Modeling Challenges for CPS Systems

Software Engineering for Smart Cyber-Physical Systems

David Garlan

Carnegie Mellon University
Pittsburgh, PA, USA

May 17, 2015

Acknowledgements

- Joint work with faculty
  - Bruce Krogh (Electrical Engineering)
  - Andre Platzer (Computer Science)
  - Bradley Schmerl (Software Engineering)
  - Javier Camara (Software Engineering
- … and students
  - Ajinkya Bhave (multi-view synthesis)
  - Akshay Rajhans (compositional verification)
  - Ivan Rutchkin (architecture and tools)
  - Roykrong Sukkerd (task automation)
- With funding/support from
  - National Science Foundation
  - Bosch Corporation
  - Toyota Corporation

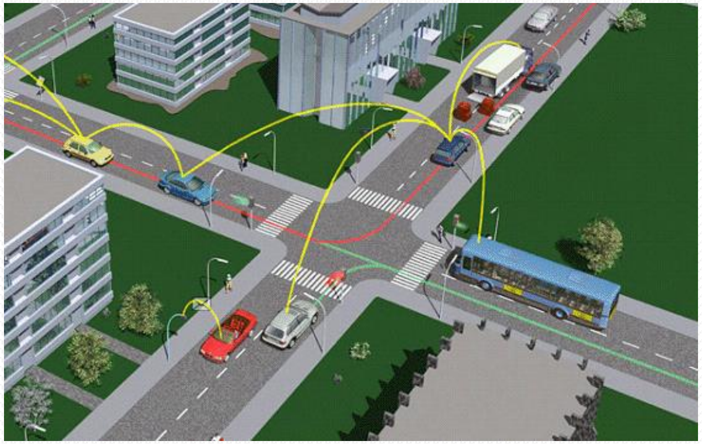
This Talk – Three Themes

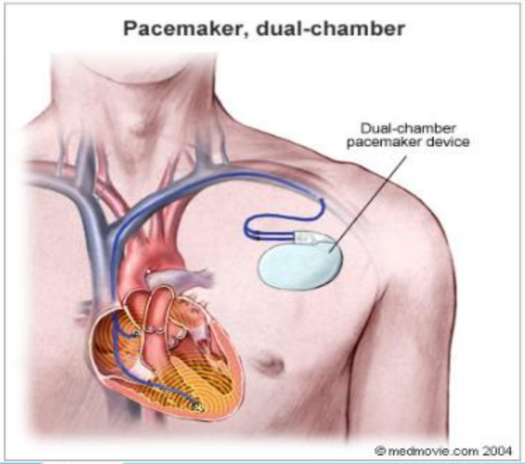
- Theme 1: CPS is challenging in fundamental ways
  - Heterogeneity
  - Complexity
  - Uncertainty
- Theme 2: SE can help … but with modifications
  - Model-driven engineering
  - Architecture (and abstraction in general)
  - Tools
- Theme 3: But SE needs more to make it "smart"
  - Dealing with uncertainty
  - Important special case: human-in-the-loop systems

Outline

- Characteristics of cyber-physical systems and the role of models
- Today’s model-based CPS methods have many problems
  - Difficult to make trade-offs and ensure consistency/completeness
  - Difficult to integrate the different modeling approaches
  - Difficult to integrate humans “in the loop”
- Approach:
  - Unified representation through extensions of software architecture and using architectural views to support heterogeneous modeling and analysis
  - Tools for dependency analysis and coordination
  - Stochastic multi-player games
- Various examples along the way
  - Quad-rotors, Smart highways, Real-time systems, Smart homes

Cyber-Physical Systems





What is a Cyber-Physical System?

- Many of today's systems involve complex combinations of software and physical elements
- Examples:
  - Energy-efficient buildings (heating, cooling, power, …)
  - Smart electric grid
  - Transportation: automotive control, rail control, air traffic control
  - Security systems
  - Smart homes
- These are hard to design and implement
  - Requires expertise from many domains, including control systems, networking, software applications, etc.
  - Often difficult to analyze and test

Problems

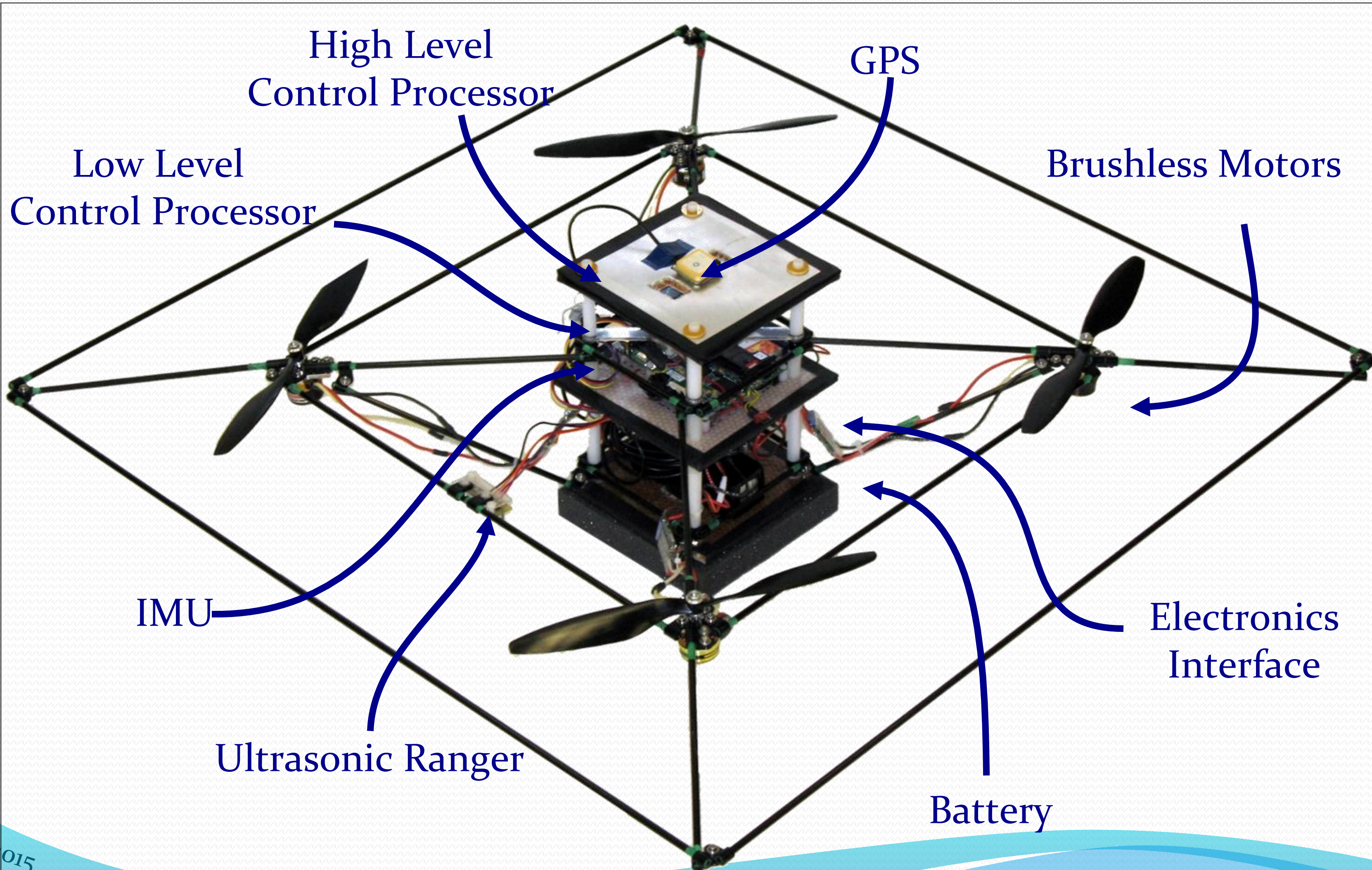
Today's approaches to designing cyber-physical systems (CPS)

- Inherantly multi-discplinary
- Requires a variety of formalisms and methods :
  - physical dynamics
  - control law development
  - hardware platform
  - software architecture

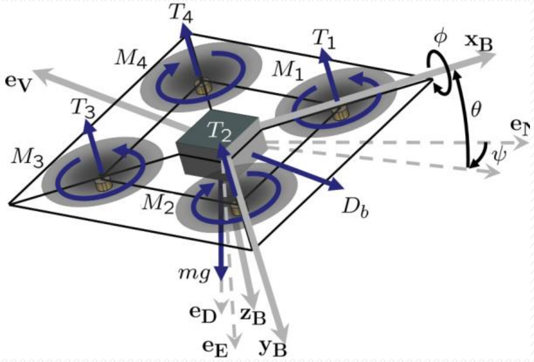
- Problem 1: Making tradeoffs across different engineering dimensions and domains
- Problem 2: Completeness and consistency of models
- Problem 3: Performing whole-system analyses
- Problem 4: Accounting for human behavior

Example CPS: STARMAC

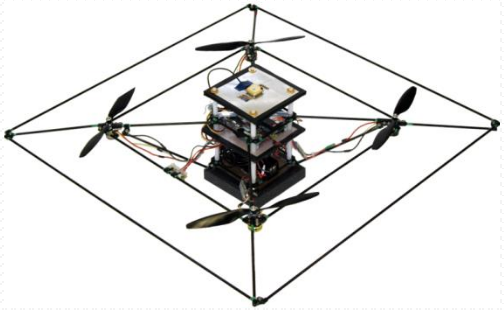
- Stanford Testbed for Autonomous Rotorcraft for Multi-Agent Control (http://hybrid.eecs.berkeley.edu/starmac/)
- Four rotors, arranged symmetrically on frame


High Level
Control Processor
GPS
Low Level
Control Processor
Brushless Motors
IMU
Electronics
Interface
Ultrasonic Ranger
Battery

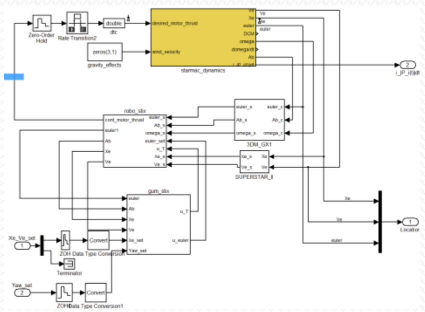
Multiple Models



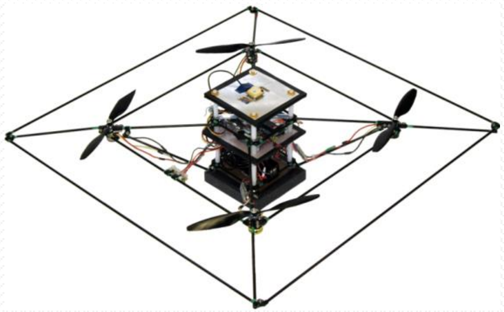
Physical Model

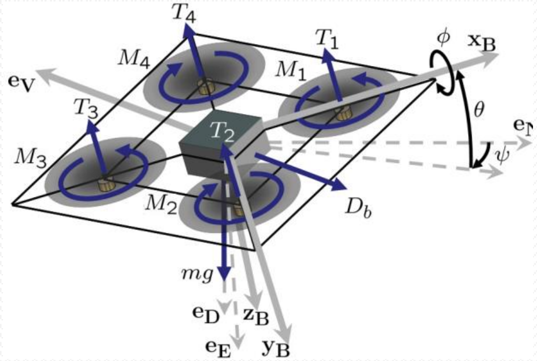


Multiple Models



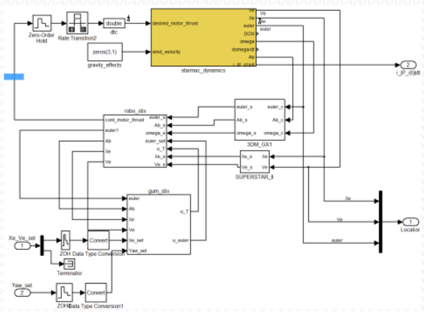
Control Model



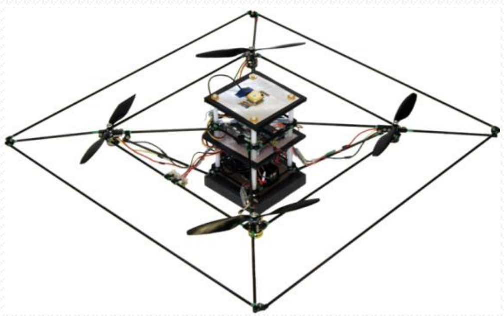


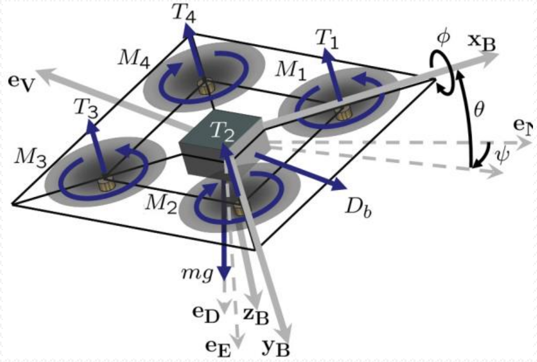
Physical Model

Multiple Models

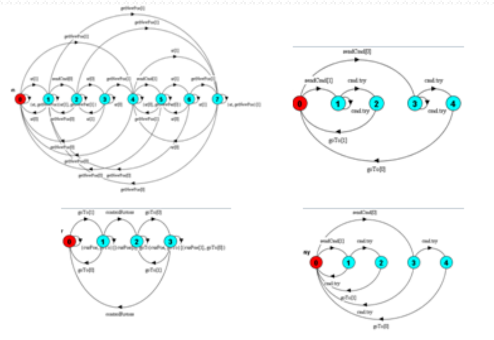


Control Model



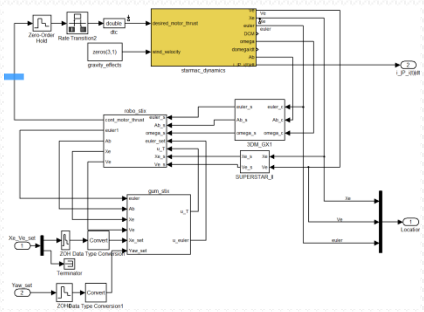


Physical Model

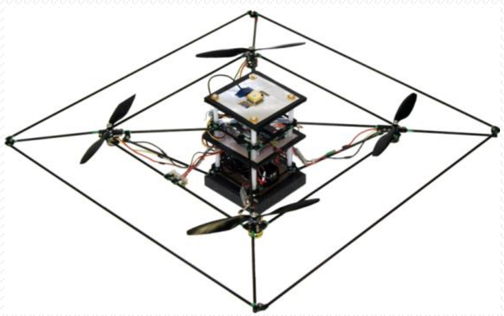


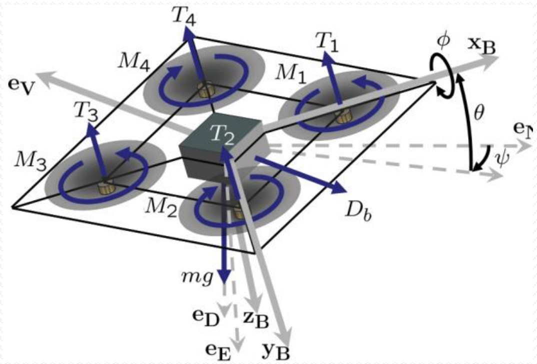
Software Model

Multiple Models

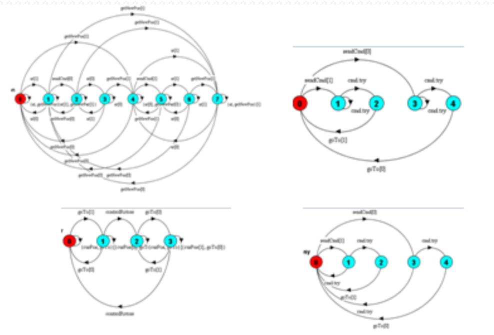


Control Model

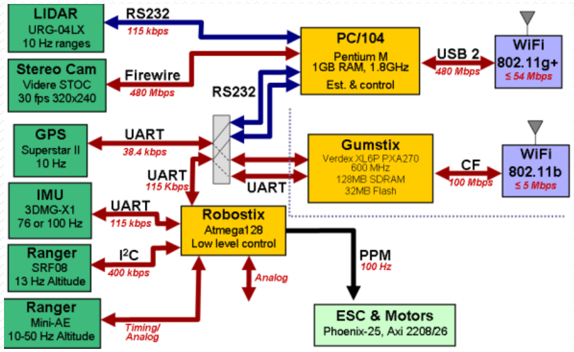




Physical Model

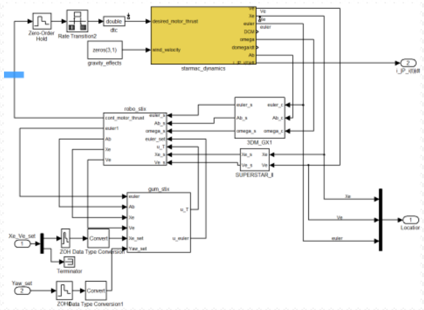


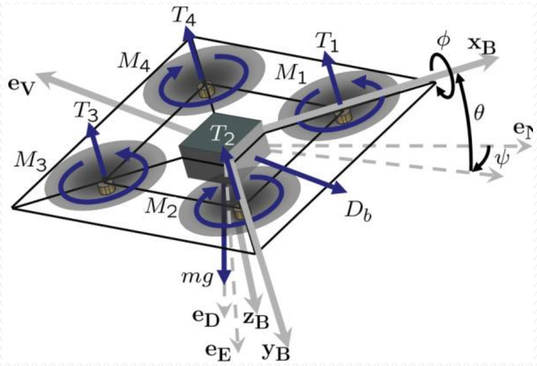
Software Model

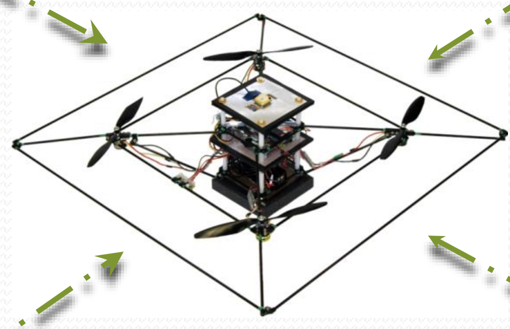


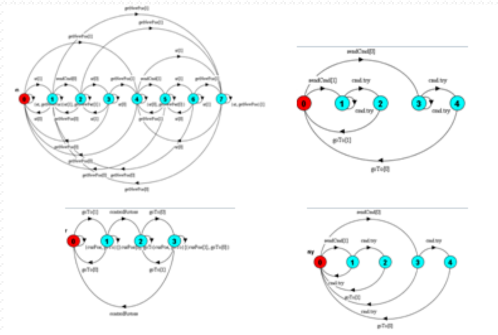
Hardware Model

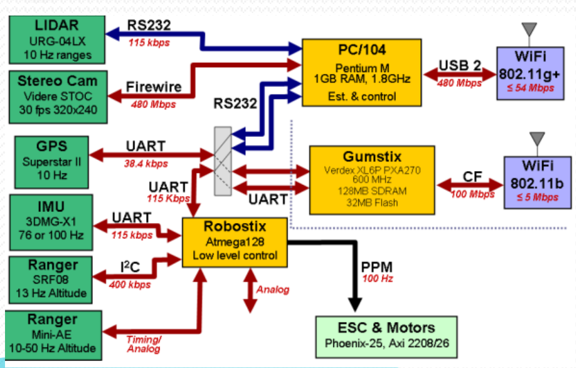
Do they represent the system?



