

# MO Services, SOIS and SAVOIR Harmonisation: Project Status

Peter Mendham, Bright Ascension

Simon Reid, RHEA System

Andreas Wortmann, OHB System

Andreas Jung, Mehran Sarkarati, ESA



# Agenda

- Introduction to the MOSS activity and its goals
- The MOSS approach to consolidation and consolidation aims
- Technology review
- Consolidating the user needs and requirements
- The consolidated architecture
- Looking ahead to the harmonised architecture
- Technology issues arising during consolidation and recommendations
- The proof-of-concept prototype
- Next steps
- Summary and conclusions



# The MOSS activity



# Activity objectives

- Analyse three technologies with a view to using them to create a **single harmonised architecture**
  - CCSDS Mission Operations Services (MO)
  - SAVOIR Onboard Software Reference Architecture (OSRA)
  - CCSDS Spacecraft Onboard Interface Services (SOIS)
- The harmonisation must be driven by user needs and requirements
  - The original user needs and high-level requirements for the technologies
  - If these are not currently documented they must be elicited
- A suitable harmonised architecture should be proposed
  - This must take into account to expected migration path
  - Intermediate architectures may be necessary
  - The relationship with PUS(-C) should be captured
- Key aspects of the resulting architecture should be prototyped

# Approach

- Balance of bottom-up and top-down activity
- Technology-driven – *bottom-up*
  - Technologies of interest are clearly identified in the scope of work
  - Technologies are selected due to their potentially strategic importance
  - Stage of development of the technologies likely permits influence over direction
- Requirements-driven – *top-down*
  - Starting point for consolidation is user needs and requirements
  - Consistent approach taken to requirements gathering across technologies
  - Requirements are consolidated before technologies
- Breadth rather than depth
  - Many consolidation issues are to be found at architectural level
  - Design and prototyping attempts to cover the full breadth of the architecture
  - Prototype will not go into full detail in all aspects

# Feedback and recommendations

- Breadth of MOSS activity allows checking for alignment
  - Flight and ground differences in conceptual approach
  - Alignment between MO and the OSRA
  - Further examination of alignment between MO and SOIS
- Feedback to relevant working groups
  - SAVOIR and CCSDS (MOIMS and SOIS)
- Recommendations of changes to permit better alignment
  - Essential and advisable changes
  - Short-term and long-term changes
- Recommended adoption approach
  - Especially for the adoption of MO onboard



# Technology review



# MO: vision

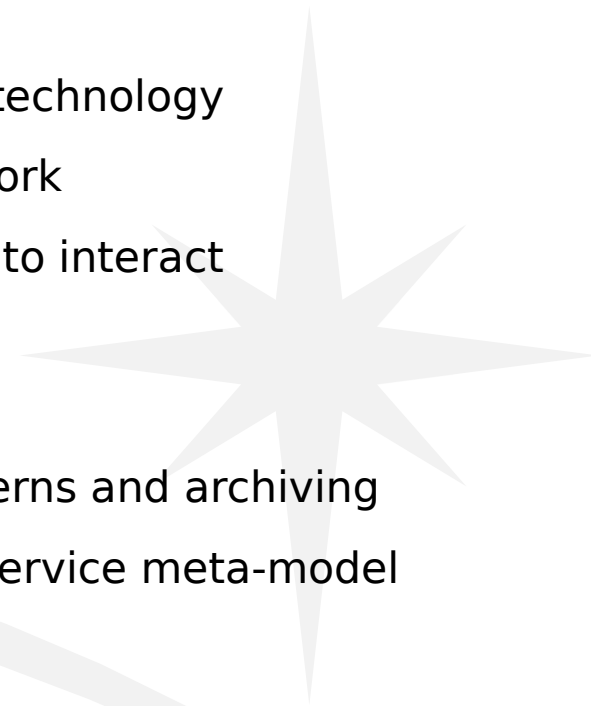
- Service infrastructure for large-scale distributed system
  - Spans local area, wide area and space-ground networks
- Layered approach
  - Well-defined, rich, semantic services
  - Service interactions
  - Communications protocols
- Key drivers and motivation:
  - Greater cooperation, e.g. inter-agency, hosted payloads etc.
  - More efficient operations with greater automation
  - End-to-end traceability and audit trail for operations
  - Technology-independence and long-term maintainability
  - Modular systems and plug-in components
  - Taking a space system-centric approach through development and operations

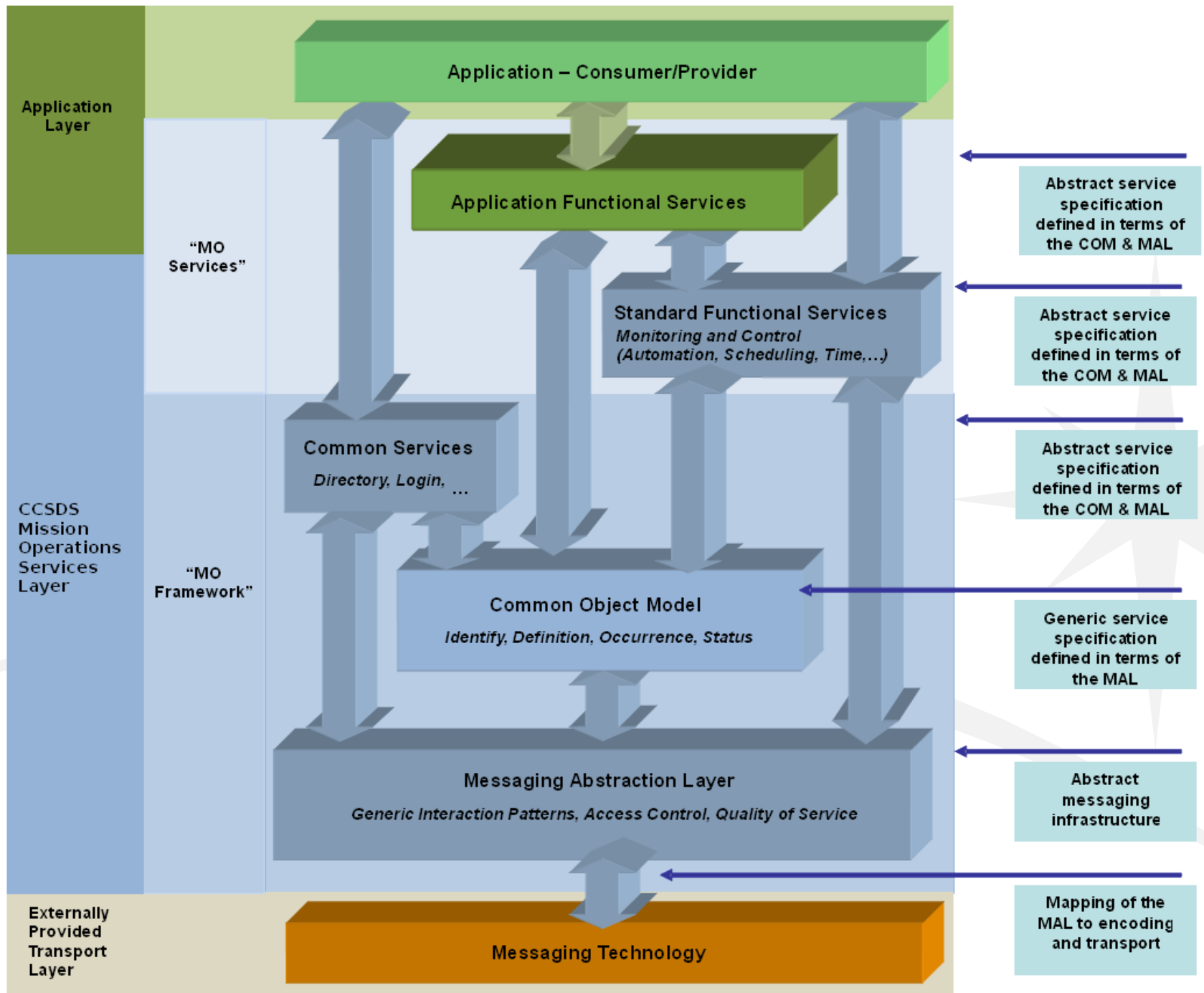




# MO: context and concepts

- Key terminology
  - Mission Operations (MO) is used to refer to the complete technology
  - MO Framework is the underlying service-oriented framework
  - MO Services are the services which utilise the framework to interact
- MO Framework
  - Common services (e.g. directory, login etc.)
  - Common Object Model (COM) – meta-model, service patterns and archiving
  - Message Abstraction Layer (MAL) – abstract messaging, service meta-model
  - Concrete communications protocol binding
- MO Services
  - Monitor and Control (M&C) Services, in draft
  - Automation, Scheduling, Time, Remote Buffer Management, File Management and Broker Services all to be defined
- Extended by application services





