TERMA
ALLIES IN INNOVATION

OSRAD

ONBOARD SOFTWARE REFERENCE ARCHITECTURE DEMONSTRATOR

DECEMBER 2021

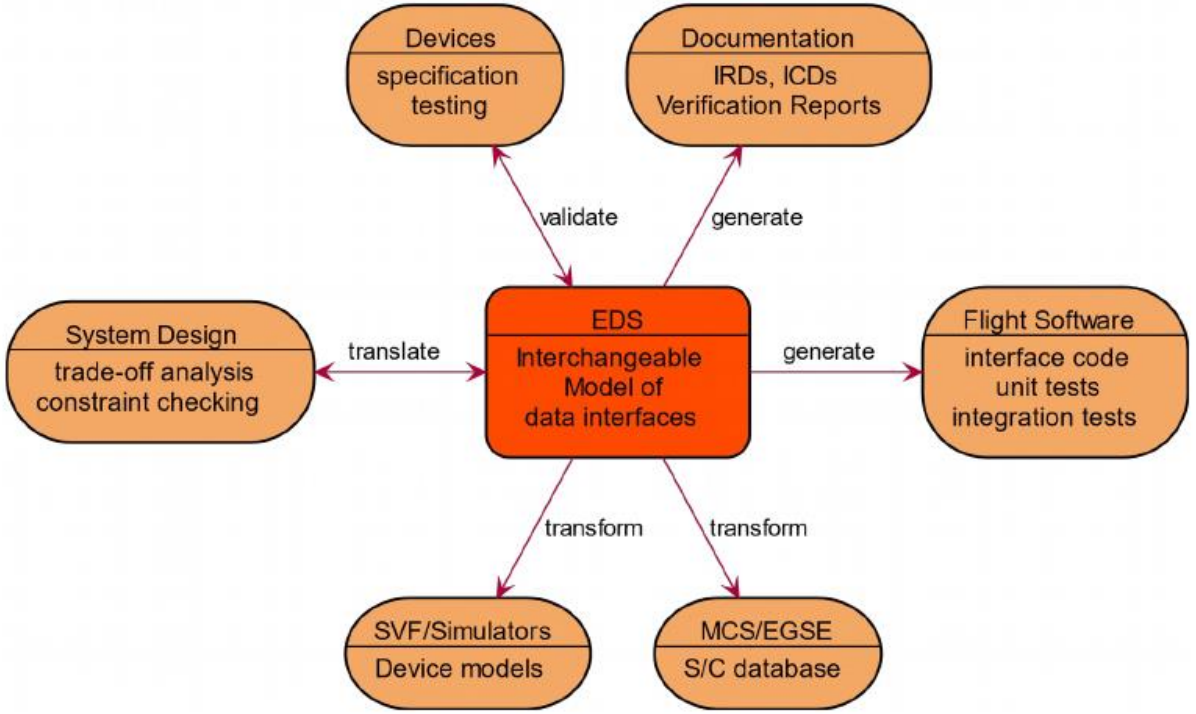
Background

The problem:

- Maintain Interface Control Document
- Consistency of User Manual

The solution:

- One source of information
- Well-defined syntax and semantics
- Machine readable
- Auto processing

