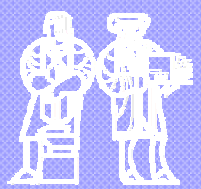
Spacecraft Autonomy

Seung H. Chung

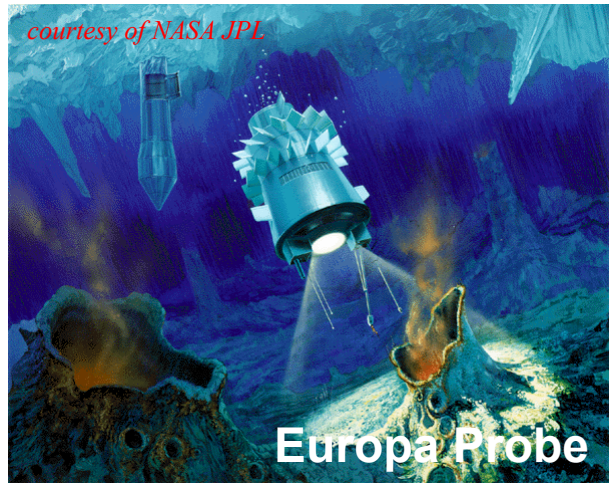


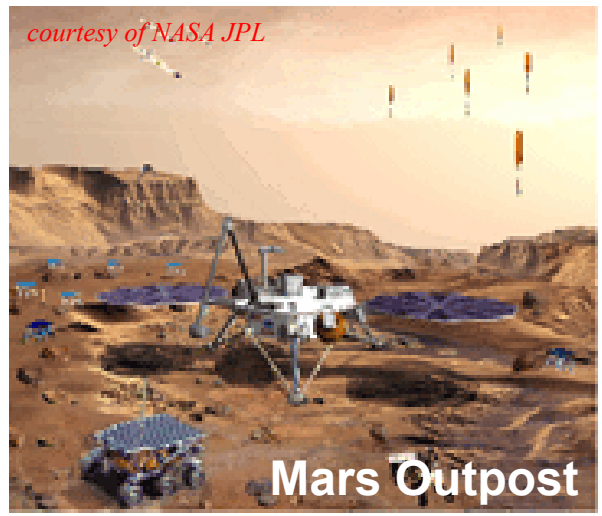
Massachusetts
Institute of
Technology

16.851
Satellite Engineering
Fall 2003

Why Autonomy?

- **Failures**
- **Anomalies**
- **Communication**
- **Coordination**





Autonomy Technologies

- Fault Detection, Isolation and Recovery
- Planning & Scheduling
- Intelligent Data Understanding

- Path Planning
  - Gradient method
  - Mixed integer linear programming (Prof John How)
  - Graph search (Prof Brian Williams)
- Localization & Mapping
  - Concurrent mapping and localization (Prof John Leonard)

Why Fault Detection Isolation & Recovery (FDIR)?

- Improve the likelihood of mission success by minimizing the downtime.
  - Increase productivity
  - Prevent loss of opportunities
  - Reduce safety risk

