ADCSS 2015
SOIS Session

Part 1 – How to use SOIS in current and future flight software

20/10/2015

THALES ALENIA SPACE INTERNAL

ThalesAlenia Space
A Thales / Finmeccanica Company

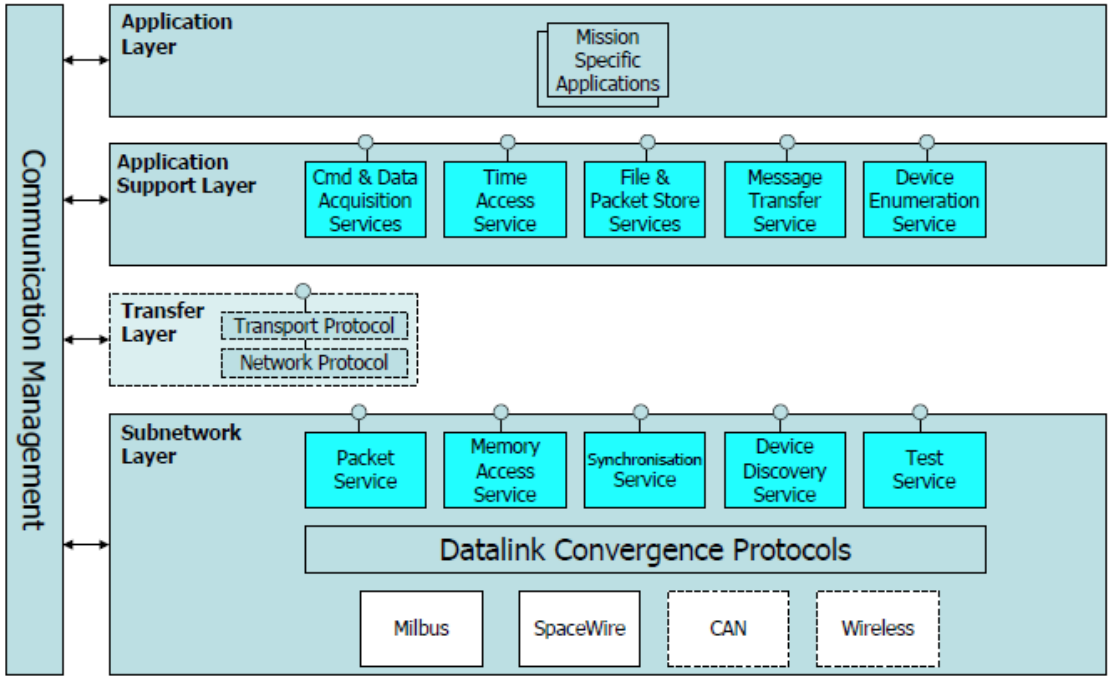
Outline

- The TAS “SAVOIR Communication Architecture” GSTP
  - The reference mission for the GSTP
  - Principles and goals
  - Selection of SOIS services in the TAS GSTP

- Practical results of the GSTP and TAS assessment

SOIS basic constituents

A static service architecture



OSI-level service primitives definition

READ.request (
MASAP Address,
Destination Address,
Transaction ID,
Memory ID,
Start Memory Address,
Size,
Priority,
Channel,
Authorisation (optional))

READ.indication (
MASAP Address,
Destination Address,
Transaction ID,
Memory ID,
Start Memory Address,
Size,
Priority,
Channel,
Data,
Result Metadata)

Service configuration (often non normative)

