Modeling the Mission Data System Reference Architecture in AADL

April 27, 2009


Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

California Institute of Technology

Contents / Agenda

Modeling of MDS

End-to-end latency analysis of control loops

Modeling and analysis of the goal network

The Mission Data System

A reference architecture

- To be instantiated for different applications

An embedded systems architecture

- Consists of physical system, computing hardware, application software

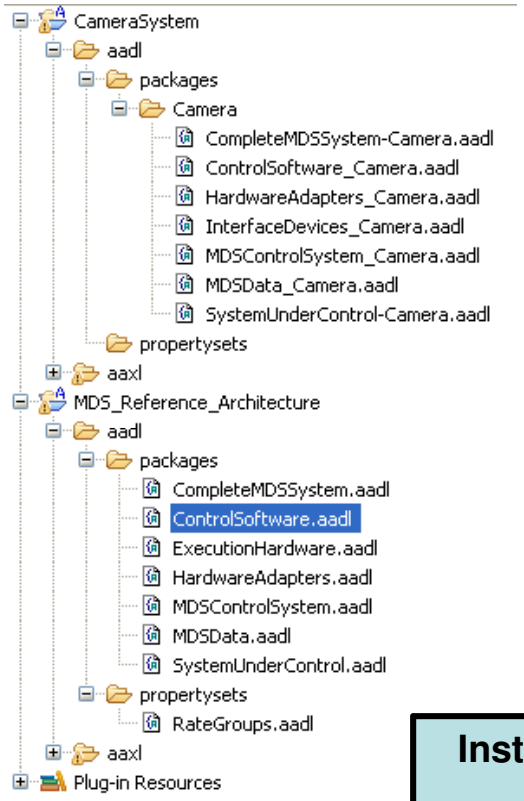
A control systems architecture

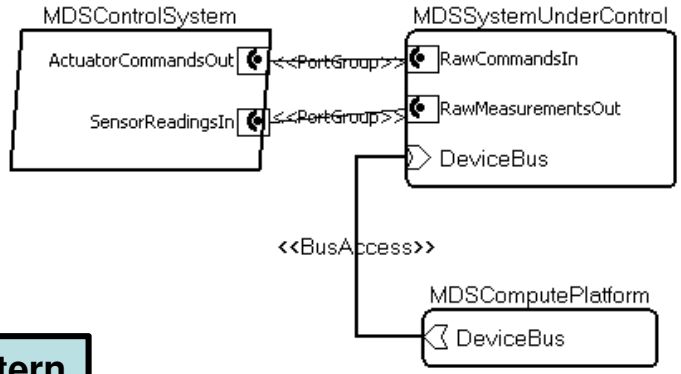
- Feedback loops in application architecture
- Feedback loops in data management system

