No More Band-Aids: Integrating FM into the Onboard Execution Architecture

Michel Ingham
Systems Engineering Section
JPL / Caltech
Objectives

• Provide an example of an alternative system architecture, in which Fault Management is “integrated” with nominal execution.

• Encourage thinking outside the usual box, when it comes to Fault Management architecture.
Large collections of devices must work in concert to achieve goals
• Devices indirectly observed and controlled.
• Must manage significant redundancy.
• Need quick, robust response to anomalies throughout life.
Europa Mission Concepts

Europa Orbiter Mission perform geophysical measurements (“Water” science)

Multiple-Flyby Mission perform remote measurements (“Chemistry” and “Energy” science)

Europa Landed Mission perform detailed in-situ characterization at a Europan landing site assessing key habitability science objectives
3) **Deorbit Burn**, perform via SRM and monoprop, integrated on high precision IRU w/ real time burn profile updates

1 & 2) **Reconnaissance Imagery** and **Site Certification Process** select prioritize list of landing sites in a target zone

4a) **Terrain Relative Navigation (TRN)** determines location and reachable sites

4b) **TRN** performs divert to top priority reachable site

5) **Low altitude Hazard Avoidance**

**Leverage 15 Years of Technology Development in Human & Mars Programs**

Knowledge Uncertainty

Safe Precision Landing Target Area: 10m x 10m

HA Area: 100m x 100m (before HA divert)

Reachable Sites: 3km x 3km (before TRN divert)

Target Zone: 6km x 3km (before SRM burn)
Typical Spacecraft Execution Architecture

Command Sequence

Sequence Execution, Real-Time Behaviors, & Fault Protection

System Under Control

Observations

Commands
Typical Spacecraft Execution Architecture

Command Sequence

Sequence Execution

Fault Protection

Real Time Behaviors

System Under Control

Observations

Commands
Typical Spacecraft Execution Architecture

Time-tagged sequences of low-level commands and “macros” ...

... executed by a nominal sequencing engine...

Sequence Execution

Fault Protection

Real Time Behaviors

... with fault protection software running in parallel, ready to “take over” from nominal sequence execution when a fault monitor is triggered.

... augmented with event-driven behaviors when necessary...

Command Sequence

Observations

System Under Control

Commands
Limitations of the Typical Architecture

Sequence designers’ intent is not explicit in the sequence.

System requirements and understanding of behavior are not always directly traceable to the flight software design.

Fault Protection is often considered an “add-on” capability, adjunct to the nominal control system and developed late in the project lifecycle, despite the fact that its design can uncover problems with the nominal control design.

The boundary between State Determination and State Control is sometimes blurred, with no explicit representation of “State” in the software.

Complex interactions between these elements make it difficult and costly to validate flight software, and to have confidence that it will work reliably and robustly.
Desirable Architectural Features

Control Specification
- Simple state-based control specifications with explicit intent
- Models that are writable/inspectable by systems engineers

Onboard Executive
- Fault-awareness (in-the-loop recoveries)
- Separation of state determination from control, with an explicit notion of state at the boundary

System Under Control
- Control system manages low-level plant interactions

Observations

Commands
Model-based Programs and Executives Provide These Features

Model-based Program

- Simple state-based control specifications with explicit intent
- Models that are writable/inspectable by systems engineers

Model-based Executive

- Fault-awareness (in-the-loop recoveries)
- Separation of state determination from control, with an explicit notion of state at the boundary
- Control system manages low-level plant interactions

System Under Control

Observations

Commands
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.
**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.
Model-based Program

