Carnegie Mellon SCHOOL OF COMPUTER SCIENCE

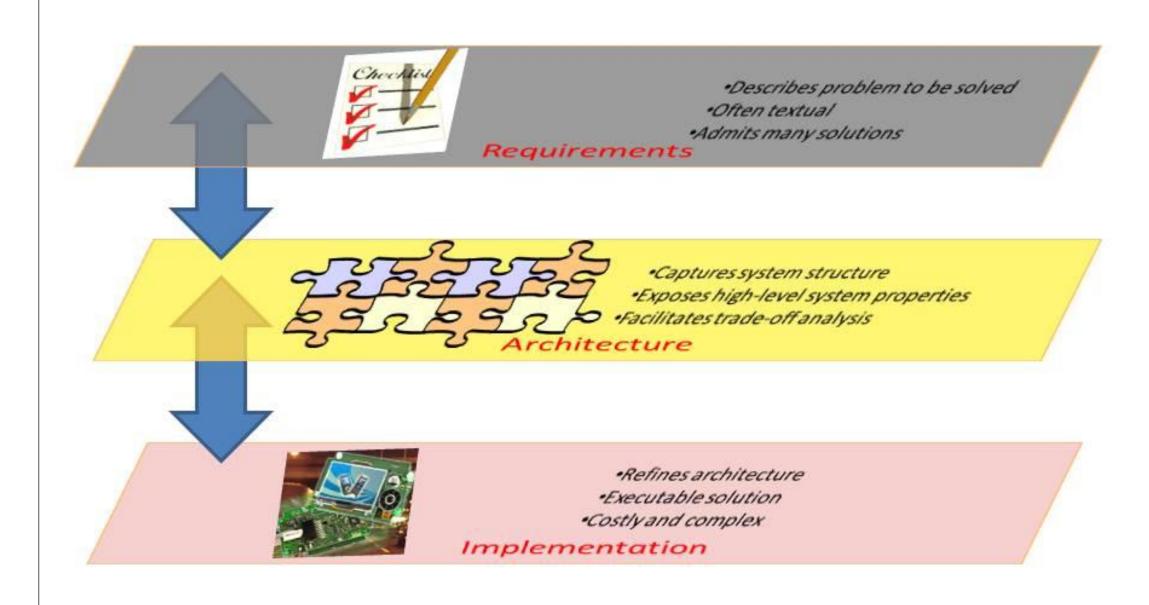
## **Objectives and Approach**

- Blend disparate design and analysis approaches for software and physical systems into a unified approach for cyber-physical systems
- Extend the model structures and analyses from software architecture to cyber-physical systems
  - structural annotations to specify and check correct interconnections and interfaces
  - semantic annotations for formal analysis —
  - design trade-offs at the architectural level
  - reuse recurring architectural patterns

Electrical & Computer

#### Architectures

**Carnegie Mellon** 



#### Software architecture

- Provides principled approach for design and analysis of software systems
- Well-established description languages, e.g., Acme
- Design environments, e.g., AcmeStudio

#### Modeling extensions for cyber-physical systems

- Properties of physical components and physical environments
- Compositions of physical elements
- Interfaces and interconnections between cyber and physical domains

#### **Analyses of cyber-physical architectures**

- **Correctness of dynamic behaviors**
- Impact of communication on performance

# **Design & Analysis of Cyber-Physical System Architectures**

Shang-Wen Cheng, David Garlan, Bruce Krogh, Akshay Rajhans, Bradley Schmerl and Bruno Sinopoli

#### Architectural primitives for cyber-physical systems

#### Components

- Cyber: computation, data-storage, controller, estimator
- Physical: hybrid dynamic system, physical subsystem
- Cyber-physical: transducer

