Goal-Based Fault Management

2012 NASA Spacecraft Fault Management Workshop

Daniel Dvorak
Systems & Software Division
JPL / Caltech
Purpose of this Talk

Broaden our thinking about architectures for fault management

“A mind once stretched by a new idea never regains its original dimensions.”

Oliver Wendell Holmes Jr.
Outline

• Objective
• Assessment criteria
• Overview of two FM architectural styles
  – Monitor-Response-Inhibit style
  – Goal-based style
• Example of goal-based FM
• Compare and contrast
• Summary
FM Architecture Assessment Criteria

1. Emphasize *failure* detection over *fault* detection
   – more important to respond to loss of function, no matter how it occurs, than to a set of anticipated faults
   – Lack of faults ≠ health

2. Upon failure detection ...
   – allow unaffected activities to continue executing (graceful degradation)
   – decide responses based on context

3. Be compatible with flight software control architecture
   – Having different mechanisms for nominal control versus fault management creates an interface that is harder to engineer and validate
Monitors

- Each monitor detects a specific problem
- Monitors have tunable thresholds, e.g., ‘confidence’, ‘persistence’ and ‘decay’
- Thresholds are constant, not activity-specific
- A tripped monitor triggers a specific response (rule-like: “when x do y”)
**Monitor-Response-Inhibit (2 of 2)**

### Responses

- A response may kill a sequence and/or start a sequence
- Monitors & responses typically integrated after nominal control system working
- Often called “rule-based” or “autonomy”

### Inhibits

- Inhibit flag on monitors and responses can be set/reset by ground and by responses
• State variables are explicit
  – corresponding to states of the system under control

• Each goal represents ...
  – “a goal to be achieved”
  – acceptable behavior for a single state variable
  – a required condition

• Each goal has a success criterion that is...
  – specific to *that* activity
  – monitored for success/failure
• A goal may have supporting goals
  – To control x, must also control y
• Goals are the basis for operating the system
• Goal failure means that
  – a behavior is not acceptable
  – a required condition no longer holds
• How to respond to a goal failure is informed by the next higher goal
Context-Aware Failure Response

• A command sequence is just a sequence of commands; it does not represent intent

• When a failure occurs, the most appropriate response often depends on the context
  – i.e., on system state and on intent

• Goal-based operation enables cognizant control
  – Fault response decisions are based on context
A Goal Elaboration Hierarchy

- A top-level goal is elaborated to add all of its supporting goals
- Looking upward explains why a goal exists
- Looking downward explains how it will be achieved and/or what it requires
- Goal failure response is informed by the next higher goal
Goal Failure Response Animations

**Safing:**

Safe goal net

Done. Safe goal net installed, all previous goals removed

**Partial Goal Shedding:**

Parent goal decides to shed

Goal Fails

Done. Failed goal and its supporting goals removed.

**Re-elab. & Re-sched.:**

Parent goal decides to re-elab.

Go: Fail

Done. Failed goal and its support removed and replaced by new goals.

**Best-Effort:**

Keep trying to achieve!

Goal Fails

Done. No change because nothing better can be done.
Framework vs. Application Complexity

- **Simple**
  - Operations Complexity
  - Monitor-Response-Inhibit Architecture
  - Goal-Based Architecture
  - Model-Based Architecture

- **High**

Framework Complexity

Operations Complexity

Simple

High
Summary

• Single architecture for both nominal activities and fault tolerance

• All activities represented as goals on behavior of system under control

• Activity goals have supporting goals for every dependency (device modes, resources, necessary conditions, etc)

• All goals monitored during execution

• Goal failure, whether due to a hardware fault or environment, triggers response from goal elaborator