CCSDS MO Services, SAVOIR OSRA and CCSDS SOIS

Peter Mendham, Alex Mason
Bright Ascension
Simon Reid, Manuel Gonzalez Vega
RHEA System
Andreas Wortmann, Roland Lampke
OHB System

Overview

- The MOSS activity
- Introduction to the three technologies
  - MO Services
  - OSRA
  - SOIS
- Technology requirements
- Approach to consolidation
  - The consolidated architecture
  - The harmonised architecture
- Prototyping the consolidated architecture
- Feedback 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

Consortium

- Consortium led by Bright Ascension Ltd
  - BAL also acted as onboard software specialist
  - Prior experience of SOIS and the OSRA
- RHEA System accompanied BAL providing wide ground segment experience
  - Act as ground segment specialist
  - Brought prior experience of MO
- OHB System acted as an external assessor and gave feedback at each stage in the process
  - Prior experience of SOIS and the OSRA
  - Brought team members with onboard, ground and operability focus
  - Provided a high-level, top-down “reality check”

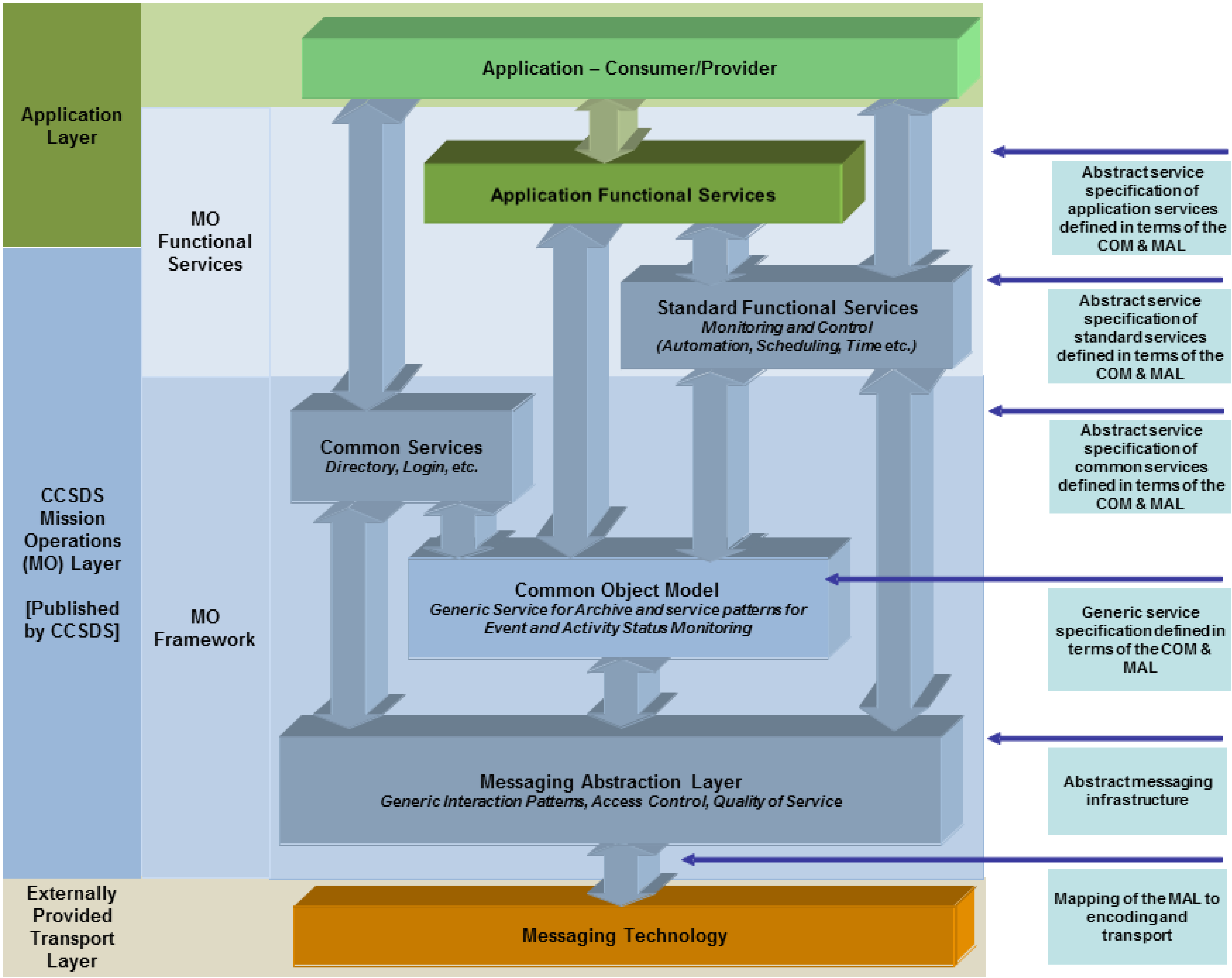
Technology review

MO: vision

- Service infrastructure for large-scale distributed system
  - Spans local area, wide are 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


Application Layer
MO Functional Services
Application – Consumer/Provider
Application Functional Services
Standard Functional Services
Monitoring and Control
(Automation, Scheduling, Time etc.)
CCSDS Mission Operations (MO) Layer
[Published by CCSDS]
MO Framework
Common Services
Directory, Login, etc.
Common Object Model
Generic Service for Archive and service patterns for Event and Activity Status Monitoring
Messaging Abstraction Layer
Generic Interaction Patterns, Access Control, Quality of Service
Externally Provided Transport Layer
Messaging Technology
Abstract service specification of application services defined in terms of the COM & MAL
Abstract service specification of standard services defined in terms of the COM & MAL
Abstract service specification of common services defined in terms of the COM & MAL
Generic service specification defined in terms of the COM & MAL
Abstract messaging infrastructure
Mapping of the MAL to encoding and transport

Relationship with PUS(-C)

- By design, all PUS services are planned to be covered (functionally) by MO
- Key services already covered
  - Mostly by MO M&C Service
- Many PUS services to be covered by forthcoming MO services

PUS Service	CCSDS MO Service
[1] request verification	Use of COM Activity Tracking service pattern
[2] device access	M&C – Action
[3] housekeeping	M&C – Aggregation
[4] parameter statistics reporting	M&C – Statistic
[5] event reporting	M&C – Alert
[8] function management	M&C – Action
[12] on board monitoring	M&C – Check
[20] parameter management	M&C – Parameter

