Flight Software
Ground System Impacts

Mark G. Walker

Ground System Architectures Workshop
April 1, 2008

Contributors:
Judy Kerner, Larry Miller, and Phil Schmidt, The Aerospace Corporation
James Kramer, Integral Systems, Inc.
Outline

- Motivation
- Flight software impacts on ground systems
- Areas for improvement
- Next steps
Motivation

- Spacecraft flight software is increasing in size and complexity
- Flight software has major impact on ground systems
- More development time is spent on software vs. hardware issues
- Greater complexity and capability drive rethinking system design assumptions, both ground and space
## Flight Software Major Impacts

### Flight Software

- **Changing processor types and architectures**
  - Memory organization
  - Memory dump and reprogramming telemetry and command interfaces
  - Upload formats
  - Non-standard table structures

- **Increased autonomy**
  - On-board maneuver schedulers
  - Dynamic tasking
  - Constellation-based control
  - Maintain on-board state / configuration

### Ground Impact

- **Changes to core memory management functions**
  - Memory mapping and compare
  - Upload utilities
  - Downlink data formats and conversions (1750, IEEE, …); custom conversions and calibrations
  - Custom table readout algorithms

- **Custom tools**
  - Create and upload maneuver tables
  - Format and upload goals
  - Evaluate autonomy performance
  - Provide “observability” to software actions
### Flight Software Major Impacts (continued)

#### Flight Software

**Increased fault handling complexity and autonomy**
- Autonomous spacecraft reconfiguration in response to faults
- Threshold uploads and response configurations
- Software faults as well as hardware

**Variations in spacecraft “Product Lines”**
- Flight software changes
  - Even if it’s “just another ___ bus”
- Software customization to meet requirements
- Software evolution (often arbitrary)

#### Ground Impact

**New requirements**
- Manage fault handling configurations
- Provide “observability” of automatic responses
- Trying to identify anomalies through “alarm storms”

**Unexpected changes**
- Changes needed to accommodate unexpected variations in product line architectures
- Often occur late in program, which increases cost
## Flight Software Major Impacts

<table>
<thead>
<tr>
<th>Flight Software</th>
<th>Ground Impact</th>
</tr>
</thead>
</table>
| **Stored program languages**  
- Uploadable macros to customize software after launch | **More operator requirements**  
- Compile and validate uploads  
- Monitoring and reporting |
| **Proprietary interfaces**  
- Satellite manufacturers use proprietary telemetry and command interfaces  
- Standards (e.g., CCSDS) not yet widely embraced  
- No higher layer standards | **Multiple interfaces must be supported**  
- Satellite-specific telemetry decommutation and command formats |
| **Increasing payload processing**  
- Late changes to space-ground partitioning | **Late changes to ground system requirements**  

---

**Continued...**
Areas for Improvement

- Communication between flight and ground teams
  - Ground is usually expected to accommodate flight
    - Often ground impacts are not known by flight software teams
  - Resolve the “culture clashes”
    - Improve collaboration between flight and ground software developers
  - Develop standard methods of data delivery (e.g., via satellite attribute databases)

- Flight software development process maturity
  - Tight coupling with satellite hardware makes application of standard development processes difficult
    - Often results in late changes that impact ground system and architecture
  - Bring into the flight software development domain the same rigor that we’ve seen accomplished on the ground
Areas for Improvement (continued)

- Coordination between flight and ground architectures
  - Requirements and modeling need to be extended to cover flight software interactions with ground systems
  - Replace local flight code optimizations with system optimizations

- Support for multiple missions and mission types
  - Ground software often optimized for a specific application
  - Architect across a variety of spacecraft and flight software versions
    - COTS software benefits from a wider base of supported types and versions
    - Consider multi-mission architectures

- High-level “operational” standards
  - Standards are often discussed at GSAW, but at a low-level (e.g., communication and networking standards)
  - Abstract ground interfaces to on-board software
Operational Standards

- Spacecraft operations are often similar, but the details vary widely
  - Ground software often architected as a point solution
- A *Spacecraft Abstraction Layer* can provide standard interfaces for common spacecraft operations
  - Memory management, thermal tasks, battery operations, and maneuvers
- Allows “quick fit” of existing Common Ground System Functions to new bus types or flight software versions
- Apply standard interface approaches like those used in other industries
  - Robotics
  - Network management

<table>
<thead>
<tr>
<th>Common Ground System Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Spacecraft Abstraction Layer</strong> (customized for each S/C)</td>
</tr>
<tr>
<td>T&amp;C Interface Layer</td>
</tr>
<tr>
<td>Infrastructure Layer</td>
</tr>
</tbody>
</table>
Next Steps

- Architect ground systems to minimize impact of flight software
  - Accept that flight software may be different for each spacecraft
- Architect ground systems to be evolvable
  - Flight software can change over the mission
    • Before delivery and on-orbit (with uploaded patches/macros)

How?

- Begin to address *Flight Software Ground System Impacts* through:
  - Working group at GSAW2009
  - Email discussion group to collect issues and share solutions