An Architectural Approach to the Design and Analysis of Cyber-Physical Systems

Akshay Rajhans, Shang-Wen Cheng, Bradley Schmerl, David Garlan, Bruce H. Krogh, Clarence Agbi, Ajinkya Bhave

Carnegie Mellon University

October 6, 2009

Outline

Outline

Motivation

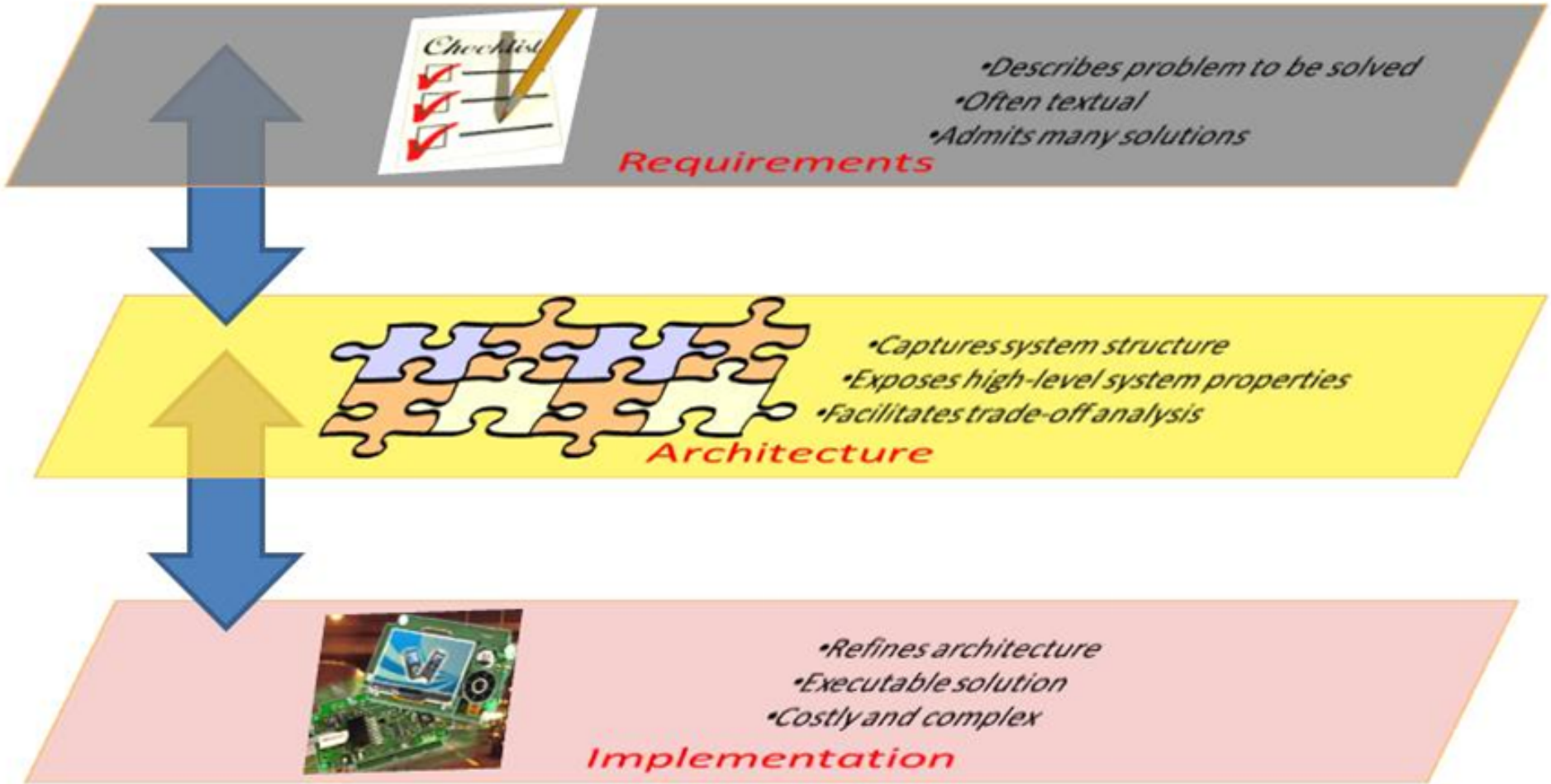
Today's approaches

- Early separation between cyber and physical parts of system design
- Different formalisms and methodologies from CS and engineering
- Problem: Tradeoffs and alternatives assesment extremely difficult if cyber and physical parts tightly coupled

Need for a unified approach that

- Blends the heterogenity between software and physical systems
- Treats cyber and physical elements equally
- Allows for the hierarchical design and compositionality

Typical system design flow



Software architecture like approach for CPS?

A little bit about software architecture (SA)

- Typically models a system as a graph of components and connectors
  - Components: computational elements, e.g., servers, databases, etc.
  - Connectors: communication pathways, e.g., protocols like RMI, http, etc.
  - Properties: abstract behavior of elements, e.g., expected load on servers, communication latencies, transaction rates of databases

Software architecture like approach for CPS?