MIB

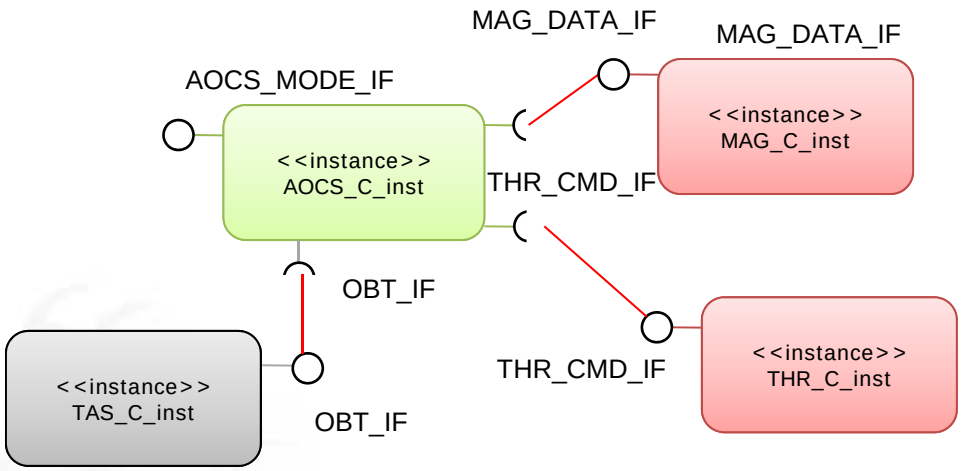
The reference mission for the GSTP

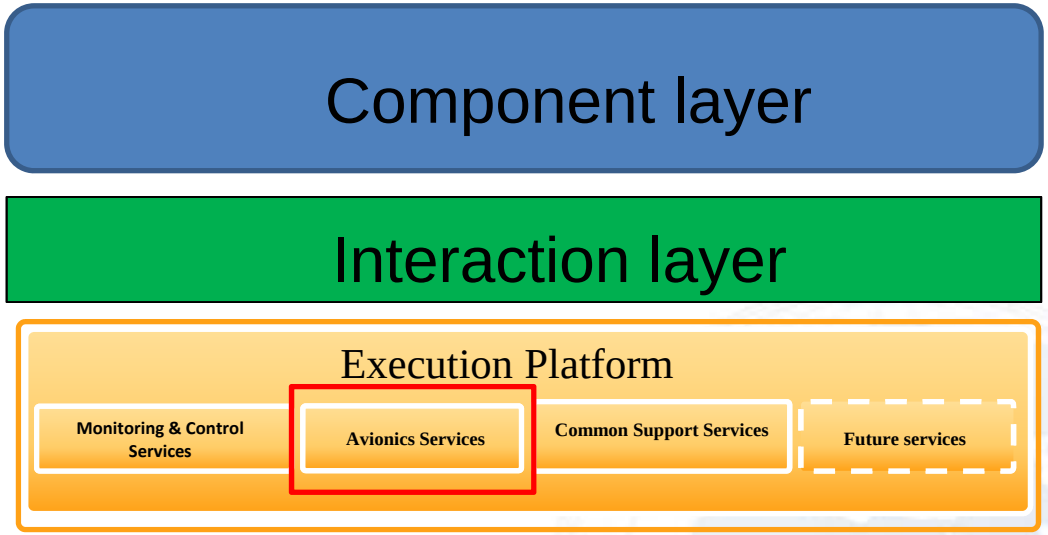
- AOCS Sensors and Actuators
  - 1) On dedicated SpW links
  - 2) Acquisitions / Commanding via remote I/O boards
    - Connected to the OBC via SpW
    - Use of acquisition lists
- Complex payload
  - With dedicated processor
  - Payload RTU comprising equipment with commanding / acquisition via the platform software
- Overall: 2 different SpW links, 1553B, discrete I/O links

- The goal of the GSTP was
  - to replace all existing equipment layer and communication stack, with an implementation of equivalent SOIS services
  - to re-run all the relevant validation tests of the reference mission on the newly developed stack

Principles, goals and challenges (I)

- We tried to implement SOIS in a manner that is compatible with
  - An OSRA-compliant development process
  - Users of the stack are mostly components designed with OSRA-like principles





