Integrated Flight-Ground Software for Rapid Mission Development

Mark McCrum, Alex Mason, Peter Mendham
About us

- **Space technology company** specialising in software
  - Based in Scotland, UK
  - Founded in 2011
  - Experience with a wide range of **upstream software**
    - Flight
    - Mission control/operations
    - Simulation
  - Mixture of software **products and services**
    - Supplying products to nano-satellite missions worldwide
    - Partnered with spacecraft integrators offering full software service
    - Ongoing R&D projects and consultancy with ESA, UKSA and others
Typical Space System

- Interface specified at the level of telecommand and telemetry packets
  - Impoverished view of the functionality provided
  - High level patterns of interaction not readily captured
  - Labour-intensive
  - Error-prone
  - Hard to change (particularly across organisational + contractual boundaries)
  - Complicates operation + automation

- Problems
  - Doesn’t scale well
  - Doesn’t support rapid, agile development
Some alternative approaches ...

CSP

PUS

The Consultative Committee for Space Data Systems

Mission Operations

European Cooperation for Space Standardization
GenerationOne

- **GenerationOne** is a core software technology and reference architecture
  - Applied to onboard software through a **Flight Software Development Kit**
  - Applied to operations through **Mission Control Software**
- GenerationOne addresses the key challenges by combining
  - **Model-based** software engineering
    - Model captures software at an architectural level
  - **Component-based** software engineering
    - Software is built from regularly structured modules (components)
  - A **service-oriented** architecture
    - Interactions between components conducted using services

- GenerationOne was not solely designed up-front
  - Started simple with features introduced gradually
  - Has been iterated and improved upon
  - Uses experience from many projects and missions
  - Features tested in a practical environment
  - Continues to be extended and improved
Characteristics of GenerationOne

- **Model-based** software engineering
  - Permits machine comprehension of software architecture
  - Enables tooling to assist with software development
  - Tools can also be used to assist with product/quality assurance
  - Model describes architecture across flight and ground
  - Model used across life-cycle from early development to end of operations

- **Component-based** software engineering
  - Key enabler for re-use
  - Each component includes implementation, tests and documentation
  - Complete system built from components
  - Lightweight underlying framework connects and supports components
  - Many (most) components portable across platforms and operating systems

- **Service-oriented** architecture
  - Provides consistent semantics for component interactions
  - Enables low level parts of software to be expressed as components
  - Raises the semantic level of operations
  - Separates interaction semantics from implementation protocol
Example software components

- **Subsystem components**, represent hardware
  - EPS, battery, ADCS, payload
  - Support for many off-the-shelf hardware subsystems and OBCs
    - AAC Microtec, Clyde Space, GOMspace, ISIS, Pumpkin, Vorago and many more
- **Data handling and monitoring components**
  - Sampling, data pool, aggregation, logging, monitoring, statistics
  - Support for most common onboard monitoring functions
- **Communications components**
  - Packet handling, telemetry reporting
  - Support for a number of different communications protocols
    - Includes support for ECSS PUS, CFDP
- **Automation components**
  - Absolute and relative time scheduling, orbit-based scheduling
  - Event-based automation
  - Onboard scripting
- **Mission-specific custom components**
  - Mode management, deployment sequencing, orbit counting
Typical workflow

1. Define Component Type
2. Implement Component Type
   - Generate Skeleton
3. Component Library
   - Generate Library Documentation
4. Define Deployment
   - Generate XML
5. Configure Component Instances
   - Generate C
   - Generate SCDB
   - Generate HTML
6. Deploy onto Target
   - Build Binary
### Component OBT

Instance of component type: subox.OBT

component OBT

Component ID: 4 (0x0004)

The OBT component provides an interface to the onboard system.

---

**MCS View**

The MCS View is a software interface designed for managing and monitoring spacecraft operations. It includes features for viewing and managing data, configuration, and events related to the spacecraft's mission. The interface includes tools for monitoring parameters, managing transfers, and viewing system state information. The MCS View is an essential tool for mission control teams to ensure the successful execution of missions.

---

<table>
<thead>
<tr>
<th>Time</th>
<th>Level</th>
<th>Source</th>
<th>Message</th>
</tr>
</thead>
</table>
Results

- 10 Diverse Cubesat missions within the company
  - Small team, rapid development
- 8 Satellites currently operating with gen1 technology
- Application Diversity
  - Commercial, Science, Education
- Platform & Technology Diversity
  - AAC-Clyde, Pumpkin, GOM Space, Custom
  - Linux, RTEMS, FreeRTOS, Bare metal
  - ARM, x86, Leon-2, SPARC, MSP430
- Centralised & Distributed Architectures
Conclusion

- Move towards a more unified treatment of space and ground through a shared functional architectural model
  - Facilitates **rapid iteration**
  - Raises the **semantic level** of the space / ground interface for improved **operability**, **autonomy** and **scalability**

**Bright Ascension Ltd**
www.brightascension.com
enquiries@brightascension.com
+44 (0) 1382 602041

Come and see us at
**Stand B6**
Interface to a component

- **Actions**
  - Commands/operations/methods
  - Can 'invoke' from onboard or ground

- **Parameters**
  - Data fields/attributes
  - Can 'get' or 'set' (if not read-only) from onboard or ground

- **Exceptions**
  - Status code
  - Returned by synchronous operations (e.g. get/set/invoke)
  - Indicates abnormal or unusual operation, usually an error

- **Events**
  - Issued asynchronously and usually logged onboard
  - Indicates abnormal or unusual operation, usually an error

- This is the interface that is “seen” from ground
- Components can be grouped together to form a logical hierarchy
The GenerationOne approach

- Provide common functions off-the-shelf as **software components**
  - Model captures the available component types
- Flight software rapidly **assembled** from these components
  - Within the “glue” of the GenerationOne framework
  - Model captures the architecture resulting from component assembly
- The software can therefore be **tailored for the mission**
  - Which components are used
  - How many of each type of component is used
  - The ways in which components are connected together
  - Custom components for the mission
- Component framework is relatively simple
  - Not flying a lot of unnecessary complexity
- Component interfaces follow regular structure
  - Limits issues caused by component interactions

- **Reduce** development **time, cost** and **risk**
  - Software **available earlier** to support AIT
Component interface from ground

Component

Action

Parameter
Components and services

- A component interface and a service are on different semantic levels
- Component interfaces (as defined by the OSRA)
  - Describe *what*
  - For example
    - An attribute
    - An event
  - Bindings bind a *thing* to a *thing*
    - e.g. an attribute to an attribute
- Service interfaces (as defined by MO and, to a lesser extent, SOIS)
  - Describe *how*
  - For example
    - Parameter service
    - Event service
  - Bindings bind a *mechanism* to a *mechanism*