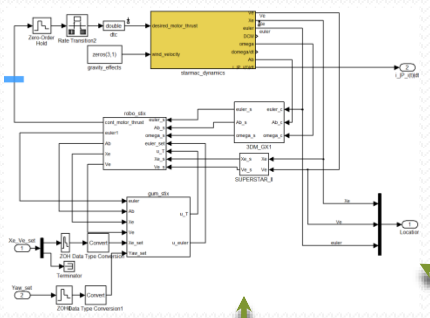


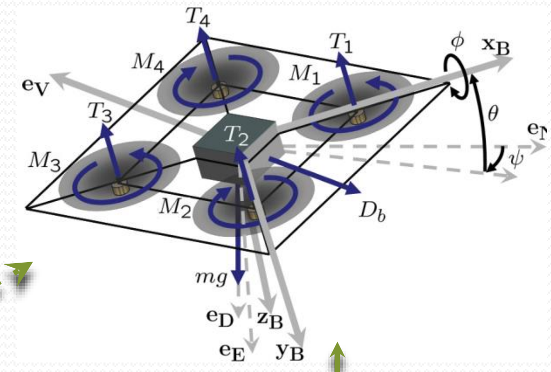




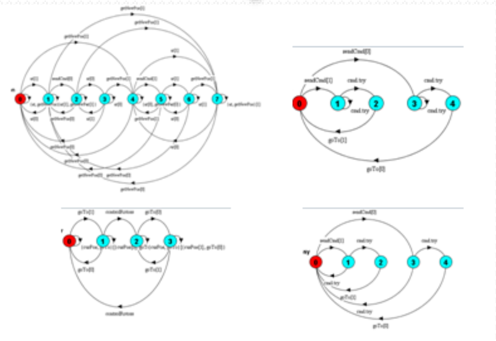


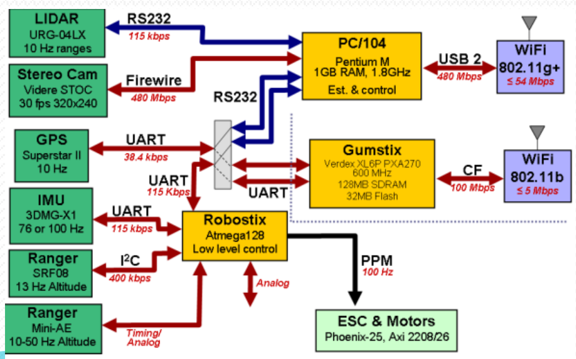
Are the views consistent?



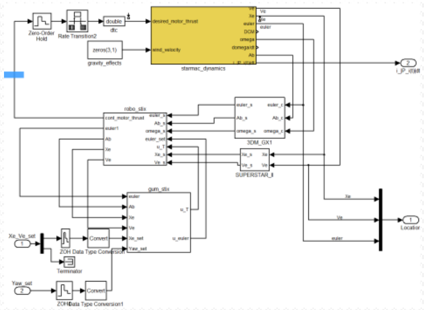


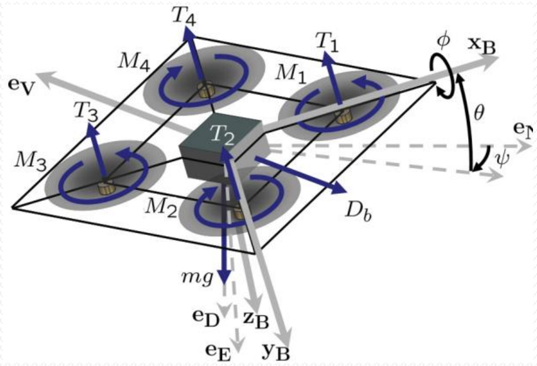
?



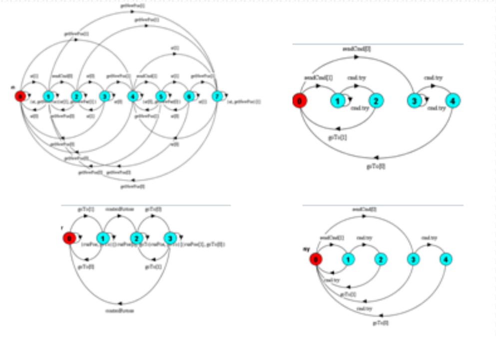


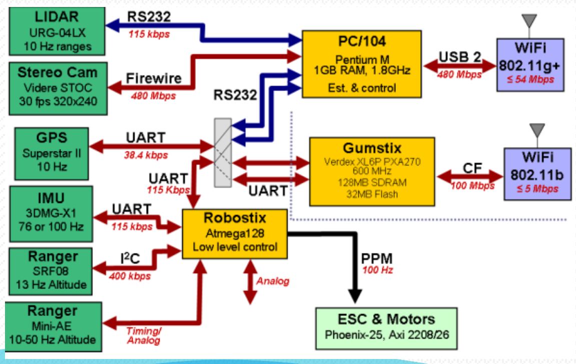
Is there a unifying representation?





?





What we would like

- An approach that unifies both cyber and physical design
  - Allows one to describe the complete system
  - Supports tradeoff analysis
- But allows a multiplicity of models and analyses
  - Detects inconsistencies and mismatched assumptions
  - Can reason about completeness of design models
- Supported by tools
  - Allowing automated checking and linkage to legacy analysis tools

Approach (work in progress)

1. Extend software architecture to support both physical and cyber elements through a CPS architectural style
2. Support heterogeneous models and analyses through views
3. Determine consistency criteria for multiple views
4. Support development through extensions to software architecture modeling tools

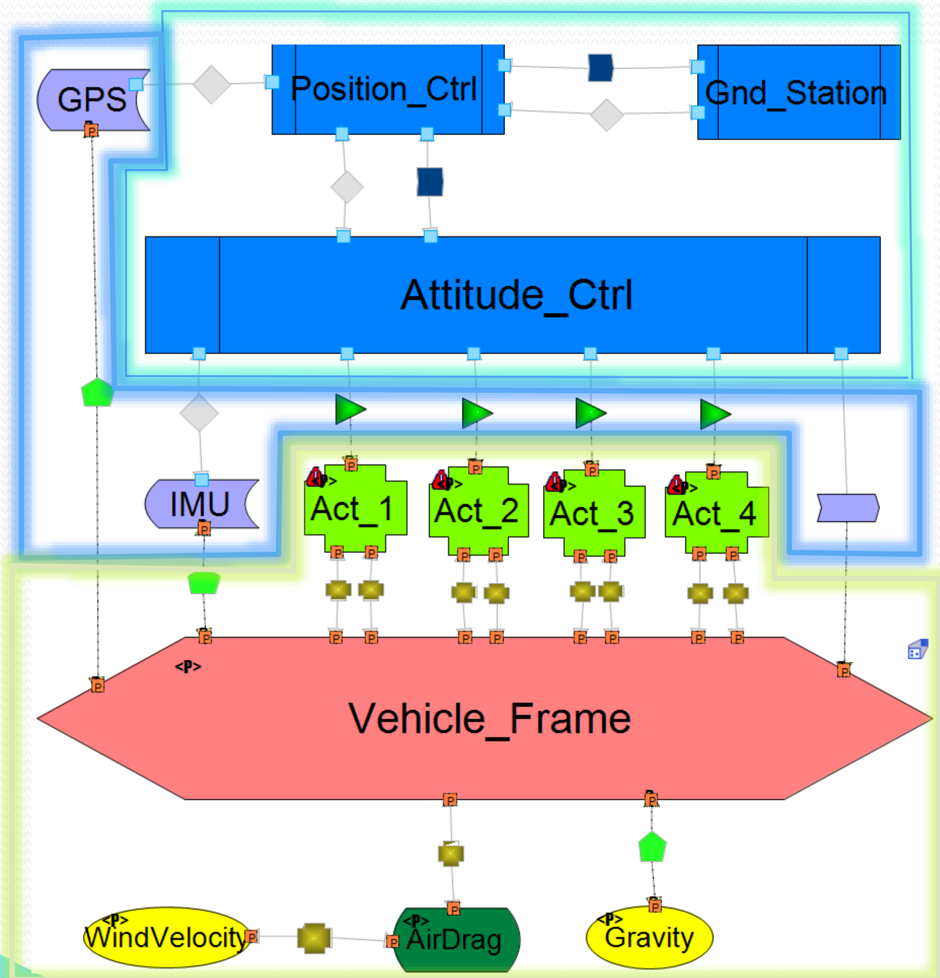
Software Architecture

- Models a system as a graph of components and connectors
  - Components: computational elements (databases, servers, etc)
  - Connectors: communication pathways (RMI, http, etc)
  - Properties: abstract behavior of elements (expected load, latencies, transaction rates)
- Benefits of software architecture
  - Abstraction reduces complexity
  - Supports design-time analysis and tradeoffs
- However, does not usually consider physical modeling, beyond simple sensors and actuators

Extended with Physical Elements

- Include physical system as a set of interacting components with shared variables/coupled constraints
  - Components: Physical elements (mechanical, electrical, thermal, environmental,…)
  - Connectors: Physical interactions (conservation laws, energy flows, …)
  - Behavior: Dynamic behavior of elements (DAEs, LHA, …)
- Bridging elements link physical elements to cyber elements

Quadrotor (base) Architectural Model



Cyber elements

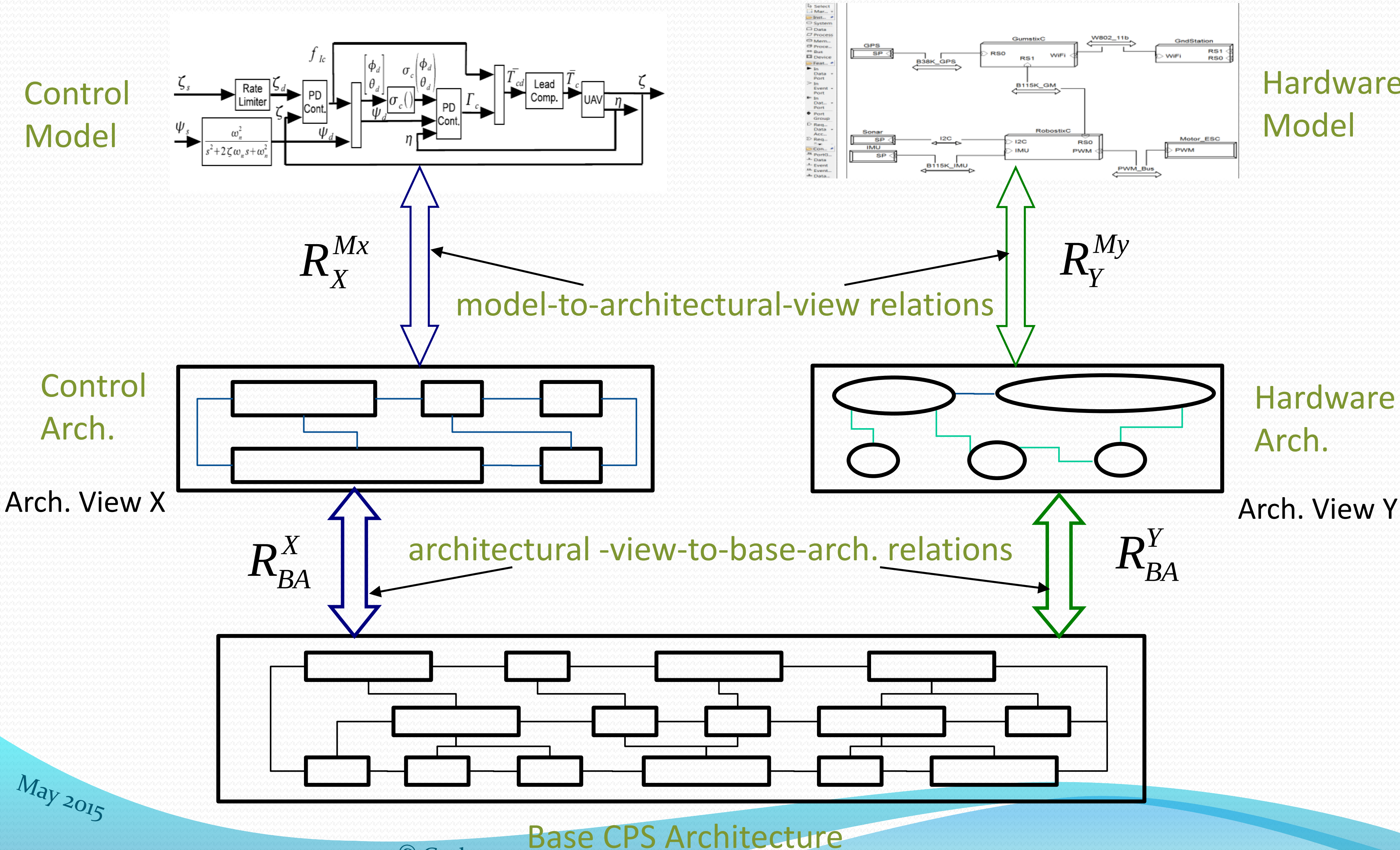
Bridging elements

Physical elements

Behavioral Modeling

- Behaviors are associated with subsets of the architecture suitable for analysis
  - Ex 1: Simulink model focuses on control performance, abstracts scheduling and communication jitter in software.
  - Ex 2: Software behavior modeling focuses on commun-ication between position ground station and position controller, abstracts away physical aspects.
- Leads to need for *multiple models*
  - Tailored to particular behavior/analysis
  - Related via the base architectural model **through views**

Models as Architectural Views



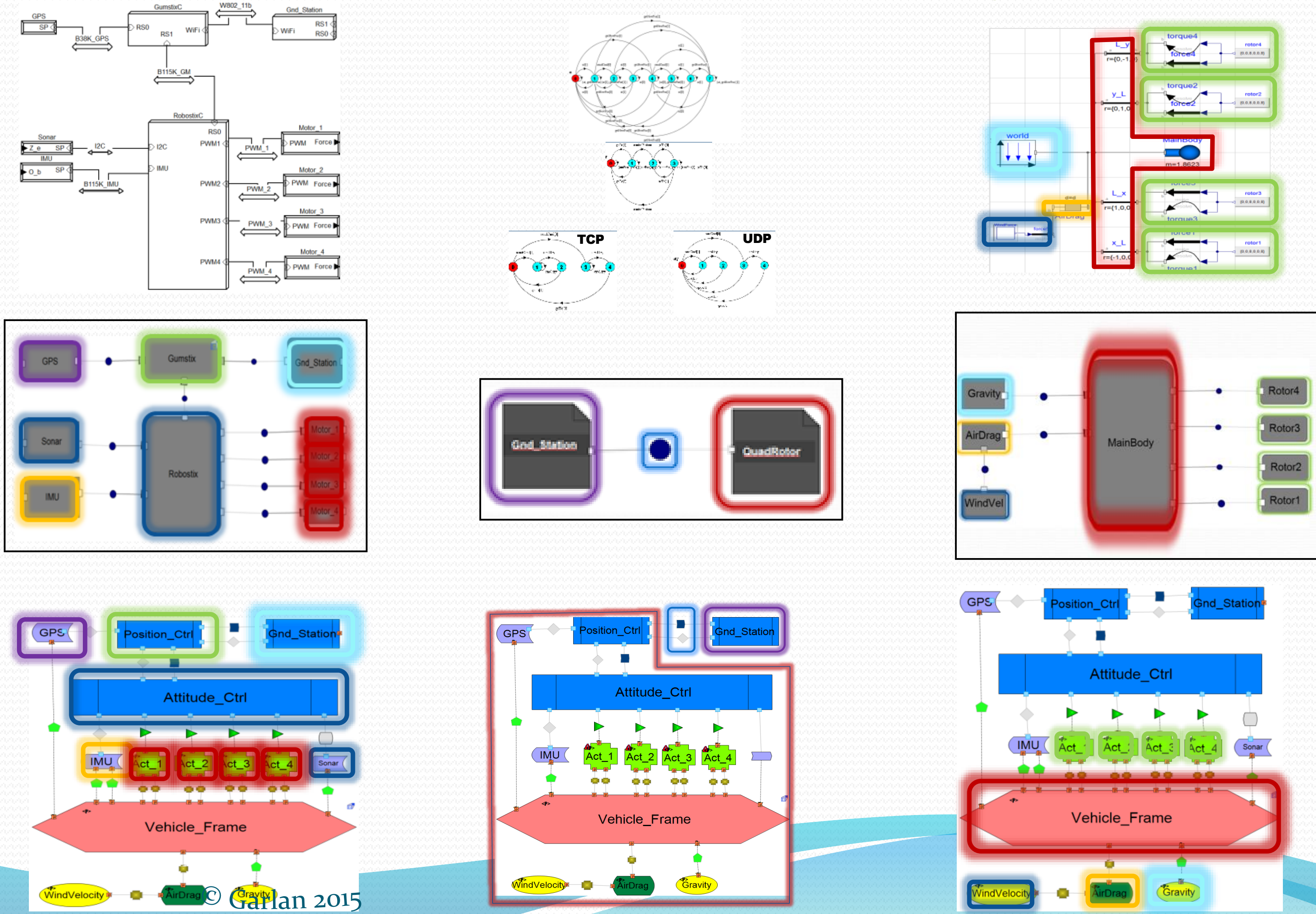
STARMAC Architectural Views

Hardware (AADL) | Software (FSP) | Physical (Modelica)

