Solid State Recorder Data Organization Software and Functional Emulator for Mission Planning and Early Integration

Workshop on Spacecraft Flight Software 2014

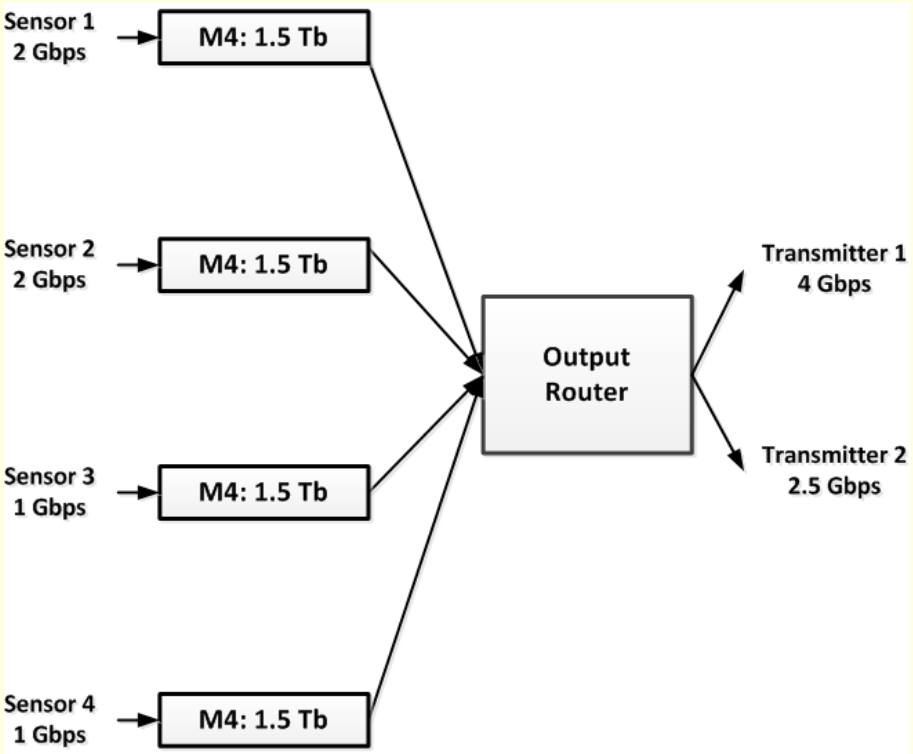
Michael Koets
Space Science and Engineering Division
Southwest Research Institute

Introduction

- Approach to software development for components within a complex software system
  - Complex functionality encapsulated behind a narrow, well-defined API
- Example Applications
  - Data link layer for inter-satellite radio network
  - Solid state data recorder
- Benefits
  - Early integration with application software
  - Responsive to changes in required functions or interfaces
  - Early testing of flight software components
  - Enforced abstraction of component

SSR Application Overview

- High Performance Solid State Recorder
  - Multiple Tb flash memory
  - Multiple data streams and storage boards
  - Input data at Gbps
  - Output data at Gbps
  - Simultaneous record and playback
- Flash Memory Constraints
  - Slow programming rate
  - Asymmetric access rates
  - Block organization

