Enabling Cross-Platform Satellite Applications with XML

GSAW 2005

Paul André
John Cosby
Greg Edwards

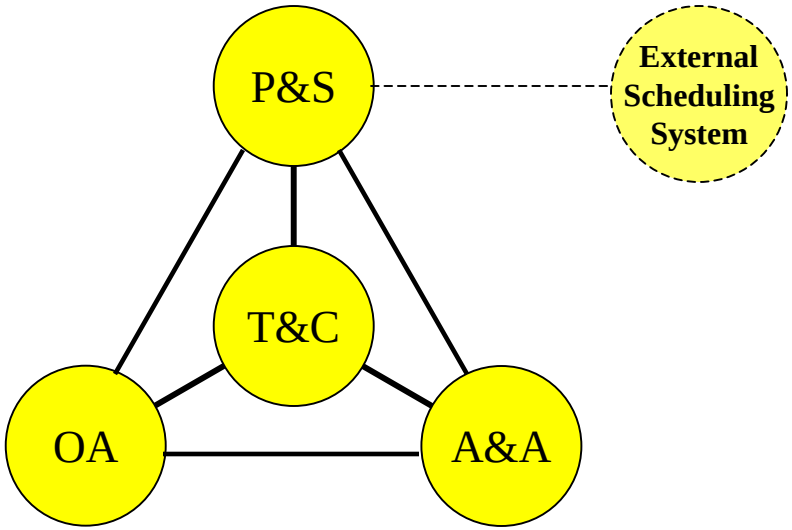
Science Applications International Corporation
20201 Century Blvd.
Germantown, MD 20878

Case Study Integrating Satellite GS Applications

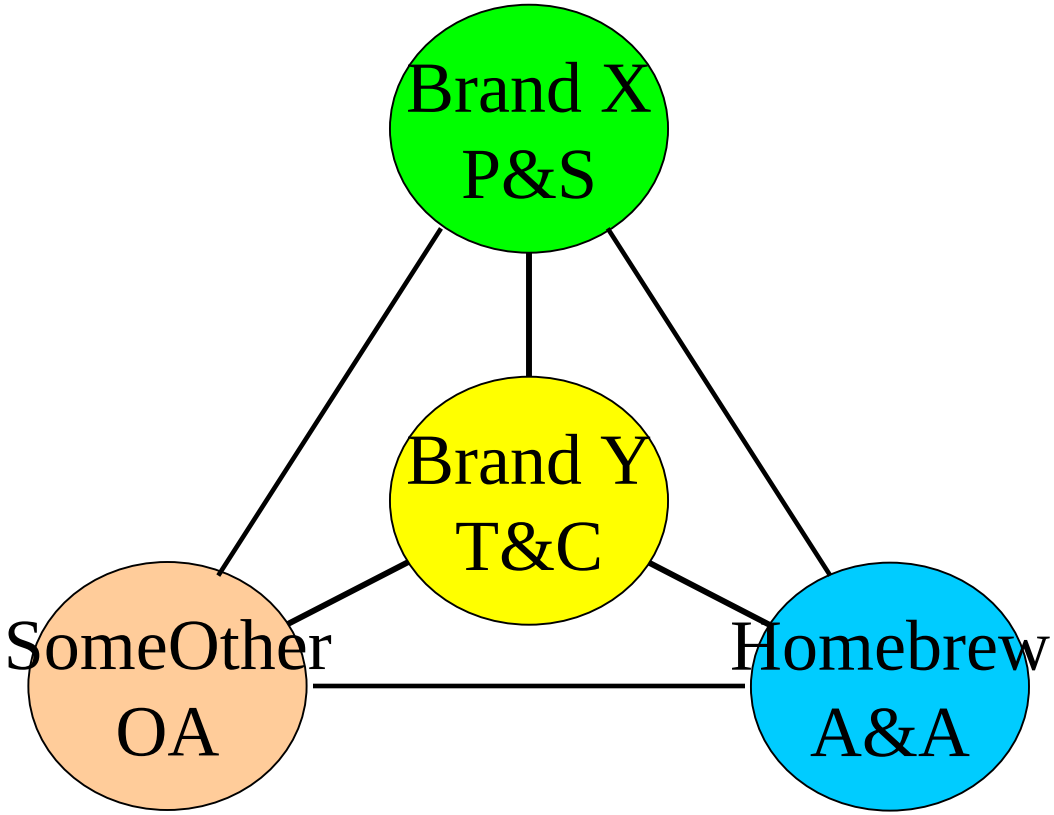
- Our experience is primarily based on the development of the Mission Planning & Scheduling (MPS) Subsystem on the CCS-C Program.
- We'll look at the architectural approach we used to enable interfaces with other CCS-C Subsystems
- We'll discuss the various techniques used to implement the interfaces
- We'll go through the pros and cons of those techniques
- Then we'll discuss how we we're able to extend an XML based interface to a different T&C product subsequent to CCS-C