A little bit about software architecture (SA)

- Typically models a system as a graph of components and connectors
  - Components: computational elements, e.g., servers, databases, etc.
  - Connectors: communication pathways, e.g., protocols like RMI, http, etc.
  - Properties: abstract behavior of elements, e.g., expected load on servers, communication latencies, transaction rates of databases

What's good about software architecture?

SA supports:

- Structured complexity reduction – abstraction and encapsulation
- Uniform treatment – whether implemented in software or hardware
- Compositionality – partial analysis of the full system

Outline

Building a CPS architectural modeling vocabulary

Considerations

- Treating cyber and physical elements equally
- Balance between generality and specificity
  - Specificity: focus on embedded monitoring and control systems domain
  - Generality: keep it general enough so it becomes a foundation for more specific application areas within this domain

Approach

- cyber family – cyber components and connectors
- physical family – physical components and connectors
- cyber-physical interface family – inherits both cyber and physical families, adds its own set of elements

Cyber family

Cyber components

- Data Stores:
  - Hold data, e.g., memory blocks
- Computation components:
  - Operate on and update data, e.g., filters, state estimators, controllers
- IO interface software:
  - Raw readings to/from data in usable form, e.g., smart sensor software

Cyber connectors

- Call-return connector:
  - Represents one-to-one communication between software components, e.g., a subroutine call
- Publish-subscribe connector:
  - Represents one-to-many communication, e.g., writer with multiple readers

Physical family

Physical components

- Sources:
  - deliver power, have only output ports (sinks are negative sources)
- Energy storage:
  - elements with dynamics, store energy, power transfer both ways
- Physical transducers:
  - convert energy, e.g., motors - electrical to mechanical

Physical connectors

- Power flow connectors:
  - bidirectional, dynamic coupling between components
- Shared-variable connectors:
  - equality constraints, no directionality
- Measurement connectors:
  - one-way directionality, similar to signal lines in Simulink

Cyber-physical family

Adds new elements that bridge the gap between computational and physical systems and model the interactions between them

Cyber-physical components

- P2C transducer and C2P transducer components
  - have ports to cyber elements on one side and ports to physical elements on the other side

Cyber-physical connectors

- P2C and C2P directed connectors
  - model simple sensors and actuators respectively

Outline

A temperature control system

The goal

- to maintain the temperature of a room within a desired band around a set point

A temperature control system

The goal

- to maintain the temperature of a room within a desired band around a set point

The structural elements

- a room whose temperature is to be maintained in the desired range
- a thermostat that has the set point, and gets the temperature readings from the room
- a furnace heats the room when on, room cools due to the ambient otherwise

