



# Model-based Autonomy

Michel Ingham  
Jet Propulsion Laboratory  
California Institute of Technology

Paul Robertson  
Model-based Embedded Robotic Systems Group  
MIT Computer Science and Artificial Intelligence Laboratory

Sunday, October 9<sup>th</sup>, 2005

1



## *Who are we?*



- Dr. Michel Ingham (JPL)
- Dr. Paul Robertson (MIT)
- Acknowledgment: Prof. Brian Williams (MIT)

2



## *Who are you?*



- How about briefly sharing:
  - Your name
  - Your affiliation
  - What you do
  - The reason for your interest in this topic
- This way, we can try to tailor some of the discussion to your interests...

3



## *Logistics*



- 8:00am to 11:50am
- 15 minute break around 9:50am
- Feel free to interrupt with questions at any time!

4



## Outline



- Introduction & Overview
- Model-based Programming
- Execution of Model-based Programs
- Fundamentals of Model-based Reasoning
- Modeling via State Analysis
- Advanced Methods
- Conclusion

5



## Outline



- **Introduction & Overview**
  - Motivation
  - Illustrative Scenario
  - Model-based Autonomy Architecture
- Model-based Programming
- Execution of Model-based Programs
- Fundamentals of Model-based Reasoning
- Modeling via State Analysis
- Advanced Methods
- Conclusion

6

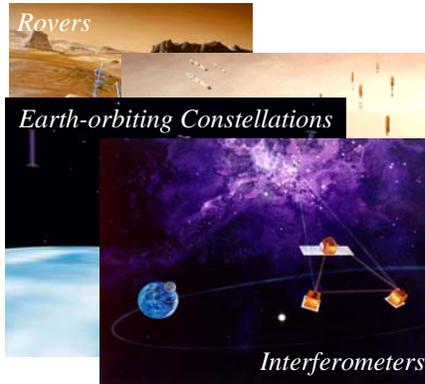
# Introduction & Overview

## Vast Networks of Complex Embedded Systems

- We are creating vast networks of embedded systems that perform critical functions over long periods of time, often in harsh and uncertain environments.
- These long-lived systems achieve their increasingly ambitious goals by coordinating a complex network of devices.
- Spacecraft must achieve robustness by managing a *complex set of subsystems*, over a range of possible *nominal and off-nominal* scenarios.
- Programming these systems is becoming an increasingly daunting task.



- We are creating vast networks of embedded systems that perform critical functions over long periods of time, often in harsh and uncertain environments.
- These long-lived systems achieve their increasingly ambitious goals by coordinating a complex network of devices.
- Spacecraft must achieve robustness by managing a *complex set of subsystems*, over a range of possible *nominal and off-nominal* scenarios.
- Programming these systems is becoming an increasingly daunting task.



- We are creating vast networks of embedded systems that perform critical functions over long periods of time, often in harsh and uncertain environments.
- These long-lived systems achieve their increasingly ambitious goals by coordinating a complex network of devices.
- Spacecraft must achieve robustness by managing a *complex set of subsystems*, over a range of possible *nominal and off-nominal* scenarios.
- Programming these systems is becoming an increasingly daunting task.



- We are creating vast networks of embedded systems that perform critical functions over long periods of time, often in harsh and uncertain environments.
- These long-lived systems achieve their increasingly ambitious goals by coordinating a complex network of devices.
- Spacecraft must achieve robustness by managing a *complex set of subsystems*, over a range of possible *nominal and off-nominal* scenarios.
- Programming these systems is becoming an increasingly daunting task.



11

- Time-tagged nominal command sequences

```

GS, SITURN, 490UA, BOTH, 96-355/03:42:00.000;

CMD, 7GYON, 490UA412A4A, BOTH, 96-355/03:47:00:000, ON;
CMD, 7MODE, 490UA412A4B, BOTH, 96-355/03:47:02:000, INT;
CMD, 6SVP, 490UA412A6A, BOTH, 96-355/03:48:30:000, 2;
CMD, 7ALRT, 490UA412A4C, BOTH, 96-355/03:50:32:000, 6;
CMD, 7SAFE, 490UA412A4D, BOTH, 96-355/03:52:00:000, UNSTOW;
CMD, 6ASSAN, 490UA412A6B, BOTH, 96-355/03:56:08:000, GV, 153, IMM, 231,
GV, 153;
CMD, 7VECT, 490UA412A4E, BOTH, 96-355/03:56:10.000, 0, 191.5, 6.5,
0.0, 0.0, 0.0,
96-350/
00:00:00.000, MVR;
SEB, SCTEST, 490UA412A23A, BOTH, 96-355/03:56:12.000, SYS1, NPERR;
CMD, 7TURN, 490UA412A4F, BOTH, 96-355/03:56:14.000, 1, MVR;
MISC, NOTE, 490UA412A99A, , , 96-355/04:00:00.000, , START OF TURN; ,
CMD, 7STAR, 490UA412A406A4A, BOTH, 96-355/04:00:02.000, 7, 1701,
278.813999, 38.74;
CMD, 7STAR, 490UA412A406A4B, BOTH, 96-355/04:00:04.000, 8, 350, 120.455999,
-39.8612;
CMD, 7STAR, 490UA412A406A4C, BOTH, 96-355/04:00:06.000, 9, 875, 114.162,
5.341;
CMD, 7STAR, 490UA412A406A4D, BOTH, 96-355/04:00:08.000, 10, 159, 27.239,
89.028999;
CMD, 7STAR, 490UA412A406A4E, BOTH, 96-355/04:00:10.000, 11, 0, 0.0, 0.0;
CMD, 7STAR, 490UA412A406A4F, BOTH, 96-355/04:00:12.000, 21, 0, 0.0, 0.0;

```

12

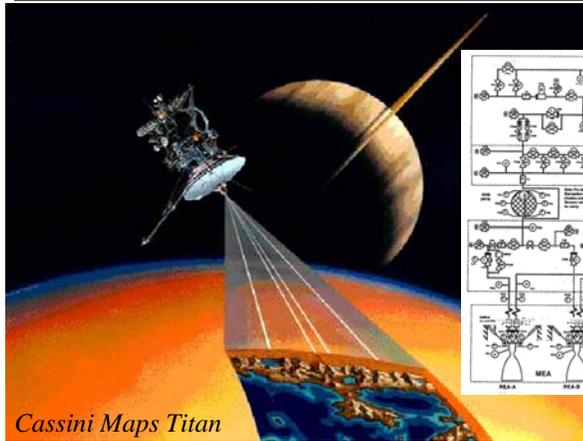
- Time-tagged nominal command sequences
- If absolutely necessary, conditional behavior via rule-based monitors or hard-coded state machines

5.1.2.2.6	If the supply current for OXIDIZER TANK PRIMARY HEATER, multiplied by the bus voltage, is greater than TBD watts, then the FPP shall issue a command to turn OFF OXIDIZER TANK PRIMARY HEATER.
5.1.2.2.7	If the supply current for HELIUM TANK PRIMARY HEATER, multiplied by the bus voltage, is greater than TBD watts, then the FPP shall issue a command to turn OFF HELIUM TANK PRIMARY HEATER.
5.1.2.2.8	If the supply current for STAR TRACKER A, multiplied by the bus voltage, is greater than TBD watts, then the FPP shall issue a command to turn OFF STAR TRACKER A.
5.1.2.2.9	If the supply current for STAR TRACKER B, multiplied by the bus voltage, is greater than TBD watts, then the FPP shall issue a command to turn OFF STAR TRACKER B.
5.1.2.2.10	If the supply current for IMU PPSM-A, multiplied by the bus voltage, is greater than TBD watts, then the FPP shall issue a command to turn OFF IMU PPSM-A.
5.1.2.2.11	If the supply current for IMU PPSM-B, multiplied by the bus voltage, is greater than TBD watts, then the FPP shall issue a command to turn OFF IMU PPSM-B.
5.1.2.2.12	If the supply current for REACTION WHEEL 1, multiplied by the bus voltage, is greater than TBD watts, then the FPP shall issue a command to turn OFF REACTION WHEEL 1. In addition, the corresponding rules that monitor power dissipation of the remaining three reaction wheels shall be disabled. REACTION WHEEL 1 shall be flagged as "unavailable" to the G&C task in the MP.

