ISWHM: Tools and Techniques for Software and System Health Management

Johann Schumann, SGT, Inc, NASA ARC
Ole J. Mengshoel, CMU, NASA ARC
Adnan Darwiche, UCLA

SHM TIM, 05/2010

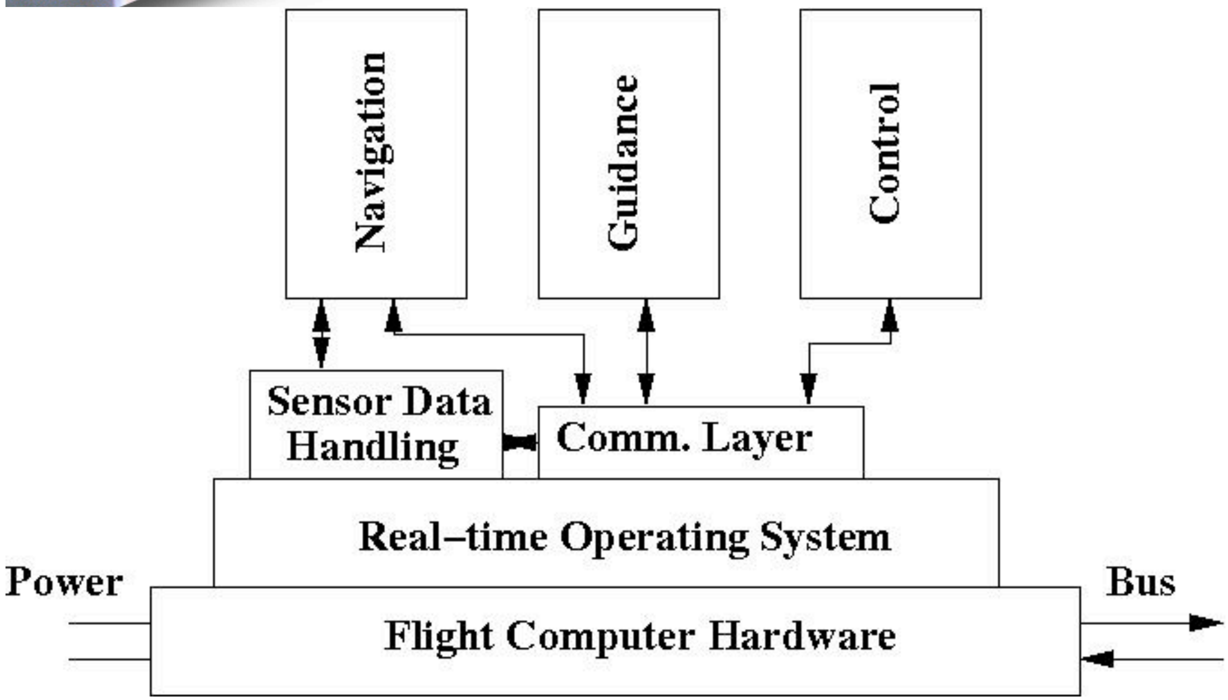
Overview

- Ingredients of a GN&C System
- Selected GN&C Testbed example
- HM of major ingredients
- ISWHM testbed architecture
- Conclusions and next Steps

GN&C

- GN&C (Guidance, Navigation, and Control) is one of the most central software system in an aircraft/spacecraft
- Guidance: "*Where do I want to go and how do I get there?*"
- Navigation: *"Where am I?"*
- Control: *"Which thrusters do I need to use to keep my attitude stable?"*

Typical GN&C Architecture



Typical architecture:

• PowerPC 750, RAM, Flash,

• IObus: MIL 1553 or CAN bus (automotive)

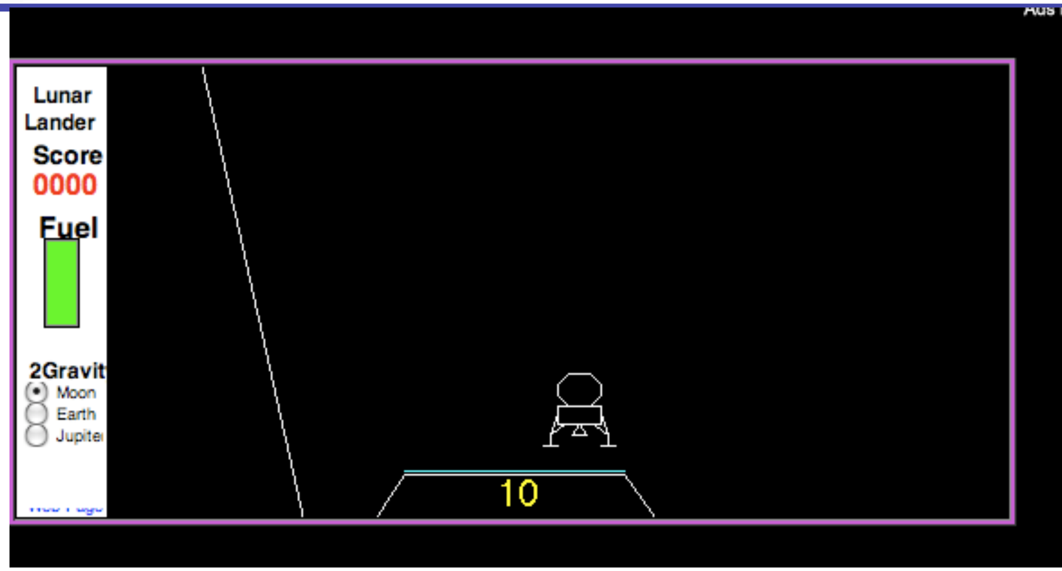
• OS: Real-time: VxWorks, RTLinux, OSEK compl.,...

- "Black box"
- run at different speeds
  - G: 2Hz
  - N: 10hz-100Hz
  - C: 100Hz
- in different processes
- use comm layer

SW Characteristics

- Are there specific software constructs used in specific components?
- These play a major role on how SWHM models will be constructed
- Typical characteristics include
  - numerical computations?
  - branches? mode logic? state machines?
  - loops?
  - complex algorithms? (e.g., optimization)
  - communications structure
  - signals