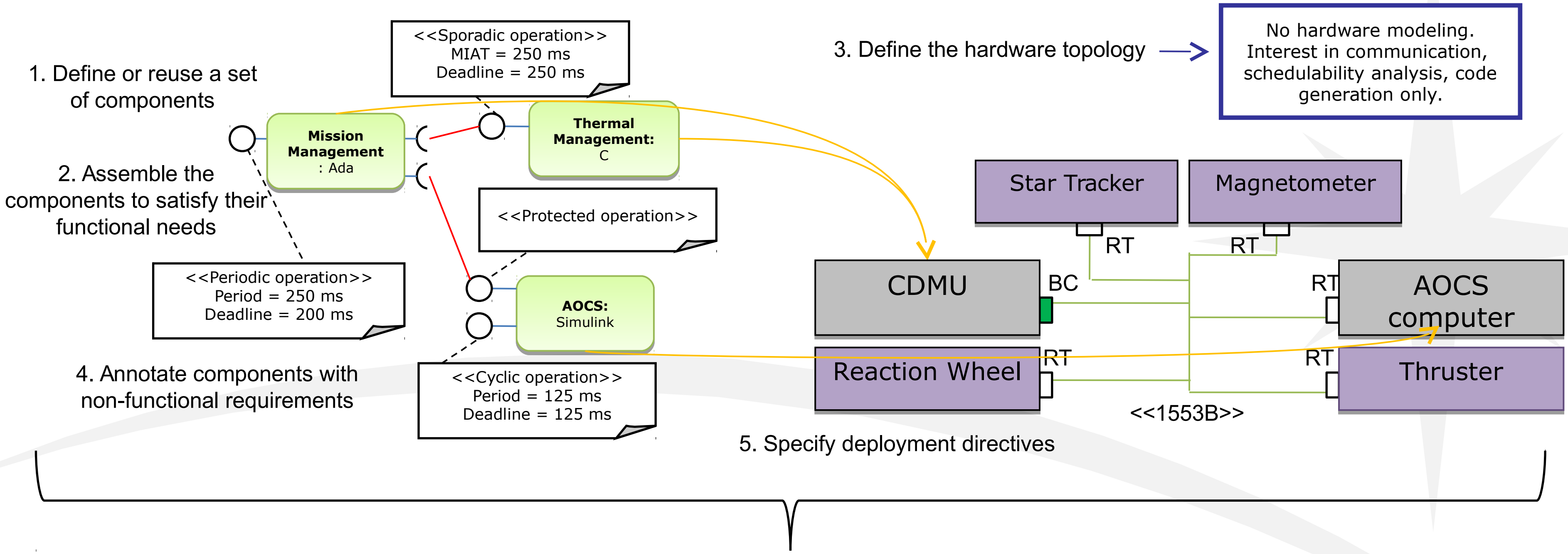
MO: use cases

- 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

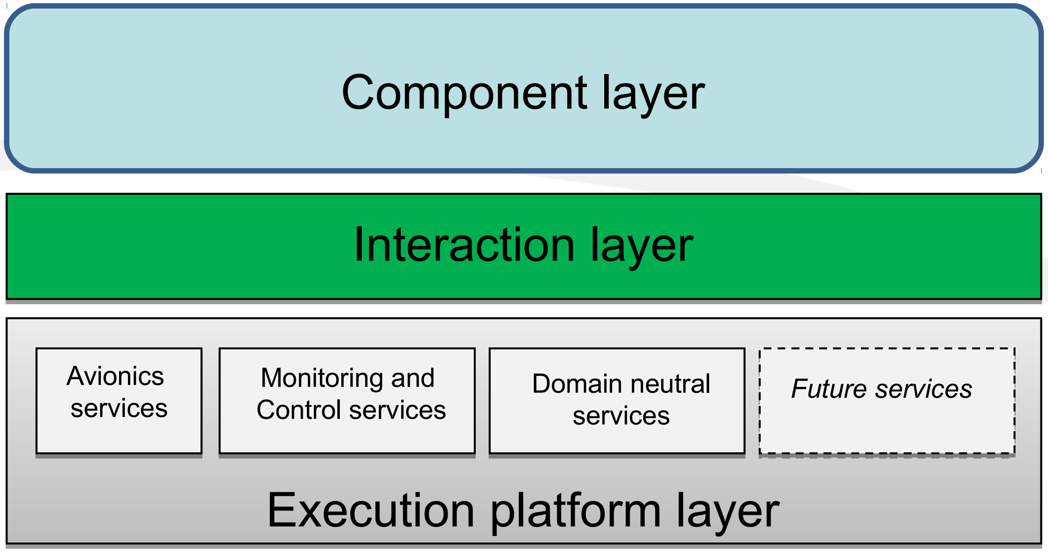


6. Automatically analyse the software **model** to ascertain that the **whole set** of non-functional properties can be fulfilled