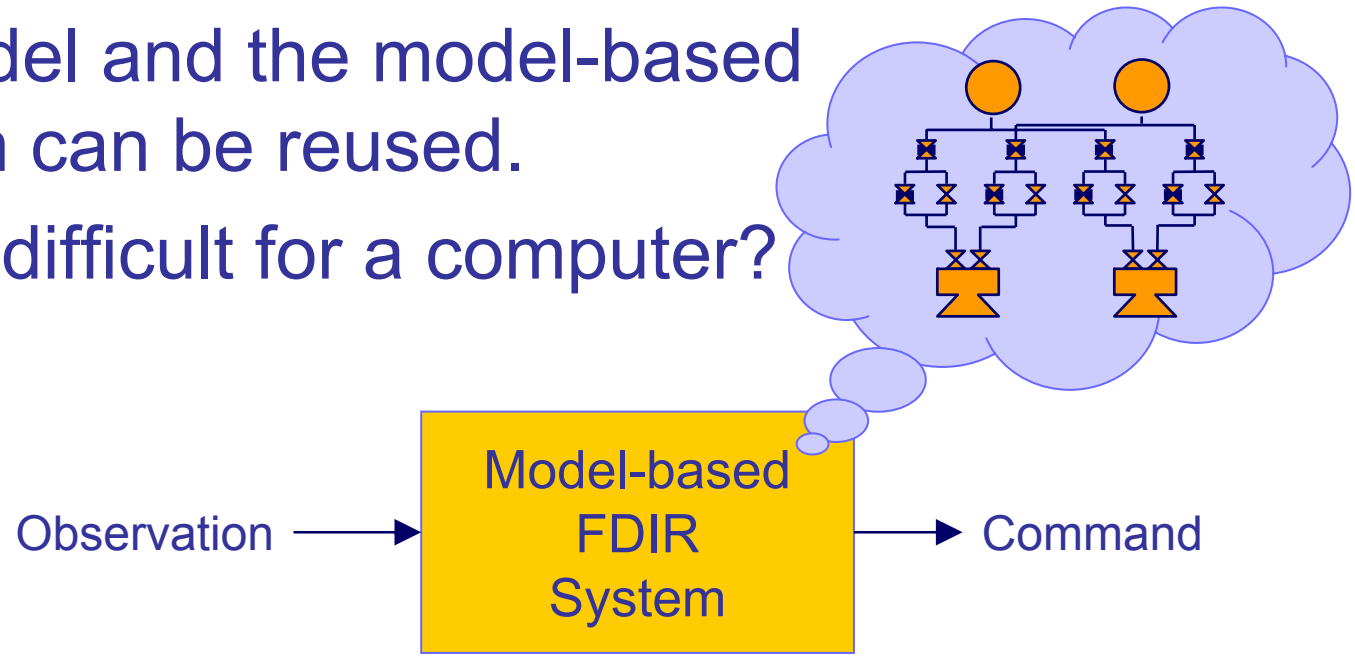
- For manned missions, longer system downtime implies higher risk to the astronauts.

FDIR Techniques

- If-then-else
  - Hard coded set of FDIR statements
- Rule-based
  - Set of rules written by the engineers
  - Fires a rule (i.e. executes a rule) when the rule is satisfied
  - Example
    - #24 (ID > 1A) And (Ishunt_D > 6A) for 10 sec, then Try_Sec_Bus_Reg_Off.
    - #27 (Red Battery Charger is ON) for 5 sec, then rule (28,29) stop.
  - The core software is reusable.
  - Engineers must enumerate all possible faults and combinations thereof along with the corresponding recovery methods.
  - Verifying the validity of the rules is difficult.

Model-based FDIR Technique

- Engineers model the behavior of the system (i.e. components).
- Computer detects/isolates/recovers faults by reasoning on the model of the system.
- Both the model and the model-based FDIR system can be reused.
- Problem too difficult for a computer?



Planning & Scheduling

- Planning
  - Given:
    - Set of actions a system can perform and the associated requirements and effects of the actions
    - Current state
    - Desired goal state
  - Objective: Compute a sequence of actions that achieves the desired goal state.

- Scheduling
  - Given: Set of tasks to execute and the associated constraints (i.e. time, resource, …)
  - Objective: Compute the proper order of the tasks that satisfies the constraints.

Planning Example

- Goal: Take an image of Alpha Centauri
- Plan:
  1. Compute current position and attitude
  2. Compute the necessary position and attitude for Alpha Centauri to be in view
  3. Initialize and warm-up the imaging system
  4. Change the position and point toward Alpha Centauri
  5. Open the shutter
  6. Take image

Why Planning & Scheduling?