A temperature control system

The goal

- to maintain the temperature of a room within a desired band around a set point

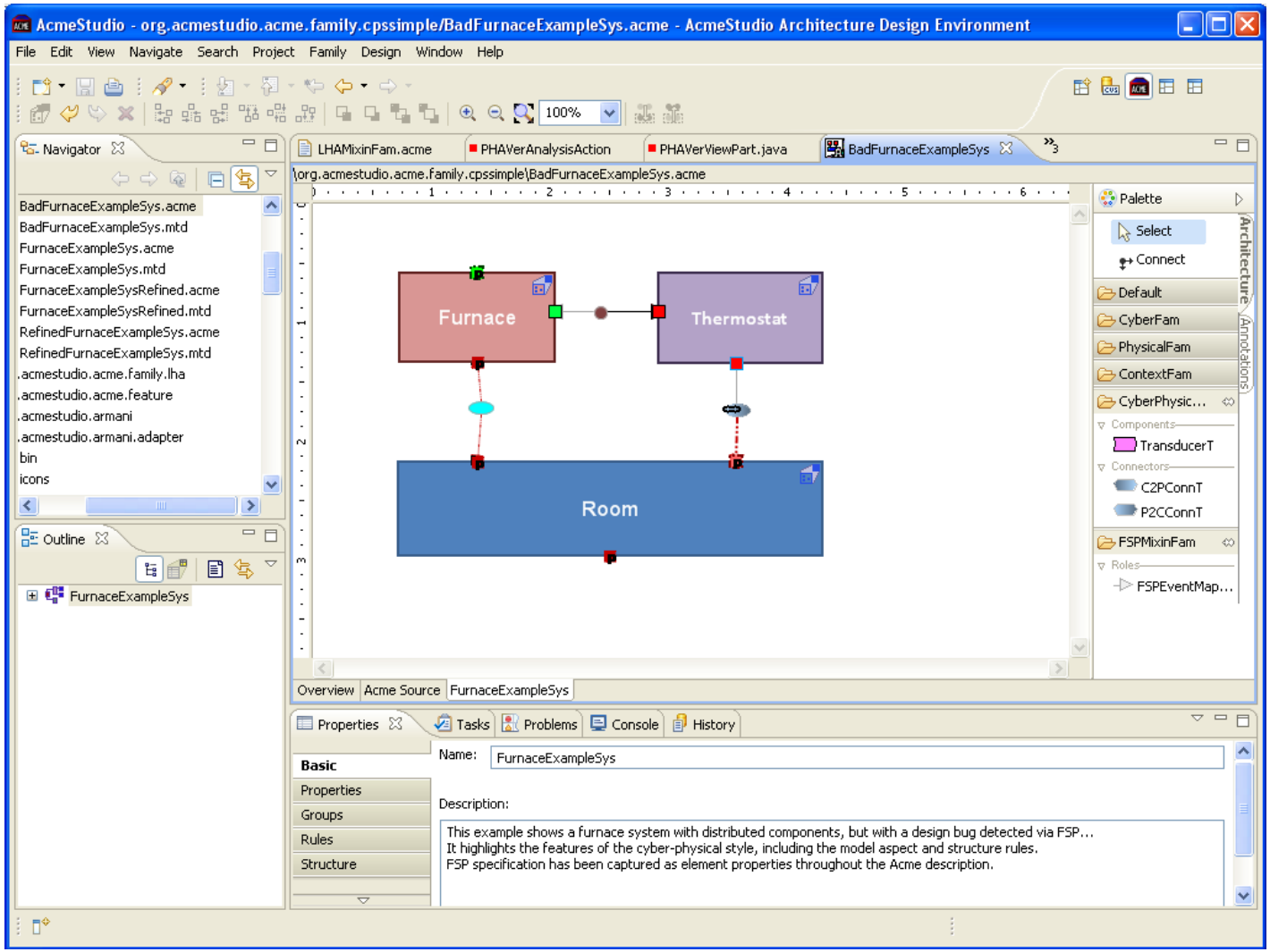
The structural elements

- a room whose temperature is to be maintained in the desired range
- a thermostat that has the set point, and gets the temperature readings from the room
- a furnace heats the room when on, room cools due to the ambient otherwise

The interconnections

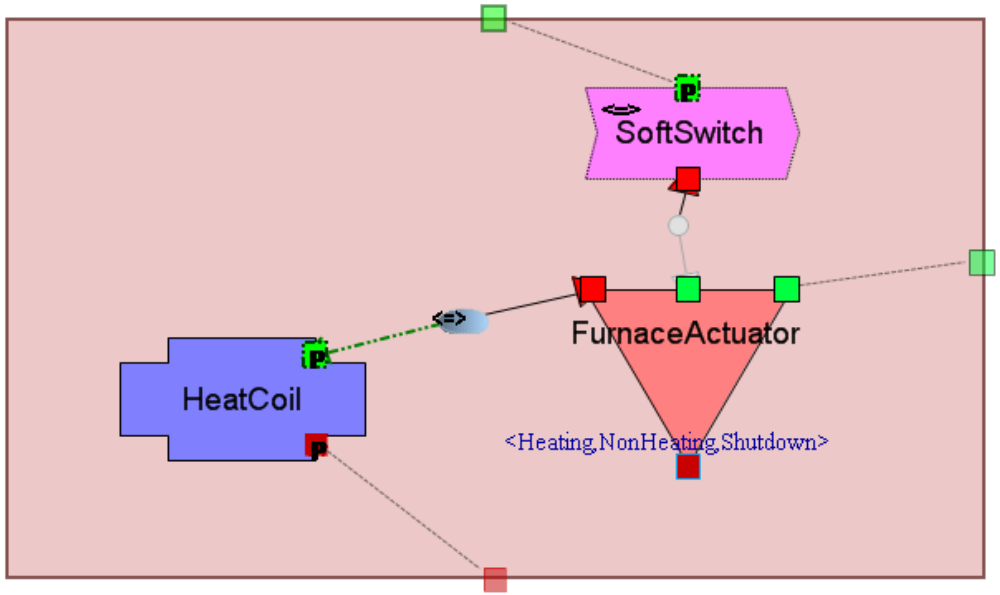
- room to thermostat: temperature reading
- thermostat to furnace: commands to start and stop heating
- furnace to room: heat