A multi-layered architecture

- From low-level control loops to goal-oriented planning and plan execution

A Reference Architecture





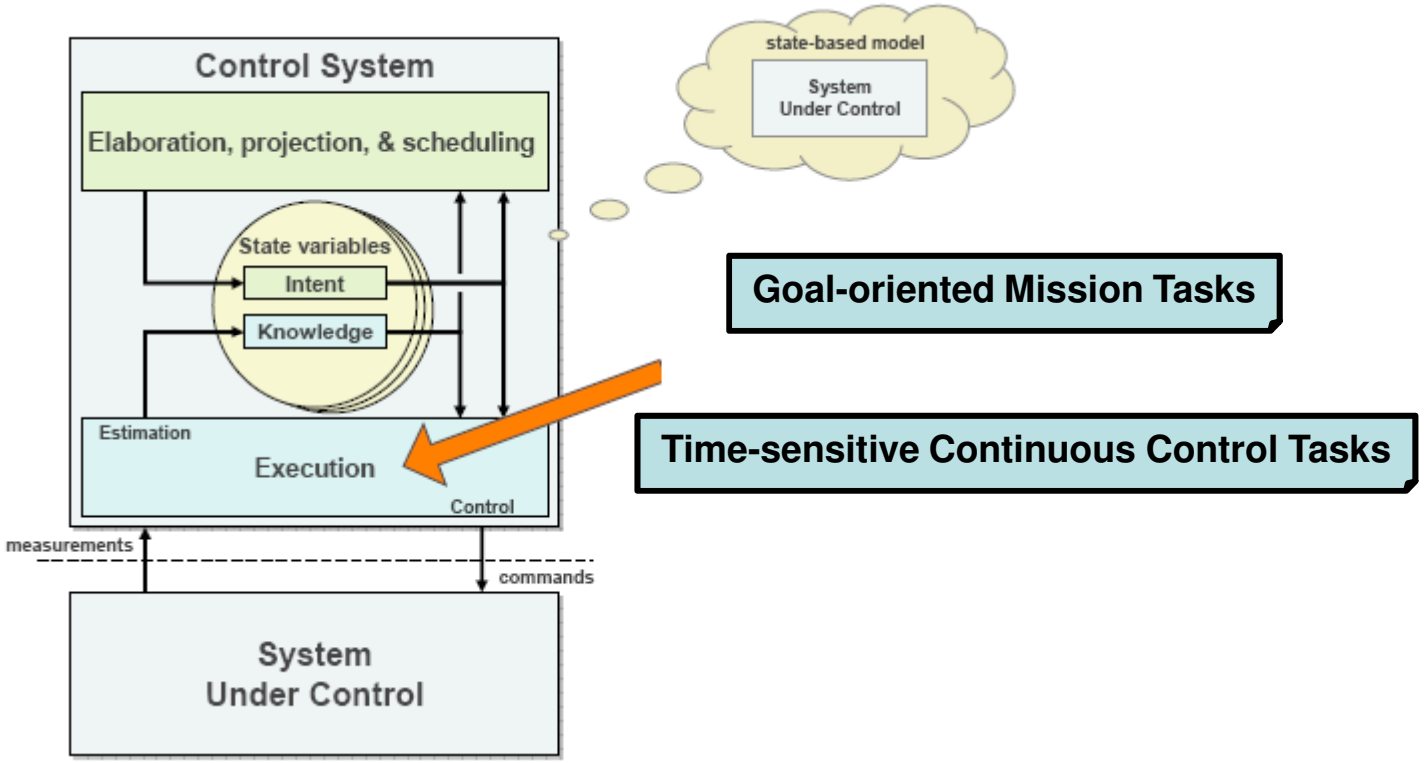
Generic Architecture Pattern

With Connection Topology

Instantiation of Application Architecture

Computing Platform, and Physical System

Mission and Control Processing



Model of the MDS Application Control System

Excerpt from the Textual Specification:

process implementation MDSControlSystem.basic

subcomponents

GoalPlanner: **thread group** ControlSoftware::GoalPlanner;

GoalExecutive: **thread group** ControlSoftware::GoalExecutive;

GoalMonitor: **thread group** ControlSoftware::XGoalMonitor;

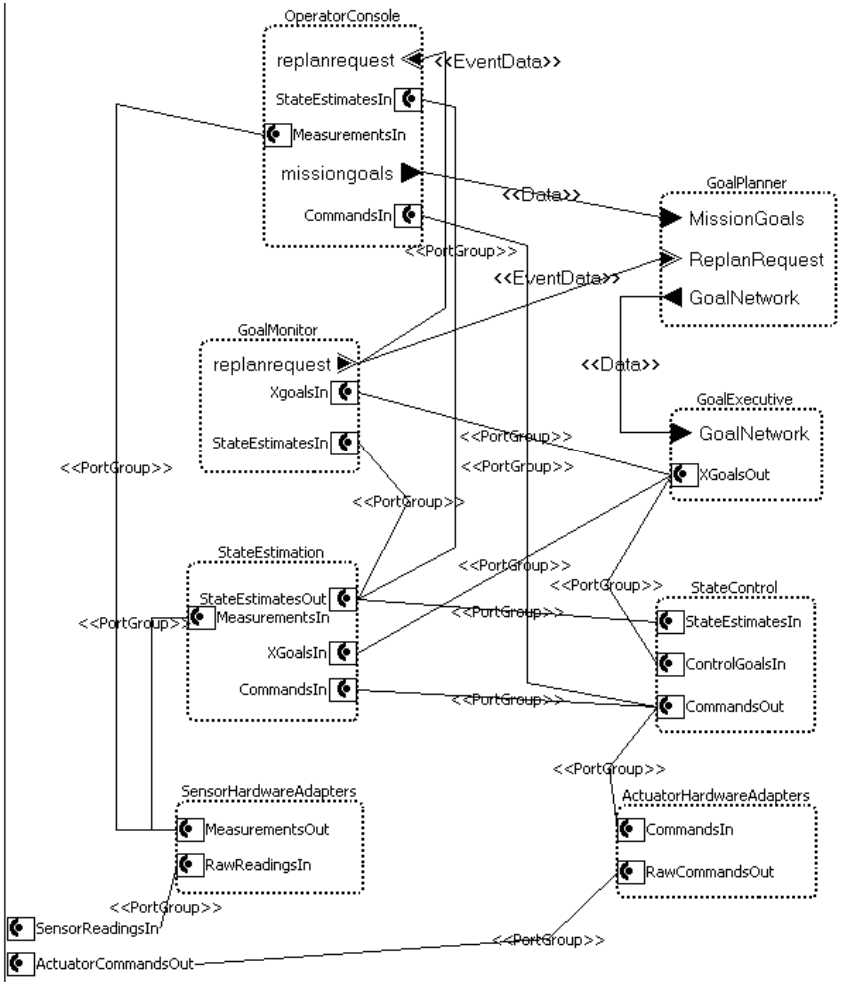
StateEstimation: **thread group** ControlSoftware::estimator;

