Applying MBSE across Flight and Ground on Small and Nano-Satellite Missions

Peter Mendham

September 2020

Introduction

- Space software company based in Scotland
- Selling our innovative **space software products and services**
  - Customers across the world
  - Working across a wide range of applications
- First mission in 2014
  - Now have **15 spacecraft** flying our software
  - Majority of these also using our Mission Control Software
  - Spacecraft masses from 3kg to ~50kg
  - Many current customers are expanding to constellations
- Many more missions in development
  - At least ten more due to launch in the next year
- Customers on **six continents**
  - Mixture of product and services customers
- Collaborate in a number of industry-leading R&D activities
  - Commercial collaboration
  - Institutional and agency contracts (e.g. ESA)

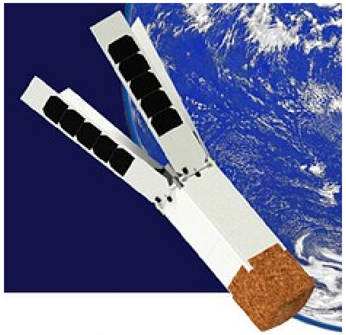
bright ascension

Commercial context

- **Rapid development** times
  - 6-12 months per satellite typical
  - 1-2 years from conception to initial commercial service
- **Large numbers** of satellites
  - Constellations of 10-50 satellites common
- High degree of **heterogeneity** between satellites in a constellation
  - Different payloads, generations, degradation
- **Complex payloads** which embed much of their functionality in software
  - For example software defined radios and use of reprogrammable logic
- Use of **COTS hardware** from a wide range of vendors
  - Thriving and competitive ecosystem for products
- Need for highly **automated operations**
  - Typically aiming for unattended operations for a week at a time
- Use of commercial **Ground Station Network** (GSN) providers

- In response to lower cost will **accept greater risk**

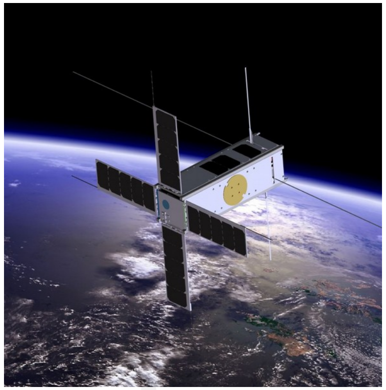
User needs



- **Availability**
  - Software must be available early (at least partially) to support test
- **Flexibility**
  - Requirements change as design is iterated
- **Rapidity**
  - Overall schedule is short and software must be ready quickly
- **Capability**
  - Many spacecraft functions are implemented in software
- **Operability**
  - Software must make it easy to achieve mission goals
- **Reliability**
  - Software is mission-critical and must be robust
- **Scalability**
  - Flight and ground software must integrate to form part of a complete system
  - May include multiple spacecraft and/or flight computers

bright ascension

GenerationOne technology



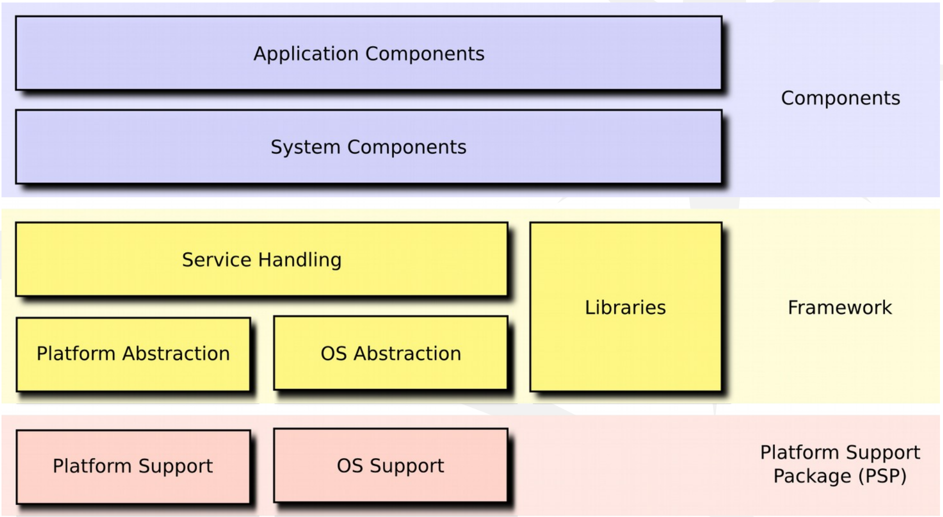
- **Model-based software** engineering
  - Machine comprehension of software architecture
  - Tools to assist with software development and product/quality assurance;
- **Component-based software** engineering
  - Reuse of software across a wide range of scenarios and applications
  - Combines software with its documentation and tests within libraries
- **Service-oriented** architecture
  - Consistent and well-defined semantics for component interactions at all levels
  - Enables low-level aspects of the system to be expressed as components
  - Improves operability

