

# Integrating Advanced Payload Data Processing in a Demanding CubeSat Mission

Mark McCrum, Peter Mendham



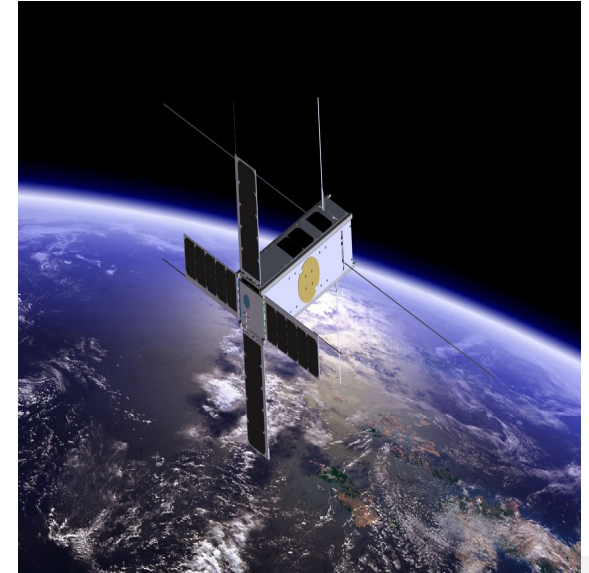
# CubeSat mission capability

- Nano-satellites missions are **increasing in capability**
  - Constellations
  - Distributed ground segment
  - Ground segment
  - Onboard autonomy
  - Advanced payloads
- What makes an **advanced payload**?
  - Challenging concept of operations
  - High-data rate
  - Requirement for substantial onboard processing
- How can advanced payloads be incorporated into a mission?
  - Whilst controlling risk
  - Minimising AIT complexities
  - Without requiring complex operations on the ground



# The PICASSO mission

- Atmospheric science mission
- Led by by the Belgian Institute of Space Aeronomy
  - Managed by European Space Agency
  - Clyde Space leading spacecraft manufacture
  - Bright Ascension leading software work
  - Imager by VTT
- Main target is **science of the upper atmosphere**
  - Stratospheric Ozone distribution
  - Mesospheric Temperature profile
  - Electron density in the ionosphere
- Two instruments
  - Miniaturised **hyperspectral imager** for solar disc imaging in the limb
  - Multi-needle **Langmuir probe**



# PICASSO mission challenges

- Hyperspectral imager produces a lot of data
  - **640 Mbps** during measurement period
  - Data must be windowed (in real time)
  - Then compressed (not real time, between measurements)
  - Then archived onboard before next available downlink
  - S-band downlink permits 1Mbps
- **Timing** of measurement periods is critical
  - Measurements must be taken in the limb
  - All measurements must be precisely timestamped
  - Using GPS as a time source
- **Attitude** during measurements is critical
  - Largely handled by ADCS
  - Coordination of ADCS with platform and payload operations is critical

