *Challenges*:
  - understanding and
  - predicting behavior

Integration Inside Abstraction Layers: Composition



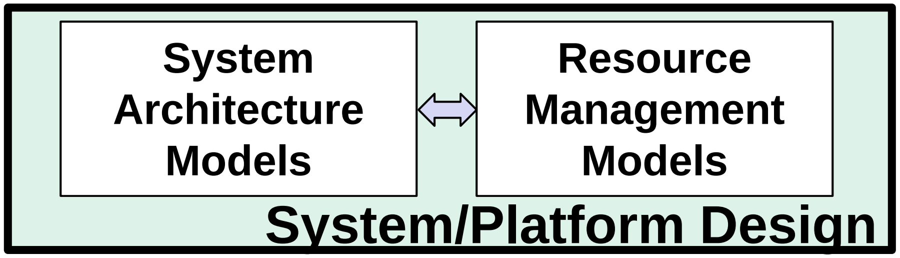
Dynamics: $B(t) = \kappa_p(B_1(t),\ldots,B_j(t))$

- *Properties*: stability, safety, performance
- *Abstractions*: continuous time, functions, signals, flows,…



Software : $B(i) = \kappa_c(B_1(i),\ldots,B_k(i))$

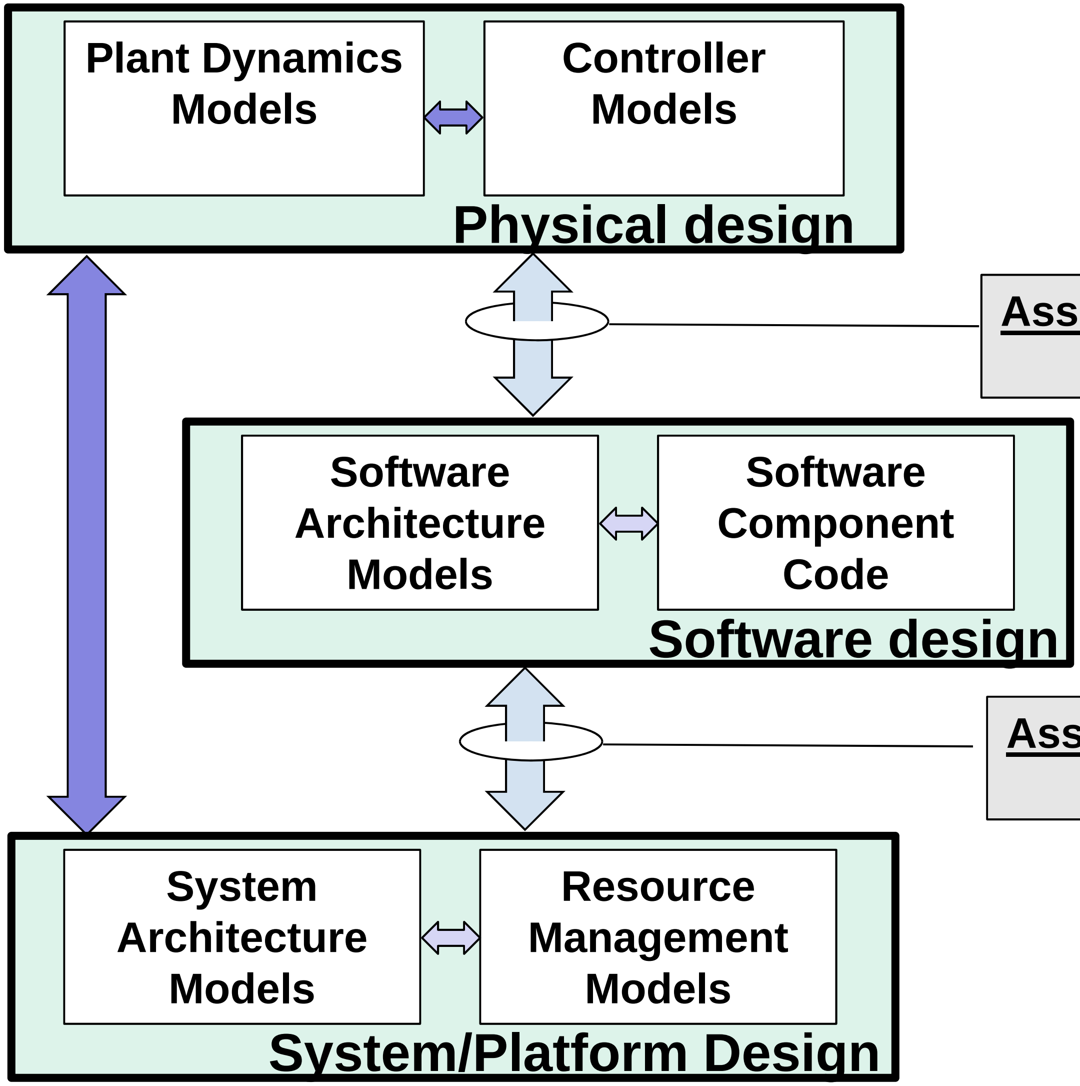
- *Properties*: deadlock, invariants, security,…
- *Abstractions*: logical-time, concurrency, atomicity, ideal communication,..