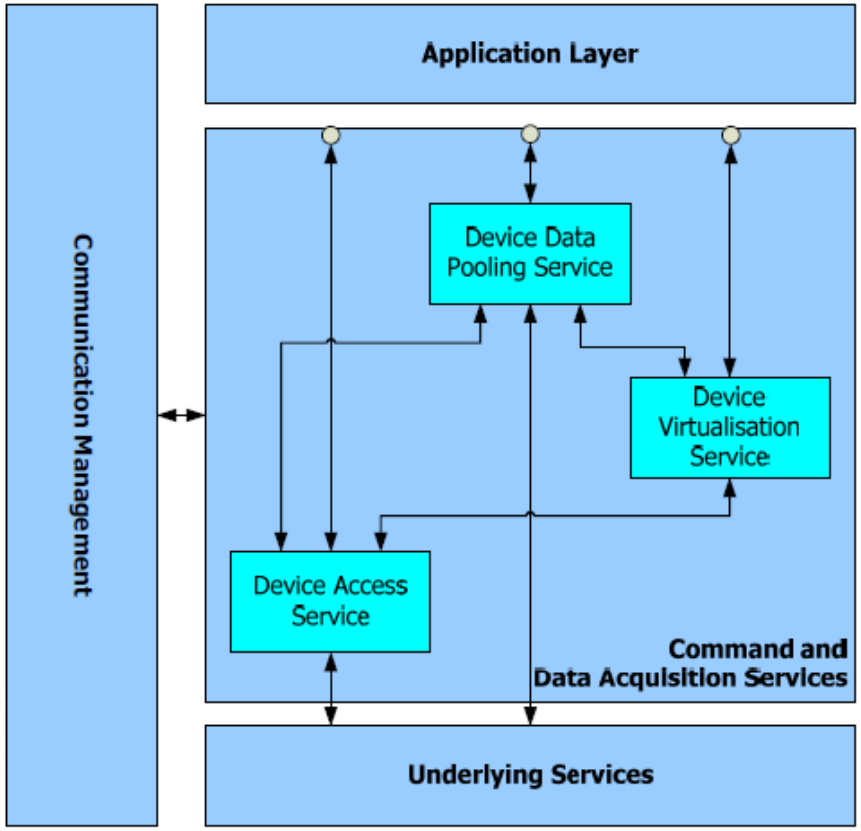
* Figures: Copyright ESA 2014-2015 – OSRA Training Material

Principles, goals and challenges (II)

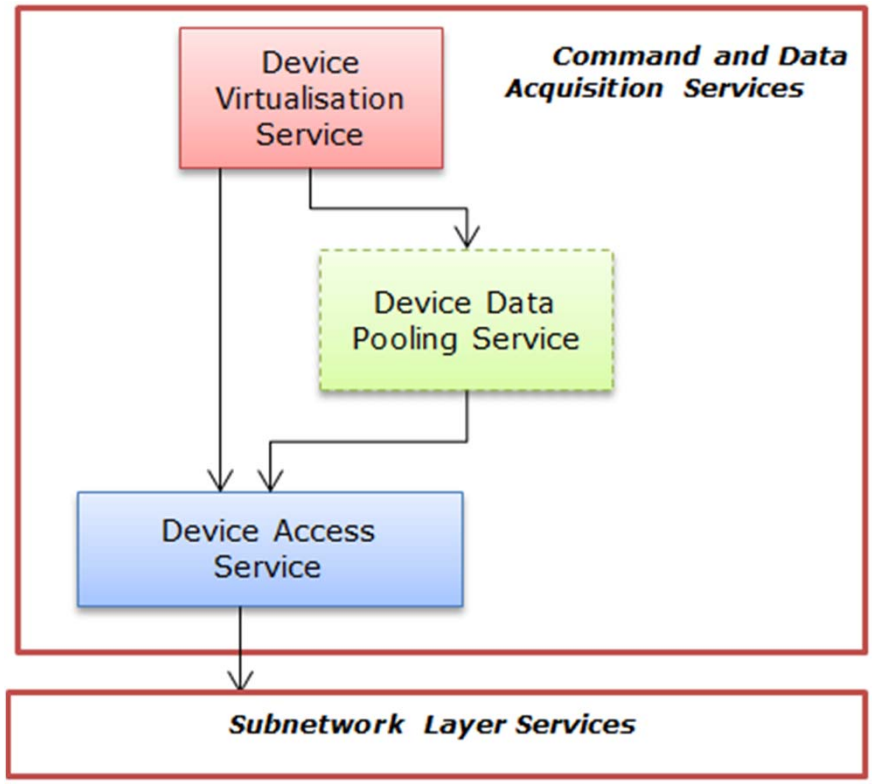
- Goals and challenges
  - Can SOIS easily support current and future avionics technologies and network topologies?
    - Use of RTUs, equipment on communication links more complex than a point-to-point SpW
  - How can we introduce redundancy and cross-strapping management in the SOIS implementation?
  - How can we relate to existing FDIR mechanisms and policies?
  - How do we manage aspects related to the dynamic architecture?
  - How well can we plug the SOIS implementation into an existing TM/TC dispatching system?
  - How can everything be satisfactorily put in place in an existing model-based development process?

