Model driven SW architecture: ESA Euclid and PLATO control SW examples.

Emanuele Galli

emanuele.galli@inaf.it

On behalf of CDPU-SW Team

(A.M. Di Giorgio, M. Farina, G. Giusi, S. J. Liu, S.Pezzuto, A. Russi)

TEchnologies for Telescopes and Instrument control Software

Some definitions common to MDA

- **Models** consist of sets of elements that describe some physical, abstract, or hypothetical reality.
  - Good models serve as means of communication
  - they're cheaper to build than the real thing
  - and they can be transformed into an implementation

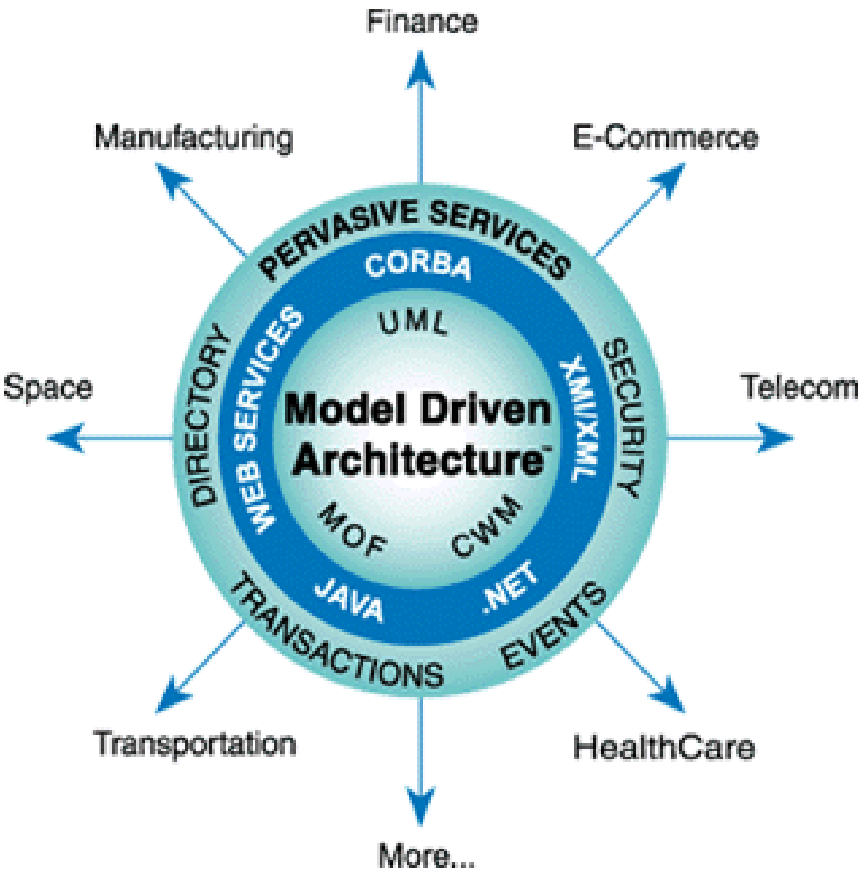
- **Metamodel** is basically a model of a modelling language and defines structure, semantics and constraints

- **Platform** is the specification of an execution environment for a set of models

What is Model Driven Architecture?

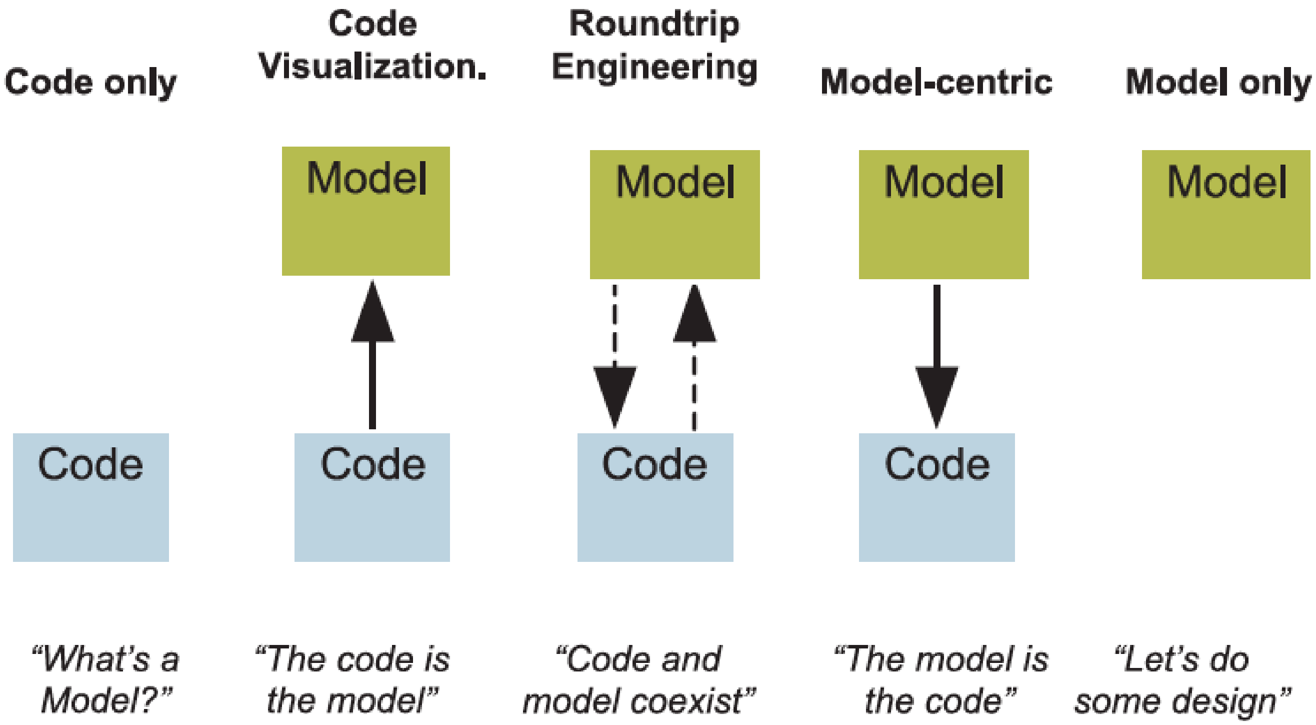
- ***What is MDA?*** MDA is actually three things.
  1. An Object Management Group initiative to develop standards based on the idea that modelling is a better foundation for developing and maintaining systems.
  2. A brand for standards and products that adhere to those standards.
  3. A set of technologies and techniques associated with those standards.