Our GN&C Testbed Example



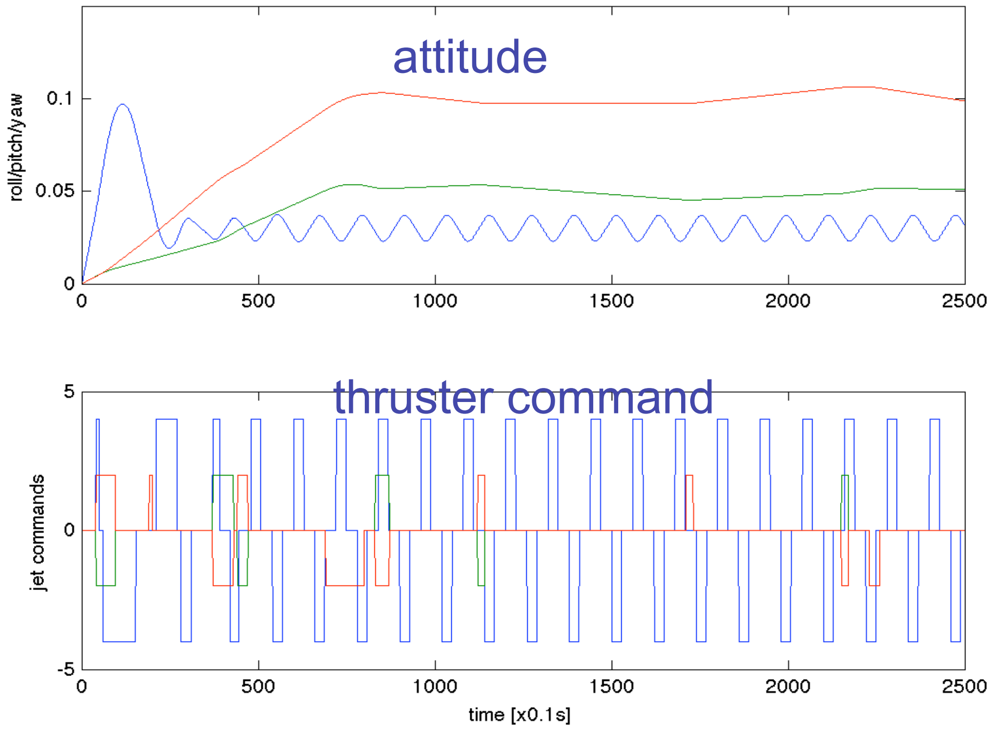
- "re-designed" Apollo lunar lander Autopilot
- non-trival GN&C example
- non-ITAR Simulink and Stateflow model
- Downloadable from Mathworks

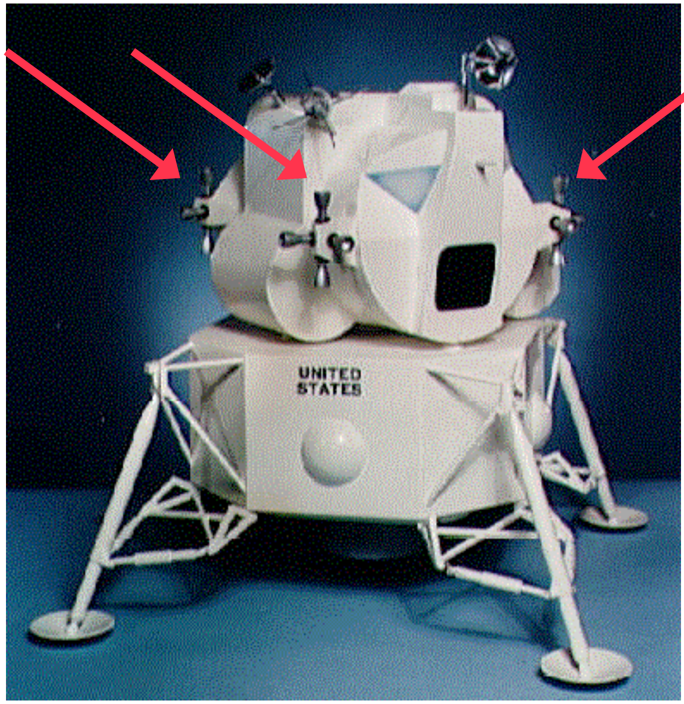
GN&C Architecture

- The selected GN&C architecture is typical for many aeronautics applications
  - guidance for autopilot functions (in particular for UAV)
  - navigation based on sensors (e.g., inertial reference unit, GPS)
  - control of actuators (mainly control surfaces)
  - implemented in software (often using model-based approaches), running on an embedded processor
- Examples of related architectures are NASA IFCS, Dryden Platform Precision AutoPilot,

Our demonstration example was selected because of realistic GN&C functionality, easy availability of model (non-ITAR and does not contain proprietary code) and straight-forward models of components not relevant for SWHM.

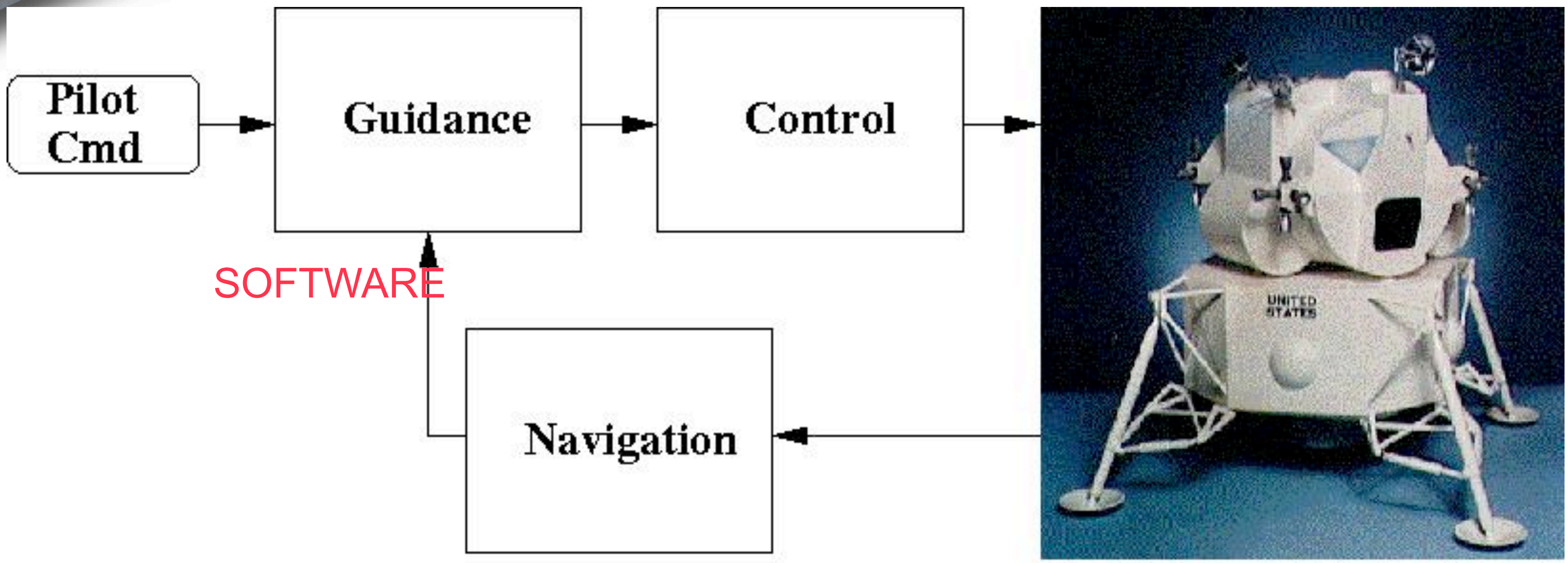
Operation





- task: from attitude (0,0,0) obtain and keep attitude (0.1,0.05,0.02)
- use the given set of control thrusters