StateControl: **thread group** ControlSoftware::controller;

OperatorConsole: **thread group** ControlSoftware::OperatorConsole;

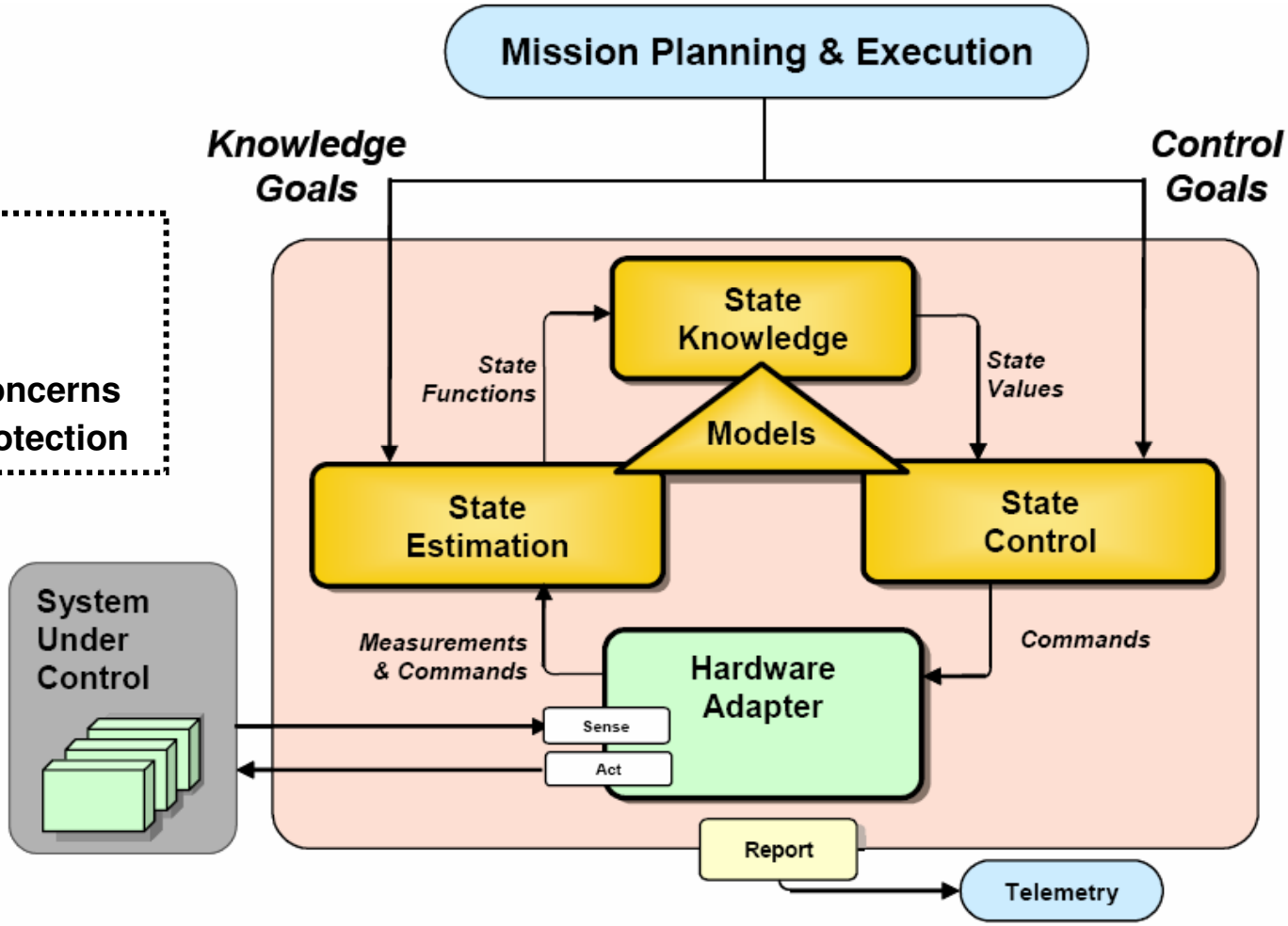
Textual & Graphical Representations

Focus on Information Flow



Mission Data System (MDS) Architecture*

- Closed loop
