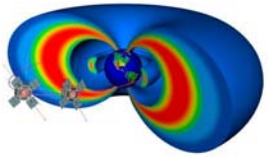


RBSP Radiation Belt Storm Probes

Managing the Impacts of Flight Software Changes on the Ground Software Element (The RBSP Experience)

Mark Reid

RBSP Flight Software Lead

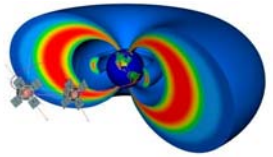
Johns Hopkins University, Applied Physics Lab

Mark.Reid@jhuapl.edu

(443) 778-2667

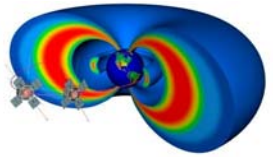
© 2012 by *JHU/APL. Published by The Aerospace Corporation* with permission.

Agenda



- Introduction
- Proper Requirements Management
- Control of Interface Definitions
- Key Engineering Roles
- Architectural Decisions
- Conclusion / Questions

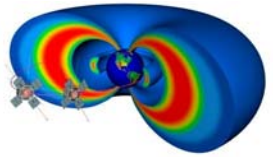
Introduction / Agenda



- **RBSP is a NASA mission to study near-Earth space radiation.**
- **The mission consists of two nearly identical spacecraft.**
- **Each spacecraft hosts 5 instrument suites, with a total of 10 unique science instruments.**

- **This presentation will discuss 4 areas of system engineering that helped RBSP minimize the system wide impact of software changes.**
  - Requirements Management
  - Interface Definitions
  - Key Engineering Staff
  - Architectural Decisions

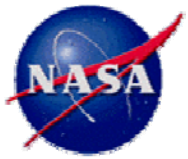
Requirements Management (Know Your System)



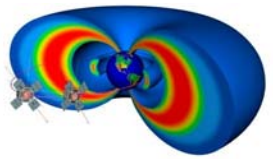
- **What is your system?**
  - End-To-End Mission? Observatory? Spacecraft? Flight Software (FSW)? Ground System? Ground Software (GSW)?
  - How do your requirements fit with everyone elses?

- **How are your requirements documented?**
  - Requirements Flow Down
  - Traceability
  - Verification

- **How are changes to these requirements managed?**
  - Documented Change Requests
  - Change Control Board (CCB)

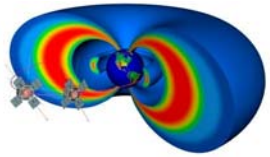


The RBSP System



- **Two Spacecraft**
  - 1 Mission Operations Center (MOC)
  - 2 Prime Strings of Workstations (RBSP-A and RBSP-B)
  - Plus Backup systems
- **Five Instrument Suites per Spacecraft**
  - 5 Science Operations Centers (SOC)s
- **Both Flight Software (FSW) and Ground Software (GSW) Requirements and Test Cases are Documented in Telelogic DOORS.**
  - Provides traceability of flow down and verification
- **Requirements changes were tracked in Atlassian's JIRA Tool**
- **Changes to baseline document are approved by a software CCB**
- **56 FSW requirements issues addressed in 5 document revisions**

Interface Definitions
(Know Your Interfaces)

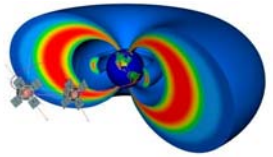


- **What other systems are coupled to (or impact) your system?**
  - How can a change in someone else's requirements impact you?
  - How can a change in your requirements impact someone else?

- **How are these interfaces documented?**
- **You need to be able to see the forest as well as the trees**
  - Identify areas of coupling between systems
  - Document these areas of coupling as clearly as possible with words, diagrams and tables

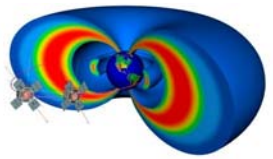
- **How are interface changes managed?**

RBSP GSW / FSW Interface



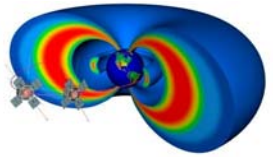
- **Documented in a Ground Software (GSW) to Flight Software (FSW) Interface Control Document (ICD)**
- **Documents RBSP's use of:**
  - CCSDS Uplink Command Formats
  - CCSDS Downlink Telemetry Formats
  - CCSDS File Delivery Protocol (CFDP) for playback of recorded data
  - Onboard Memory Management (Including objects)
  - Events and Command Status Formats
  - Instrument Time-tagged Command Buffers
- **Changes are tracked in JIRA and approved by CCB**
- **45 ICD issues addressed in 6 document revisions**

Key Engineering Staff (Know Your Roles)



- **Who is Responsible for the Various Aspects of the Software, and at What Level?**
- **Who Owns the Data Formats?**
- **Who Defines the Interface Protocols?**
- **How are These Roles Documented?**