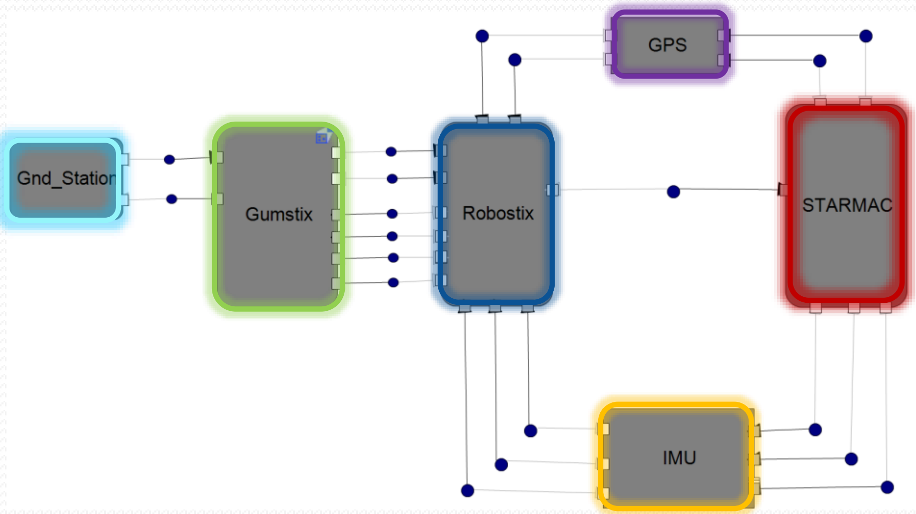
Model

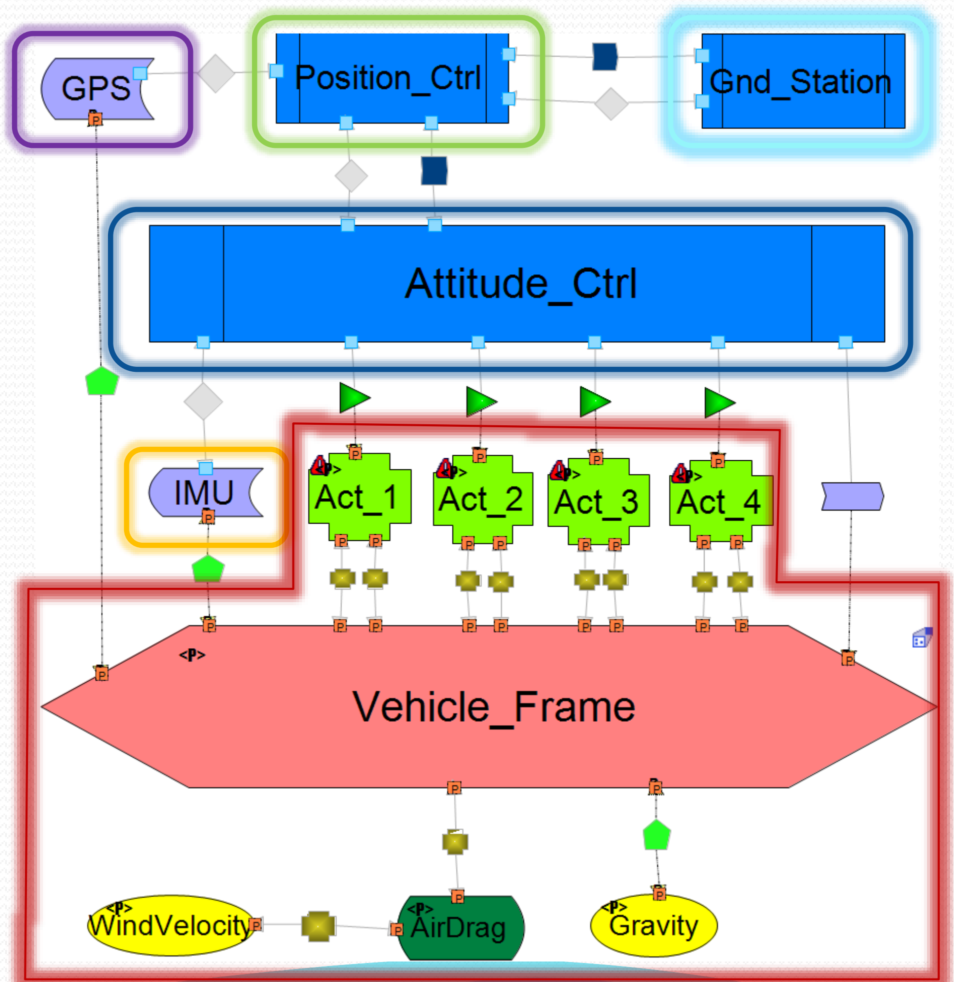
Arch. View

Base Arch.

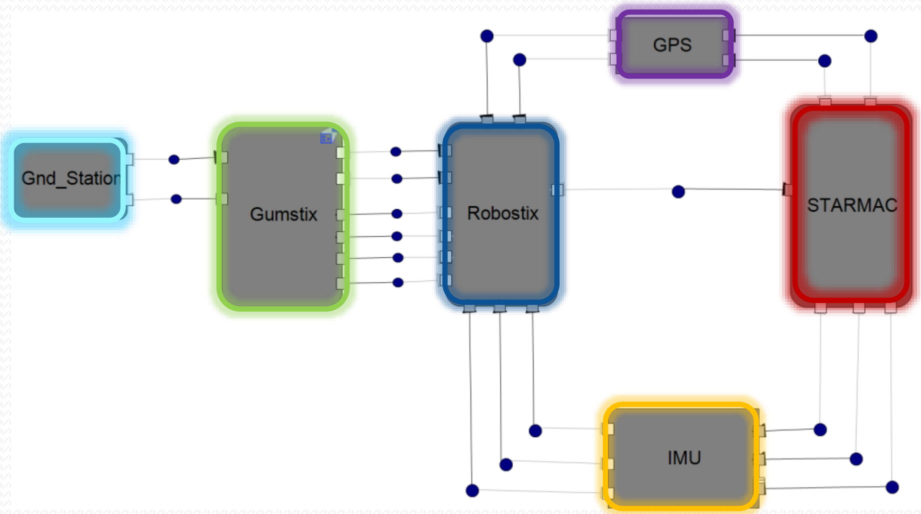


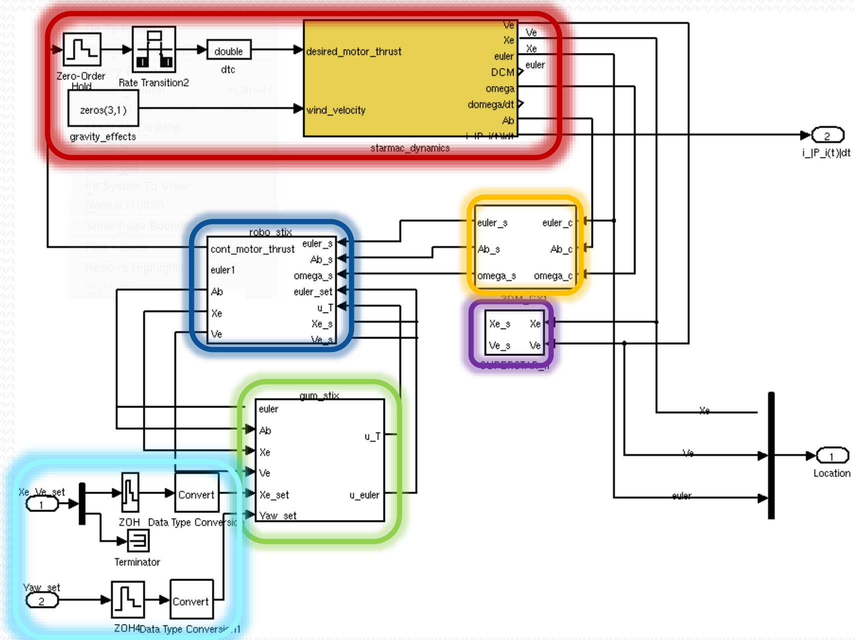
Simulink Architecture View



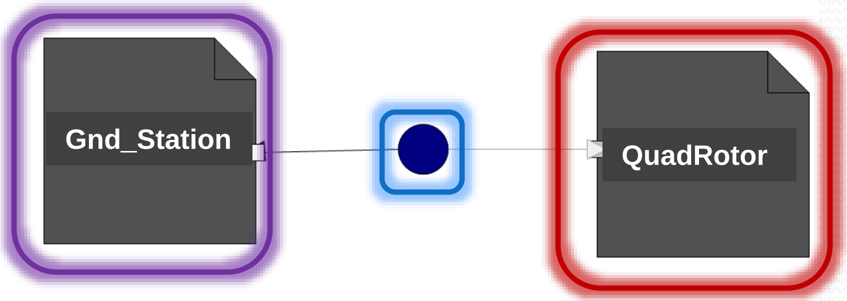


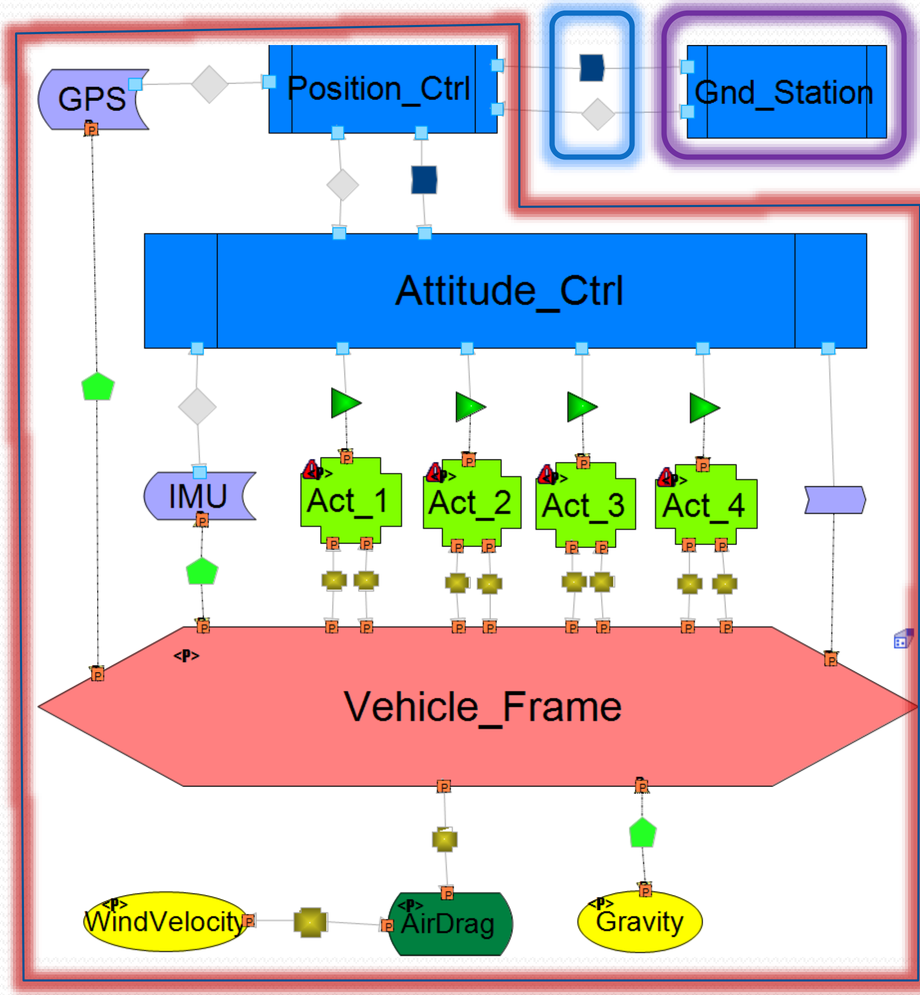
Simulink Model



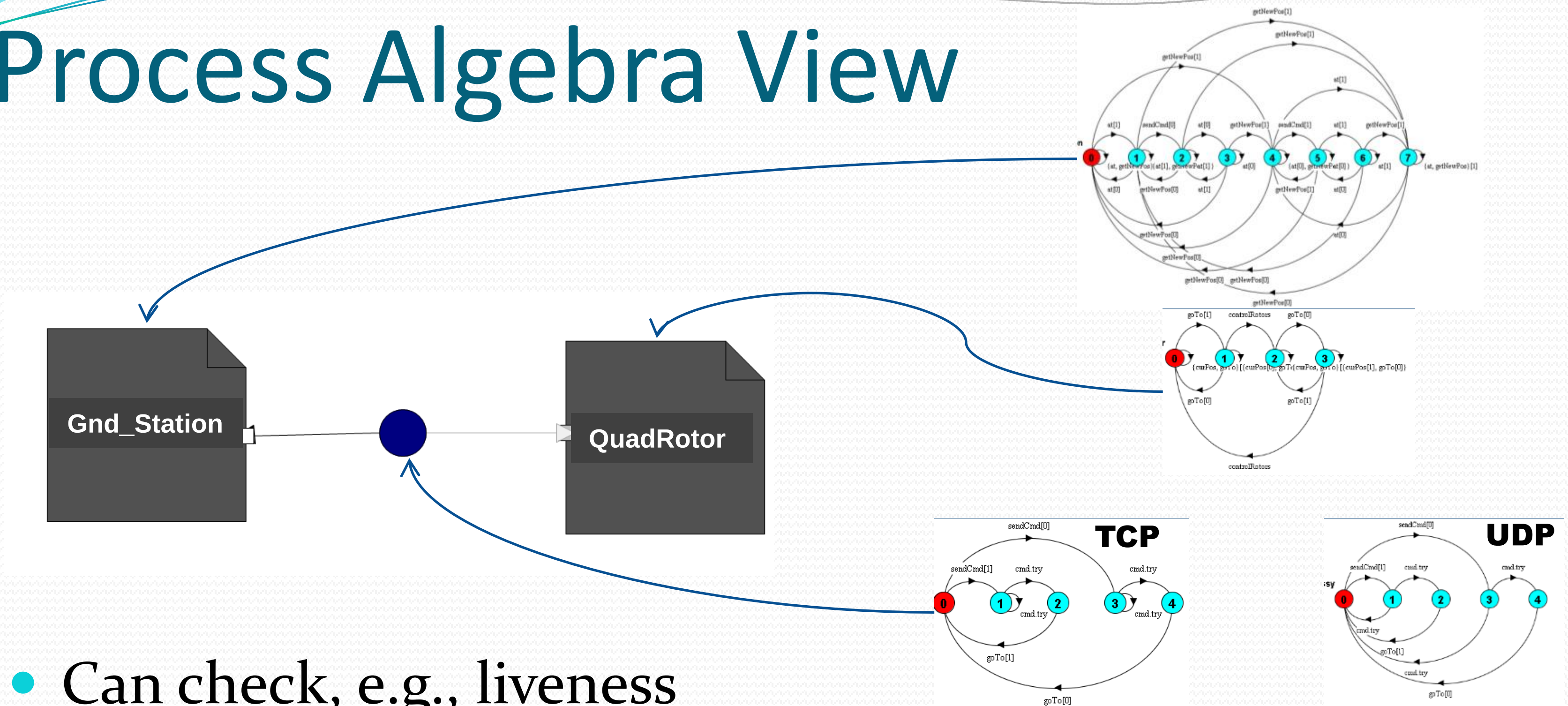


FSP Architecture View





Process Algebra View

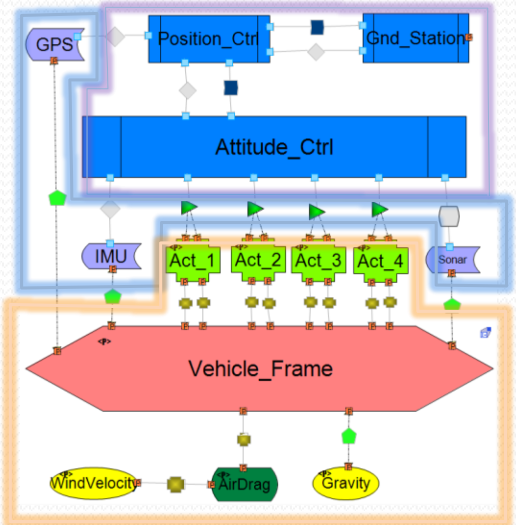


- Can check, e.g., liveness
  - If user tells ground station to move rotor to location A, ground station will eventually receive a status message from the position controller that it is at new location
  - Allows us to reason about connection over lossy, wireless network
    - Retry (TCP) connector allows liveness property to be satisfied

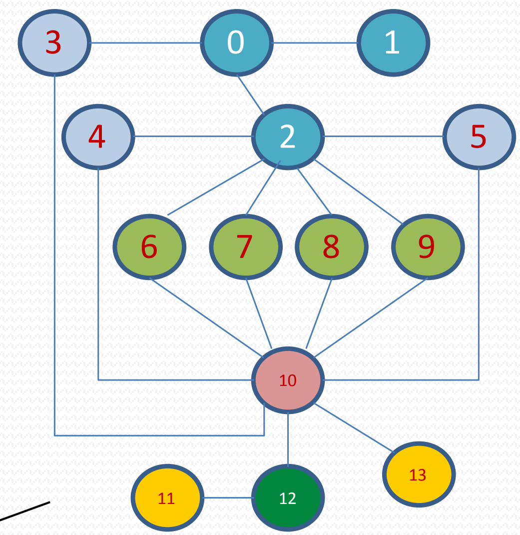
What about Consistency?

