Warts, Bumps, and Blemishes

Experimenting with Sensor Webs Using EO-1
2 March 2004

Stuart Frye
Mitretek Systems
EO-1 Systems Engineer
Contents

◆ EO-1 Sensor Web Functionality
◆ Mission Systems Configuration
◆ System Modeling for Autonomy Migration
◆ Interface Scripts and Glue Code
◆ Mission Planning Activities
◆ Ground System Accommodations/Upgrades
◆ Flight Software Changes
◆ Issues, Warts, Bumps, and Blemishes
◆ Lessons Learned
EO-1 Sensor Web Functions

Trigger Detection/Posting/Clearinghouse
- Detect/Locate Triggering Events
  - Hot Pixels
  - Cloud Coverage
  - Edges/Boundaries
  - Etc.
- Post Trigger Info, Send to Subscribers
- Use Raw Sensor Data, Derived Products, or Model Outputs as Source (Direct Broad-Cast/Rapid Delivery…)

Sentinel Monitoring/Scheduling/Status
- Watch for Triggering Events
- Parse Science Goals For Data Requests
- Negotiate Follow-up Observations with Participating Platforms
- Track Status of Data Acquisition/Delivery

Observation Execution/Processing/Assimilation
- Implement On-board Scheduling/Processing/Feature Identification/Diagnosis Software
- Coordinate Normal Operations with Experiment Activities
- Determine Tie-breakers for Competing Requests
- Perform Special Reacquisition Maneuvers/Sequences

New Triggers
Data Delivery Status
EO-1 Mission Systems

ACRONYMS

- ASIST: Advanced Spacecraft Integration and System Test
- CASPER: Continuous Activity Scheduling, Planning, Execution, and Replanning
- C&H: Command and Data Handling
- CMS: Command Management System
- EO-1: Earth Observing One
- EROS: Earth Resources Observing Systems
- FEDS: Front End Data Systems
- FORMATS: Flight Dynamics Orbital and Mission Aids Transformation System
- MOPSS: Mission Operations Planning and Scheduling System
- USGS: United States Geological Survey
- WARP: Wideband Advanced Recorder Processor

