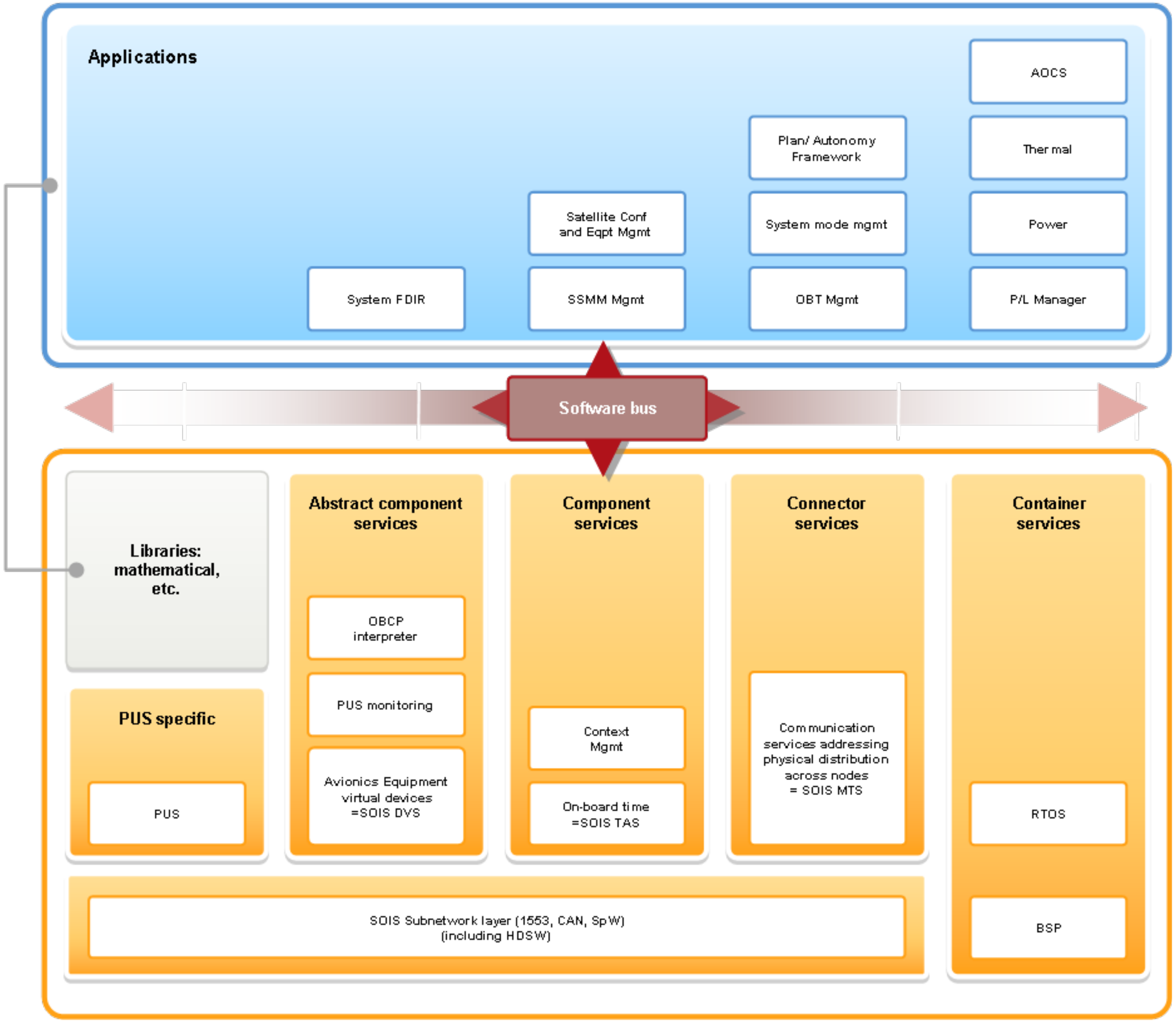
ESA software factory prototype based on TASTE

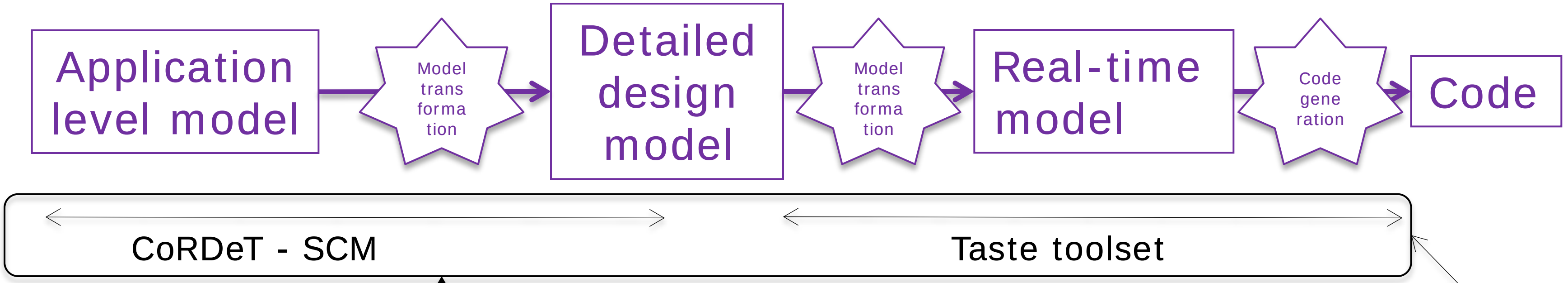
Maxime Perrotin, Andreas Jung, Jean-Loup Terraillon
with contributions of GMV in COrDeT-2 studies
...and Suzanne Guerreiro as Estec trainee

The software reference architecture


Applications
AOCS
Plan/ Autonomy Framework
Ther mal
Satellite Conf and Eqpt Mgmt
System mode mgmt
Power
System FDIR
SSMM Mgmt
OBT Mgmt
P/L Manager
Software bus
Libraries: mathematical, etc.
Abstract component services
Component services
Connector services
Container services
OBCP interpreter
PUS monitoring
Avionics Equipment virtual devices =SOIS DVS
PUS specific
PUS
Context Mgmt
On-board time =SOIS TAS
Com munication services addressing physical distribution across nodes = SOIS MTS
RTOS
SOIS Subnetwork layer (1553, CAN, SpW) (including HDSW)
BSP

Presentation of the Architectural steps

- One principle of OSRA is “separation of concerns”: application to subsystems engineers (AOCS), architecture to software architect, implementation to real-time software engineers (supported by tools)
- Therefore a toolset (“software factory”) takes an application level model and generates the code



Application level model

Model transformation

Detailed design model

Model transformation

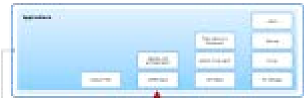
Real-time model

Code generation

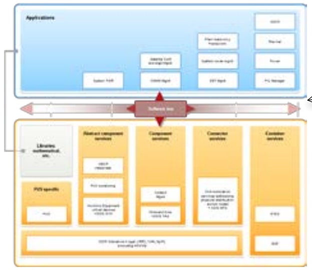
Code

CoRDeT - SCM

Taste toolset

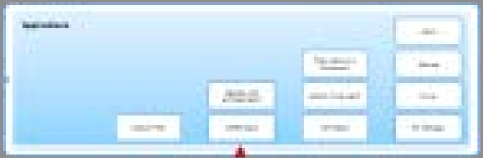


The knowledge of the architectural design (PUS and SOIS design patterns, libraries) is in the model transformation tool



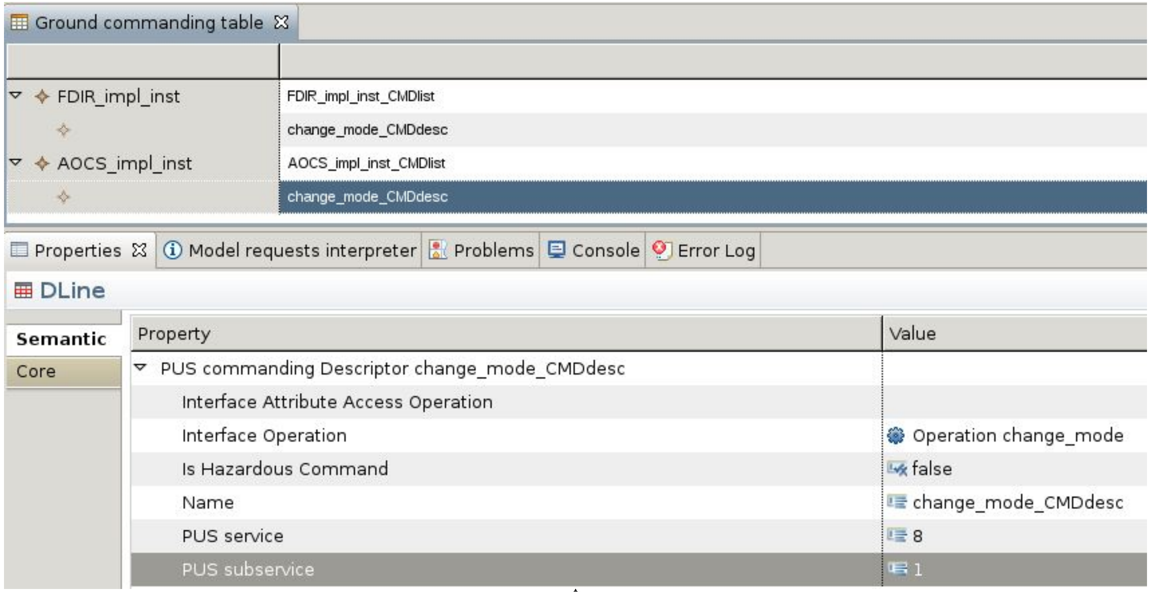
The software factory implements and generates the software bus

Application level model



Two application components, AOCS and FDIR, are placed by the Application engineer.
They are instance of a component type.

AOCS has 2 interface that needs to be reported through **PUS**, and 2 interface that need to be monitored by PUS. The green boxes are abstraction of the **PUS** design patterns.