- *Structural consistency* between the base architecture and a view
  - Determines if a view represents a valid abstraction of the base architecture
  - Weak: All elements of a view must be derived (via encapsulation) from the base architecture
  - Special case is communication integrity: Two components in a view cannot interact unless they can also interact in the base architecture
  - Strong: Every component in the base architecture is accounted for in the view (possibly within an encapsulation boundary)

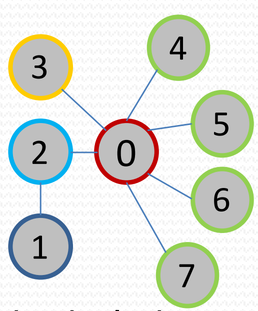
Graph Analysis for View Consistency



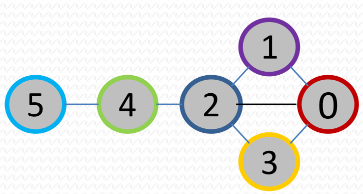
generation
of component
connectivity graph



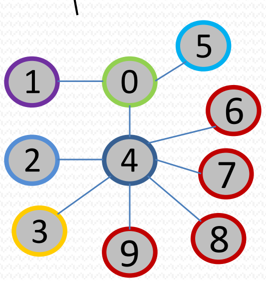
Consistency of views analyzed as graph morphisms[1]



Physical View



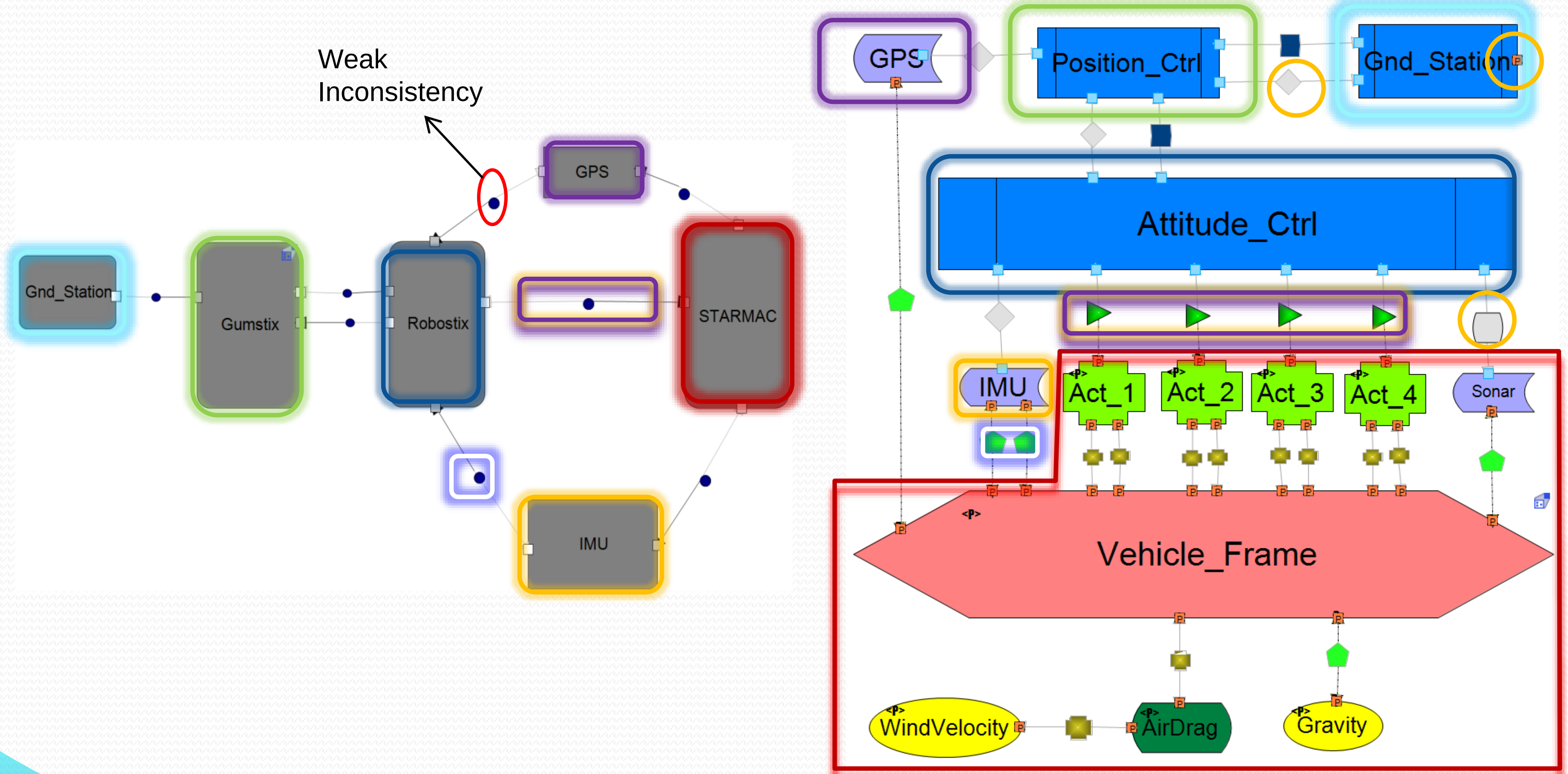
Simulink View



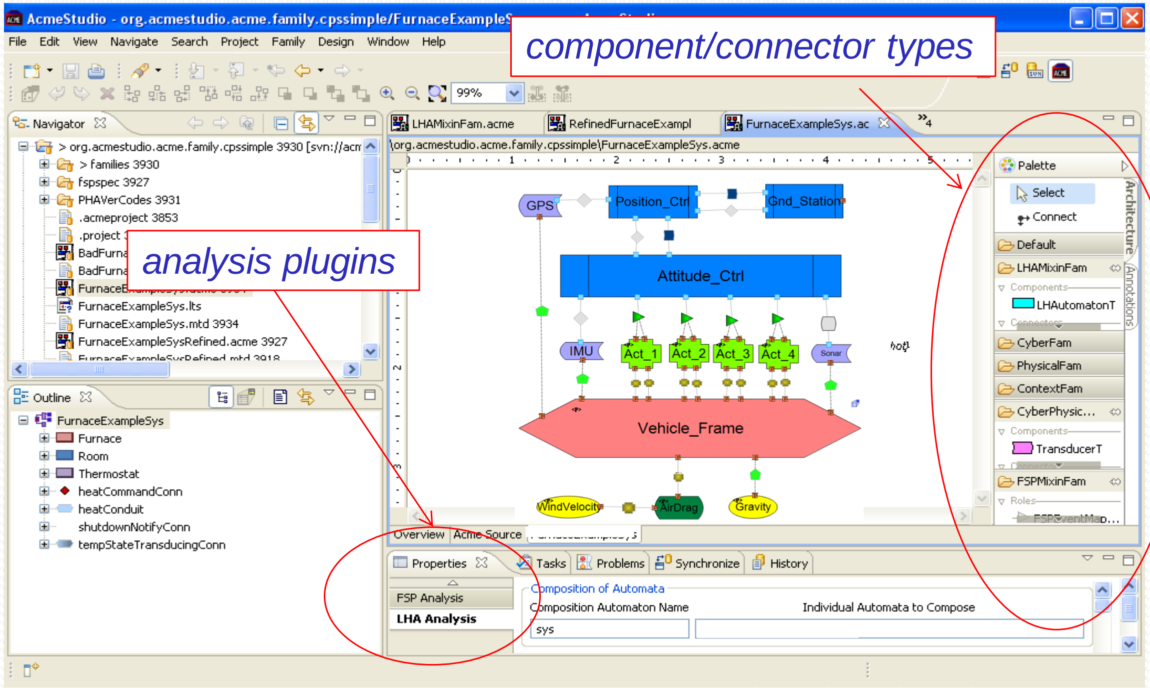
Hardware View

[1]VFLib Graph Matching Library: http://amalfi.dis.unina.it/graph/db/vflib-2.0/doc/vflib.html

Structural Inconsistency in STARMAC



Tools: AcmeStudio

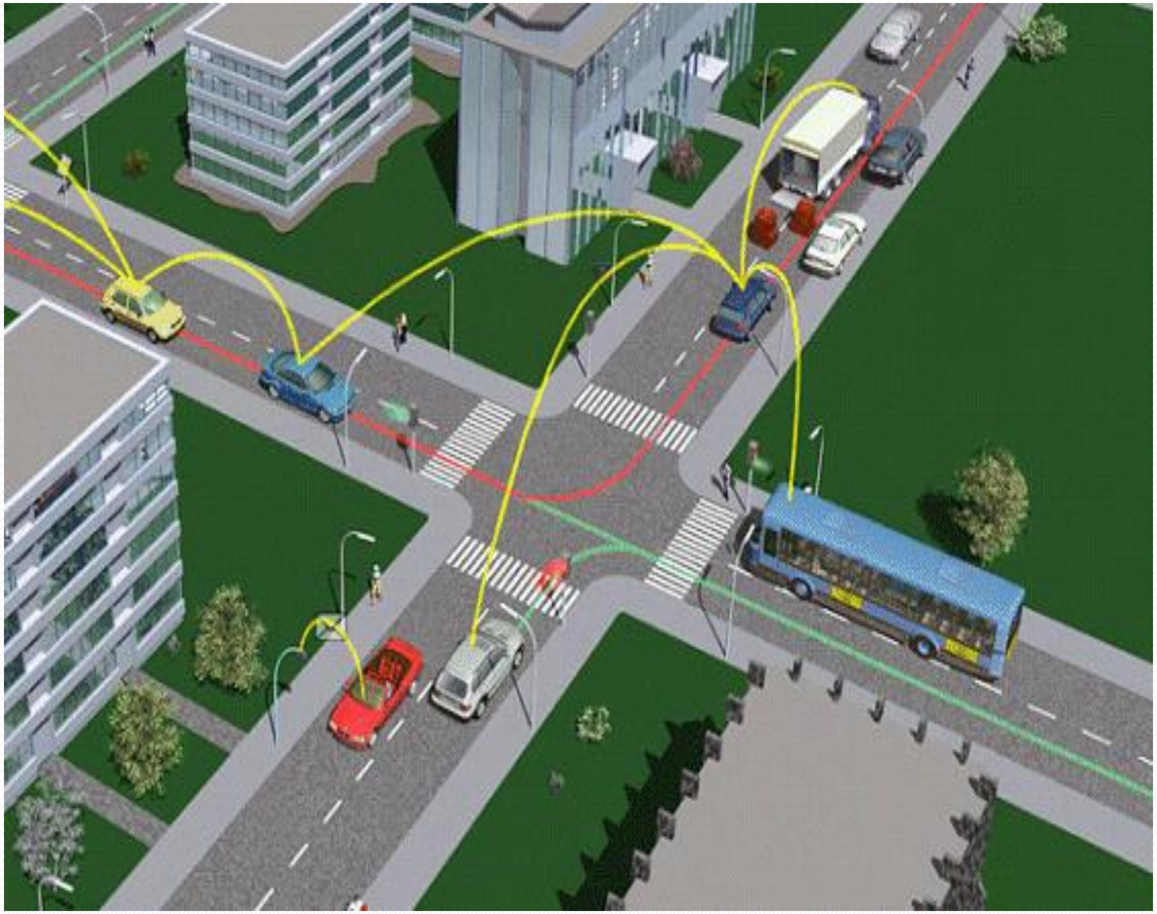


- Extensible framework for architecture design and analysis
- Adaptation to CPS:
  - support for associations between architectural views
  - augmenting views with semantic attributes and analysis
  - analysis plug-in for system-level verification

Semantic Consistency

- Each view and associated analyses guarantees certain properties
  - By analyzing properties represented in the view
  - By generating the values of other properties – e.g., allocation of processes to processors
- Each view makes assumptions about the parts of system that it is NOT modeling.
  - May assume that certain invariants hold
  - May consume values that other analyses produce
- How can we represent and check these?

Case Study: CICAS*

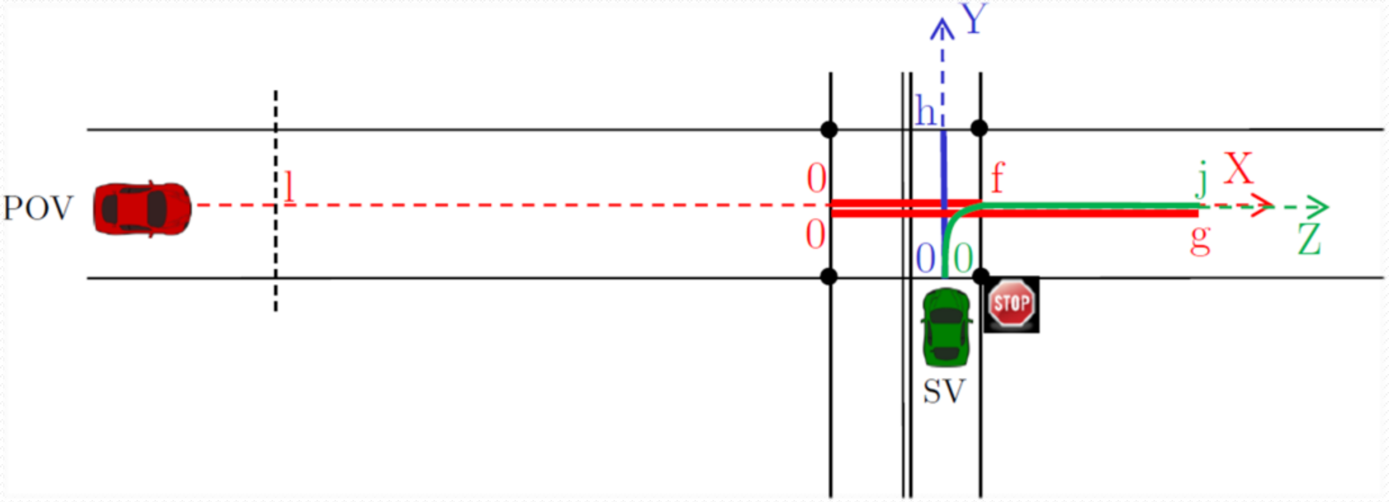


* Cooperative Intersection Collision Avoidance Systems

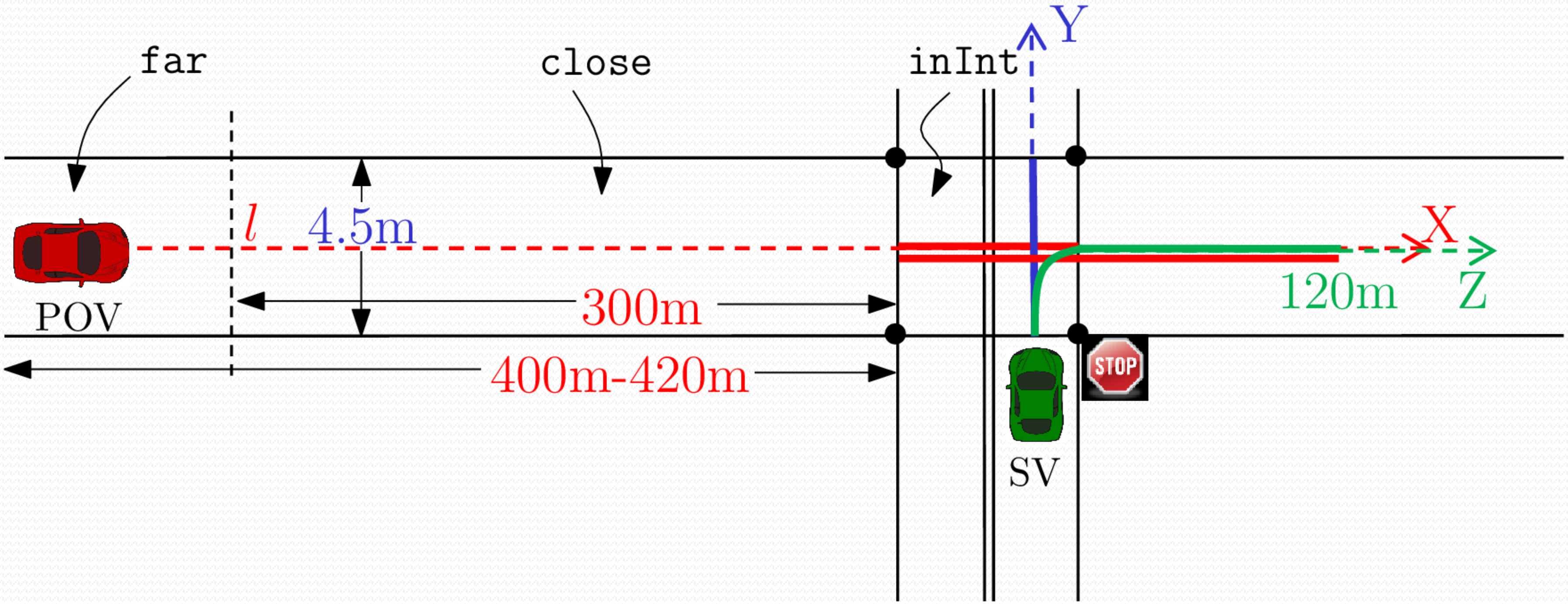
*http://www.its.dot.gov/cicas/

CICAS Sub-problem

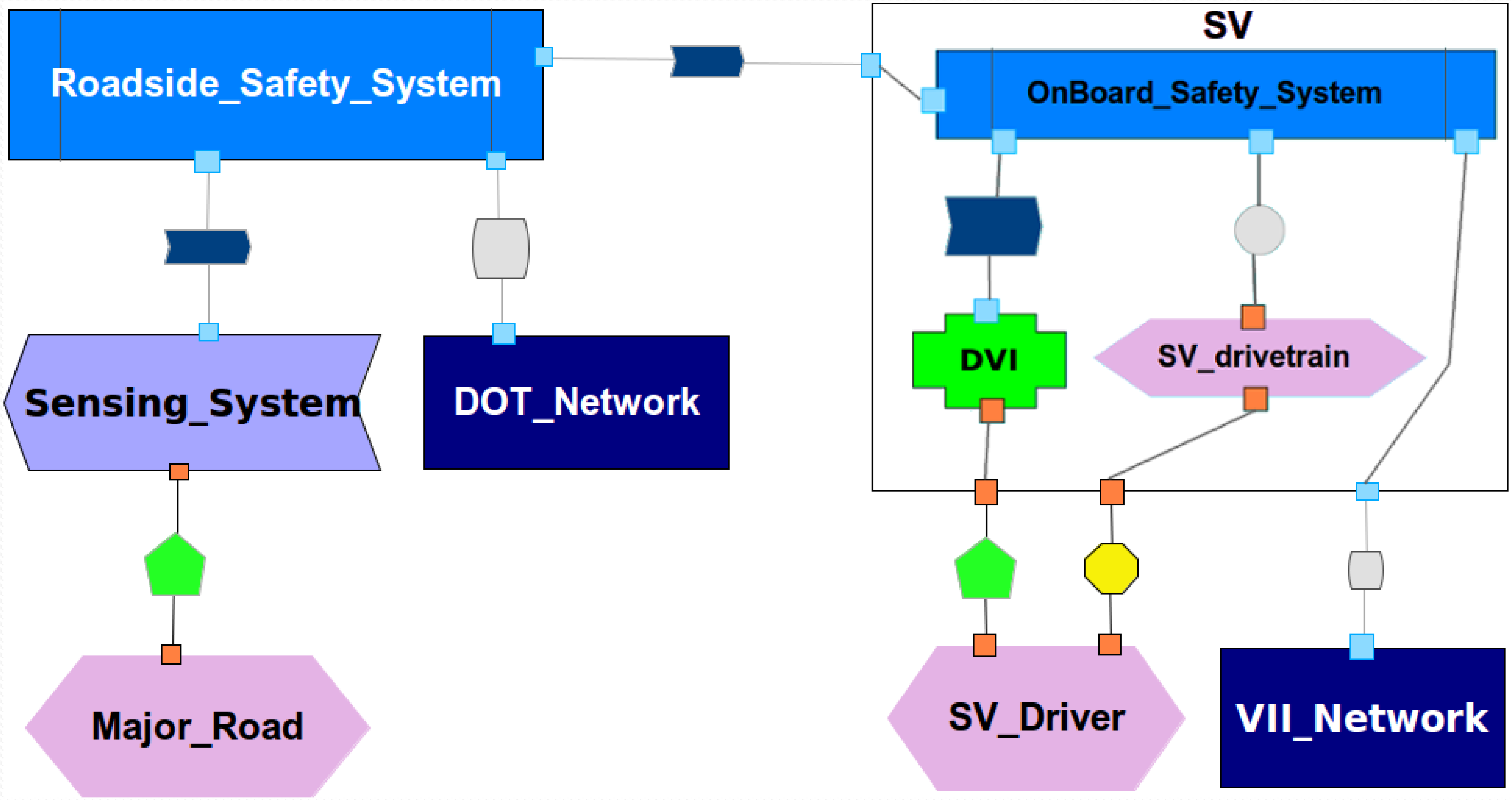
- Stop Sign Assist
  - Decide if it is safe to enter an intersection.
- Research:
  - Combining structural and semantic reasoning.