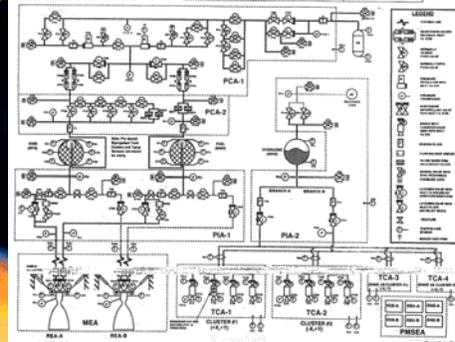
- Time-tagged nominal command sequences
- If absolutely necessary, conditional behavior via rule-based monitors or hard-coded state machines
- Usual off-nominal behavior response is "safe mode":
  - costly ground ops
  - lost science opportunities
- For critical mission sequences:
  - Safing mechanism is disabled
  - Hard-coded fault protection via highly-specialized software modules:
    - ad-hoc
    - complex
    - expensive to generate and test



Entry, descent & landing



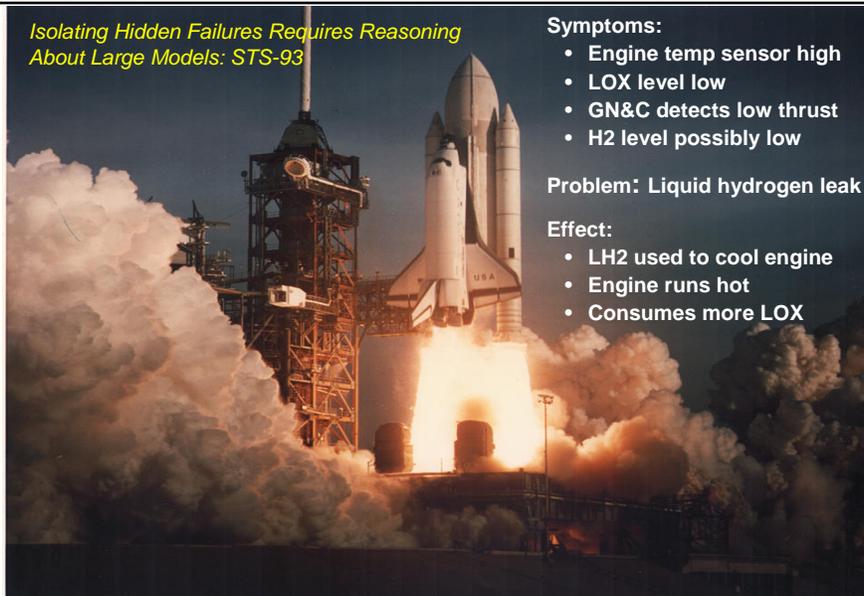
Cassini Maps Titan



Large collections of devices must work in concert to achieve goals

- Devices indirectly observed and controlled.
- Must manage large levels of redundancy.
- Need quick, robust response to anomalies throughout life.

*Isolating Hidden Failures Requires Reasoning About Large Models: STS-93*



**Symptoms:**

- Engine temp sensor high
- LOX level low
- GN&C detects low thrust
- H2 level possibly low

**Problem:** Liquid hydrogen leak

**Effect:**

- LH2 used to cool engine
- Engine runs hot
- Consumes more LOX



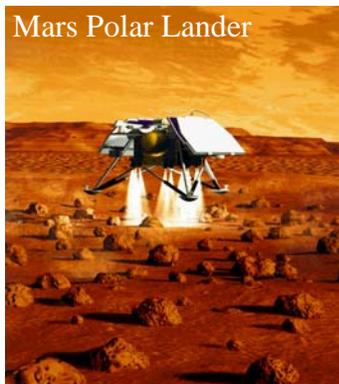
### “Houston, we have a problem...”

- Quintuple fault occurs (three shorts, tank-line and pressure jacket burst, panel flies off).
- Ground assembles novel repair.
- Swigert & Lovell work on Apollo 13 emergency rig lithium hydroxide unit.
- Mattingly works in ground simulator to identify novel sequence handling severe power limitations.

**Autonomy software should embody the innovation exemplified in Apollo 13 and other missions.**

17

Mars Polar Lander



### Leading Diagnosis:

- Legs deployed during descent.
- Noise spike on leg sensors latched by software monitors.
- Laser altimeter registers 40m.
- Begins polling leg monitors to determine touch down.
- Latched noise spike read as touchdown.
- Engine shutdown at ~40m.

*Model-based Programming:*  
Creation of embedded & robotic systems that manage interactions automatically, by reasoning from models of themselves and their environment.

↓  
Programmers are overwhelmed by the bookkeeping of reasoning about unlikely hidden states.

18



## Terminology



- **Model-based Programming** languages elevate the task to storyboarding and modeling.
  - Engineers program their high-level intentions in terms of how they would like the state of the world to evolve.
  - Programmers describe the world (system + environment) using commonsense models of normal and faulty behavior.
- **Model-based Executives** implement these intentions by reasoning on the fly.
  - They continually hypothesize the likely states of the world, given what they observe.
  - They continually plan and execute actions in order to achieve the programmer's intentions.
- **Model-based Autonomy** is the discipline of applying Model-based Programming principles to the control of complex embedded systems.
  - These systems achieve unprecedented robustness ("fault-awareness") by leveraging the capabilities of their Model-based Executives.
  - They automate onboard sequence execution by tightly integrating goal-driven commanding, fault detection, diagnosis and recovery.

19

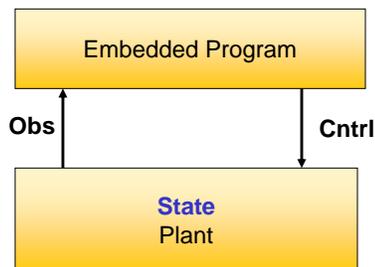


## Model-based Programs Reason about State



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

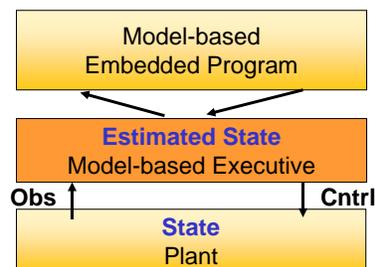
- Read sensors
- Set actuators



Programmers must reason through interactions between state and sensors/actuators.