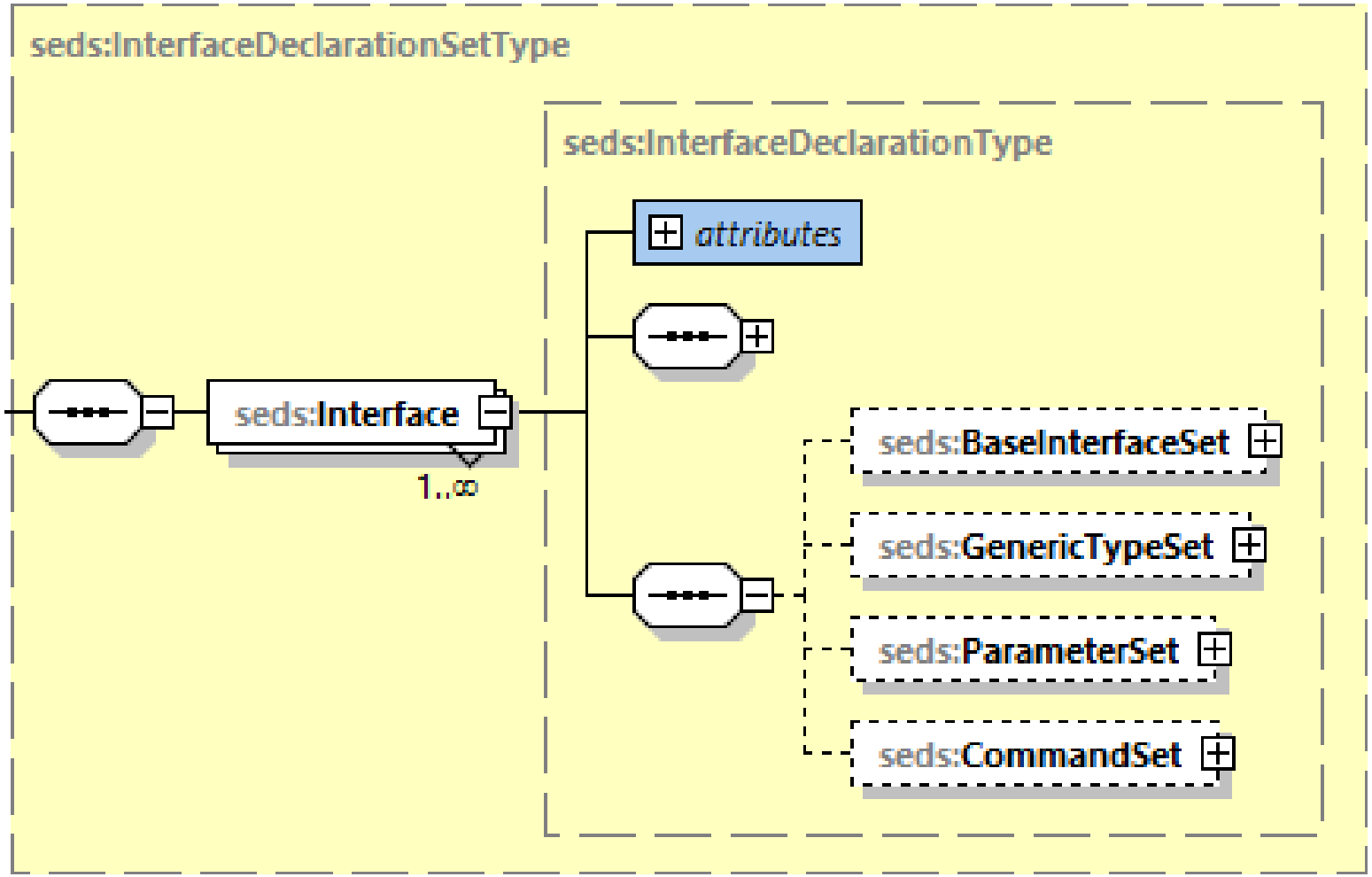


Background/EDS

SOIS Electronic Data Sheets is a concept that has been proposed to allow to capture of the relevant information about a piece of equipment.

SEDS is defined in CCSDS standards.

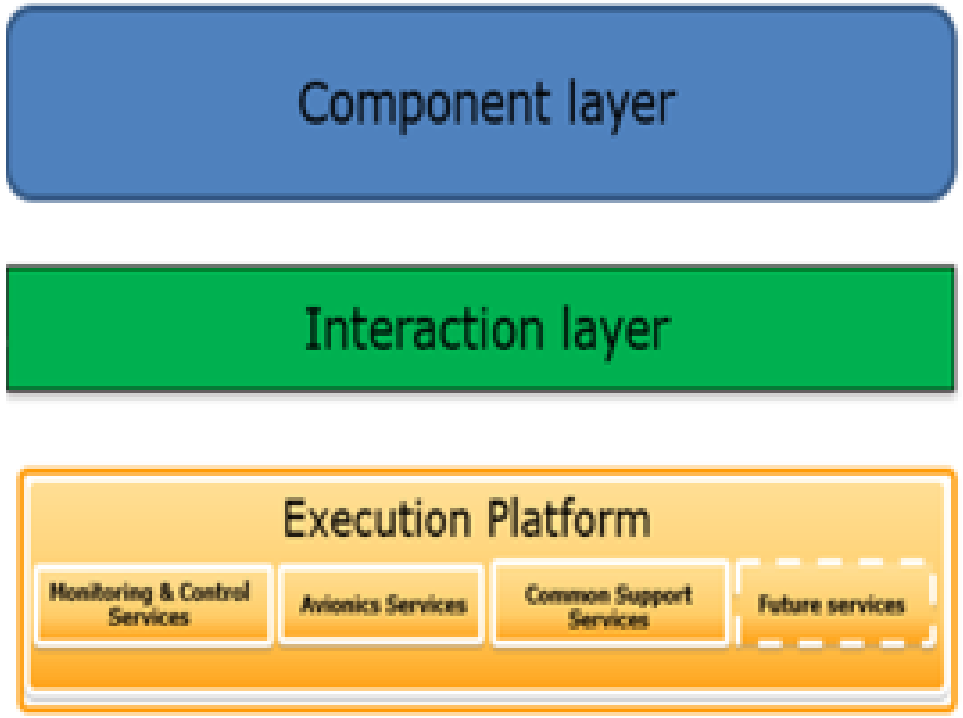
It is an xml format, delivered with a schema.