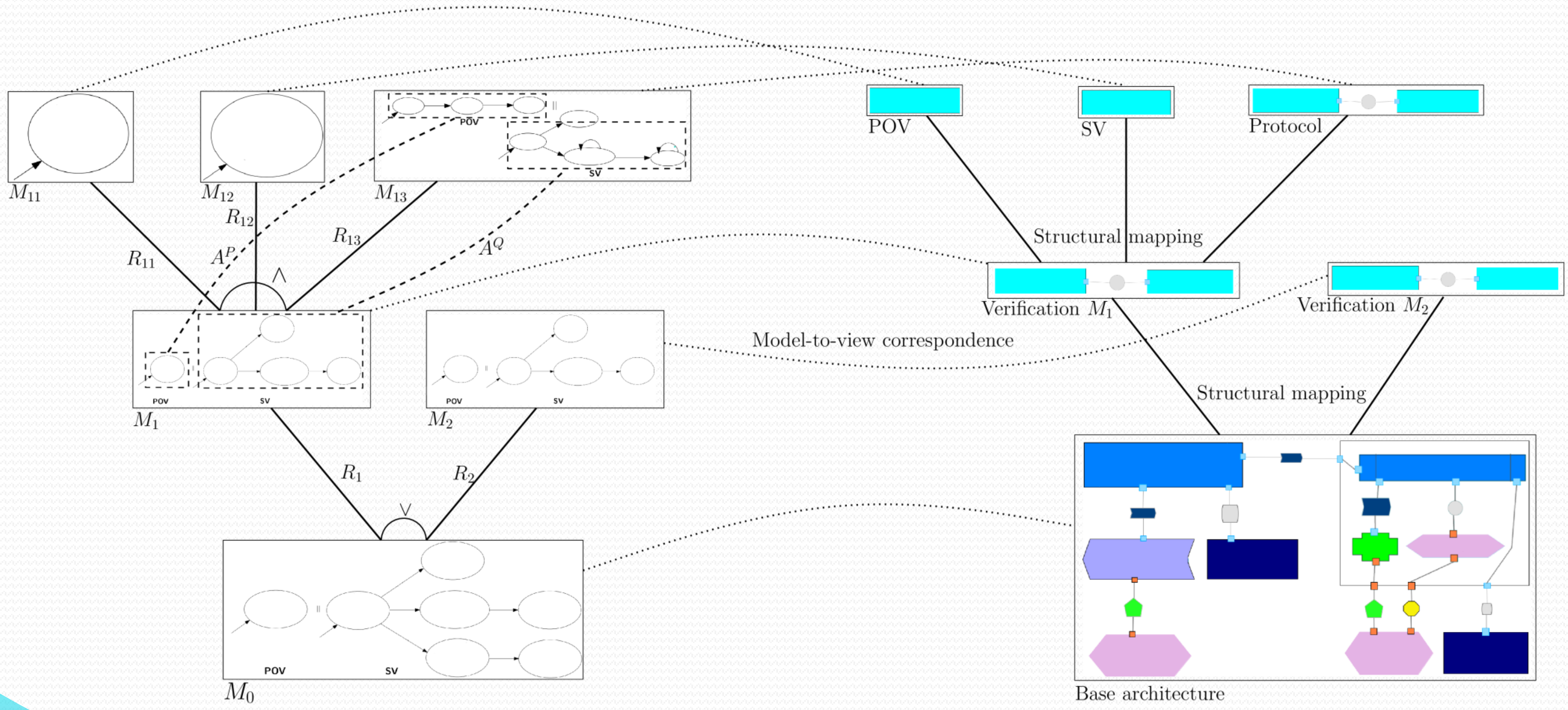
CICAS-SSA



CICAS base architecture



Semantic & structural hierarchies

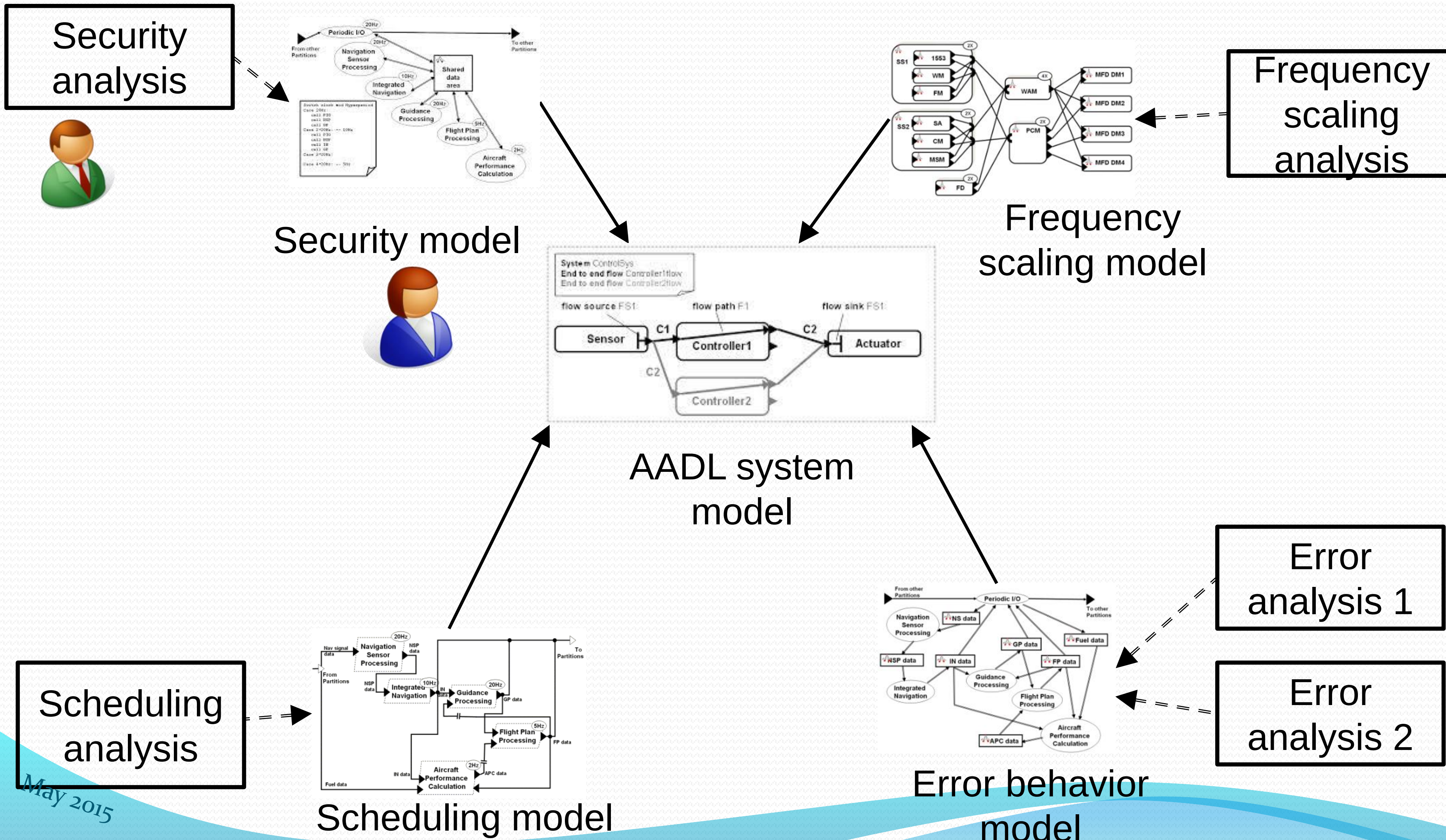


Maintaining Semantic Consistency with Heterogeneous Models

- Example: thread scheduling in multi-processor systems.
- Research problems:
  - Understanding dependencies between different views
  - Sequencing CPS analyses.
- Approach
  - Use AADL* models to represent CPS structure/semantics
  - Assume-guarantee reasoning about CPS analyses.
  - Contract verification in multiple logics and domains.

* SAE Architecture Analysis and Design Language
http://www.aadl.info/aadl/currentsite/

Modeling Ecosystem



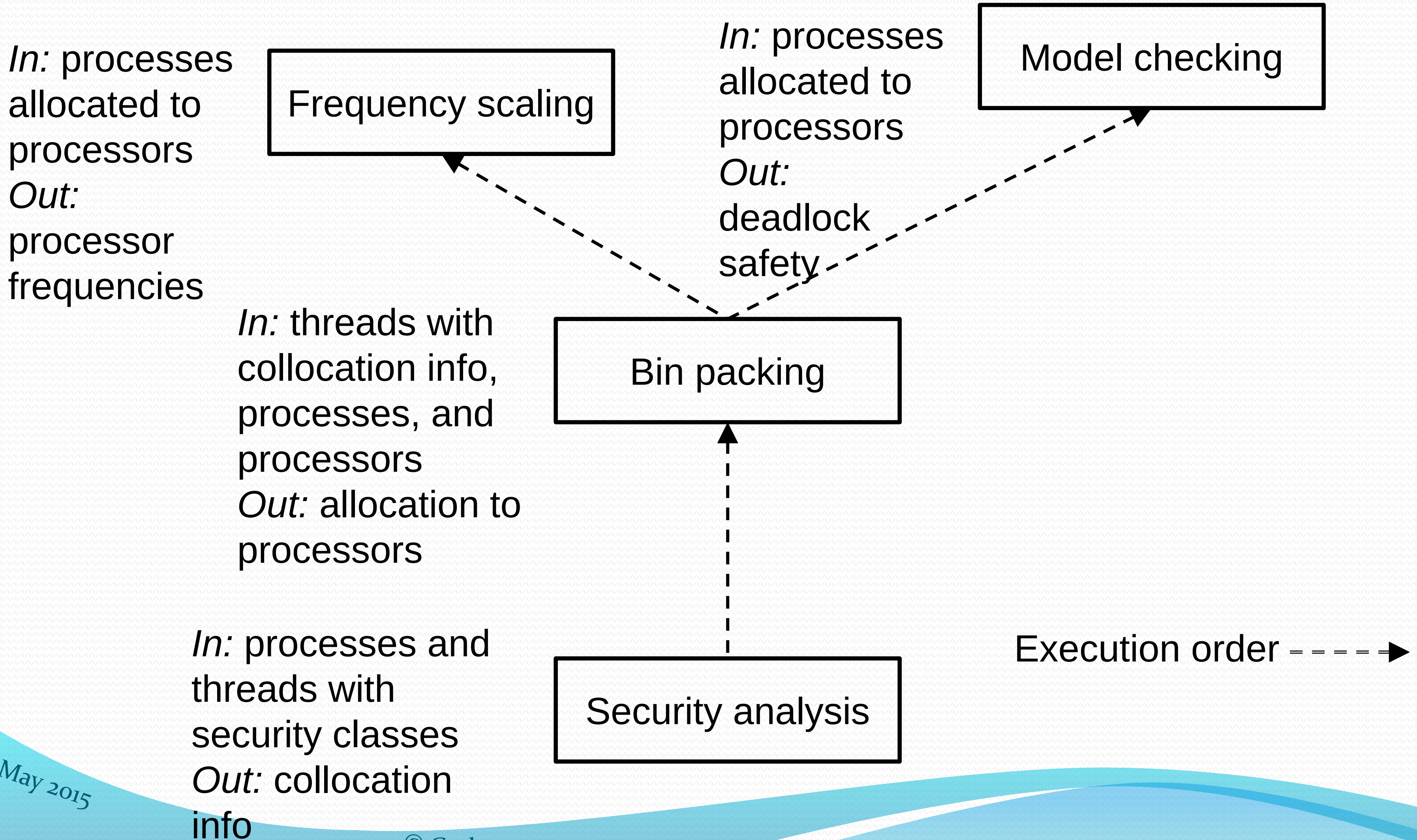
Example of Analyses

- Security (confidentiality) analysis
  - Based on security levels of threads, determine which threads can be collocated on one processor.
- Bin packing (real-time allocation) analysis
  - Allocate processes to processors.
- Frequency scaling (power efficiency) analysis
  - Minimize the processor frequency to meet the task deadlines.
- Model checking (safety) analysis
  - Assuming the threads are scheduled correctly, check if the system is safe.

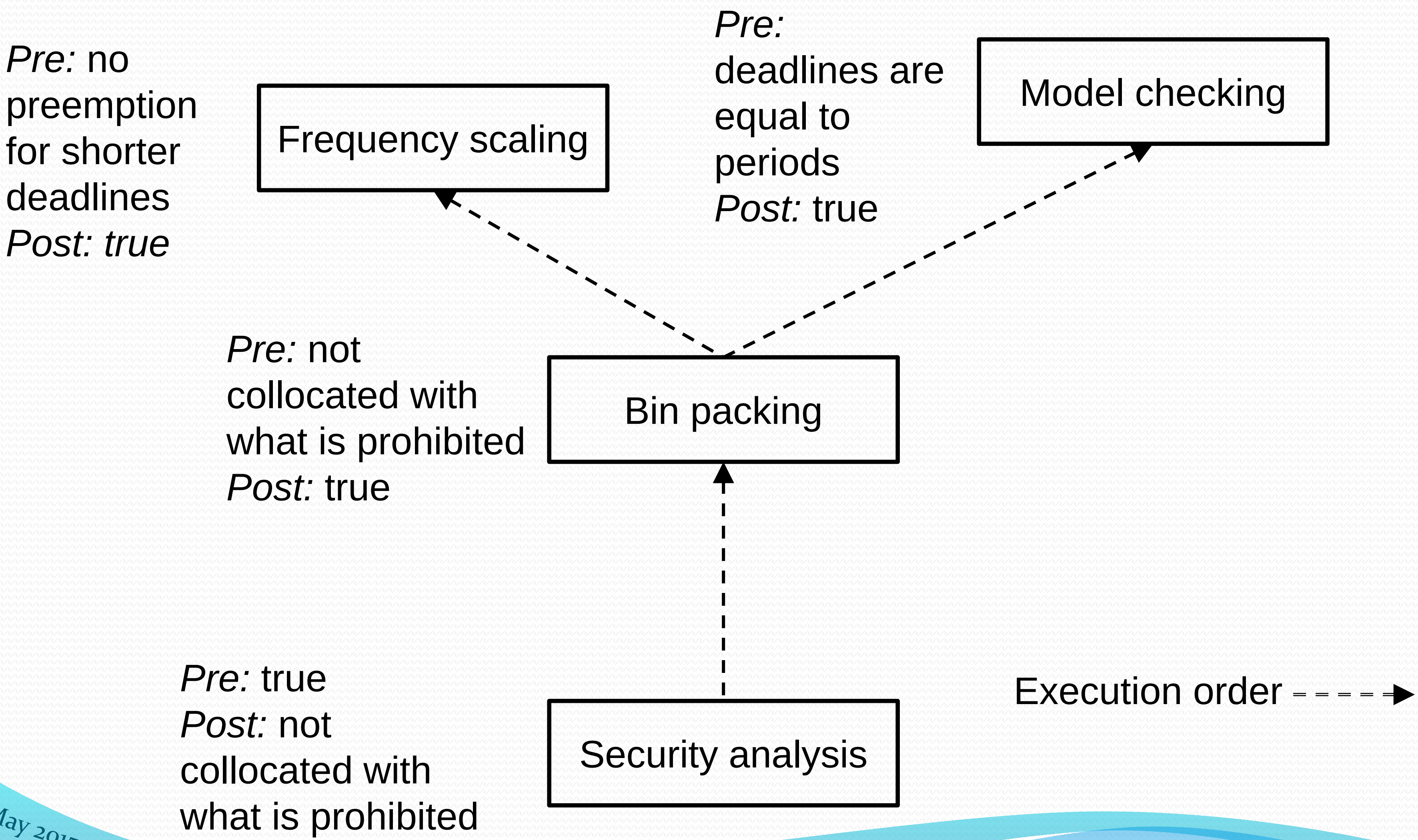
Analysis Composition Problem

- Analyses have semantic interdependencies – how can we be sure we do not violate them?
  - E.g., scheduling needs collocation restrictions
- Analyses rely on each other to work correctly – how to ensure correct composition?
  - E.g., frequency scaling relies on correct scheduling

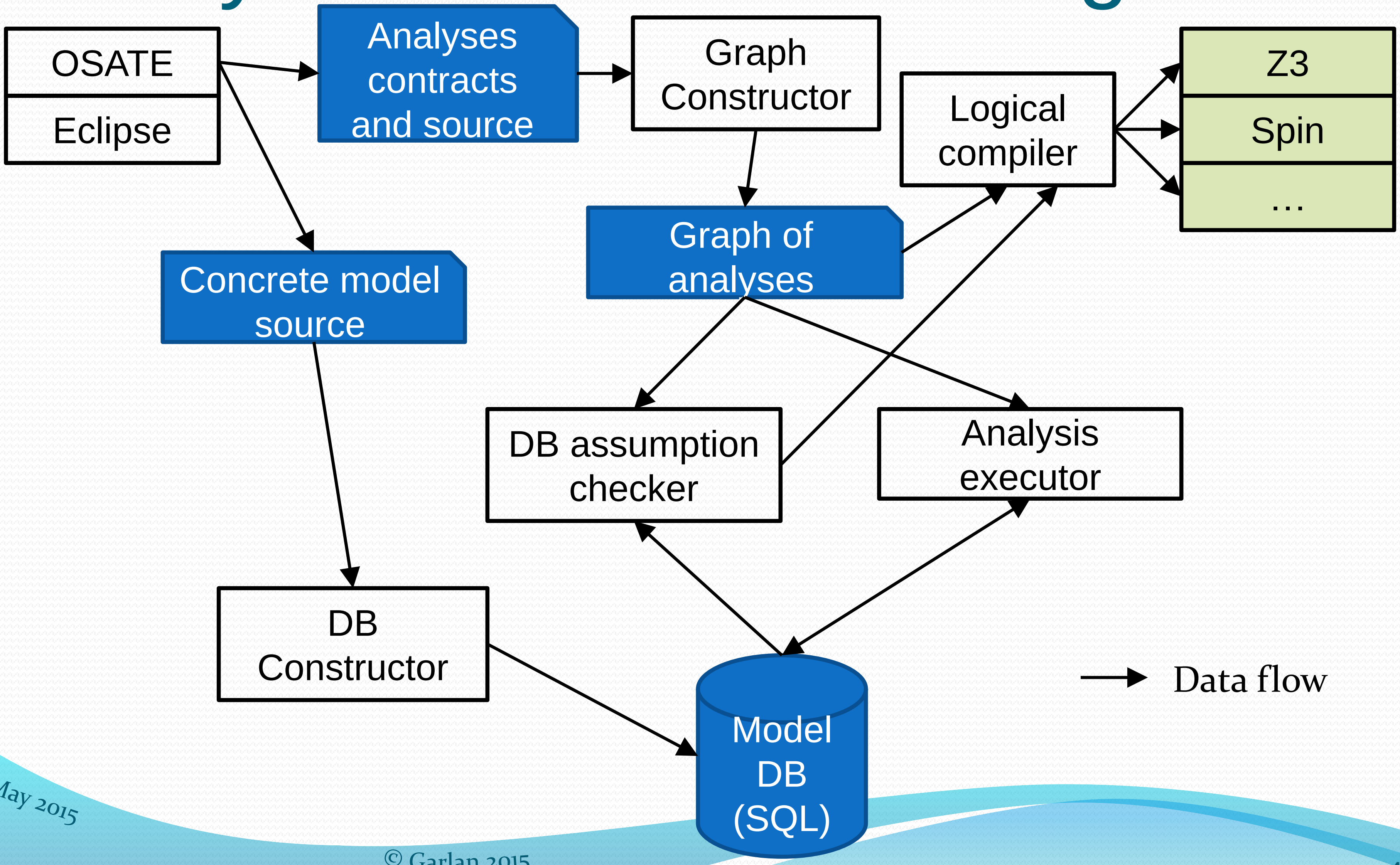
Dependency Graph



Example Analyses: assumptions & guarantees



Analysis Framework Design



Human-in-the-loop

- Many CPSs have humans in the loop
  - Smart homes with occupants
  - Air traffic control operators
  - Automated driving
- Introduces a new problem: how/when to involve humans in the CPS?