EO-1 Ground System

Flight Dynamics System

Ground & Space Network Scheduling

Ground & Space Network

USGS EROS Data Center

Mission Planning Systems

- MOPSS
- CMS
- Matlab
- Formats

ASIST

FEDS

Data Trending / Analysis

EO-1 Web Page

VirtualSat Simulator

C&H

WARP

Flight Software Lab

Science

Goal

Monitor

Interface

Glue Code

CASPER

Goal Encapsulation

Scripts
Automated Sequence Generation

- **Mission goals**
  - *E.g.* – *image Kilauea (Lat/Lon)*

- **To Command Sequence**

```
2003:233:16:49:57 CMD ACSETWHLBIAS(INERTIAL,X=0.341589,Y=1.1749,Z=-0.118046);
2003:233:17:56:57 CMD ACGOTOMANEUVER(ORBITAL,TIME=900,XLIMDEG=0.02,YLIMDEG=0.062699,…);
2003:233:18:07:06 CMD I_SETFPEPOWER(POWER_MASK=5);
2003:233:18:07:06 CMD YHEASTBY;
2003:233:18:07:16 CMD YHEASETSWIR(GAINA=1,GAINB=1,GAINC=1,GAIND=1,…);
2003:233:18:07:26 CMD YHEASETVNIR(VNIRALV8,VNIRBLV8,VNIRCLV8,VNIRDLV8);
2003:233:18:11:06 CMD I_CONFIGFPE(CONFIG_COMMAND=16908); …
2003:233:18:17:06 CMD BCMMODESCRS422;
2003:233:18:17:16 CMD WRMSREC(IDWS=65535,IDWV=65535,…);
2003:233:18:17:54 CMD I_SET_FPE_DG(DURATION=-1);
...```
Uses Model of Activities

Resources

Activity: Science Image Acquisition

States

Uses 14 files; uses XXX memory

Other Activities

requires target pointing

These models are then combined to model the world as it changes due to activities
Earth Observing-1

EO-1 Sensor Webs/GSFC System Engineering Seminar

### Hyp State

<table>
<thead>
<tr>
<th>Hyp State</th>
<th>Mode</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcqAndUnb</td>
<td>Standby</td>
<td>Standby</td>
</tr>
<tr>
<td>All Cvr0</td>
<td>Close</td>
<td>Close</td>
</tr>
<tr>
<td>BfFrmmode0</td>
<td>Standby</td>
<td>Standby</td>
</tr>
<tr>
<td>Eclipse</td>
<td>Standby</td>
<td>Standby</td>
</tr>
<tr>
<td>GroundStat View</td>
<td>Standby</td>
<td>Standby</td>
</tr>
<tr>
<td>IlmStat</td>
<td>Standby</td>
<td>Standby</td>
</tr>
<tr>
<td>X-Band</td>
<td>Standby</td>
<td>Standby</td>
</tr>
</tbody>
</table>

### Day collect

- **Warp mode**
- **Hyperion Preparation**
- **Target in view**
- **X-band Ground Station**
- **WARP file count**
- **WARP data volume**

### Night collect

- **Downlink**

<table>
<thead>
<tr>
<th>Downlink</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-band</td>
<td>Standby</td>
</tr>
<tr>
<td>Ground Stat</td>
<td>Standby</td>
</tr>
<tr>
<td>WARP file count</td>
<td>Standby</td>
</tr>
<tr>
<td>WARP data volume</td>
<td>Standby</td>
</tr>
</tbody>
</table>
CASPER Planning

- CASPER can implement nominal procedures through decompositions (similar to scripts)
  - *In order to image: do x, then y, then z...*

- CASPER can also perform planning “from scratch” via search
  - *If want ACS-mode state variable = standby, consider adding an activity that changes ACS-mode (then the requirements of these activities may be new conflicts,...)*
  - *Most commonly used search framework “iterative repair”*
Activities, Constraints, Repairs

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Property that must hold for plan to be valid</th>
<th>Must always use less power than available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conflict</td>
<td>Violation of a constraint</td>
<td>Current plan uses more power than available from 18:00-18:30</td>
</tr>
<tr>
<td>Repair Method</td>
<td>Modification to plan that may remove conflict</td>
<td>Delete activity using power during conflict</td>
</tr>
<tr>
<td>Repair Choice</td>
<td>Which activity to delete</td>
<td>Delete largest user?</td>
</tr>
</tbody>
</table>
Constraint Resolution Tree

Conflict Type:
- Unground Parameter
- Undetailed Activity
- Mismatched Decomposition
- TimeLine
- Unplaced Activity
- Violated Temporal Constraint
- Open Temporal Constraint
- Violated Dependency

Repair Method:
- Ground
- Detail
- Redetail
- Add
- Move
- Delete
- Lift
- Place
- Abstract
- Disconnect
- Connect
- Apply Dependency

Method Choices:
- Start Time Interval
- Start Time
- Duration
- Activity
- Constraint
- Activity
- Constraint
- Activity
- Activity
- Activity
- Activity
- Activity
- Activity
- Abstract
- Disconnect
- Connect
- Apply Dependency

EO-1 Sensor Webs/GSFC System Engineering Seminar

March 2, 2004
Repair Algorithm

Start
(if conflicts exist and user time-limit not exceeded)

Select a conflict

Select a repair method
move

Select an activity

Select a start time

Perform the action, collect the new conflicts, and repeat
Interface Scripts and Glue Code

- **PERL Scripts handle traffic between SGM and EO-1 MOC via formatted Email running on Secure Shell**
  - Scripts send target Lat/Lon to Matlab and return computed target In-view times to SGM