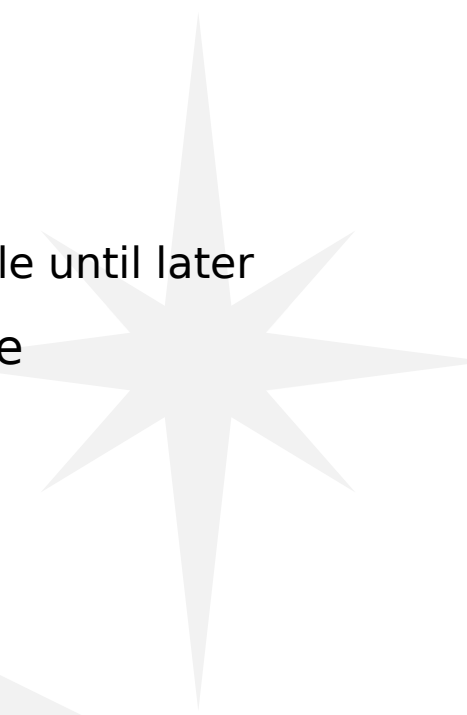
# MO: use cases

- Short term focuses on practical application of MO framework
  - Makes use of communications framework and service interactions
- Long term focuses on semantics
  - Makes use of semantic services for more efficient operations and development
- Short-term use cases
  - Generic M&C for payload operations
  - Inter-agency hosted payloads
  - Automated payload functions
- Long-term use cases
  - Semantic payload operations
  - Automation services
  - System-level design with MO

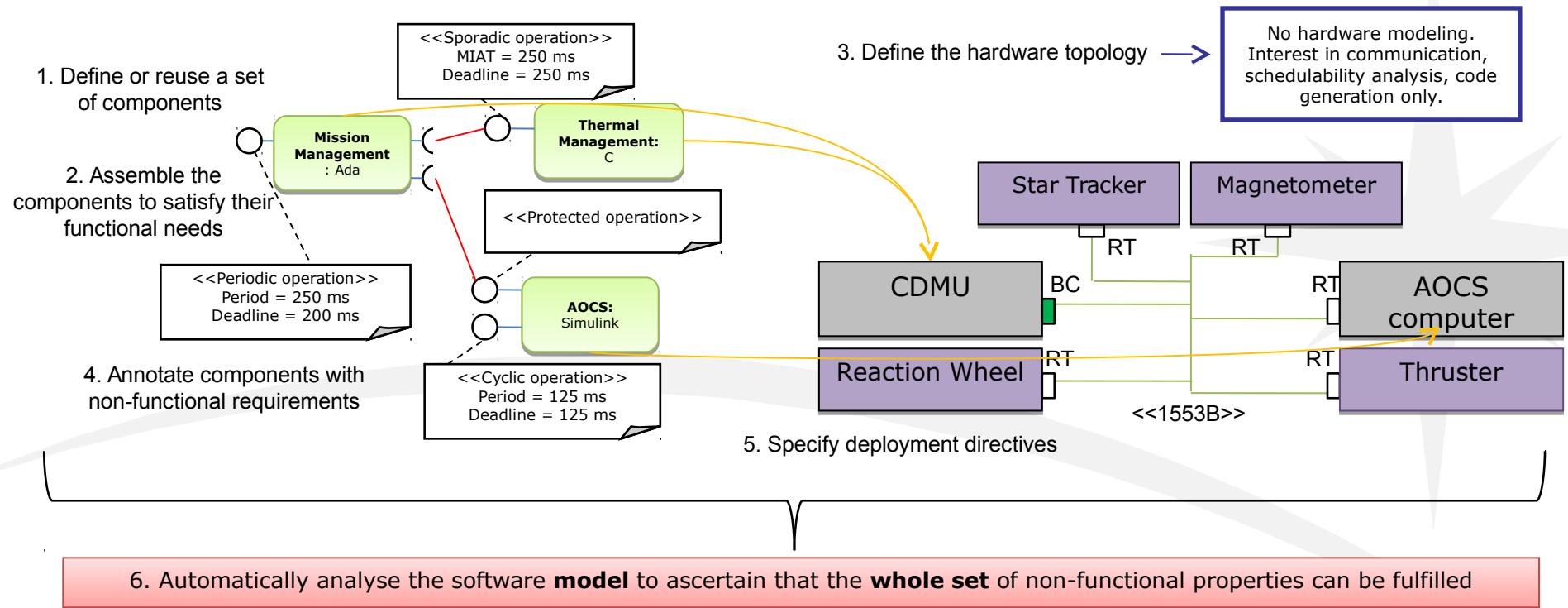


# OSRA: vision

- Address the pressures on onboard software
  - Greater functionality
  - Greater value for money
  - Schedule pressure to have software available sooner but flexible until later
- Work within the environment for onboard software in Europe
  - Assurance requirements
  - Commercial and geopolitical constraints
- Utilises
  - Model-based software engineering
  - Component-based technology
- Encourages software reuse and the emergence of product lines in software



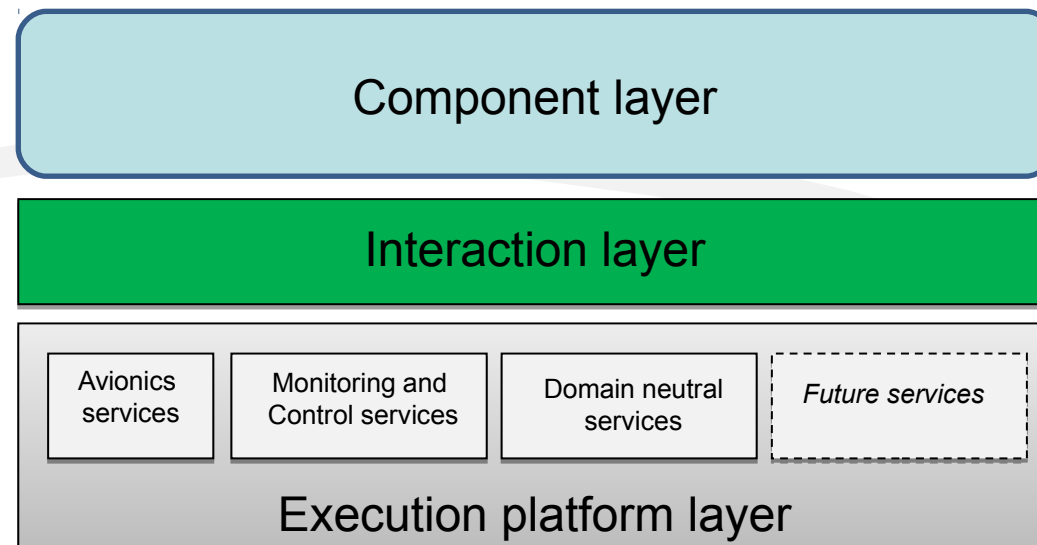
# OSRA: development process



Taken from the OSRA Training Material

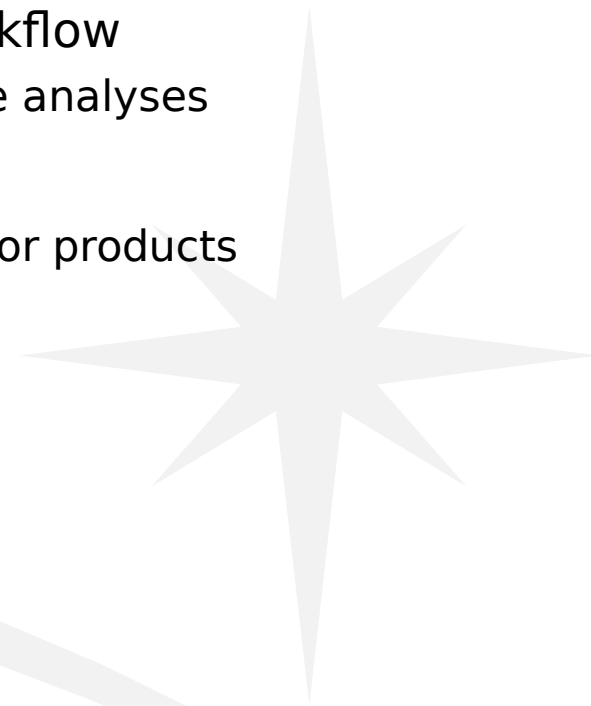
# OSRA: run-time architecture

- OSRA components sit inside their containers and utilised connectors
  - Both of these rely on functions offered by the underlying Execution Platform
- Containers and connectors are **tool-generated at deployment time**
  - They form the **Interaction Layer**



# OSRA: use cases

- Short term focuses on improving the development workflow
  - Software reuse within organisations, automation of simple analyses
- Long term focuses on extending the role of the OSRA
  - Software reuse across organisations and the introduction of products
  - More complex analyses and support to assurance
- Short-term use case
  - Developing reliable software in an uncertain environment
- Long-term use case
  - Rapid development of assured software