Ground commanding table

FDIR_impl_inst	FDIR_impl_inst_CMDlist
	change_mode_CMDdesc
AOCS_impl_inst	AOCS_impl_inst_CMDlist
	change_mode_CMDdesc

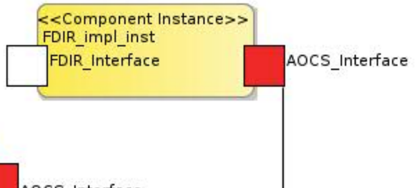
Properties | Model requests interpreter | Problems | Console | Error Log

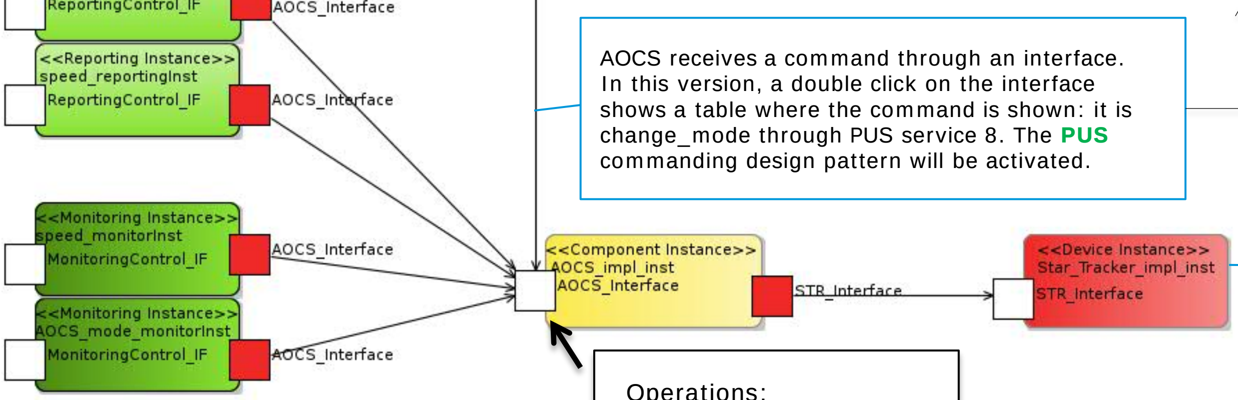
DLine

Semantic

Core

Property	Value
PUS commanding Descriptor change_mode_CMDdesc	
Interface Attribute Access Operation	
Interface Operation	Operation change_mode
Is Hazardous Command	false
Name	change_mode_CMDdesc
PUS service	8
PUS subservice	1

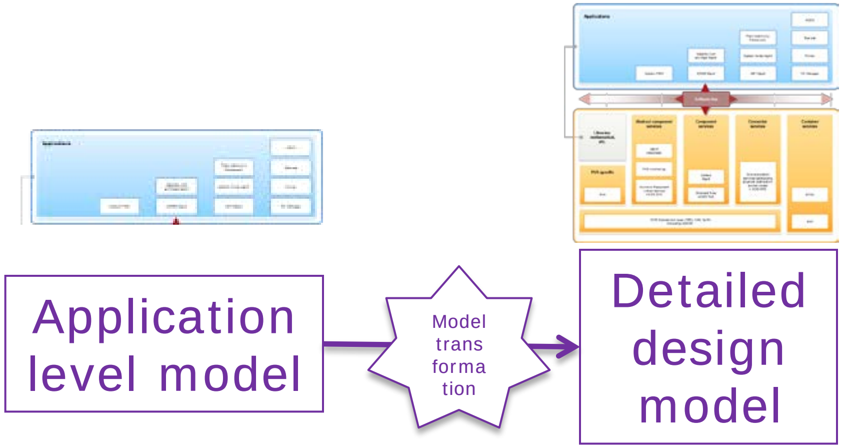




AOCS receives a command through an interface. In this version, a double click on the interface shows a table where the command is shown: it is change_mode through PUS service 8. The **PUS** commanding design pattern will be activated.

AOCS talks to a star tracker though a proxy that is an abstraction of the **SOIS** services

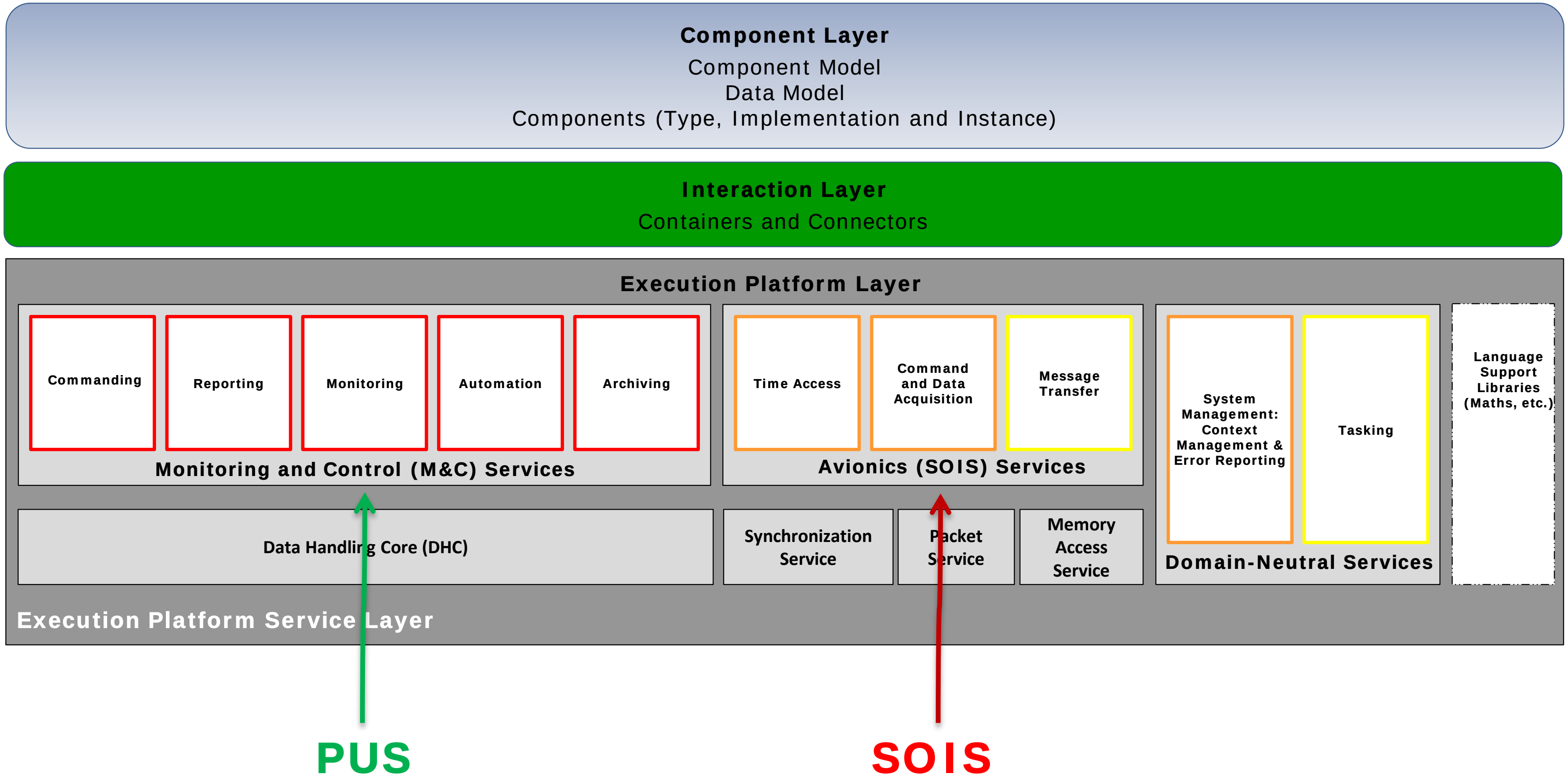
Operations:
setMode(parameter);
Attributes:
currentMode
currentPointAccuracy
…



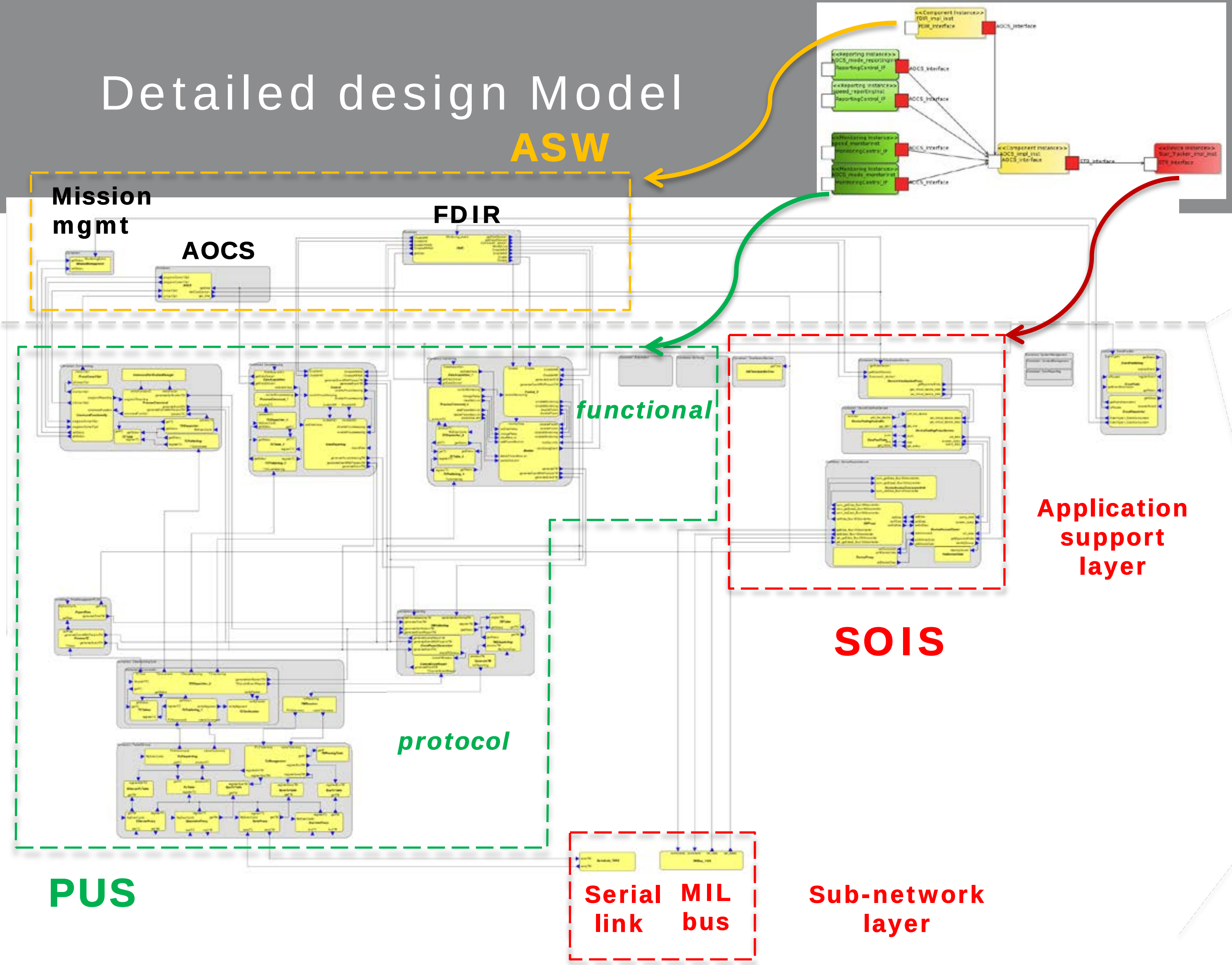
Now we model-transform from application level model to detailed design model