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 I**nteraction Layer**



OSRA: use cases

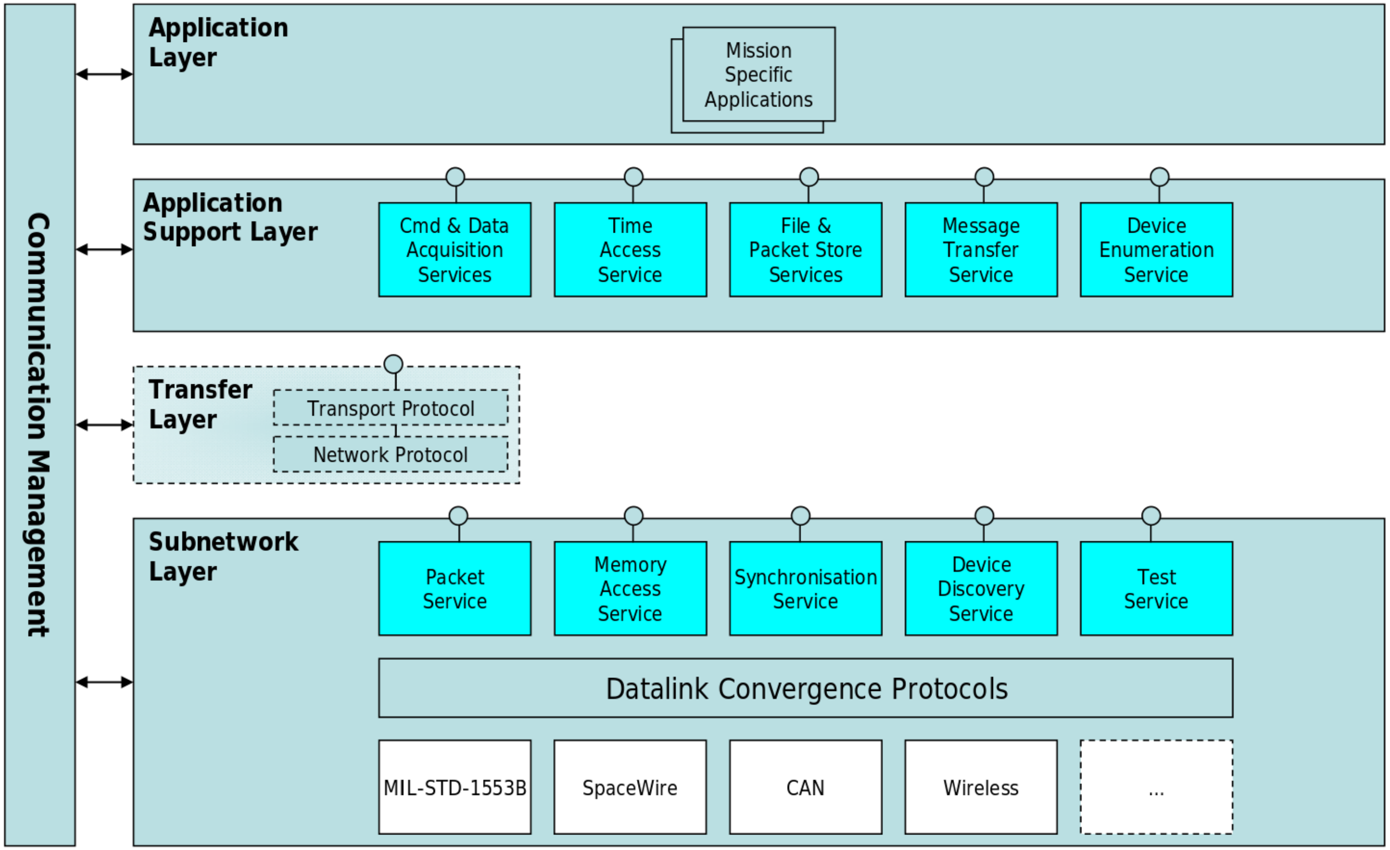
- Short term focuses on improving the development workflow
  - Software reuse within organisations, automation of simple analyses
- Long term focuses on extending the role of the OSRA
  - Software reuse across organisations and the introduction or products
  - More complex analyses and support to assurance
- 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 abstract services representing interactions between flight software and hardware
  - Increase potential for interoperability and reuse by creating a reference approach
- 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 abstract services, each defined by their interface
- Concrete services or 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
  - (Conceptually) 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
  - In the OSRA, tooling is used to *remove* design-time layering and abstraction at run-time
    - Could also apply to SOIS depending on how standards are interpreted
- 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

Concerns from technology analysis

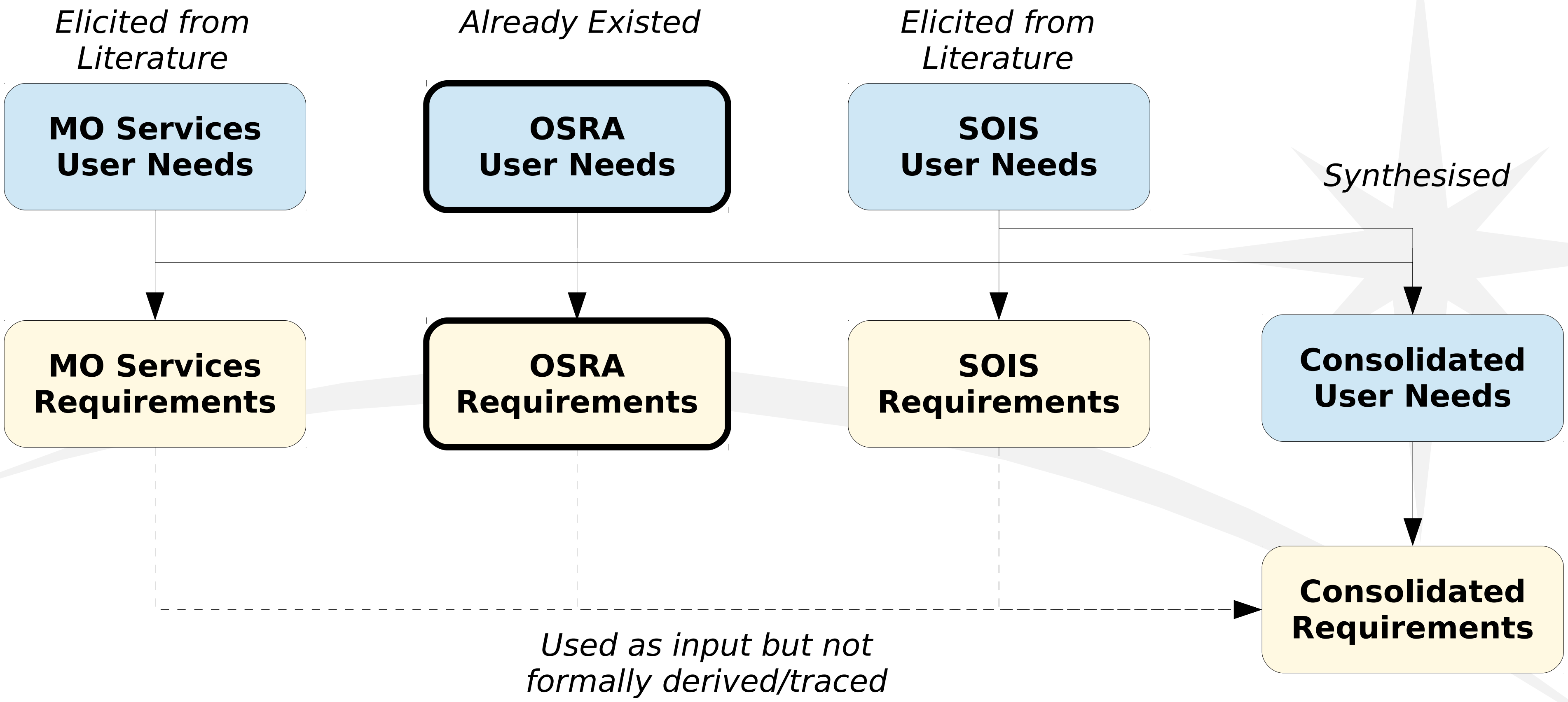
- Concerns with MO
  - Documentation and definitions insufficiently organised according to stakeholder
  - Insufficient distinction between framework and services
  - Unclear overlap and relationships with other standards
- Concerns with the OSRA
  - Too limiting in places (strict separation of concerns)
  - Some aspects not stressed enough (analysability/FDIR)
- SOIS
  - Insufficient adoption
- In general
  - Excessive layering and inefficiency
  - Technical/semantic gap between the technology and the underlying implementation
    - Can only be addressed through good tooling