# SOIS: vision

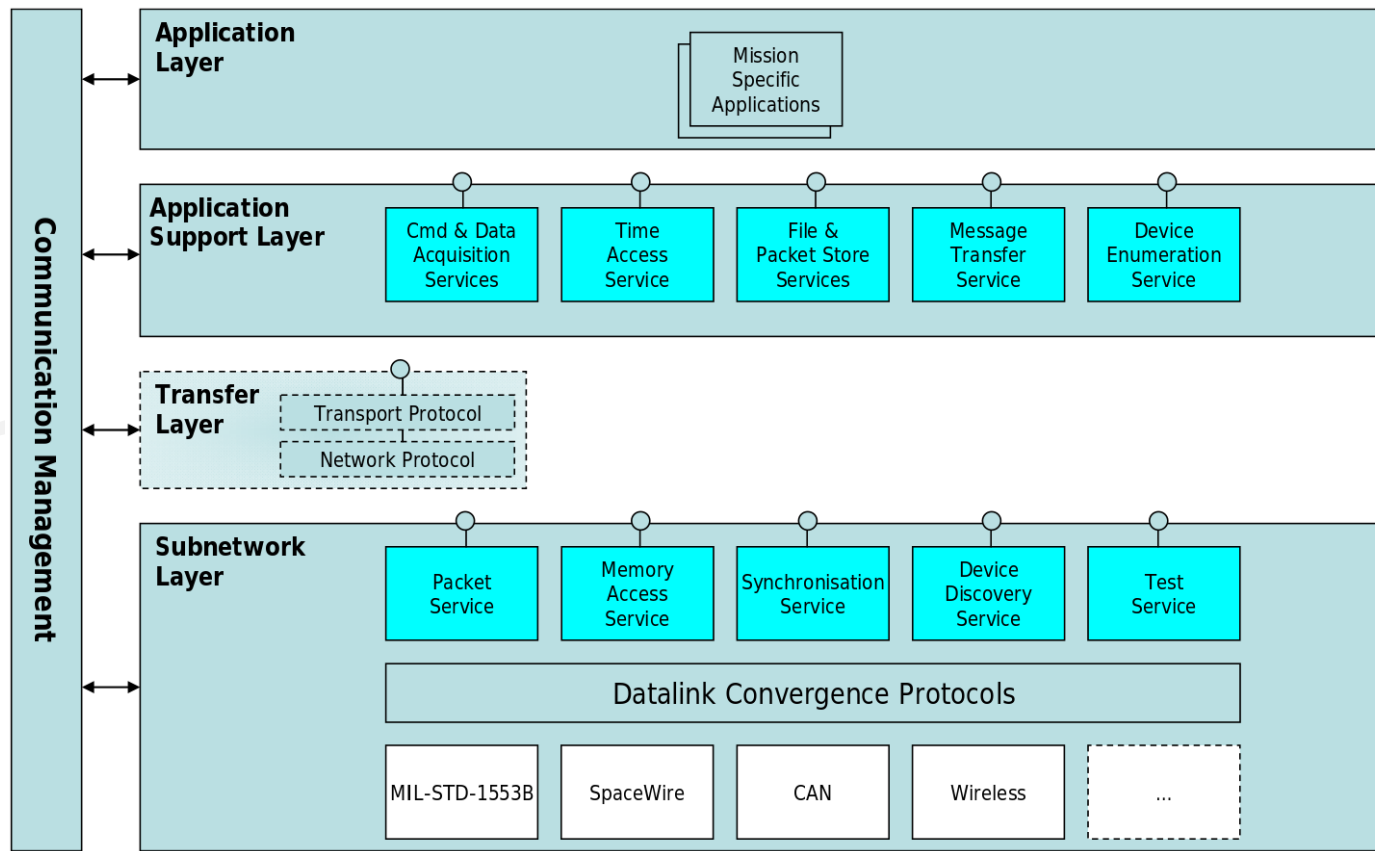
- Improve the design and development process of onboard data systems
  - Defining generic services for interactions between flight software and hardware
  - Increase potential for interoperability and reuse
- Potential benefits include
  - Reduced development cost and risk
  - Shorter development times
  - Easier integration
  - Encouraging the emergence of off-the-shelf equipment
- Utilises a reference communications architecture
  - Spanning hardware and software
- Includes an approach to “plug-and-play”
  - Spans design-/development-time approaches as well as run-time approaches





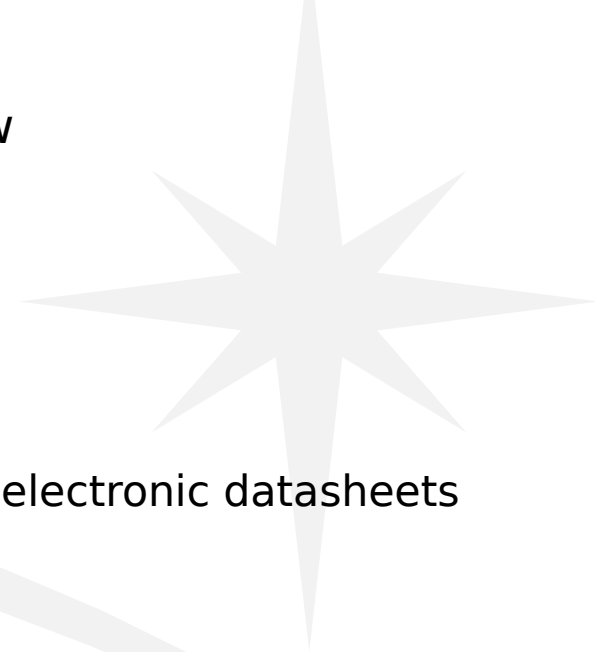
# SOIS reference architecture

- Specified in terms of services, each defined by their interface
- Service implementations are not specified



# SOIS: use cases

- Short term focuses on encouraging reuse through standard interfaces
  - Such as avionics hardware, software or EGSE reuse
- Long term focuses on improving development workflow
  - Introduction of plug-and-play technologies
- Short-term use case
  - Reuse of avionics hardware across missions
- Long-term use case
  - Rapid integration of avionics using standard services and electronic datasheets



# Technology aims: common aims

- Common aims:
  - Reduced development costs
  - Increased development adaptability
  - Decreased risk
- Common Architectural characteristics:
  - Modularity to enable reuse
  - Encapsulation and abstraction to control complexity
  - Semantically-structured interfaces
  - Standardised interfaces
- The different foci of the technologies also creates some differences



# Technology aims: differences (1)

- Operations is the main focus for MO
  - Operations is run-time behaviour
  - MO also cares about the way software is constructed
    - Need to allow cross-agency operations
- Development is the main focus for SOIS and the OSRA
  - This is design-time behaviour
  - For example: 13 User Needs presented for the OSRA
    - 12 design-time User Needs
    - 1 run-time User Need
  - Abstraction and layering are seen as useful at design time and harmful at run time
  - Certainly in the OSRA, less so in SOIS, tooling is used to *remove* design-time layering and abstraction at run-time
- This is a crucial difference between the technologies



# Technology aims: differences (2)

- A key point worth emphasising is difference of scope
- Scope of SOIS
  - Single process space
  - Onboard a spacecraft
- Scope of the OSRA
  - Multiple process spaces (if necessary)
  - Onboard a spacecraft
- Scope of MO
  - Multiple process spaces
  - Across the complete space-ground system
    - Or system-of-systems
- What defines a system as “onboard”?
  - Embedded, real-time, resource-constrained, high-dependability
  - Subject to monitoring and control



# Consolidated user needs/requirements

- Bring together superset of user needs and high-level requirements
- Combine user needs/requirements where they align
- Resulting vision addresses development-time and operational concerns
  
- Introduce semantically sound, generic services
- Support improvements to operability, observability and automation
- Promote improvements to development without sacrificing assurance
- Encourage the complete space system to be treated as a whole for
  - Development
  - Operations

# Consolidated architecture

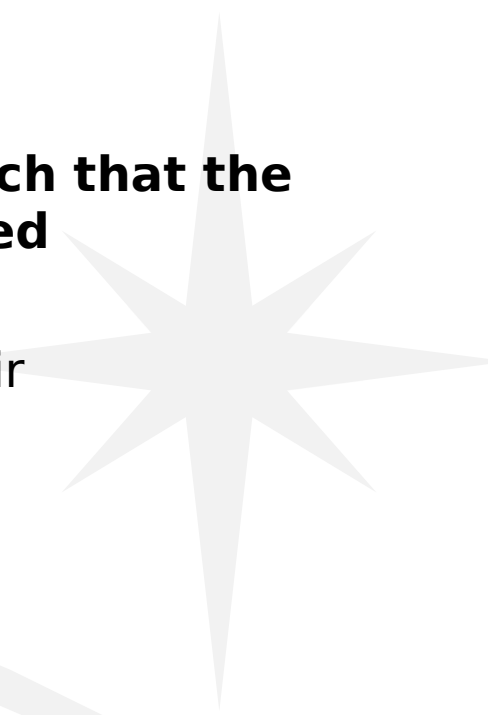


# Consolidation aim

- The aim of consolidation is to

**allow the combination of MO, SOIS and the OSRA such that the benefits of each individual technology are maintained**

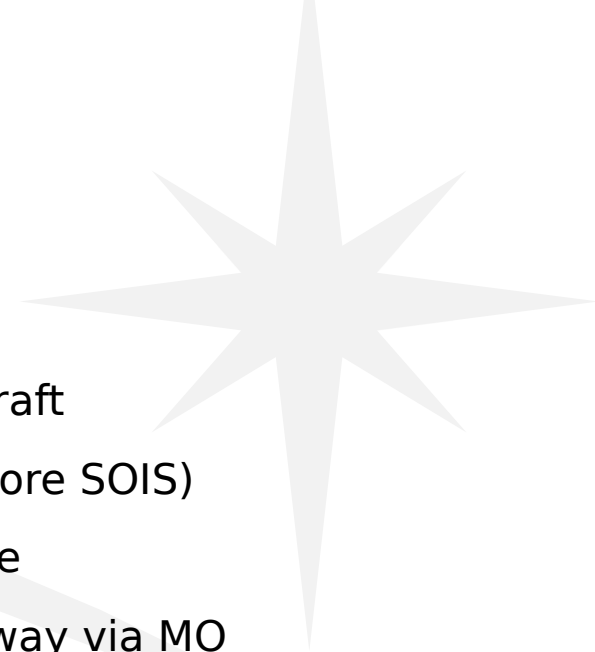
- The benefits of each technology should be captured in their
  - User Needs
  - High-level Requirements
- Need to take into account
  - Development-time and run-time needs
  - Differences in scope
  - Should not force the use of a particular technology
- Balances top-down and bottom-up approaches
  - Technology-focussed but aims to meet consolidate high-level requirements