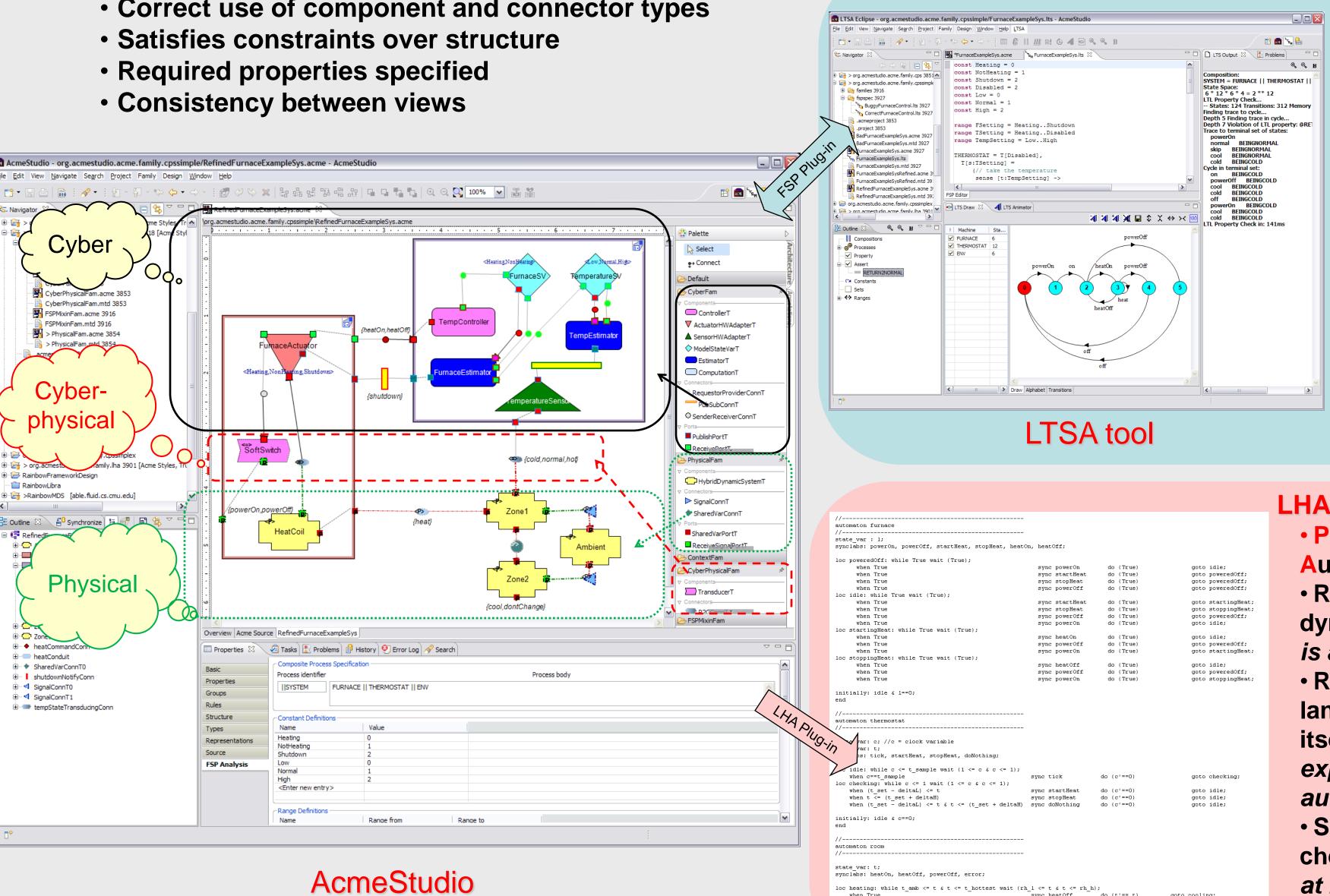
#### Connectors

- Cyber: point-to-point, publish-subscribe
- Physical: signal-flow (directed), shared-variable (undirected)
- Cyber-physical: cyber-to-physical, physical-to-cyber

#### **Architectural and behavioral analysis**

#### **Architectural** analysis

- Correct use of component and connector types

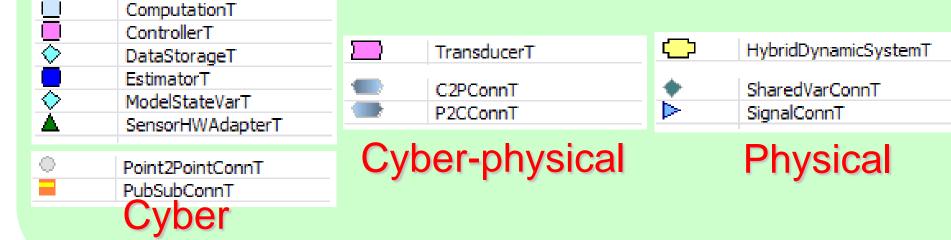






Supported by Grant: CNS0834701

### Architectural types in AcmeStudio ActuatorHWAdapterT



#### **Behavioral** analysis

#### **FSP Analysis:**

- Labelled Transition System Analyser
- Safety properties, e.g., Temperature never exceeds max value
- Liveness properties, e.g., *Temperature* eventually becomes normal whenever it gets cold
- Protocol checking, e.g., deadlock-freedom