Project Objectives

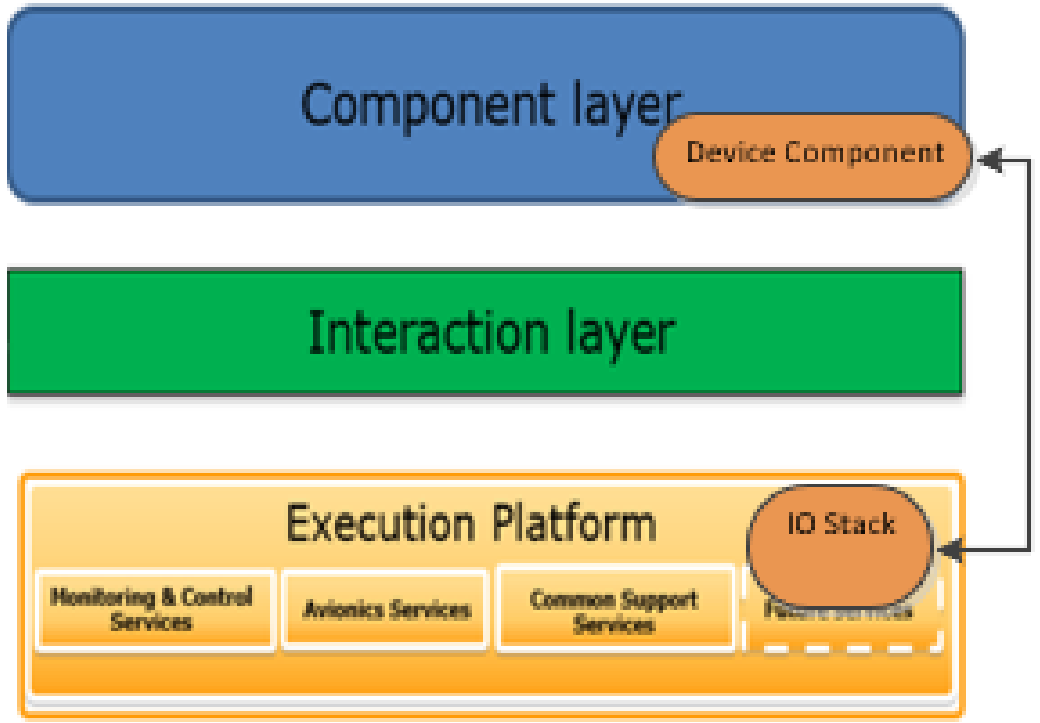
- *EDS Integration into OSRA toolchain*
  Exemplified by the Terma Star Tracker.
  The process of mapping EDS to OSRA toolchain shall be described, and missing capacities in the EDS specification shall be identified.
- *EDS Integration into Execution Platform*
  Exemplified by the Terma Star Tracker and corresponding IO stacks.
  Investigate if an EDS specification is sufficient for automatically generation of executable code.
- *Design and implement a representative application that demonstrates the inclusion of the EDS based component and examines the OSRA development process.*

Background/Onboard Software Reference Architecture (OSRA)



- Application software is designed by specifying "software components". Non-functional aspects are only "declaratively specified". Their implementation is left to the automation capabilities of the design environment
- Automatically generated layer that implements the non-functional concerns declared by components (e.g., tasking, synchronization)
- Provides services to the component layer (e.g., context saving) and interaction layer (e.g., tasking primitives, access to devices, message transfer)

Onboard Software Reference Architecture (OSRA)

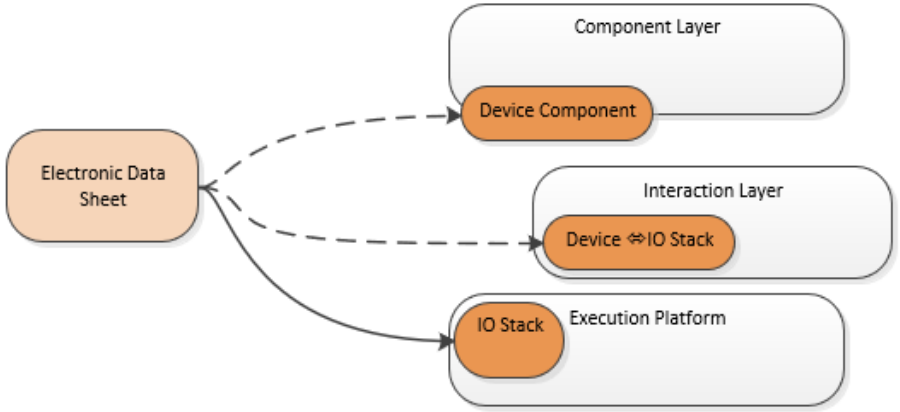


Device Component:

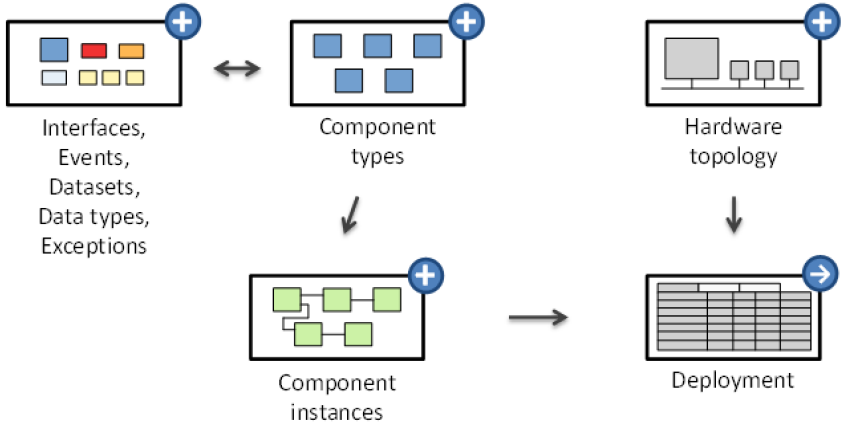
A *pseudo* component, providing the services of a *device*