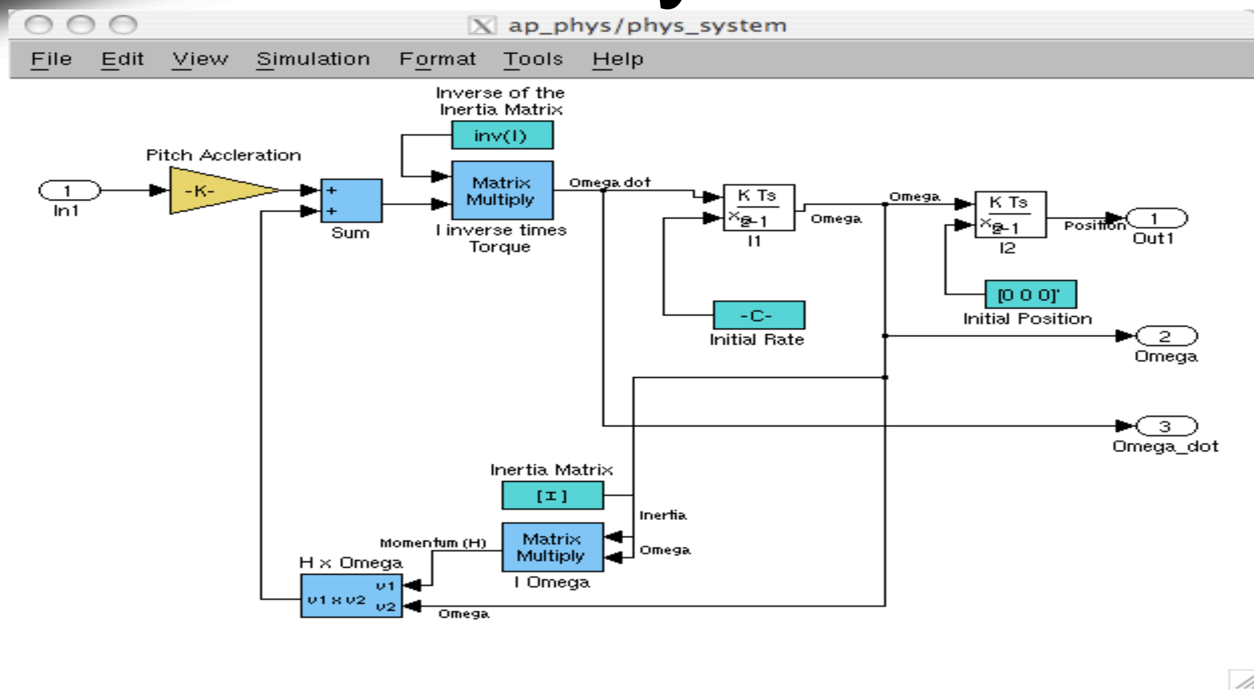
Top Level structure



plant (model)

The original example does not contain any navigational components. The "true" attitude and position is fed back to the Guidance and Controller.
We are adding some "mock-up" sensors and some navigational code.

Physical Plant Model



- is used for simulation only.
- dynamics are described using differential equations (given acceleration, calculate position, rates, and rate changes)

Pilot Command Handling

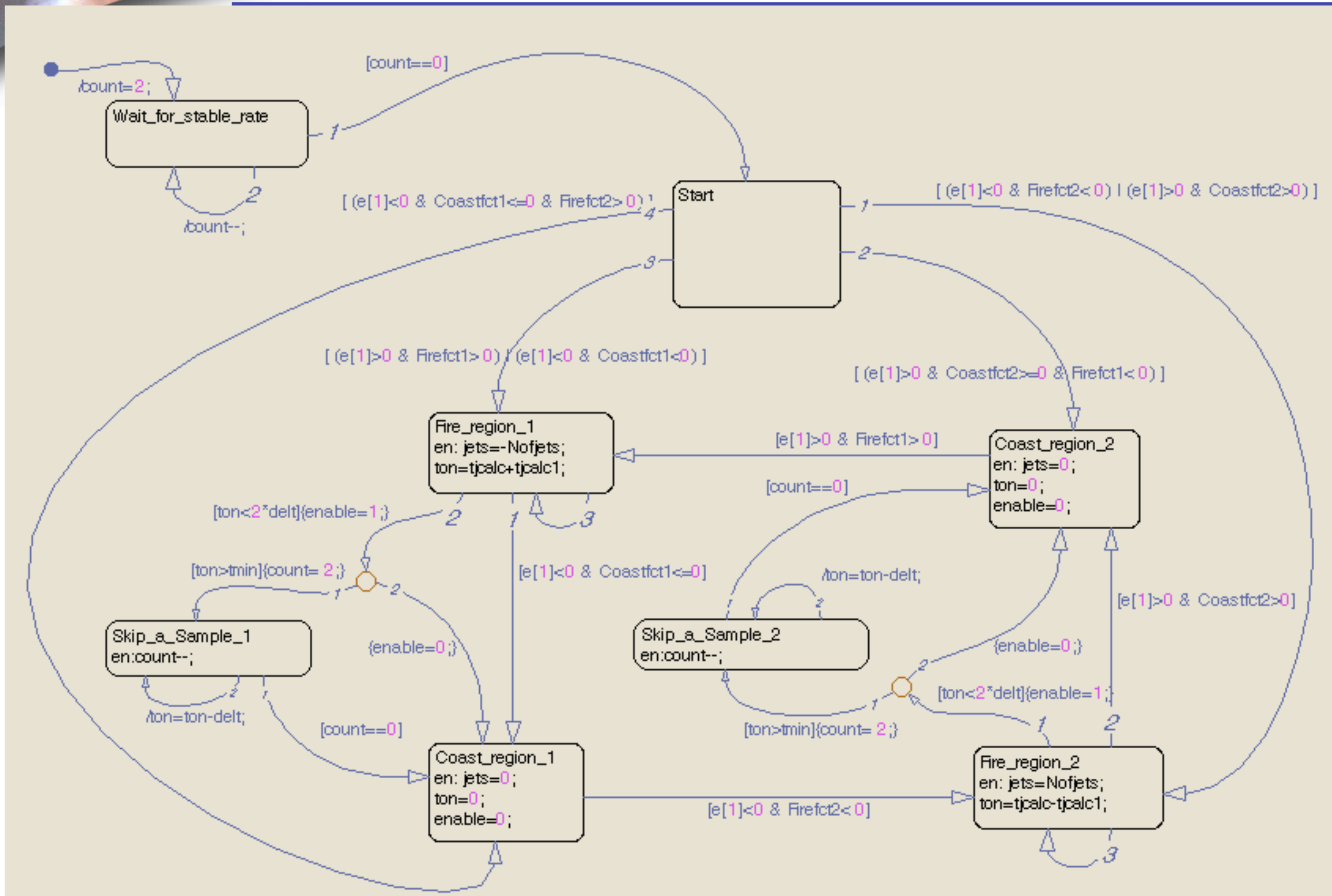
- extremely simple in this example
- only the final position/attitude is provided

- SW characteristics:
  - data communication
  - range issues
  - timing issues

Guidance

- The guidance component contains the actual algorithmic “meat”
- Given the current state and the target state, find an optimal trajectory using as little fuel as possible and other constraints
- This algorithm uses a fairly elaborate state machine that is modeled here as a Stateflow diagram