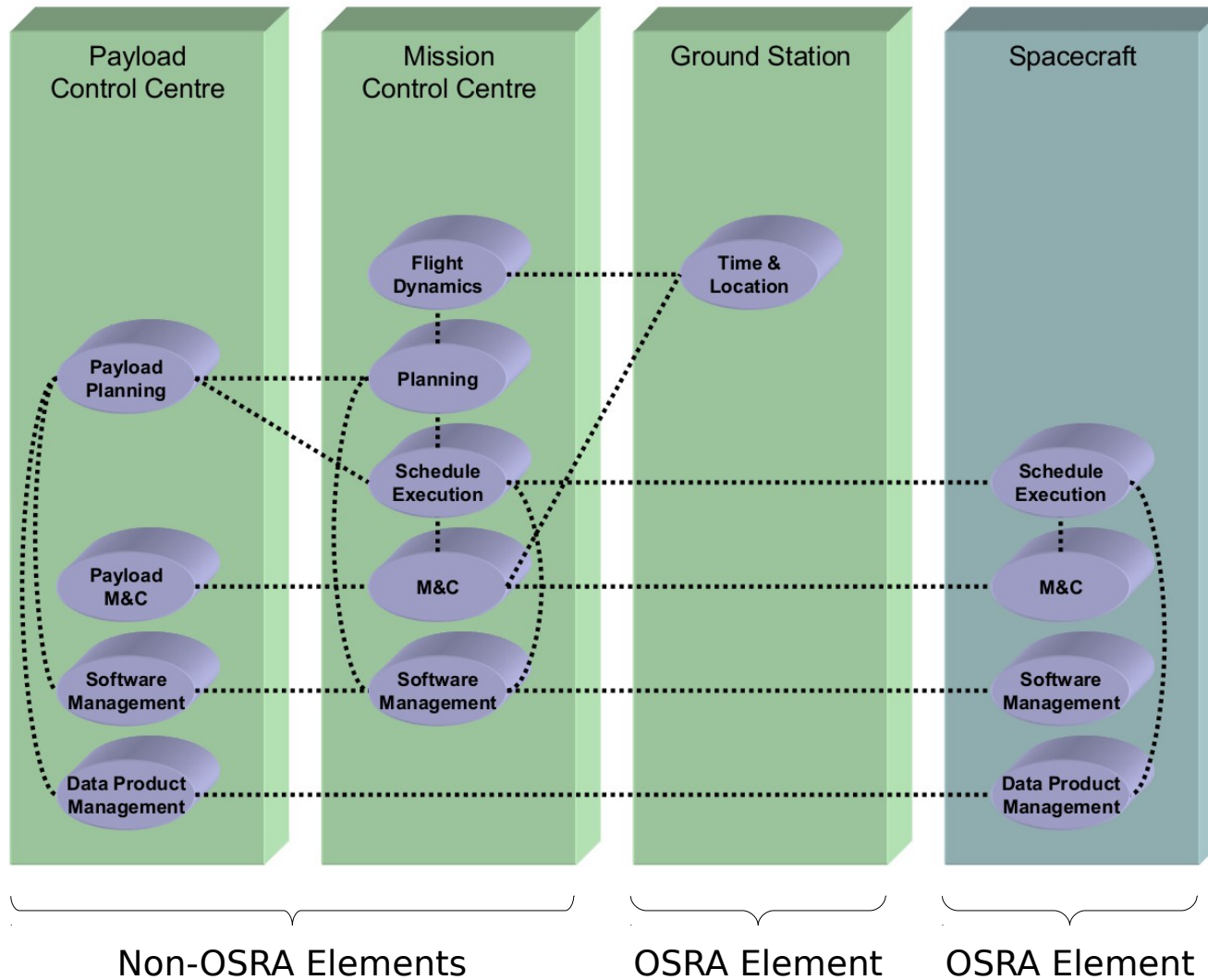


# Consolidation approach

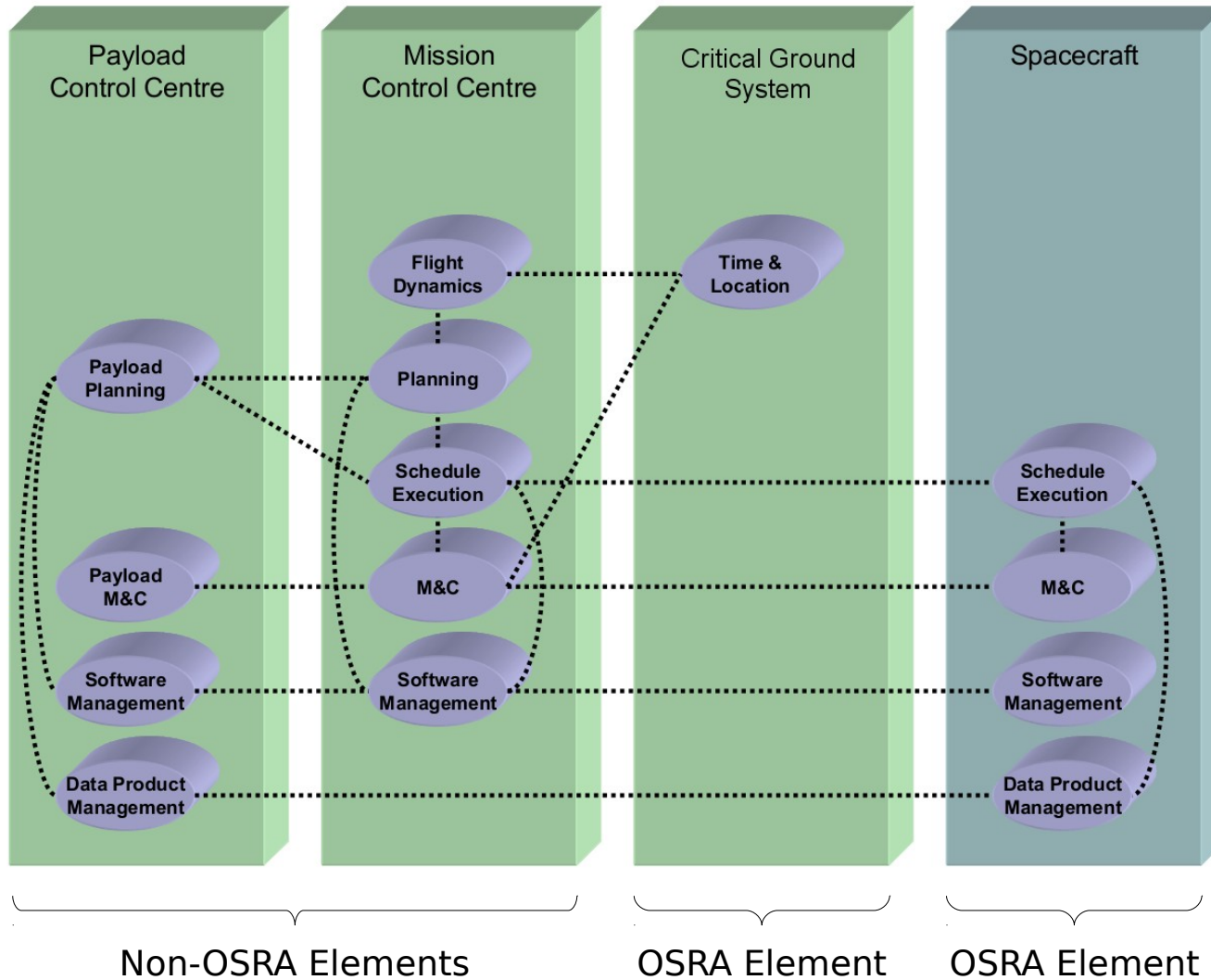
- Allow the application of the OSRA and SOIS within an MO architecture
- Identify specific regions or *elements* of an MO system
  - These are developed using the OSRA
  - Also allow use of SOIS
- No need to apply the OSRA/SOIS to other elements
- For example
  - MO system spanning MCC, PCC, Ground station(s), Spacecraft
  - Spacecraft could be developed using the OSRA (and therefore SOIS)
  - Remaining systems largely unaffected at development time
  - At run time, functions of OSRA/SOIS exposed in a uniform way via MO
  - The OSRA approach could be applied to other embedded, real-time, resource-constrained, high-dependability systems subject to monitoring and control
    - e.g. Payload performance monitoring