Project Objectives

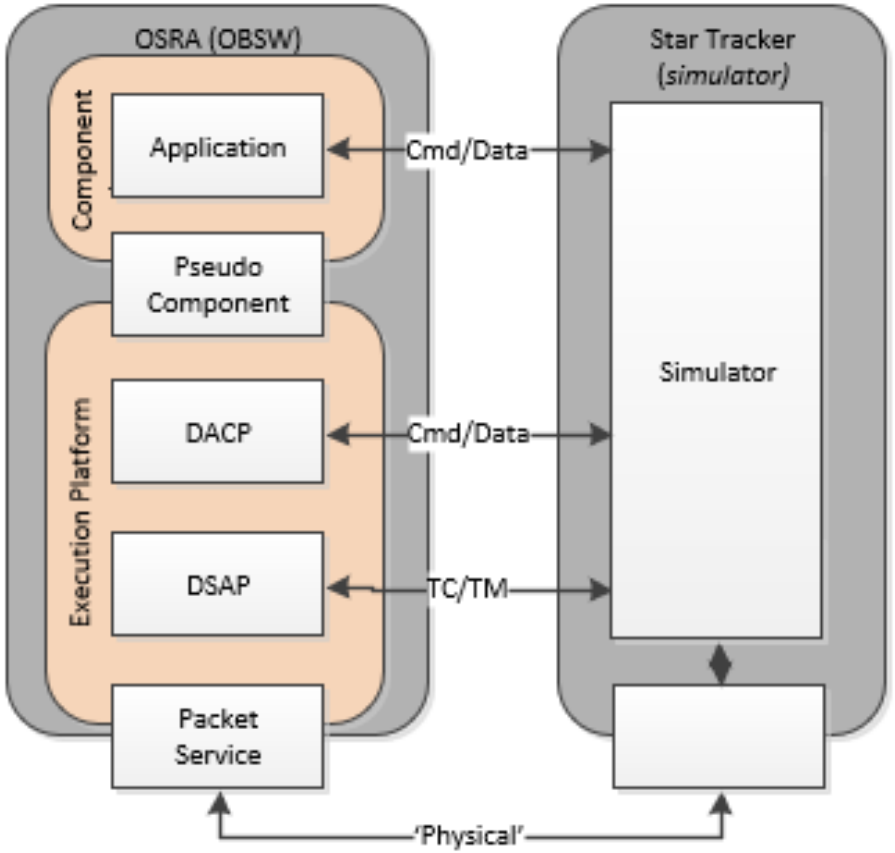
- Integration of EDS



- Application Development



Integration of EDS/IO Stack

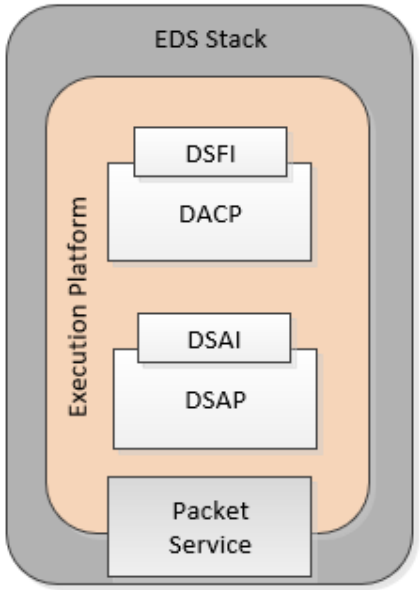


DACP: Device Abstraction Communication Protocol

DSFI: Device Specific Functional Interface

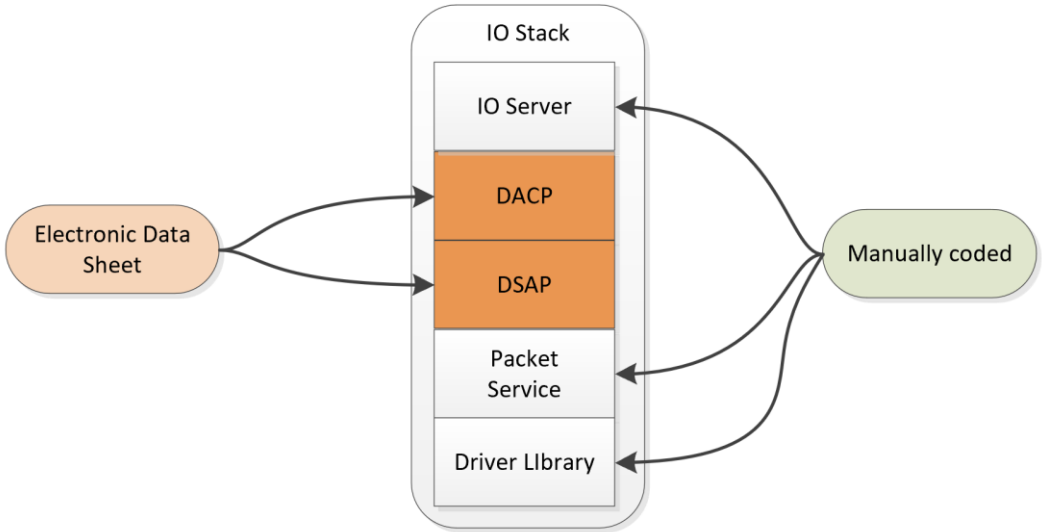
DSAP: Device Specific Access Protocol

DSAI: Device Specific Access Interface



Integration of EDS/IO Stack

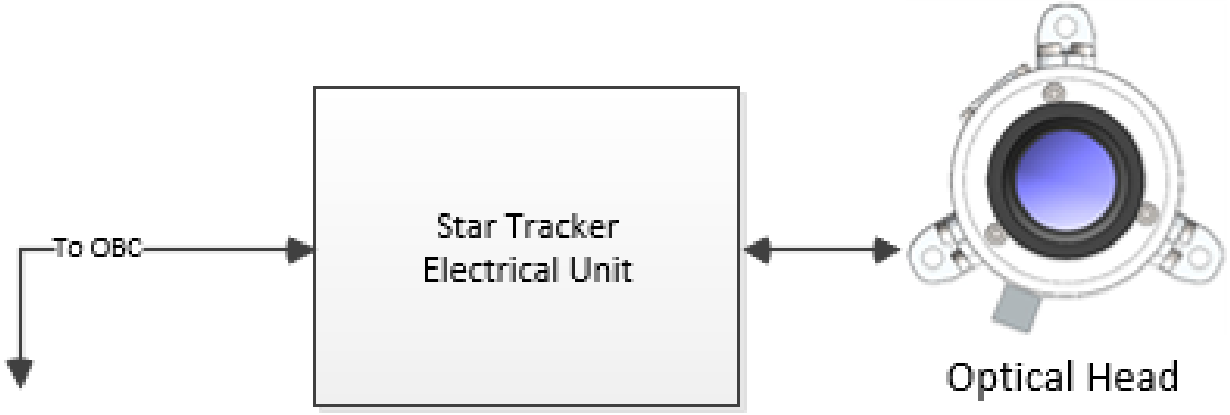
EDS:
Specifies the characteristics of the device (environment not taken into account)



