Multi-View Consistency in Architectures for Cyber-Physical Systems

1 Ajinkya Bhave, Bruce Krogh, Akshay Rajhans
2 David Garlan, Sarah Loos, Andre Platzer, Bradley Schmerl

1Dept. of Electrical & Computer Engineering
2School of Computer Science
Carnegie Mellon University

14 June, 2011
A Cyber-Physical System (CPS): STARMAC Quadrotor*

*http://hybrid.eecs.berkeley.edu/starmac/
Multiple Models Of A CPS

Physical Model
Multiple Models Of A CPS

Control Model

Physical Model
Multiple Models Of A CPS

Control Model

Software Model

Physical Model
Multiple Models Of A CPS

Control Model

Software Model

Physical Model

Hardware Model
Multiple Models Of A CPS

Control Model  ⇌ physical  ⇌ cyber  ⇌ Software Model

Physical Model

Hardware Model
Motivation

■ Developing complex cyber-physical systems requires analyses of multiple models using different formalisms and tools

■ How can we:
  • guarantee the models represent the actual system?
  • guarantee models are consistent with each other?
  • infer system-level properties from heterogeneous analyses of heterogeneous models?
Related Approaches

- **Universal Modeling Language (e.g., SysML)**
  - Create a language that encompasses *everything*
  - Intractable for complex systems
  - Does not support separation of concerns

- **Model transformations (e.g., HSIF, GME)**
  - Automatically translate between models
  - Universal translation requires universal language
  - Tool-specific translation not scalable
An Architectural Framework For CPS

**Goal:** Unify heterogeneous models through *light-weight* representations of their structure and semantics using *architecture description languages* (ADLs)

- Does each model adhere to system structure & constraints (consistency)
- Are all system elements represented in at least one model (completeness)
What Is Architecture?

The set of structures needed to reason about the system, which comprise functional elements, relations among them and properties of both*

- We focus on component-connector runtime architectures of a system
- Current ADLs are more focused on the cyber side of systems

* Documenting Software Architecture: Views and Beyond, 2nd Ed. Clements et al. 2010.
What Should Architecture Address?

- **System’s base architecture** defines
  - component connectivity & physical coupling
  - data, control, & physical signal flows

- Allow each model structure to be checked against base architecture throughout system design process

- Code generation not focus of current framework
Extending Architecture To Support CPS

- ADLs that describe both cyber and physical elements (and their interactions) with equal expressiveness

- Relating models from different frameworks to underlying system architecture

- Reconciling multiple system views to have
  - consistent structure
  - compatible semantics
Adding Support For Physical Domains*

- Allow physical sub-systems (with dynamics) to be represented in the base architecture
- Assess design trade-offs and structural impact across cyber-physical boundary
- Define relations between heterogeneous models (cyber and physical) for design integrity

*Augmenting Software Architectures With Physical Components, Bhave et. al., ERTS² 2010.
CPS Architectural Style

Architectural extension based on mathematical framework for interconnected physical systems*

- **Components**: Physical elements (energy sources, stores, dissipators, converters, ...)
- **Connectors**: Physical interactions (conservation laws, energy flows, ...)
- **Properties**: Dynamic behavior of elements (DAEs, LHA, ...)

Interface elements link physical elements to cyber elements (sensors, actuators, ...)

Quadrotor CPS Architecture

- GPS
- Position_Ctrl
- Gnd_Station
- Attitude_Ctrl
- IMU
- Act_1
- Act_2
- Act_3
- Act_4
- Sonar
- Vehicle_Frame
- WindVelocity
- AirDrag
- Gravity
- Battery
- Ultrasonic Ranger
- Electronics Interface
- Brushless Motors
- High Level Control Processor
- Low Level Control Processor
Quadrotor CPS Architecture

Cyber elements
Standard software
ADL components
and connectors
Quadrotor CPS Architecture

Physical components

Energy sources, stores, dissipators
Quadrotor CPS Architecture

“Kirchhoff” laws: energy conservation, force balance, ...

Physical coupling connectors

Position_Ctrl
Gnd_Station
Attitude_Ctrl
Vehicle_Frame
GPS
IMU
Act_1
Act_2
Act_3
Act_4
Sonar
WindVelocity
AirDrag
Gravity
Quadrotor CPS Architecture

Equality of physical variables from different subsystems

Physical equality connectors
Quadrotor CPS Architecture

- Inertial, position, altitude sensors, pwm control signals

Interface elements
Models As Architectural Views*

- Most models constructed as collections of interacting components or modules
- Each model has structure with syntax and semantics defined by modeling formalism
- Define consistent relations between these models at some level of abstraction

*Multi-domain Modeling of Cyber-Physical Systems Using Architectural Views, Bhave et. al., AVICPS (RTSS) 2010.
Models As Architectural Views