# OSRA elements with an MO architecture

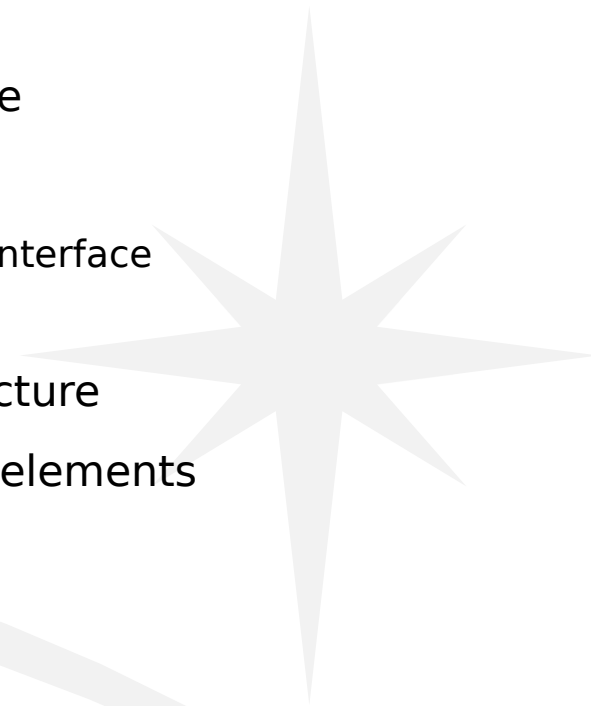


# OSRA elements with an MO architecture



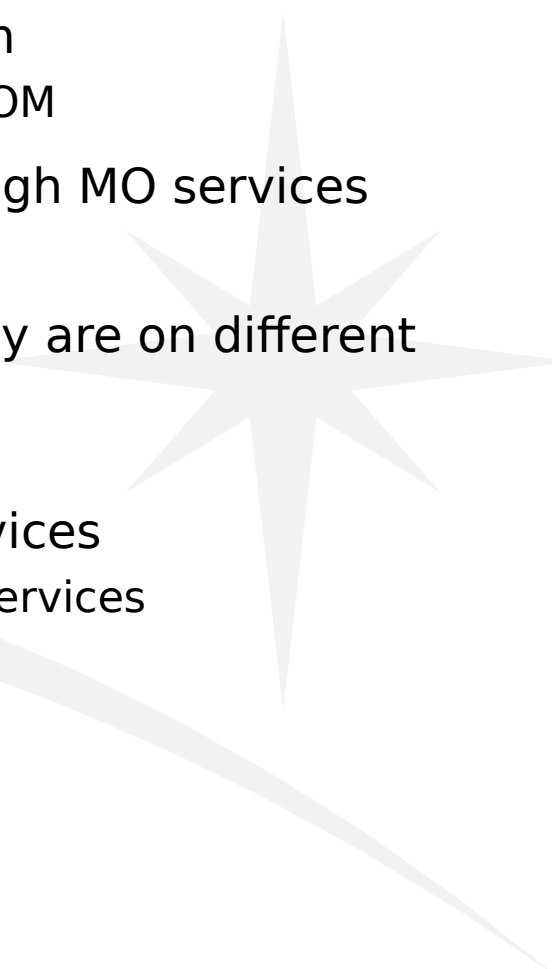
# The consolidation vision

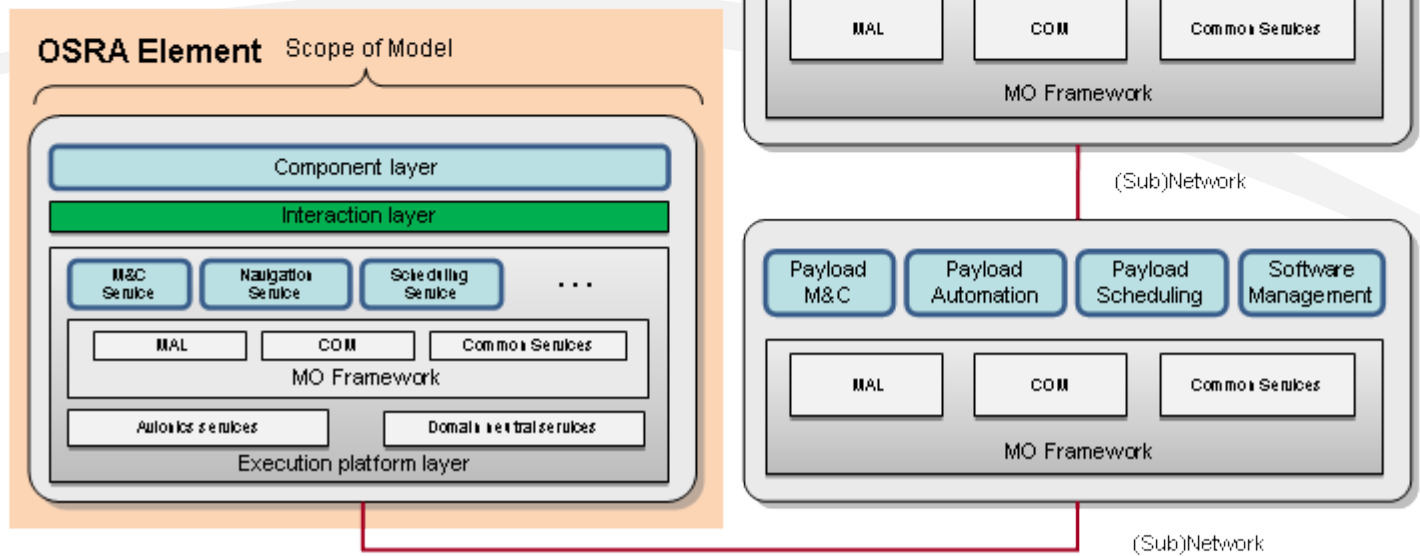
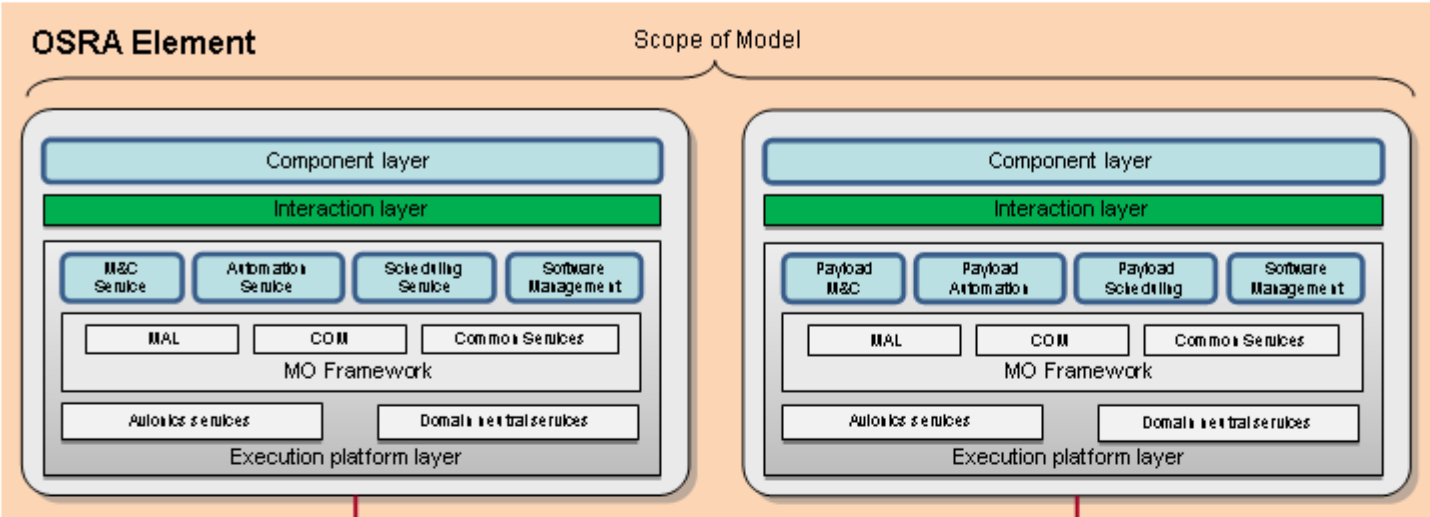
- Design time
  - Adopt a global, model-based, service-oriented architecture
  - Allow the exchange of design information
    - Corresponds to a single, shared model of services and their interface
    - Extend the MO service to support the necessary information
  - Permit component-based development within this architecture
  - Allow support for quality assurance of high-dependability elements
  - Adopt standard service interfaces to promote portability
- Run time
  - Maintain the service-oriented architecture at run-time
  - Ensure all system elements utilise consistent external interfaces
  - Adopt communications standards to allow interoperability