Model-based Program
- Control Program
- System Model

Model-based Executive

System Under Control
- Observations
- Commands
Model-based Program

Model-based Program

Control Program

System Model

System Under Control

Model-based Executive

Systems engineers think in terms of state trajectories…

Observations

Commands
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) ))
)
```

RMPL: Reactive Model-based Programming Language
Compiling RMPL to HCA

- Hierarchical Constraint Automata (HCA): graphical specification language for control programs, in the spirit of StateCharts
- Writable, inspectable by systems engineers
- Directly executable by Control Sequencer
Compiling RMPL to HCA

 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))))

(legend)

MAINTAIN (EAR OR EBR)

- EAS (EngineA = Standby)
- EAF (EngineA = Failed)
- EAR (EngineA = Firing)
- EBS (EngineB = Standby)
- EBF (EngineB = Failed)
- EBR (EngineB = Firing)
- CO (Camera = Off)

MAINTAIN (EAF)

- (EAS AND CO)
- EAF AND EBS AND CO
Engineers reason about how to achieve state trajectories using models of system behavior.
System Model: Formal Descriptions of State Behavior

System Model describes behavior of each component:

– nominal and off-nominal behavior
– qualitative constraints
– probabilistic transitions
– costs/rewards

One state machine per component, operating concurrently
Concurrent Constraint Automata

**Engine Model**
- Nominal modes
- Firing
- Standby
- (eng_cmd = off)
- (thrust = zero) & (power_in = zero)
- (eng_cmd = standby)
- (flow_in1 = nom) & (flow_in2 = nom) & (eng_cmd = fire)
- Failed

**Camera Model**
- On
- Resettable
- Off
- (cam_cmd = turnon)
- (power_in = zero) & (shutter = closed)
- (power_in = nominal) & (shutter = open)
- (cam_cmd = turnoff)
- (cam_cmd = reset)

**Guarded Probabilistic Transitions**
- Modal constraints
- Modal rewards

**Fault Modes**
- thrust = full
- power_in = nominal
- flow_in1 = nominal
- flow_in2 = nominal

**Modal Rewards**
- $P_{\tau} = 99.9\%$
- $P_{\tau} = 0.1\%$
Translating CCA to Propositional Logic

- System Model captured as CCA
- CCA representation translates directly to clauses in propositional logic
- Logical representation is used by reasoning algorithm in Deductive Controller

\[
\text{mode = open} \implies (p_{in} = p_{out}) \land (f_{in} = f_{out})
\]

\[
\text{mode = closed} \implies (f_{in} = 0) \land (f_{out} = 0)
\]

\[
(\text{mode = open}) \land (\text{cmd-in} = \text{close}) \implies \\
(\text{next (mode = closed)})
\]

\[
(\text{mode = closed}) \land (\text{cmd-in} = \text{open}) \implies \\
(\text{next (mode = open)})
\]

…
Model-based Executive

Model-based Program
- Control Program
- System Model

Model-based Executive
- Control Sequencer
- Deductive Controller

System Under Control
- Observations
- Commands

State estimates
- Configuration goals
The Control Program is compiled into an executable form.

The Control Sequencer is responsible for generating, in real time, the sequence of configuration state goals prescribed in the Control Program.
The System Model is compiled into a form suitable for reasoning.

The Deductive Controller is responsible for estimating the most likely current state based on observations from the system, and issuing commands to achieve the configuration goals.
Model-based Executive

Model-based Program
- Control Program
- System Model

Model-based Executive
- Control Sequencer
- Deductive Controller
  - Estimation
  - Reconfig (Control)

System Under Control
- Observations
- Commands

Configuration goals
State estimates
Executing HCA - Step 1

Camera = Off;
Engine A = Standby;
Engine B = Standby

Control Program → State estimates → Configuration goals

System Model → System Under Control

Deductive Controller

Observations → Commands

Legend:
EAS (EngineA = Standby)
EAF (EngineA = Failed)
EAR (EngineA = Firing)
EBS (EngineB = Standby)
EBF (EngineB = Failed)
EBR (EngineB = Firing)
CO (Camera = Off)

Maintain (EAS or EBS)

Maintain (EAF)

(EAS AND CO) → EAS AND CO → EAR

(EAF AND EBS AND CO) → EAF AND EBS AND CO → EBR