Model-based programs interact with the system's (hidden) state directly:

- Read state
- Set state

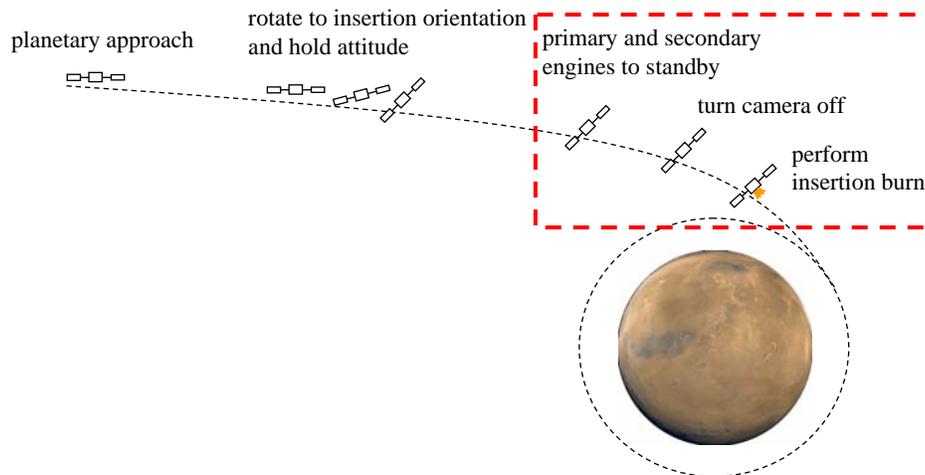


Model-based Executives automatically reason through interactions between states and sensors/actuators.

20



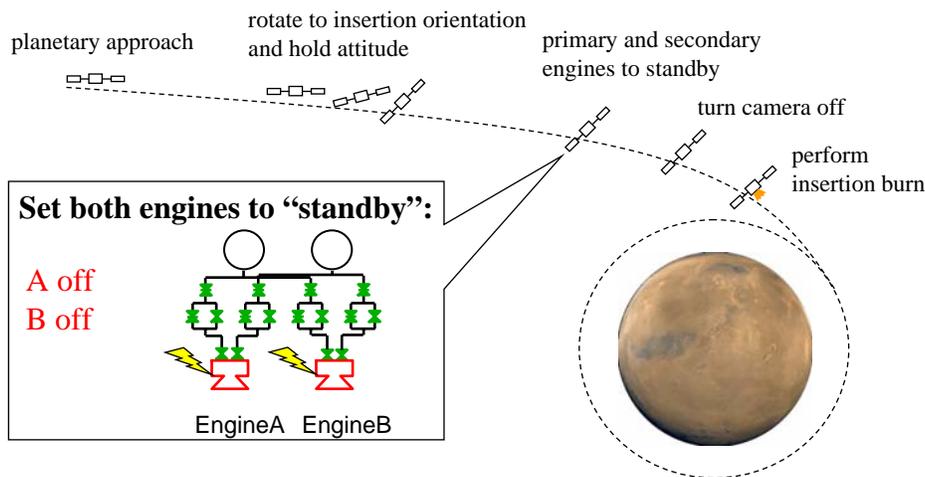
## Orbital Insertion Sequence: State-based Specification



21



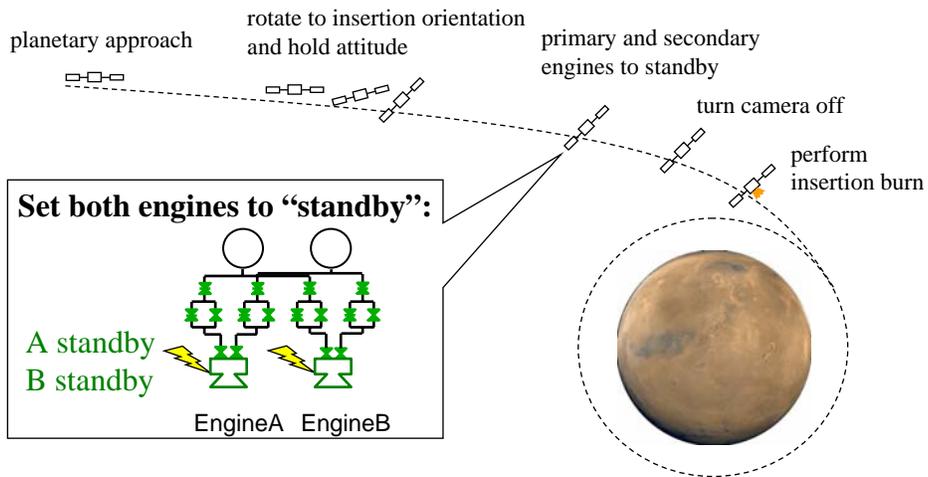
## Orbital Insertion Sequence: State-based Specification



22



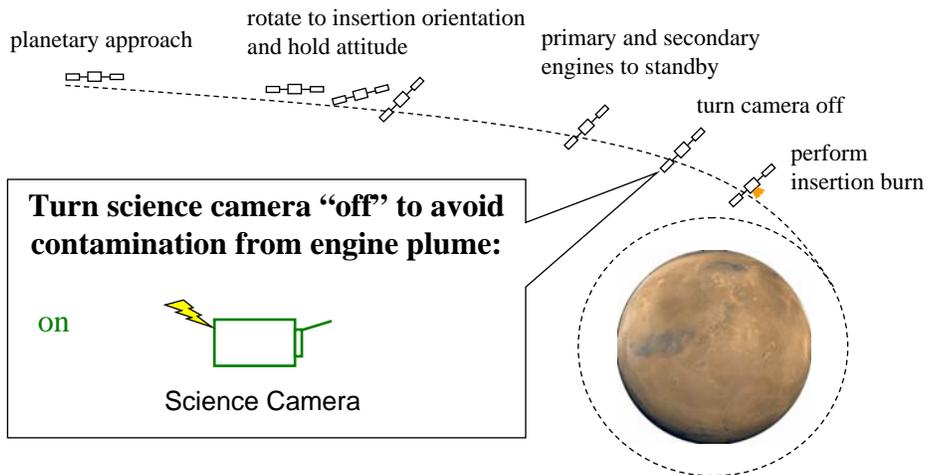
## Orbital Insertion Sequence: State-based Specification



23



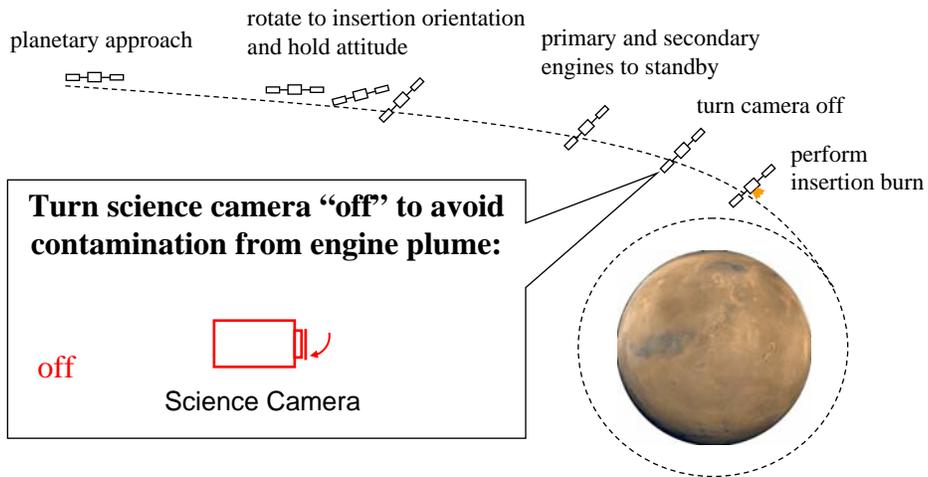
## Orbital Insertion Sequence: State-based Specification