Architectural modeling of the thermostat system

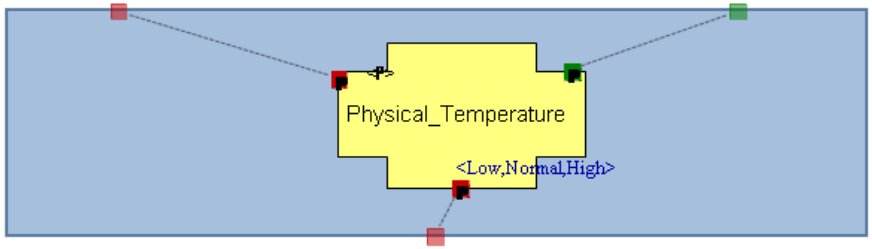


Hierarchical design: top level structure

Architectural modeling of the thermostat system

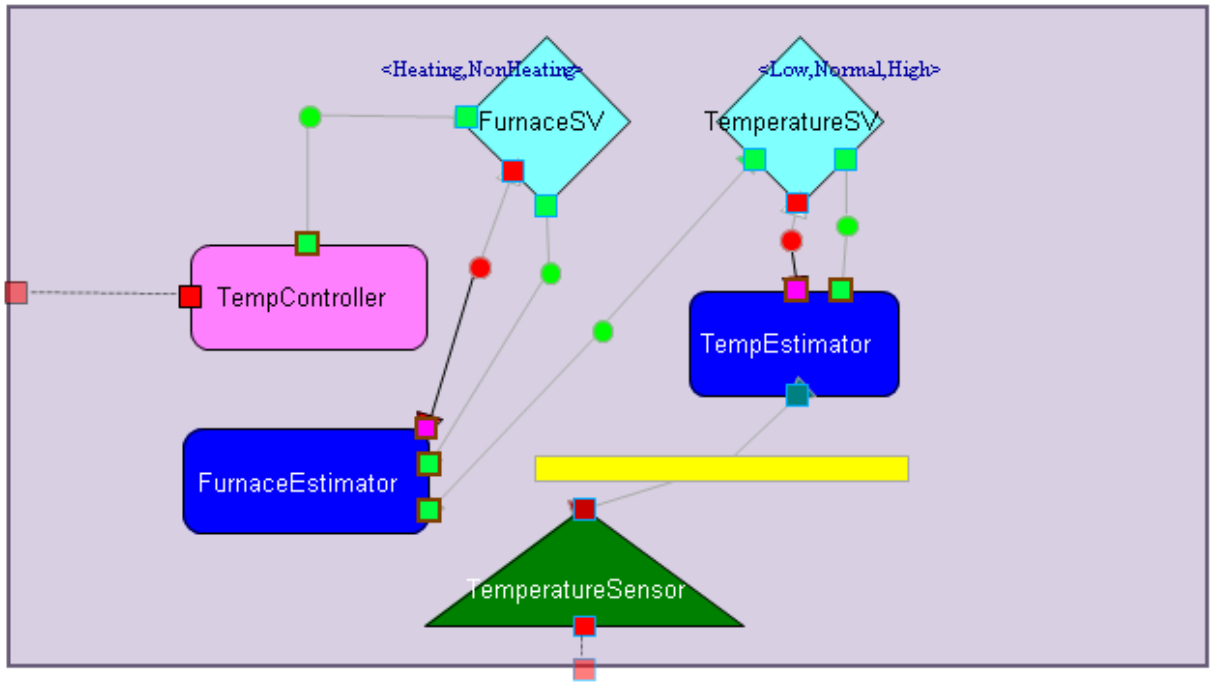


Building the substructure of the hierarchical component ‘furnace’ (cyber-physical)



Building the substructure of the hierarchical component ‘room’ (entirely physical)

Architectural modeling of the thermostat system



Building the substructure inside the component 'thermostat' (entirely cyber)

Outline

Behavioral annotation

Behavioral information

- Behavioral information not a part of system structure
- Pertains to elements or to the whole system
- Different formalisms, e.g., textual, FSP, LHA, matlab code, ...
- Possibly subject to different implementations

Behavioral annotation

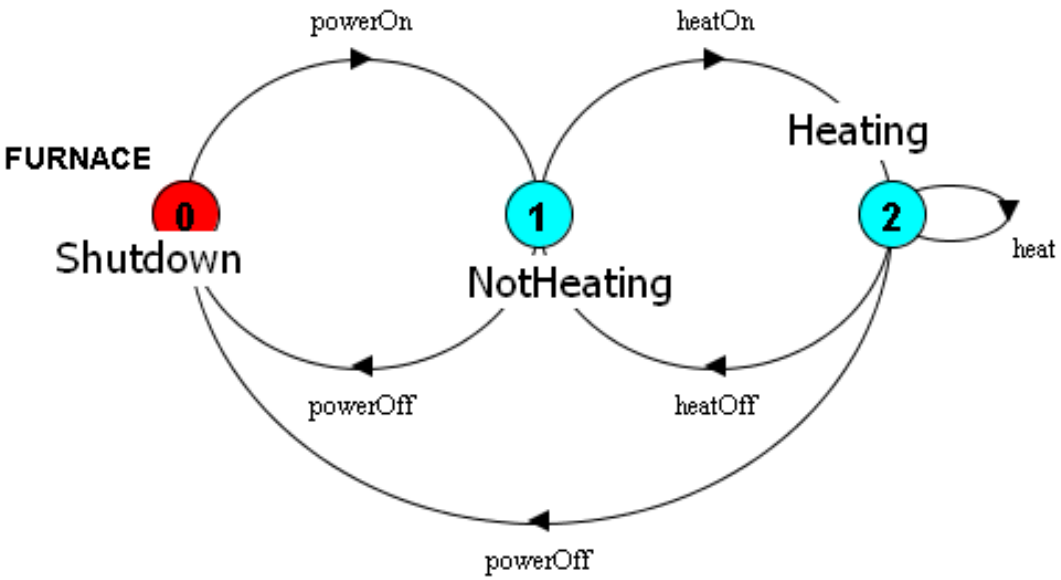
Behavioral information

- Behavioral information not a part of system structure
- Pertains to elements or to the whole system
- Different formalisms, e.g., textual, FSP, LHA, matlab code, ...
- Possibly subject to different implementations