- Goal-Directed
- Explicit models
- Separation of Concerns
- Integral Fault Protection



* M. Bennett, R. Knight, R. Rasmussen, M. Ingham, "State-Based Models for Planning and Execution, 2006-08-11.

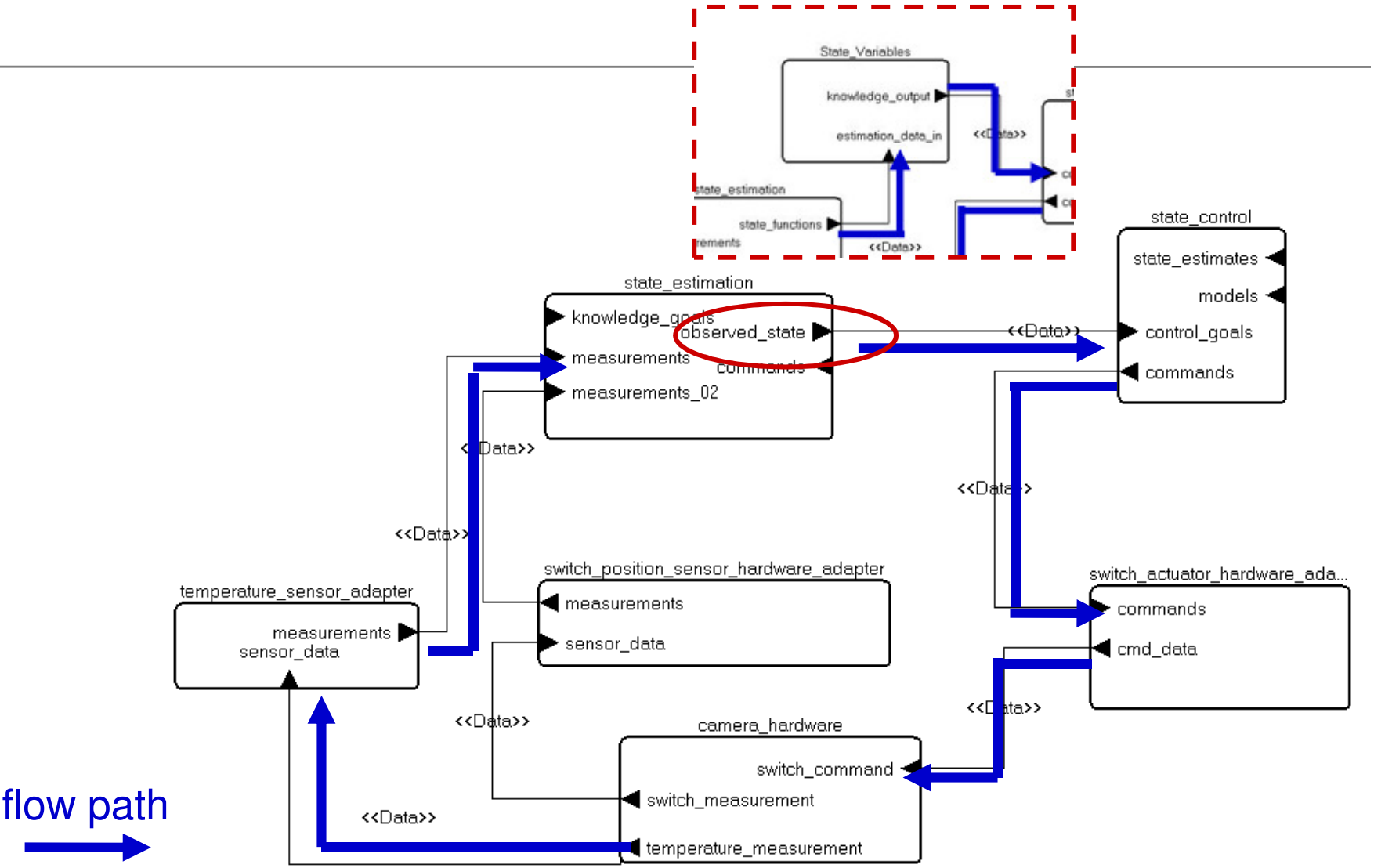
Separation of Concerns: Data Management System