Guidance

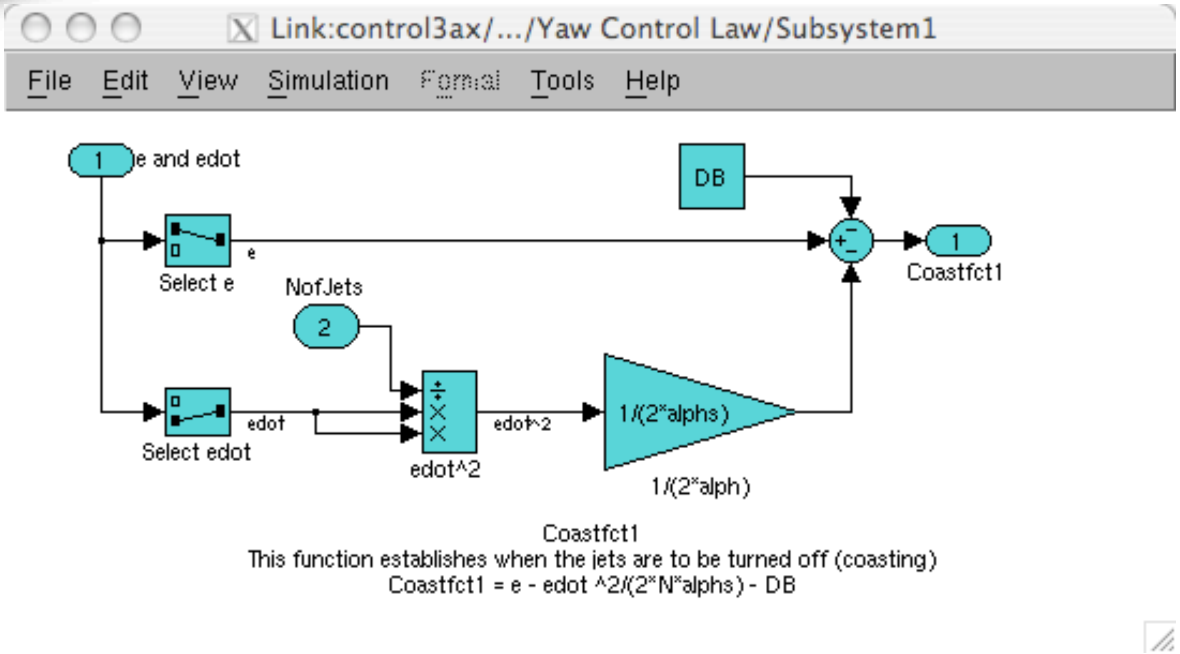


/count=2;
Wait_for_stable_rate
/count--;
[count==0]
Start
[(e[1]<0 & Coastfct1<=0 & Firefct2> 0)]
[(e[1]<0 & Firefct2< 0) | (e[1]>0 & Coastfct2>0)]
[(e[1]>0 & Firefct1> 0) | (e[1]<0 & Coastfct1<0)]
[(e[1]>0 & Coastfct2>=0 & Firefct1< 0)]
Fire_region_1
en: jets=-Nofjets;
ton=tjcalc+tjcalc1;
[e[1]>0 & Firefct1> 0]
Coast_region_2
en: jets=0;
ton=0;
enable=0;
[count==0]
[ton<2*delt]{enable=1;}
[ton>tmin]{count= 2;}
[e[1]<0 & Coastfct1<=0]
/ton=ton-delt;
Skip_a_Sample_1
en:count--;
{enable=0;}
Skip_a_Sample_2
en:count--;
[e[1]>0 & Coastfct2>0]
{enable=0;}
[ton<2*delt]{enable=1;}
/ton=ton-delt;
[count==0]
[ton>tmin]{count= 2;}
Coast_region_1
en: jets=0;
ton=0;
enable=0;
[e[1]<0 & Firefct2< 0]
Fire_region_2
en: jets=Nofjets;
ton=tjcalc-tjcalc1;

Guidance

- SW characteristics
  – mainly discrete logic
    - if-then-else's,
    - state machines, ...
  – internal state variables

Control



- (mathematical) relation between sensors and actuators
- Characteristics: arithmetic code, delays, parameters. Usually very little SW branches

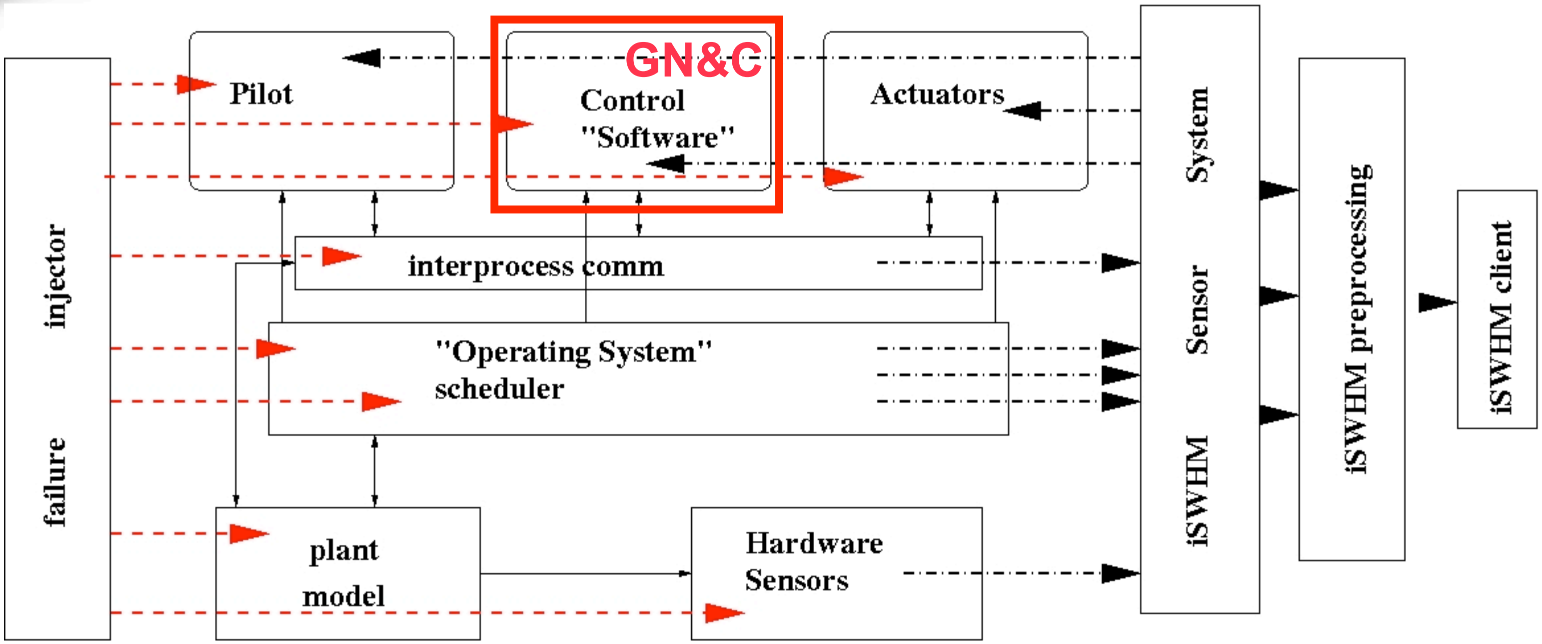
Navigation

- tries to estimate state of the vehicle given (noisy) sensor readings
- SW characteristics
  - Signal flow architecture
  - Kalman filters (recursive least square)
  - coordinate conversions
  - float-point arithmetic and matrix operations
  - few but important if-then-elses (e.g., to reset a diverging Kalman filter)

Injected Failures

- This model can be “broken” at multiple parts
  - broken or noisy sensors/actuators
  - singularities in navigation (“crossing the date line”, see F22 Raptors flying to Japan)
  - logic errors in the guidance state machine
  - communication between G,N,C SW components
  - OS “problems”: timing, stack, memory, ...