(Control Program, System Model, Observation, System Under Control
Deductive Controller estimates state and issues commands to achieve goals

**Camera**

Goal: Off

- (power_in = zero) & (shutter = closed)

- (cam_cmd = turnon)
- (power_in = nominal) & (shutter = open)

- (cam_cmd = turnoff)

Obs: Shutter is closed
Deductive Controller estimates state and issues commands to achieve goals.
Deductive Controller estimates state and issues commands to achieve goals.

Engine A

Goal: Standby

Off

EBS
EAS
CO

Main

EBS
EAS
CO

Power Switch

Goal: Closed

Open
Closed
Stuck open
Stuck closed

Obs: power_out = nominal

(thrust = zero)

(power_out = zero)

(power_in = nominal) & (eng_cmd = standby)

(power_in = nominal) & (flow_in1 = nom) & (flow_in2 = nom)

(power_in = nominal) & (flow_in1 = nom) & (eng_cmd = fire)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power_out)

(power_in = power[out])
Executing HCA - Step 1

MAINTAIN (EAR OR EBR)
- EAS (EngineA = Standby)
- EAF (EngineA = Failed)
- EAR (EngineA = Firing)
- EBS (EngineB = Standby)
- EBF (EngineB = Failed)
- EBR (EngineB = Firing)
- CO (Camera = Off)

Legend:
- MAINTAIN (EAR OR EBR)
- EAS (EngineA = Standby)
- EAF (EngineA = Failed)
- EAR (EngineA = Firing)
- EBS (EngineB = Standby)
- EBF (EngineB = Failed)
- EBR (EngineB = Firing)
- CO (Camera = Off)

Not yet Achieved
- EAS
- EBS
- CO

Achieved
- EAR

(EAS AND CO)

Camera = Off; Power Switch = Closed

Control Sequencer

Deductive Controller

Control Program

State estimates

Configuration goals

System Under Control

Observations

Commands
Executing HCA - Step 2

LEGEND:
EAS  (EngineA = Standby)
EAF  (EngineA = Failed)
EAR  (EngineA = Firing)
EBS  (EngineB = Standby)
EBF  (EngineB = Failed)
EBR  (EngineB = Firing)
CO   (Camera = Off)

And so on…
Deductive Controller estimates state and issues commands to achieve goals

Achievement of Standby on Engine B proceeds similarly, in parallel…
Executing HCA - Step 2

LEGEND:
- EAS (EngineA = Standby)
- EAF (EngineA = Failed)
- EAR (EngineA = Firing)
- EBS (EngineB = Standby)
- EBF (EngineB = Failed)
- EBR (EngineB = Firing)
- CO (Camera = Off)

MAINTAIN (EAR OR EBR)
- EAS
- EBS
- CO

MAINTAIN (EAF)
- (EAS AND CO)
- EAS AND CO
- EAR

Engine A = Standby; Engine B = Standby
Executing HCA - Step 3

LENGEND:
- EAS (EngineA = Standby)
- EAF (EngineA = Failed)
- EAR (EngineA = Firing)
- EBS (EngineB = Standby)
- EBF (EngineB = Failed)
- EBR (EngineB = Firing)
- CO (Camera = Off)

**MAINTAIN (EAR OR EBR)**
- EAS
- EBS
- CO

**MAINTAIN (EAF)**
- (EAS AND CO)
- EAS AND CO → EAR

- (EAF AND EBS AND CO)
- EAF AND EBS AND CO → EBR

Engine A = Firing

Control Sequencer

Deductive Controller

System Under Control

State estimates

Configuration goals

Commands

Observations

Control Program

System Model

State estimates
Deductive Controller estimates state and issues commands to achieve goals

**Engine A**

**Goal:** Firing

- **Standby** (eng_cmd = standby)
- **Firing** (eng_cmd = fire)
- **Failed** (thrust = zero)

**Legend:**
- EAS (EngineA = Standby)
- EAF (EngineA = Failed)
- EAR (EngineA = Firing)
- EBS (EngineB = Standby)
- EBF (EngineB = Failed)
- EBR (EngineB = Firing)
- CO (Camera = Off)

**States and Conditions:**
- **Standby:** (power_in = nominal) & (flow_in1 = nom) & (flow_in2 = nom) & (eng_cmd = standby)
- **Firing:** (thrust = full) & (power_in = nominal) & (flow_in1 = nominal) & (flow_in2 = nominal)
- **Failed:** (thrust = zero) & (power_in = nominal) & (eng_cmd = standby)

**Additional Conditions:**
- (thrust = zero)
**Engine A**

Goal: **Firing**

(Deductive Controller estimates state and issues commands to achieve goals)

**LEGEND:**
- **EAS** (EngineA = Standby)
- **EAF** (EngineA = Failed)
- **EAR** (EngineA = Firing)
- **EBS** (EngineB = Standby)
- **EBF** (EngineB = Failed)
- **EBR** (EngineB = Firing)
- **CO** (Camera = Off)

**MAINTAIN (EAR OR EBR)**

**Valve A1**

**Goal:** **Open**

Obs: **outflow = nominal**

(inflow = outflow = zero)

(inflow = outflow = zero)