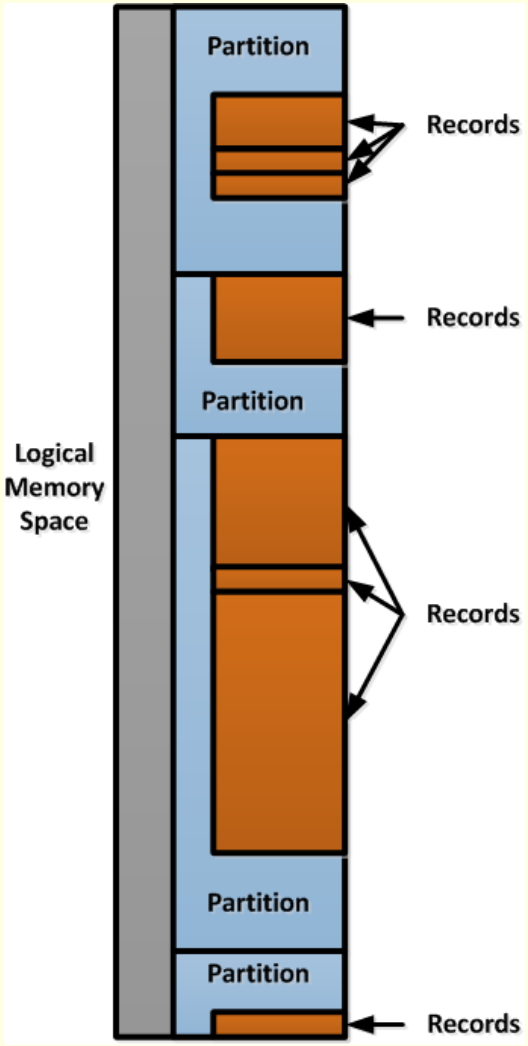


Data Organization Software

- Data Organization Software Functions
  - Associates application level data items with physical storage locations
  - Manages addresses for recording, playback, and erasure
  - Provide control interface to spacecraft application
  - Provide status information and telemetry
- Executes on SSR single board computer
  - Coordinates activities of flash memory boards
  - Interface to application software
    - Physical communications channel (i.e. MIL-STD-1553)
    - Integrated software on a single processor

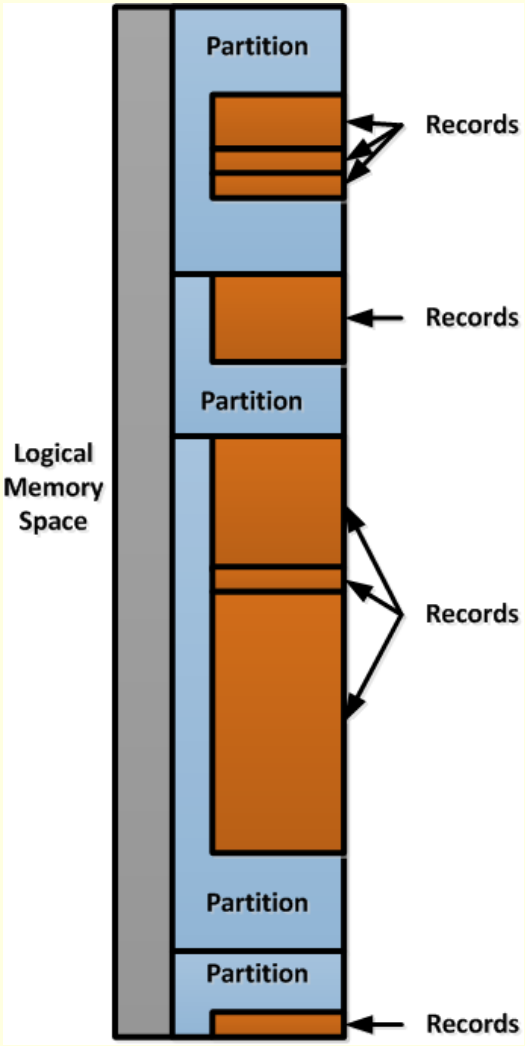
Data Organization Approach

- Usage Scenarios
  - FIFO
  - Simplified File System
- Manages Multiple Independent Data Stores
  - Typically one data store for each sensor
  - Analogous to multiple drives
- Partitions
  - 1-4 partitions
  - Define contiguous region of logical storage
  - Partitions managed as FIFOs
  - Erasure limited to oldest data in a partition
- Records
  - Contiguous regions within a partition
  - Data stored to and read from records
  - Random access addressing within a record for data readout
- Storage Granularity
  - Minimum size of addressable storage unit
  - Several MB (i.e. 48 MB)
  - Logical address space: 4096 storage units



Data Organization API

- Data organization operations available to applications:
  - Create New Record
    - Open a new record in the specified partition
  - Write Data to Record
    - Store data to the specified record
  - Close Record
    - Terminate recording to the specified record and close the record
  - Read Record
    - Read some or all of data from specified record
    - Data is sent to the ORB
  - Erase Data from Record
    - Partially erase data from specified record
  - Delete Record
    - Completely erase and remove specified record
  - Status Information
    - Current activities
    - Directory listing