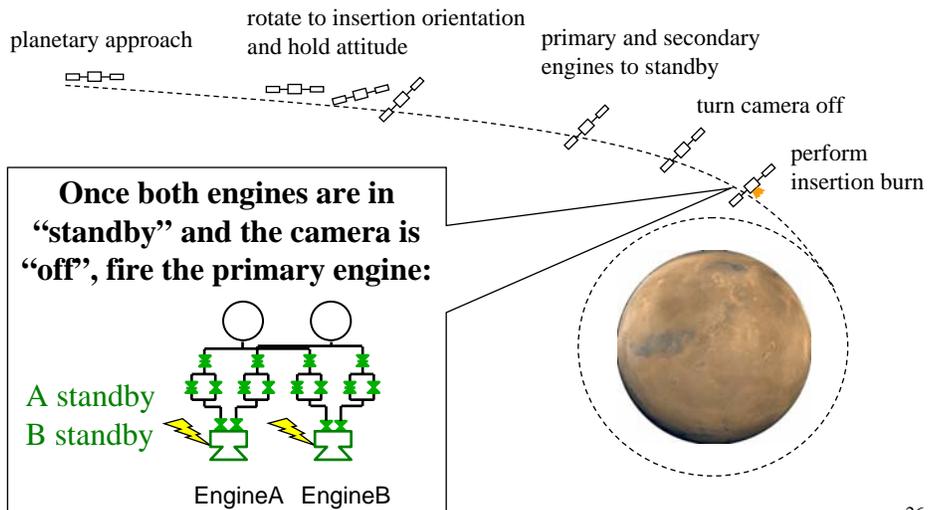
24



## Orbital Insertion Sequence: State-based Specification

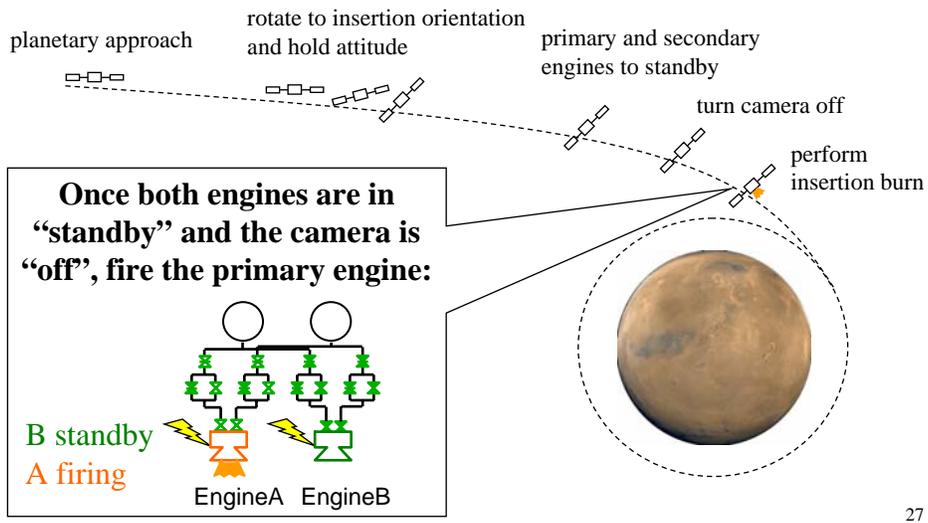


## Orbital Insertion Sequence: State-based Specification

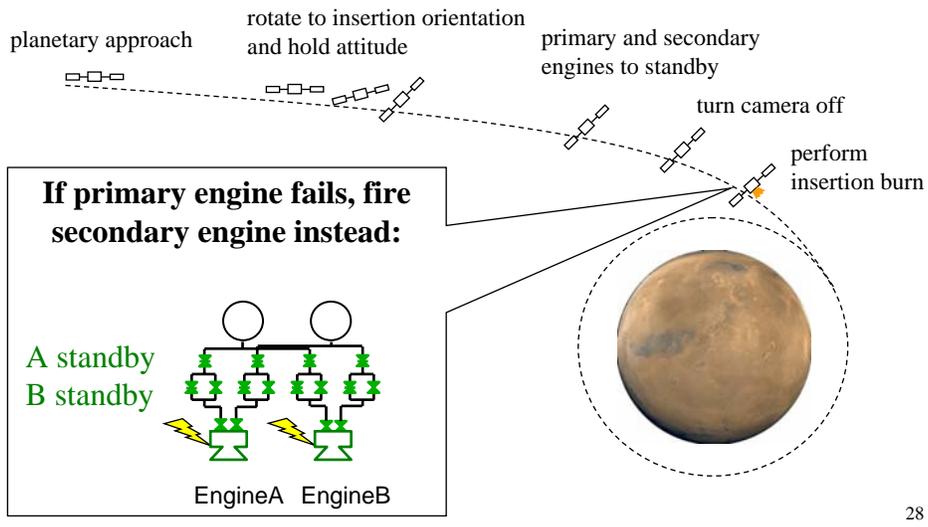




# Orbital Insertion Sequence: State-based Specification

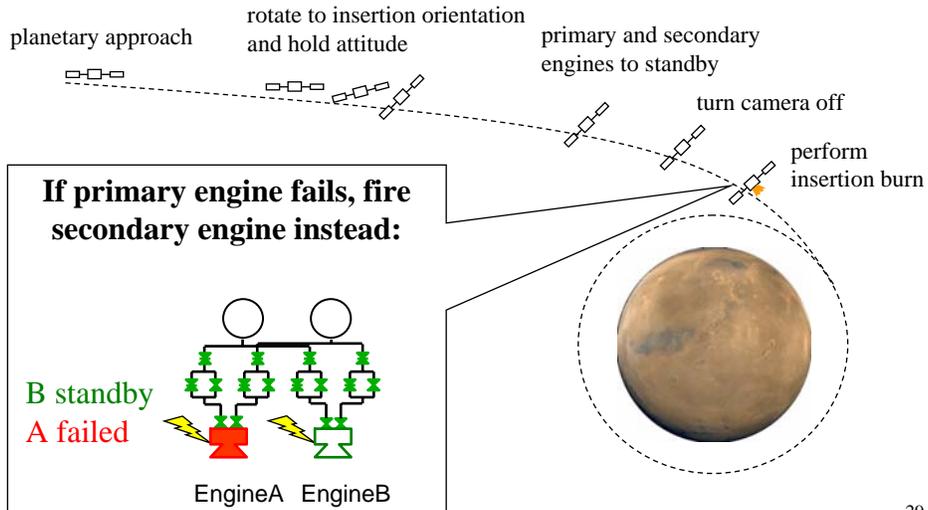