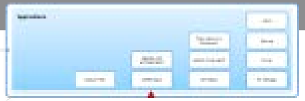
(design patterns are inserted)

The execution platform



Detailed design Model

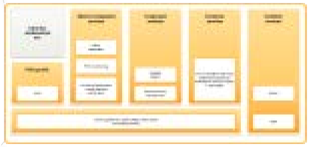




The model transformation tool has expanded the green and red boxes into design patterns

Each box will be detailed in the next slides

This diagram has more PUS services than the application level model one. This is just to give the complete view of the current implementation

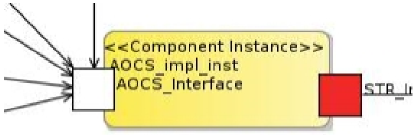


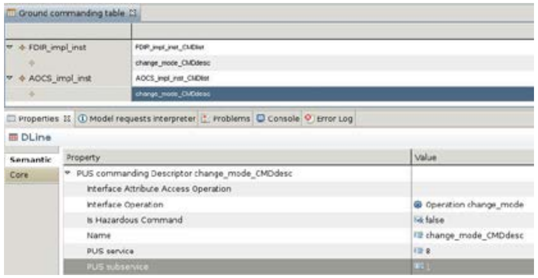
This is the "execution platform"

The SOIS implementation is incomplete… We miss the subnetwork layers (today it goes on Ethernet), and the MTS is hidden.

the PUS boxes...

The Commanding box





Ground commanding table

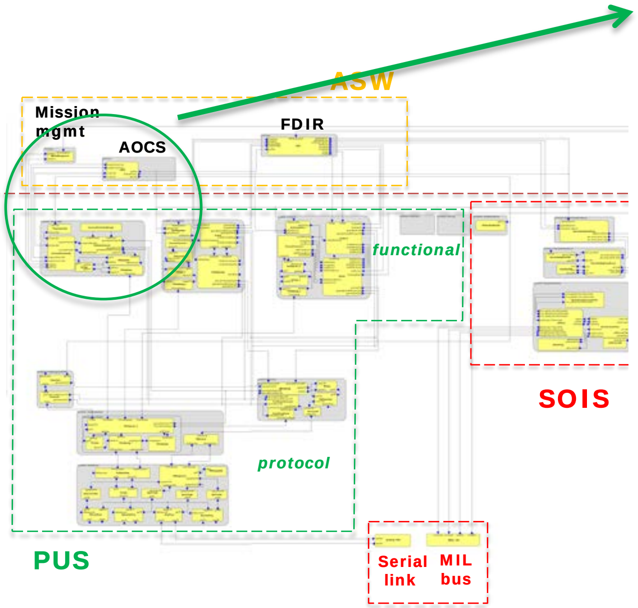
FDIR_impl_inst	FDIR_impl_inst_CMDlist
	change_mode_CMDdesc
AOCS_impl_inst	AOCS_impl_inst_CMDlist
	change_mode_CMDdesc

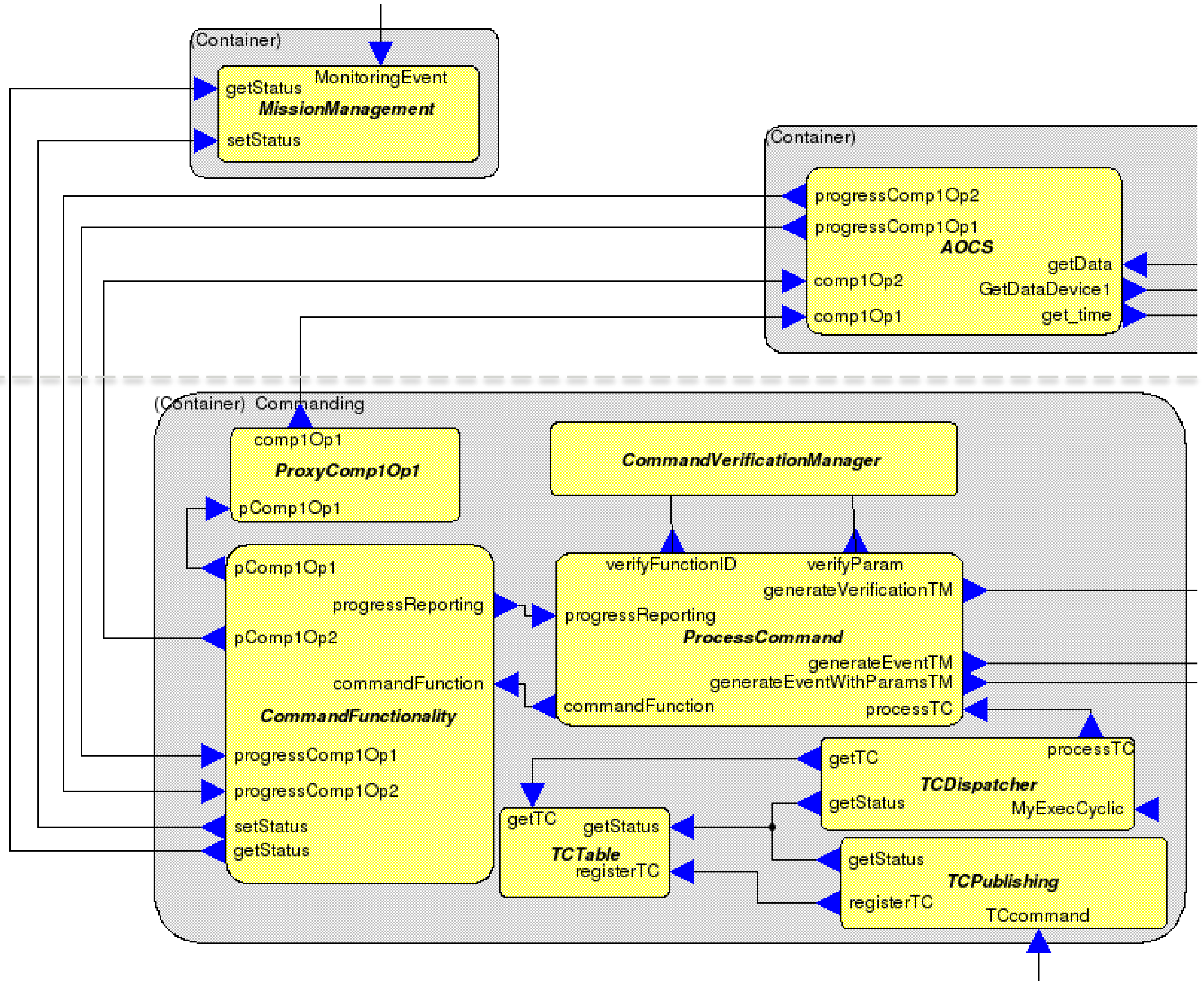
Properties | Model requests interpreter | Problems | Console | Error Log

DLine

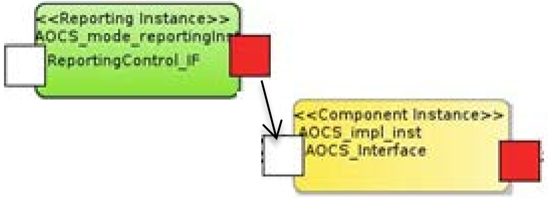
Semantic
Core

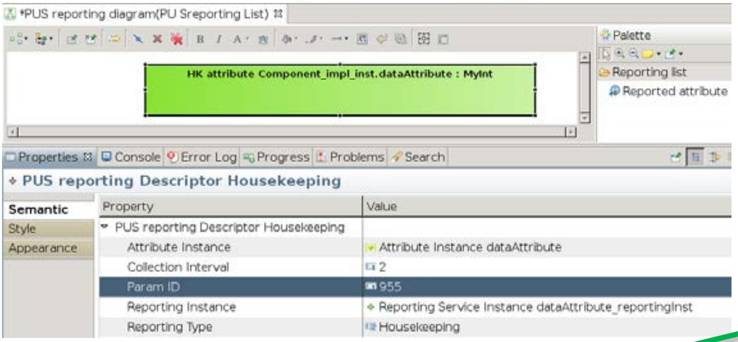
Property	Value
PUS commanding Descriptor change_mode_CMDdesc	
Interface Attribute Access Operation	
Interface Operation	Operation change_mode
Is Hazardous Command	false
Name	change_mode_CMDdesc
PUS service	8
PUS subservice	1