Systems : $B(t_j) = \kappa_p(B_1(t_i),\ldots,B_k(t_i))$

- *Properties*: timing, power, security, fault tolerance
- *Abstractions*: discrete-time, delays, resources, scheduling,

Integration Across Abstraction Layers: Much Unsolved Problems



Controller dynamics is developed without considering implementation uncertainties (e.g. word length, clock accuracy) optimizing performance.

Assumption: ~~Effects of digital implementation can be neglected~~

Software architecture models are developed without explicitly considering systems platform characteristics, even though key behavioral properties depend on it.

Assumption: ~~Effects of platform properties can be neglected~~

System-level architecture defines implementation platform configuration. Scheduling, network uncertainties, etc. are introduce time variant delays that may require re-verification of key properties on all levels.

Challenge to Compositionality: Heterogeneity

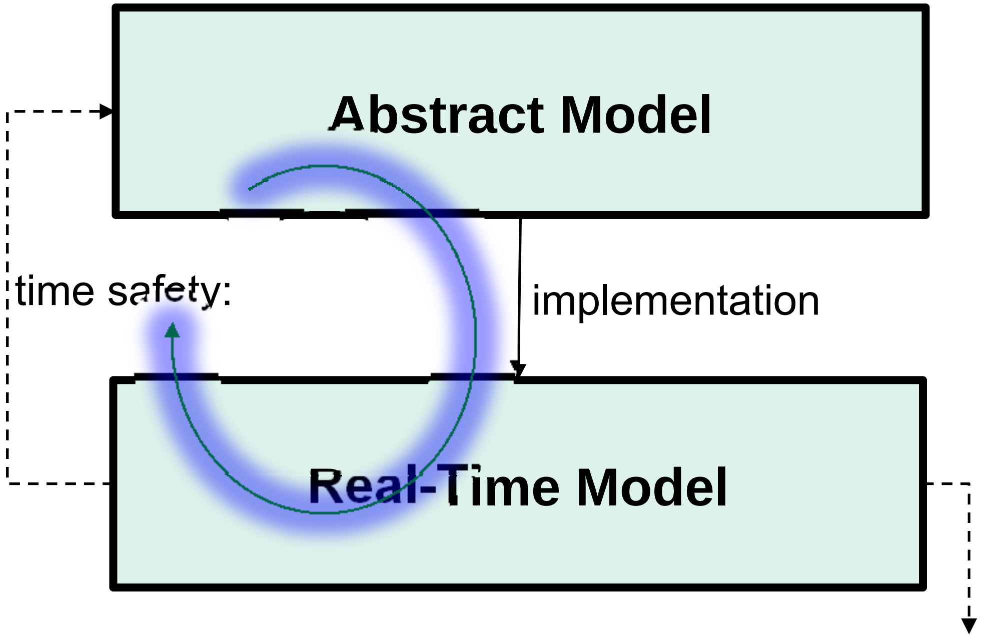
- Consequences of lack of composability across system layers:
  - intractable interactions
  - unpredictable system level behavior
  - full-system verification does not scale
- Project goals: simplification strategies
  - ***Decoupling:*** **Use design concepts that decouple systems layers for selected properties**
  - ***Cross-layer Abstractions:*** **Develop methods that can handle effects of cross-layer interactions**