We use the following two formalisms:

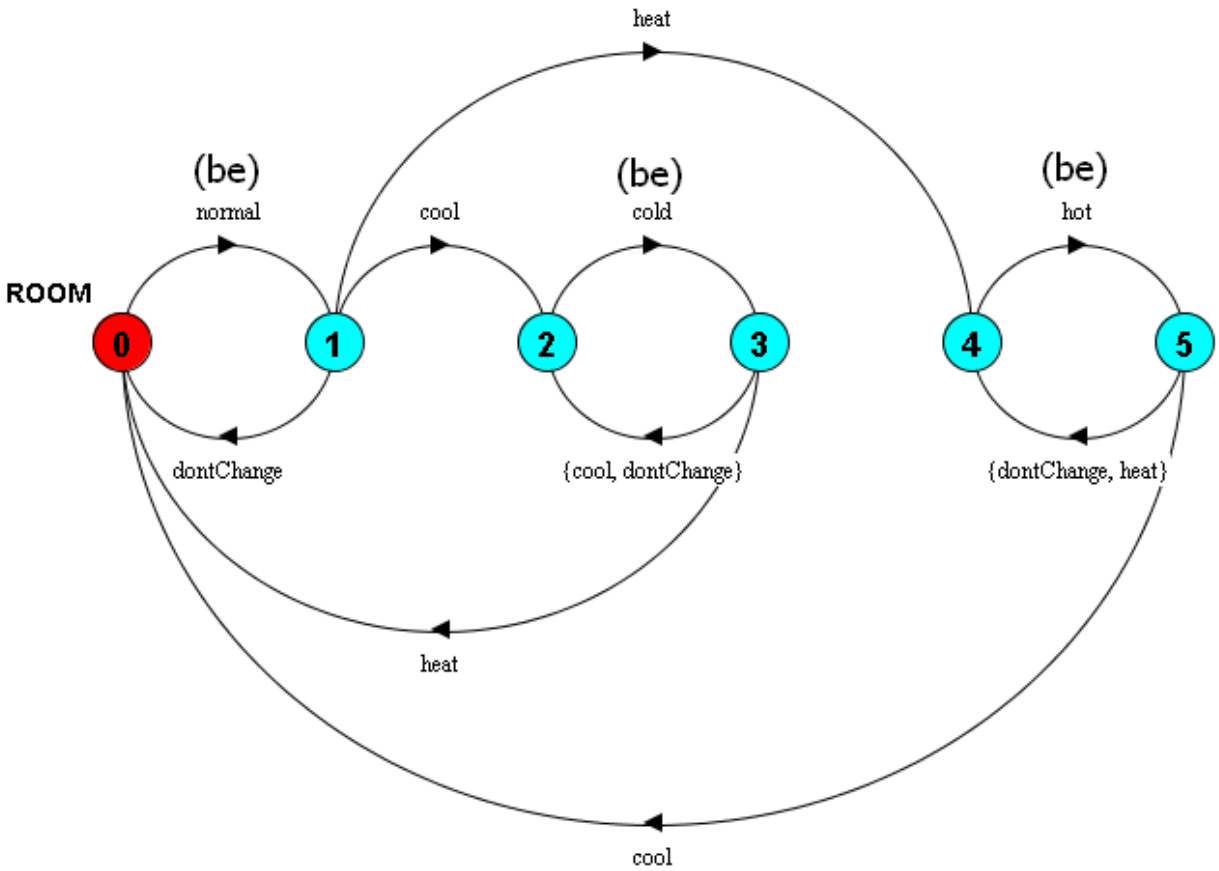
Formalism	Finite state processes (FSP)	Linear hybrid automata (LHA)
State space	purely discrete	discrete and continuous
Primitive element	(primitive) process	linear hybrid automata
Composite element	composite process	linear hybrid automaton
Interaction between primitive elements	synchronizing events	synchronizing events continuous inputs continuous outputs
Analysis tool	Labelled Transition System Analyser (LTSA)	Polyhedral Hybrid Automata Verifyer (PHAVer)