Role of state history and state variables

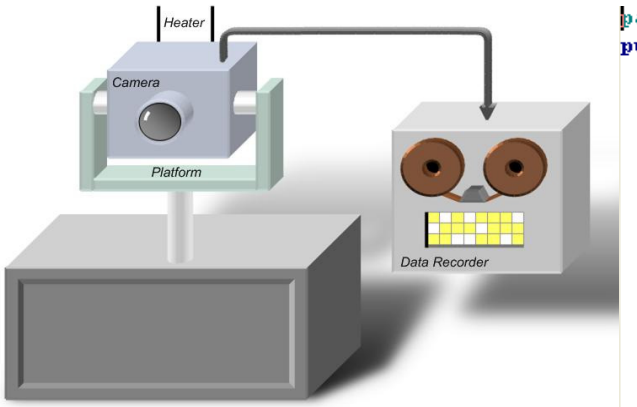
- Information flow through application architecture
  - Access to multiple data stream values
- Historical log of data stream for post mortem analysis
  - Storage management through compression
- Distributed processing between space and ground system
  - Proxies & telemetry

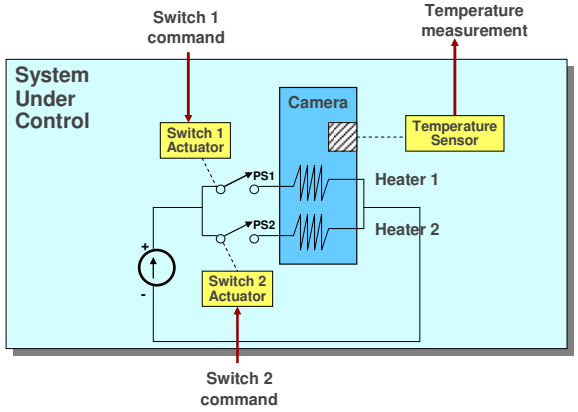
Managed data history variables

Flow-Oriented Model of Temperature Control



Reference Architecture Instantiation






```
package ControlSoftware::Camera
public
  thread group estimator
    extends ControlSoftware::estimator
    features
      StateEstimates: refined to port group MDSData::Camera::StateVariables;
      measurements: refined to port group MDSData::Camera::RequiredMeasurements;
      PreviousCommands: refined to port group MDSData::Camera::RequestedCommands;
    flows
      StateFlow: flow path measurements -> StateEstimates;
  end estimator;

  thread group implementation estimator.camera
    subcomponents
      TemperatureEstimator: thread TemperatureEstimator;
      TemperatureSensorHealthEstimator: thread TemperatureSensorHealthEstimator;
      HeaterSwitchEstimator: thread HeaterSwitchEstimator;
    connections
```