The Housekeeping box

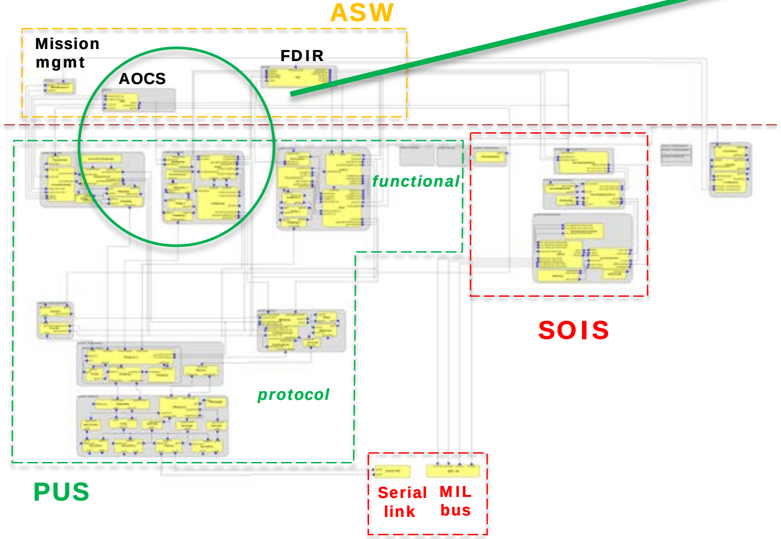


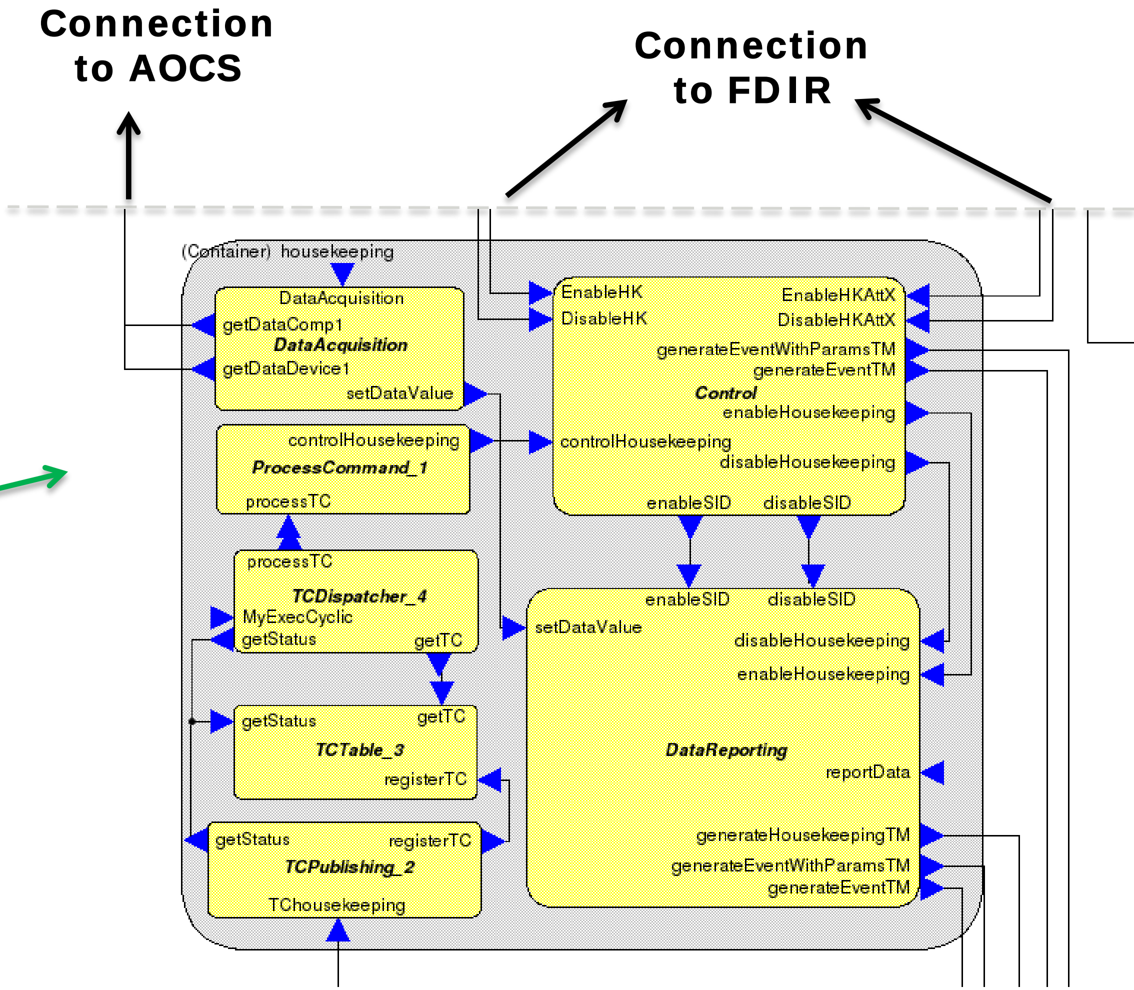

*PUS reporting diagram(PU Sreporting List)

HK attribute Component_impl_inst.dataAttribute : MyInt

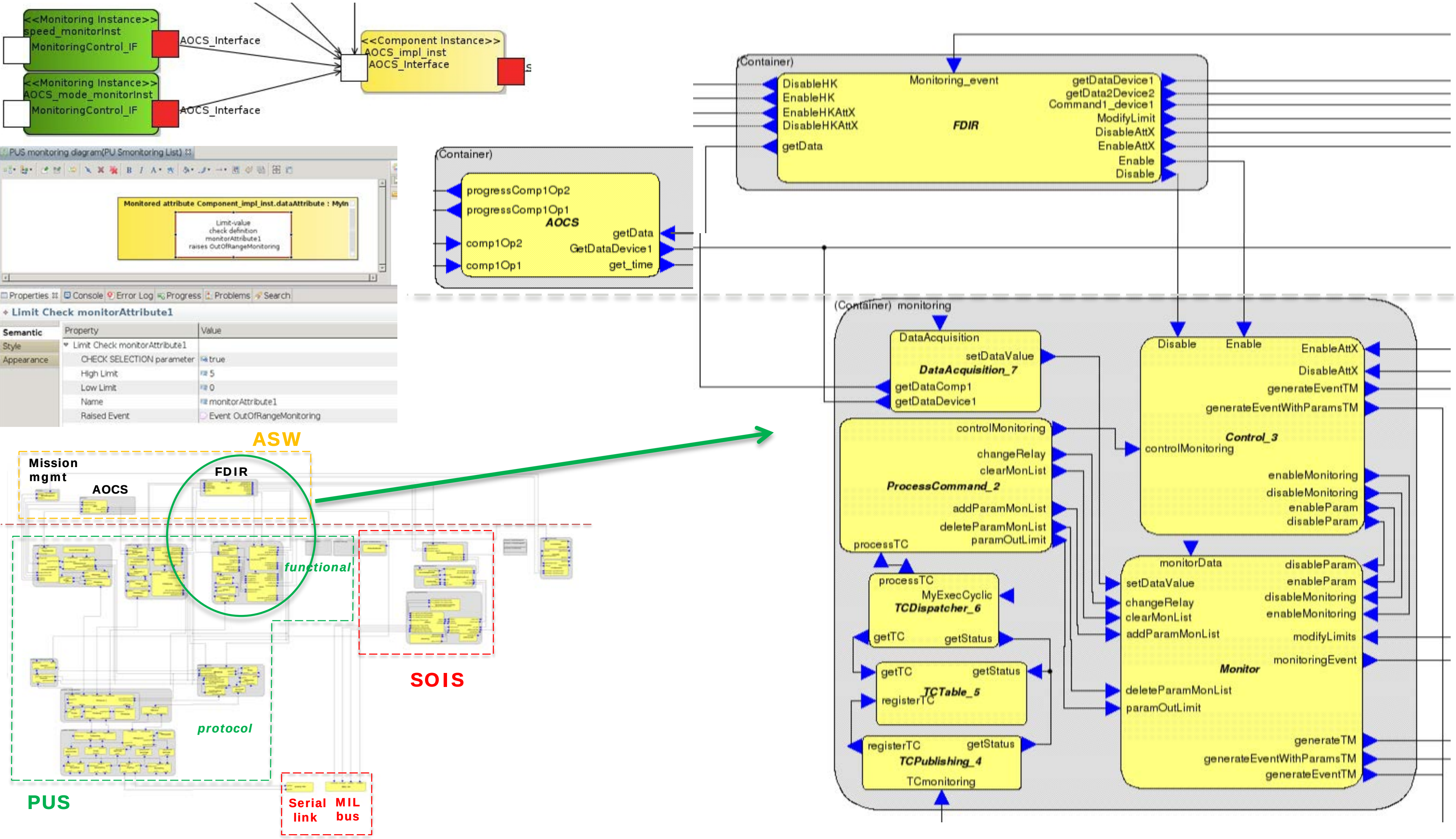
PUS reporting Descriptor Housekeeping

Property	Value
PUS reporting Descriptor Housekeeping	
Attribute Instance	Attribute Instance dataAttribute
Collection Interval	2
Param ID	955
Reporting Instance	Reporting Service Instance dataAttribute_reportingInst
Reporting Type	Housekeeping





The Monitoring box



PUS monitoring diagram(PU Smonitoring List)