Example: Indoor Air Quality Control

Air quality sensors

Air purifier

Occupant

Task:
- Maintain air quality at healthy levels
- Minimize energy consumption

Dehumidifier

Challenges

Air quality sensors

Air purifier

Occupant

Dynamic environment

Uncertainty

Interaction with people

Dehumidifier

Today’s Practice: Rule-based Control

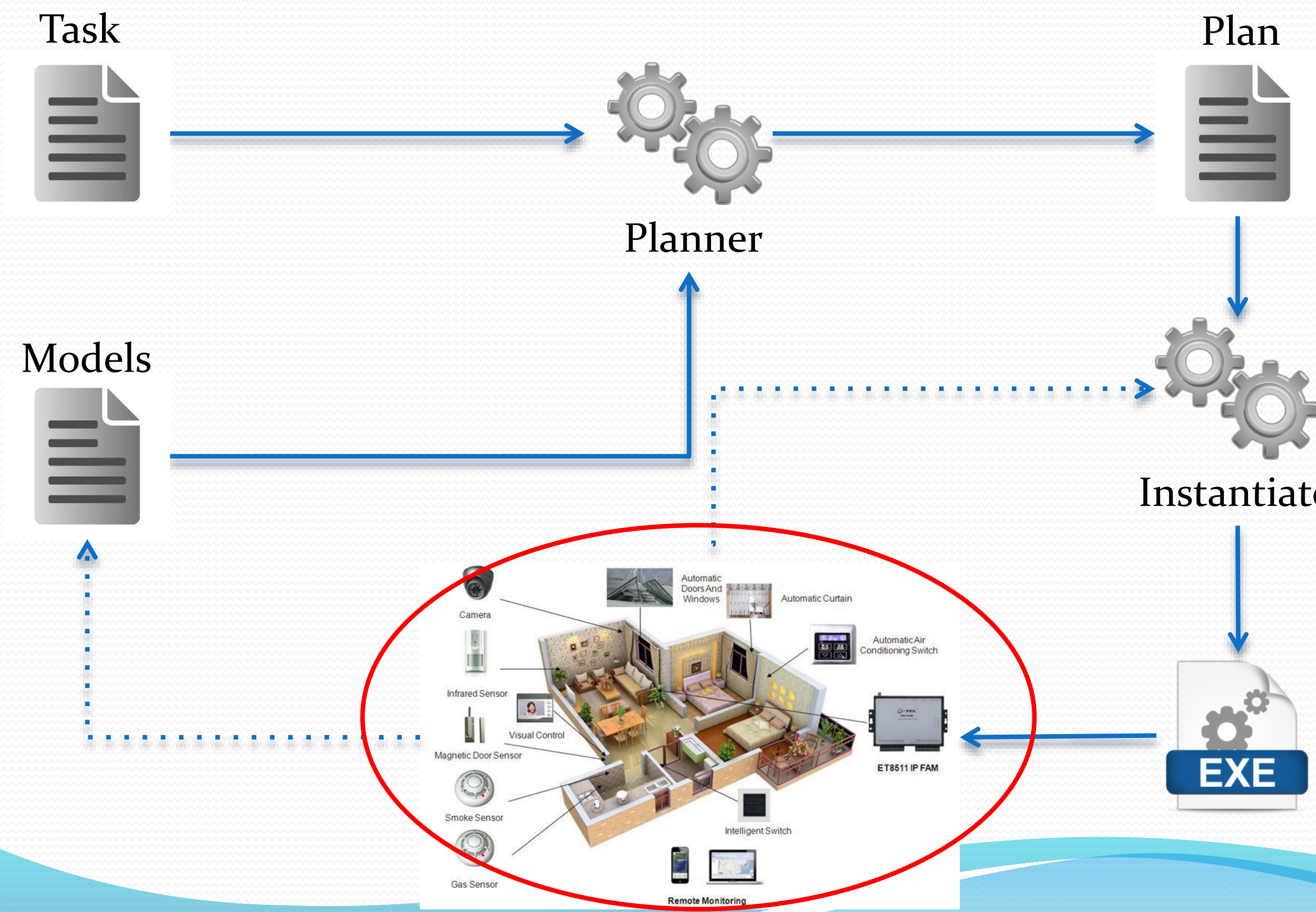
- Based on heuristics
- *Event-Condition-Action* rules

  IF occupants_at_home and PM2.5>12
  THEN turn on air purifier
- Problems
  - Complexity
  - Determining if all conditions are accounted for
  - Managing conflicts
  - Reasoning about properties and qualities of tasks

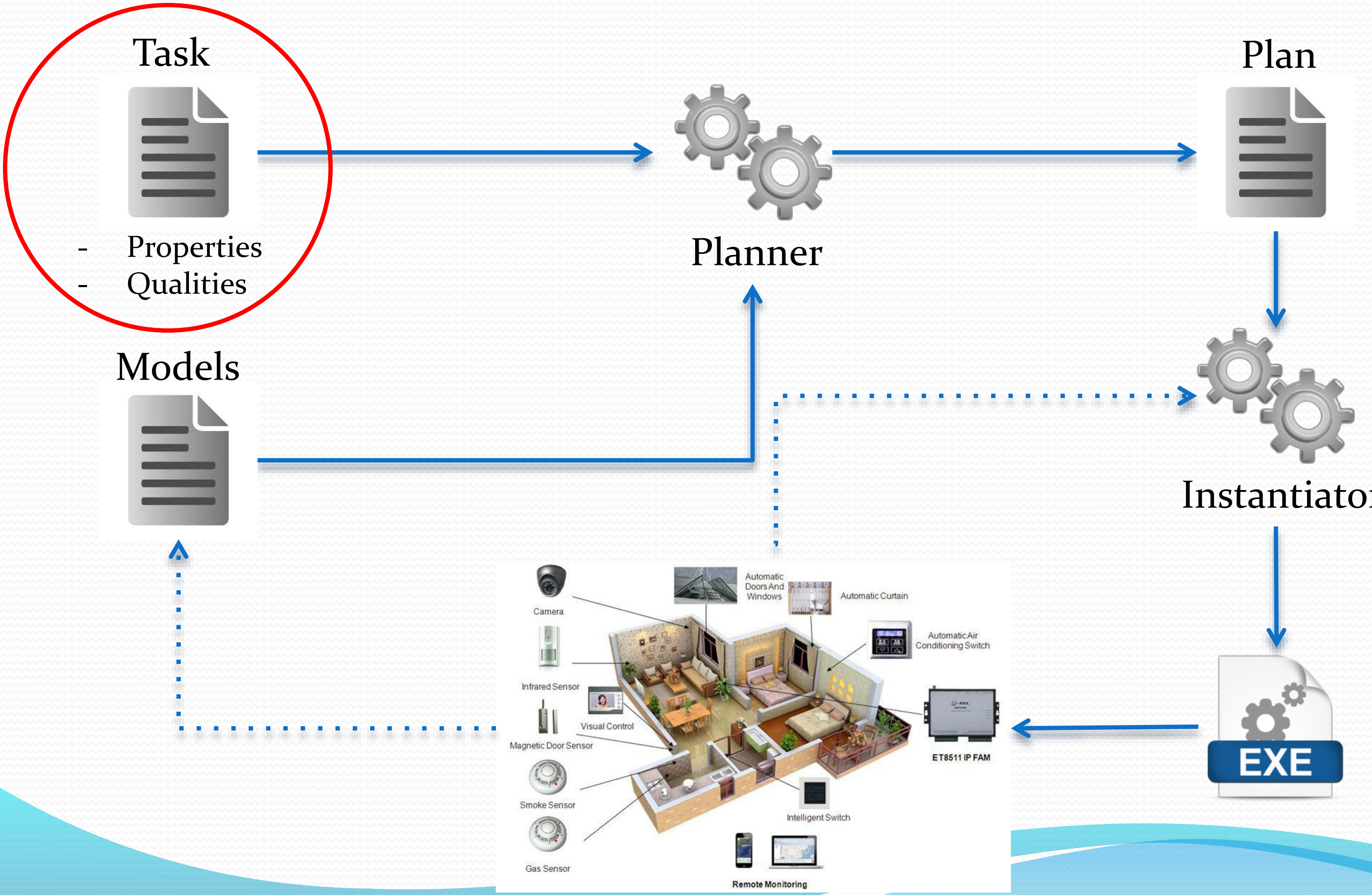
Approach: Automated Planning

- Key idea: Given a set of *models* and a *property specification*, automatically generate a plan
- Benefits:
  - No programming – task management is automatically generated
  - Models are simpler (and more reusable) than code
  - Tools can provide formal guarantees about properties and qualities of tasks