Satellite Ground Systems Have Components that Have To Communicate…

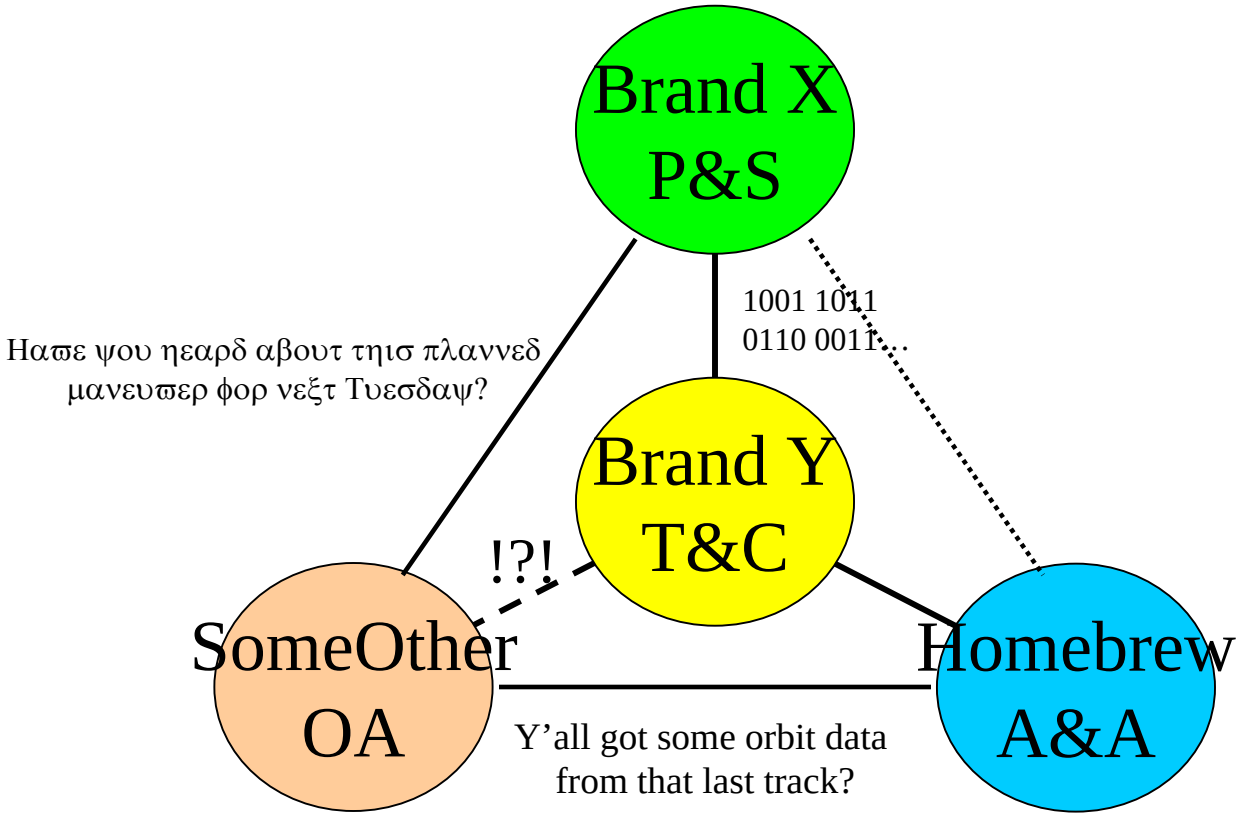
- Planning and Scheduling (our specialty) needs to talk with Telemetry and Commanding
- Telemetry and Commanding needs to talk with Archiving and Analysis
- Orbital Analysis needs to talk with Planning and Scheduling, and so on, and so forth…