Monitored attribute Component_impl_inst.dataAttribute : Mytn

Limit-value check definition monitorAttribute1 raises OutOfRangeMonitoring

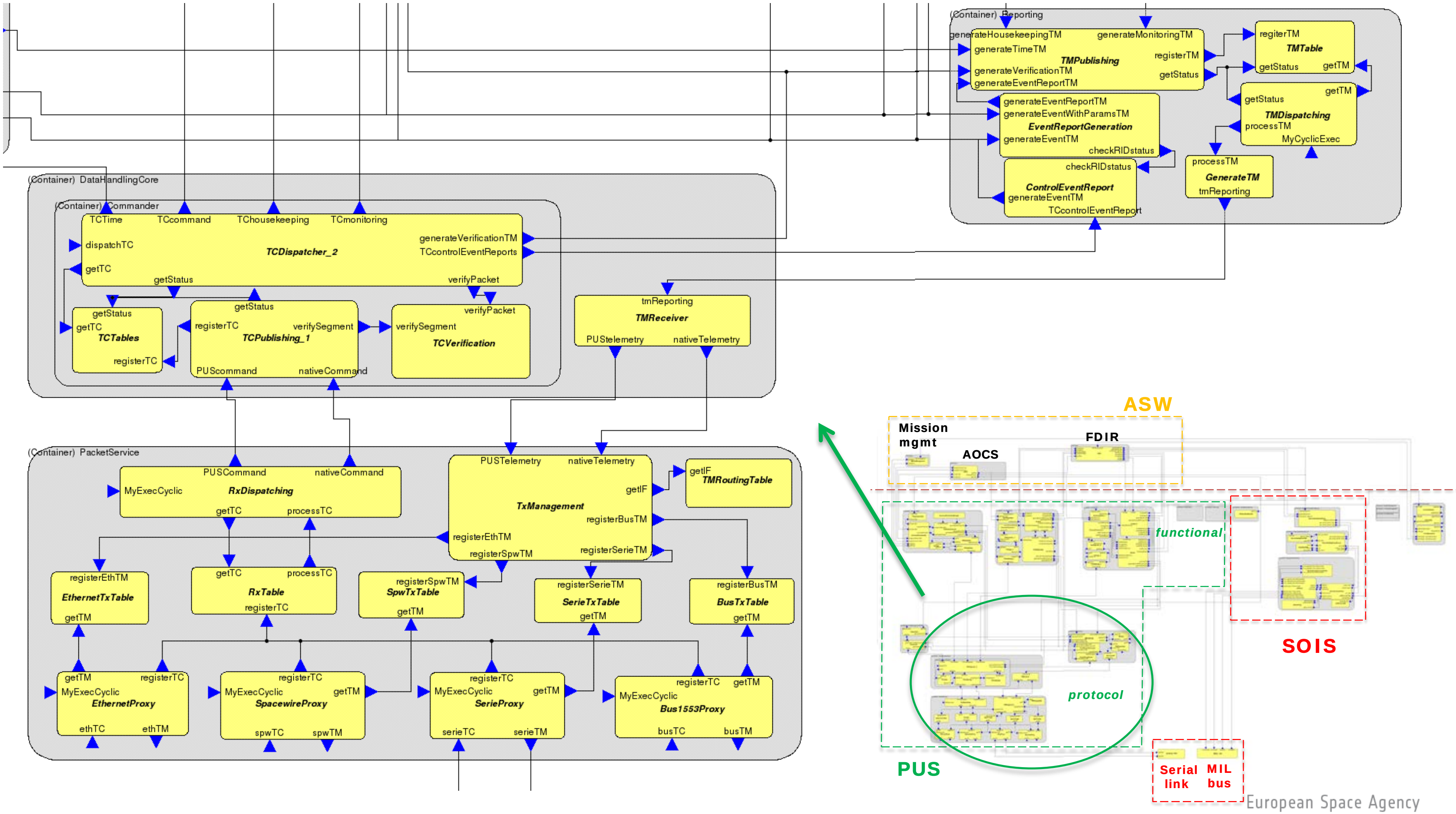
Properties | Console | Error Log | Progress | Problems | Search

Limit Check monitorAttribute1

Semantic
Style
Appearance

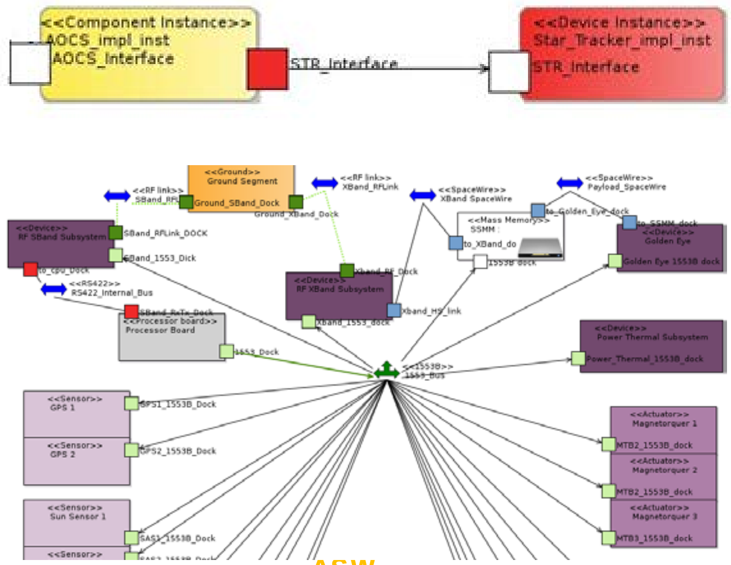
Property	Value
Limit Check monitorAttribute1	
CHECK SELECTION parameter	true
High Limit	5
Low Limit	0
Name	monitorAttribute1
Raised Event	Event OutOfRangeMonitoring

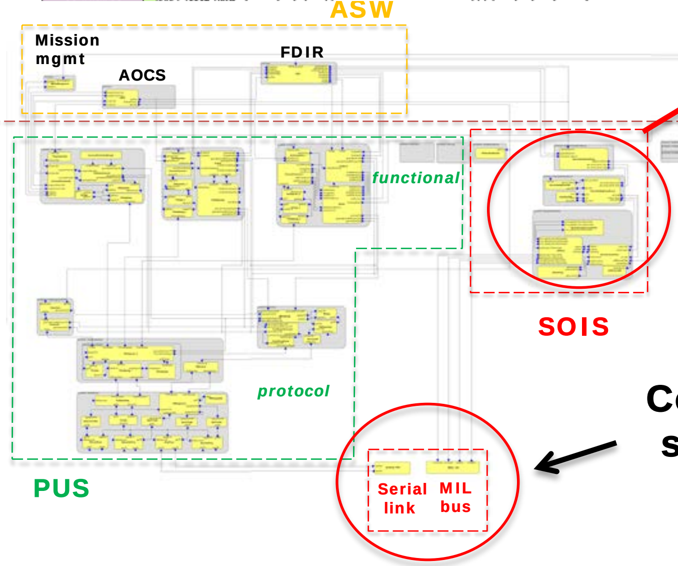
The Data Handling boxes
(packets and protocol related issues)



the SOIS boxes…

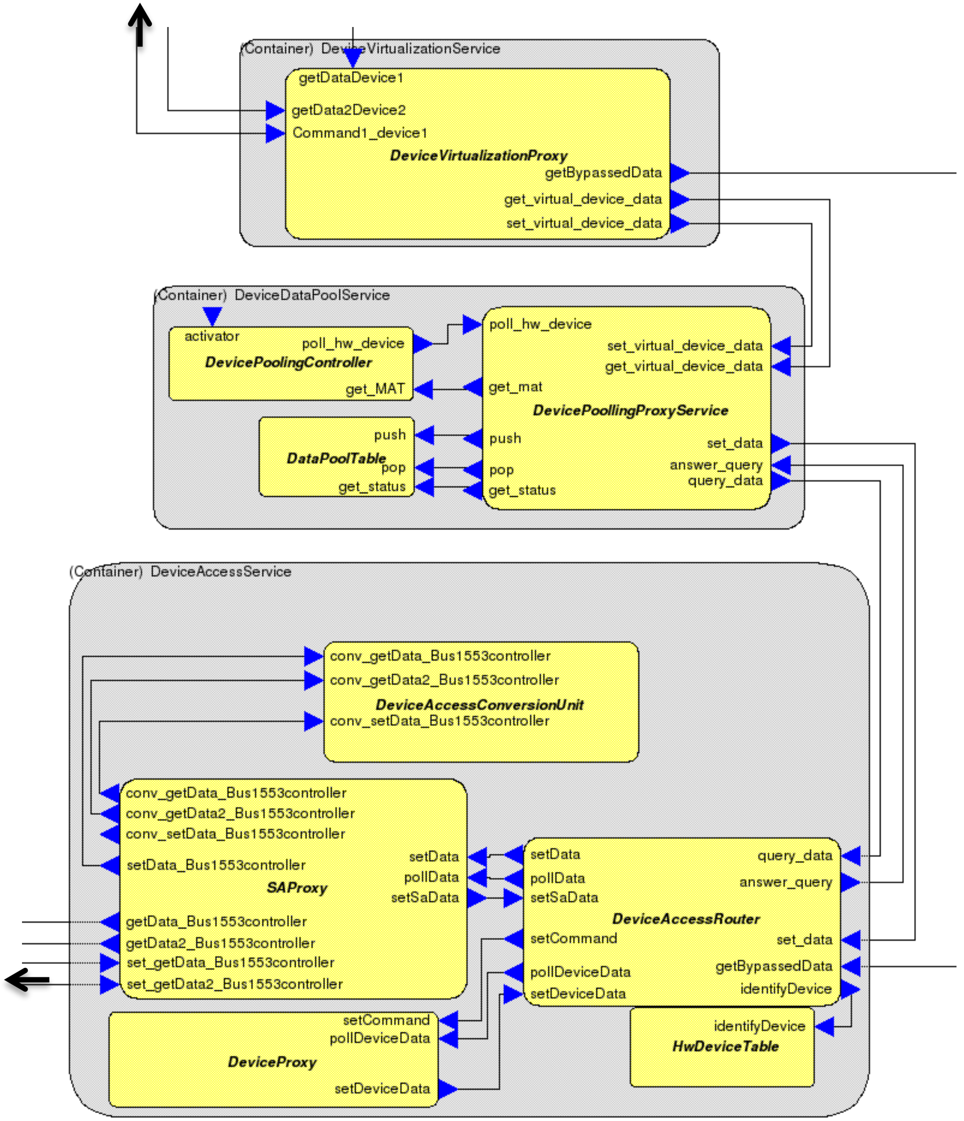
The “SOIS - Command&DataAcquisitionService” box