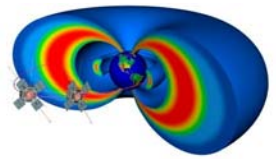
Key Roles on RBSP

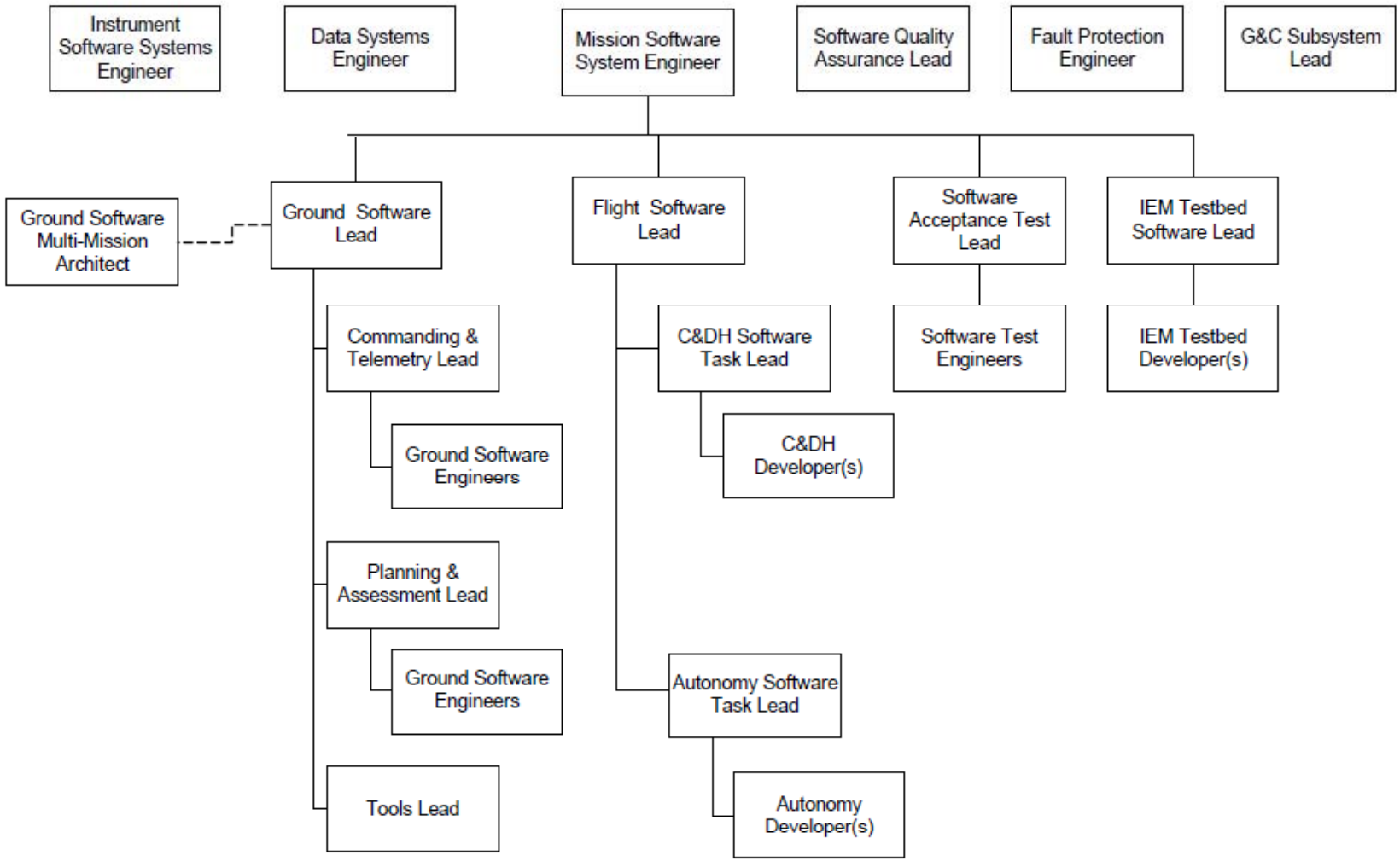


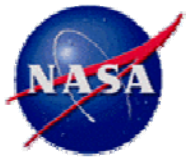
- **Documented in the Software Development and Management Plan (SDMP)**

- **Mission Software Systems Engineer**
  - Responsible for coordinating, planning, and directing the development and test of the RBSP software.
- **Data Systems Engineer**
  - Supports the Systems Engineering team to develop and manage the end to end data flow.
- **Flight Software Lead Engineer**
  - Leads FSW team and is responsible for spacecraft software elements
- **Ground Software Lead Engineer**
  - Leads GSW team and is responsible for ground software elements

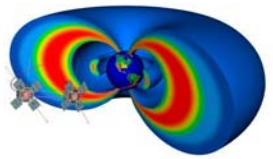
Example Org Chart







Architectural Decisions (Know Your Design)

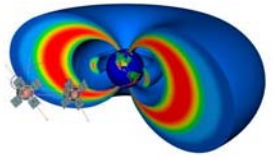


- **Minimize Coupling**
  - Whenever possible work to keep the elements of the system as loosely coupled as possible.
  - Minimize the amount and scope of information one system element needs to know about another.

- **Create Cohesive System Elements**
  - Each system element should be as cohesive as possible, performing just its defined task.

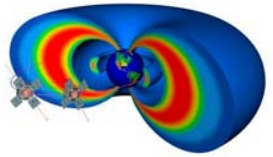
- **DON'T**
  - Cross protocol boundaries
  - Over specify to the point of tying developer's hands

RBSP's System Architecture



- **"Bent Pipe" system for "Decoupled" operations**
- **Instrument teams work directly with their instrument**
- **The spacecraft and ground system are transparent or invisible wherever possible**
- **Instrument commands are not contained within the spacecraft database in the MOC, but are instead maintained at each individual instrument team's SOC**
- **Critical instrument telemetry can be alarmed in MOC**

Conclusion / Questions



- **In order to effectively manage the impacts of changes to the flight requirements or interfaces, everyone needs to:**
  - Know Their System
  - Know Their Interfaces
  - Know Their Roles
  - Know Their Design
- **Questions?**