SAVOIR
space avionics open interface architecture

Software aspects of the reference architecture

A. Jung / J.L. Terraillon
European Space Agency ESTEC

ADCSS 2012 – 23/10/2012

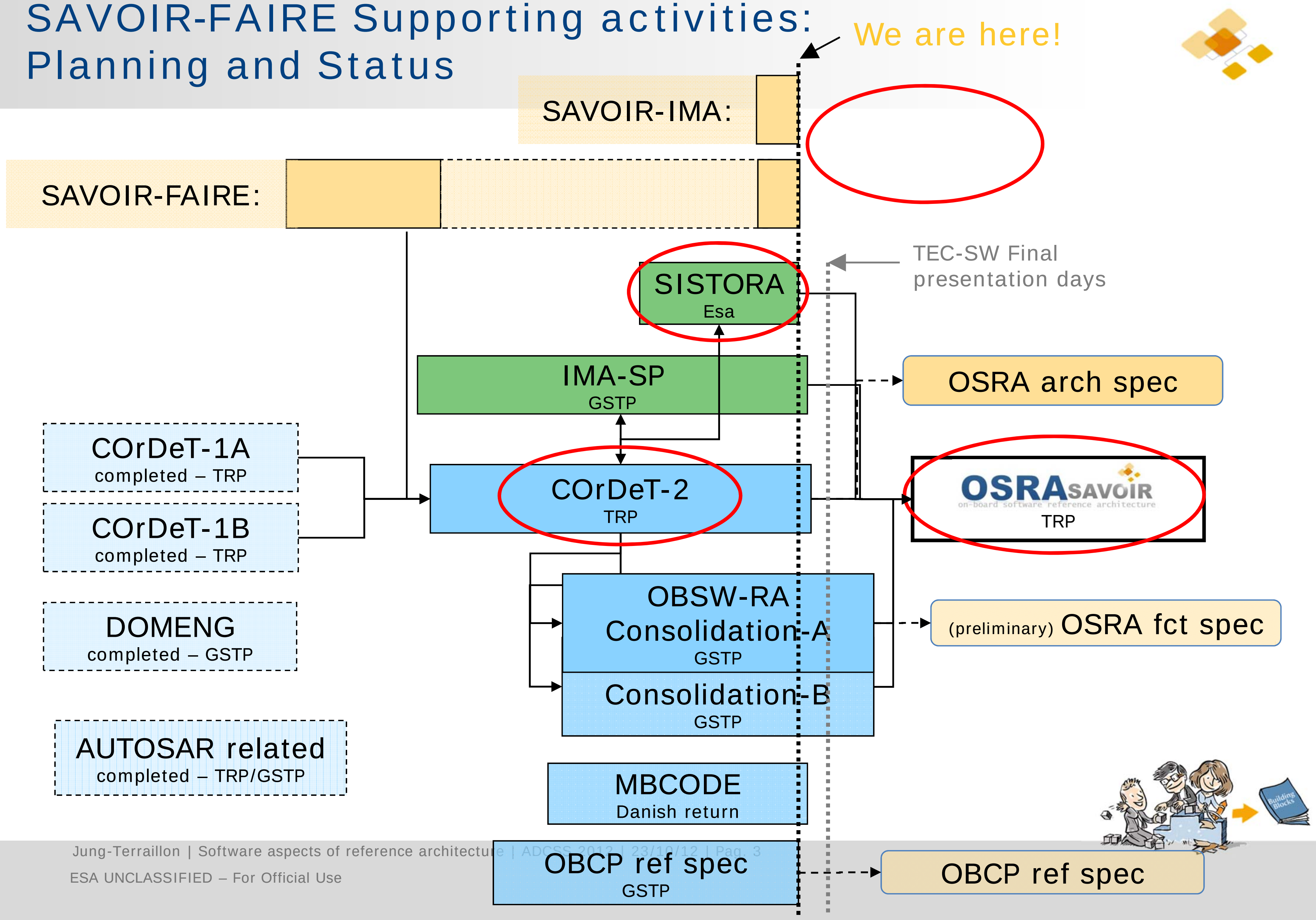
SAVOIR FAIRE
space avionics open interface architecture

Outline

- **SAVOIR-FAIRE and OBSW reference architecture**
  - Status and Schedule of activities
  - Recapitulation of OBSW-RA approach and component model implementation in COrDeT-2

- **SAVOIR-FAIRE discussions: Component model “specification”**
  - Several scenarios under discussion

SAVOIR-FAIRE Supporting activities: Planning and Status



COrDeT-2: Status

- Final OBSW-RA specification to be finalized in Nov 2012 and **Final presentation in Dec 2012** (TEC-SW final presentation days)