Achievement of Open on Valve A2 proceeds similarly, in parallel…

Powertrain:

0.01

(power_in = nominal) & (eng_cmd = standby)

0.01

(power_in = nominal) & (eng_cmd = fire)

0.01

0.01

0.01

0.01

inflow = outflow = zero

inflow = outflow = zero

Thrust:

0.01

(thrust = zero)

Goal:

*Closed*

*Open*

*Stuck open*

*Stuck closed*
Executing HCA - Step 3

Valve A1 = Open; Valve A2 = Open
Executing HCA - Step 4

LEGEND:
EAS (EngineA = Standby)
EAF (EngineA = Failed)
EAR (EngineA = Firing)
EBS (EngineB = Standby)
EBF (EngineB = Failed)
EBR (EngineB = Firing)
CO (Camera = Off)

MAINTAIN (EAR OR EBR)

MAINTAIN (EAF)

(EAS AND CO)

(EAF AND EBS AND CO)

System Under Control

Control Sequencer

Deductive Controller

Control Program

System Model

State estimates

Configuration goals

Observations

Commands

Engine A = Firing
Deductive Controller estimates state and issues commands to achieve goals

**Legend:**
- **EAS** (EngineA = Standby)
- **EAF** (EngineA = Failed)
- **EAR** (EngineA = Firing)
- **EBS** (EngineB = Standby)
- **EBF** (EngineB = Failed)
- **EBR** (EngineB = Firing)
- **CO** (Camera = Off)

**Goal:** Firing

- **Standby**
  - (eng_cmd = standby)

- **Firing**
  - (eng_cmd = fire)

- **Failed**
  - (thrust = zero)

- **Off**
  - (eng_cmd = off)

- **(power_in = nominal) & (eng_cmd = standby)**
  - 0.01

- **(flow_in1 = nom) & (flow_in2 = nom) & (eng_cmd = fire)**
  - 0.01

- **(thrust = full) & (power_in = nominal) & (flow_in1 = nominal) & (flow_in2 = nominal)**
  - 0.01

- **(thrust = zero) & (power_in = nominal)**
  - 0.01

**States:**
- **MAINTAIN (EAR OR EBR)**
- **MAINTAIN (EAF)**
- **(EAS AND CO)**
- **(EAF AND EBS AND CO)**
Executing HCA - Step 4

MAINTAIN (EAS OR EBR)

LEGEND:
- EAS (EngineA = Standby)
- EAF (EngineA = Failed)
- EAR (EngineA = Firing)
- EBS (EngineB = Standby)
- EBF (EngineB = Failed)
- EBR (EngineB = Firing)
- CO (Camera = Off)

MAINTAIN (EAF)
- (EAS AND CO) → EAR
- (EAF AND EBS AND CO) → EBR

Control Program → Control Sequencer
System Model
Deductive Controller
System Under Control
Observations
Configuration goals
Commands
State estimates
Executing HCA - Step 5

- (EngineA = Firing) achieved in this step
- maintenance condition violated, HCA block exited
What About Off-nominal Execution?

**LEGEND:**
- EAS (Engine A = Standby)
- EAF (Engine A = Failed)
- EAR (Engine A = Firing)
- EBS (Engine B = Standby)
- EBF (Engine B = Failed)
- EBR (Engine B = Firing)
- CO (Camera = Off)

Camera = Off; Engine A = Standby; Engine B = Standby

- **Control Sequencer**
  - State estimates
  - Configuration goals

- **Deductive Controller**
  - Observations
  - System Under Control
  - Commands

- **System Model**
  - Control Program
Model-based executive provides in-the-loop robustness

Camera

Goal: Off

- (power_in = zero) & (shutter = closed)
- (cam_cmd = turnoff)
- (power_in = nominal) & (shutter = open)

Resettable

- (cam_cmd = turnon)
- (cam_cmd = reset)

Obs: Shutter is open
Executing HCA - Step 1

Legend:
- EAS (EngineA = Standby)
- EAF (EngineA = Failed)
- EAR (EngineA = Firing)
- EBS (EngineB = Standby)
- EBF (EngineB = Failed)
- EBR (EngineB = Firing)
- CO (Camera = Off)

Control Sequencer
- Configuration goals
- State estimates

Deductive Controller
- System Model
- Observations
- System Under Control
- Commands

Camera = Resettable; Power Switch = Closed
Executing HCA - Step 2

Legend:
- EAS (EngineA = Standby)
- EAF (EngineA = Failed)
- EAR (EngineA = Firing)
- EBS (EngineB = Standby)
- EBF (EngineB = Failed)
- EBR (EngineB = Firing)
- CO (Camera = Off)

Control Sequencer

Deductive Controller

System Under Control

Observations
Commands

State estimates
Configuration goals

Camera = Off;
Engine A = Standby;
Engine B = Standby

Control Program

System Model

Camera = Off;
Engine A = Standby;
Engine B = Standby
Model-based executive provides in-the-loop robustness

Camera

Goal: Off

- (power_in = zero) & (shutter = closed)
- (cam_cmd = turnoff)
- (power_in = nominal) & (shutter = open)

Resettable

- (cam_cmd = turnon)
- (cam_cmd = reset)

Obs: Shutter is open