Technology requirements

Consolidated user needs/requirements

- Elicit/extract User Needs and High-Level Requirements for all three technologies
  - Only the OSRA had these formally captured
- Bring together superset of User Needs and combine where they align
- Determine an overall “vision” of the consolidated architecture based on these User Needs
  - Addresses development-time and operational concerns
- Extract High-Level Requirements from Consolidated User Needs with reference to technology requirements and overall vision

Requirements Derivation



Requirements summary

- Created
  - 24 consolidated User Needs
  - 16 consolidated High-Level Requirements

- Highlights:
  - 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 treated as a whole for
    - Development
    - Operations

Technology consolidation

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

Creating a reference architecture

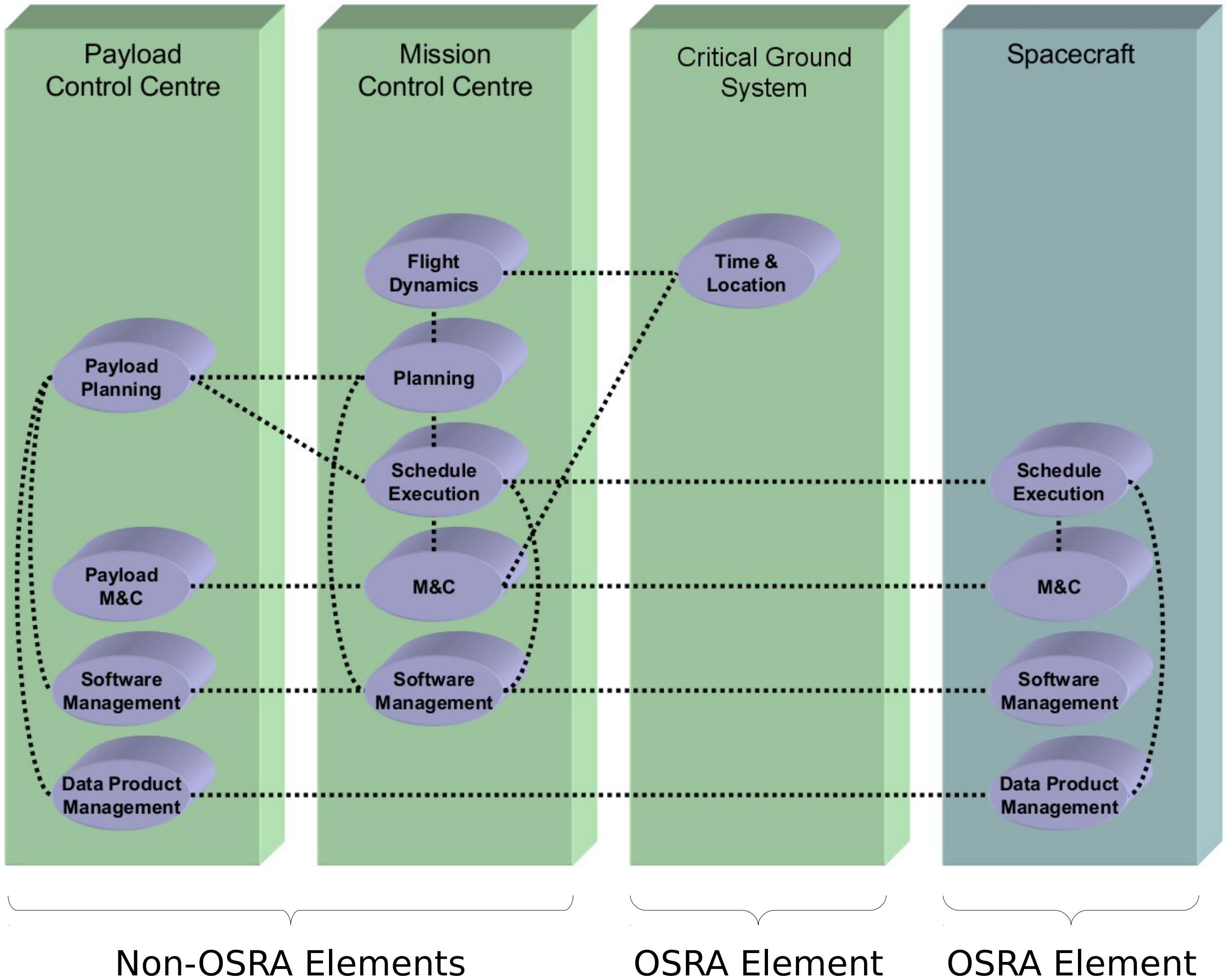
- Although the technologies have compatible goals there are some details which lead to difficulties in integration
- This led to two approaches
- In the **short term**
  - Change as little as possible
  - Aim to achieve most, but not all, of the consolidated User Needs
  - Provide a fully integrated space-ground system
  - This is the **Consolidated Architecture**
- In the **long term**
  - Allow more to be changed
  - Aim to achieve all consolidated User Needs
  - Provide a full integrated meta-model
  - This is **Harmonised Architecture**
- This also aligns with the short/long term goals for each technology

Consolidated architecture

Consolidation approach

- Allow the application of the OSRA and SOIS within an MO architecture
- Identify specific regions or *elements* of an MO system
  - Based on characteristics (e.g. timeliness, restricted resources)
  - 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