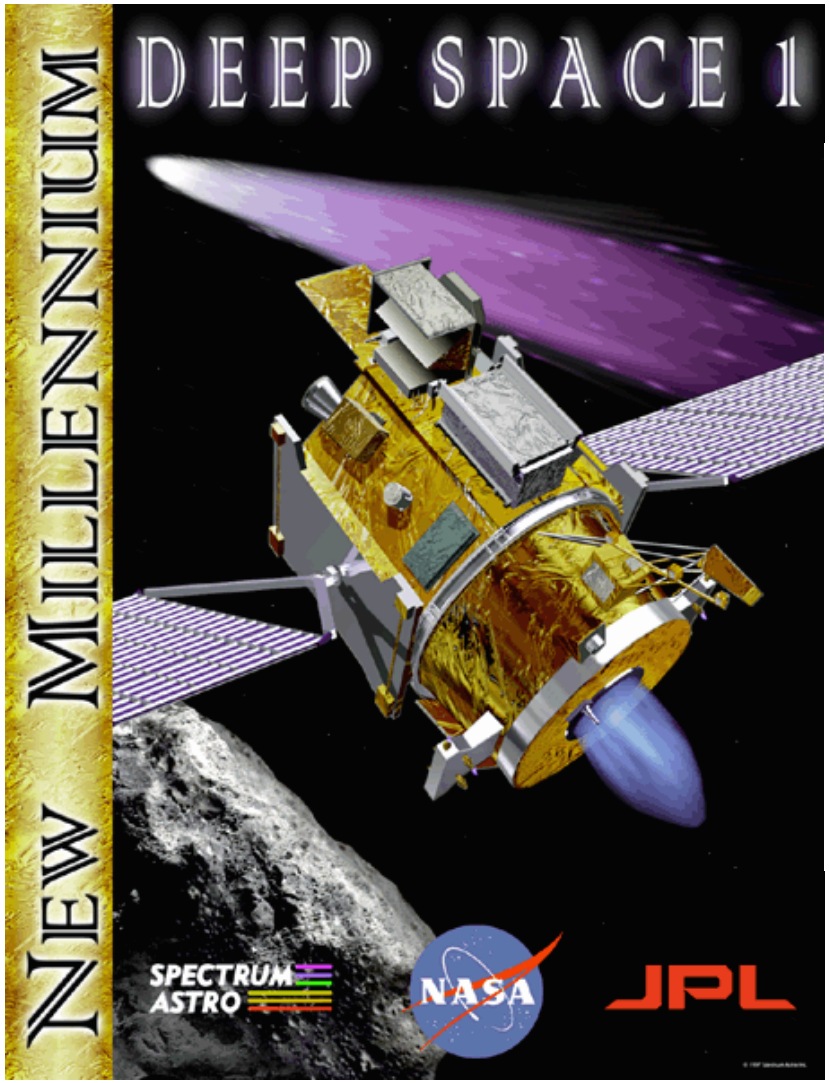
- Simplify spacecraft commanding.
- Simplify mission operations work.
- Enable timely replanning when necessary without communication time-delay issues.

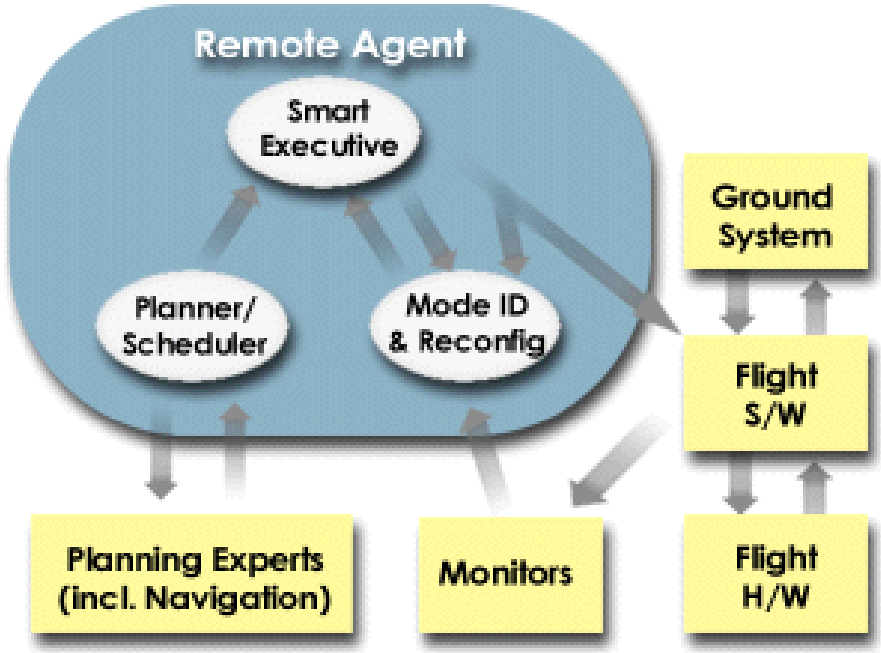
Intelligent Data Understanding

- What is it?
  - Knowledge Discovery: Is this something new, something interesting?
  - Pattern Recognition: What are the identifiable characteristics?
  - Classification and Clustering: Does this belong to some category of information?