Application Support Layer (I)

Original SOIS CDAS architecture



Modified CDAS architecture



Application Support Layer (II)

- Advantages of the new CDAS architecture
  - Device Data Pooling Service (DDPS)
    - Used to decouple access to acquired data from the acquisition
      - In application of the separation of concerns principle
    - Essential to manage acquisition lists on I/O boards or remote RTUs
  - User applications access data systematically via DVS
    - They shall not be concerned about how this data is acquired and on the aggregation in terms of acq lists of those data
  - Access to the subnetwork layer services performed only via DAS
    - Architectural choice to organize systematically communications
    - May incur a penalty in performance

Application Support Layer (III)

Major Service	Service	Use in the GSTP	Comments
Device Virtualisation Service (DVS)	-	Implemented	Refer to modified CDAS architecture.
Device Data Pooling Service (DDPS)	-	Implemented	Refer to modified CDAS architecture. Used to manage acquisition lists.
Device Access Service (DAS)	-	Implemented	Refer to modified CDAS architecture.
Device Enumeration Service (DES)	-	Implemented	Quite unusable as presently defined. Large extension to support redundancy and cross strapping (make it become a full-fledged equipment manager).
File and Packet Store Services (FPSS)	File Access Service / File Management Service	Assessed	No file system used in reference mission. Mapping to CNES “FMS” ~OK.
File and Packet Store Services (FPSS)	Packet Store Access Service	Designed	Considered too low-level, incomplete, sometimes cumbersome to implement. Forces clients to know internal organisation of PS.
File and Packet Store Services (FPSS)	Packet Store Management Service	Not considered	Not considered, as reference mission is operated with PUS ver. A
Time Access Service (TAS)		Designed	Should be extended to account for different types of time sources (wall-clock time vs. logical time in TSP, different time precisions)
Message Transfer Service (MTS)		Not considered	No value added compared to existing mechanisms. Link with AMS brings unnecessary complexity.

Subnetwork Layer (I)