Instantiation of reference architecture through refinement of AADL model

Deployment on different computing hardware platforms

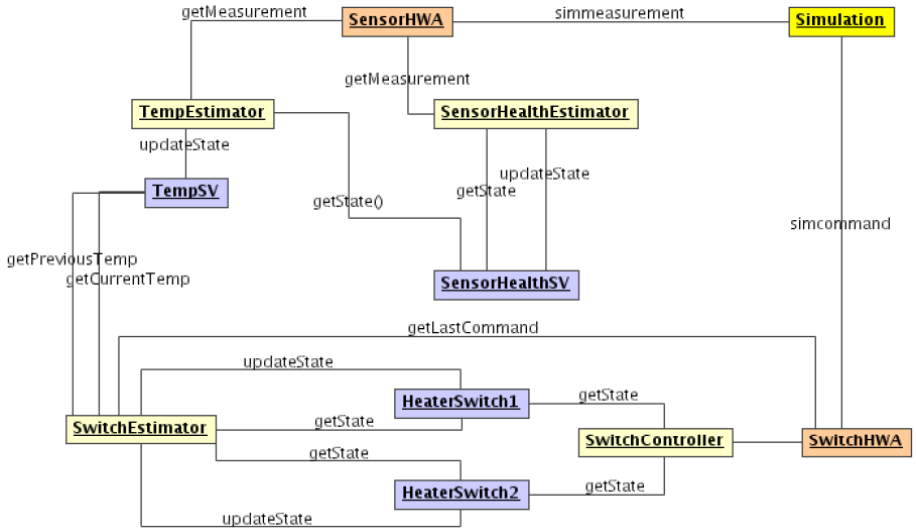
Contents / Agenda

Modeling of MDS

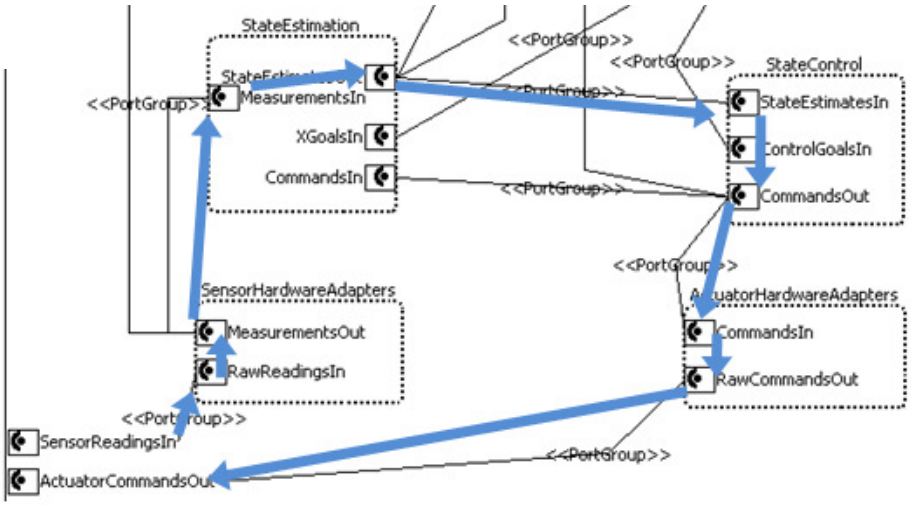
End-to-end latency analysis of control loops

Modeling and analysis of the goal network

Temperature Control AADL Representation



Use of immediate & delayed connections to achieve deterministic sampling



flow path

Latency Contributors

Control engineering view:

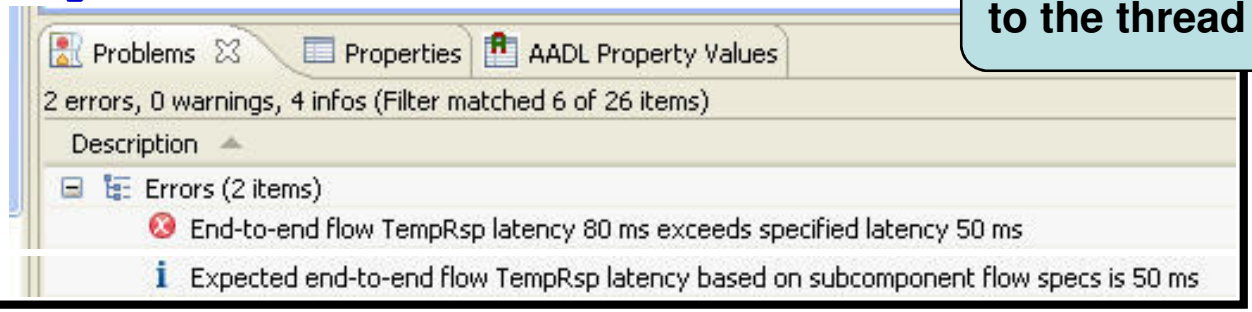
- Processing latency, sampling latency, physical signal latency

Embedded software systems engineering view:

- Preemption, processor speed, resource contention, communication delay, rate group optimization, partitioned architecture, migration of functionality

Transport Latency Analysis Results

Analysis Results*:



Analysis can be extended to the thread level

Excerpt from the Textual Specification:*

```
flows
TempRsp: end to end flow camera_hardware.TempRsp1 -> DC02
 -> temperature_sensor_adapter.TempRsp -> DC04 -> state_estimation.TempRsp
 -> DC07 -> State_Variables.TempRsp -> DC08 -> state_control.TempRsp
 -> DC06 -> switch_actuator_hardware_adapter.TempRsp
  -> DC03 -> camera_hardware.TempRsp {latency => 50 ms;};
```

```
flows
   TempRsp: flow path control_goals -> commands {Latency => 20 ms;};
```

```
flows
TempRsp: flow sink switch_command -> DataConnection1
 -> switch_actuator.TempRsp;
TempRsp1: flow source temperature_sensor.TempRsp
->  DataConnection5 -> temperature_measurement;
```