Our ISWHM Testbed



- Prototype testbed architecture to run the example software with/without failures and gather information from HW and SW sensors for further processing by the ISWHM reasoner (ISWHM server, not shown)

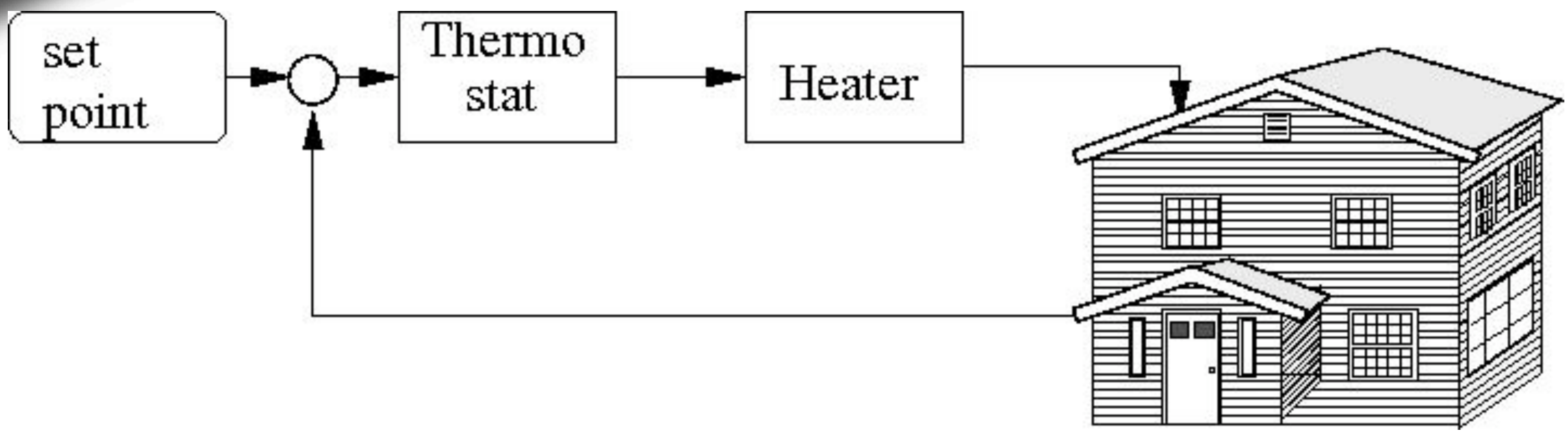
SW Characteristics

	G	N	C
mode logic	**Y**	N	N
numerical	N	**Y**	**Y**
coordinates	N	**Y**	N
state machine	**Y**	N	N
control loop	N	N	**Y**
Kalman Filter	N	**Y**	N
Algorithm	**Y**	Y	N
Comm/OS	Y	Y	Y
Update	slow	med	fast

Analysis of Feedback

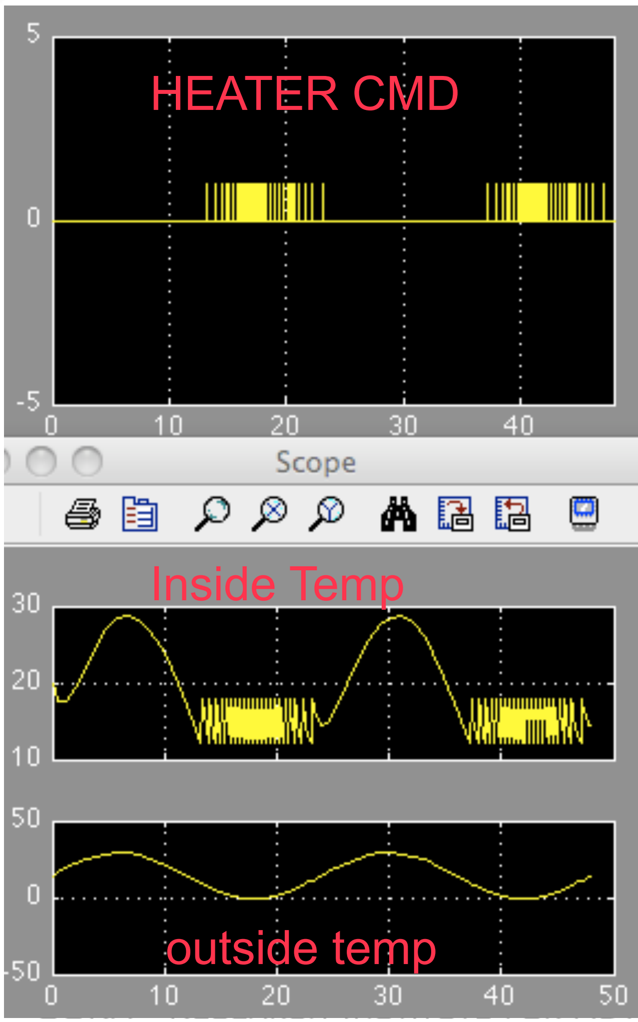
- (Feedback) loops are important for all kinds of iterative update
  - feedback control
  - iterative loops (for, while)
  - Kalman filters (Navigation)
  - optimization (while converging,...)

Simple Thermostat



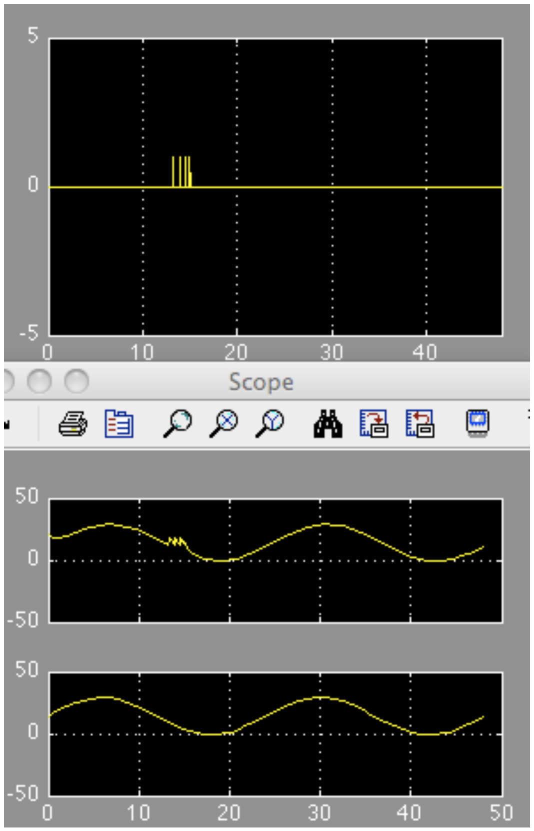
- Thermostat is a simplest possible feedback control loop.
  - combines arithmetic/calculation with feedback
- Easy to understand, several ways to inject failures
- Similar to a highly simplified aircraft control system, where the heater could be a control surface and the sensed temperature corresponds to, e.g., a roll-rate sensor.