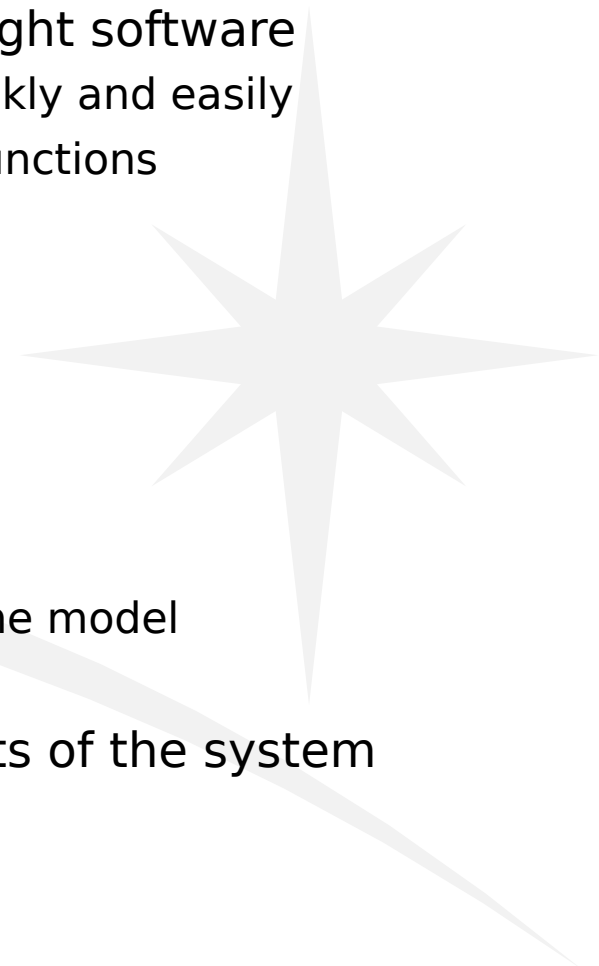


# PICASSO functional architecture

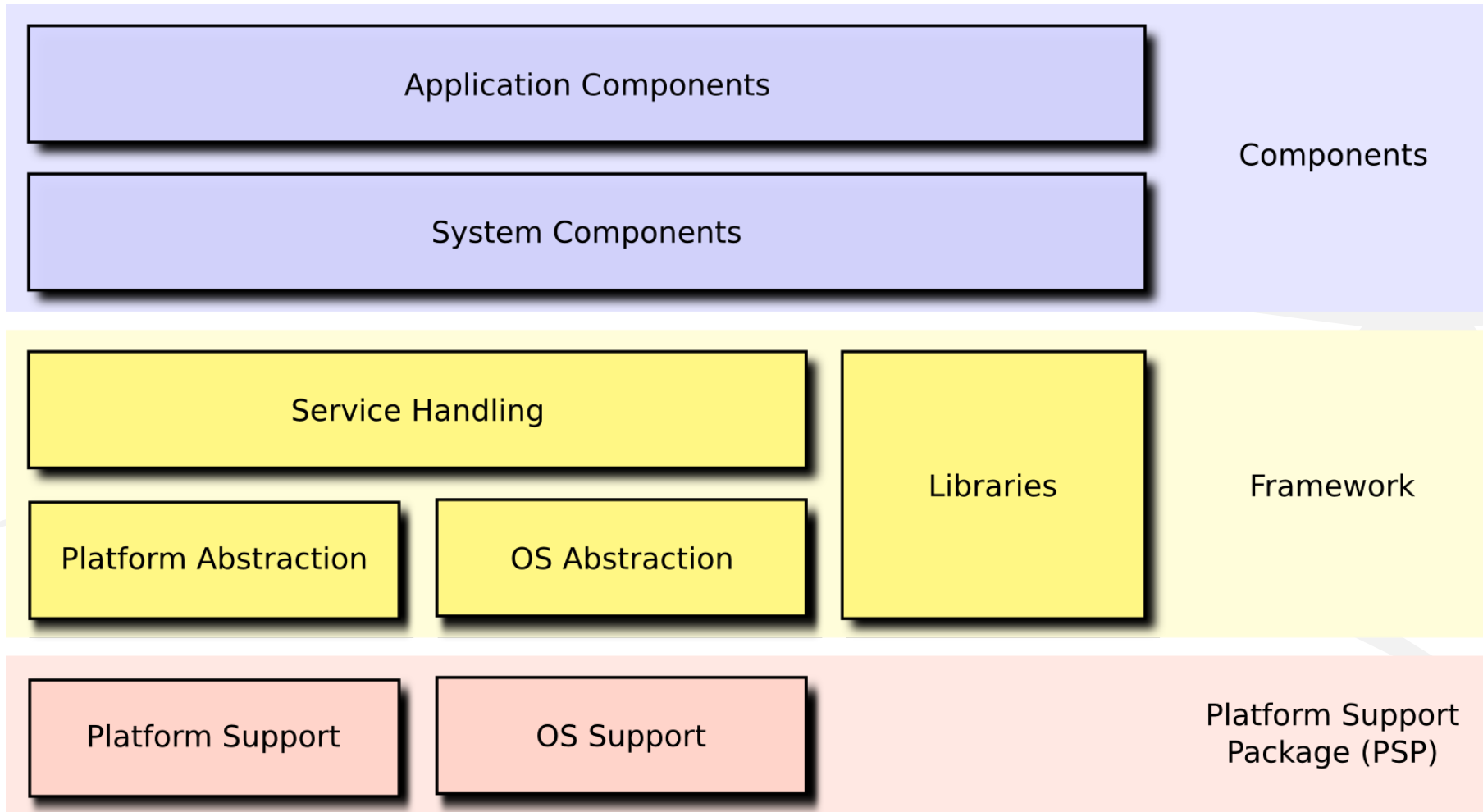
- High performance computer necessary for payload data handling
  - Large **memory** requirements
  - Large **mass storage** requirements
  - High-speed **I/O** interface required
  - High-performance **processing** required for real-time windowing
- Trade space for payload processing is very different to platform
  - Platform computing requirements
    - Dependable
    - Real time
    - Low power
    - Requirements for memory/performance are low
- Selected separate platform and payload computers
  - Platform: GOMspace Nanomind
  - Payload: Xiphos Q7
- Results in a **distributed architecture** on board