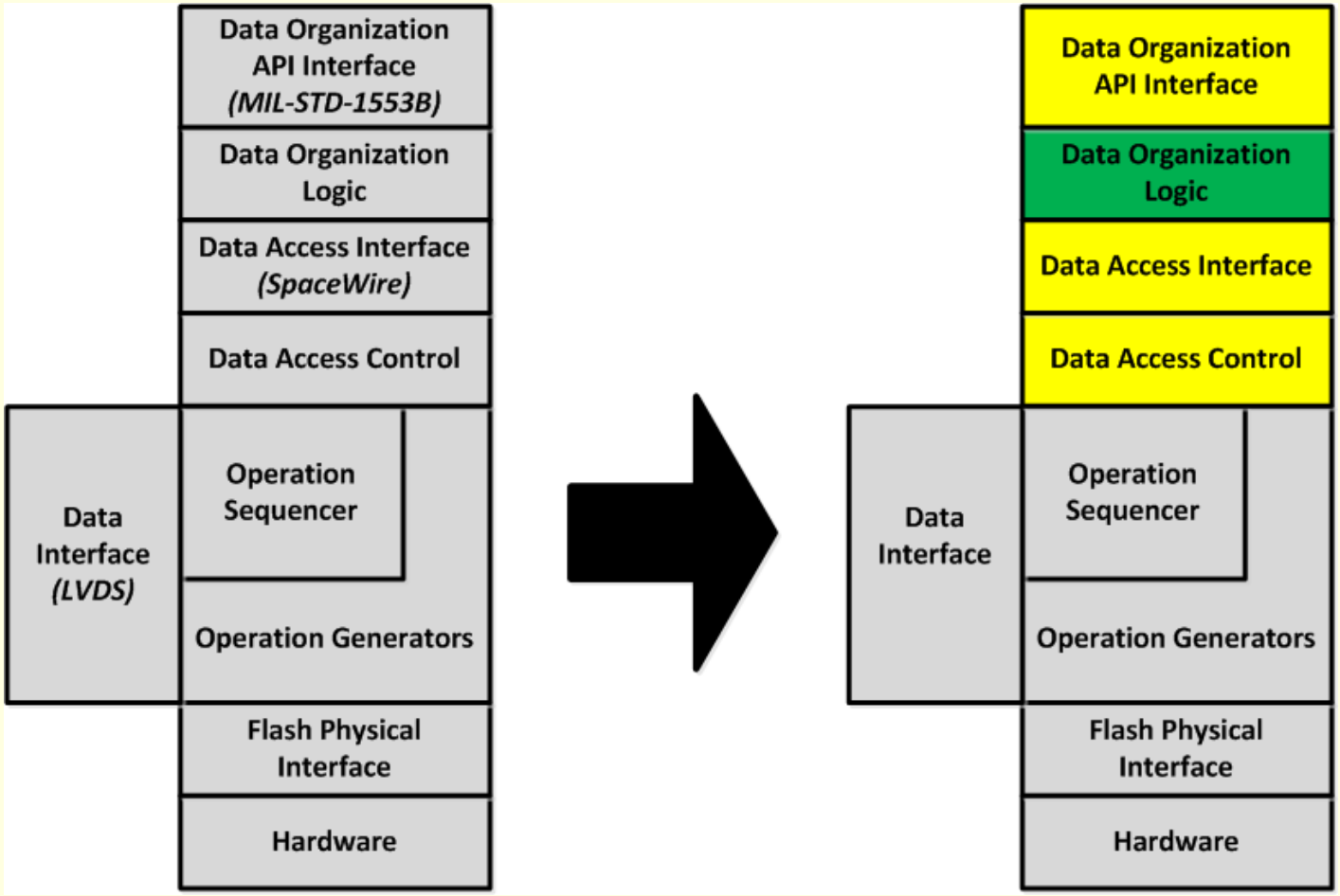
Other Data Organization Functions

- Wear Leveling
  - Flash memory has a limited endurance to programming and erasure
  - Software attempts to cycle through available physical memory to avoid excessive use of specific regions
- State Logging
  - Must recover software state after reset
  - Continuously stores updated state information
  - Log-oriented file system approach
    - Updated copies of state information distributed throughout physical memory
- Bad Block Management
  - Flash memory subject to detectable, localized failures
  - Software must detect such failures and map around them
  - Transparent to application software
    - Consistent, continuous logical address space