# Orbital Insertion Sequence: Off-Nominal States

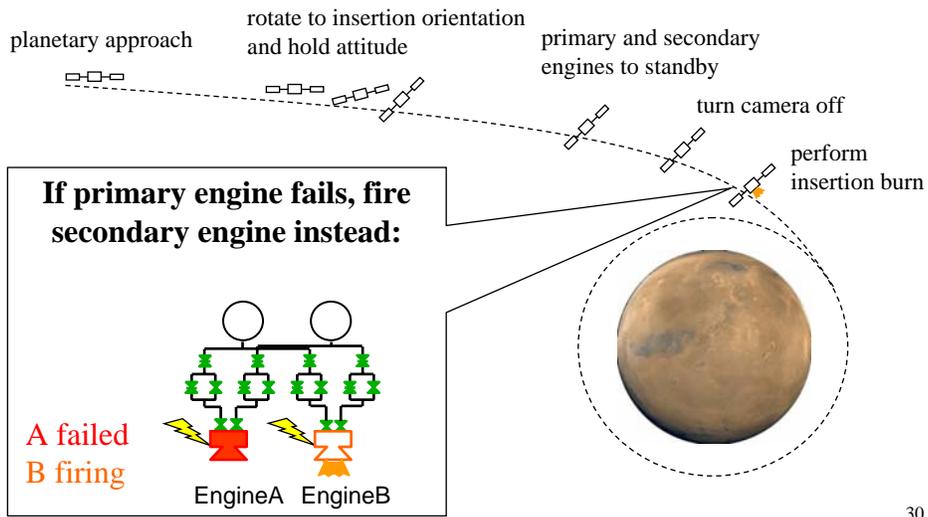




## Orbital Insertion Sequence: Off-Nominal States

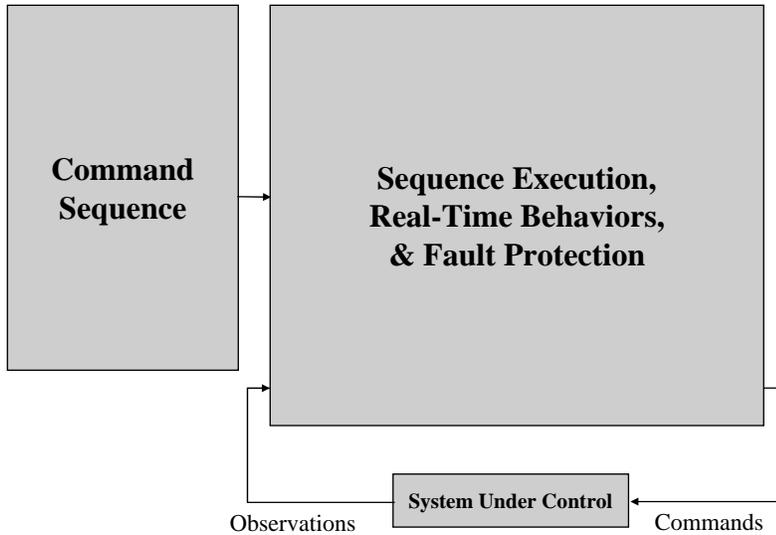


## Orbital Insertion Sequence: Off-Nominal States

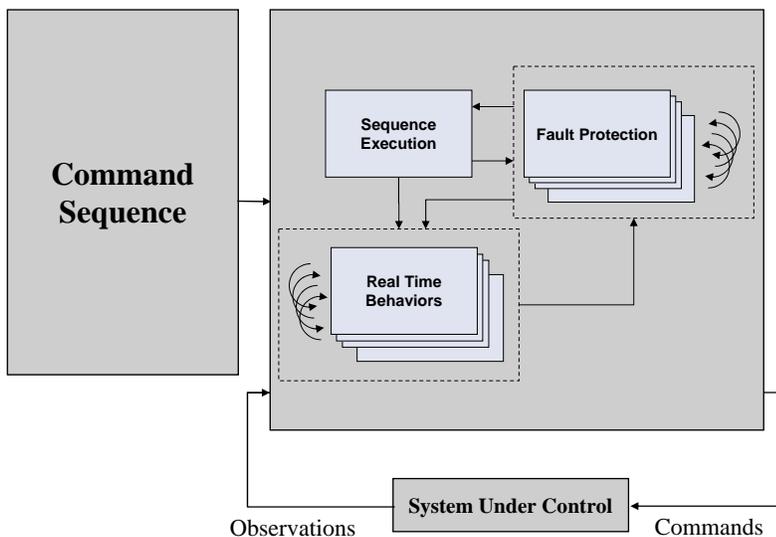


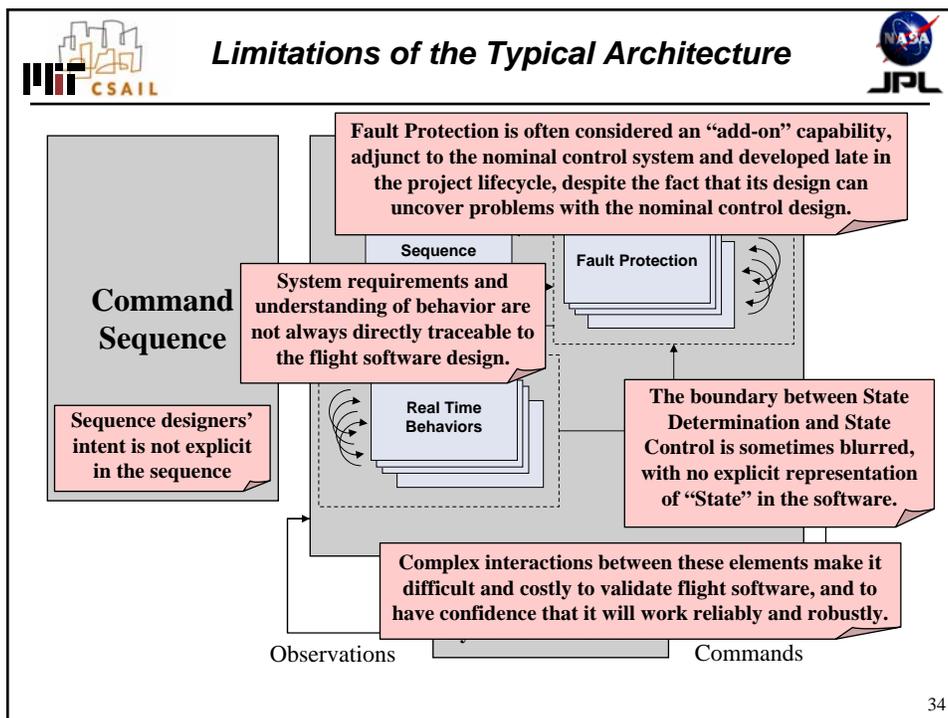
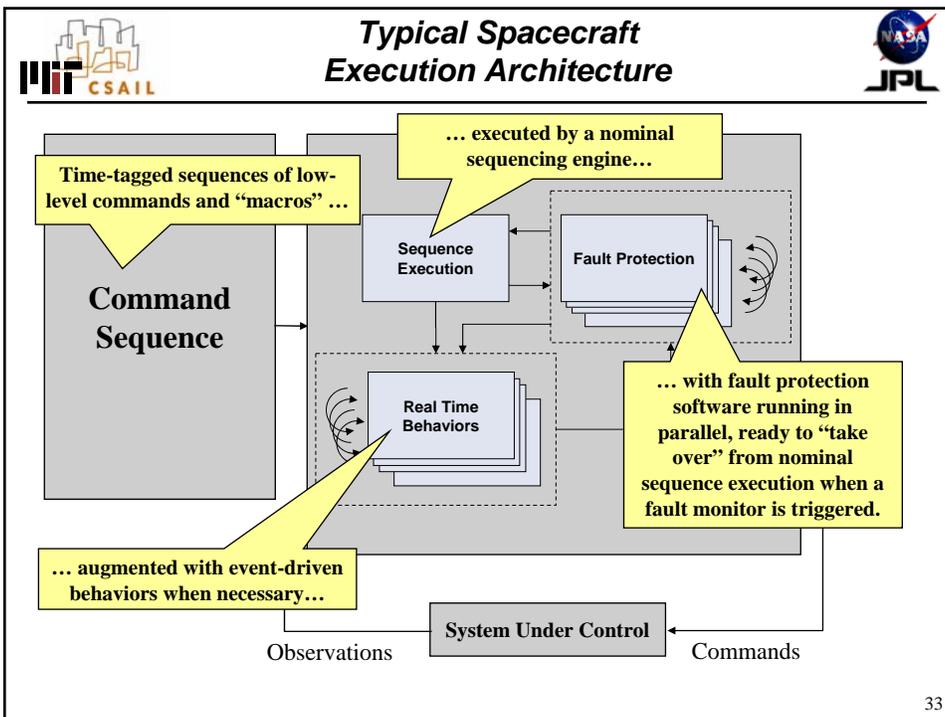