Executing HCA - Step 2

MAINTAIN (EAR OR EBR)

LEGEND:
- EAS (EngineA = Standby)
- EAF (EngineA = Failed)
- EAR (EngineA = Firing)
- EBS (EngineB = Standby)
- EBF (EngineB = Failed)
- EBR (EngineB = Firing)
- CO (Camera = Off)

MAINTAIN (EAF)

(EAS AND CO) → EAS AND CO → EAR

(EAF AND EBS AND CO) → EAF AND EBS AND CO

Camera = On;
Engine A = Standby;
Engine B = Standby
Executing HCA - Step 3

**MAINTAIN (EAR OR EBR)**

LEGEND:
- EAS (EngineA = Standby)
- EAF (EngineA = Failed)
- EAR (EngineA = Firing)
- EBS (EngineB = Standby)
- EBF (EngineB = Failed)
- EBR (EngineB = Firing)
- CO (Camera = Off)

**MAINTAIN (EAF)**

(EAS AND CO) → EARS AND CO → EAR

(EAF AND EBS AND CO) → EAF AND EBS AND CO → EBR

Camera = Off

Control Sequencer

Deductive Controller

System Model

Control Program

State estimates

Configuration goals

Observations

System Under Control

Commands
Model-based executive provides in-the-loop robustness

Camera

Goal: Off

(power_in = zero) & (shutter = closed)

Off

Resettable

(power_in = nominal) & (shutter = open)

On

Obs: Shutter is closed

(cam_cmd = turnoff)

(cam_cmd = turnon)

(cam_cmd = reset)

M A I N

E A S

E B S

C O

(EAS AND CO)

(EAS AND CO)

(EAF AND EBS AND CO)

(EAF AND EBS AND CO)

Ma i n
Executing HCA - Step 3

And execution of HCA proceeds normally…

Camera = Off
Desirable Architectural Features, Revisited

Model-based Program

Control Program

System

Simple state-based control specifications with explicit intent

Models that are writable/inspectable by systems engineers

Fault-awareness (in-the-loop recoveries)

Control Sequencer

Configuration goals

State estimates

Separation of state determination from control, with an explicit notion of state at the boundary

Deductive Controller

Estimation

Reconfig

(C)ontrol

Control system manages low-level plant interactions

Fault-awareness

System Under Control

Observations

Commands
Key Take-Away Points

- Single control architecture for both nominal and off-nominal execution
  - Off-nominal situations handled at different layers in the architecture, as appropriate.
  - Design trade: in which layer will a given failure be handled?
- State is central and intent is explicit
- Models are used directly in each layer of the architecture
- Details of the implementation are less important than the architectural features (principles)
  - A variant of this architecture could employ more traditional estimation and control techniques/algorithms, and still get benefit.
New Frontiers 4 Selection

• Select NF-4 from among:
  - Comet Surface Sample Return
  - Lunar South Pole-Aitken Basin Sample Return
  - Saturn Probe
  - Trojan Tour and Rendezvous
  - Venus In Situ Explorer

• No relative priorities among these are assigned.

• If the selected NF-3 mission addresses the goals of one of these, remove that one from the list.
Flagship Missions
(in priority order)

1. Begin NASA/ESA Mars Sample Return campaign: Descoped Mars Astrobiology Explorer-Cacher (MAX-C)

2. Detailed investigation of a probable ocean in the outer solar system: Descoped Jupiter Europa Orbiter (JEO)

3. First in-depth exploration of an Ice Giant planet: Uranus Orbiter and Probe

4. Either Enceladus Orbiter or Venus Climate Mission (no relative priorities assigned)
For More Information


BACKUP
The System Under Control

Battery → Power Switch → Camera

Battery

Power Switch

Camera

Fuel

Oxidizer

Valve A1

Valve A2

Valve B1

Valve B2

Engine A

Engine B
Example: Model-based Executive

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

![Diagram of Oxidizer and Fuel tanks with cross symbols indicating controls and statuses.]
Example: Model-based Executive

Mode Estimation

Oxidizer tank

Fuel tank

Deduces that thrust is off, and the engine is healthy

Mode Reconfiguration

Plans actions to open six valves and executes them, one at a time

Deduces that a valve failed - stuck closed

Mode Reconfiguration

Determines valves on the backup engine that will achieve thrust, plans needed actions and executes them.

Mode Estimation