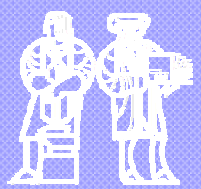
- Why?
  - The communication bandwidth does not allow transmission of all available data.
  - Serendipitous events…

Remote Agent Experiment





Model-based Embedded and Robotic Systems Group



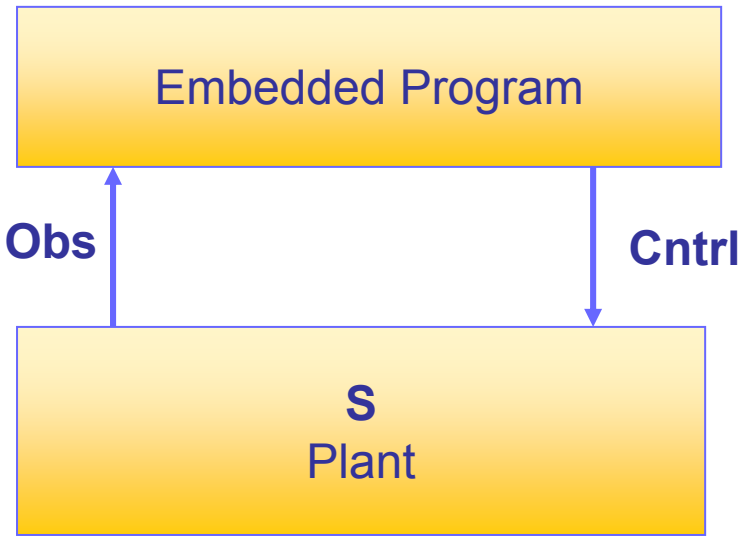
Massachusetts Institute of Technology

16.851
Satellite Engineering
Fall 2003

Model-based Programs Reason in Terms of State

Embedded programs interact with the system's sensors/actuators:

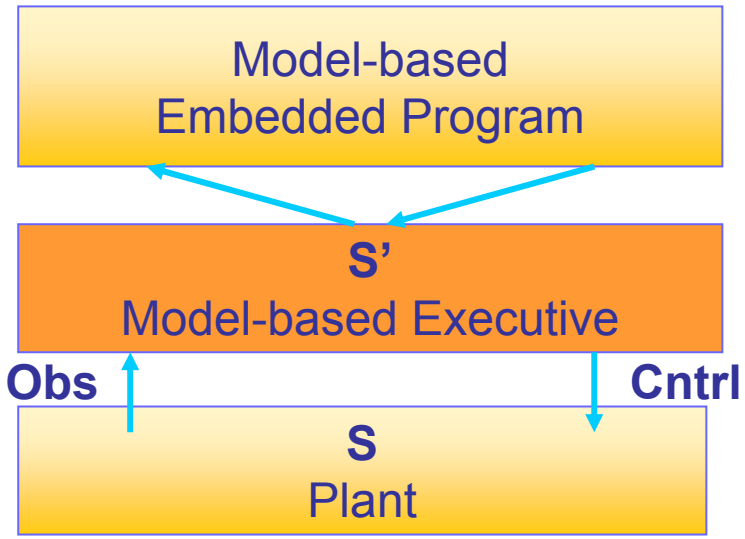
- Read sensors
- Set actuators



Programmer must map between state and sensors/actuators.

