Background & Introduction to
Mission Data System
& Other Software Architectures

Selected slides from:
Daniel Dvorak,
Bob Rasmussen,
Al Sacks,
Nicolas Rouquette

Copyright 2000 California Institute of Technology
Material may be subject to pending patent.
Then:

- Big-budget missions spaced years apart
- Deep space missions tend to be one-of-a-kind
- Limited processor speed & memory, highly tuned flight software
- Little flight software reuse

Now:

- Better-faster-cheaper missions, monthly launches
- Still true, but they share common needs: pwr, comm, acs, nav, fault protection, ...
- Processor speed & memory no longer so limiting
- Need to apply s/w engineering resources more effectively
- Thus, MDS project initiated in April 1998 to rethink mission software lifecycle
MPF’s Fault Protection ⇒ a hand-crafted jewel

- Response functions are hand-written C functions
- Monitor designs vary per manager

Copyright 2000 California Institute of Technology
Material may be subject to pending patent.
DS1’s Fault Protection ⇒ the little engine that could...

⇒ Response functions are generated from statecharts
⇒ Monitor designs follow a uniform architecture

Copyright 2000 California Institute of Technology
Material may be subject to pending patent.
Themes

MDS Spans the Mission and System

- A Unified Flight, Ground, and Test Architecture
- An End-to-End Information System
- A Multi-Mission Enterprise
- Flexible & Reusable
- State Based
- Evolvable
Mission Data System

Unified Flight, Ground & Test
Data System Architecture for Space Missions

Objectives
- More effective use of software engineering resources
- Earlier collaboration of mission, system & software design
- Unified flight, ground & test architecture
- Reusable components & simpler interfaces
- Goal-directed operations
- Automation
- Evolvable to in situ exploration & other autonomous applications
- Customer-controlled complexity:
  - Simple to sophisticated, depending on mission requirements
  - Make it easy to exercise options

Key Architectural Themes
- Plan to migrate capability from ground to flight
- Formal representation and manipulation of state is central
- Express domain knowledge in models, not program logic
- Goal-directed operation specifies intent; simplifies operations
- Closed-loop control for real-time in situ reaction to events
- Fault protection is integral part of design, not an add-on
- Real-time resource management (for power, fuel, etc)
- Clean separation of state determination from control
- State uncertainty is acknowledged & used in decision-making
- Clean separation of data management from data transport
- Navigation and attitude control build from common base
- Upward compatibility through careful design of interfaces

Customers
- X2000 First and Second Deliveries
- OP/SP missions:
  - Europa Orbiter
  - Pluto Flyby
  - Solar Probe
- Space Interferometry Mission (SIM) — under study
- Potential application to Mars and other programs
- DSMS (Deep Space Mission System)

Copyright 2000 California Institute of Technology
Material may be subject to pending patent.
The 3 Meanings of "MDS"

- **MDS, the architecture:**
  - establishes a unified approach to flight, ground, and test systems for space missions
  - reflects a dozen architectural themes aimed at: simplified operations, adaptability, clean design, evolvable autonomy

- **MDS, the project:**
  - is managed by TMOD (Telecommunications & Mission Operations Directorate) at JPL
  - is a "change agent" at JPL defining processes and aligning resources, technology, and organizations
  - is working with first mission customer *Europa Orbiter* toward a November 2003 launch

- **MDS, the system:**
  - is a set of frameworks and examples to be *adapted* to meet the needs of each mission

---

Copyright 2000 California Institute of Technology
Material may be subject to pending patent.
8 MDS Themes
The Meaning of “Theme”

These are “themes” in the sense that they:
• have **broad impact** on operations or software structure
• are **notably different** than current practices

These themes are **not** novel ideas; they:
• are sensible design concepts drawn from control systems, robotics, software engineering, computer networking, artificial intelligence, etc
Theme 1: An Architectural Approach

Traditional Approach:
- Subdivide along conventional lines
  - Flight - Ground - Test
  - Design - Test - Operations
  - Engineering - Science
  - ACS - Nav - Power - Prop - Telecom - Thermal ...
  - Downlink - Uplink
- Apply customized solutions within each realm
- Integrate and iterate, integrate and iterate, ...
- Minimal re-use

MDS Approach:
- Construct subsystems from architectural elements, not the other way around!
  - Find the common problems and create common solutions
  - Tailor the general solutions to the particular problems
- Managing interactions is the foundation of a design
  - Find interaction mechanisms
  - Create coordination services
  - Use these common services rather than private function-to-function agreements

Copyright 2000 California Institute of Technology
Material may be subject to pending patent.
Theme 2: Ground-to-Flight Migration