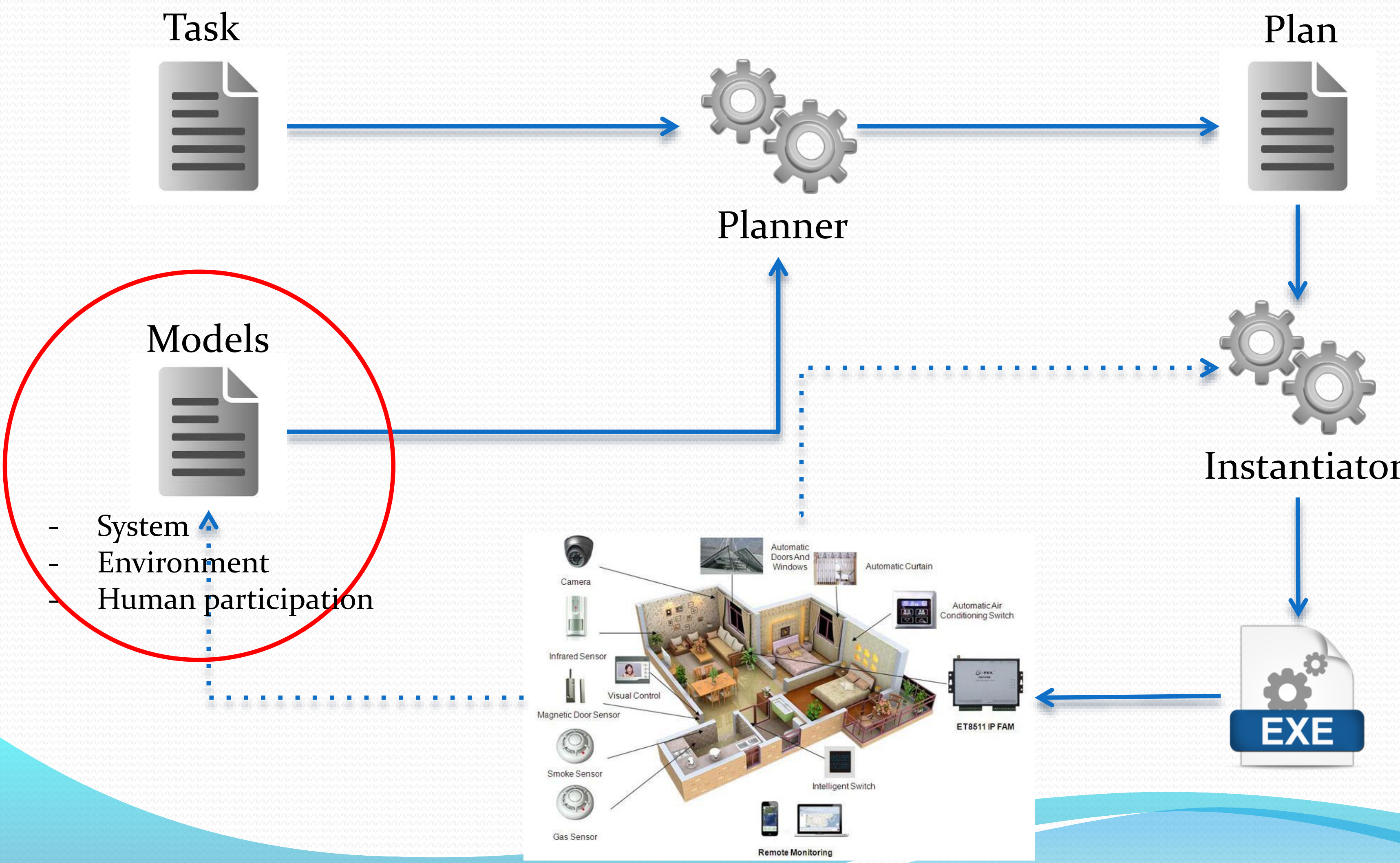
Engineering Process



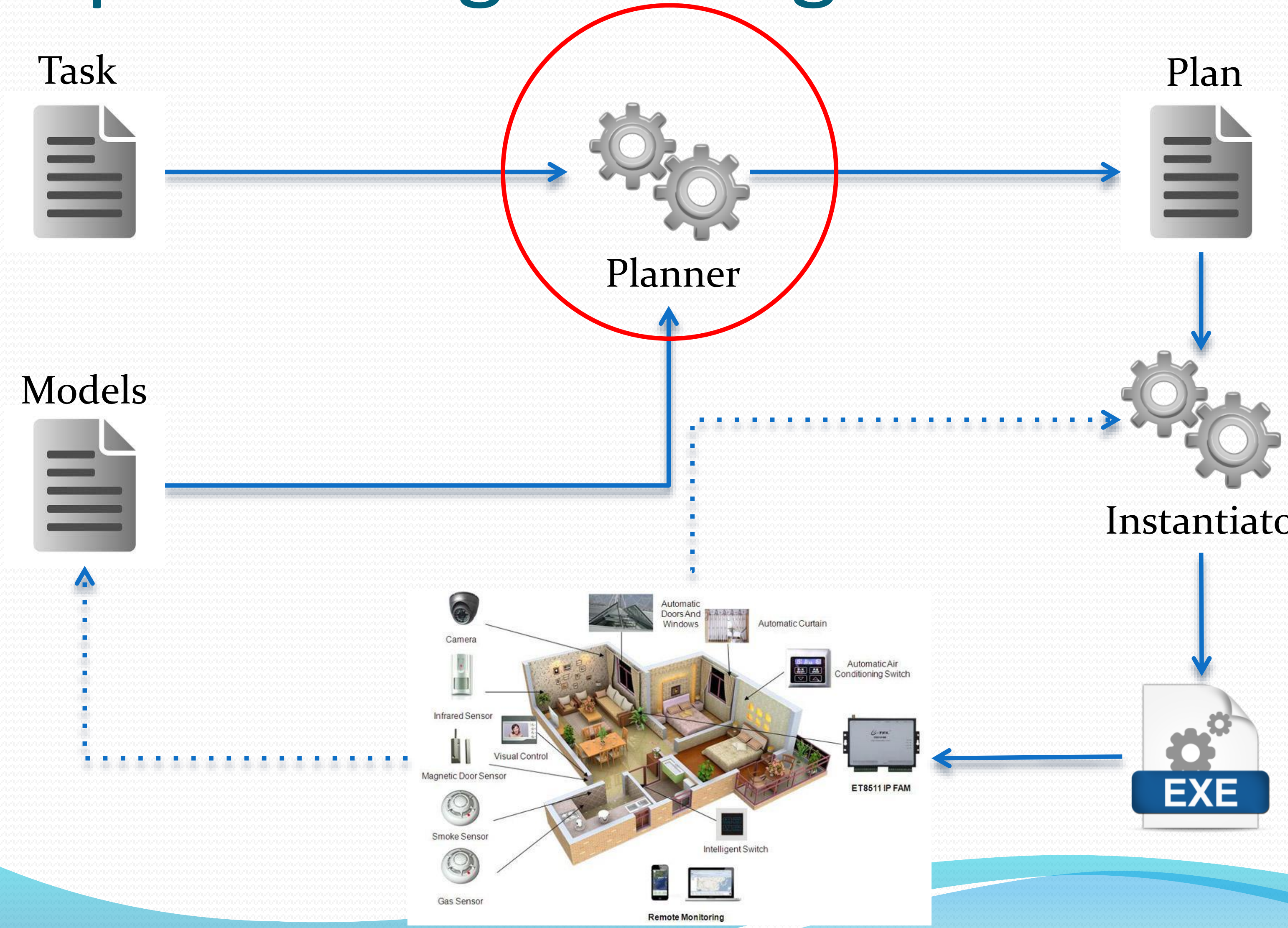
Proposed Engineering Process



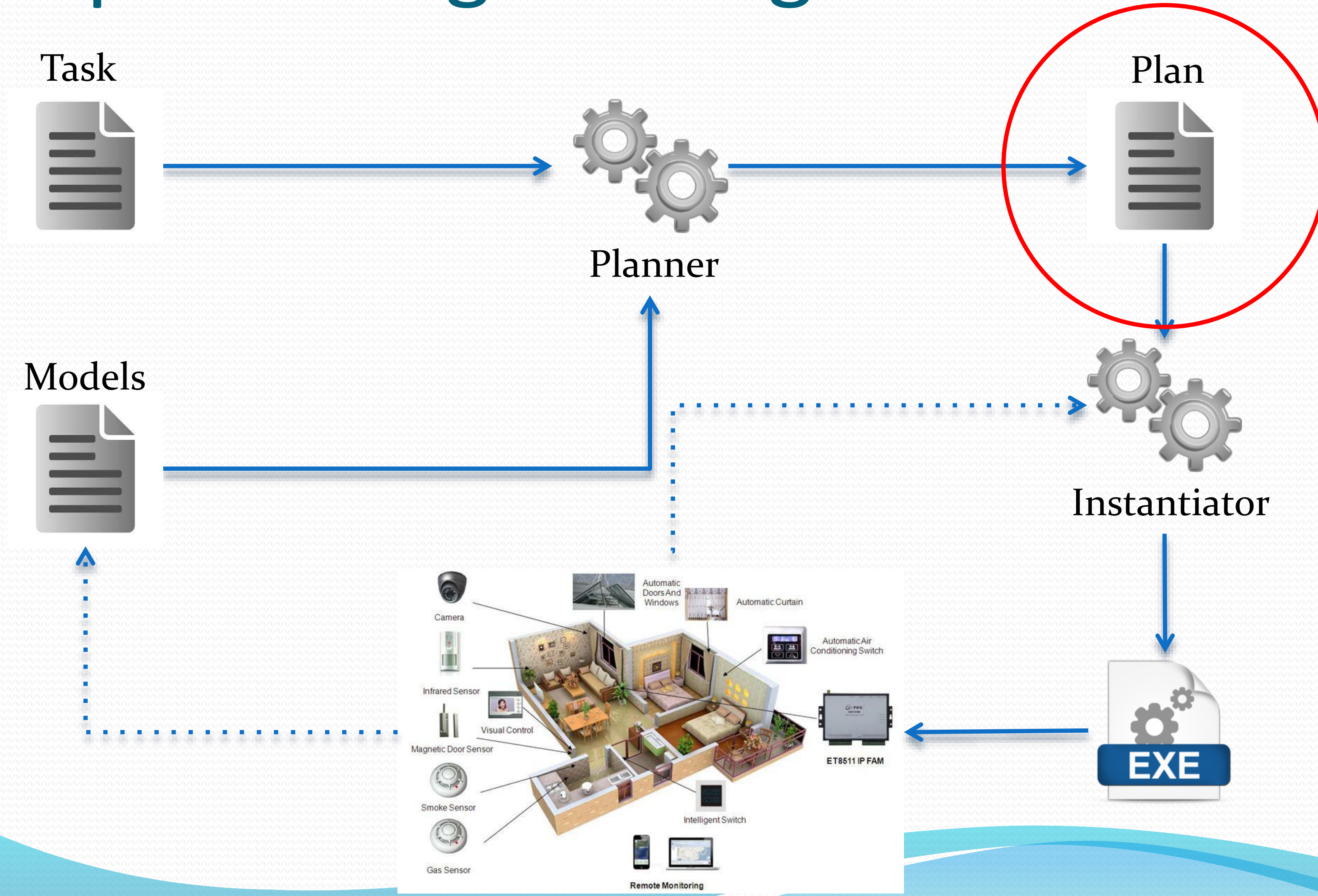
Proposed Engineering Process



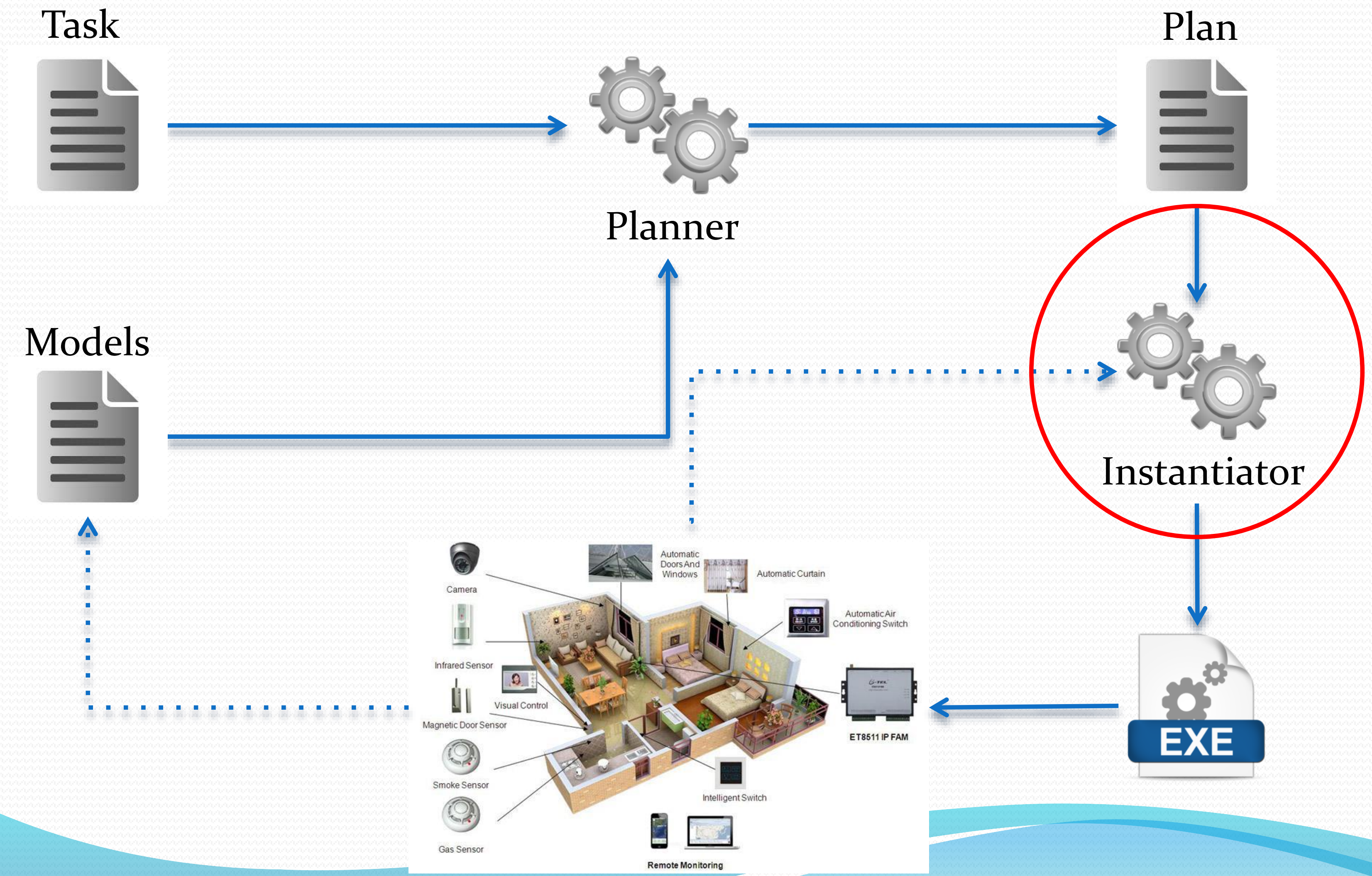
Proposed Engineering Process



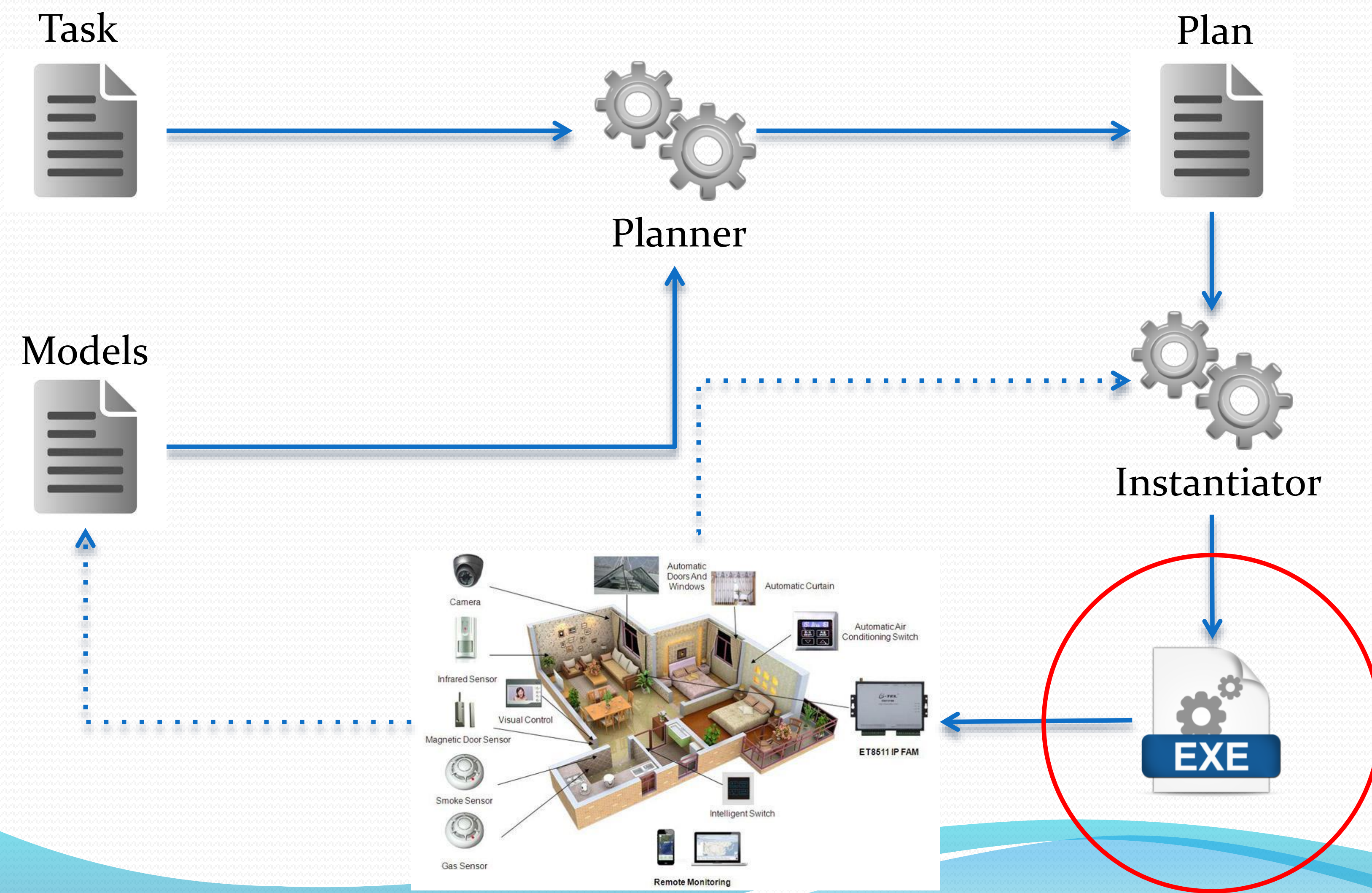
Proposed Engineering Process



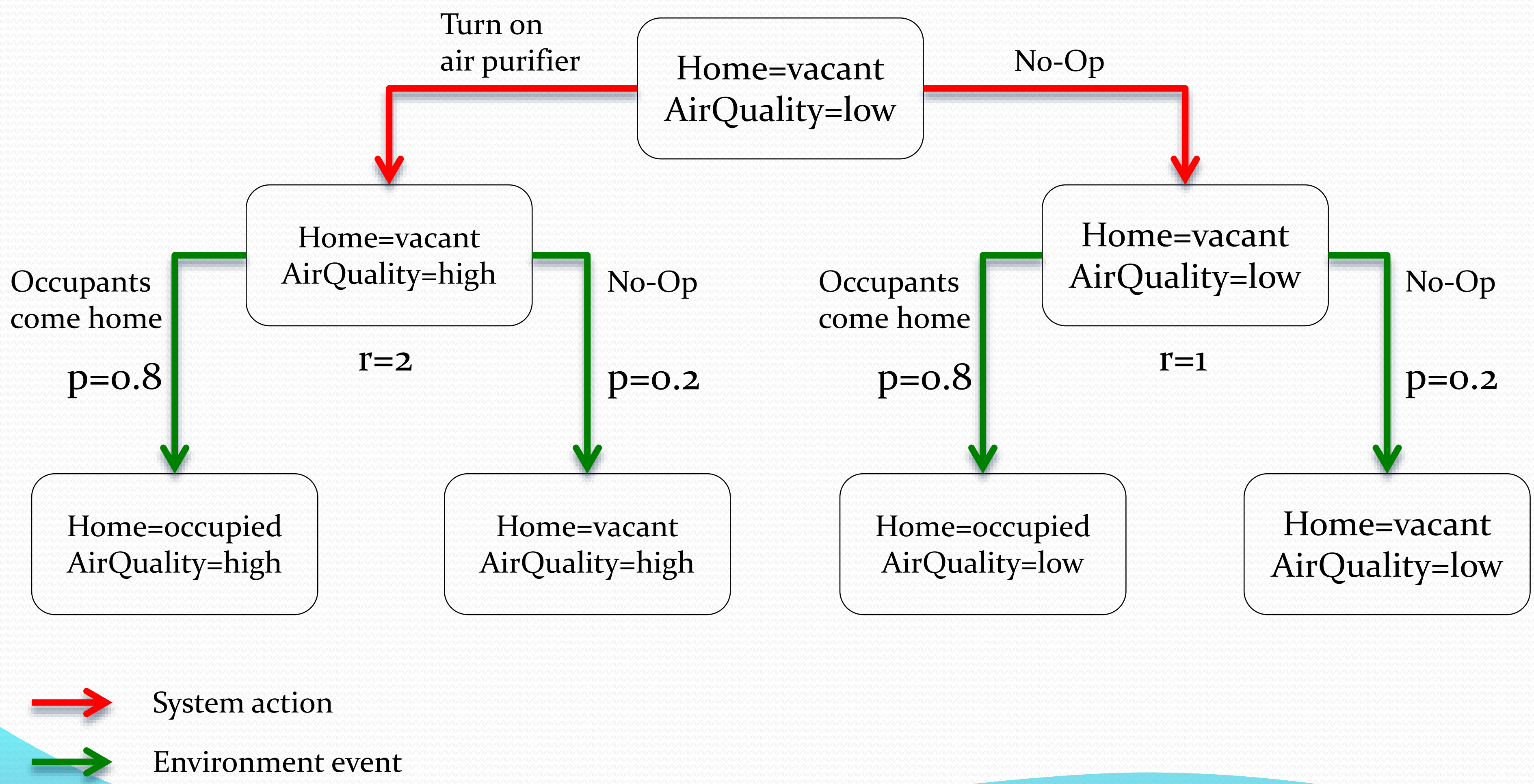
Proposed Engineering Process



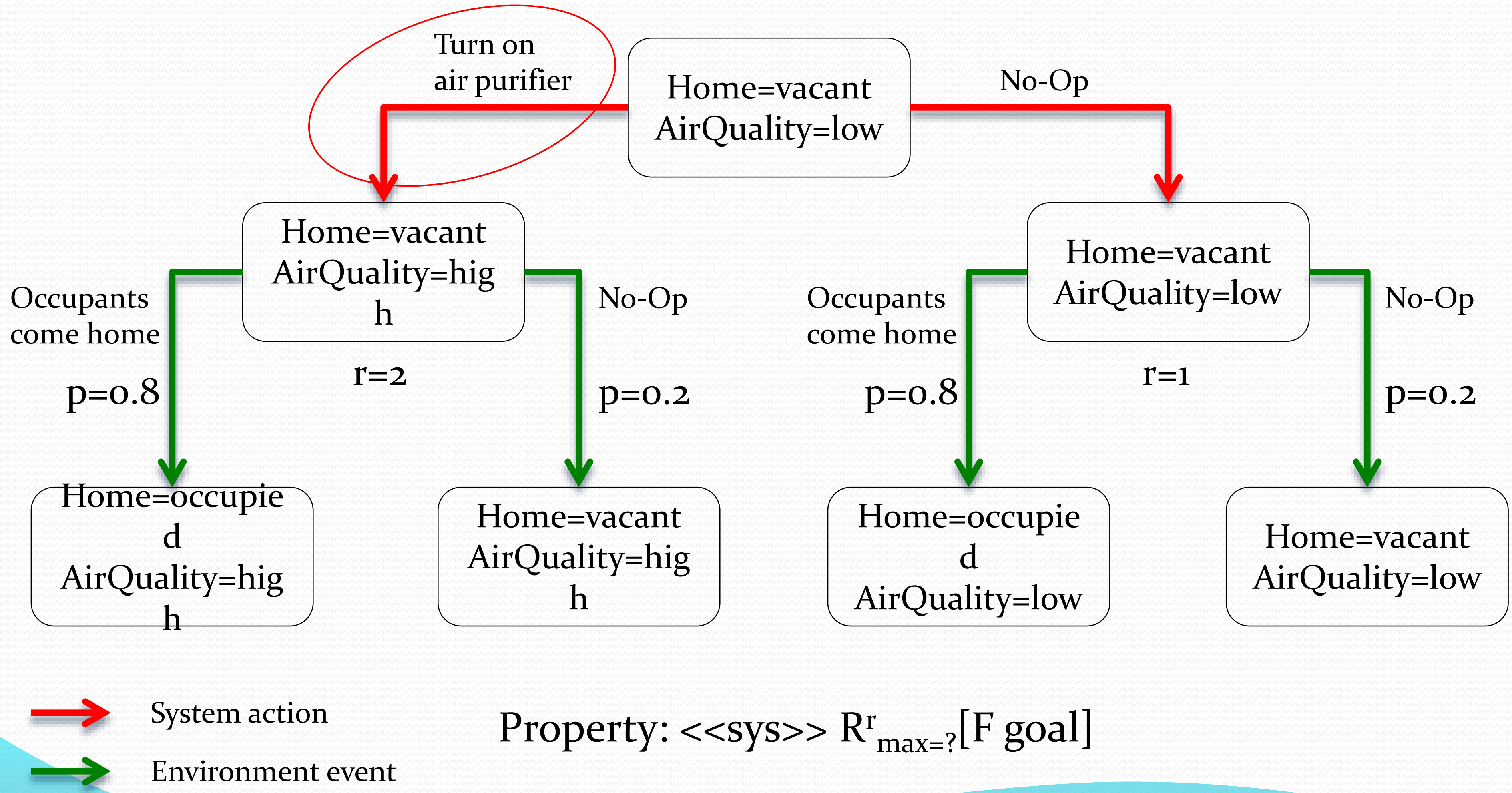
Proposed Engineering Process



Stochastic Multiplayer Games (SMGs)



Strategy Synthesis of SMGs



Property: <<sys>> $R^{r}_{max=?}$[F goal]