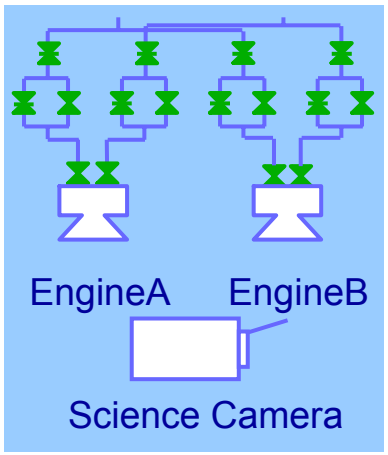
Model-based programs interact with the system's state:

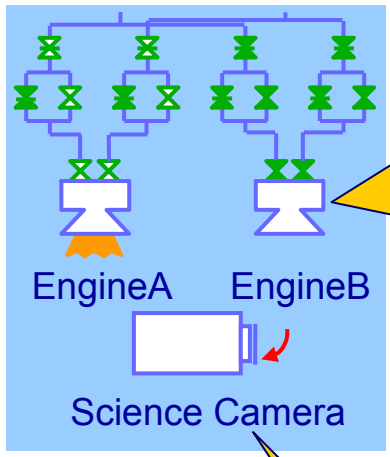
- Read state
- Set state

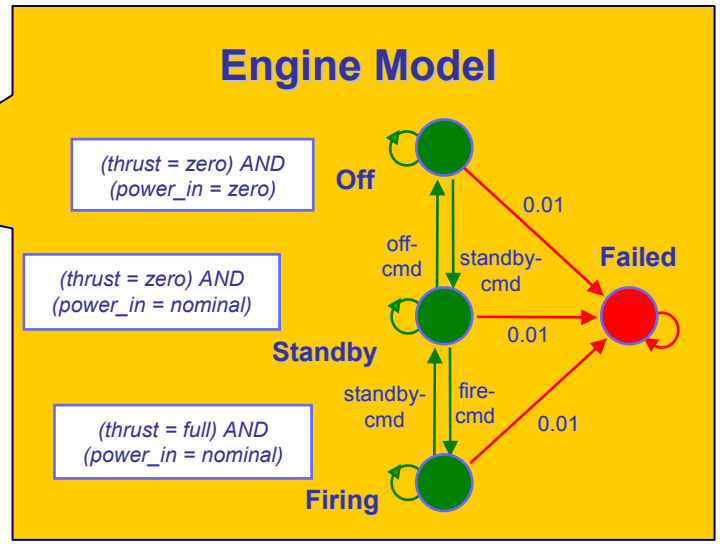


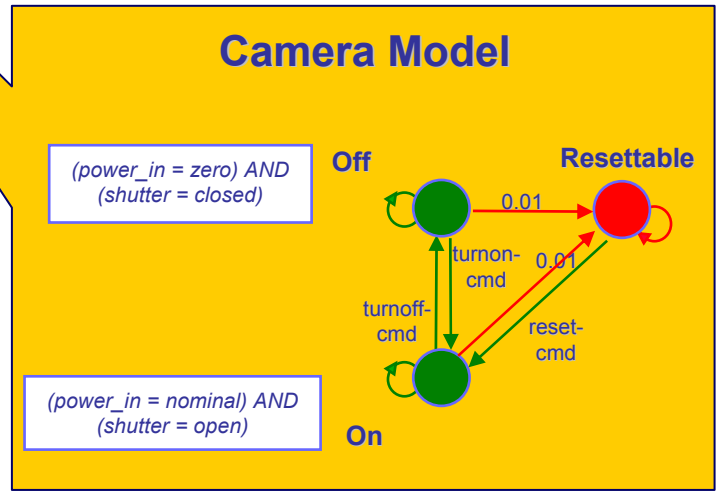
M-B Executive maps between states and sensors/actuators.