They Aren't Always By The Same Vendor…



And Dey Don’ Always Talk Da Same, or even know the others exist…



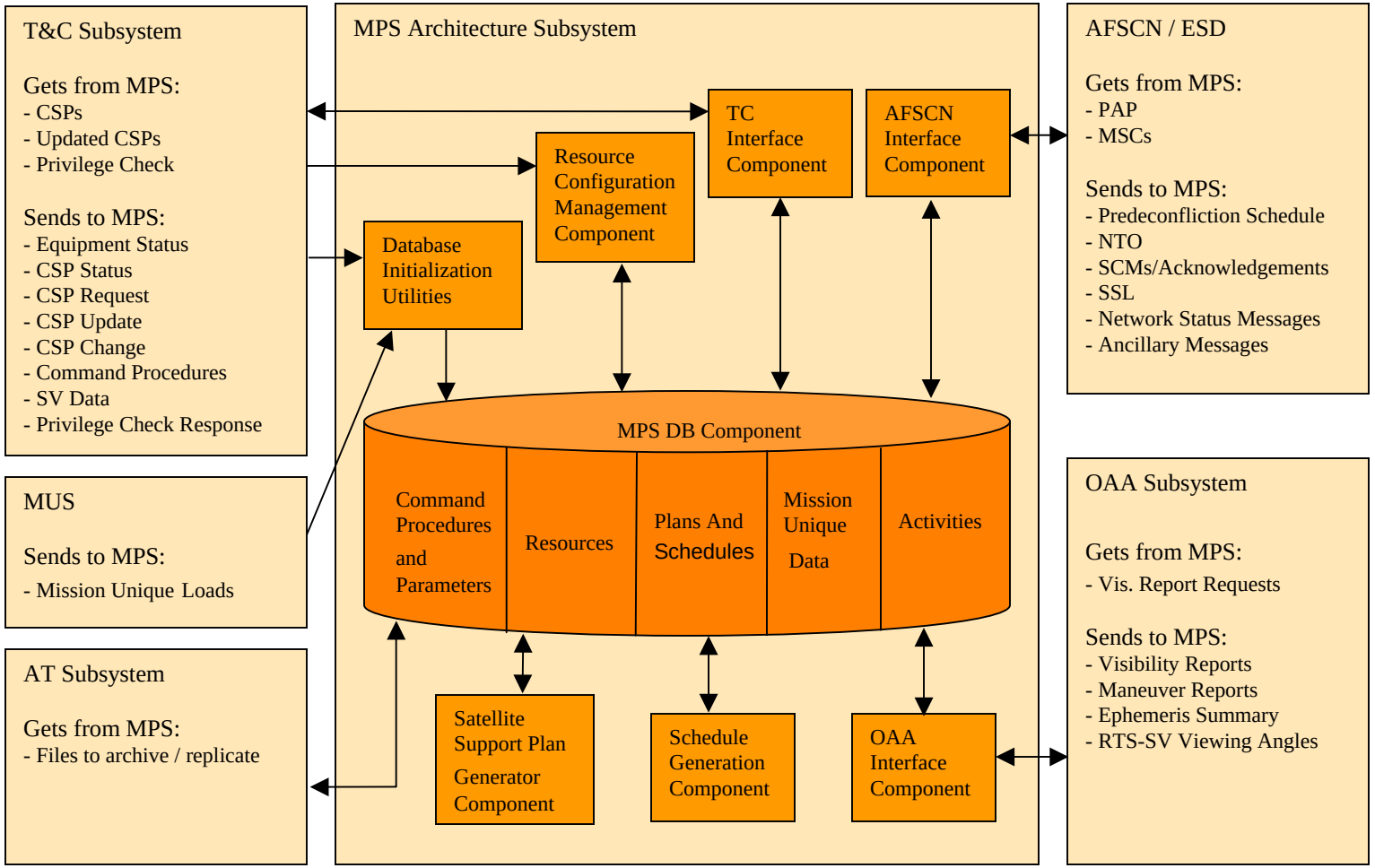
Putting These Together Can Be A Significant Challenge