Furnace process



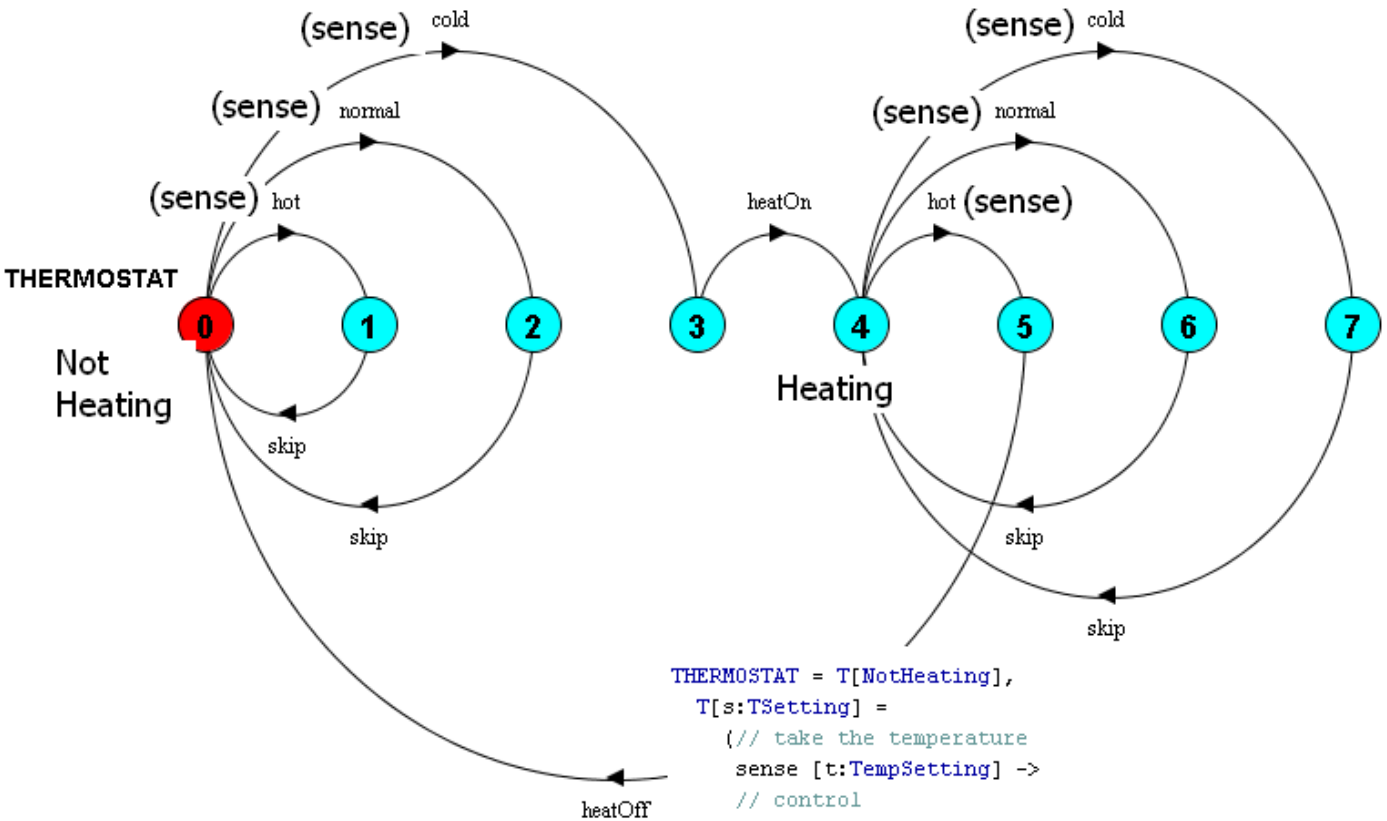

```
FURNACE = F[Shutdown],
  F[s:FSetting] =
    // Shutdown case
    (when (s==Shutdown) powerOn -> F[NotHeating]
    // NotHeating case
    |when (s==NotHeating) heatOn -> F[Heating]
    |when (s==NotHeating) powerOff -> F[Shutdown]
    // Heating
    |when (s==Heating) heatOff -> F[NotHeating]
    |when (s==Heating) heat -> F[Heating]
    |when (s==Heating) powerOff -> F[Shutdown]
    ).
```

Room process




```
ROOM = E[Normal],
  E[temp:TempSetting] =
    (be[temp] ->
     C[temp])
    ,
  C[temp:TempSetting] =
    (// Low temp cases
     when (temp==Low) heat -> E[Normal]
    |when (temp==Low) cool -> E[Low]
    |when (temp==Low) dontChange -> E[Low]
    // Normal temp cases
    |when (temp==Normal) heat -> E[High]
    |when (temp==Normal) cool -> E[Low]
    |when (temp==Normal) dontChange -> E[Normal]
    // High temp cases
    |when (temp==High) heat -> E[High]
    |when (temp==High) cool -> E[Normal]
    |when (temp==High) dontChange -> E[High]
    ).
```

Thermostat process




```
THERMOSTAT = T[NotHeating],
  T[s:TSetting] =
    (// take the temperature
     sense [t:TempSetting] ->
     // control
        (when (s==NotHeating && t==Low) heatOn -> T[Heating]
        |when (s==Heating && t==High) heatOff -> T[NotHeating]
        |when (t==Normal || (s==Heating && t==Low) || (s==NotHeating && t==High)) skip -> T[s]
        )
    ).
```

A quick liveness check in LTSA

If room temperature drops, can it always eventually get back to normal?

fluent BEINGCOLD = <cold, {normal, hot}>
fluent BEINGNORMAL = <normal, {cold, hot}>
fluent BEINGHOT = <hot, {cold, normal}>
assert RETURN2NORMAL = [](BEINGCOLD -> <>BEINGNORMAL)

A quick liveness check in LTSA

If room temperature drops, can it always eventually get back to normal?

fluent BEINGCOLD = <cold, {normal, hot}>
fluent BEINGNORMAL = <normal, {cold, hot}>
fluent BEINGHOT = <hot, {cold, normal}>
assert RETURN2NORMAL = [](BEINGCOLD -> <>BEINGNORMAL)

No!