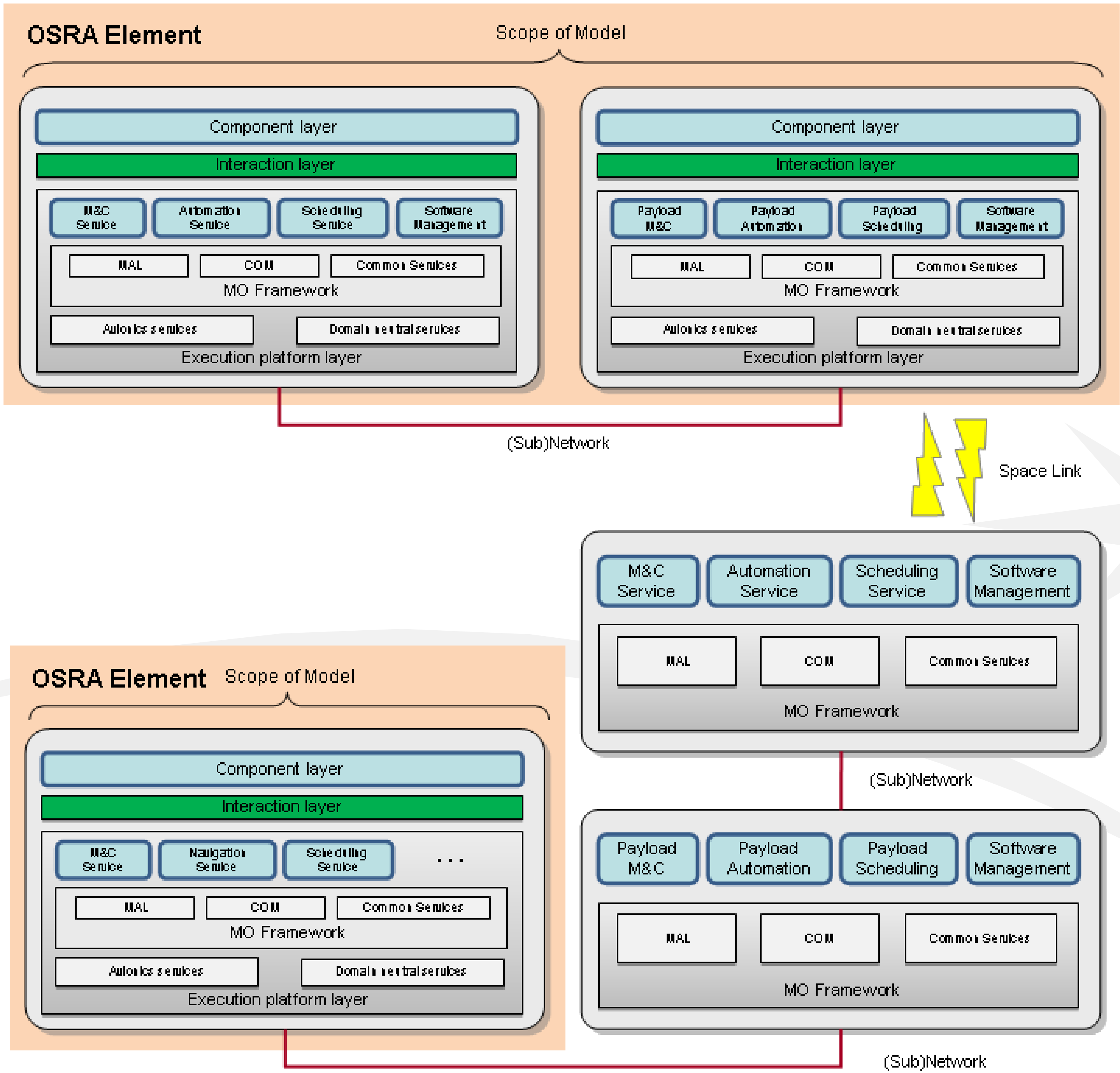


The consolidation vision

- 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)



OSRA Element
Scope of Model
Component layer
Interaction layer
M&C Service
Automation Service
Scheduling Service
Software Management
MAL
COM
Common Services
MO Framework
Avionics services
Domain neutral services
Execution platform layer
Payload M&C
Payload Automation
Payload Scheduling
(Sub)Network
Space Link
Navigation Service
Scheduling Service
Software Management

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 through **service specifications**
  - 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

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

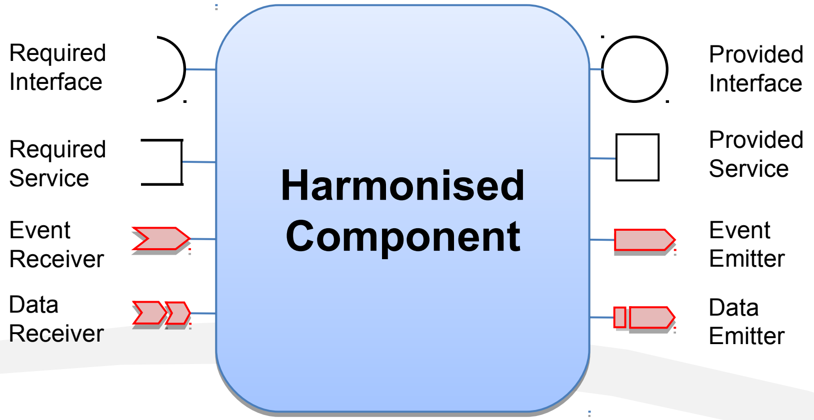
- A component interface and a service are on different semantic levels
- Component interfaces (as defined by the OSRA)
  - Describe *what*
  - For example
    - An attribute
    - An event
  - Bindings bind a *thing* to a *thing*
    - e.g. an attribute to an attribute
- Service interfaces (as defined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