Operation



- outside temp is a sin curve (bottom curve)
- desired temp=15deg (constant)
- controller: with threshold
- shown:
  - heater command (on/off)
  - inside temperature
  - outside temperature
- nominal case

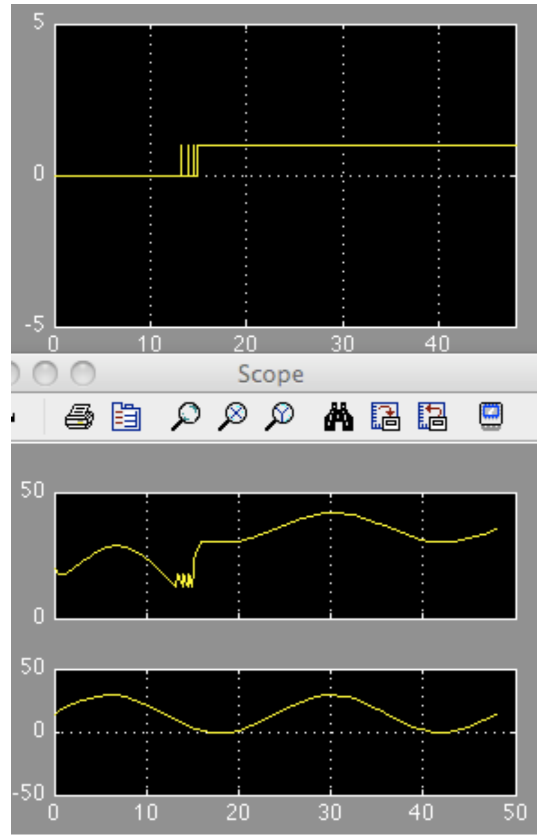
Off-nominal



HEATER CMD

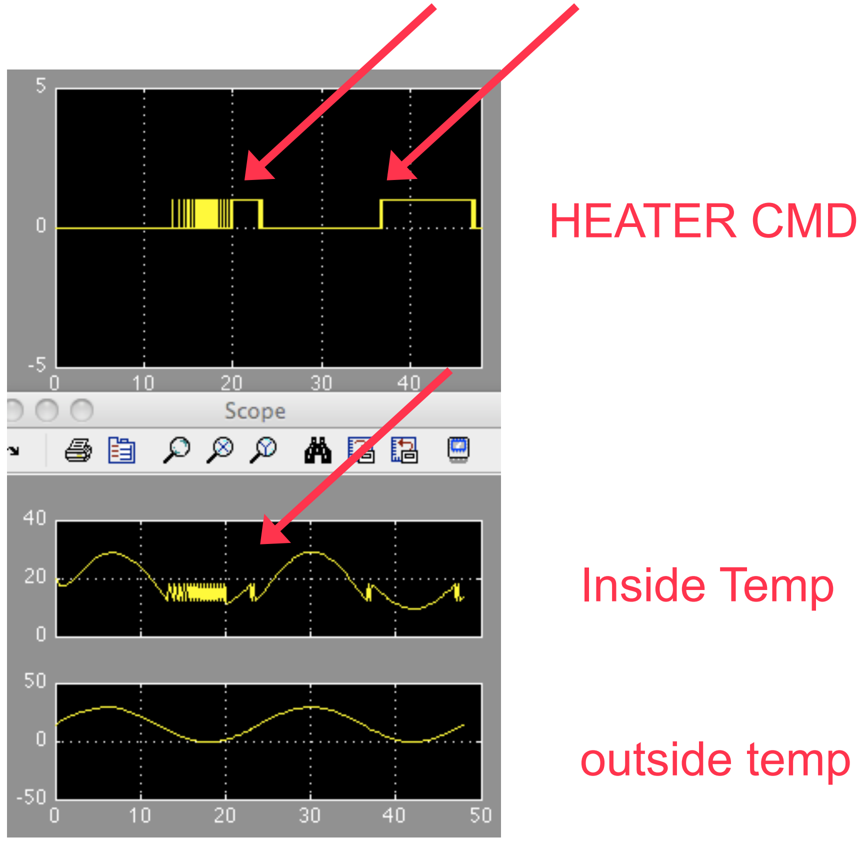
Inside Temp

outside temp



- controller stuck open and close at t=15

Off-nominal II

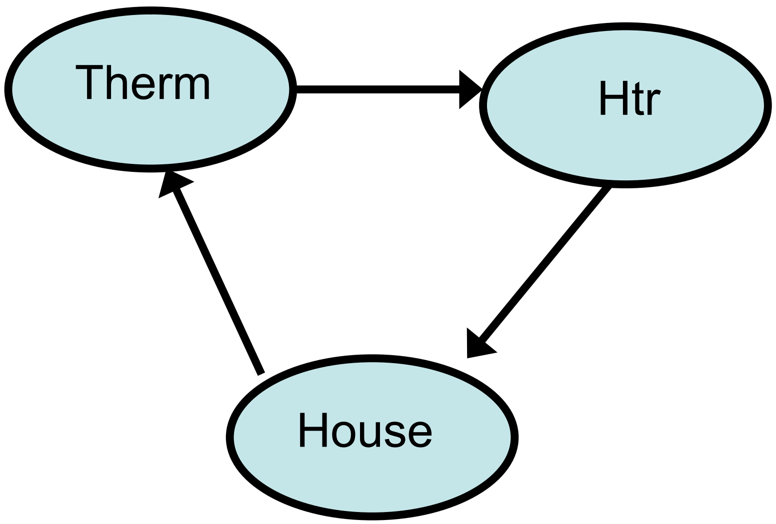


- door remains open at t=20

Observables

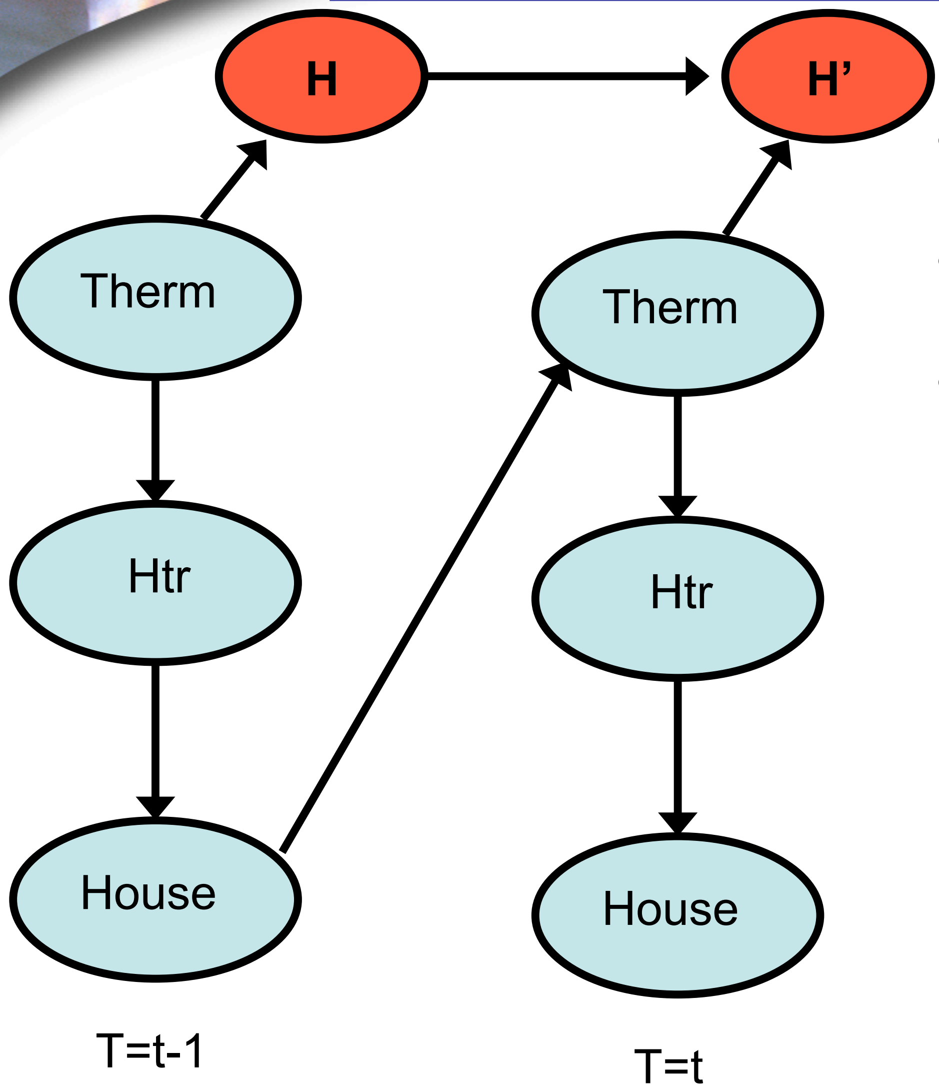
- observables are time-series data
  - room temperature
  - heater on/off