Model $M_X$

Model $M_Y$

Base CPS Architecture
Models As Architectural Views

Model $M_X$

View $V_X$

Model $M_Y$

View $V_Y$

Base CPS Architecture
Models As Architectural Views

View $V_X$ to Model $M_X$ relation: $R_{VX}^{Mx}$

View $V_Y$ to Model $M_Y$ relation: $R_{VY}^{My}$

Model $M_X$ to View $V_X$ relation: $R_{BA}^{Vx}$

Model $M_Y$ to View $V_Y$ relation: $R_{BA}^{Vy}$

Base CPS Architecture
Models As Architectural Views

Model $M_X$  →  View $V_X$  →  Base CPS Architecture  →  View $V_Y$  →  Model $M_Y$

$R^{My}_{V_Y}$  \\
$R^{Vx}_{BA}$  \\
$R^{Vx}_{Vx}$  \\
$R^{Mx}_{Vx}$

model-to-view relations

view-to-base-arch. relations

Base CPS Architecture
Models As Architectural Views

- X is the design concern of view
- $A_{Vx}$ is derived from model structure
- Element types are defined by X

Base CPS Architecture
Models As Architectural Views

- 1:1 or encapsulation of model entities
- creates componentized model structure
- semantics based on modeling formalism
Models As Architectural Views

- encapsulation or refinement of elements in base arch.
- many-to-many maps not allowed

Base CPS Architecture
Creation Of Control View

A_{Control View}

Simulink model
Control View To BA Mapping

$A_{ControlView}$

$R^{Vx}_{BA}$

Base Arch
Structural View Consistency*

- Check if model makes valid assumptions about structure of underlying system
- Ensures equivalent component connectivity and physical signal flows with base arch.
- Assures that design changes to system structure are reflected correctly in models

*View Consistency In Architectures For Cyber-Physical Systems, Bhave et. al., ICCPS (CPSWeek) 2011.
How Can Views & BA Be Related Structurally?

Control View

Physical View

Software View

Hardware View

Base Architecture
How Can Views & BA **Graphs** Be Related?

- Control View
- Physical View
- Software View
- Hardware View

Base Architecture

Graph Morphism
View Consistency With Base Arch

- A monomorphism exists between view and base arch graphs

- Enforces that
  - Every component in the view should be accounted for in the BA
  - Every communication pathway and physical connection existing between view elements should be allowed in the BA

- Useful when view focuses on a sub-part of system
View Completeness With Base Arch

- An isomorphism exists between view and base arch graphs
- Enforces that every element in the BA must be represented in the view in some manner
- **Coverage** condition rather than a sufficiency check
- Useful when view is describing whole system
Control View Inconsistency

Control View

Base Architecture
Inconsistency Traced To Model
Inconsistency Corrected
Possible Impact On System Design

- Mismatch of connectors from GPS sensor to attitude controller
  - represents position and velocity readings
  - could expose wrong assumptions about source of sensor readings
  - could affect stability and performance guarantee of attitude control algorithm
Control View Incompleteness
Possible Impact On System Design

- **Missing connector from sonar sensor to attitude controller**
  - represents error-prone height readings
  - could affect complexity of onboard control algorithm

- **Missing connector from ground station to position controller**
  - represents telemetry data transmission
  - could affect wireless channel bandwidth assumptions
Parametric View Consistency*

- First step towards describing semantic model inter-dependency at architectural level
- Allows verification of heterogeneous models by exposing key model parameters
- Check if each model has correct and consistent assumptions about behavior of rest of system

*Using Parameters In Architectural Views To Support Heterogeneous Design And Verification, Rajhans et. al., CDC 2011 (submitted).
Parameterized Architecture Views

System Models

\[ C_1(P_1), M_1 \models S_1 \]

\[ C_2(P_2), M_2 \models S_2 \]

\[ \ldots \]

\[ C_n(P_n), M_n \models S_n \]

Arch. Views

\[ C_1^{ext} \]

\[ C_2^{ext} \]

\[ \ldots \]

\[ C_n^{ext} \]

Base Arch.

\[ C_{aux} \]

\[ P_0 \]

Universal Model

\[ C_0(P_0), M_0 \models S_0 \]
AcmeStudio Tool Framework

- Semantically extensible framework for architecture design and analysis
- Rich toolset for integration of externally developed tools
- Inbuilt support for
  - system structure and connection constraints
  - para-functional annotated properties
  - element types and families (styles)
CPS style created using support for extensible architecture families
AcmeStudio Tool Framework

- Architectural views for different design concerns created using well-defined styles
AcmeStudio Tool Framework

- Core language extended to define view-to-base relations as *Acme maps* in Multi-View Editor
AcmeStudio Tool Framework

- Core language extended to define view-to-base relations as *Acme maps* in Multi-View Editor
Next Steps

- Model transformation from domain-specific tools and frameworks
- Taxonomy of system errors detectable by view consistency and completeness checks
- Semi-automated view relations: graph transformation/rewrite rules?
- Capture dynamic behavioral assumptions in predicates by extending to temporal logic formulas
Your Views?