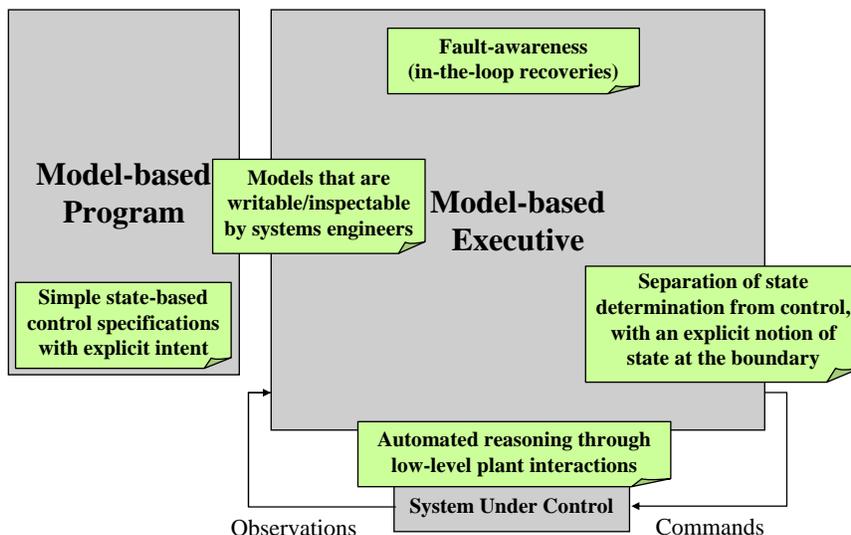
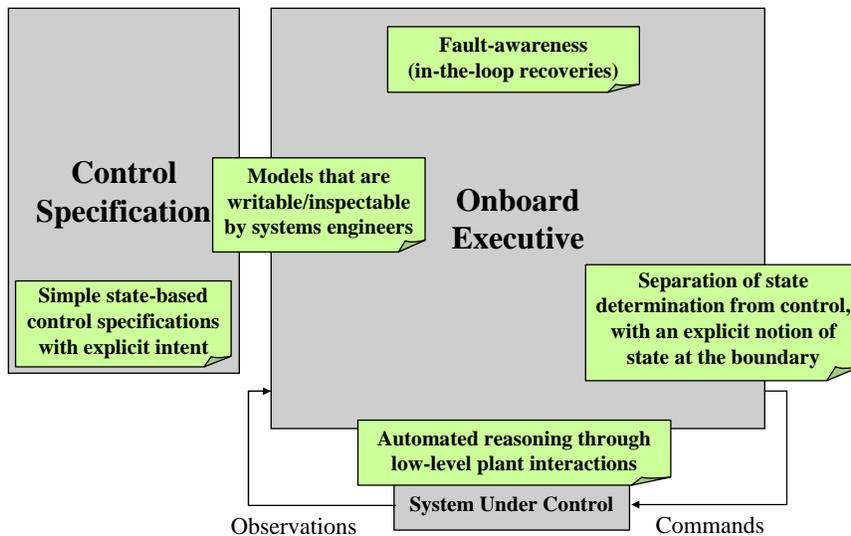
# Typical Spacecraft Execution Architecture

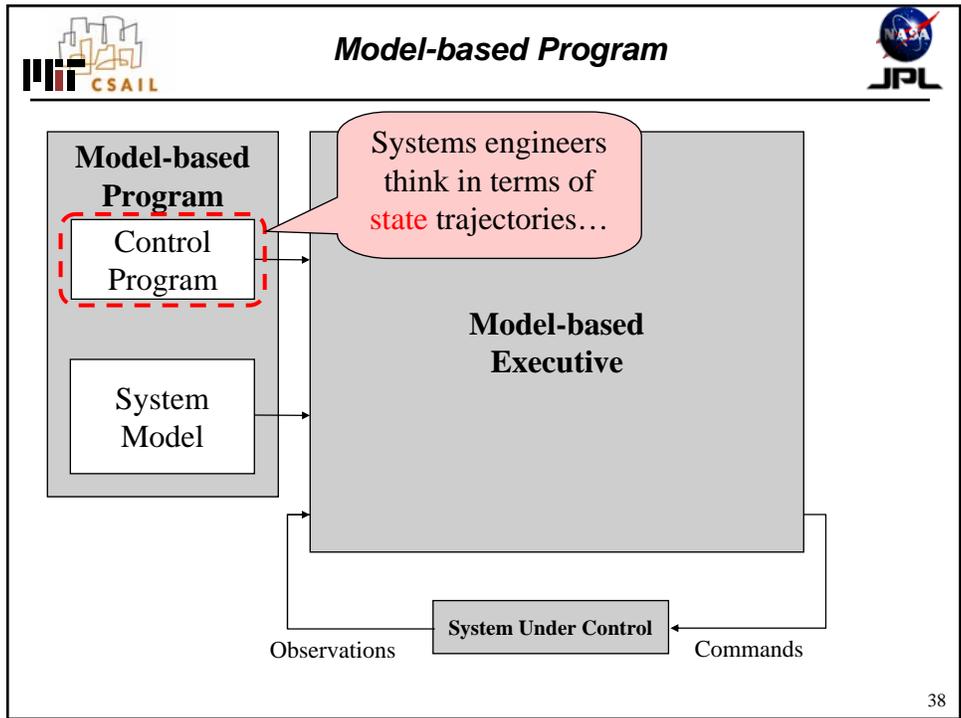
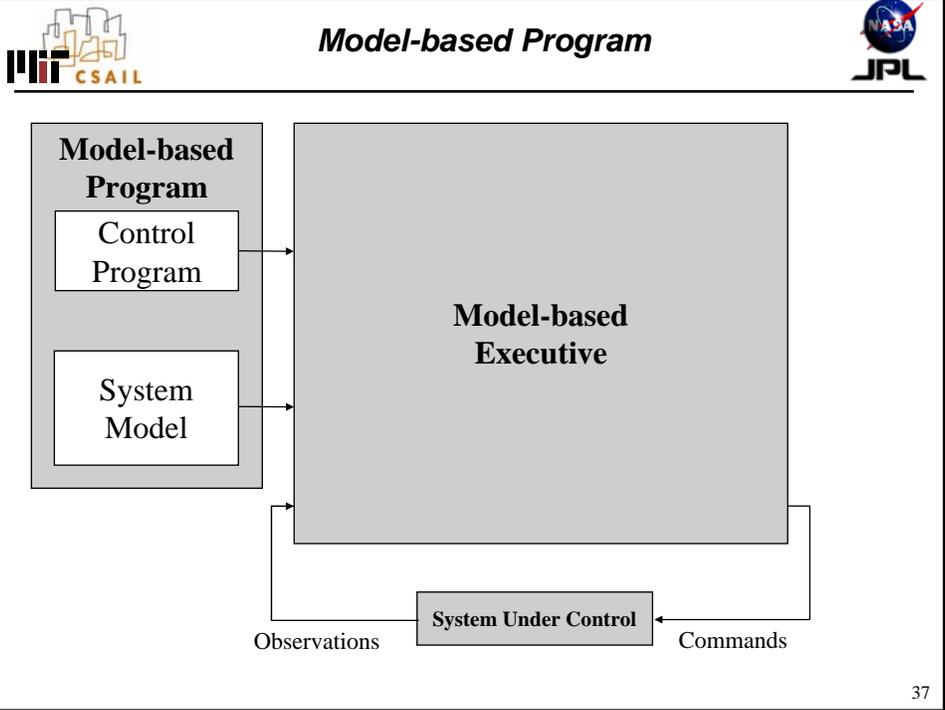