  - Matlab interfaces converted from GUI versions to command line callable routines

- SGM-generated observation requests are ingested to MOPSS

- Target information and maneuver files exported from MOPSS are encapsulated into CASPER Goal Files

- Tie-breaker selections received from SGM are encapsulated in commands and uplinked to spacecraft

- CASPER replans for triggered target and executes new plan on-board
New Mission Planning Activities

- **Experiment Time Slots Need to be Integrated into Commercial Observation Schedule for Every Experiment**
- **Placeholder Target and Comm Requests Are Inserted to Pre-Populate Schedule**
- **SGM-generated Records Are Ingested and Placeholders Overwritten**
- **Image Sequences Are Exported as Input to Goal File Generation Scripts for CASPER-Scheduled Requests**

- **Exported Sequences Are Removed from MOPSS Command Load**
- **Overlap Between Exported CASPER Sequences and Command Load Sequences Are Verified/De-Conflicted**
  - Including Verifying Continuity of Scene Information from Exported Sequences in Load Reports for Downstream Antenna Operations, Science Data Processing, and Scene Tracking Systems
- **Need to pick which comm contact to use to load/jump to new on-board code to avoid other overlapping operations on processor**

**Don’t forget to push the blue button at 8:07GMT**
Ground System Accommmodations/Upgrades

- Created new procedures for sending sensor web triggers to spacecraft, loading new code on-board, jumping to new executables
- Modified command uplink acknowledgement scheme and timeout settings to handle large code uploads
- Modified command database for new autonomy commands
- Modified telemetry database for new autonomy telemetry
- Modified Systems Test and Operations Language (STOL) procedures to perform code load, checksum, uncompress, jump, goal/script activation, WARP reset
- Modified max slew rate from .25 to .433 deg/sec (Re-image scenario)
- Increased number of retransmit entries in FEDS command queue
- Upgraded trending system to pickup new telemetry mnemonics

No, not THAT blue button!
FSW Overview (Block Diagram)
WARP Software Architecture

- Memory Scrub Task (MS)
- Health & Safety Task (HS)
- MSSP I/F Task (MP)
- PM I/F Task (PM)
- CFBIU I/F Task (CF)
- Memory Dwell Task (MD)
- Checksum Task (CS)
- Recorder Management Task (RM)
- Software Manager Task (SM)
- Time Code Task (TC)

Software Bus (SB)

VxWorks / Tornado (OS)

- 1773 RT Task (RT)
- MSSP Driver
- PM Driver
- CFBIU Driver
- 1773 RT-Driver

Legend:
- Interrupt-Driven Device Driver
- Newly Developed Task for EO-1 WARP
- Re-Used Task from MIDEX/MAP
Integrated “Plug and Play”, using SCL as adapter

Here is how we implemented it on EO-1, an existing on-orbit satellite, as an experiment.

EO-1 TLM Channels VC0 & VC3
  2 APIDs for SCL Control
  2 APIDs for SCL real-time control

SCL Commands
  (to C&DH M5 via WARP Remote Terminal)

SCL Script Tasks
SCL Command Tasks
SCL Telemetry Tasks
CASPER & Science Task

Livingstone Task

Onboard diagnostic tool

EO-1 TLM Channels VC0 & VC3
  2 APIDs for SCL Control
  2 APIDs for SCL real-time control

New

Onboard planning and scheduling tool

VxWorks / Tornado (OS)
Flight Software Lab

- Developed capability to reload WARP Flight Software kernel and patch to boot from new image using hijacked existing command
- Developed C&DH patches (next page)
- Integrated Spacecraft Control Language (SCL) and CASPER spacecraft autonomy software with WARP flight code
  - Developed utilities for encapsulating executables into S records for memory load STOL commands
- Upgraded VirtualSat to simulate additional command, telemetry, and event message traffic
- Implemented remote access for integration work via (Tight) Virtual Network Computing
- Implemented file transfers for code loads via Secure Shell
- Developed ability to compress and decompress executable code loads to reduce uplink bandwidth requirements
- Procured and integrated two additional test strings
  - 2 C&DH Mongoose 5, 2 WARP Mongoose 5,
  - 2 VirtualSat simulators, 1 Spare Mongoose 5

Now I see why they didn’t fly that board!
On-Board Changes to C&DH

- Software Routing updates to allow Commanding from the WARP
- Telemetry Filter Table modifications to accommodate CASPER/SCL Telemetry Downlink and On-board Recording
- New Telemetry Statistics Monitor (TSM) to automatically enable sun maneuver avoidance TSM upon daylight entry every orbit

It didn’t work that way in the lab!
On-Board Changes to WARP

- Reloaded entire WARP code image and jumped to it via patch