# Consolidated architecture

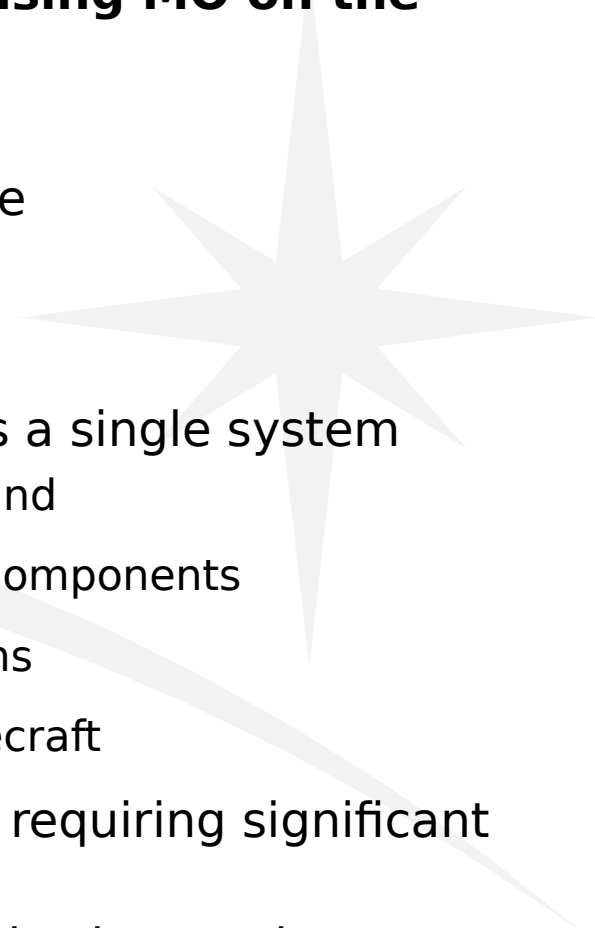
- MO is used as the mechanism for integrating the system
  - The common model is MO service specifications and the COM
- The capabilities of OSRA components are exposed through MO services
  - Custom services to match the component interfaces
- MO is used for OSRA components to communicate if they are on different computing nodes
- MO is used within the OSRA Execution Platform
- SOIS is used to provide interfacing and platform I/O services
  - The capabilities of SOIS devices are exposed through MO services
  - Custom services to match the device interface
  - Other SOIS services are mapped to MO services
- Nodes which do not require OSRA/SOIS are unaffected
  - Just use MO as envisaged by MO
- Staged migration from PUS is possible (and valuable)





# Transition to the consolidated architecture

- Consolidated architecture can be applied **without utilising MO on the space-ground link**
  - i.e. can retain PUS on the space link
- Most important aspects of the consolidated architecture
  - Model exchange between system elements
  - Raised semantic level of operations
- In a system with a single spacecraft which is treated as a single system
  - MO is not necessary onboard to enable use of MO on ground
  - High semantic level of OBSW can be introduced through components
  - Components can appear as MO services to ground systems
  - MO to TM/TC (e.g. PUS) bridge used to interact with spacecraft
- Adds significant value to space-ground system without requiring significant changes to onboard Execution Platform
- Valuable **migration path** to use of all of the technologies in practice



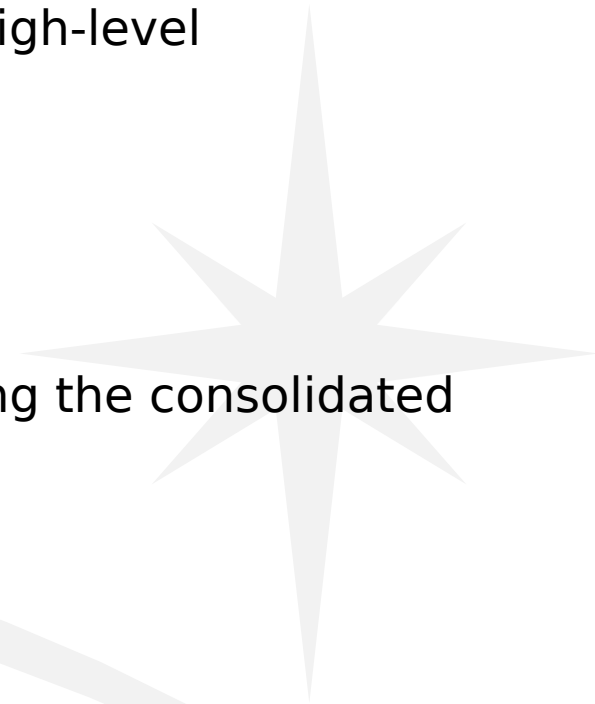
# Harmonised architecture





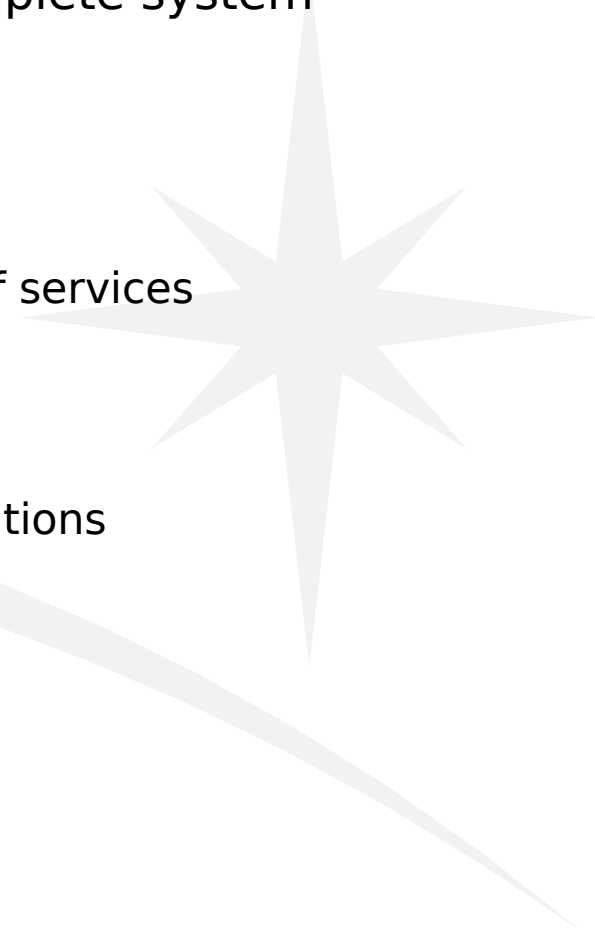
# Beyond the consolidated architecture

- Consolidated architecture aims to meet consolidated high-level requirements
  - But balances these against technology “requirements”
  - Aims to keep current technologies largely intact
  - Short-term approach
- Harmonised architecture is entirely focussed on meeting the consolidated high-level requirements
  - Aims to learn from technologies
  - Does not aim to reuse technologies as-is
  - Suggestions a longer-term migration path
- Adoption of the harmonised architecture offers the greatest benefits
  - Also solves a number of existing problems, both technical and industrial
  - However, a number of significant problems must be solved
- Harmonised architecture is the long-term approach



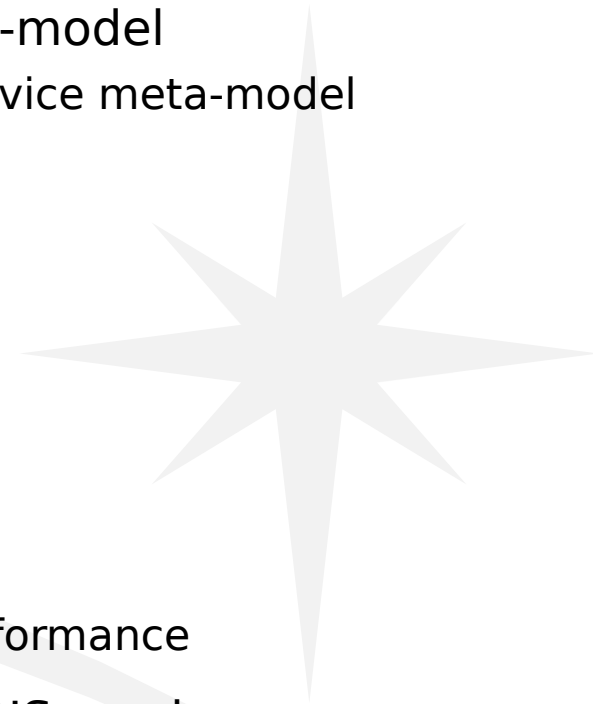
# Harmonised architecture vision

- Create a single conceptual architecture across the complete system
- One component meta-model for all elements
  - Combine component- and service-oriented approaches
  - Accommodate static and dynamic binding
  - Based on the OSRA component model with the addition of services
- One model to represent the complete system
  - The same model across development and run time
  - Model is dynamic: evolves during development and operations
  - Model can be queried at design and run time
- Key aspects of deployment captured in the model
  - Logical deployment (as SSM) c.f. MO domains
  - Physical deployment c.f. MO network zones
- Much more powerful approach than consolidated architecture



# Moving to the Harmonised Architecture

- Requires a new concept of a component → a new meta-model
  - To accommodate OSRA component meta-model + MO service meta-model
  - Accommodate dynamic binding
- Model needs to be stored in a dynamic way
  - Capture meta-model as COM objects?
- Introduce optional separation of concerns
  - Better support for reuse
  - Better support for analysability
  - Can combine concerns where needed for flexibility or performance
- Could extend harmonised architecture to also cover SOIS services
  - Long-term future