# Typical Spacecraft Execution Architecture



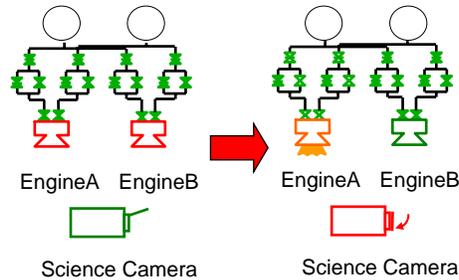






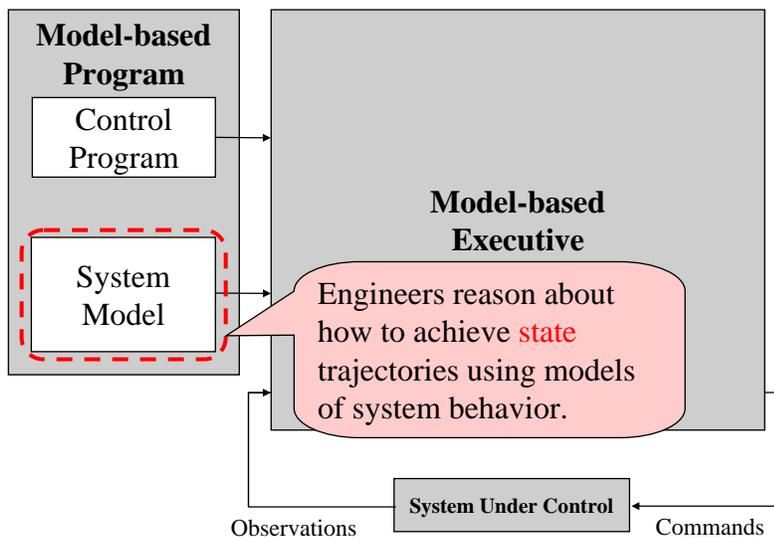
Control Program specifies **state** trajectories:

- fires one of two engines
- sets both engines to 'standby'
- prior to firing engine, camera must be turned off to avoid plume contamination
- in case of primary engine failure, fire backup engine instead

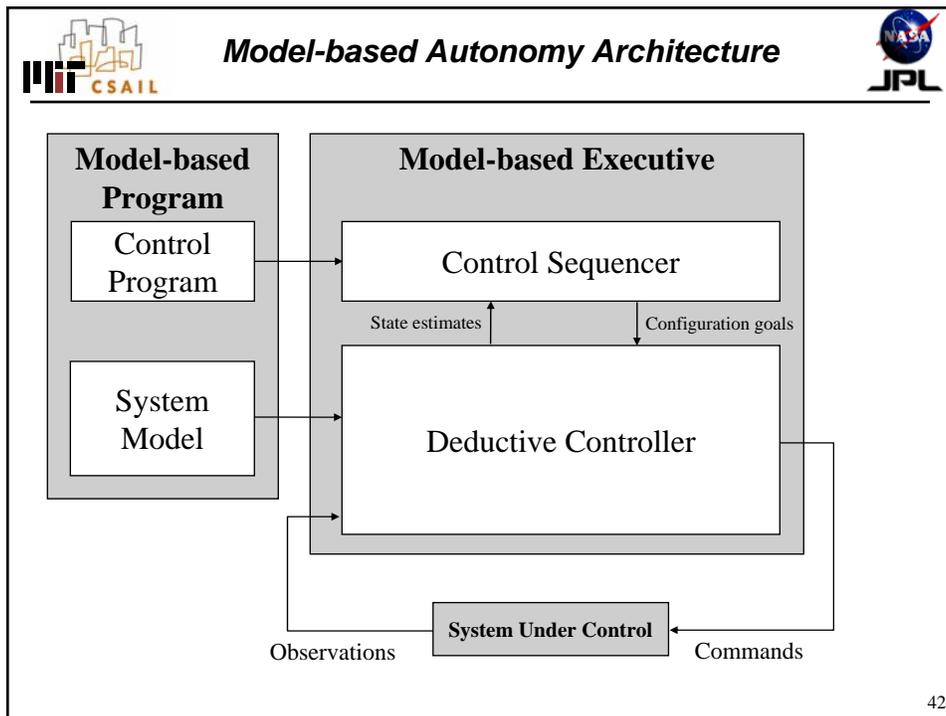
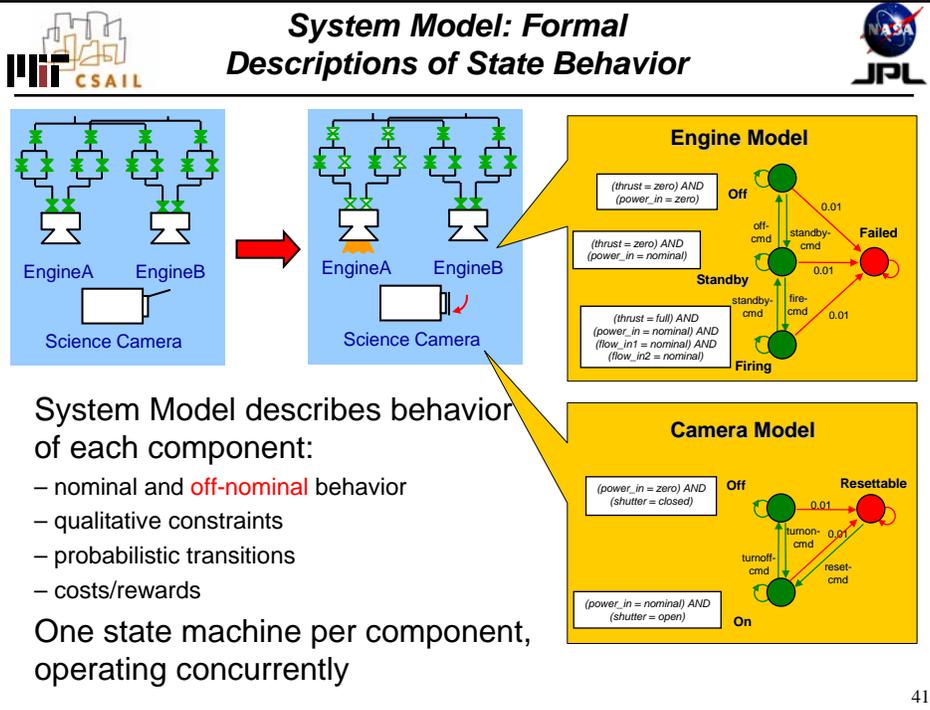