** Note that illustrative values are used for this model and the results are not indicative of the results for any existing MDS implementation.*

Contents / Agenda

Modeling of MDS

End-to-end latency analysis of control loops

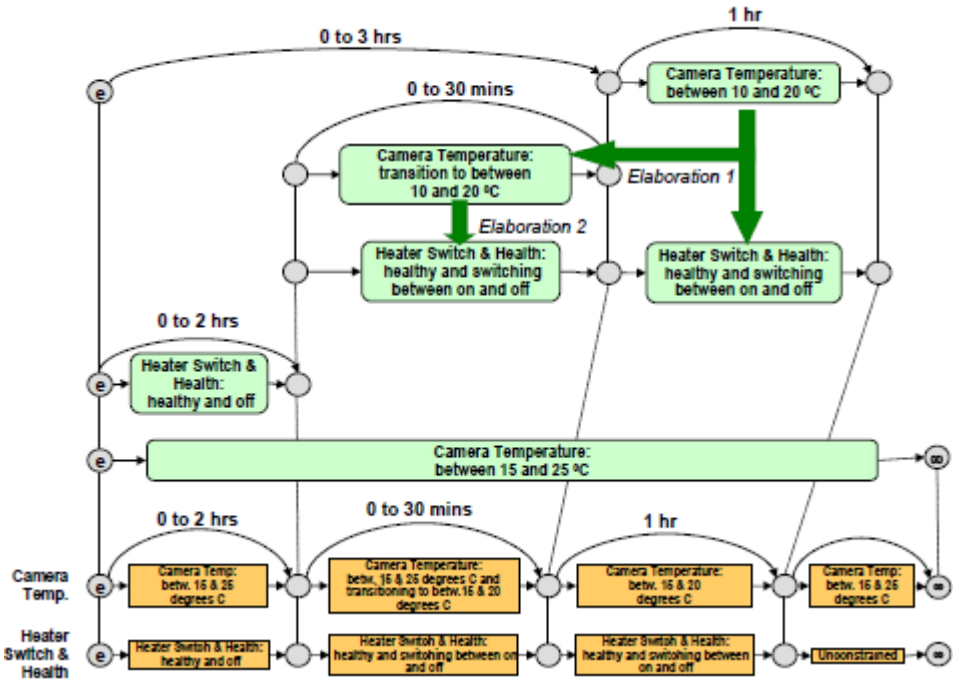
Modeling and analysis of the goal network