Model-based Programming Example









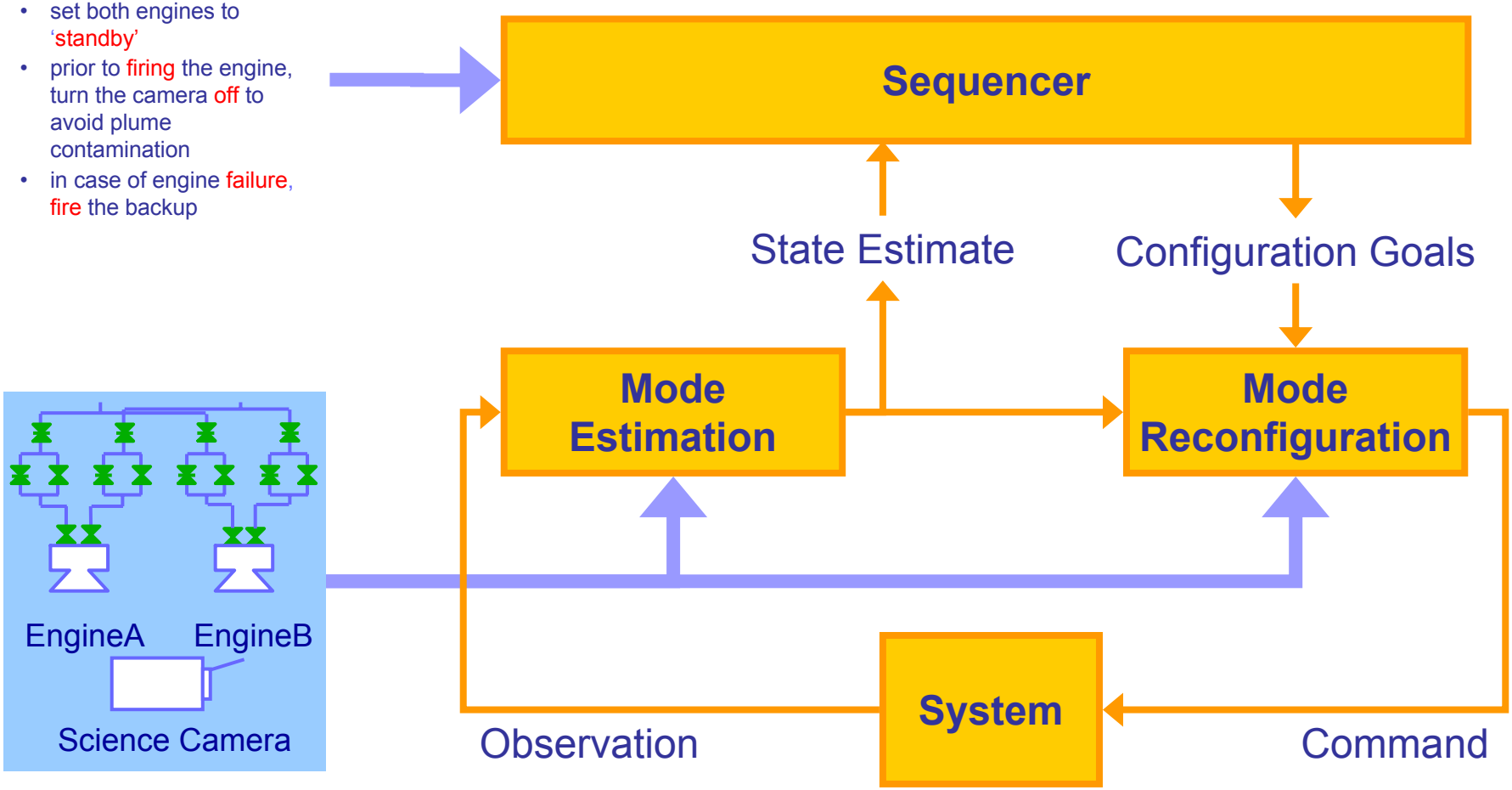
Systems engineers think in terms of state trajectories:

- goal: fire one of the two engines
- set both engines to 'standby'
- prior to firing the engine, turn the camera off to avoid plume contamination
- in case of engine failure, fire the backup