Indoor Air Quality Control: Human-in-the-Loop

Air quality sensors

Air purifier

Humans have their own objectives & priorities

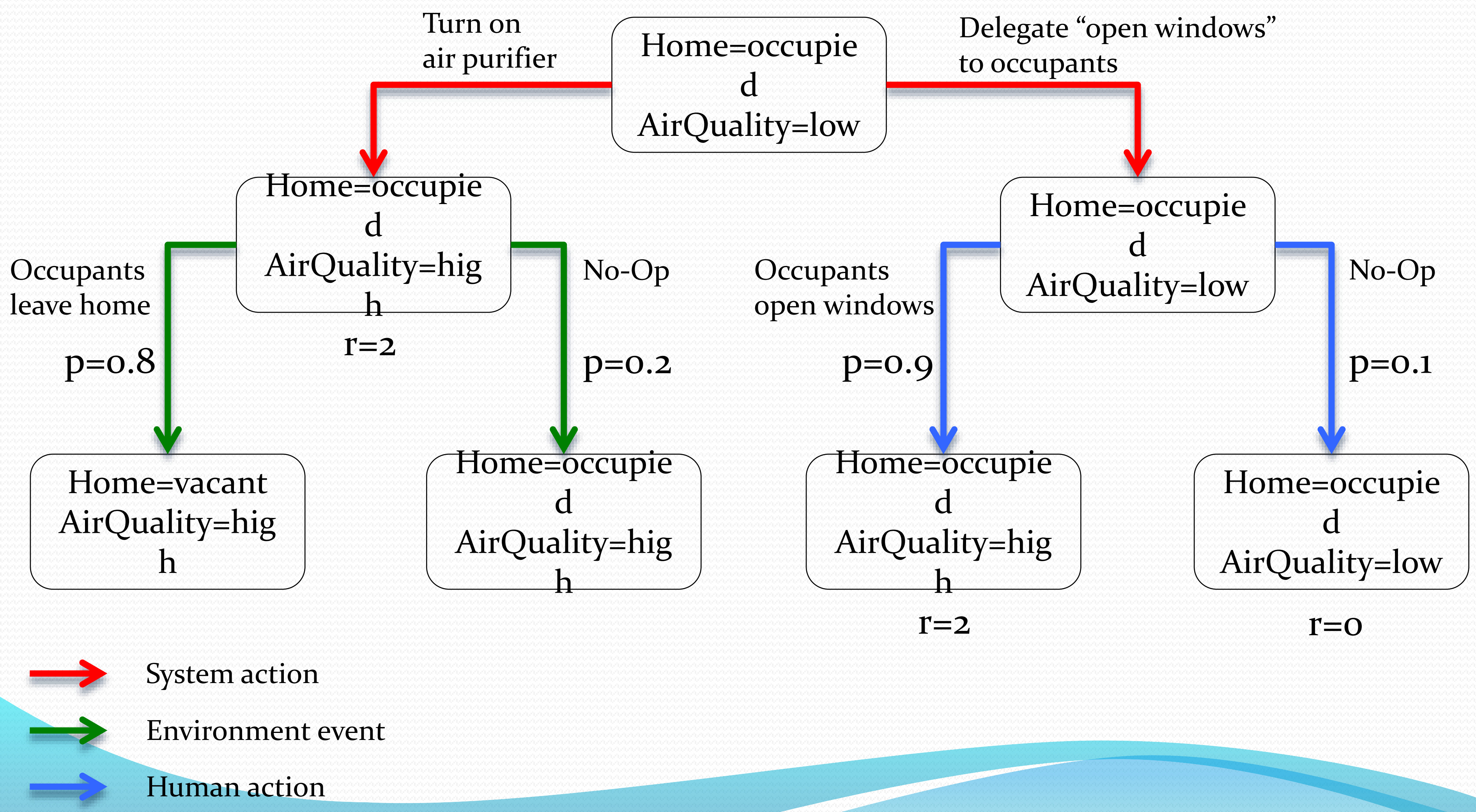
Uncertainty from humans

Human experience

Occupant/
Human Actuator

Dehumidifier

Delegation



Turn on
air purifier
Home=occupied
AirQuality=low
Delegate “open windows”
to occupants
Home=occupied
AirQuality=high
r=2
Home=occupied
AirQuality=low
Occupants
leave home
No-Op
Occupants
open windows
No-Op
p=0.8
p=0.2
p=0.9
p=0.1
Home=vacant
AirQuality=high
Home=occupied
AirQuality=high
Home=occupied
AirQuality=high
r=2
Home=occupied
AirQuality=low
r=0
System action
Environment event
Human action

Opportunity-Willingness-Capability

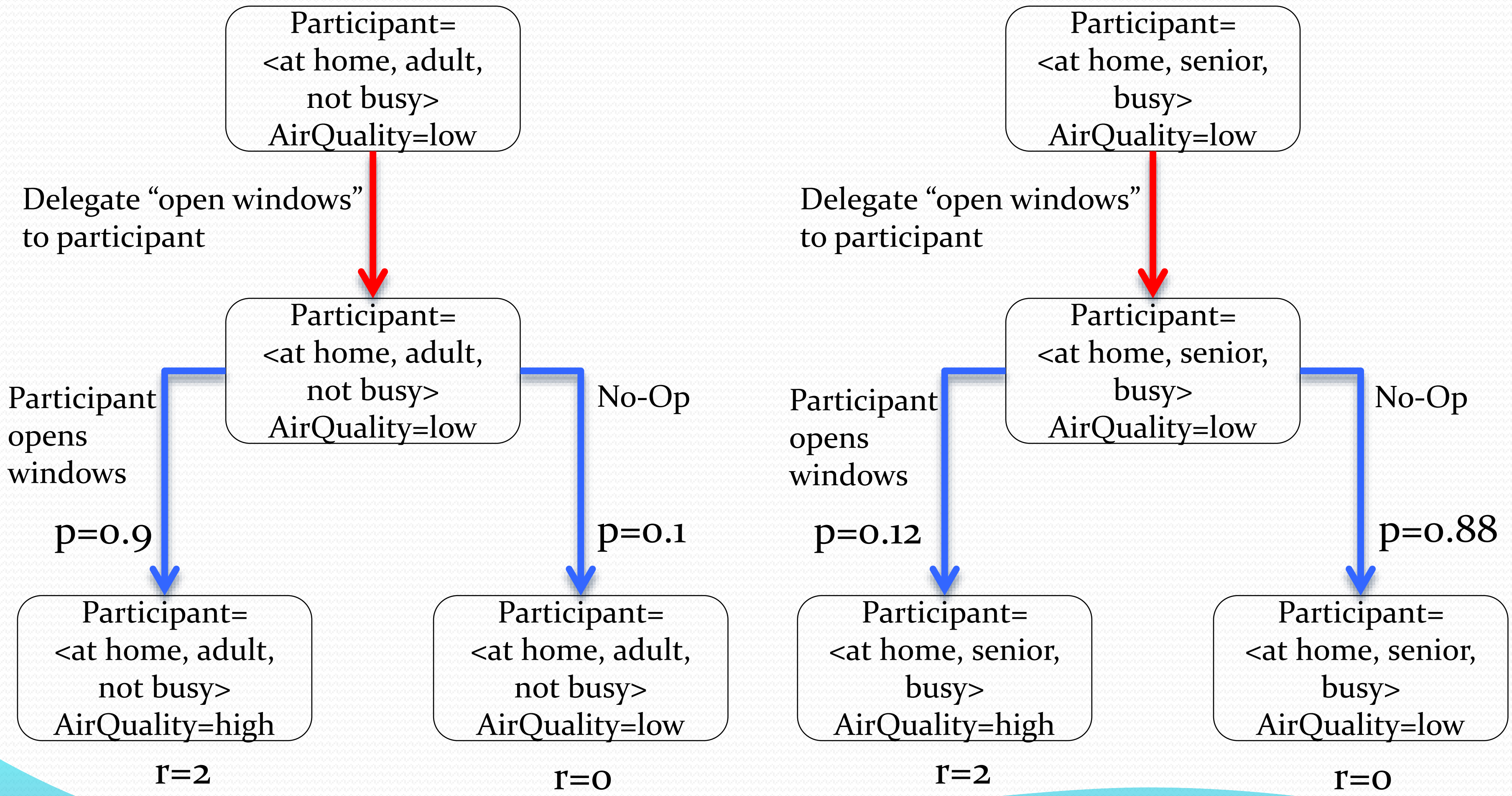
- Opportunity
  - Prerequisites for task performance
- Willingness
  - Desire of participants to perform task
- Capability
  - Capability of participants to perform task

Example OWC Model

τ = open windows		
Types	Elements	Functions
Opportunity	Participant's location	**Opportunity function** = is participant at home?
Willingness	Participant's availability	• If participant is busy: **Willingness probability** = 0.2 • If participant is *not* busy: **Willingness probability** = 0.9
Capability	Participant's age range	• If participant is adult: **Capability probability** = 1.0 • If participant is senior: **Capability probability** = 0.6

Given opportunity, success probability of τ is WP*CP

OWC Model in Delegation



Conclusion

- CPS requires unified treatment of cyber and physical aspects of systems design
- We are exploring the integration of heterogeneous modeling and analysis through architecture views
  - Provides formal criteria for structural and semantic consistency
  - Can be supported by tools that manage dependencies
- Humans in the loop require special treatment
  - We are investigating stochastic multi-player games to do automated control synthesis
- Many challenges remain

This Talk – Three Themes

- Theme 1: CPS is challenging in fundamental ways
  - Heterogeneity
  - Complexity
  - Uncertainty
- Theme 2: SE can help … but with modifications
  - Model-driven engineering
  - Architecture (and abstraction in general)
  - Tools
- Theme 3: But SE needs more to make it "smart"
  - Dealing with continuous behavior
  - Dealing with humans

References

- Ivan Ruchkin, Bradley Schmerl and David Garlan. Architectural Abstractions for Hybrid Programs. In Proc. of the 18th International ACM Sigsoft Symposium on Component-Based Software Engineering (CBSE 2015), Montréal, May 2015.
- A. Y. Bhave, D. Garlan, B. Krogh, A. Rajhans and B. Schmerl. Augmenting Software Architectures with Physical Components. In Proc. of the Embedded Real Time Software and Systems Conference (ERTS^2 2010), May 2010.
- Bhave, A., B.H. Krogh, D. Garlan, and B. Schmerl. View Consistency in Architectures for Cyber-Physical Systems. In 2011 IEEE/ACM International Conference on Cyber-Physical Systems (ICCPS), 151-160, 2011.
- Rajhans, Akshay, and Bruce H. Krogh. Heterogeneous Verification of Cyber-physical Systems Using Behavior Relations. In Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control, 35-44. HSCC'12. New York, NY, USA: ACM, 2012.

- A. Rajhans, S.-W. Cheng, B. Schmerl, D. Garlan, B. Krogh, C. Agbi and A. Y. Bhave. An Architectural Approach to the Design and Analysis of Cyber-Physical Systems. In Electronic Communications of the EASST, Vol. 21: Multi-Paradigm Modeling, 2009.
- A. Y. Bhave, B. Krogh, D. Garlan and B. Schmerl. Multi-domain Modeling of Cyber-Physical Systems using Architectural Views. In Proceedings of the 1st Analytic Virtual Integration of Cyber-Physical Systems Workshop, 2010. Co-located with RTSS 2010.
- A. Rajhans, A. Y. Bhave, S. Loos, B. Krogh, A. Platzer and D. Garlan. Using Parameters in Architectural Views to Support Heterogeneous Design and Verification. In 50th IEEE Conference on Decision and Control (CDC) and European Control Conference (ECC) December 2011.

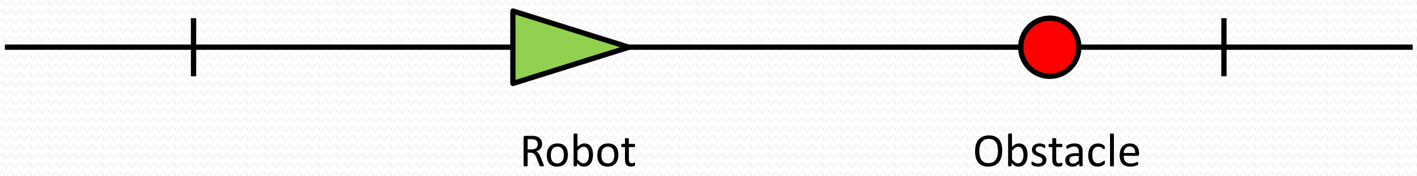
The End

Auxiliary Slides

Other case studies: Robotics

- Robotic control – drive to destination, avoiding collision with obstacles.
- Research problems:
  - Architecture-aware hybrid modeling.
  - Architectural support for theorem proving.

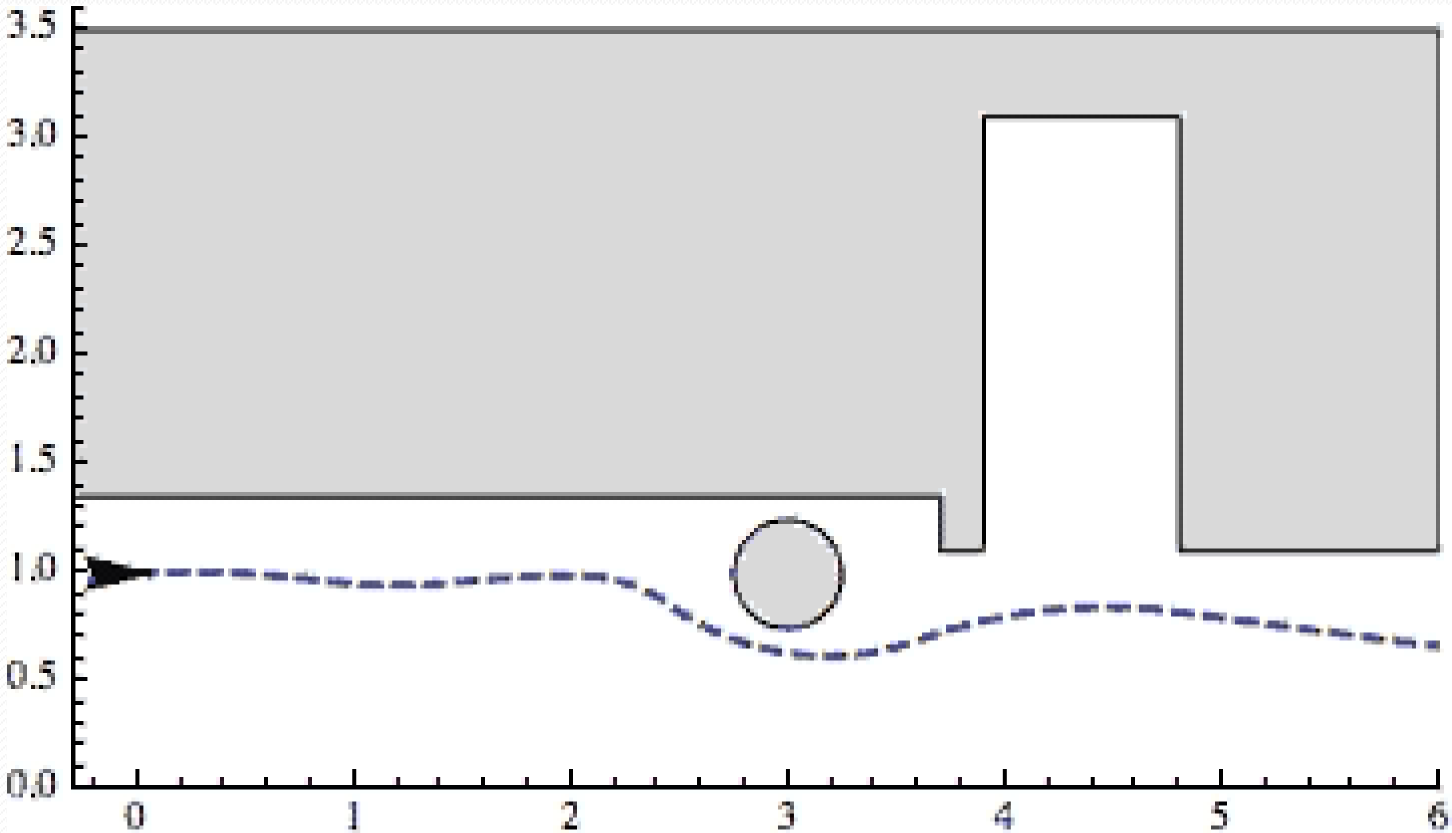
Example: Robot collision avoidance



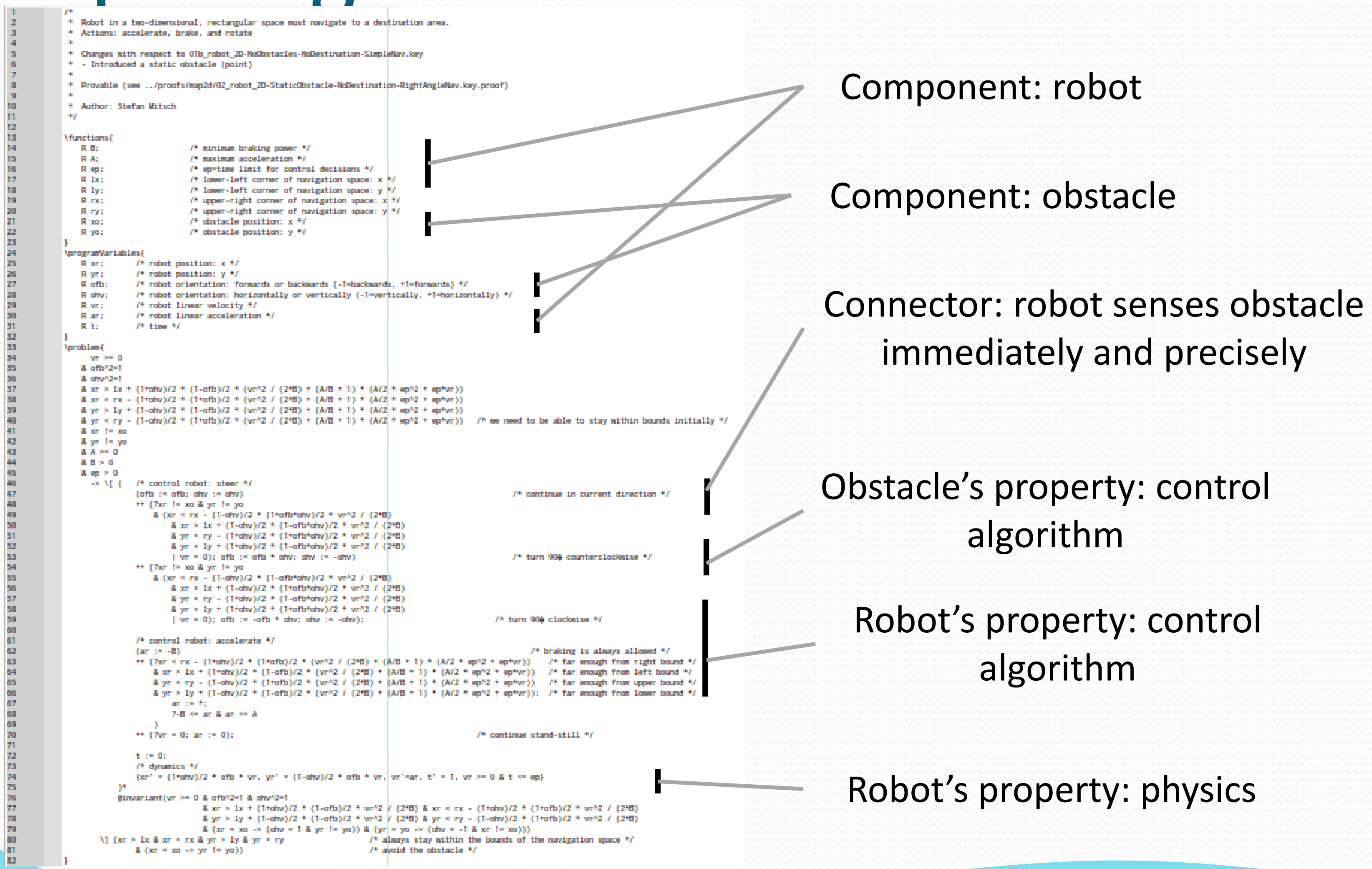
- A robot and an obstacle move in a one-dimensional space.
- The robot periodically senses the surrounding and may decide to accelerate or brake.
- The robot knows the bounds and senses the obstacle's location.
- Obstacle is assumed to travel at less than maximum speed.

Safety property: robot does not collide with the obstacle or the bounds.

Robot trajectory



Exposing Architecture



Solution: annotations