- **GenerationOne** is
  - A meta-model
  - A language-independent set of service and protocol definitions
  - Cross platform tools and framework implementations for target platforms
- Technology applied to **both flight and ground software**

bright ascension

Applying GenerationOne

- GenerationOne **runtime architecture** uses a lightweight framework
- Almost the entire software system can be expressed as **components**
  - Applications
  - Data handling
  - Communications protocols
  - Hardware drivers
- Backed up by a **range of tools**
  - Model handling and exchange
  - Code generation
  - Documentation generation
- Applied to **flight software**
  - C language
  - RTOS or Linux
- Applied to **ground software**
  - Java and Python
- **Single model** used to represent complete flight-ground system



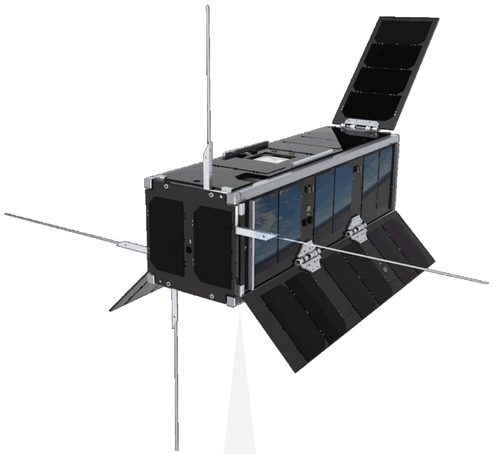
bright ascension

MBSE with GenerationOne

- Focus of GenerationOne model is **software architecture**
  - No representation of behaviour (e.g. no state machines etc.)
  - Best meets the needs of our target market
- Model captures **functional elements** of the system and **interactions**
  - Functional elements = components
  - Major functional interactions = services
  - Services concept permits both static and dynamic services binding/addressing
- Use of components permits management of **software reuse**
  - Product-oriented approach
- Model is general enough to capture both flight and ground concepts
  - Including **real-time concepts** for onboard software where applicable
  - Ground/flight usage differs primarily in how **dynamic** the system is
- Model is intended to be used across the life-cycle
  - From early prototyping through to operations and EoL
  - Permits rapid and easy **adaptation to change**
  - **Configuration effort** and maintenance significantly reduced

bright ascension

Evolution of GenerationOne

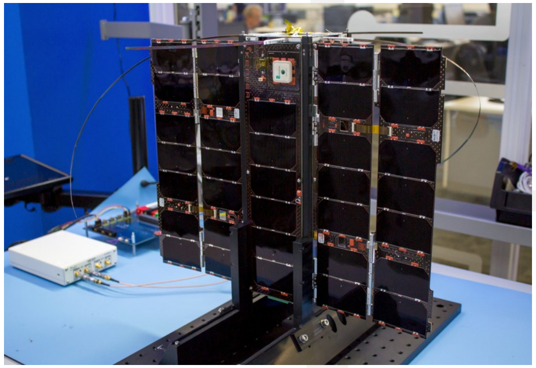


- GenerationOne as it is now was **not designed up-front**
  - Has evolved over time on the basis of mission experience
  - Continues to evolve now, and will in the future
  - We see this as a key strength of the technology
- Technologies such as GenerationOne rely on **careful use of abstraction**
  - Good abstractions make the overall system/process more efficient
  - Can only be designed from experience with a large sample of requirements
  - Always best tested in practice
- We introduce new features into GenerationOne and use them internally
  - Released into the product once they have been **mission-proven**
- Important part of process has been experience on ESA CubeSat missions
  - Such as **QARMAN** and **PICASSO**
- Missions which require significant mission-specific work result in change
  - “Shift the envelope” of missions that can be handled
  - Improve the applicability and capabilities of the product
- All missions are approached **in terms of the product**

Example missions: KIPP and CASE

- Pair of communications satellites
  - Customer is Kepler Communications
  - Satellite manufacturer was AAC Clyde Space
- Target delivery of Internet-of-Things services
  - Ku- and Ka-Band
  - Up to 100Mbps
- 3U CubeSats – mass of 3-4kg each
- Single payload
  - Reconfigurable FPGA-based
- Platform responsible for payload management
  - Including FPGA reconfiguration
- Major payload functionality is a black box to the platform