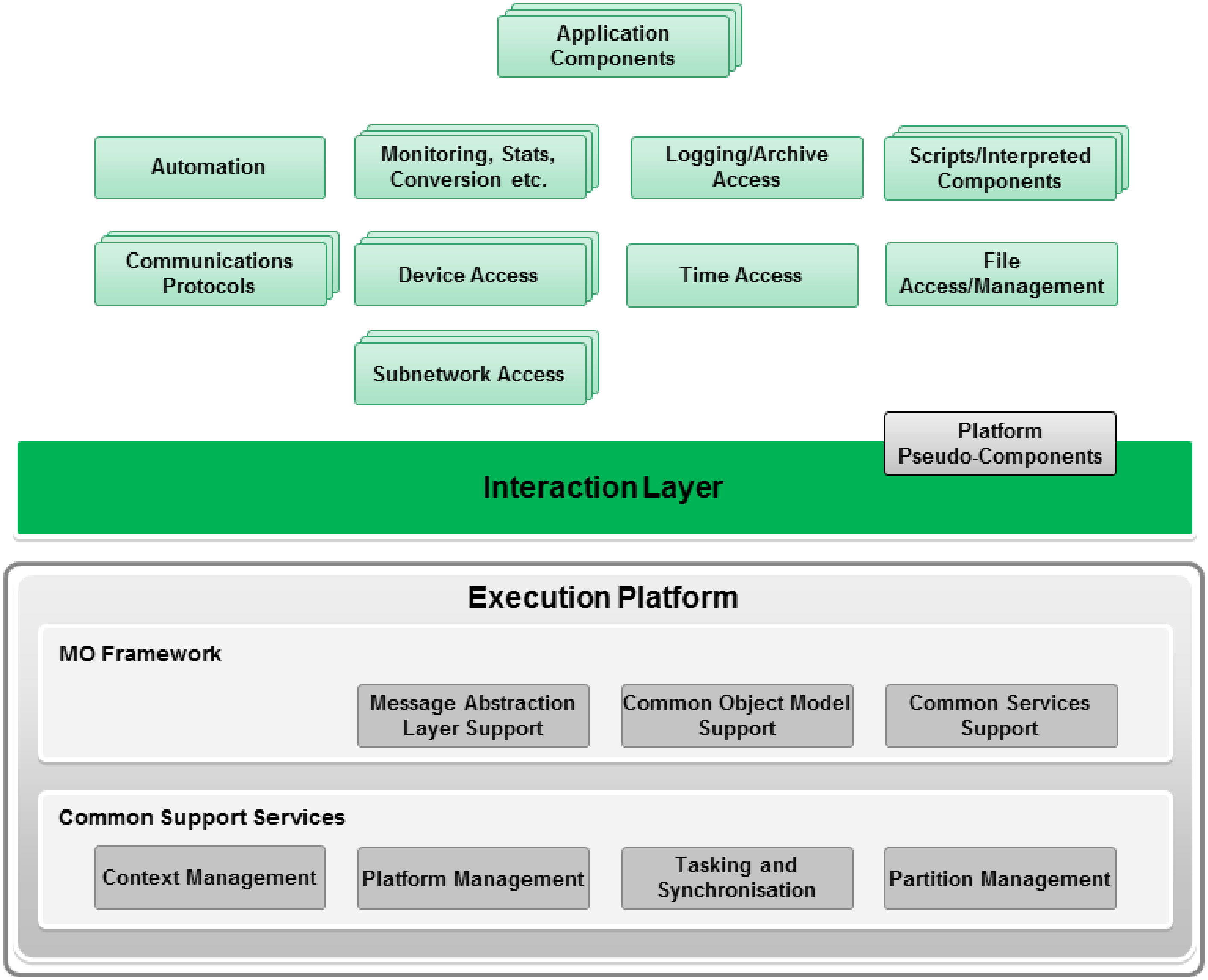
- Binding of components is more of a challenge using OSRA-style tooling
- Need to support more dynamic bindings
  - Introduction of new views
- Tooling expected to be used across development and operations
- Operational view of the spacecraft identical to development view
  - Still based on components

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

Harmonised architecture layers



Majority of functions built, validated, maintained and reused as components

Execution Platform contains only core features necessary to support 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
  - Better documentation and framing of SOIS as a reference, not concrete services
  - 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

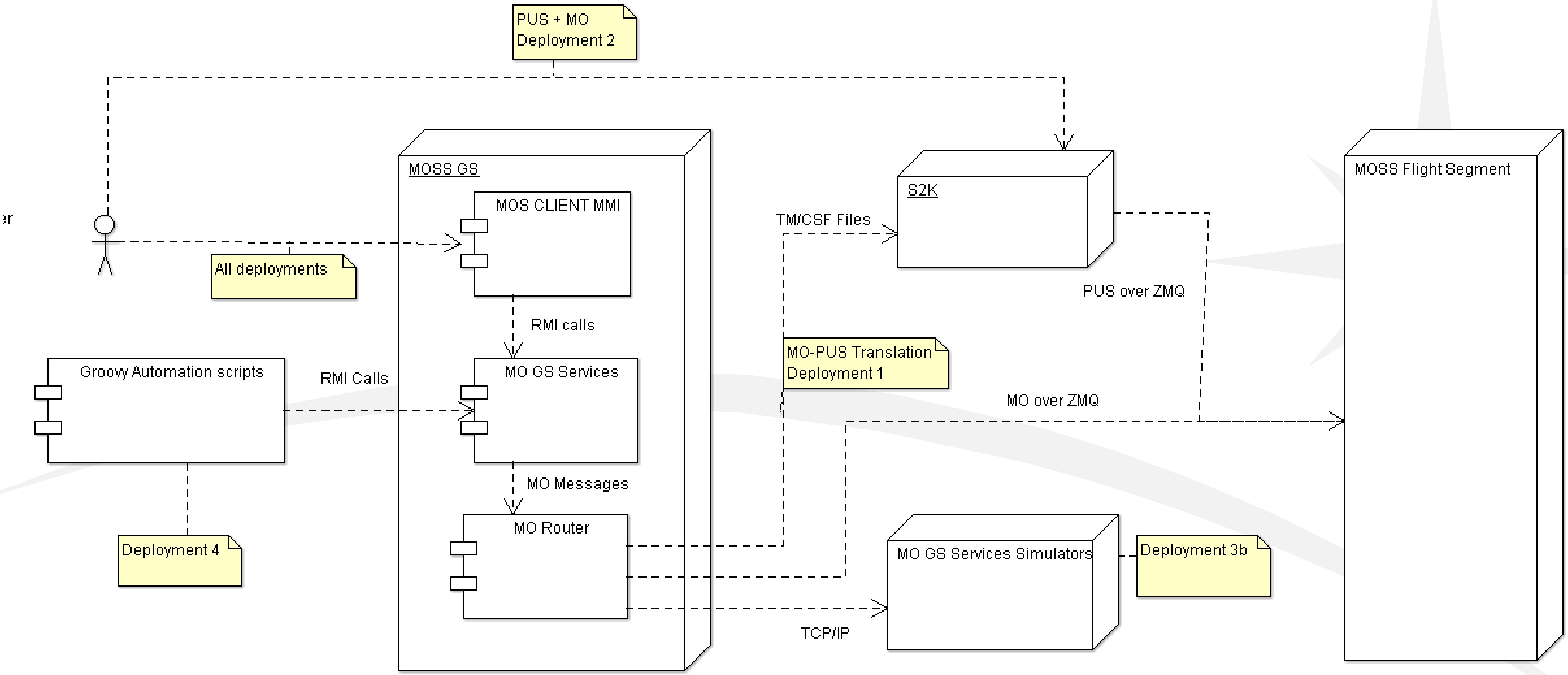
Prototype goals

- Demonstrate the consolidated architecture
- Investigate the proposed relationship between the three technologies
- Examine the practicality of implementing the consolidated architecture
- Investigate relationship between the consolidated architecture and PUS
    - Especially the use of PUS over the space link whilst retaining MO services
- Review potential migration steps towards a consolidated or harmonised architecture