The golden rule of problem solving:
If the problem gets tough, the tough changes the problem.

Decoupling Example 1: Robust Implementation of R-T Systems

Abdellatif, Combaz, Sifakis [2010]:
Model-Based Implementation of
Real-Time Applications

However, essential system properties such as stability, safety, performance are expressed in terms of physical behavior

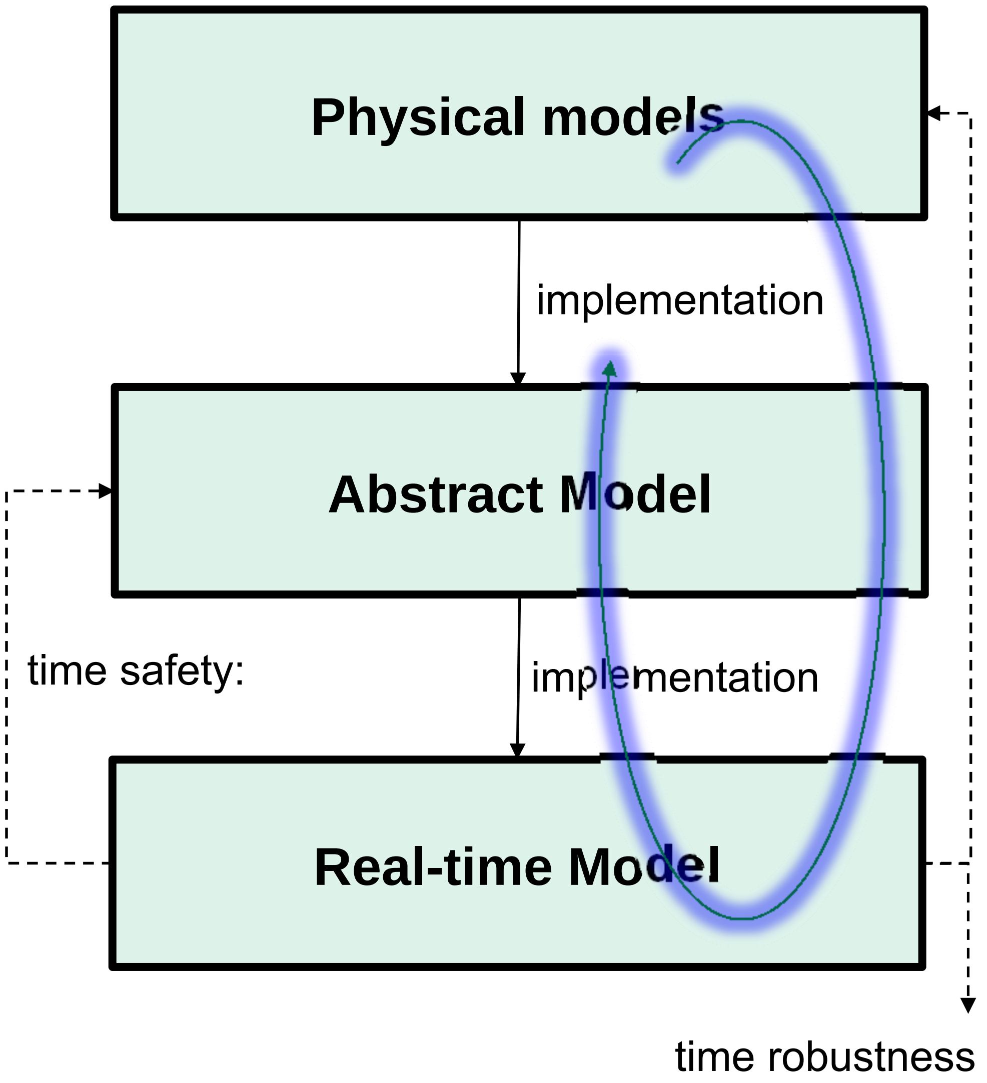


- M : Based on Logical Execution Time (LET)
  Based on Timed Automata
  Actions are atomic and timeless
  They can be executed after release time and before the due time

- M_φ : real-time system. Models the behavior of the software on a platform. Actions are assigned with WCET

time robustness

Decoupling Example 2: Passive Dynamics



Goals:

- **Compositional verification of essential dynamic properties**
  - stability
  - safety
- **Passivity guarantees stability independently from implementation induced uncertainties**
  - time varying delays
  - network uncertainties (packet drops, delays)
- **Decreased verification complexity**

Kottenstette, N., J. Hall, X. Koutsoukos, P. J. Antsaklis, and J. Sztipanovits, "Digital Control of Multiple Discrete Passive Plants Over Networks", *Int. J. of Systems, Control and Communications 2010*

Project Organization

- **PI:** Janos Sztipanovits
- **coPIs:** Panos Antsaklis, John Baras, Gabor Karsai, Xenofon Koutsoukos, Shige Wang
- **Participating Institutions:** Vanderbilt, Notre Dame, University of Maryland, General Motors Research
- **Five Thrusts:**
  - Theory of Compositionality for Heterogeneous Systems
    - Physical Layer (Antsaklis)
    - Software Layer (Koutsoukos)
    - Network/Platform Layer (Baras)