LTL Property Check...
– States: 21 Transitions: 71 Memory used: 8946K
Finding trace to cycle...
Depth 11 – States: 71 Transitions: 266 Memory used: 8832K
Finding trace in cycle...
Depth 1 – States: 1 Transitions: 1 Memory used: 9236K

Violation of LTL property: @RETURN2NORMAL

Trace to terminal set of states:

powerOn	
normal	BEINGNORMAL
skip	BEINGNORMAL
cool	BEINGNORMAL
cold	BEINGCOLD
heatOn	BEINGCOLD
powerOff	BEINGCOLD
powerOn	BEINGCOLD
cool	BEINGCOLD
cold	BEINGCOLD
cool	BEINGCOLD

Cycle in terminal set:

powerOff	BEINGCOLD
powerOn	BEINGCOLD

LTL Property Check in: 0ms

Where was the problem?

Observation of what happened

1. there is a power glitch, furnace gets turned off

Where was the problem?

Observation of what happened

1. there is a power glitch, furnace gets turned off
2. temperature drops and thermostat sends the heatOn command

The problem

Where was the problem?

Observation of what happened

1. there is a power glitch, furnace gets turned off
2. temperature drops and thermostat sends the heatOn command
3. furnace gets powered back on

The problem

- Furnace misses the message from thermostat

Where was the problem?

Observation of what happened

1. there is a power glitch, furnace gets turned off
2. temperature drops and thermostat sends the heatOn command
3. furnace gets powered back on
4. thermostat thinks since it sent the command, furnace must be heating

The problem

- Furnace misses the message from thermostat
- Thermostat has no knowledge of the actual furnace state

Where was the problem?

Observation of what happened

1. there is a power glitch, furnace gets turned off
2. temperature drops and thermostat sends the heatOn command
3. furnace gets powered back on
4. thermostat thinks since it sent the command, furnace must be heating
5. furnace will continute to operate in not heating mode eternally

The problem

- Furnace misses the message from thermostat
- Thermostat has no knowledge of the actual furnace state

Possible solutions

Where was the problem?

Observation of what happened

1. there is a power glitch, furnace gets turned off
2. temperature drops and thermostat sends the heatOn command
3. furnace gets powered back on
4. thermostat thinks since it sent the command, furnace must be heating
5. furnace will continute to operate in not heating mode eternally

The problem

- Furnace misses the message from thermostat
- Thermostat has no knowledge of the actual furnace state

Possible solutions

1. Thermostat to read the actual furnace state

Where was the problem?

Observation of what happened

1. there is a power glitch, furnace gets turned off
2. temperature drops and thermostat sends the heatOn command
3. furnace gets powered back on
4. thermostat thinks since it sent the command, furnace must be heating
5. furnace will continute to operate in not heating mode eternally