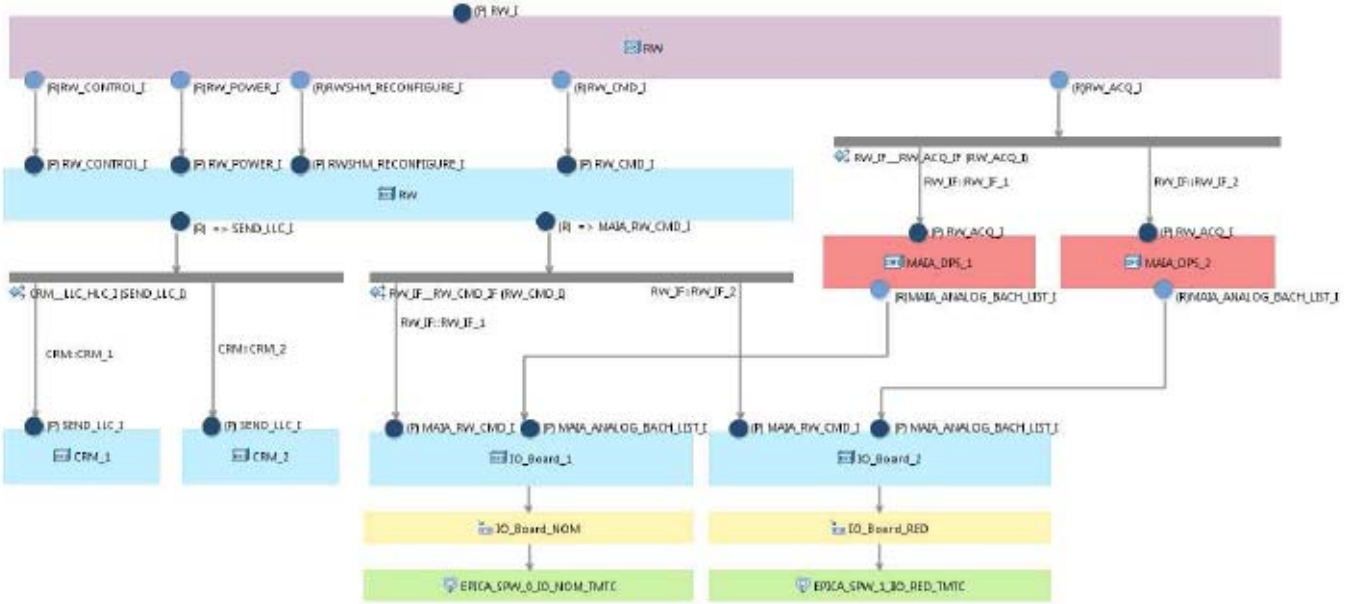
- PS / MAS were implemented to support all communication links
- The relationship between DAS and PS / MAS was quite debated internally during the project
  - Whether buffering of requests at PS / MAS was desirable
  - Implementation-specific interpretation of Channel / Priority to map towards underlying links
    - Mandatory for any complex link or for scheduled communications
  - Whether the DAS / PS / MAS chains are adequate when much of the communication effort is performed by intelligent controllers
  - Relationship between DAS and PS / MAS and their task executors was arbitrarily chosen
    - Lack in SOIS of any view for establishing a dynamic architecture, when necessary
    - An implementation in line with separation of concerns was chosen (e.g., declaration of entry-points, separate allocation to tasks)

Subnetwork Layer (II)

Major Service	Use in the GSTP	Comments
Packet Service (PS)	Implemented	See previous slide.
Memory Access Service (MAS)	Implemented	See dedicated slide.
Test Service (TS)	Implemented, not used in validated binary	Implemented and tested for all communication links but not included in validated binary.
Device Discovery Service (DDS)	Implemented, not used in validated binary	It indicates rechability of devices. Rather incomplete as building block for an “extended DES”. Implementation not obvious for devices behind an RTU. Implemented yet not used in the validated binary for the reference mission (impact on communication traffic).
Synchronisation Service (SS)	Not implemented	

Application of Model-Driven Engineering to SOIS

- The SOIS implementation was realized by using a (partial) MDE process
  - Model for static service architecture definition, layering and assembly
  - Configuration of relationship between services and redundancy / cross-strapping
  - (Partial) code generation from a DSL