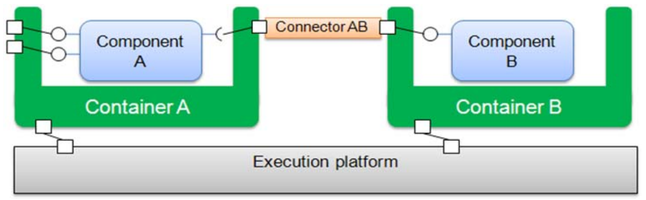
- **Focus on:** Overall architecture, component model and Execution platform services definition
- **Out of focus:** (internal) details of the Execution platform, Time and Space Partitioning (TSP) (covered in IMA-SP and SISTORA)

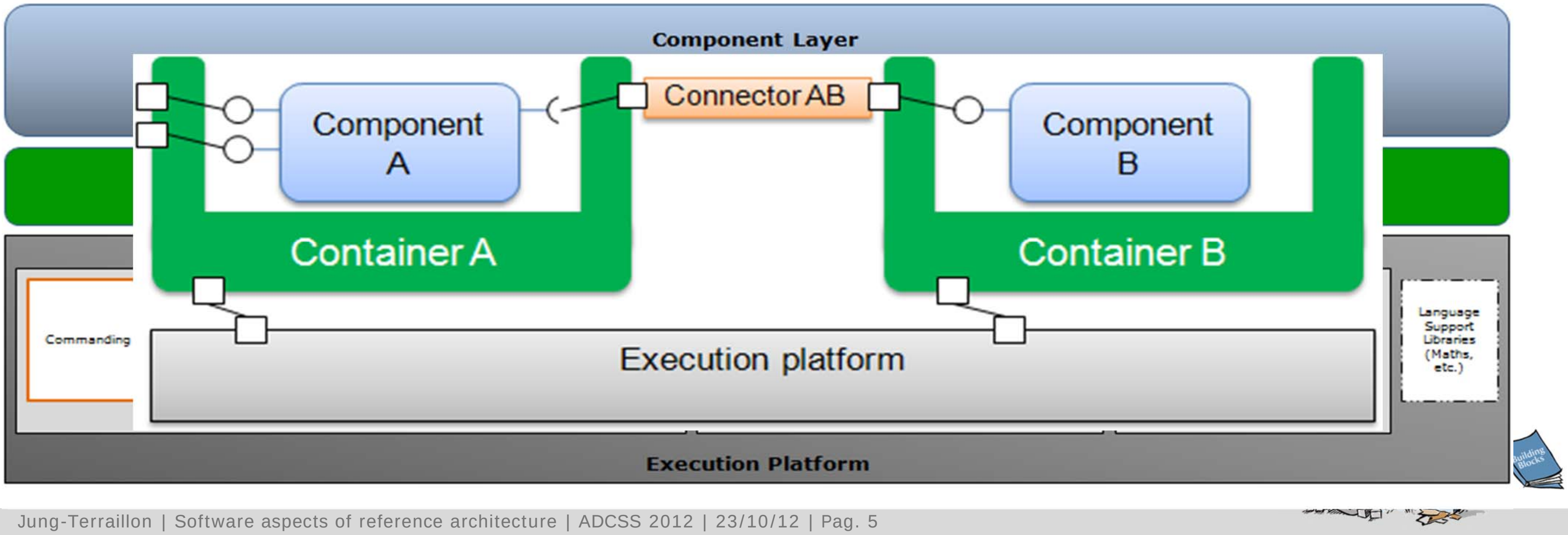
- **Results**: overall description of the OBSW-RA, prototype implementation of the OBSW-RA component model (metamodel + graphical editor), prototype implementation of back-end toolchain (code generators), realization of a use case.
- **Open points and future work identified:** a large number of details, but overall reference architecture baselined and agreed

COrDeT-2: High level architecture

Important ingredients and high-level view of the OBSW reference architecture:

- Component based approach
- Model driven engineering
- Separation of concerns
- ...