Modeling Approaches



- translation of loop into a Bayes Network
- naive model is not a DAG
- BN must talk and reason about time series
- we experiment with several modeling approaches also using dynamic Bayes Nets

Dynamic BN

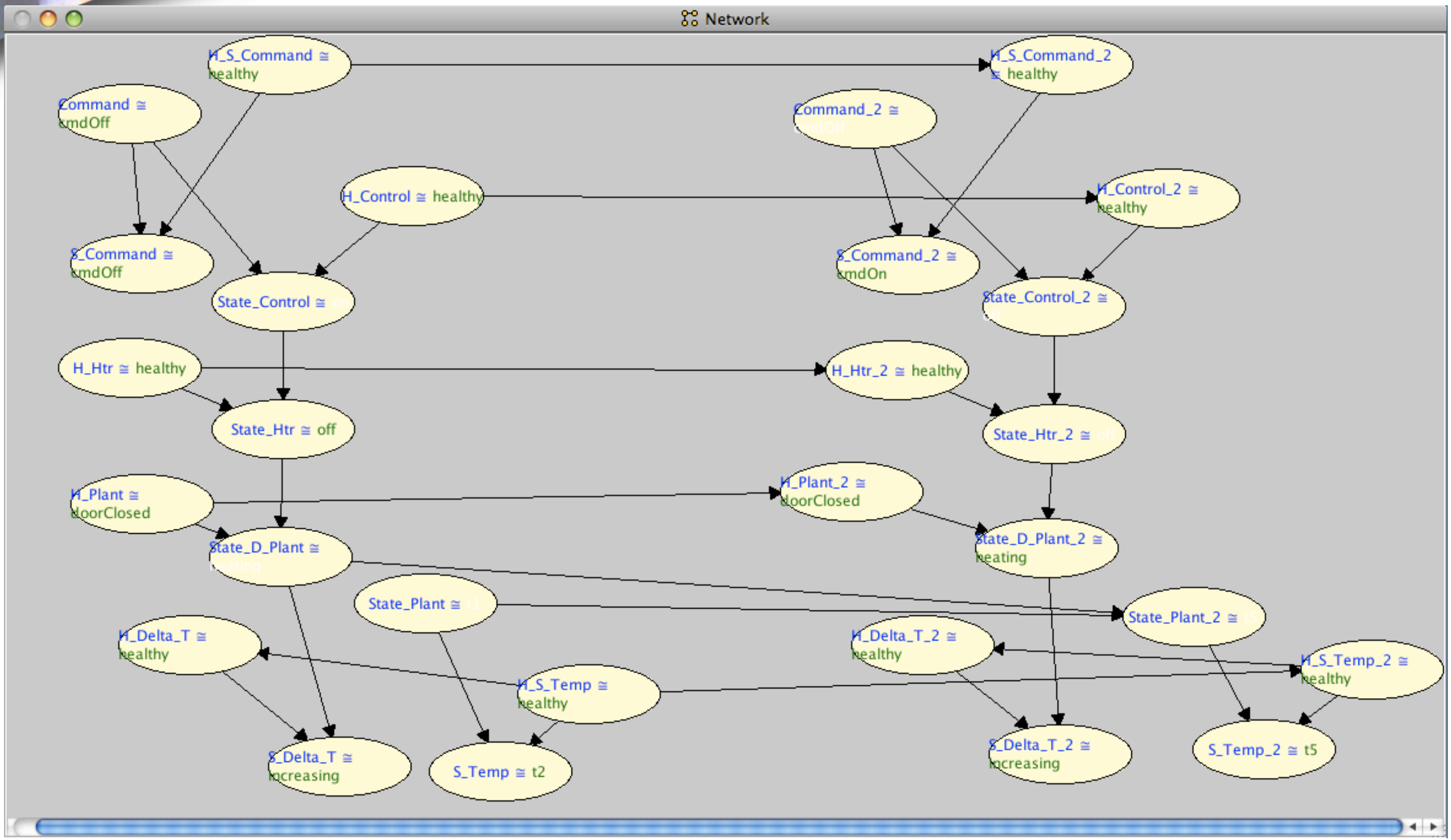


- temporal break-up
- adding Sensor nodes
- adding Health nodes

	HtrOK		HtrBAD	
	CMD=		CMD=	
Htr	Off	On	Off	On
Off	**0.99**	0.01	**0.99**	**0.9**
On	0.01	**0.99**	0.01	0.1