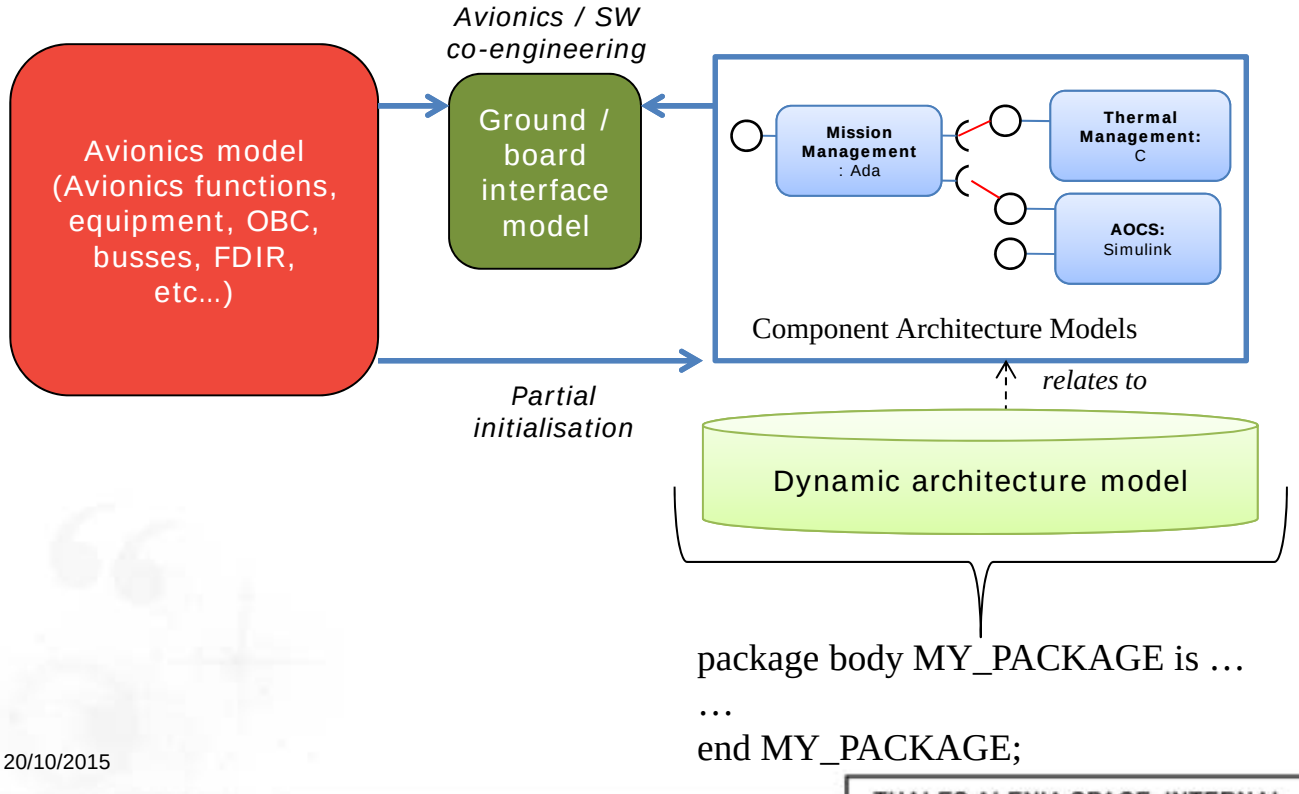
Practical results of the GSTP and TAS assessment (I)

- SOIS *as it is currently defined*, does not provide significant advantages to our current communication architecture
  - The non-recurrent cost of modification is too high compared to the observed advantages
  - Value added found in SOIS architectural layering and separation of concerns rather than in use of SOIS primitives as specified

- In our opinion, at the current state, SOIS tries to solve the OBSW development problems we had 10 years ago, with the means we had 10 years ago
  - In the meantime avionics architectures have evolved, and new breakthroughs are imminent
    - e.g. systematic use of rather complex RTUs; high-throughput, predictable / guaranteed / deterministic network communications (e.g., Spacewire-D, SpaceFibre, AFDX, TTEthernet)

Practical results of the GSTP and TAS assessment (II)

- In our opinion, at the current state, SOIS tries to solve the OBSW development problems we had 10 years ago, with the means we had 10 years ago (cont'd)
  - Development practices for OBSW in TAS operational projects have radically changed in the meantime



1. Operational use of ***full-fledged*** component models
2. Models for OBSW architecture and component architecture exchange
3. Systematic OBSW code generation
4. Increased opportunities for effective avionics / SW co-engineering

In our context, SW-SW interface problems are less and less relevant

Practical results of the GSTP and TAS assessment (III)

- The SOIS static architecture has some interesting ideas on layering and separation of concerns that may help improve our reference architecture
  - In particular w.r.t. to the DVS / DAS / DDPS / SL layering
- There is a lack of foundations for dynamic architecture or well-formed input for its definition and unclear allocation of non-functional aspects (e.g., buffering, part of the QoS) to services
- A lot of focus is placed on service primitives which sometimes are too generic to be used in a properly designed and modular OBSW architecture
  - E.g., they seem to be designed to minimize the maintenance effort of the standard rather than promote modularity and minimize the complexity of SOIS stack implementation

COMMAND_DEVICE.request (
Transaction Identifier,
Virtual Device Identifier,
Value Identifier,
Value)

1. Fetch the target of the command
2. Re-interpret conditionally the command according to the command target
3. Re-interpret the “generic type” of the value conditionally according to the command target and actual command