# 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*

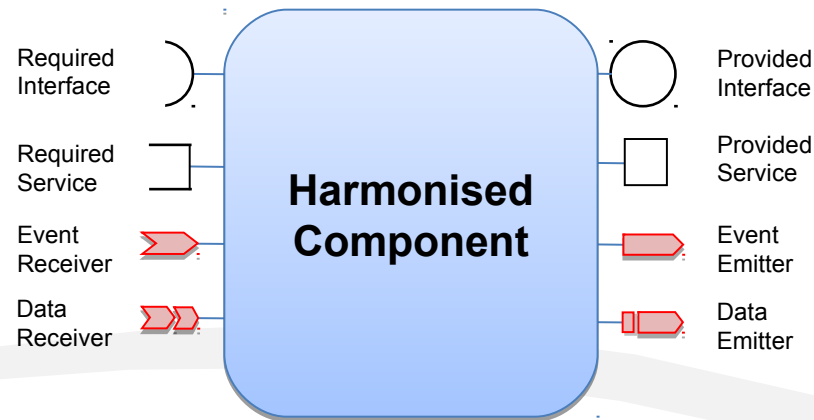


# The harmonised component model

- Start with the OSRA component meta-model
- Rationalise and improve type system
- Relax constraints on separation of concerns
  - Components which obey separation of concerns are marked as “pure”
- Add services as a first-class part of a component specification
  - Service operations similar to those in the MO service meta-model
  - Introduce different types of service bindings permitting dynamism
- Introduce standard component services
  - Reused from MO with minor modifications
  - Action, Parameter, Event, Data
  - These map onto other component model artefacts (e.g. interface attributes)
- Standard framework services introduced to permit introspection and dynamic binding (where required)
- Also introduce framework services for component persistence

# Components in the HCM

- Extend component model to include services
- Example graphical notation



- Binding of components is more of a challenge using OSRA-style tooling
- Need to support more dynamic bindings
  - Introduction of new views
- Tooling expected to be used across development and operations
- Operational view of the spacecraft identical to development view
  - Still based on components

# Implications of the HCM

- Model is a complete snapshot of the system
- All types of functionality appears as a component
  - Including scripts and OBCPs (OBOPs and OBAPs)
- History of system is captured in the model
  - Audit trail
  - Includes the addition or removal of scripts
- Model supports assurance
- Allows seamless interaction with simulation
  - During development and operations
- Model can be used to incorporate SOIS services
- The monolithic Execution Platform shrinks in size considerably
  - Most elements can be represented as components
  - Can **reuse existing implementations** wrapped and modelled as components



# Analysis outcomes and recommendations





# Analysis outcomes

- Architectural design process brought to light inconsistencies between the technologies
  - Should be addressed to promote interoperability
  - Even if consolidated/harmonised architecture is not used
- Suggested changes to all three
  - Many issues were common across all technologies
- Some changes already captured in the scope of other activities
  - e.g. issues with SOIS



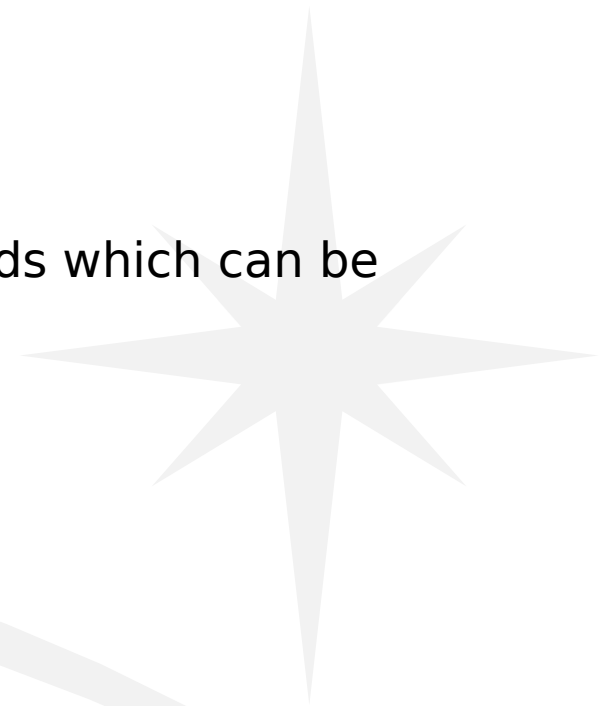
# Technology Recommendations

- Recommendations for the OSRA
  - Rationalisation of the type system
  - Simplifications and modifications to meta-model for flexibility
  - Reconsideration of concept for some Execution Platform services
- Recommendations for MO
  - Rationalisation of type system
  - Extensions to Parameter/Aggregation services
  - Separation of the service and object meta-model from services (inc. MAL)
- Recommendations for SOIS
  - Introduction of a dynamic reference architecture, especially for scheduled subnetworks
  - Architectural adjustments for Command and Data Acquisition Services
  - Refactoring services for better alignment
  - Drop MTS in favour of MAL



# Other recommendations

- Some minor recommendations for PUS-C
  - Automation and layering
  - Parameter Service definition
- Some recommendations for CCSDS to develop standards which can be used across multiple areas
  - A single, robust, syntax-independent **type system**
  - A single, domain-specific **ontology**



# Consolidated architecture prototype



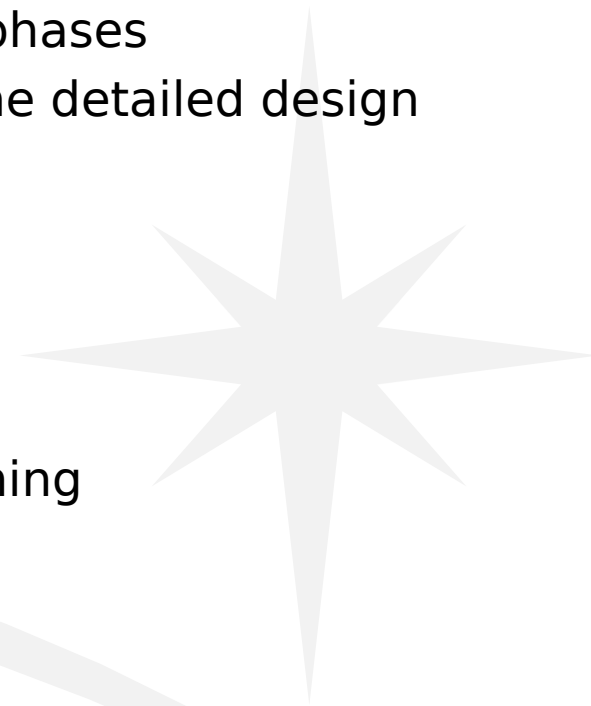
# Prototype aims and scenarios

- Explore key areas of the consolidated architecture
- Examine as many areas as possible
  - Focus on architecture rather than implementation accuracy
  - Breadth rather than depth
- Adopt a scenario-driven approach which covers
  - The important User Needs
  - The expected migration approach
- Scenarios
  - MO space-ground interface
  - MO service within a PUS mission
  - Replacing PUS with a full MO services-based implementation
  - Model-based development with tooling support
  - Automation and movement of functionality between flight and ground



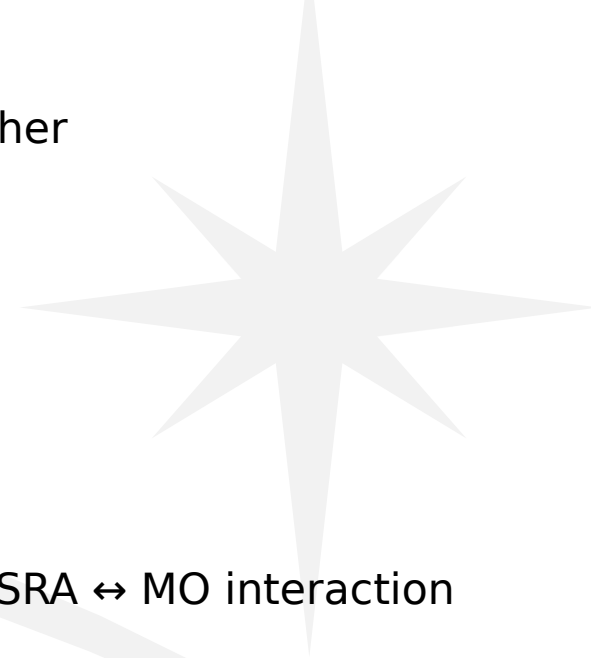
# Prototype approach

- Iterative approach across design and implementation phases
- Early iterations focus on prototyping and working on the detailed design
  - De-risking parts of the architecture and technologies
  - Familiarisation with tooling
- Later iterations focus on implementation
  - Including prototyping of tooling
- Meetings at start/end of iterations for review and planning
  - Including the customer in all meetings
- Leverage existing work
  - TASTE toolchain
  - OSRA/COrDeT tooling
  - OPS-SAT MO adapter for SCOS 2000
  - ESA MO implementations: MAL, tooling, space packet bindings