MDS takes a unified view of flight and ground capabilities because of opportunity and need:

**Opportunity**
- more powerful flight computers can perform more functions
- capabilities situated on ground can be migrated to flight when operational characteristics well understood and routine
- migration reduces operational burden and may reduce demands on DSN (deep space network)

**Need**
- round-trip light-time delay will preclude earth-in-the-loop control for some missions:
  - Pluto fly-by will occur with a round-trip delay of 9 hours
  - autonomous landing on comet while avoiding gas jets

Copyright 2000 California Institute of Technology
Material may be subject to pending patent.
Theme 3: *State* is a Unifying Idea

- System state is formally analyzed to discover its constituent components
- Each is represented with a model, including effects from other states
- Measurements (with models) determine state
- Goals (with models) control state
- Communication, visualization, simulation, and many other functions also use state
Theme 4: Express Knowledge in Models

**Example Model Types**

- Relationships on state
  - Power varies with solar incidence angle
- Conditions on state
  - Gyros saturate above a certain rate
- Sequential state machines
  - Some sequences of valve operations are okay; others are not
- Dynamical state models
  - Accelerating to a turn rate takes time

**Uses for Models**

- Process a measurement
  - Tracker updates indicate that estimated turn rate is too high
- Prepare for the future
  - Sun occultation will occur soon
- Decide what actions to take
  - The thrusters cannot be used until they are heated
- ... or Not to take
  - Turning that instrument on now will use too much power
- Discover faults
  - It wasn’t supposed to do that

Copyright 2000 California Institute of Technology
Material may be subject to pending patent.
Theme 5: Goal-Directed Operation

- A goal specifies *intent*, in the form of *desired state*.

- A *goal* is a *constraint* on the *value* of a *state variable* during a *time interval*.

- Goal-directed operation is simpler because a goal is easier to specify than the actions to accomplish it.

- Goal-achieving modules (GAMs) attempt to accomplish submitted goals.

- A GAM may issue primitive commands and/or sub-goals to other GAMs.

- A GAM must either accomplish a goal or responsibly report that it cannot.
Theme 6: Closed-Loop Control

Problems:

- Open-loop control is both brittle and risky because it is "flying blind".
- Open-loop control forces ground operators to consider many failure scenarios when planning a sequence.
- Open-loop control precludes missions that demand quick reaction to unpredictable events.

Solution:

- Determine spacecraft state in real time from onboard measurements, not on time-delayed ground predictions.
- Employ closed-loop controls onboard to accomplish goals using continuously updated state feedback.

Copyright 2000 California Institute of Technology
Material may be subject to pending patent.
Theme 7: Integral Fault Protection

Anecdote:
• The day that they enabled fault protection in Cassini AACS they learned more about the spacecraft in one month than they had in the previous 6 months. [Ras]

Definition:
• Fault protection = fault detection, localization, reconfiguration, and recovery.

Approach:
• In MDS, fault protection will be an integral part of the design—not an add-on—because it is:
  – essential for robust control
  – extremely valuable for verification and debugging
Theme 11: Separate Data Management from Data Transport

**Problem:**
- Traditionally flight and ground data management has been tightly coupled with data transport making the design, programming, testing, and evolution harder for both.

**Solution:**
- Keep data management and transport cleanly separated
  - e.g. CCSDS packet format known only to data transport
- Elevate data products as entities in their own right:
  - exist as objects and files
  - can be used onboard
  - can be updated, summarized, aged, etc.
  - not necessarily destined for ground
- Data transport can access *any* data products, as needed
- Decoupling allows independent improvements

Copyright 2000 California Institute of Technology
Material may be subject to pending patent.
Top Level View of Architecture

State Knowledge

State Determination

State Control

Models

State

Actions

Measurements

Hardware Proxies

Sense
Act

Report

Telemetry

Goals

Coordinate
Elaborate

Telecommand

Copyright 2000 California Institute of Technology
Material may be subject to pending patent.
The Common Model
Common Model Structure and Use

- **Framework** is the collection of most core classes within the MDS architecture
  - Developed and maintained exclusively by MDS
  - Uniform (except for versioning) across MDS adaptations

- Each project does an **Adaptation** of the framework
  - Captures project requirements and scenarios
  - Extends framework classes to address functions and configurations specific to the project
  - Reusable extensions are generalized (if necessary) and moved to the framework

- Several **Deployments** of the adaptation are defined
  - These are the executable configurations to be used in various settings (test beds, flight...)