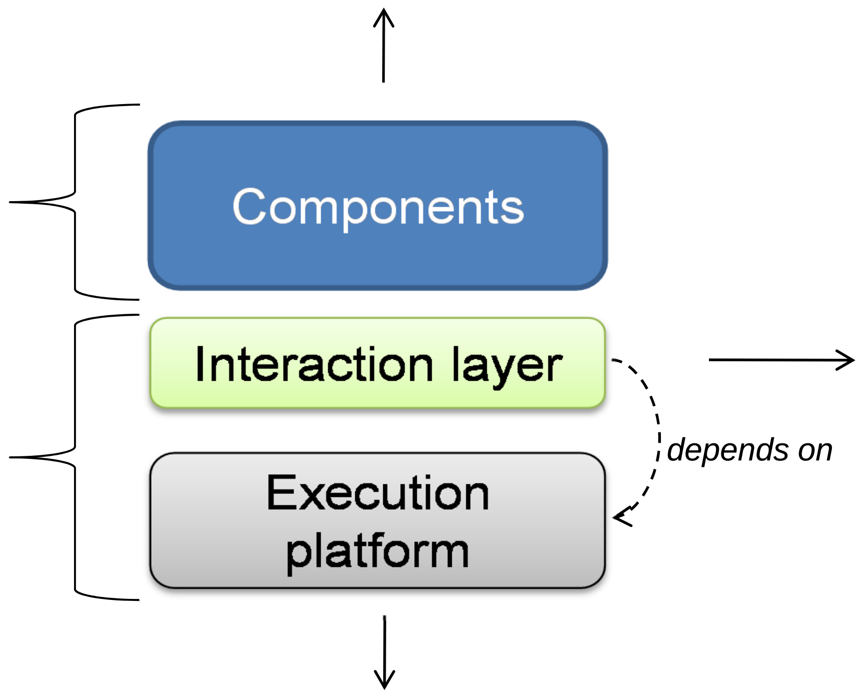
COrDeT-2: High level architecture – Goals and dependencies

Application level
- realization of functional requirements by an assembly of components

Independent of the chosen computational model

Independent [*] of the execution platform

Implementation dependent on the chosen computational model



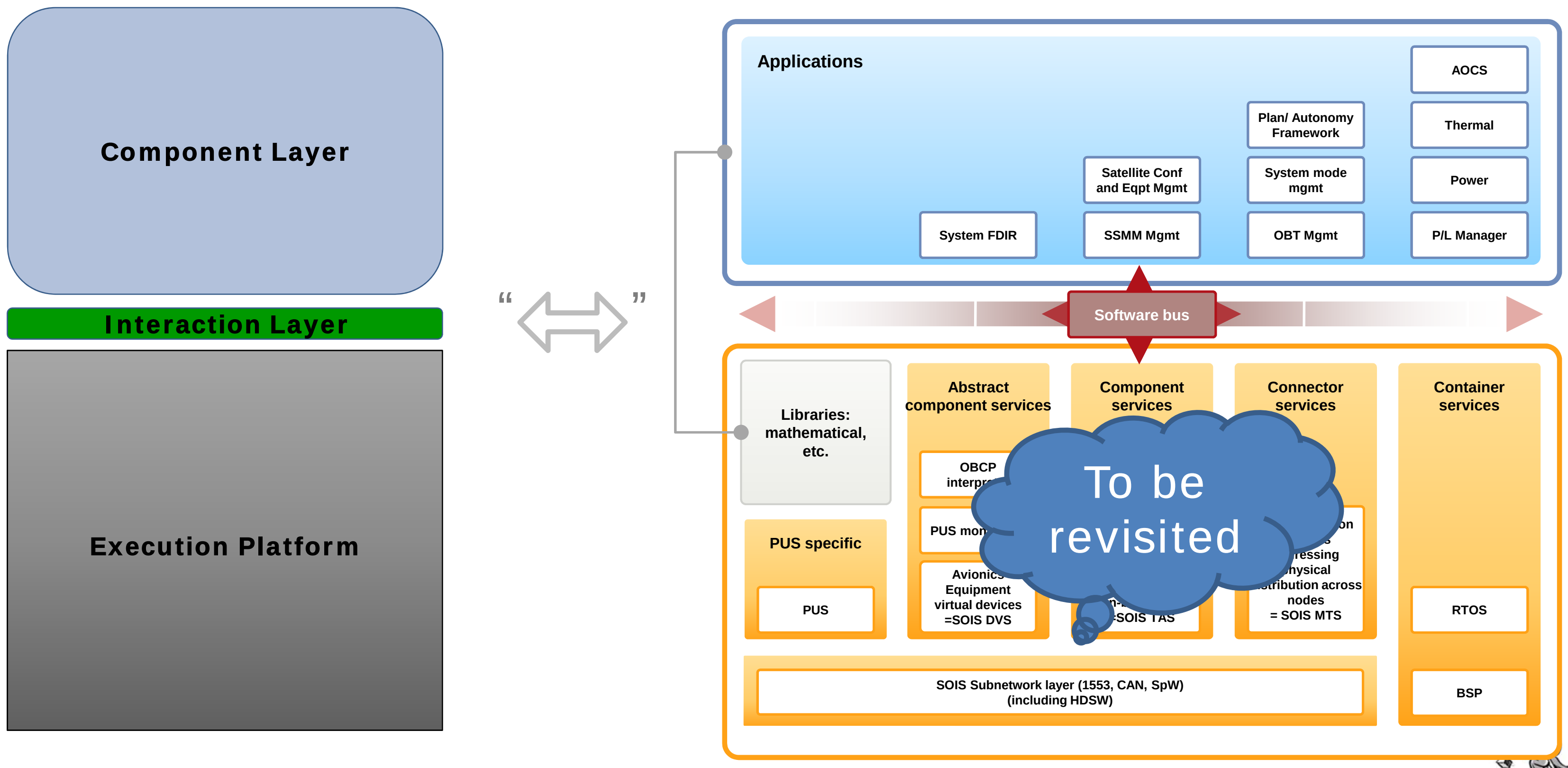
Containers and connectors
- realization of non-functional properties + communication code