Demonstration scenarios

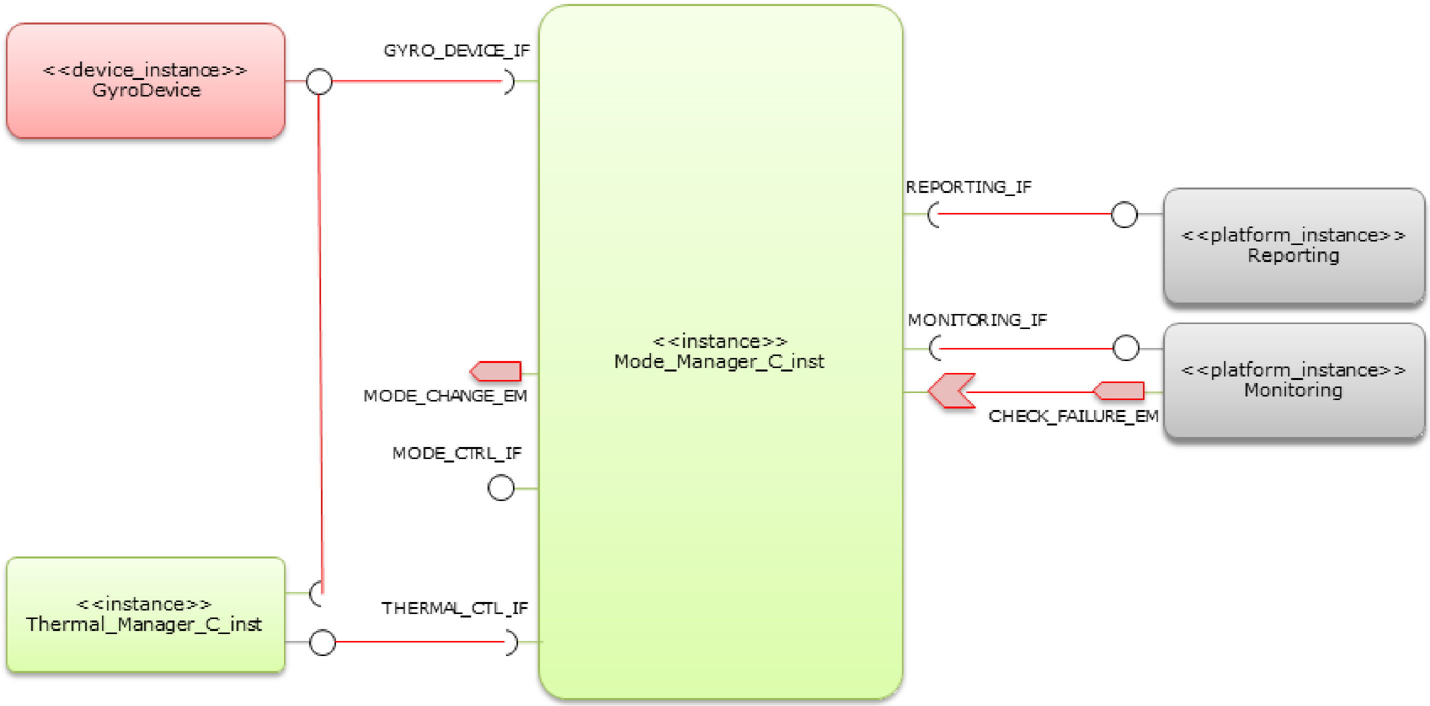
- Scenario A: Fully PUS mission with an external MO interface
  - Fully PUS space segment
  - PUS ground segment offers an external interface through MO
- Scenario B: Largely PUS mission with MO services
  - One onboard MO service on a PUS spacecraft
  - MO service accessed using MO API on the ground but via PUS
- Scenario C: Replacing PUS with a full MO services-based implementation
  - Fully MO space segment (complete spacecraft)
  - Fully MO ground segment
- Scenario D: Development scenario
  - Examine automation in the development tooling
  - Flight and ground segments
- Scenario E: Automation
  - Automation added to ground segment

Ground architecture

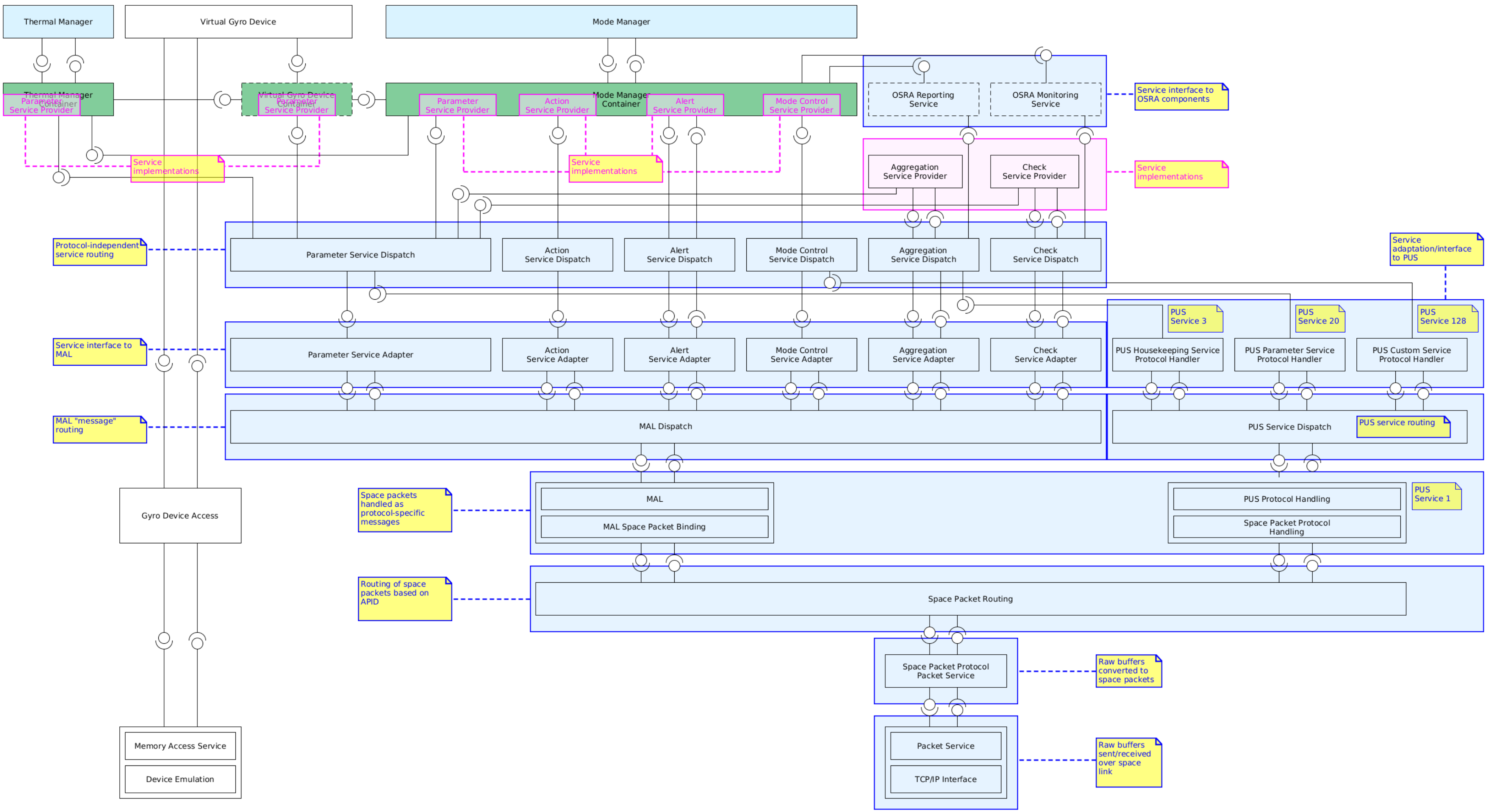


Onboard architecture (1)