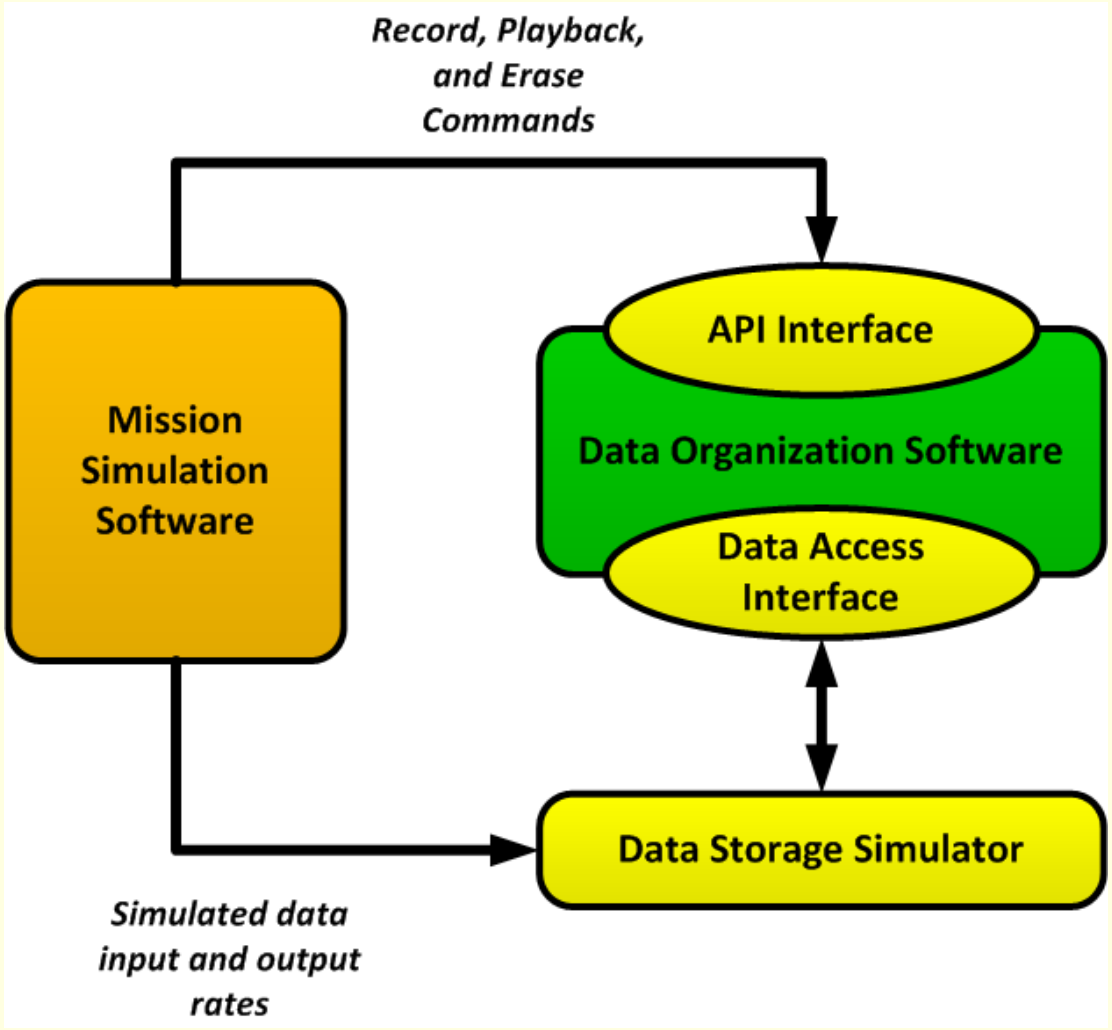
Software Emulator

- Software tool supports integration of SSR into mission design
  - Executes in PC environment
  - Model mission activities and characterize SSR performance
  - Plan partition structure and recording sizes
  - Provide platform for early integration of mission level flight software
  - Aid discovery of additional interface capabilities that enhance SSR operation
- Software Structure
  - Data organization logic implementation
    - Flight code
    - Implements data organization operations
  - Simulation classes model SSR behavior in response to high level operations