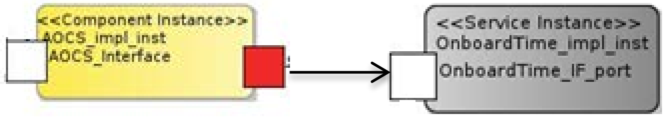


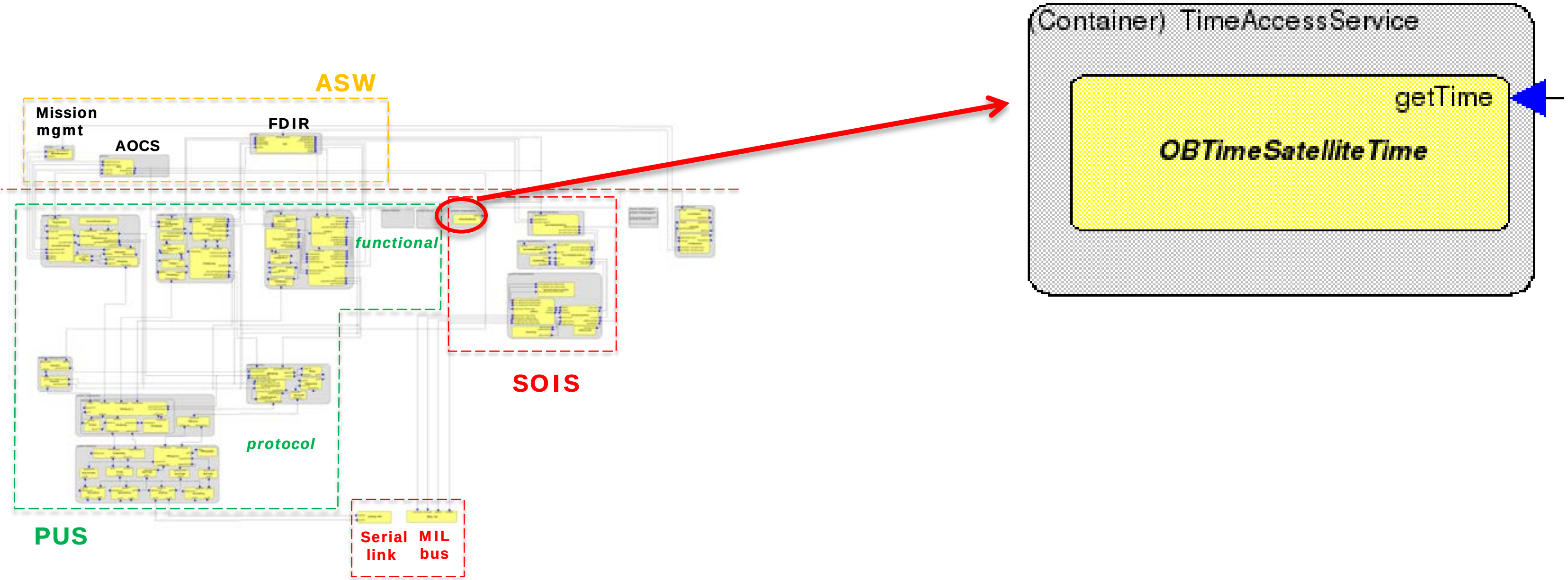
Connection to sub network layer

Connection to AOCS

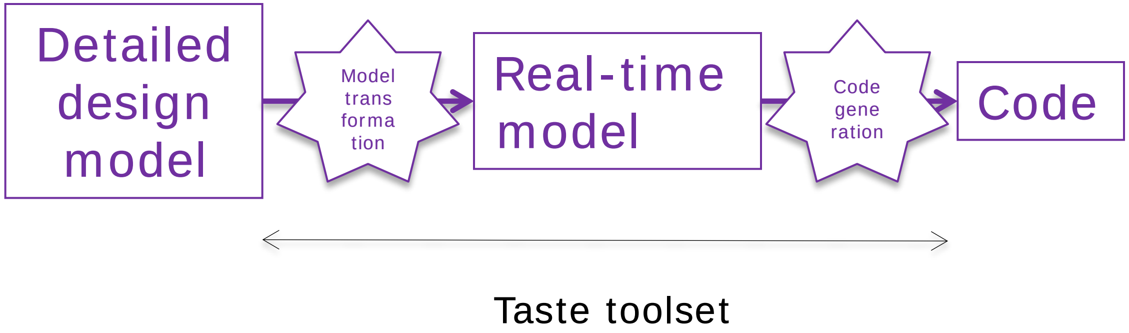


The SOIS Time management box





Taste as a back end



Then, from the detailed design provided by the previous step

And from the behaviour of the functions provided by specific tools (Simulink. State machines)

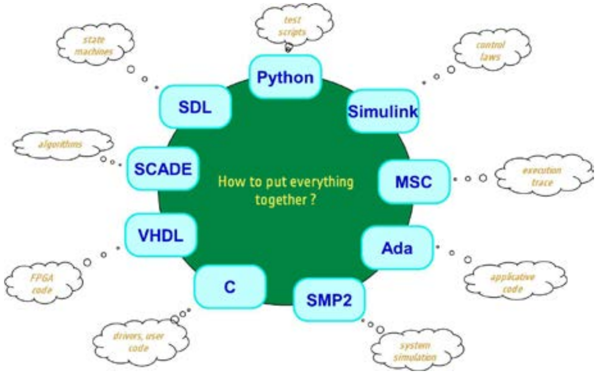
Being given the deployment view (physical topology, hardware)

Taste generates the code

Why Taste?

ESA builds TASTE as an exploration platform implementing state-of-the-art software technologies and targeting:

- **Distributed** on-board software
- Communication with many equipment, embedded devices
- **Heterogeneity** everywhere
  (state machines/control laws,
  integrator/subcontractor,
  hw/sw co-design,
  languages & technologies)
- Based on free, open-source software



Use cases for Taste

TASTE eases the development of consistent software made of:

- Embedded and ground software, GUIs, databases, algorithms
- Software where communication is a central aspect
- Safety-critical components

It serves as a laboratory platform to experiment new technology

It helps ESA to support project engineering phases and reviews

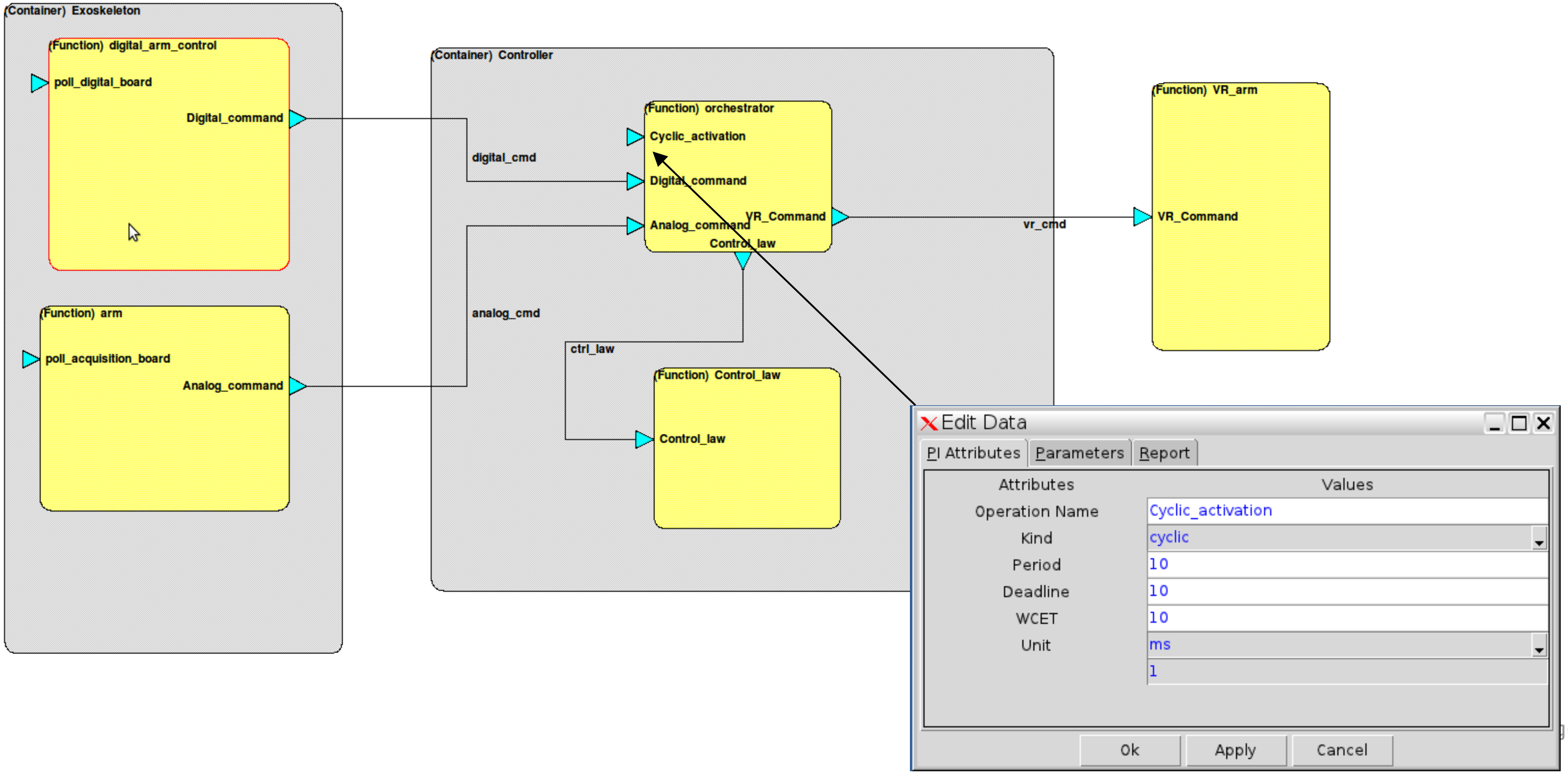
  - Understand the scope of software, the actors, the design
  - Detect issues, ambiguities
  - Run simulations, analyse scenarii

TASTE Software factory - philosophy

- Use existing technologies – glue them together when semantics are compatible
- Don’t reinvent the wheel, software modelling is not new: learn, use and build on top of languages that are mature and widely used in other industries (AADL, ASN.1, SDL, Simulink)
- Let application designers choose the technology that is the most appropriate for each purpose – don’t try to code drivers in UML!
- Automate everything that can be
- Be open and build tools that are ready for technology exploration (multicore, advanced analysis tools, model checking)
- Develop tools that make the life of developers easier – keep the right balance between abstraction and concrete implementation. Both count!
- Target software and systems, not models. Models are just a mean!

How do the tools look like (1)?