- Representative use of
  - Components
  - Device pseudo-components
  - M&C pseudo-components



Onboard architecture (2)



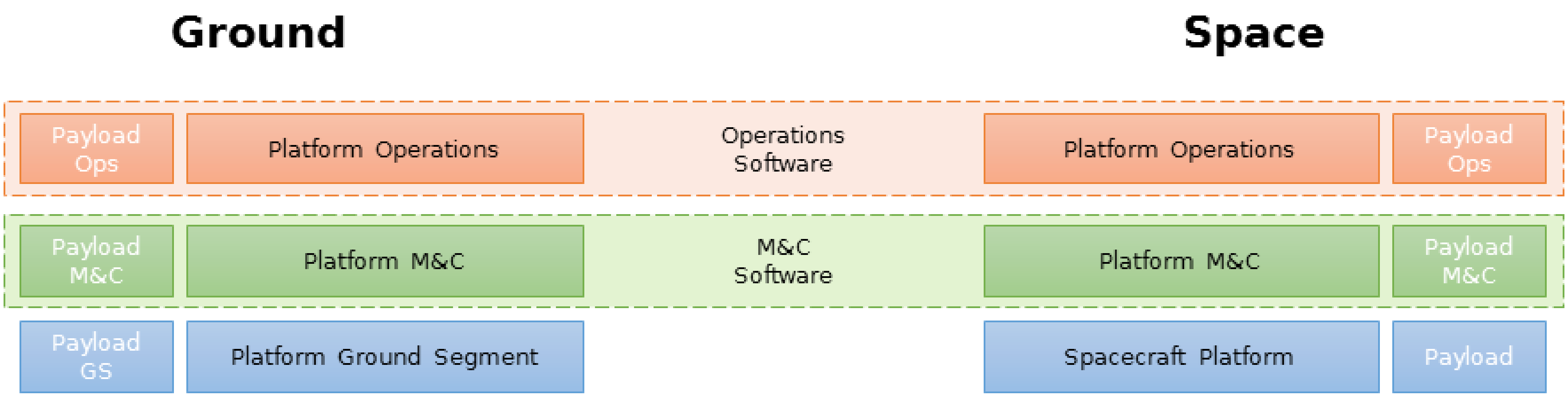
Lesson learned

- Key value of the consolidated architecture is MO **service specifications**
  - Permits a single description to be used across space-ground
  - Can be used independently of implementation
  - With a specified mapping, can be used with PUS
- The onboard part of the consolidated architecture is valuable
  - Consistent approach to modularity
  - Combines MO, SOIS and the OSRA into a coherent model
- Tooling can be used to assist with generation
  - Flight and ground
- There is a straightforward migration path from PUS to MO
  - This is greatly assisted by a modular implementation architecture
  - Valuable to do as MO meets long term needs better than PUS
  - PUS and MO can coexist even providing access to the same functions/services

Conclusions and Recommendations

What if....

- Software was not divided into “flight” and “ground”?
- The space-ground ICD was not the interface/barrier between groups/sites?
- What software did was not based around the lowest common denominator at the space link?
- We could focus on the best way to deliver mission results, overall, rather than having to fix space-ground trades very early in development?



Conclusions (1)

- Treating space-ground as a single system is a powerful way of obtaining greater value from a mission
- The three technologies studied are complementary
- The consolidated and harmonised architectures have value
  - Only together can they realise the fully set of User Needs
  - All three technologies have shortcomings
  - Short-term consolidated architecture addresses most UNs
  - Long-term harmonised architecture addresses all UNs
- Aspects of the consolidated/harmonised architectures can be adopted individually
  - Of central importance is the MO/MAL service model
  - MO concepts can be used to add structure to the OSRA Execution Platform
  - Key OSRA concepts could be introduced as a bolt-on deployment model for MO
- A model-based approach would connect well with life-cycle management

Conclusions (2)

- Consolidated architecture is practical for implementation
  - Is likely to introduce overhead when compared to a monolithic implementation
  - Cannot realise the UNs with a monolithic implementation
  - Difficult to compare different modular implementations without considering specifics of implementation
- Technologies are at different states of readiness
  - Core part of MO is ready for use now
    - Need a number of other services defined to cover basics required for a mission
    - Would benefit from meta-model separation for wider use
    - Document restructure would be beneficial to ease learning curve and take up
  - OSRA is largely ready for use now
    - Will benefit from stable SAVOIR standards
    - Some limitations and complexities, especially around non-use of PUS
  - Key concepts of SOIS are very useful but should not be applied naively
    - Apply SOIS as a reference model
    - Needs very careful application

Conclusions (methodology)

- This study had a technology focus
  - Starting point was technologies
  - Requirements derived from technologies (“reverse engineered”)
- Consolidation would always be effectively bottom-up
  - Technology-driven requirements
  - Technology-driven solution
- This is fine but should be aware of the limitations of this approach
- Alternative is to take a top-down approach
  - Base requirements on new analysis of end-to-end User Needs
  - Consider development, programmatic, commercial and operability needs
- This may produce a different result

Recommendations for future work

- Conduct a study on end-to-end User Needs and requirements
  - Create end-to-end software requirements
  - Make recommendations on how software could be handled end-to-end
  - Include programmatic, commercial and standardisation/assurance concerns
- Produce a first version of the harmonised component model
  - Create and document meta-model
  - Small-scale prototype to demonstrate
- Create a core set of tools for the harmonised component model
  - Small set of core tools which offer value and can be applied easily
  - Must be developed suitably assured for flight and ground software use
- Demonstrate use of the harmonised model
  - Target a lower assurance/nano-satellite mission
  - Target an existing mission, *not* a dedicated mission
  - Next step is a payload or something intermediate to full use