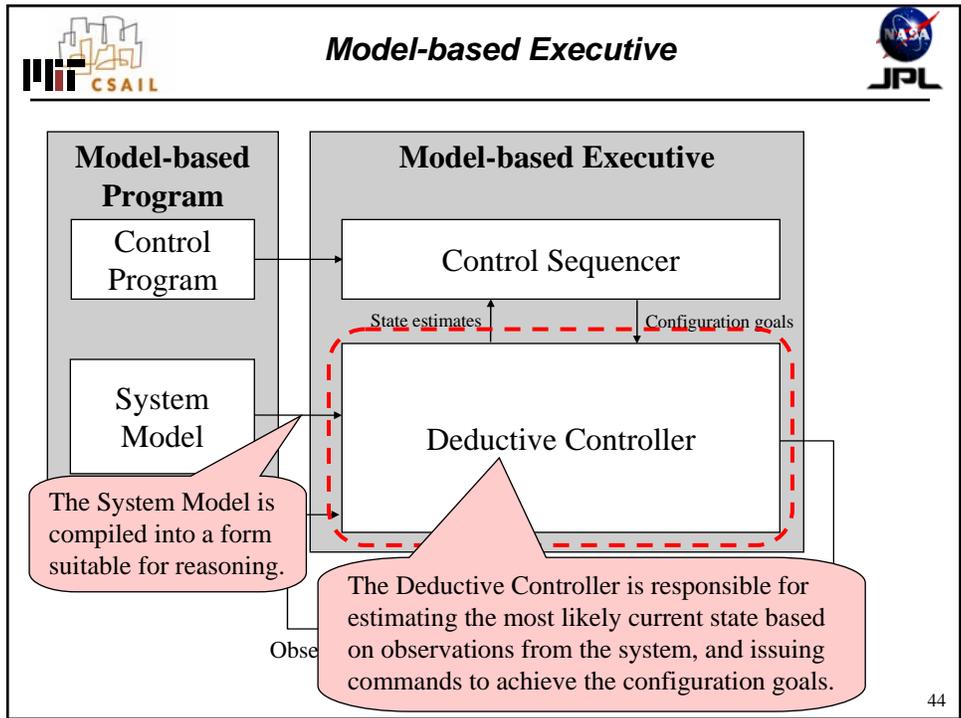
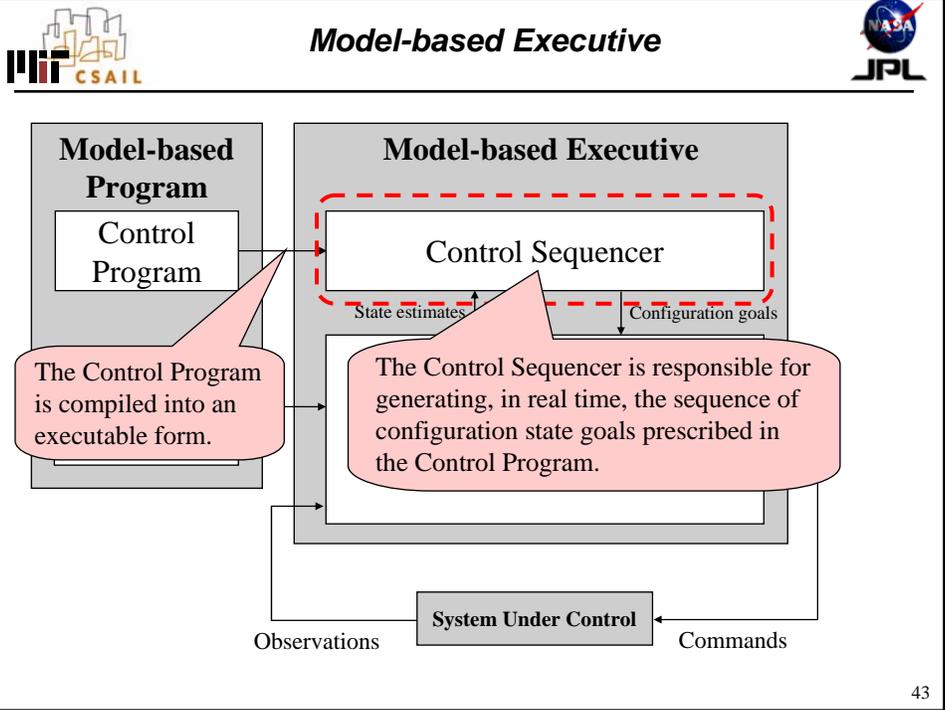
```
OrbitInsert():
(do-watching ( (EngineA = Firing) OR (EngineB = Firing) )
(parallel
  (EngineA = Standby)
  (EngineB = Standby)
  (Camera = Off)
  (do-watching (EngineA = Failed)
    (when-donext ( (EngineA = Standby) AND (Camera = Off) )
      (EngineA = Firing) ) )
  (when-donext ( (EngineA = Failed) AND (EngineB = Standby) AND (Camera = Off) )
    (EngineB = Firing) ) ) )
```

39

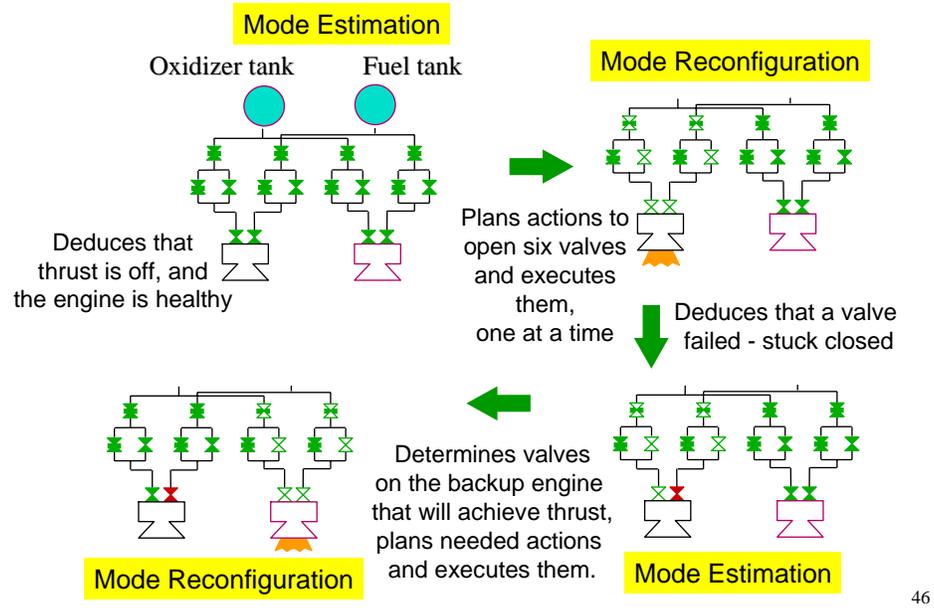
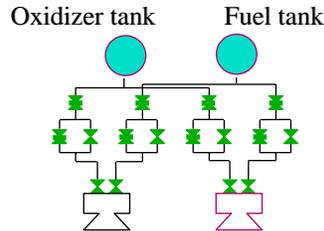


40





- States like (*EngineA = Firing*) are not necessarily DIRECTLY observable or controllable
- When the Control Sequencer issues the configuration goal (*EngineA = Firing*), the Deductive Controller...



- Introduction to Model-based Programming:
  - Control Programs
  - System Models