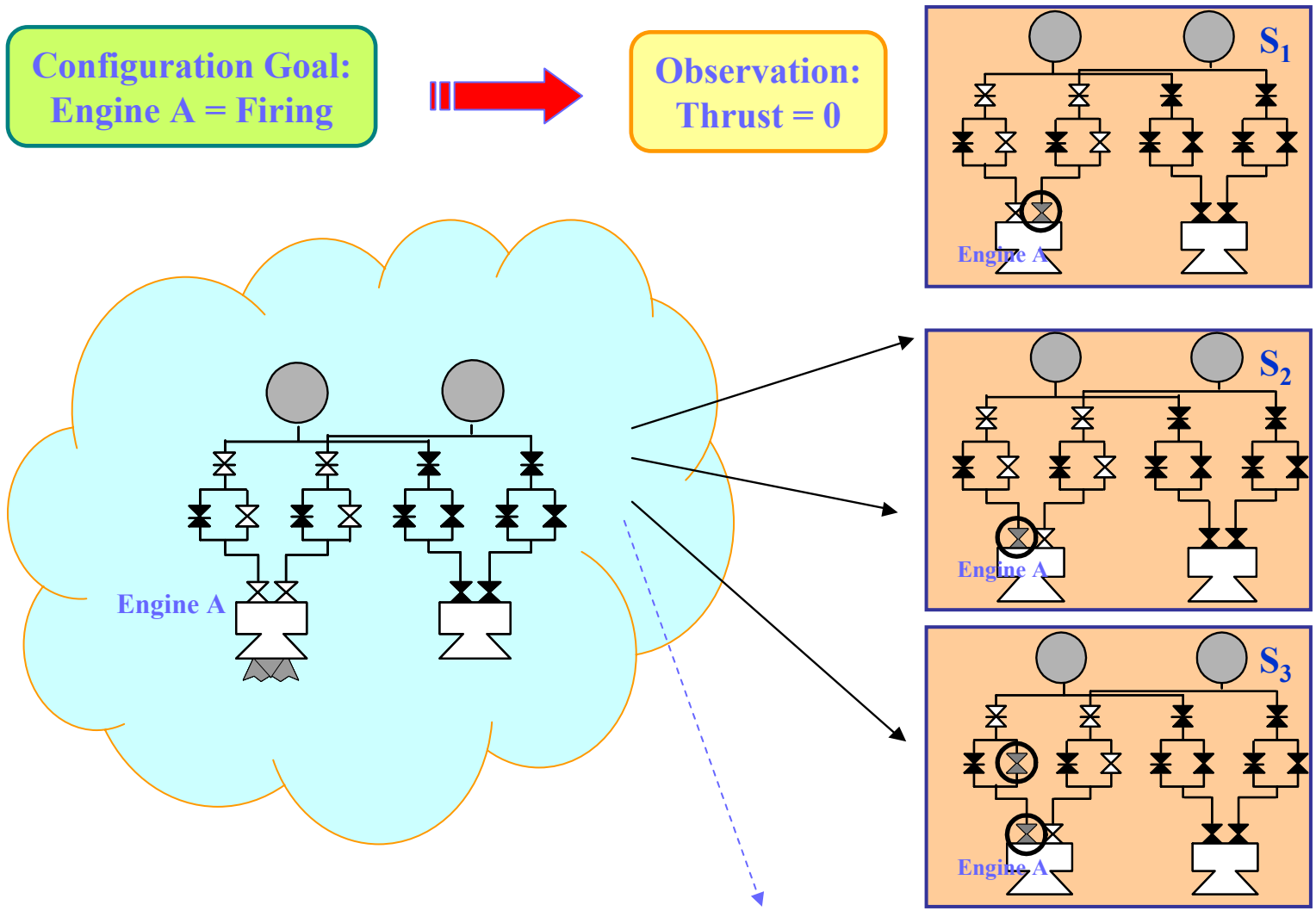
Engineers reason how to achieve state trajectories using component models

Model-based Executive "Executable Specification"