# GenerationOne flight software

- GenerationOne is a **software development kit** for flight software
  - A framework and tooling to allow software to be built quickly and easily
  - A library of validated components for common onboard functions
- GenerationOne is **component-based**
  - Allows clean and easy reuse of heritage code
  - Easy integration of new functionality
  - More streamlined testing and integration
- GenerationOne is **model-based**
  - Allows tooling to “understand” your software
  - Ground software and onboard software can share the same model
  - Enables lots of automation and code generation
- Software architecture **cleanly abstracts** different parts of the system
  - Hardware independence
  - Operating system independence
  - Protocol independence



# GenerationOne architecture



# Example software components

- **Subsystem components**, represent hardware
  - EPS, battery, ADCS, payload
  - Support for many off-the-shelf hardware subsystems
  - Clyde Space, GOMspace, ISIS and 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 and more to come
- **Automation components**
  - Absolute and relative time scheduling, orbit-based scheduling
  - Event-based automation
  - Onboard scripting
- **Mission-specific custom components**
  - Mode management, deployment sequencing





# Component interface from ground

The screenshot displays the TMTc Lab - GenerationOne OBSW interface. The main window is titled "Spacecraft Explorer" and shows a hierarchical tree of components. The "lifeTime" parameter is selected, and its details are shown in the right-hand pane. The details include the parameter ID (3584 (0x0E00)), a description of its function, and a "Parameter Data" section with a "Data" field containing "0000000C". The "Debug/Event Console" at the bottom shows a log of events, including a successful deployment initialization.

**Component** (points to the tree structure)

**Action** (points to the "refresh" icon)

**Parameter** (points to the "lifeTime" parameter)

**Model captures all information** (points to the entire interface)

**Parameter lifeTime**

param uint32 lifeTime

Parameter ID: 3584 (0x0E00)

How long a pooled parameter can be considered valid in seconds.

The if the time since the DataPool component was last refreshed exceeds the lifeTime value, then the poolParameters will be considered invalid and accessing the parameters without refreshing will cause a read-through. An OBT component is required to calculate whether a pooled parameter has become invalid.

Current size: Fixed size

Parameter Data

First row:  Last row:   Resize

Data:

Data should be specified in hexadecimal with no prefixes. Spaces should be used to separate rows.

Parameter accessed successfully

Time	Level	Component	Filename	Line	Message
Wed 14:01:59	DEBUG	framework	LoggerCore.c	111	Channel has incorrect record size, reformatting
Wed 14:01:59	DEBUG	framework	FileStoragePr...	298	Overwriting
Wed 14:01:59	DEBUG	framework	FileStoragePr...	298	Overwriting
Wed 14:01:59	INFO	framework	Deployment.c	36	Deployment initialisation successful