- Similar to other COTS Integration efforts
  - Things are easier if you only use one vendor, but they may not meet all your needs
  - If vendors don't guarantee interoperability (and they don't), the integrator has to glue things together (a major project in itself)
- The investment means you're stuck with what you build for a while
  - Hacks live on for a loooong time

We'd like to share some lessons

- SAIC built the Planning and Scheduling segment of CCS-C, interfacing with ISI's OAA and T&C COTS
  - Integral Systems has an integrated set of COTS for satellite ground systems
    - OASYS is one of the leading OAA packages
    - EPOCH 2000 is a capable, mature T&C system
  - Mission Planning and Scheduling had some unique requirements – we built software to meet them
- The interfaces have been a learning experience
- Data-driven interfaces using XML schemas are flexible, extensible, and easier to maintain.

SAIC's MPS Architecture



Modular External Interfaces

- Our system encapsulated each external system interface to insulate us from change.
  - AFSCN, T&C, OAA were the main ones
  - Change to each was hidden from the rest of the system
- Communication internally was through the database, triggered by event-based messaging
  - Interfaces pull their data from the DB, trigger events to initiate functionality
- Data transfers outside MPS through reports, messages, and shared files
  - Mechanism internal to each interface

General Types of Interfaces

- There are three different general types of interfaces in systems like these…
  - File-based – one side generates a file, the other detects the file, parses it, and extracts information
  - API-based – one side publishes an interface specification, the other writes software that uses it directly
  - Message-based – one side publishes a set of messages that will be sent and received
    - Primitive versions are socket-based with binary or text format messages
    - More modern versions use HTTP and XML
    - Bleeding-edge – WSDL and SOAP

MPS-OASYS – File-based

- ISI's OASYS product generates human-readable files
  - Format based on customizable report specification
  - Never meant as a data exchange mechanism
- In CCS-C, Orbital Analysts generate reports using OASYS; Mission Planners ingest the reports into MPS
  - Visibility
  - Maneuvers

File-Based Report Issues

- Not automatic
  - Humans in the loop – can be overcome
    - ISI has batch request API for automated report generation
- Formatting bugs
  - Human-readable != machine-readable
    - Humans are much more flexible at parsing meaning out of arcane formats in their area of expertise
    - Computers need defined relationships; tweak the report format, break the parser
  - OASYS lets users tweak report formats easily – nice for humans, tough for computers
- Result – interface is relatively breakable, not suitable for real-time or interactive interfaces

MPS-T&C – API-based

- Wrapped interfaces with ISI's T&C subsystem in their own component of the MPS architecture – isolated from change
- Wrote a library for T&C to link in to invoke MPS services on request
  - Request schedule for a block of time from MPS
- Well suited for near-real-time and real-time systems

API Issues

- Languages
  - Our software in C++, their software in C++ and Java – cross-language programming is a known problem, solvable, but still can cause headaches for well-known reasons
- Data
  - Configuration data in Oracle tables
    - We provided libraries and views for some, APIs for other functions
  - Messages in XML, static or dynamically-generated
    - Up-to-the-second context where required
    - Files generated where distribution is required
    - Schemas, not DTDs

API Issues

- Bugs
  - Debugging new APIs is a well-known "adventure"
    - Whose bug is it, and who fixes it?
    - Assumptions live up to the old joke
    - Memory Leaks are a particular pain
  - Data Mapping
    - T&C "1" == P&S "3"? Enumerations are a pain.