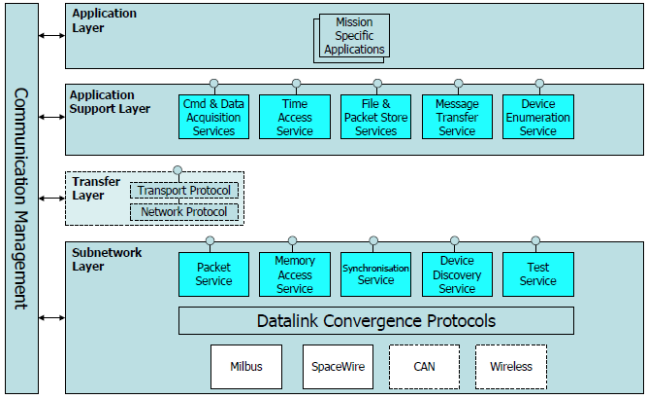
WE LOOK AFTER THE EARTH BEAT

ADCSS 2015
SOIS Session

Part 2 – Conclusions and Recommendations

Conclusions and recommendations (I)

A static service architecture



OSI-level service primitives definition

READ.request (
MASAP Address,
Destination Address,
Transaction ID,
Memory ID,
Start Memory Address,
Size,
Priority,
Channel,
Authorisation (optional))

READ.indication (
MASAP Address,
Destination Address,
Transaction ID,
Memory ID,
Start Memory Address,
Size,
Priority,
Channel,
Data,
Result Metadata)

Service configuration
(often non normative)