Services for execution support
[communication services, tasking, synchronization, libraries]
- warrants non-functional properties

[*] with the exception of numerical precision

COrDeT-2: Evolution of SAVOIR-FAIRE architecture

COrDeT-2 High-Level Architecture and **SAVOIR-FAIRE architecture**:



Component model specification – Context

Component model

- It’s the **cornerstone** of the approach
  - All stakeholders will interact in the development through the component model and its supporting tools

- The component model ***description***
  - have been available in various stages of refinement for nearly 2 years (SAVOIR-FAIRE, Marco Panunzio’s PhD thesis, COrDeT-2 refinements)
  - however ***no formal specification*** of the component model
    → ***difficult***

How can we then effective finalize and industrialize the component model?

Component model specification – Possible scenarios

Scenario 1

A common "design language" is mandated / elected for the component model (e.g. UML, DSL)

↓

A companion metamodel is mandated

↙ ↘

Scenario 1.a

The same toolset is used by each user of the component model

Scenario 1.b

Different toolsets (based on the same metamodel) are used by each user of the component model

Scenario 2

Freedom to implement the component model with a design language of choice (e.g. UML, DSL)

↓

Different languages (i.e. metamodels) are used by different users of the component model

↓

Scenario 2

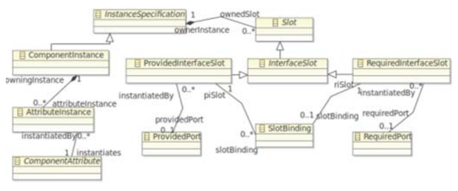
Different toolsets are used by different users of the component model

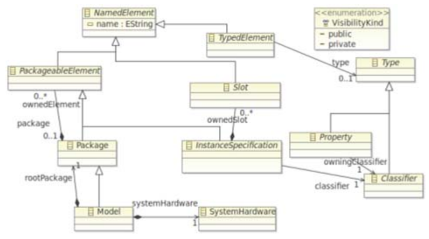
There are several pros and cons for each scenario.

SAVOIR-FAIRE discussed the scenarios and position papers were written by the members. A possible way forward will be presented in the next slides.

Way forward – Scenario 1.b-2 – Specification

- **Specification** of component model:
  - ***Syntax:*** Use a domain specific language (i.e. create a meta-model based on ecore), defining all entities of the component model
    - advantage of DSL is that it allows us to describe exactly what we need without being constrained by any pre-existent language choice / limitation

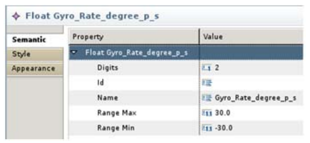




  - ***Sematics:*** English text explain, what each element of the meta-model means

Way forward – Scenario 1.b-2 – Use

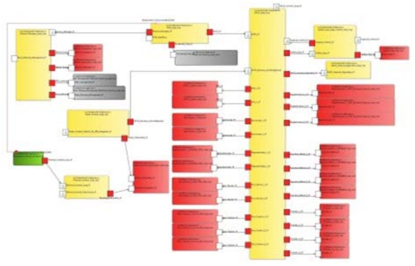
- **Use** of the component model:
  - i. **Choice #1 (DSL):**
    Meta-model: *DSL* used for specification of component model
    Tool: build a tool based on DSL, e.g., using *Obeo Designer* or *Graphiti (open source)*

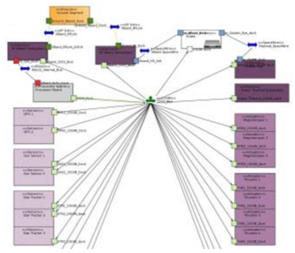


Float Gyro_Rate_degree_p_s

Semantic | Style | Appearance

Property	Value
Float Gyro_Rate_degree_p_s	
Digits	2
Id	
Name	Gyro_Rate_degree_p_s
Range Max	30.0
Range Min	-30.0





  - ii. **Choice #2 (e.g. UML):**
    Meta-model: (i) *e.g. UML+MARTE profile*
    (ii) *Bi-directional model transformation to the DSL*
    Tool: *e.g., (open-source/commercial) UML tools* that can be customized and extended to support the user needs

→ *The choices and their implications have to be evaluated in SAVOIR-FAIRE*

Contact

Feedback: savoir@esa.int

Contact SAVOIR-FAIRE

Jean-Loup.Terraillon@esa.int

Andreas.Jung@esa.int