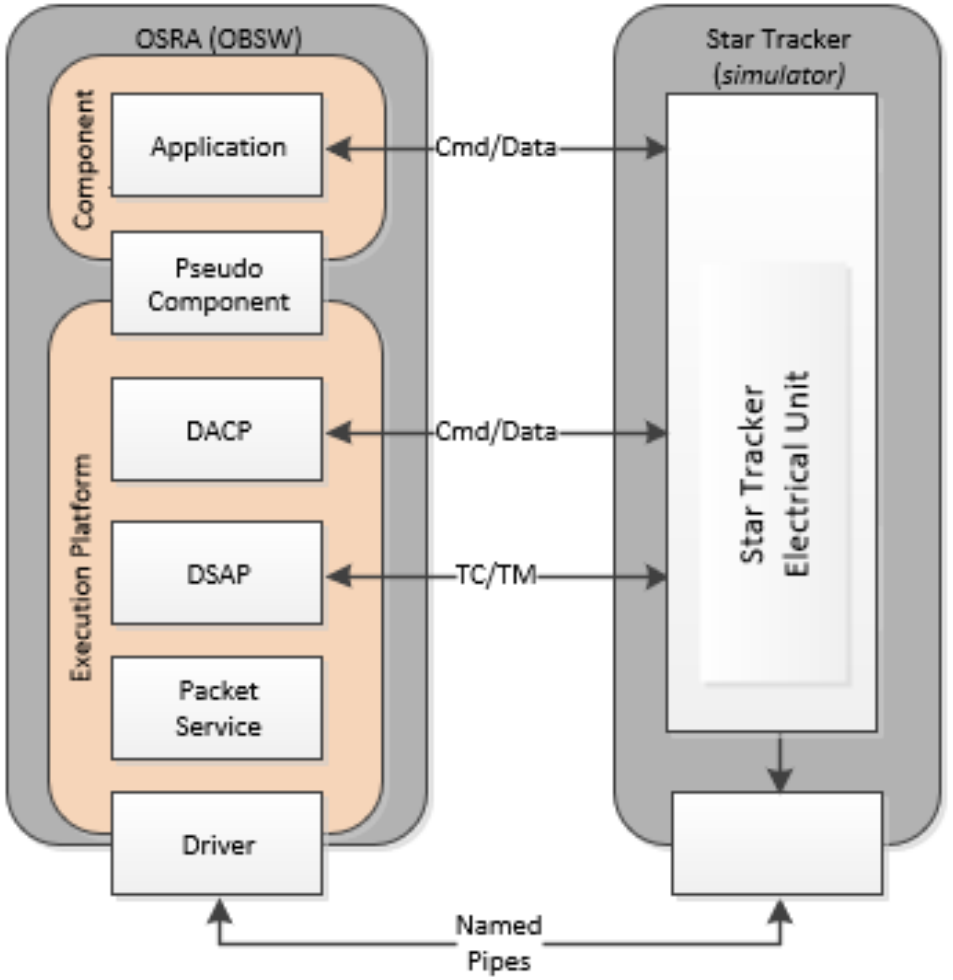
EDS Findings

- The EDS is difficult to read and very laborious to write
- It is not possible to define reactions to restriction violation
- The sematic is not defined (standard specifies the syntax)
- Validation of an EDS. It is straight forward to validate the syntax towards the provided schema. Semantic validation is much more complicated
- It is not possible to define endianness on interface level
- The EDS is sequential and does thus not have means for protection against data overriding

Star Tracker Simulator



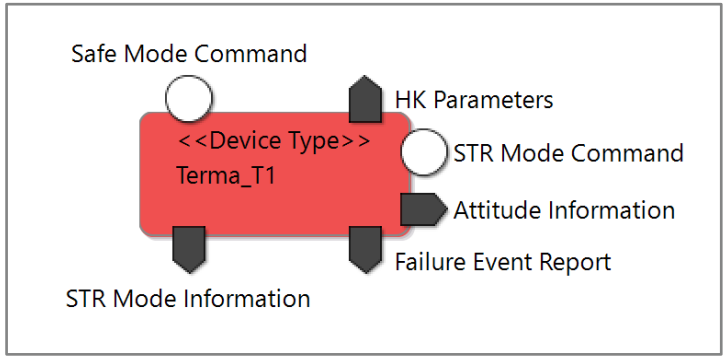
PUS Service	
Service type 1	Telecommand Verification
Service type 3	House Keeping
Service type 5	Event Reporting
Service type 6	Memory Management
Service type 9	Time Management
Service type 20	Parameter Management
Service type 128	Mode Management
Service type 129	Attitude and Image Management