KEPLER



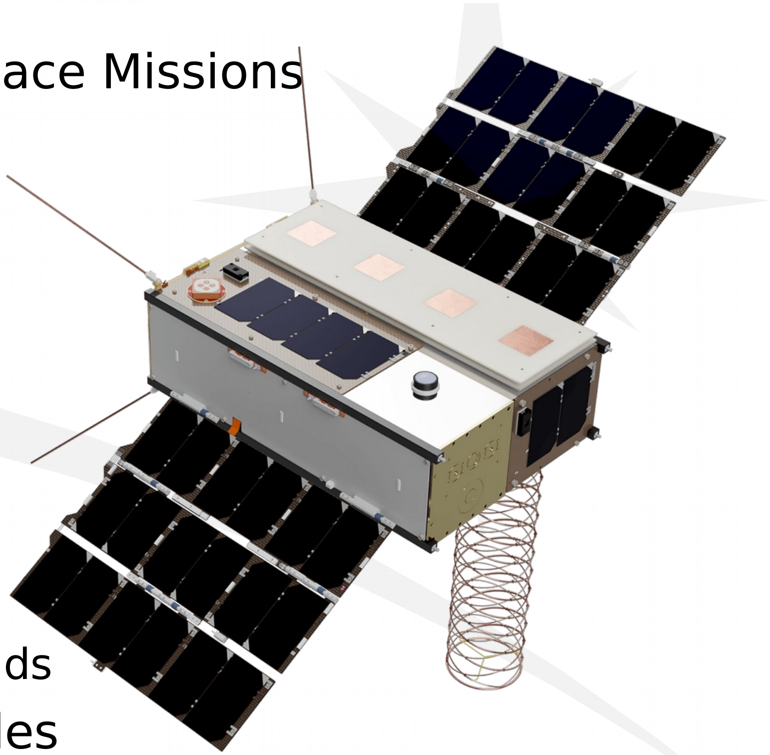
AAC CLYDE SPACE

bright ascension

Example missions: Faraday-1

- Hosted payload mission
  - Flies multiple payloads for different customers
  - Intended to be the first of a series
- Satellite integrator and service provider is In-Space Missions
- Six payloads
  - Including four software defined radios
- 6U CubeSat
  - Around 10kg mass
- Highly distributed onboard systems
  - Total of six compute platforms on board
  - Two platform OBCs running an RTOS
  - Four SDRs running Linux
  - SDRs capable of running multiple software payloads
- Demanding mission with many operational modes

INSPACE MISSIONS



bright ascension

Mission development workflow

- Development process begins with the **concept of operations**
  - Drives the requirements on how the mission should be operated
  - Operations-led focus drives flight and ground requirements simultaneously
- Allows selection of hardware and elements such as communications
- Development is typically iterative
  - Starts with initial software image based purely of **library components**
  - Usually deployed onto a development flatsat at this point
  - Can be used for payload development and testing
  - Incorporate **end-to-end testing** involving ground software from the start
- Per-mission development focussed on two areas
  - Selection and configuration of components from libraries
  - Development of mission-level orchestration components
- For most missions the majority of software comes from library components
  - These can be configured and are not modified for the mission
  - Maintaining the **separation and integrity of the product** is important

Use of GenerationOne for KIPP and CASE

- Overall development of the satellites was 8 months
  - Well-known platform
  - New payload – unspecified at project kick off
  - Payload development during this time also
- Software development was 6 months of effort over 5 months duration
- New software component developments for
  - Payload interfacing and management
  - Mission orchestration (e.g. mode management)
- Both spacecraft launched in 2018
  - Operating successfully since launch
  - Delivering a commercial service
- Customer responsible for operations
  - Model developed during spacecraft manufacture passed to customer
  - Permits rapid configuration of the Mission Control Software

Use of GenerationOne for Faraday-1

- Distributed system with multiple “software payloads”
  - Complete system included 13 different software images
- Involved the expansion and evolution of the product to accommodate
  - Better support for distributed systems
  - New platform and protocol support
  - Expanded model concepts and handling
- Complete flight-ground system captured as a single model
  - Including all flight computers
- Can accommodate different elements changing during flight
  - To accommodate software payload changeover
- Physical payloads use standard protocol which allows them to be modelled
  - Conceptually part of the overall system
- Development was long (2 years) but many lessons learned
  - Have been successfully applied to a number of more recent missions

- Launch failure in July 2020 – re-flight scheduled for 2021

bright ascension

Benefits of a flight/ground approach

- The spacelink is a **highly inefficient** place to put a major system division
  - Especially a commercial/programmatic division
- Results in
  - Poor architecture
  - Poor maintainability
  - Inefficient development process
  - Inefficient operations
- Technical environments and challenges for **flight/ground are different**
  - But system must work together to deliver mission
  - Success of GenerationOne shown modelling as a single system is possible
  - Significant gains in **efficiency, scalability and adaptability**
- Remaining drivers which encourage/enforce this division not applicable
  - Commercial/industrial environment different in “New Space”
- Frees organisations to deliver missions and services **more effectively**
  - Better value for money
  - Better time to market

Future directions for GenerationOne

- Greater introduction of **operations concepts** at spacecraft development
  - Increase the extent to which development process is operations-led
- Improvements in **operability**
  - Greater range of standard services
  - Improvements to semantics of existing services
- Increase **expressiveness** in model
  - Extend service model
  - Expand range of structural types to better express reuse and similarity
  - Permit parametric re-use of portions of the architecture
- Expanded **range of tools** for model handling
  - Will be introducing some GUI-based tools
- Provide “pure” ports of GenerationOne to **Java and Python**
  - Current implementations are partial
  - Need further work to align with current GenerationOne state-of-the-art
- Expand the range of available **platforms and components**

bright ascension

Speak to us

Question, comments or suggestions

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

generationone