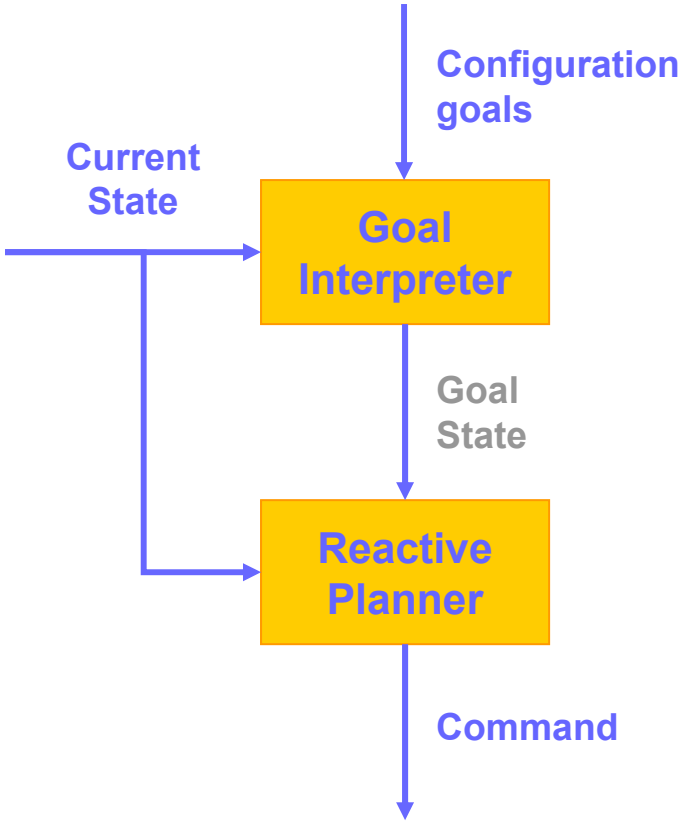
- goal: fire one of the two engines
- set both engines to 'standby'
- prior to firing the engine, turn the camera off to avoid plume contamination
- in case of engine failure, fire the backup

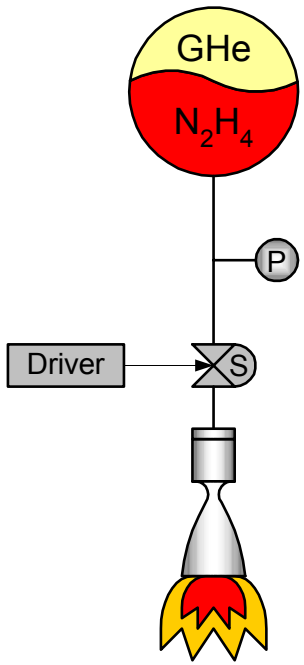


Mode Estimation



Mode Reconfiguration





INPUT

- Configuration Goal
  - Trust = on
- Current State
  - Tank = full
  - Pressure = nominal
  - Driver = off
  - Valve = closed
  - Thruster = off

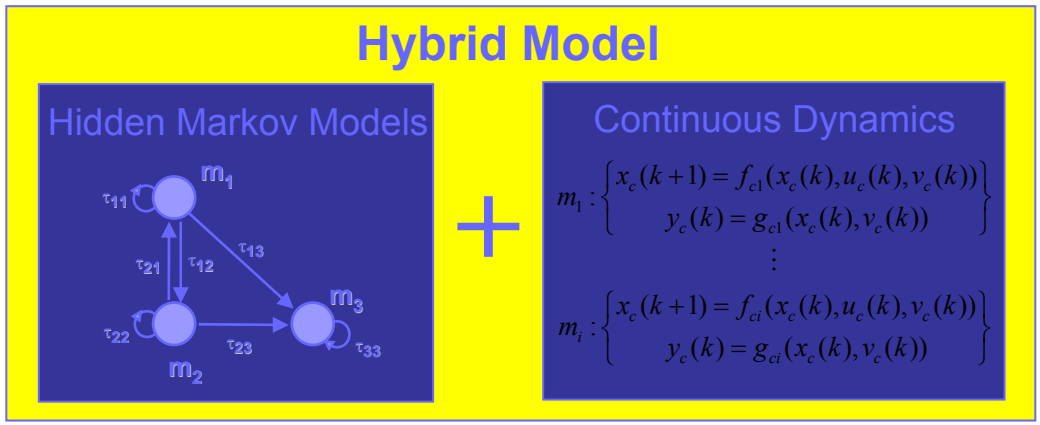
OUPUT

- Command
  - Turn driver on

Hybrid Mode Estimation

- Failures can manifest themselves through coupling between a system's continuous dynamics and its evolution through different behavior modes
  - $\Rightarrow$ must track over continuous state changes and discrete mode changes

- Symptoms initially on the same scale as sensor/actuator noise
  - $\Rightarrow$ need to extract mode estimates from subtle symptoms



Difficulty with Autonomy

- Most problems require exponential time…
  - Unacceptable for real-time systems that have hard-time requirement
- Possible Approach
  - Use divide-and-conquer approach
  - Provide additional knowledge that guides the search for solution
  - Use suboptimal solution
  - Perform the difficult computations offline and execute the results online