Stack View of Emulator

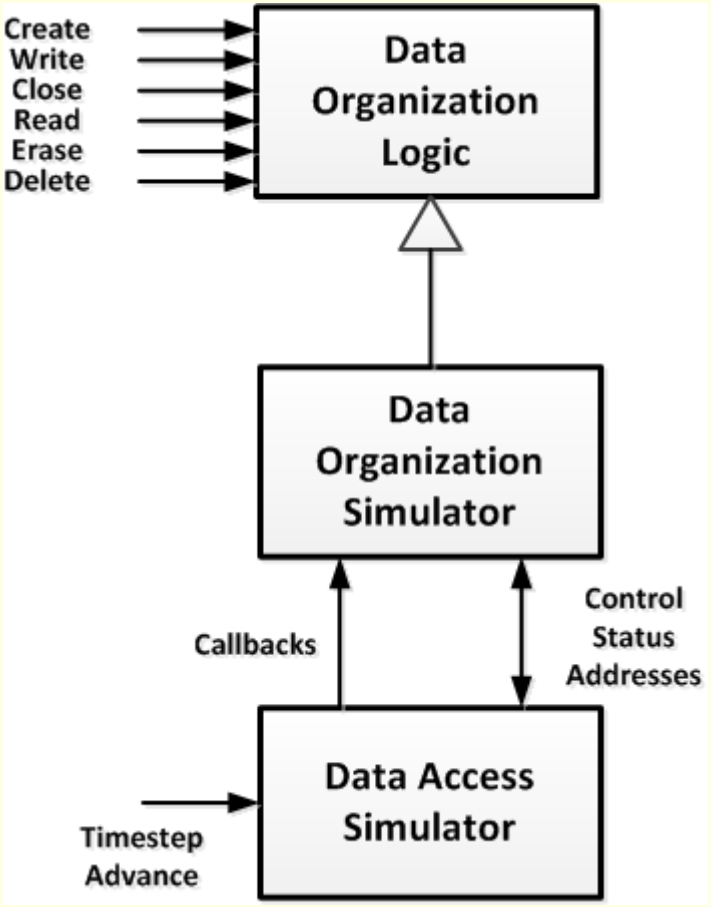


Simulation Environment



Software Structure

- ANSI Standard C++ Code
- Data organization simulator class inherits from flight data organization class
  - Shares internal data structures and algorithms
  - Provides API to application
- Data access simulator class connects to data organization simulator
  - Rate and capacity information configured at instantiation
  - Provides memory board control methods
  - Makes callbacks to data organization software
  - Controls simulation timing
- Code structure supports HWIL validation of data organization software on embedded platform



Simulating Control Interfaces

- Data organization simulator object provides methods implementing API calls
- Control actions synchronized to simulated data flows via discrete time simulation step
  - Time step size trades simulation fidelity against execution time
- All internal software data structures use flight software
  - Status responses and telemetry reflect in-flight behavior
- Key Use Cases
  - Validate application command sequences
  - Evaluate API changes or command overlays
  - Facilitate rapid debugging in a PC environment