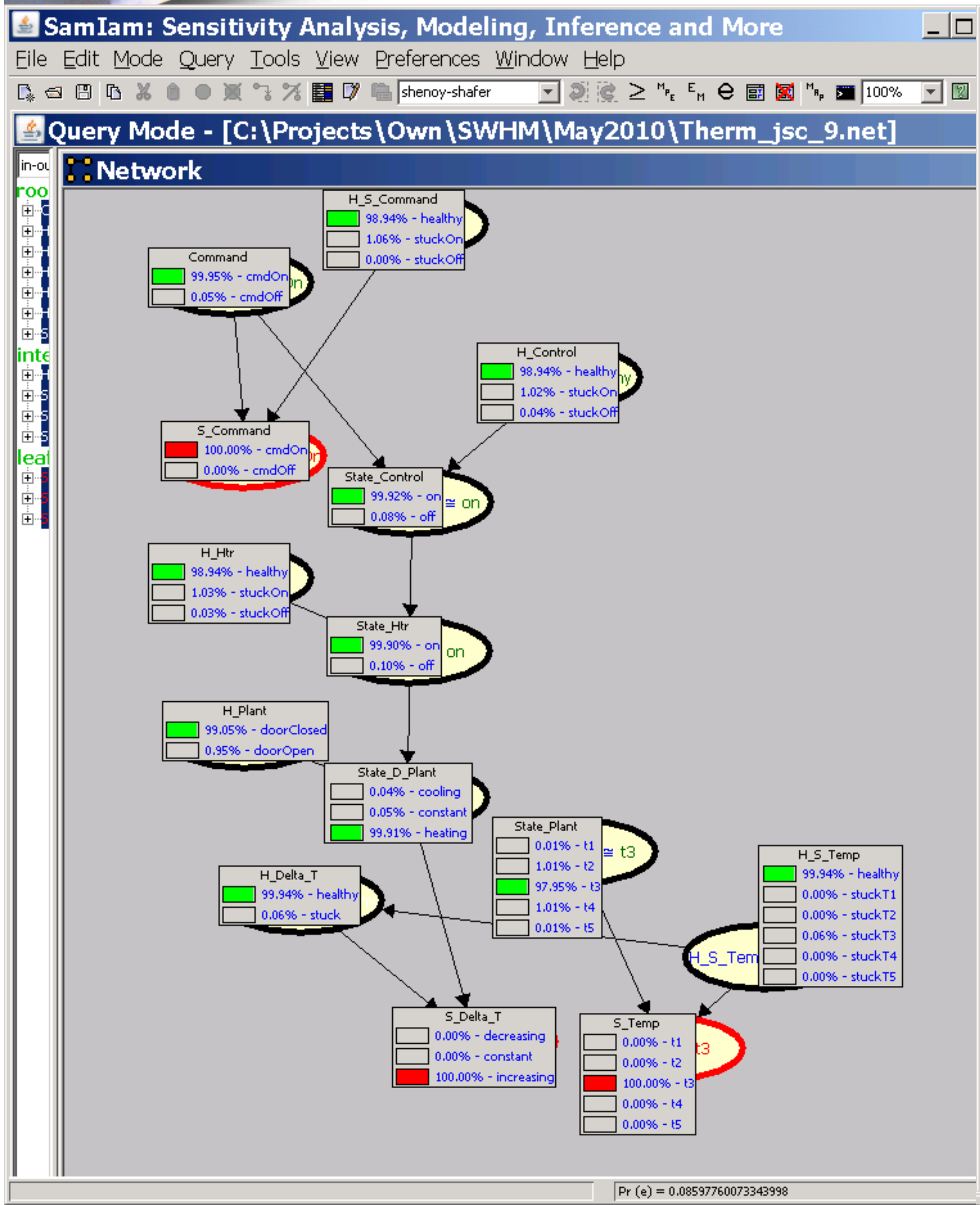
local CPT (Conditional Probability Table)

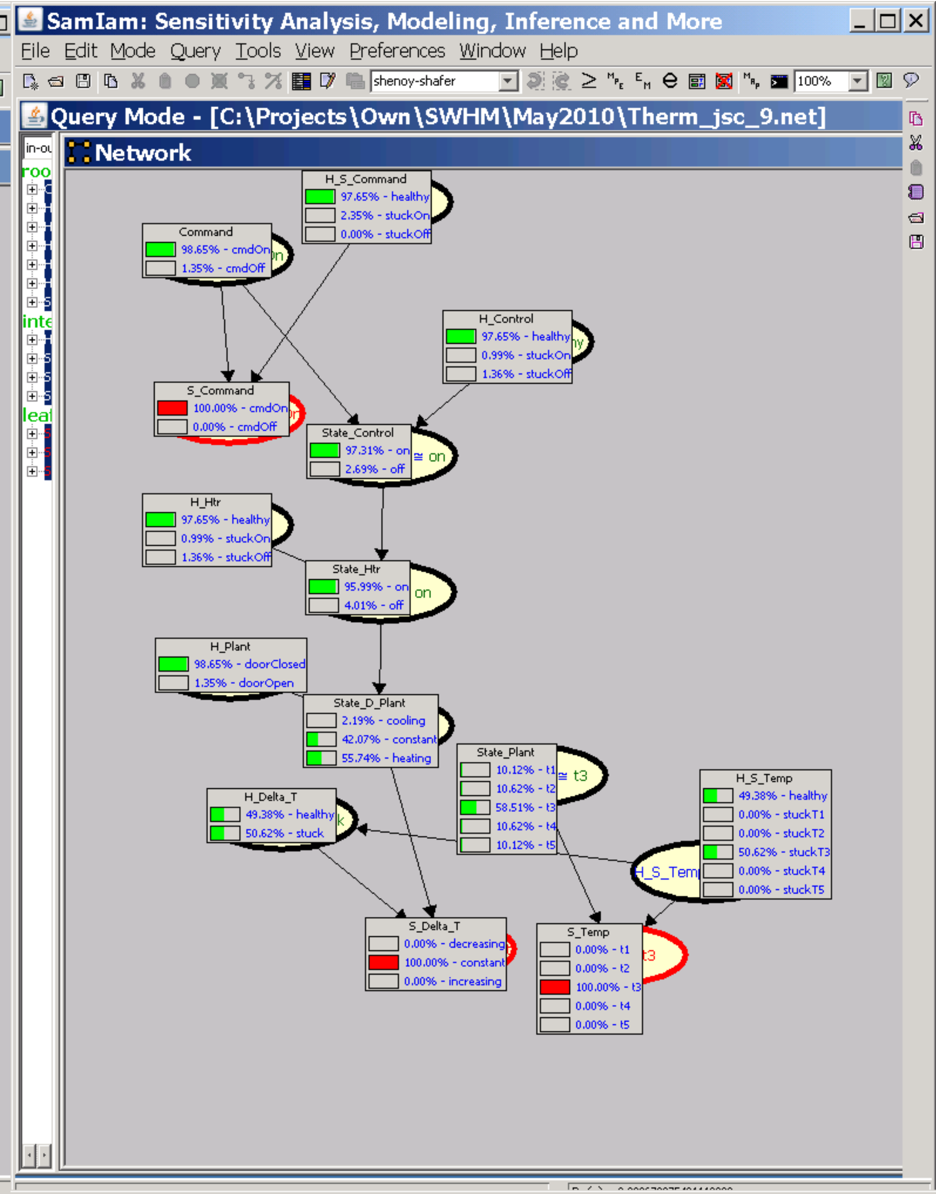
The full BN



Network
H_S_Command ≅ healthy
Command ≅ cmdOff
H_Control ≅ healthy
S_Command ≅ cmdOff
State_Control ≅
H_Htr ≅ healthy
State_Htr ≅ off
H_Plant ≅ doorClosed
State_D_Plant ≅
State_Plant ≅
H_Delta_T ≅ healthy
H_S_Temp ≅ healthy
S_Delta_T ≅ increasing
S_Temp ≅ t2
H_S_Command_2 ≅ healthy
Command_2 ≅
H_Control_2 ≅ healthy
S_Command_2 ≅ cmdOn
State_Control_2 ≅
H_Htr_2 ≅ healthy
State_Htr_2 ≅
H_Plant_2 ≅ doorClosed
State_D_Plant_2 ≅ heating
State_Plant_2 ≅
H_Delta_T_2 ≅ healthy
H_S_Temp_2 ≅ healthy
S_Delta_T_2 ≅ increasing
S_Temp_2 ≅ t5

Nominal/Off-nominal





Conclusions/Next Steps

- SWHM has to take onto specific SW characteristics
- BN has the potential
  - suitable for different SW “ingredients”
  - monitoring on different layers (OS, middle-ware, process level, individual SW component); modularity
  - potentially: generate SWHM BN from Simulink model (NOTE: ADAPT IVHM generates BNs from wiring diagrams)
- improvement of test-bed
  - navigation component
  - failure injection
  - ARINC653 or OSEK model
  - SWHM modeling for testbed system