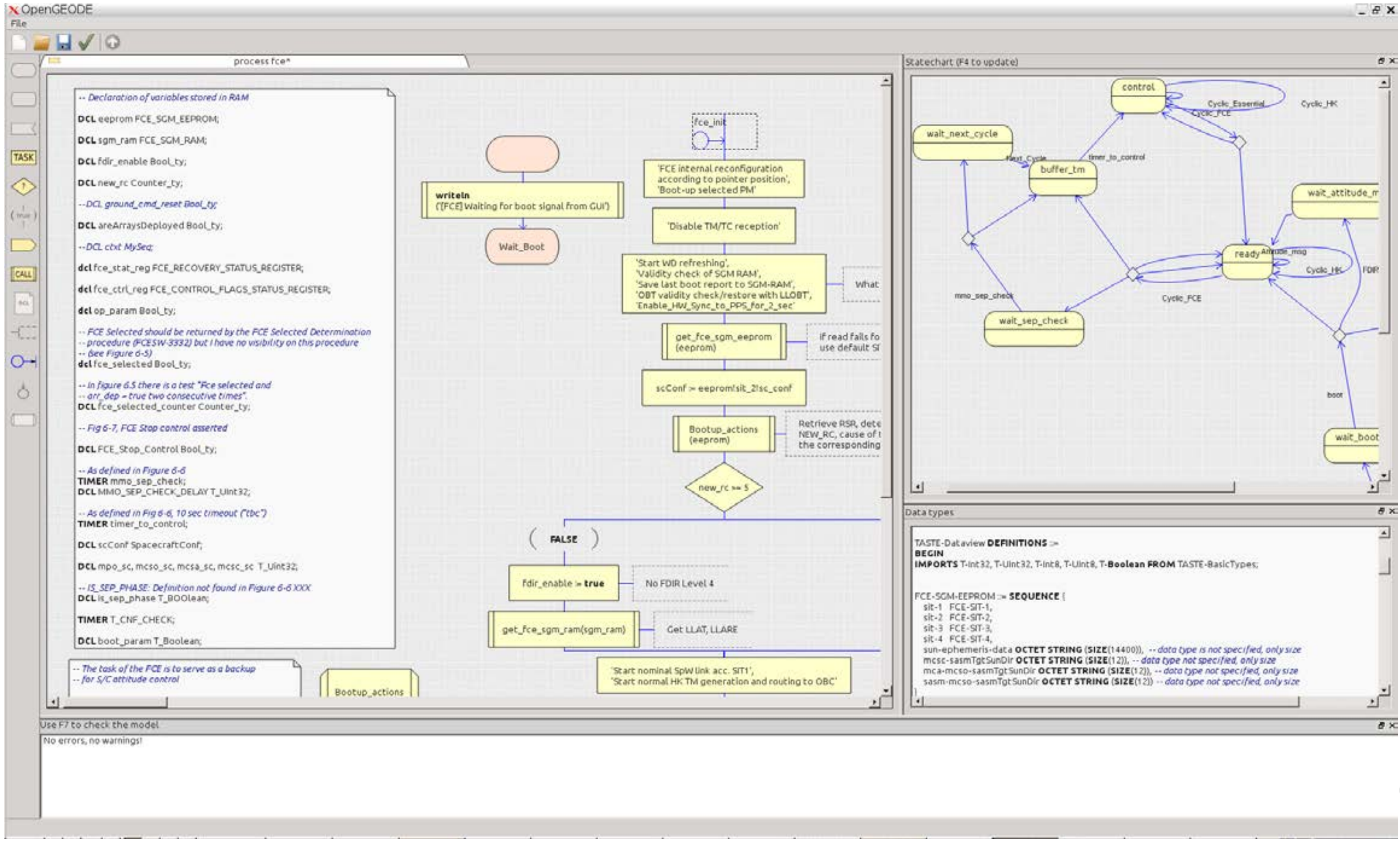
- Graphical approach to unambiguously capture the system architecture and its real-time properties



Attributes	Values
Operation Name	Cyclic_activation
Kind	cyclic
Period	10
Deadline	10
WCET	10
Unit	ms
	1

How do the tools look like (2)?

- Tools to describe state machines – they are complex, and capture the core of the system behaviour



```
-- Declaration of variables stored in RAM

DCL eeprom FCE_SGM_EEPROM;

DCL sgm_ram FCE_SGM_RAM;

DCL fdir_enable Bool_ty;

DCL new_rc Counter_ty;

--DCL ground_cmd_reset Bool_ty;

DCL areArraysDeployed Bool_ty;

--DCL ctxt MySeq;

dcl fce_stat_reg FCE_RECOVERY_STATUS_REGISTER;

dcl fce_ctrl_reg FCE_CONTROL_FLAGS_STATUS_REGISTER;

dcl op_param Bool_ty;

-- FCE Selected should be returned by the FCE Selected Determination
-- procedure (FCESW-3332) but I have no visibility on this procedure
-- (see Figure 6-5)
dcl fce_selected Bool_ty;

-- In figure 6.5 there is a test "Fce selected and
-- arr_dep = true two consecutive times".
DCL fce_selected_counter Counter_ty;

-- Fig 6-7, FCE Stop control asserted

DCL FCE_Stop_Control Bool_ty;

-- As defined in Figure 6-6
TIMER mmo_sep_check;
DCL MMO_SEP_CHECK_DELAY T_UInt32;

-- As defined in Fig 6-6, 10 sec timeout ("tbc")
TIMER timer_to_control;

DCL scConf SpacecraftConf;

DCL mpo_sc, mcso_sc, mcsa_sc, mcsc_sc  T_UInt32;

-- IS_SEP_PHASE: Definition not found in Figure 6-6 XXX
DCL is_sep_phase T_BOOlean;

TIMER T_CNF_CHECK;

DCL boot_param T_Boolean;
```

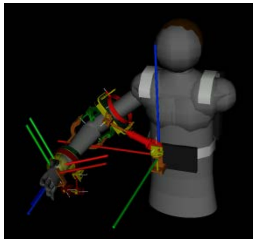
```
-- The task of the FCE is to serve as a backup
-- for S/C attitude control
```

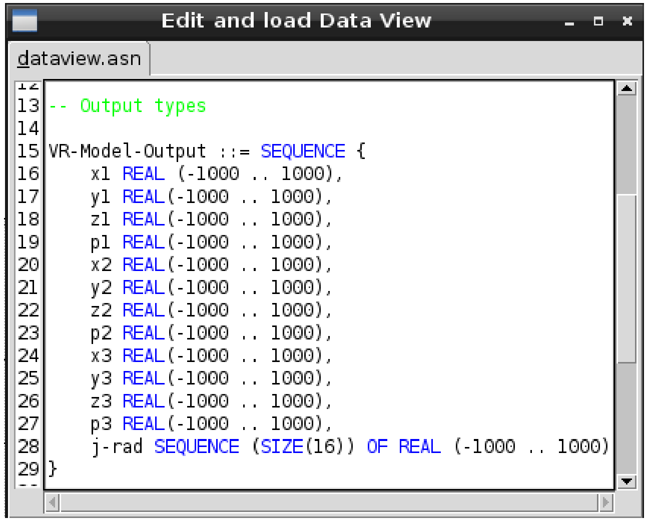
```
TASTE-Dataview DEFINITIONS ::=
BEGIN
IMPORTS T-Int32, T-UInt32, T-Int8, T-UInt8, T-Boolean FROM TASTE-BasicTypes;


FCE-SGM-EEPROM ::= SEQUENCE {
   sit-1   FCE-SIT-1,
   sit-2   FCE-SIT-2,
   sit-3   FCE-SIT-3,
   sit-4   FCE-SIT-4,
   sun-ephemeris-data OCTET STRING (SIZE(14400)),  -- data type is not specified, only size
   mcsc-sasmTgtSunDir OCTET STRING (SIZE(12)), -- data type not specified, only size
   mca-mcso-sasmTgtSunDir OCTET STRING (SIZE(12)), -- data type not specified, only size
   sasm-mcso-sasmTgtSunDir OCTET STRING (SIZE(12)) -- data type not specified, only size
}
```

```
No errors, no warnings!
```

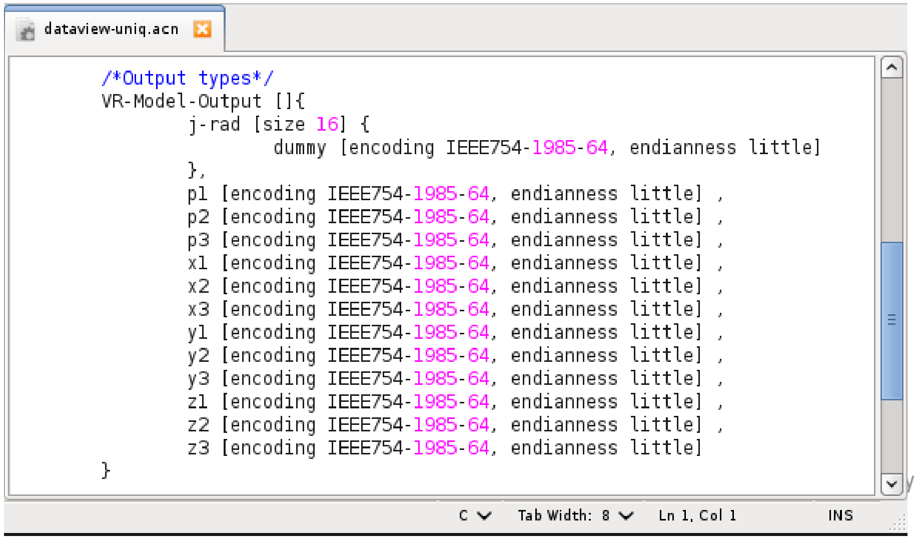
ASN.1 to describe interfaces

- A simple notation to describe software and hardware interfaces
- Our tools generate code for embedded systems (no malloc, no system call, support for C and [Spark] Ada)





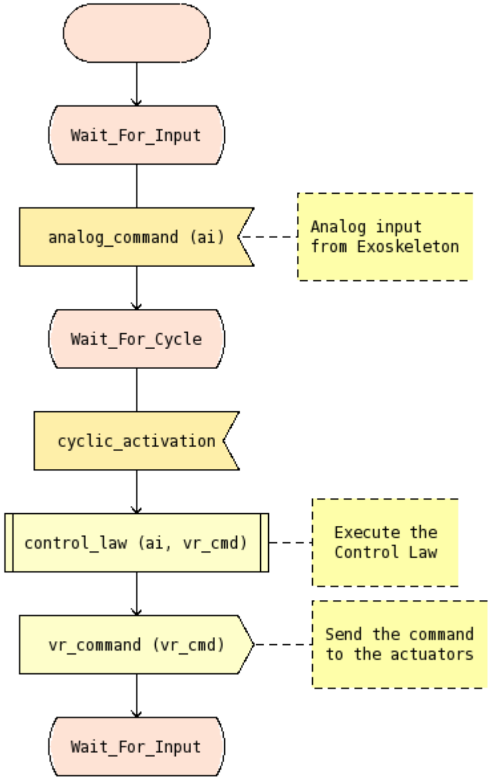
```
-- Output types

VR-Model-Output ::= SEQUENCE {
    x1 REAL (-1000 .. 1000),
    y1 REAL(-1000 .. 1000),
    z1 REAL(-1000 .. 1000),
    p1 REAL(-1000 .. 1000),
    x2 REAL(-1000 .. 1000),
    y2 REAL(-1000 .. 1000),
    z2 REAL(-1000 .. 1000),
    p2 REAL(-1000 .. 1000),
    x3 REAL(-1000 .. 1000),
    y3 REAL(-1000 .. 1000),
    z3 REAL(-1000 .. 1000),
    p3 REAL(-1000 .. 1000),
    j-rad SEQUENCE (SIZE(16)) OF REAL (-1000 .. 1000)
}
```