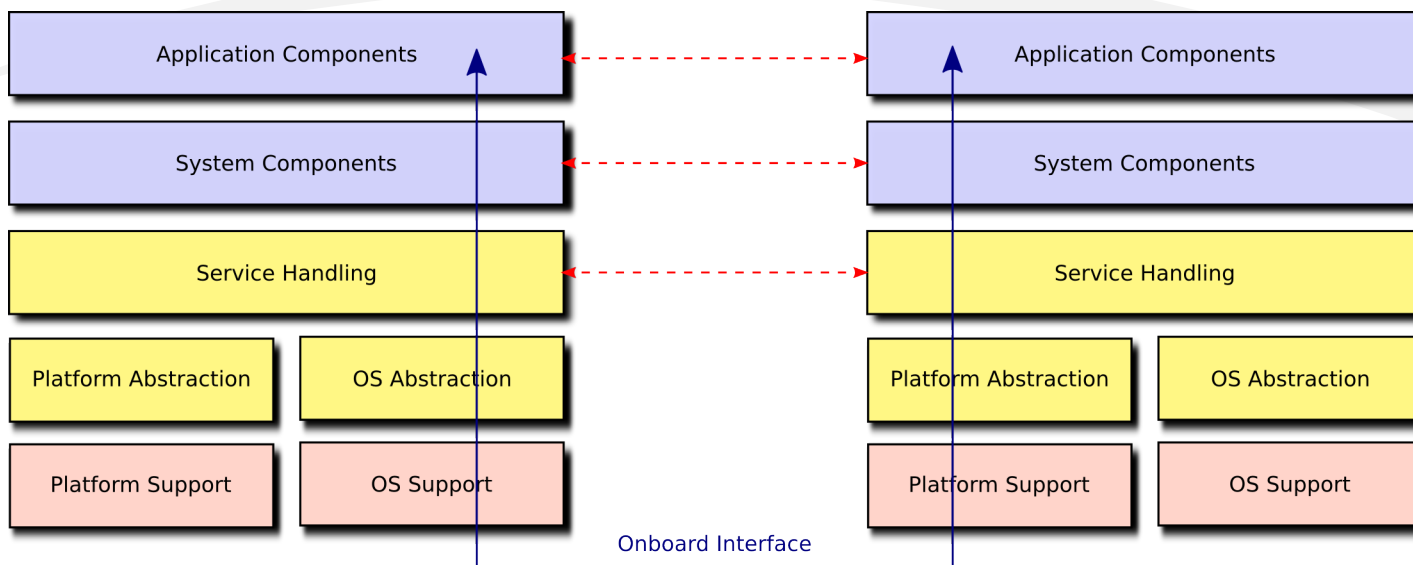
Connected

# GenerationOne and distributed systems

- GenerationOne will be used on multiple computers or subsystems
- Captured as part of the **same model**
- Component framework distributed across all computers
- Communications between the computers allow components to interact
  - Independent of location
  - Independent of communications protocol
- Ground software “sees” a single spacecraft
  - Uniform operations across multiple onboard computers/subsystems
  - Component physical location not hidden but not usually an issue for operations
- Easier and more flexible **development**
  - Can move components around to suit the mission
  - Adapt to changing requirements
  - Introduce new computers/subsystems without a large architectural impact
- Simplify **AIT**
  - Uniform way of testing and integrating
- Simplify **operations**

# Distributed software architecture

- Components interact using the standard **component interface**
  - Actions
  - Parameters
  - Events
- Component interface is dispatched using **framework services**
  - Framework services are themselves provided by system components
  - Communications stack built from components
  - Makes component interface services modular and technology-independent



# Technology independence

- Component-based technology makes **complete stack modular**
  - Hardware platform
  - Operating system
  - Drivers and subsystem interfaces
  - Communications protocols
- Permits the use of **standard network protocols** for routing
  - CCSDS Space Packets
  - Internet Protocol
  - CubeSat Space Protocol
- Permits development and test to be **independent of technologies**
  - e.g. communications technology, architecture, topology, platform, OS
  - Adapt to requirements change
  - Rapid development
  - Carry out software testing and AIT at a high level - more efficient

# GenerationOne for PICASSO

- Distributed extensions for GenerationOne are being used on PICASSO
- **Two onboard computers** each using GenerationOne
  - Platform computer hosts majority of operational software
  - Main mission management on platform computer
  - Payload operations, data processing and downlink on payload computer
  - Payload computer not always powered
- Model-based distributed approach is **streamlining development** and AIT
  - First stage integration can be done with simulated computers (using PCs)
  - Second stage integration seamless due to abstraction
  - Tests are independent of physical architecture and protocols
- PICASSO **operations simplified** through use of the model
  - More efficient operations
  - Easier to achieve automation and “lights out” operations

# Lessons for other missions

- Within the context of highly-integrated nano-satellites distributed systems can be a **powerful approach**
  - Good solution to handling high-performance, advanced payloads
  - Existing CubeSat missions already use distributed architectures for this reason
- Distributed systems **introduce complexity**
  - Introducing a network protocol has limited impact on managing complexity
  - At a high-level test and operations must account for multiple systems
- Using a **model-based solution** does help manage complexity
  - Even a simple model can help
  - Can start with a technology-dependent model
- **Technology-independence** and **modularity** further help improve development and test process
  - Help manage change and make better use of development/test time
- A model can also be used to help **manage assurance**
  - Tracing of requirements, design, test etc.
- GenerationOne SDK is available for use on your mission

# Speak to us

Question, comments or suggestions

## Bright Ascension Ltd

[www.brightascension.com](http://www.brightascension.com)

[enquiries@brightascension.com](mailto:enquiries@brightascension.com)

+44 (0) 1382 602041



generationone  
flight software development kit



bright ascension