MPS-T&C Messaging

- Part of the MPS-T&C interface is a set of XML messages
  - Satellite Support Plans
    - Contain scripts the T&C system uses to command the satellite, and metadata about that instance
  - Command Procedures
    - Templates used to generate the SSPs
  - Contact Support Plans
    - Contact data, including which SSPs to execute for that contact
- Some of these are files to ingest (CPs), some are files to transfer (SSPs), some are dynamically generated (CSPs) documents (streams)

Messaging Issues and Benefits

- In a message-driven system, latency and transport enter the equation
  - Latency – how long between sending a message and seeing action based on that message?
    - Not the same degree of real-time or near-real-time performance as with an API on the same system
  - Transport – how will it get there?
    - Highest-performance systems use shared memory
    - Lowest-performance do polling

Messaging Issues and Benefits

- Content is King
  - XML formats (schemas) allow adaptation
    - In a truly adaptive system, the schema would be negotiated at the beginning of a relationship between producer and consumer
  - Data-driven behaviors
    - A long argument can be made over whether it is better to have more interfaces, each with dedicated data, or fewer interfaces, which process differently depending on the data they receive.

External System Interfaces

- A key interface we haven't mentioned until now – interfaces with external Planning and Scheduling Systems
- Air Force Satellite Control Network is the CCS-C key interface for P&S
  - "509D" format Program Action Plan, or PAP, generated by MPS
  - Network Tasking Order reports generated by AFSCN ESD system received and processed by MPS

PAP and NTO interfaces

- Report-based interfaces
  - No wire-level electronic interface allowed – reports generated and transferred via floppy from our systems to ESD terminals, and vice-versa
- PAP – what we request in AFSCN resources
  - By IRON and time, with many special options
  - Binary report format, with ASCII and EBCDIC mixed
    - Coyote Ugly
- NTO – what we get back from the AFSCN
  - What we can have, when, for how long
  - Formatted text – we added a tracking ID, caused trouble

Scheduling System Interfaces

- We wish for a “standard” XML schedule schema
  - Who, when, where, and the details of what
    - Resources
    - Purpose
    - Priority
  - A base document schema with extensions for specific missions or systems

SAIC Integration with OS/Comet

- As an IR&D effort, SAIC worked with Harris to interface a derivative of our Mission Planning and Scheduling system with Harris's OS/Comet T&C system
  - Uses a programmatic Message Bus architecture
  - Encapsulated their T&C interface
  - XML worked!
- Integration has been successfully tested in Harris and SAIC labs

XML Interoperability

- SAIC wrote an agent that plugged into the Harris OS/Comet Message Bus and parsed the T&C interface XML messages, driving the OS/Comet API with the proper data

The same schemas, with the same data, support BOTH T&C products

Future Vision

- XML is an enabling technology
  - Negotiate schemas for strong typing across domains
- Interfaces mechanisms themselves need flexibility
  - APIs are required for near-real-time functions
    - But if they transmit significant data, why not use XML?
  - Standard messaging interfaces for non-real-time
    - XML report generators can use style sheets to display the reports in standard browsers, allow machine processing of the contents, too
  - Web Services, anyone?
    - Who wants to try to define the WSDL for T&C interfaces?
    - SOAP encapsulation is on our list for Future Research

Summary

- The basic interface mechanisms have their own strengths and weaknesses
  - Text reports need to move to XML
  - APIs required for real-time systems
  - Messaging generally needs to be fast and reliable
- XML is a powerful tool for success
  - XML reused between ISI EPOCH interface and Harris OS/Comet interface – great commonality, interoperability
  - XML enables extensible scheduler-to-scheduler interfaces
    - Subset of the P&S-T&C interfaces
  - Bow wave of technology, tools, practitioners