| powerOff, startHeat, stopHeat, heatC | n, heatOff;                     |                          |                                |  |
|--------------------------------------|---------------------------------|--------------------------|--------------------------------|--|
| le True wait (True);                 |                                 |                          |                                |  |
|                                      | sync powerOn                    | do {True}                | goto idle;                     |  |
|                                      | sync startHeat                  |                          | goto poweredOff;               |  |
|                                      | sync stopHeat                   | do (True)                | goto poweredOff;               |  |
| e wait (True);                       | sync powerOff                   | do {True}                | goto poweredOff;               |  |
| c ware (Irac),                       | sync startHeat                  | do {True}                | goto startingHeat;             |  |
|                                      | sync stopHeat                   | do (True)                | goto stoppingHeat;             |  |
|                                      | sync powerOff                   | do (True)                | goto poweredOff;               |  |
|                                      | sync powerOn                    | do {True}                | goto idle;                     |  |
| hile True wait (True);               | sync heatOn                     | do {True}                | goto idle;                     |  |
|                                      | sync neacon<br>sync powerOff    |                          | goto fale;<br>goto poweredOff; |  |
|                                      | sync powerOn                    | do (True)                | goto startingHeat;             |  |
| hile True wait {True};               |                                 |                          |                                |  |
|                                      | sync heatOff                    | do (True)                | goto idle;                     |  |
|                                      | sync powerOff                   | do (True)                | goto poweredOff;               |  |
|                                      | sync powerOn                    | do {True}                | goto stoppingHeat;             |  |
| ==0;                                 |                                 |                          |                                |  |
|                                      |                                 |                          |                                |  |
|                                      |                                 |                          |                                |  |
|                                      |                                 |                          |                                |  |
|                                      |                                 |                          |                                |  |
|                                      |                                 |                          |                                |  |
| clock variable                       |                                 |                          |                                |  |
| rtHeat, stopHeat, doNothing;         |                                 |                          |                                |  |
| reneat, scopneat, dowoening,         |                                 |                          |                                |  |
| = t_sample wait {1 <= c & c <= 1};   |                                 |                          |                                |  |
|                                      | sync tick                       | do {c'==0}               | goto checking;                 |  |
| c <= 1 wait {1 <= c & c <= 1};       | -                               |                          |                                |  |
| eltaL) <= t<br>t + deltaH)           | -                               | do {c'==0}<br>do {c'==0} | goto idle;<br>goto idle;       |  |
| eltaL) <= t & t <= (t_set + deltaH)  | sync stopHeat<br>sync doNothing | do (c'==0)               | goto idle;                     |  |
|                                      | - ]                             | (/                       | <u> </u>                       |  |
| ==0;                                 |                                 |                          |                                |  |
|                                      |                                 |                          |                                |  |
|                                      |                                 |                          |                                |  |
|                                      |                                 |                          |                                |  |
|                                      |                                 |                          |                                |  |
|                                      |                                 |                          |                                |  |
|                                      |                                 |                          |                                |  |
| eatOff, powerOff, error;             |                                 |                          |                                |  |
| t_amb <= t & t <= t_hottest wait (rh | 1 <= t & t <= rh ]              | n);                      |                                |  |
|                                      | sync heatOff                    | do {t'== t}              | goto cooling;                  |  |
|                                      | anna novorOff                   | do (+1+)                 | goto cooling:                  |  |
|                                      |                                 |                          |                                |  |
|                                      |                                 |                          |                                |  |
| PHAVer code                          |                                 |                          |                                |  |
|                                      |                                 | JUC                      |                                |  |
|                                      |                                 |                          |                                |  |

#### LHA Analysis

 Polyhedral Hybrid **Automaton Verifyer** • Richer set of hybrid dynamics, e.g., *Temperature* is a continuous variable Richer specification language - specification itself can be an LHA, e.g., expected\_behavior automaton Simulation relation checking, e.g., system does at least as much as what is required by the expected\_ behavior automaton