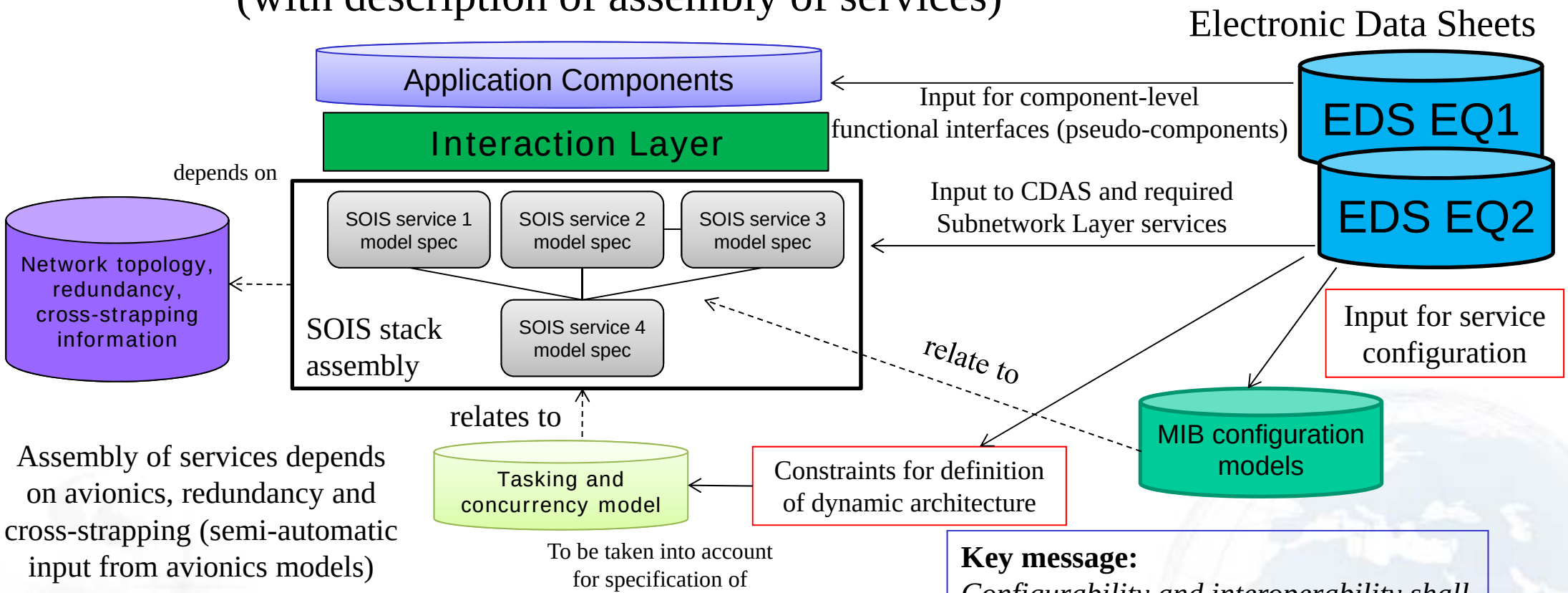
MIB

- Our proposal: re-center the effort for SOIS on
- 1) architecture definition;
- 2) compatibility of the approach with modern OBSW development paradigms (model-driven, component-oriented);
- 3) support of current and future avionics, with associated design process

Conclusions and recommendations (II)

- SOIS shall not be defined in isolation. Think about the bigger picture!

A static *model-based* service architecture for use with the OSRA (with description of assembly of services)



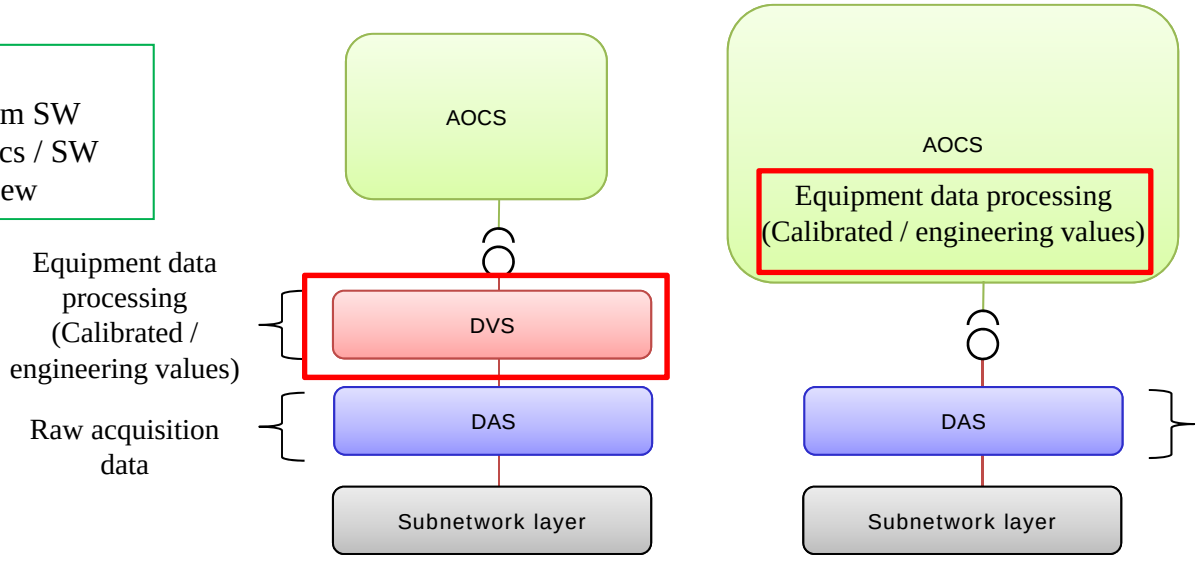
Key message:
Configurability and interoperability shall come from models, not from APIs

Conclusions and recommendations (III)

- SOIS is not only about SW development
  - It overlaps concerns for which the main decisional stakeholder is not the OBSW team

Illustrative example

Design 1:
most logical choice from SW architecture and avionics / SW product line point of view



Design 2:
Algorithms with knowledge of acquisition from a *given physical device*

Favours centralization of equipment data processing at the AOCS (mostly for early AOCS validation)

Whatever the choice, it will have a non-negligible impact on AOCS specification, AOCS simulation architecture, AOCS life-cycle

Conclusions and recommendations (IV)

- Envisaged evolutions of SOIS shall be driven by a WG
  - Similar to SAVOIR-FAIRE, but with a mandate also on EDS, equipment functional interfaces and relevant avionics aspects (e.g., communication protocols)