- Modified Memory Dwell task and S-band playback function in WARP Flight Code to read science data into RAM from near-line bulk storage
- Created various SCL and CASPER-related tasks
- Hijacked telemetry packets and commands for SCL and CASPER use
- Loaded new CASPER, SCL and cloud assessment algorithm on-board

- Added Event Messages for status reporting
- Modified checksum configuration on WARP for upload verification
- Increased WARP Watchdog timeout to prevent reset when booting to new larger code
- Turned Off CPU hogging and changed Memory Dwell task check-in error to an event – had caused warm restart
- Implemented a decompression utility on-board based on zlib library inflate function

Explain to me again why I can’t playback science data over S-band or run memory diagnostics with CASPER running
System Engineering Issues

- **CASPER knows spacecraft state and resources**
  - *Doesn’t do navigation, orbit propagation,…*
  - *Doesn’t do momentum management/maneuver planning*
  - *Has to coordinate file naming conventions with Command Load observations*
  - *Changeover from Command Load to CASPER control*

- **Better coordination required because more complex activity sequences are being undertaken**
  - *Operational sequences are not independent*
Warts (1 of 3)

- **FSW lab hardware not identical to flight hardware**
  - WARP Flight Processor has 256Mbytes RAM, but breadboard in lab has 32M memory for integration work – limits use of full on-board memory
  - Off-line WARP bulk memory cards not procured for EO-1 lab (>$1M) limits testing for image data file manipulation code
  - Insufficient memory in Flight Software Lab Breadboard caused several month delay in integration effort
  - Sensors and Mechanisms simulated using VirtualSat
- **Cannot duplicate on-board dynamics in lab (e.g., CPU starvation)**
- **Unexpected spacecraft reactions encountered during experiments**
- **On-orbit debugging required**
- **Had to use outgassing periods every 16 days to run experiments**
  - Always a stretch to define scope, schedule support, deliver tested code and unzip/jump/verify procedures in time for uplink
Bumps (2 of 3)

- Code loads to testbeds in FSW Lab slow at first - sped up by implementation of ICEPROMS and/or Ethernet on Mongoose boards
- Takes 3-4 days to uplink code loads to spacecraft
  - 15-20 ground station contacts
  - TDRS not reliable for large uplinks – can only use ground stations
  - 6Mbyte code loads to spacecraft compress by about 6-1
- Encountered problems verifying large uplinks
  - Not enough time to do full dump and compare
  - Using checksums was labor intensive and discrepancies hard to isolate
  - Made for some exciting tests….
- WARP reboots during dumps causes dump flag to hang on C&DH
  - Had to stuff WARP dump bit to YES, then send abort to clear C&DH flag
  - Still ran experiments on non-verified code – Oh Well!
On-board Cloud Detection takes 15 minutes to run on-board
- Not sufficient for look-ahead/assess/retarget scenarios
- Next load of FSW will allow selectable readout of hyperspectral bands and selectable readouts of particular rows of the image data file to speed up

Special care has to be taken to avoid invoking on-board memory operations during command load event windows
- No code loads, script updates, dumps, jumps, or other activation/deactivation memory operations during WARP Record or Playback events

Crashed WARP once – memory starvation issue
- Spacecraft was under CASPER control
- Crash occurred during image sequence – Watchdog check-in
- Left spacecraft maneuvered with instruments on and covers open
- Had to recover manually during next communications contacts
Lessons Learned

- **Build excess CPU and memory capacity into Flight Segment**
  - Enables sensor web/autonomy improvements post-launch

- **Include at least 2 flight processors on-board in future designs**
  - Can do development work without disturbing C&DH operations
  - If 2nd processor is not executing new FSW properly, reboot to old code

- **Build ground FSW Lab with identical hardware to Flight Segment**

- **Minimize time spent on development of support tools and utilities during early part of software effort**
  - Concentrate on primary functionality until better tools would save time

- **Learn through failure if it’s safe to do so – if you wait until you’re 100% sure of success, you may never get anything done**

- **Setup safeguards to auto-recover via command load after crashes are encountered during experiments**

- **Need to setup process for delivery of science data from experiments - problematic in commercial data sales setting**