The problem

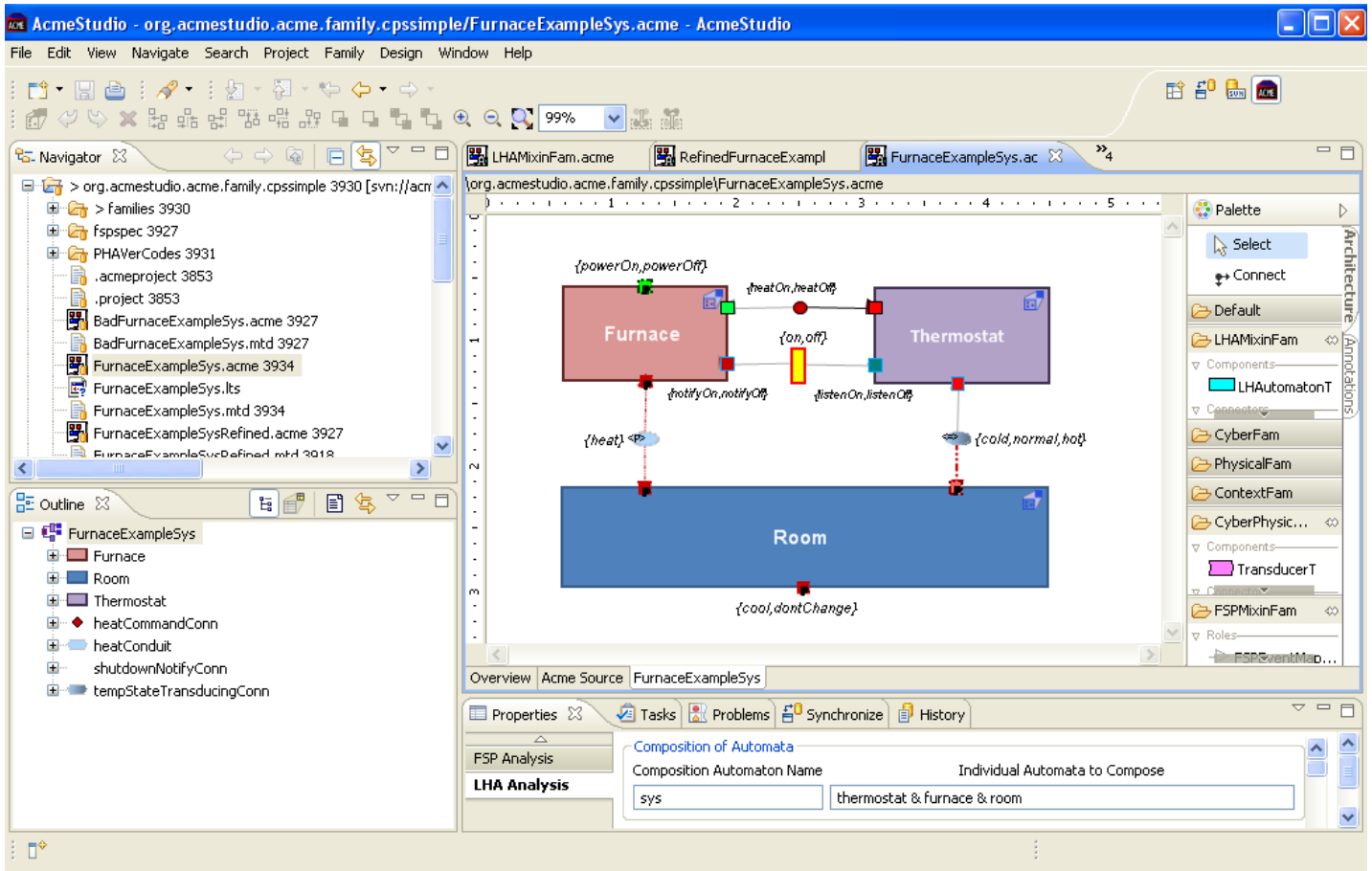
- Furnace misses the message from thermostat
- Thermostat has no knowledge of the actual furnace state

Possible solutions

1. Thermostat to read the actual furnace state
2. Thermostat times out and resends the command

Solution 1 - thermostat listens to furnace state

Add a new 'shutdown notify' connector



Corrected architecture according to solution 1

Solution 1 - thermostat listens to furnace state

Corrected FSP model of thermostat

```
THERMOSTAT = T[Disabled],
T[s:TSetting] =
(// take the temperature
sense [t:TempSetting] -> 
// control
(when (s==Disabled) listenOn -> T[NotHeating]
|when (s!=Disabled) listenOff -> T[Disabled]
|when (s==NotHeating && t==Low) heatOn -> T[Hating]
|when (s==Heating && t==High) heatOff -> T[NotHeating]
|when (t==Normal || (s==Heating && t==Low) || (s==NotHeating && t==High)) skip -> T[s]
)
).
```

LTSA analysis output

```
LTL Property Check...
– States: 212 Transitions: 589 Memory used: 6652K
No LTL Property violations detected.
LTL Property Check in: 16ms
```

Solution 2 - use a time-out

Note: no architectural change, just a behavioral change

The strategy

- thermostat sends a new command after a time-out interval if the temperature does not rise
- success of the strategy depends on various parameters
  - time-out interval
  - rates of cooling and heating of the room, upper and lower bounds if exact values unknown
  - magnitude of the hysteresis band in the thermostat, ...
- can't be analyzed in FSP - continuous time, rates, bands, intervals

Use LHA!

LHA analysis in PHAVer

PHAVer code – skeleton

```
automaton thermostat
...
end
automaton furnace
...
end
automaton room
...
end
automaton spec
...
end
sys = thermostat & furnace & room;
is_sim(sys,spec);
```

LHA analysis in PHAVer

PHAVer code – skeleton

automaton thermostat
...
end
automaton furnace
...
end
automaton room
...
end
automaton spec
...
end
sys = thermostat & furnace & room;
is_sim(sys,spec);

PHAVer output (For the right values of parameters):

Checking thermostat∼room∼furnace <= spec
– – – – – – – – – – – –
Ini states in simulation relation: yes
Finished. Exiting.
– – – – – – – – – – –

Outline

Summary

Summary

- uniform treatment of cyber and physical elements is sought
- software architecture provides a good starting point
- software architecture concepts are extended to cyber-physical systems
- new vocabulary for physical and cyber-physical elements is developed
- behavioral analysis can perfoermed using various modeling formalisms
- alternative structures and design options can be evaluated

Contributions of the paper

- architectural styles for modeling cyber-physical systems
- behavioral analysis using FSP and LHA
- evaluation of alternative architectures

Thank you

Acknowledgments

- Thanks to my co-authors for their contributions
- Thanks to NSF and AFOSR for grants
- Thanks to you for patient listening