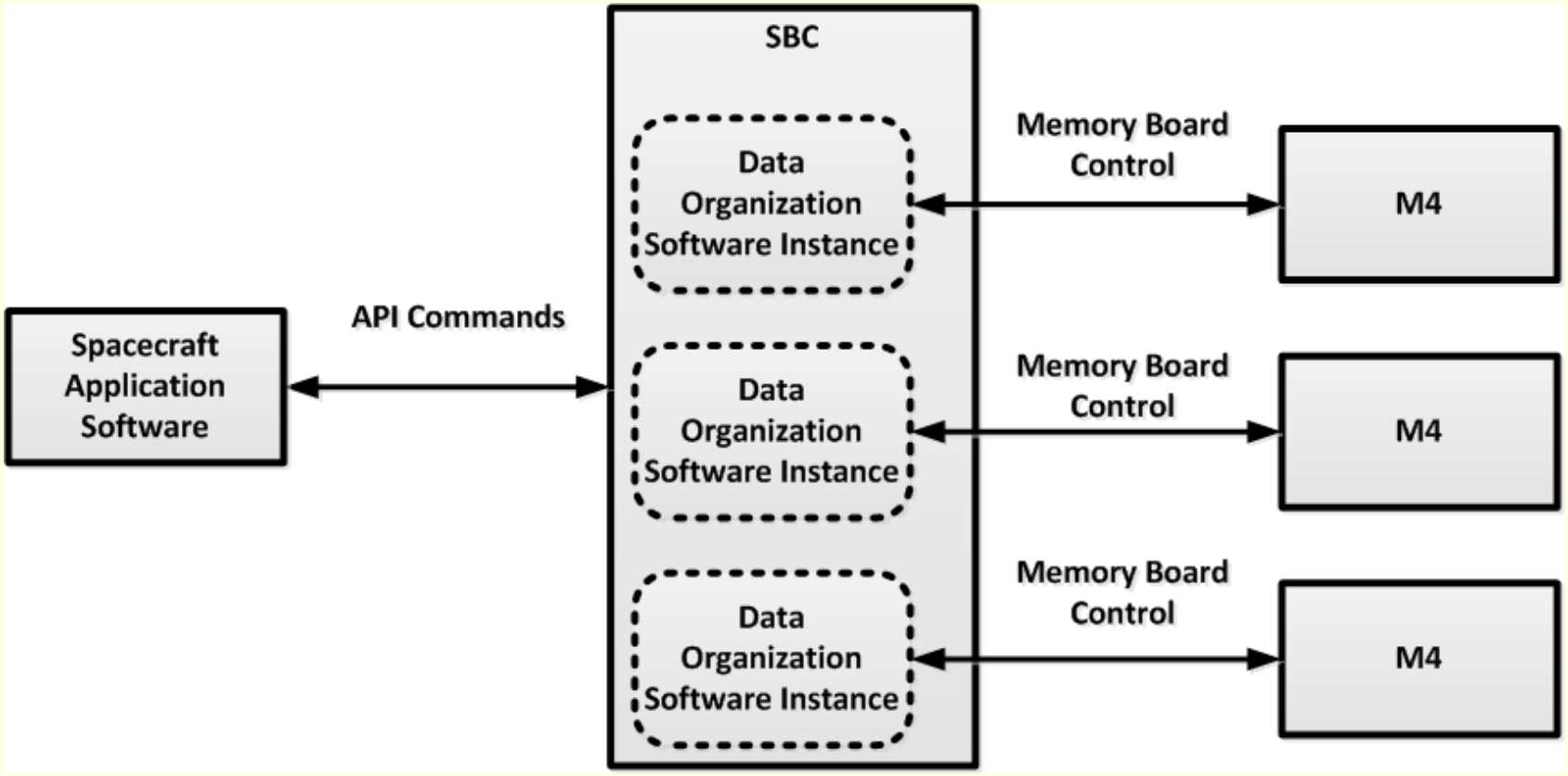
Simulating Data Flow

- Data flow simulation at reduced fidelity
  - Data flow quantities are metered
  - No actual data transfer or storage
  - Data flow modeled as aggregate data transfer over a time step
    - Source and sink behavior
    - Flow control activity
- Exercises software interfaces to memory boards
  - Memory board control methods
  - Callbacks
- Detection of invalid data flows
- Quantitative metering of data storage resources