  - Tools and Tool Architectures for System Integration (Karsai)
  - Systems/Experimental Research
    - Automotive Platform (Wang)
    - Fleet of Quadrotors (Kottenestette)
- **Five year project: kick-off meeting November 29-30, 2010**
  - Bi-weekly seminar series
  - Annual workshop, review meeting

Thrust 1-3: Compositionality Theories for Heterogeneous Systems

- Physical Layer
  - Decoupling the design of physical processes from implementational side effects using passive dynamics
  - Compositionality for stability, safety and performance properties
- Software Layer
  - Model-based software design for system integration
  - Compositionality for deadlock and invariance
  - Dynamics decoupling using passive structures
- Network/Platform Layer
  - Platform decoupling
  - Network topology

Thrust 4: Tools and Tool Architectures for System Integration

- Compositional Framework for Tool Integration
  - Compositional Metamodeling
  - Compositional construction of verifiable model transformations
  - Composition of tools with explicit dependencies and incremental change propagation
- Multi-model Simulation Environment for Incremental System Integration
  - Virtual prototyping

Thrust 5: Systems/Experimental Reserach

- Automotive Platform: Electrical Vehicle
  - Challenge Problem #1: Feature Integration
  - Challenge Problem #2: Technology Integration

- Networked Control System Platform: Fleet of Quadrotors
  - Coordinated motion control scenarios with stability, safety, and performance properties

Research Highlights: Foundations for Passivity-Based Design

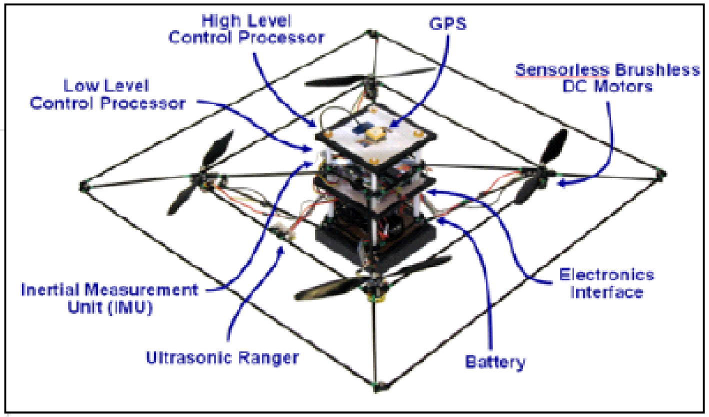
Recent results (Notre Dame) include:

- Extending notions of Passivity and Passivity Indices and Dissipativity to Switched and Hybrid Systems. Stability of Networked Passive Switched Systems. Fundamental work on Passivity of Systems in Cascade, on Networked Systems on Hybrid and Discrete Event Systems.
- Robust/Reliable Stabilization of Multi-Channel Systems via Dilated LMIs and Dissipativity-Based Certifications. Characterization of Feedback Nash Equilibria for Multi-Channel Systems via a Set of Non-Fragile Stabilizing State-Feedback Solutions and Dissipativity Inequalities.
- Multi-agent Coordination Exploiting System Symmetries. Passivity Indices for Symmetrically Interconnected Distributed Systems.Symmetry in the Design of Large-Scale Complex Control Systems using Dissipativity and Lyapunov Stability.

Research Highlights: Robust NCS Architectures

Recent results (Vanderbilt) include:

- Group coordination of networked agents
  - Surveillance and convoy tracking applications
  - Establish formation around a target
- Distributed algorithms using local communication
- Account for network delays and data loss
- Discrete-time distributed design using passivity
- Ensure l_2^m- stability
  - In the presence of network delays and data loss
  - Regardless of the overlay network topology
- Asymptotic formation establishment and output synchronization
- Collision avoidance
- Simulations using quadrotor UAVs
  - Simulink/TrueTime
  - NCSWT


High Level Control Processor
GPS
Sensorless Brushless DC Motors
Low Level Control Processor
Inertial Measurement Unit (IMU)
Electronics Interface
Ultrasonic Ranger
Battery

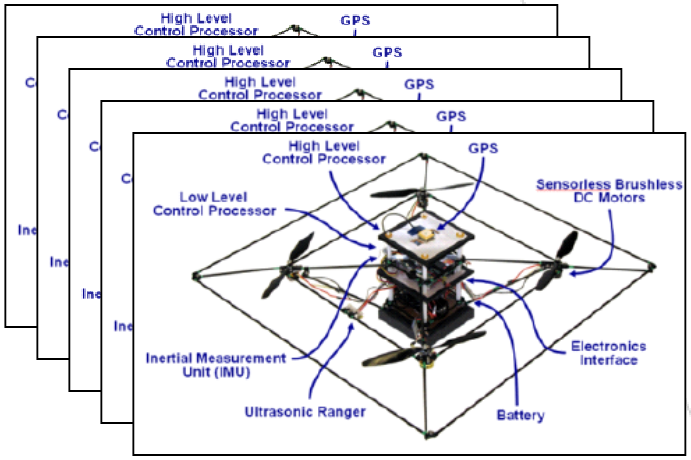
Research Highlights: Systems Heuristics for Complexity

Recent results (Notre Dame) include:

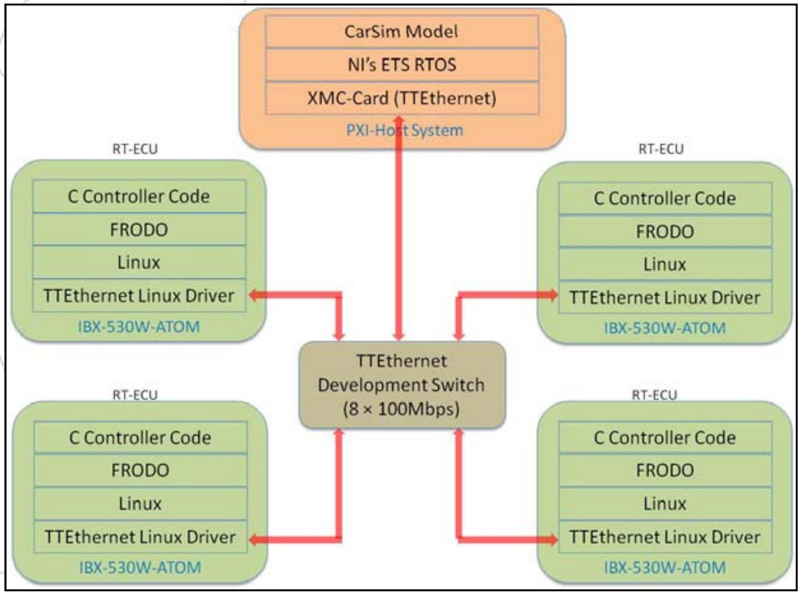
- Many intuitive design heuristics: keep coupled variables in close proximity; encapsulate data; build hierarchies...
- Interpretation within a factor graph framework causes these heuristics to become quantifiable design metrics that reduce the complexity of analyzing systems.
  - Treewidth of factor graphs generally makes the dominant contribution to analysis complexity, being exponential in this term.
  - The next factor is the size of the overall system, but complexity is only *linear* in this term.
  - For chordal graphs, treewidth is easily computed and equal to the size of the maximum clique.
- Chordality may be a useful design principle.
  - Chordal graphs are uniquely representable as a clique-separator graph.
  - The clique-separator graph generates all the join trees from the graph.
  - Join trees provide topologies that are consistent with both computation and communication requirements if one wants to build a distributed implementation

Research Highlights: Experimental Platforms

- Networked Control System Platform: Fleet of Quadrotors


High Level Control Processor
GPS
Low Level Control Processor
Sensorless Brushless DC Motors
Electronics Interface
Inertial Measurement Unit (IMU)
Ultrasonic Ranger
Battery

- Electrical Vehicle Platform GMR



CarSim Model
NI's ETS RTOS
XMC-Card (TTEthernet)
PXI-Host System

RT-ECU
C Controller Code
FRODO
Linux
TTEthernet Linux Driver
IBX-530W-ATOM

RT-ECU
C Controller Code
FRODO
Linux
TTEthernet Linux Driver
IBX-530W-ATOM

TTEthernet
Development Switch
(8 × 100Mbps)

RT-ECU
C Controller Code
FRODO
Linux
TTEthernet Linux Driver
IBX-530W-ATOM

RT-ECU
C Controller Code
FRODO
Linux
TTEthernet Linux Driver
IBX-530W-ATOM

Project Plan

Year	Milestone	Success Criteria
Year 1	*End-to-end model-based system integration.* Baseline modeling and system integration tool chains are demonstrated in both testbeds with models in all design layers and automated integration on real platform.	Integrated systems are fully functional. Tool chains are customized to testbeds.
Year 2	*Demonstration of impact of decoupling on system integration in the automotive and networked control testbed.* Theory is validated by measuring resilience of stability against changes in controller implementation (scheduling, clock rate, load changes, comm. delays). Demonstration of integration of deadlock/invariance analysis in tool chain.	Experimental proof that stability of the physical platforms are preserved under adverse implementation changes.
Year 3	*a.) Demonstration of maintaining stability, safety and performance requirements* while partially reconfiguring control architectures (features) in both testbeds. *b.) Demonstration of tool chain reconfiguration* without losing semantic integrity	Experimental proof of safety guarantees and performance optimization under decoupling. Semantics-based integration of tools.
Year 4	*a.) Demonstration of maintaining stability, safety and performance requirements* while executing platform architecture change in both testbeds. *b.) Demonstration of rapid integration of new tool components* required by the platform change	Experimental proof that platform reconfiguration and change can be completed without changing controller architecture.
Year 5	*Demonstration of incremental and continuous system integration process* for evolving vehicle architectures and the feasibility and practicality of virtual system integration in both testbeds	Experimental proof that the model-generated (simulated) behavior and physical system behavior correlates in stability, safety and performance metrics