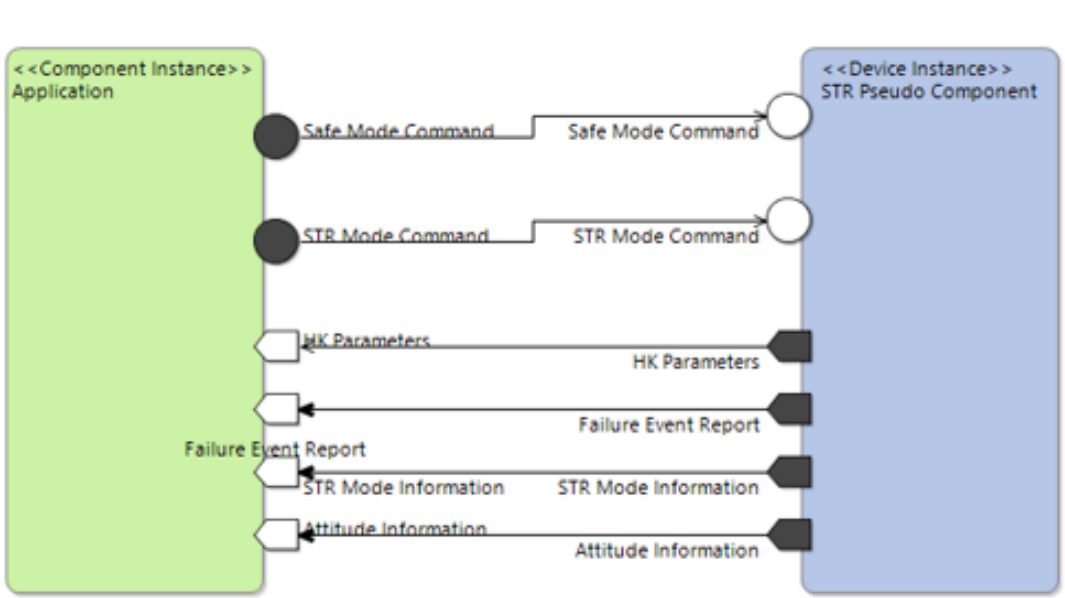
Integration of EDS/Device Component

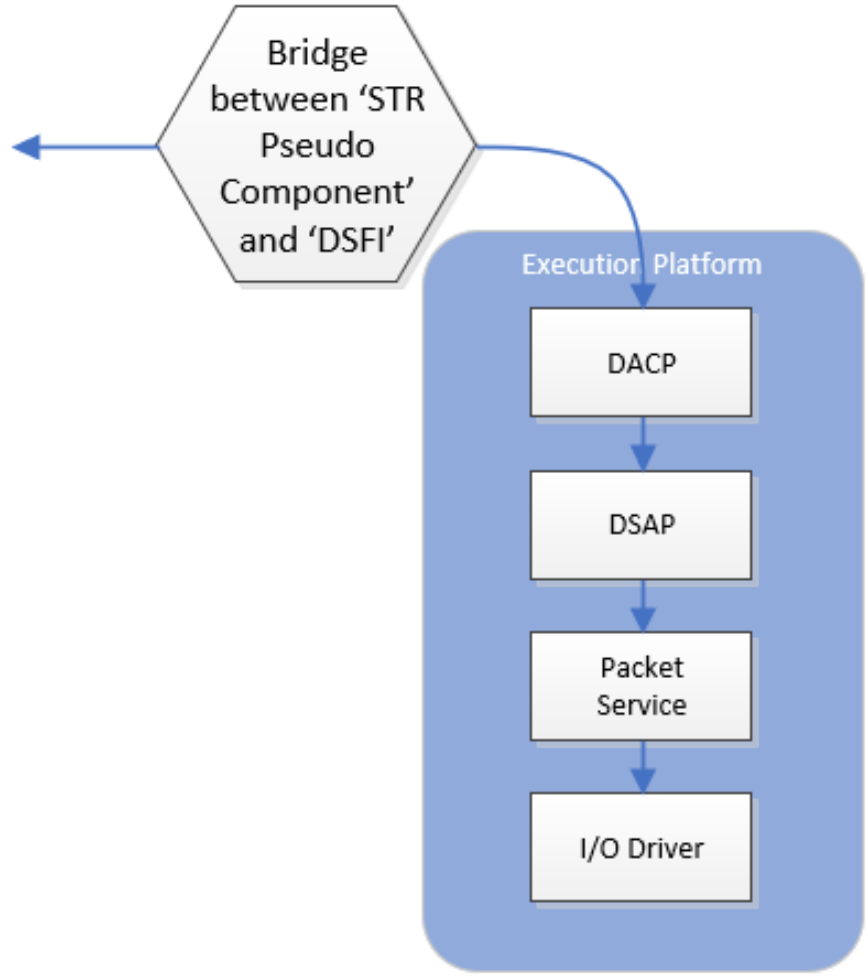
```
<Interface   abstract="true"-- shall always be ‘true’, checked
             level="functional"      -- shall always be ‘functional’, checked
             name="DSFI"            -- used directly, duplicates not allowed
             shortDescription="DSFI: device-specific functional interface">

<Parameter mode="async"                -- “async” or “sync”
             name="str_oh_attitude_parameters"
             readOnly="true"           -- “true”/“false”
             type="ROUTINE_ATTITUDE_PARAMETERS"        -- defined in the DataTypeSet
             shortDescription="Routine attitude parameters"/>
   <Command     mode="async"
                name="EnterStandbyMode">
                <Argument      mode="in"
                        name="OhId"
                        type="CCSDS/SOIS/SEDS/UINT32"/>
   </Command>
```