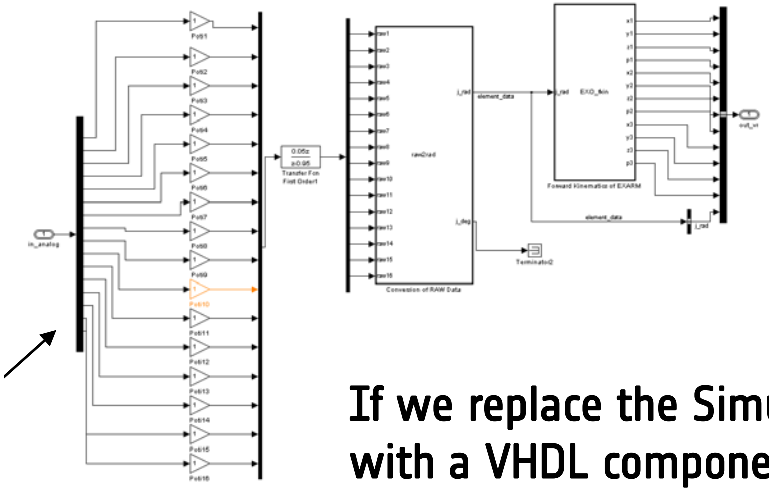


```
/*Output types*/
VR-Model-Output []{
        j-rad [size 16] {
                dummy [encoding IEEE754-1985-64, endianness little]
        },
        p1 [encoding IEEE754-1985-64, endianness little] ,
        p2 [encoding IEEE754-1985-64, endianness little] ,
        p3 [encoding IEEE754-1985-64, endianness little] ,
        x1 [encoding IEEE754-1985-64, endianness little] ,
        x2 [encoding IEEE754-1985-64, endianness little] ,
        x3 [encoding IEEE754-1985-64, endianness little] ,
        y1 [encoding IEEE754-1985-64, endianness little] ,
        y2 [encoding IEEE754-1985-64, endianness little] ,
        y3 [encoding IEEE754-1985-64, endianness little] ,
        z1 [encoding IEEE754-1985-64, endianness little] ,
        z2 [encoding IEEE754-1985-64, endianness little] ,
        z3 [encoding IEEE754-1985-64, endianness little]
}
```

Mix languages to get the best of all worlds – no “unified language” to rule them all!

- The robotic case study mixes C (drivers), SDL (RTDS – system overal orchestration and logic) and Simulink (control laws)





If we replace the Simulink block with a VHDL component, the rest of the system remains unchanged from the user point of view.

Taste capabilities and process

- Capture the system architecture to analyse the **system feasibility**
- Capture data types (ranges, units) to **ensure consistency** everywhere in the system
- Capture the software expected behaviour (state machines, algorithms) and let tools explore this behaviour to **verify** or **discover** some **properties** of the system
- Automate the production of **code** and **documentation.** Support **continuous integration**

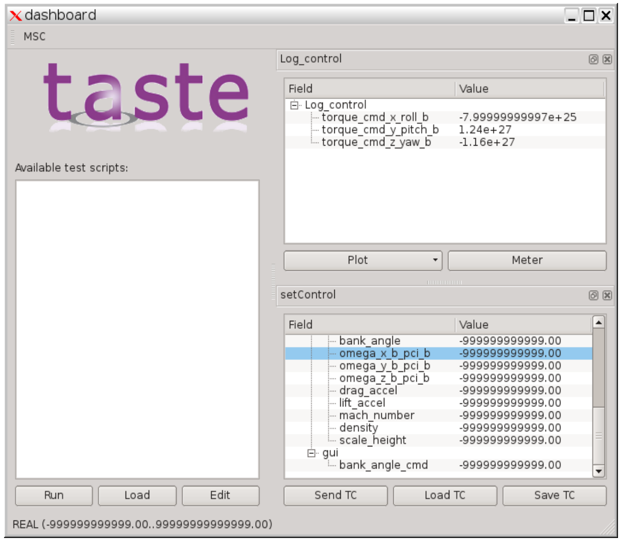
1) Describe the system logical architecture and interfaces
2) Generate code skeletons and write the applicative code or models
3) Capture the system hardware and deployment
4) Verify models
5) Build the system and download it on target
6) Monitor and interact with the system at run-time

Software factories must have strong basis

- **TASTE relies on formal languages** :
  - ASN.1 and AADL to capture the software architecture and data
  - SDL, Simulink, SCADE, C, Ada, VHDL, … to capture the software behaviour
  - MSC and Python to test
- **Combine graphical AND textual notations**
  - If anything goes wrong, human can fix textual syntax
  - Diagrams for easier understanding
  - But some information is textual by nature
- **Avoid languages with weak semantics or syntax**

And make developers and testers’ life easier

- Generate additional code to help users test their system (real-time monitoring and interaction with the binary)



dashboard

MSC

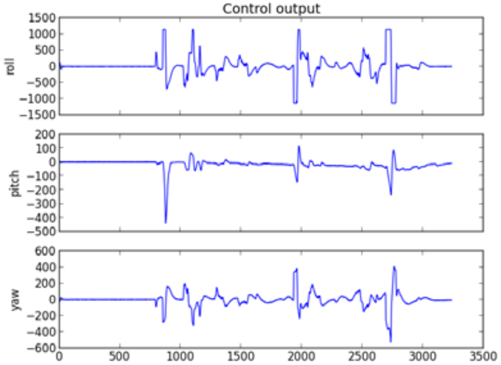
Log_control

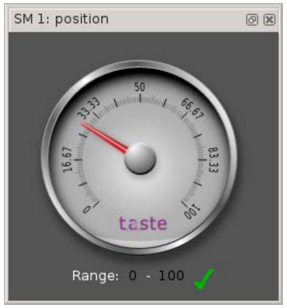
Field	Value
⊟ Log_control	
torque_cmd_x_roll_b	-7.99999999997e+25
torque_cmd_y_pitch_b	1.24e+27
torque_cmd_z_yaw_b	-1.16e+27

setControl

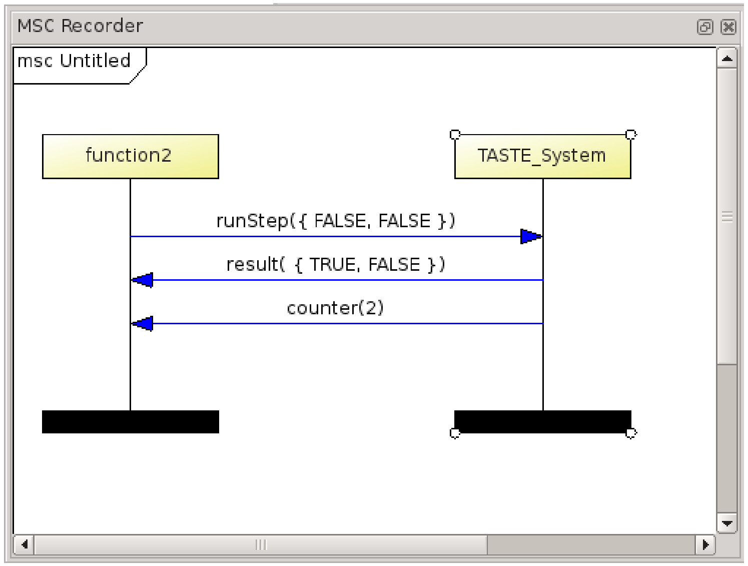
Field	Value
bank_angle	-999999999999.00
omega_x_b_pci_b	-999999999999.00
omega_y_b_pci_b	-999999999999.00
omega_z_b_pci_b	-999999999999.00
drag_accel	-999999999999.00
lift_accel	-999999999999.00
mach_number	-999999999999.00
density	-999999999999.00
scale_height	-999999999999.00
⊟ gui	
bank_angle_cmd	-999999999999.00

REAL (-999999999999.00..99999999999999.00)





SM 1: position

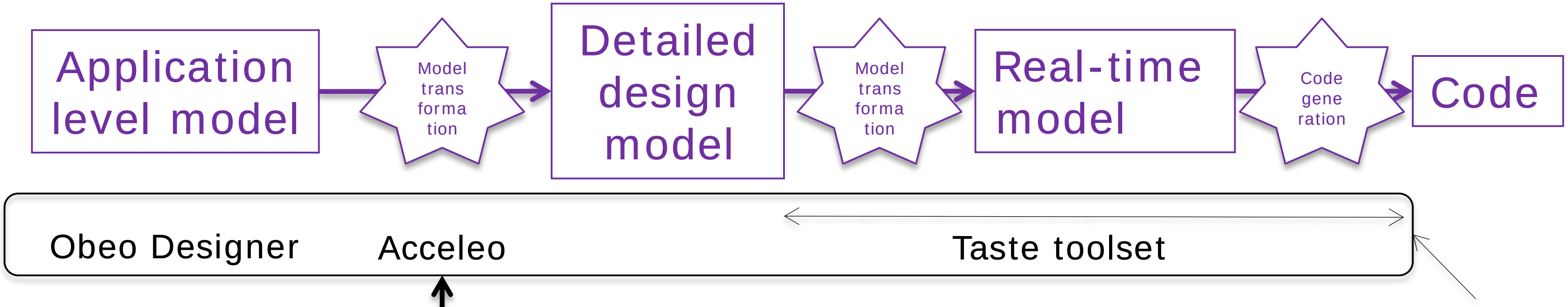


MSC Recorder

msc Untitled

Presentation of the Architectural steps

- One principle of OSRA is “separation of concerns”: application to subsystems engineers (AOCS), architecture to software architect, implementation to real-time software engineers (supported by tools)
- Therefore a toolset (“software factory”) takes an application level model and generates the code



Application level model

Model transformation

Detailed design model

Model transformation

Real-time model

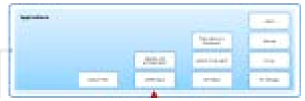
Code generation

Code

Obeo Designer

Acceleo

Taste toolset



The knowledge of the architectural design (PUS and SOIS design patterns, libraries) is in the model transformation tool

http://taste.tuxfamily.org

The software factory implements and generates the software bus

Contact

Feedback: savoir@esa.int

http://taste.tuxfamily.org