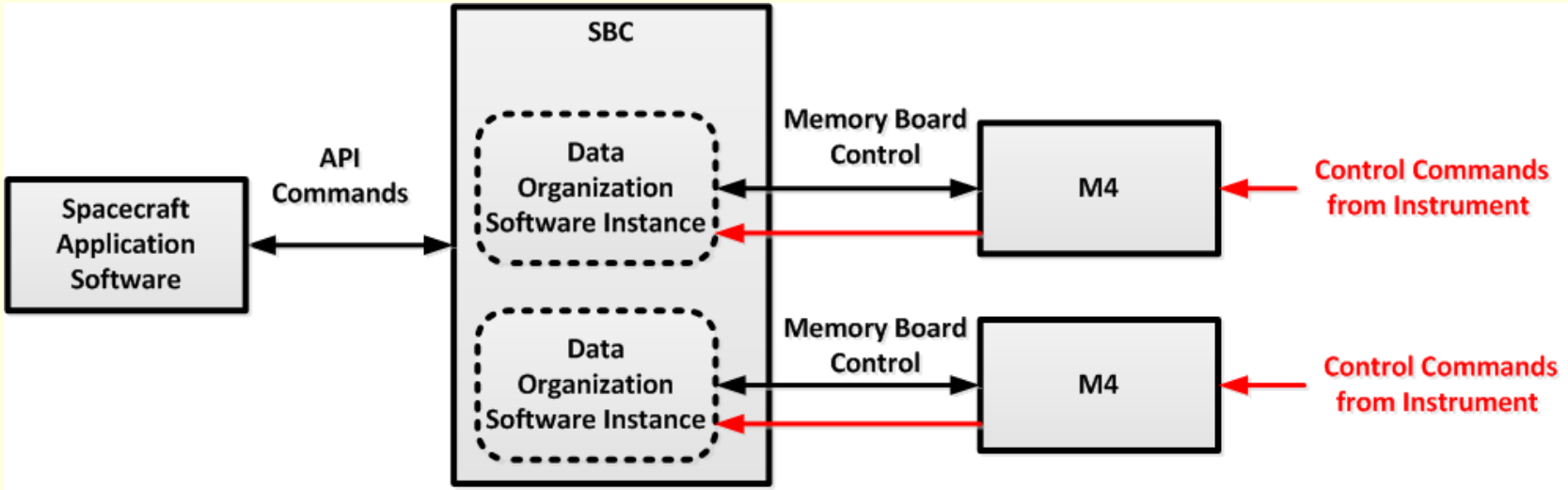
Mission Design

- Explore scheduling of mission activities
  - Observation times
  - Concurrent observations by multiple sensors
  - Data retrieval and transmission
- Monitor data rates and quantity of stored data to confirm plan is feasible
- Instrumentation
  - Standard API status reporting as in flight
  - Memory status visualization
  - Easily extended to custom instrumentation via object oriented inheritance

Case Study: Multiple Stores



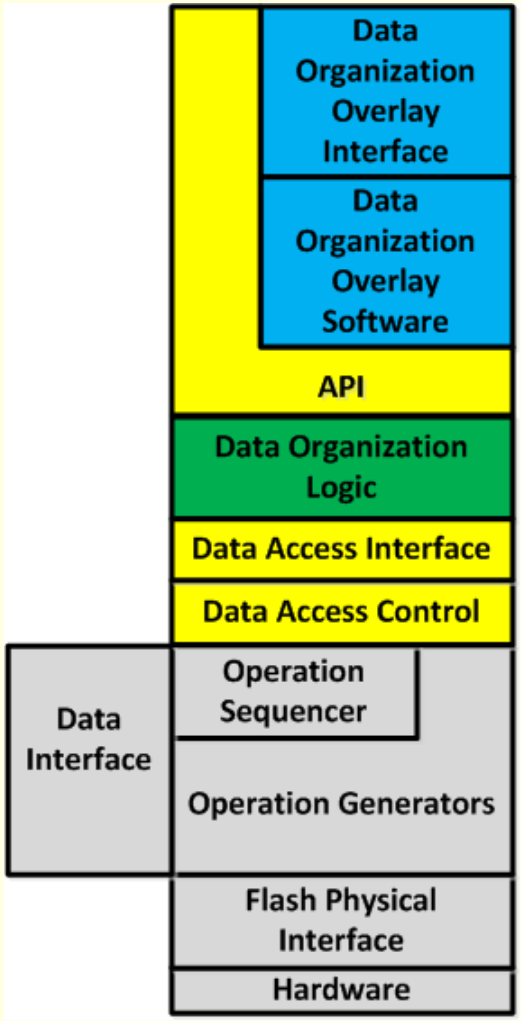
Case Study: Modifying the Control Interfaces



- Provide means for instruments to initiate recording
- Memory board relays commands from instruments to SBC
- New interrupt-driven interfaces between memory boards and data organization software

Case Study: Automatic Readout

- Provide higher-level control functions
  - Example: Automated readout sequencing and erasure via priority queue
- Extend SRR functionality via overlay software
- Overlay provides additional command interface
- Overlay interacts with existing API
- Direct API commands remain available



Conclusions

- Prototype of complex, encapsulated software
- Use object oriented design methods and language features to provide a testable version of flight software
- Early test and debugging of flight software
- Early integration with mission plan
  - Confirm software meets mission needs
- Early integration with mission application software
  - Extend software function or interfaces