# Next steps

- Prototyping so far has focused on investigating architectural concerns
  - Onboard implementation using components
  - Including both MO and PUS approaches alongside each other
  - Utilising MO within a component-oriented architecture
- This is leading to development of a detailed design
  - Detailed design review by the end of November
- Next steps are to
  - Expand architectural prototyping
  - Prototype elements of development toolchain including OSRA ↔ MO interaction
  - Increase coverage of prototype MO and PUS services
  - Add automation support and ground HMI to improve demonstration value
- Lessons learned will be captured alongside current recommendations



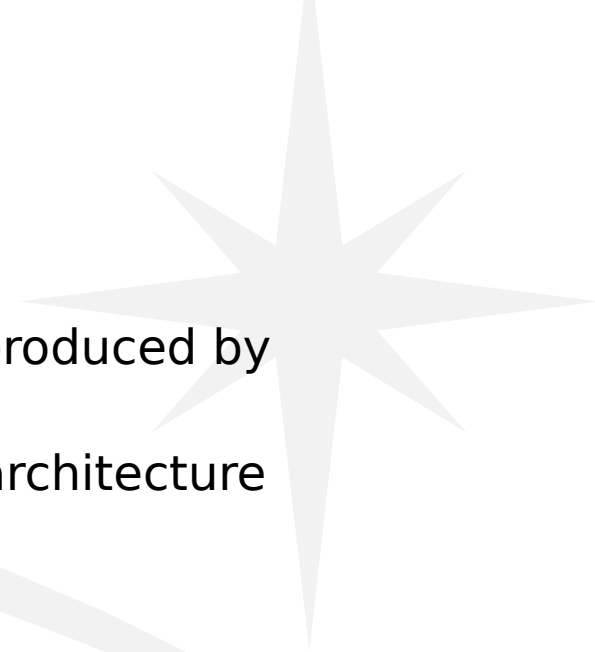
# Summary and conclusions





# Conclusions (1)

- MOSS examines three technologies starting with their user needs and requirements
  - Shared objectives
  - Overlapping scope
  - Complementary approaches and concepts
- A consolidated set of user needs and requirements is produced by combining the three
- Results in a vision for a consolidated and harmonised architecture
  - Spanning complete space-ground systems
  - Spanning development and operations
- Architectural design focuses on near term evolution
  - Consolidated architecture
  - Offers significant value for limited changes to technologies
  - Flexible migration paths limiting impact on existing implementations



# Conclusions (2)

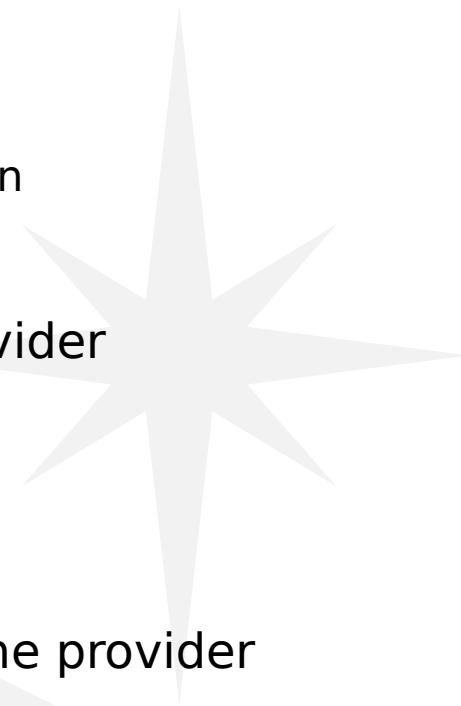
- Longer term Harmonised Architecture is outlined
  - Greater value obtained
  - Requires more work in defining and prototyping a harmonised component model
  - Tooling support is crucial
- Prototyping aims to investigate the highest risk areas for consolidation
  - Breadth rather than depth approach
  - Iterative development used to control risks and obtain maximum value
- Feedback and recommendations made on each of the technologies
  - Additional feedback to PUS-C
  - Also feedback for CCSDS as a whole
- Will be demonstrating ideas using the prototype
- Will be making an argument for harmonisation based on top-down analysis of operability and development user needs
- Will be updating recommendations and feedback at the end of the study

# Backup slides

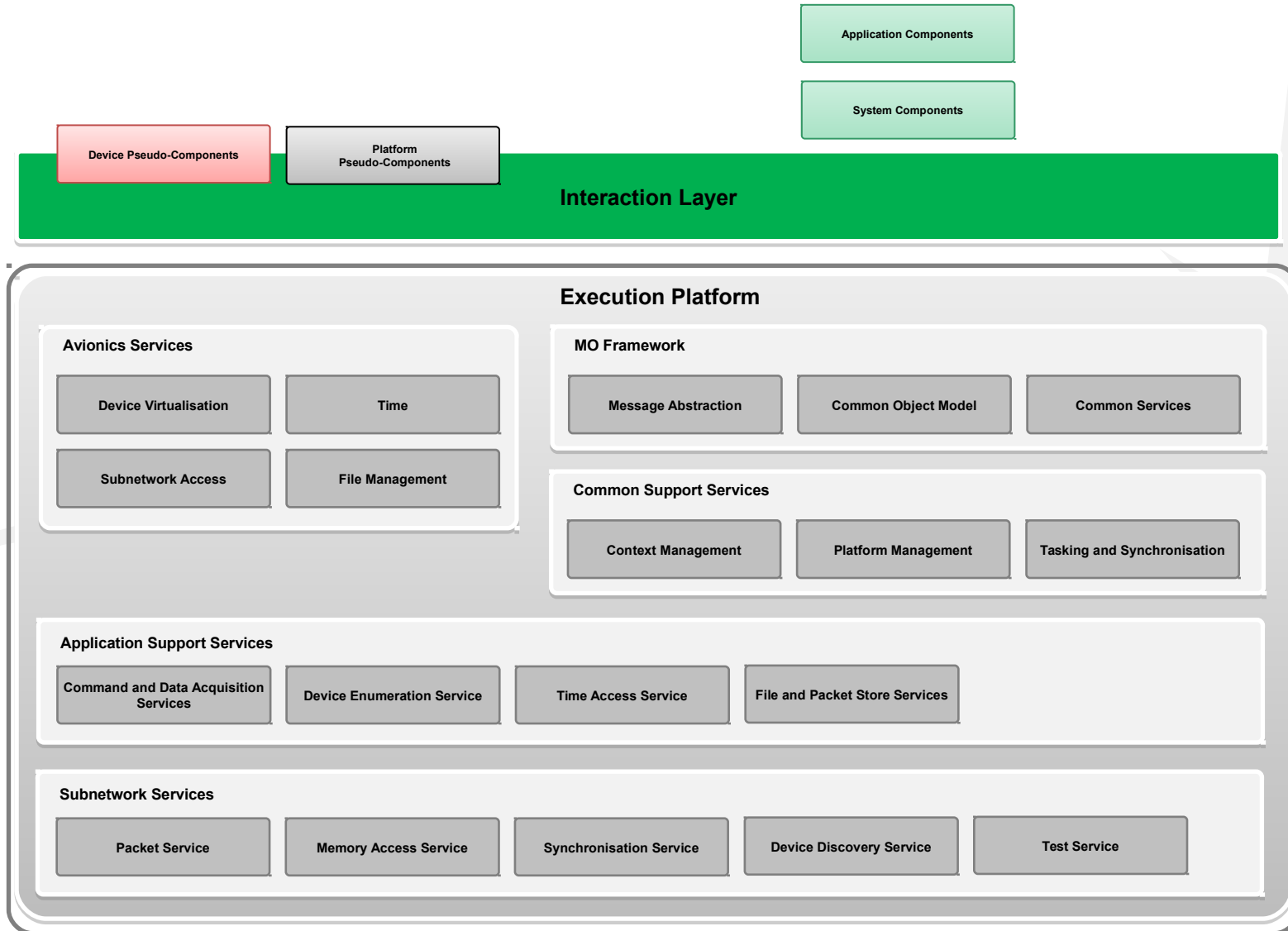


# Operations

- Service and component operations are far less clear
  - They *look* the same (or similar)
  - They could possibly be coerced into performing the same function
  - That does not mean that they **should**
- Invoking a service operation you *may* get to choose the provider
  - In a location-independent way
  - Forms the basis of dynamic binding
  - Publish-subscribe permitted
- Invoking a component operation you **never** get to choose the provider
  - Key part of static binding
  - Publish-subscribe does not make semantic sense
- Some other differences between MO/OSRA
  - e.g. parameters can be attached to all messages such as ACK and UPDATE



# Harmonised architecture EP



# Recommendations for SOIS

- Many recommendations already captured elsewhere, in addition
- Consider the position, role and interface to DDPS
- Document the reference dynamic architecture, especially in the case of scheduled subnetworks
- Document the expected use of SOIS services for time distribution and synchronisation (both static and dynamic architecture)
- Rationalisation of the EDS type system
- Alignment of the EDS component model with the OSRA
- Generalisation of the Packet Store Service
- Alignment of the File Management Service with MO
- Alignment of the Time Access Service with MO
- Drop the Message Transfer Service in favour of MAL