Copyright 2000 California Institute of Technology
Material may be subject to pending patent

Common Model
Framework

- Disciplines extend and customize the core infrastructure*
  - Partitioned as peers for modularity, acknowledgement of discipline vagaries, and the ability to aggregate functionality across disciplines as necessary

* Infrastructure
  - All of the classes embodying core, generic features, concepts, and services
  - Internally layered (hierarchical) to maximize reuse and uniformity, and to build more complex structure in manageable steps

Copyright 2000 California Institute of Technology
Material may be subject to pending patent.
• Infrastructure is divided (more or less) into ...

  - The State Based Architecture
    • Elements upon which most of the operational structure of the software is built

  - Services and Tools
    • Core elements that provide fairly conventional computing capabilities
Services
(Data Management & Transport)
• Persistent storage, retrieval and deletion of data objects
  – Built upon physical storage abstraction layer provided by OS
  – Standard serialization methods
  – Buffered access
    • Facilitate frequent access
    • Hold before committing NVRAM write cycles
  – Data base cataloging and queries by name or property

• Data Products
  – Framework for most DM content
    • Science instrument and engineering data
    • Comprised of metadata plus configurable content
    • Each type stored in cataloged collections
      – Collections of collections are possible
  – Both the standard storable unit (in simple collections) and the standard transportable unit (in transport-proxy collections)
More Data Management

- Value histories
  - A container mechanism supporting functions that produce values over time
  - Encapsulate a "back-end" interface to data management persistent storage and data transport.
  - Can be stored and transported as data products
  - Leverage the use of models to preserve continuous information using less storage space
  - Can also simply store a set of discrete value instances

Entries are combined and compressed as they age and are eventually deleted

- Name services
  - Global registry for name (character string) to value (integer) conversions
  - Along with standard serialization methods, eliminates a large portion
Services & Tools

Data Transport (DT)

- Encapsulation of space telecom links
  - Bi-directional transport
    - DSN ↔ S/C
    - Proximity space links (formations, orbiter ↔ lander, …)
  - Direct file/object transport
  - CCSDS, reliable transport protocols (e.g., CFDP), …
  - TMOD data delivery services
  - IPC with support for QoS on space links
    - Latency, errors, rate, volume, etc.

- Interface to telecom hardware
  - Management
  - Data streams
  - Radiometric data (Doppler and range) for navigation

- Data link management
  - Data rate selection
  - Relay / forwarding
Both DM and DT are visible but non-intrusive to applications
  - Can access storage and transport (via IPC) independently
  - Storage and transport use each other

Both DM and DT have a distributed deployment (resident on all nodes)
  - Unified design with corresponding flight and ground elements in each
  - Symmetric across the flight/ground interface
    • Anything you can send down you can also send up

Copyright 2000 California Institute of Technology
Material may be subject to pending patent.
DM and DT are controlled mainly via policies

- Some applications may change the contents of public collections, and...
- A few applications may interact directly via data transport, but...
- No applications (except policy managers) directly manipulate storage and transport functions
DM Policies

- DM policies establish a desired QoS for each data collection
  - Storage policies establish...
    - Level of persistence
    - Type and degree of compression
  - Transport policies (for transport-proxy collections) establish...
    - Schedule and timeliness
    - Priority
    - Required completeness and continuity

- Status of each data collection
  - Catalog of contents
  - Storage allocation status
  - Transport status per transportable data product
DT Policies

- DT policies establish a desired QoS for each link
  - Transaction policies establish…
    - Acknowledgement requirements
  - Comm policies establish…
    - Adaptation to link characteristics
      - E.g., data rate and volume transportable
    - Latency requirements

- Status of current link
  - Bit error rate (receiving side)
  - Expected completion time of link
  - CFDP file transfer status
    - Acknowledged versus unacknowledged transactions

Copyright 2000 California Institute of Technology
Material may be subject to pending patent.
What’s next for Nicolas?
System engineers take charge with Statecharts

Pre-Launch
- Design prototype
- Complete design
- As launched

Long-term maintenance
- Changing actions performed in a state
- Adding/deleting states and transitions
- New behavior

An STVM facilitates long-term maintenance because of:
- low-bandwidth requirements
- life-cycle continuity of system engineering practices
- no software programming required

Copyright 2000 California Institute of Technology. Material may be subject to pending patent.
STVM Internal Architecture

- Computes queries about topology, actions, constraints and tags

- Run-to-completion semantics (orthogonal states vs state machine)
- Transition selection algorithm
- Event management

- Generates events

- Constructs a Statechart objects from an XML Statechart definition

- Completion & execution events
- Exception signals

- Initiate actions & changes

Copyright 2000 California Institute of Technology
Material may be subject to pending patent.