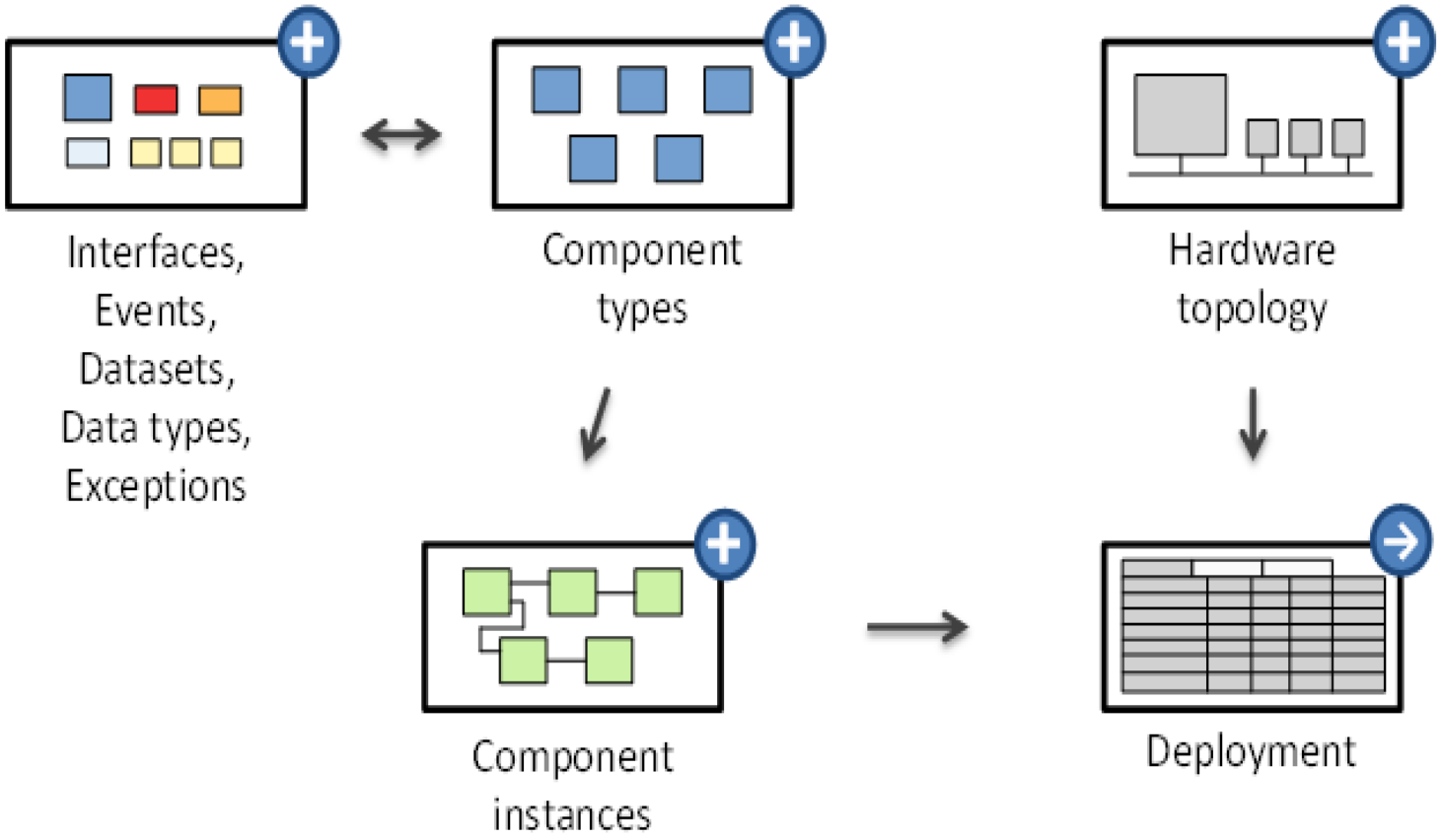
Integration of EDS/Interaction Layer



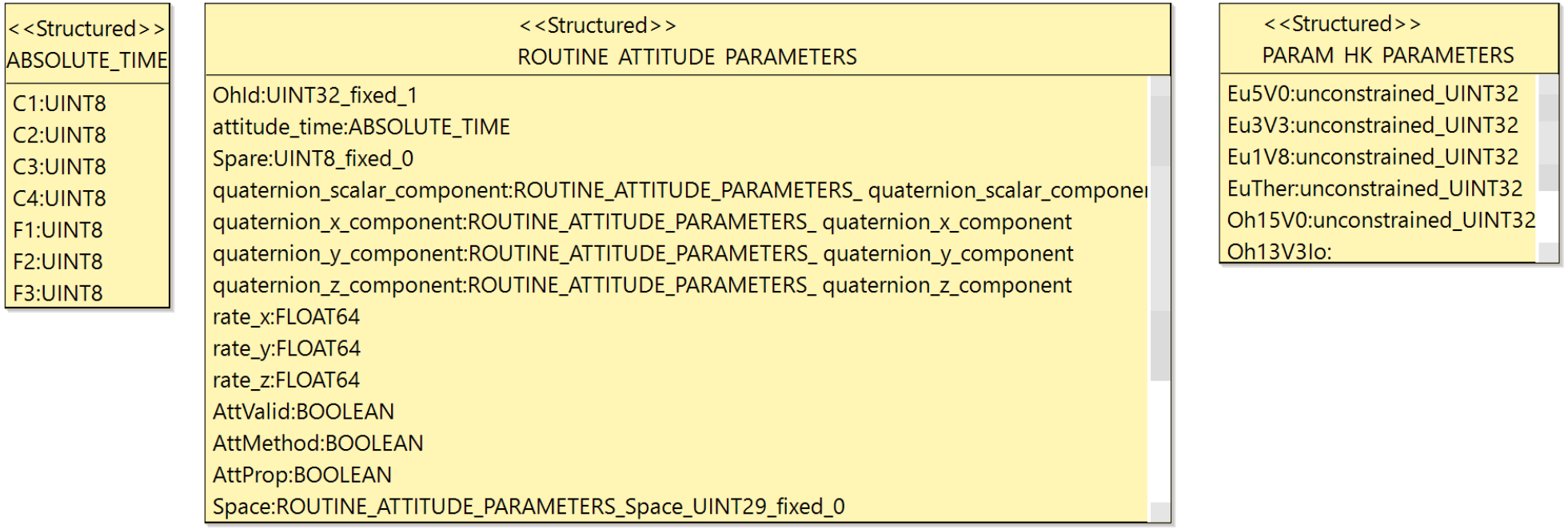


- Map outgoing functions to function calls
- Map incoming data to data set emitting

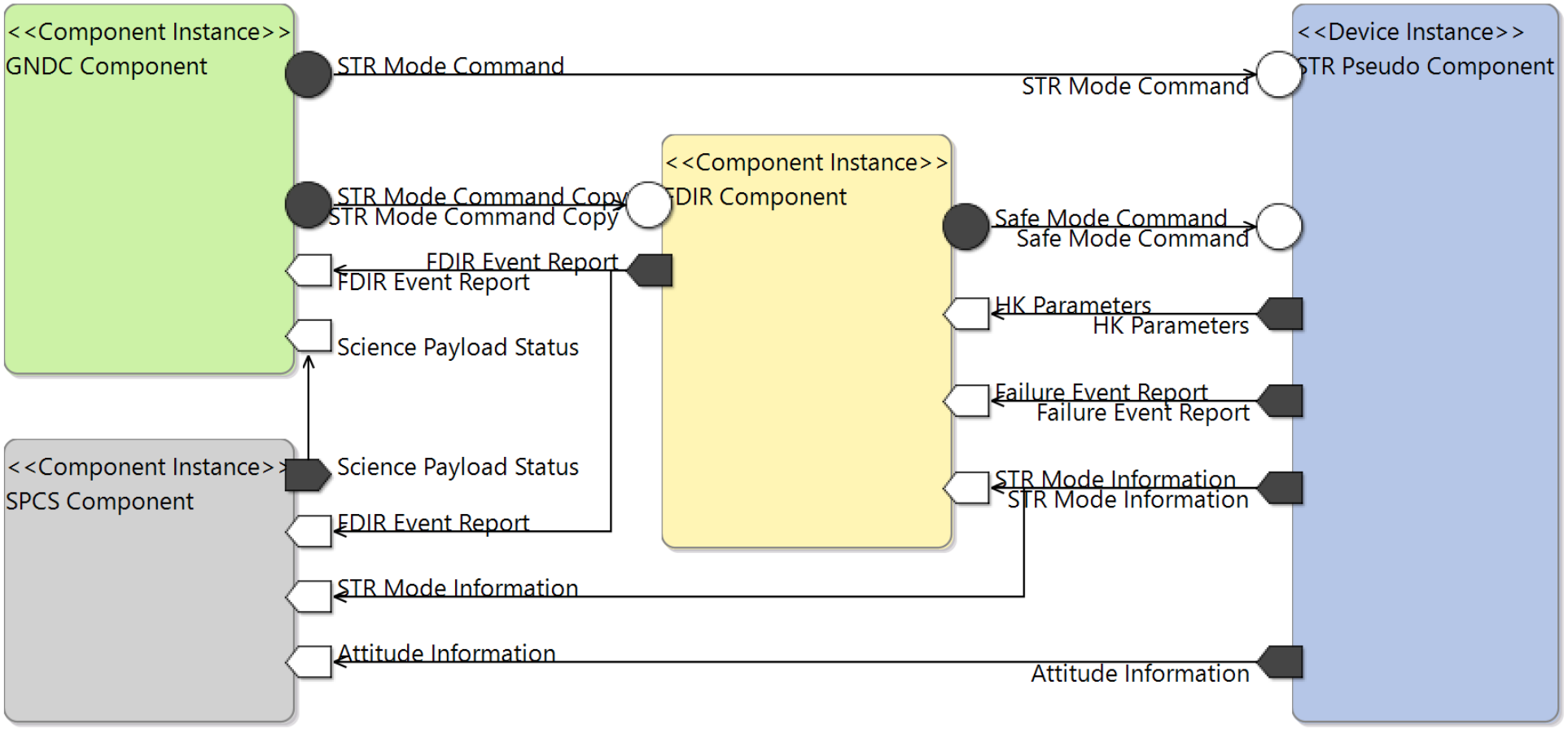
Application Development



Application Development/Types etc



Application Development/Component Instances



OSRA Findings

- Static registration as data set receiver
- Relations between provided and required operations are strictly one-to-one
- During system and software requirement specification, definition of corresponding logical models in the SCM editor allowed for reuse/exchange of the models through the requirements phase to the component design phase.

Summary/Recommendation

- Be very specific with the objective of the actual EDS. This means that it must be clear if the EDS shall define packing/unpacking of messages, if it shall be used to specify the behaviour of the device, or if it shall be used to define the ICD. The intention is of course that the EDS could be used for both. However, this requires tools for extracting and isolating the relevant parts of the EDS.
- Authoring tools for writing and validation of EDS specifications should be made available
- Compilers for transferring EDS to source code should be made available.
- Specification of the semantics of the EDS definition is needed