MDS Mission Planning & Plan Execution

Modeling and analysis framework in place by MDS

Represent planning & plan execution tasks

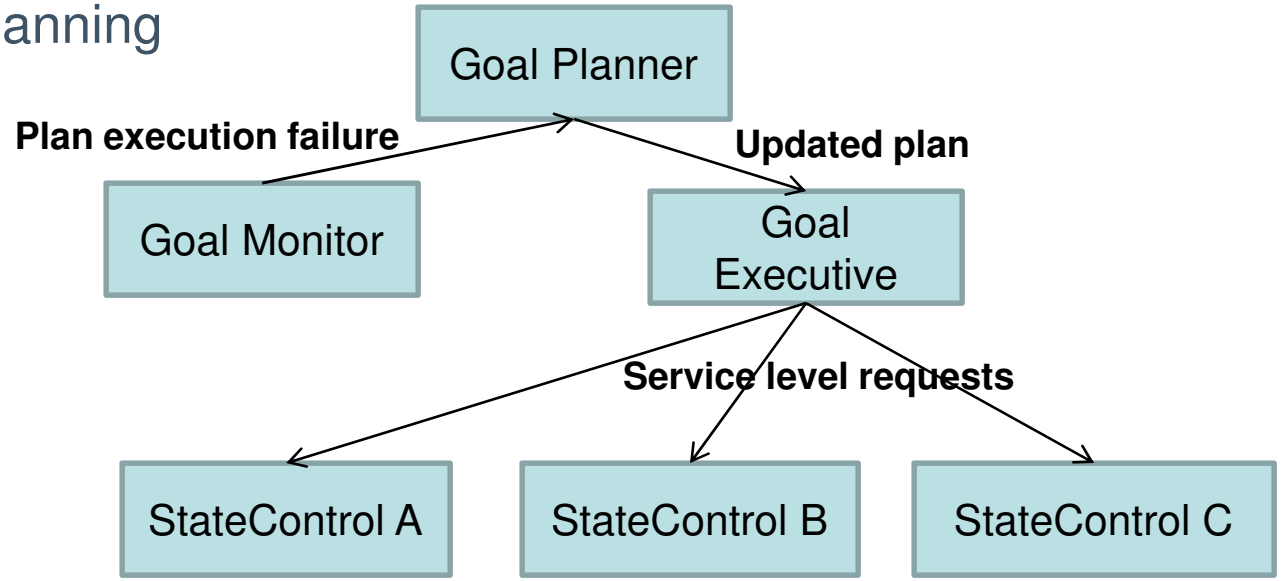
Represent goal-based fault management



Modeling of Mission Processing

Modeling of execution of goal network execution

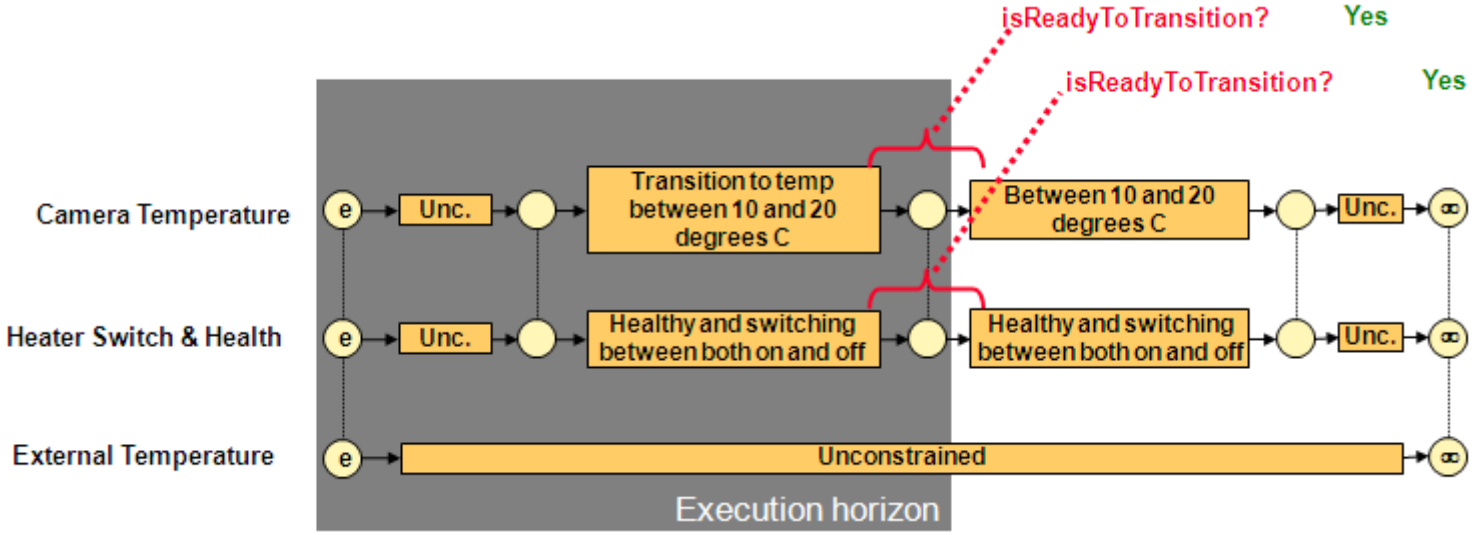
- Goal executive sends service requests to state control units
- State control units perform requests as modes incl. idle mode
- Goal monitor identifies unreachable goals (failures) & requests replanning



Workload Analysis of Goal Network

Simulated execution of goal network

- Starting set: tasks with no predecessor
- Active set between synchronization points
- Generate System Operation Mode (SOM) for instance model
- Perform mode specific scheduling analysis



Conclusion

AADL can represent reference architectures

Reference architecture can be validated

AADL supports instantiation of reference architecture for specific system

MDS case study illustrates one approach to representing planning & plan execution layer in autonomous system architecture

AADL pilot with NASA/JPL Juno project

AADL training & model development workshop at JAXA

JPL/JAXA interaction on AADL

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this presentation is not intended in any way to infringe on the rights of the trademark holder.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.