- Central to MDA is the notion of creating different models at different levels of abstraction and then linking them together to form an implementation.
- MDA start with a Platform Independent Model (PIM) to a Platform Specific Model (PSM) and then to the implementation specific



MDA is supported by the Unified Modelling Language (**UML**), Meta Object Facility (**MOF**), XML Metadata Interchange (**XMI**), and Common Warehouse Metamodel (**CWM**)

How model are synchronised?



Code only

Code Visualization.

Roundtrip Engineering

Model-centric

Model only

“What’s a Model?”

“The code is the model”

“Code and model coexist”

“The model is the code”

“Let’s do some design”

An interesting question is: which of these approaches can we describe as “model-driven?”

There is **not** a definitive answer. Growing consensus:

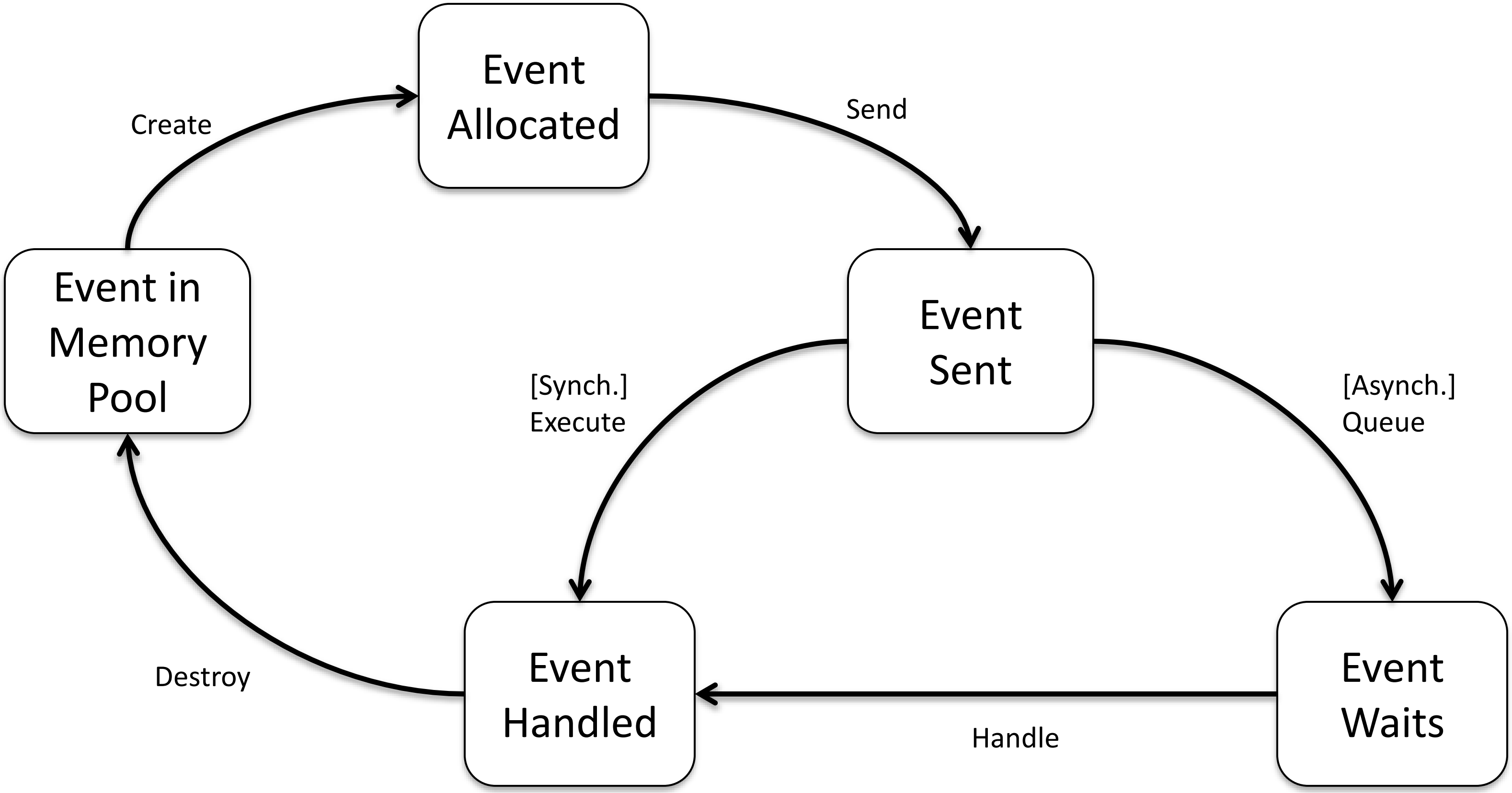
- MDA is associated with model-driven approached in which code is (semi-)automatically generated from abstract models
- uses standard specification languages for describing models and transformation between them

IBM Rational Rhapsody as MDA tool

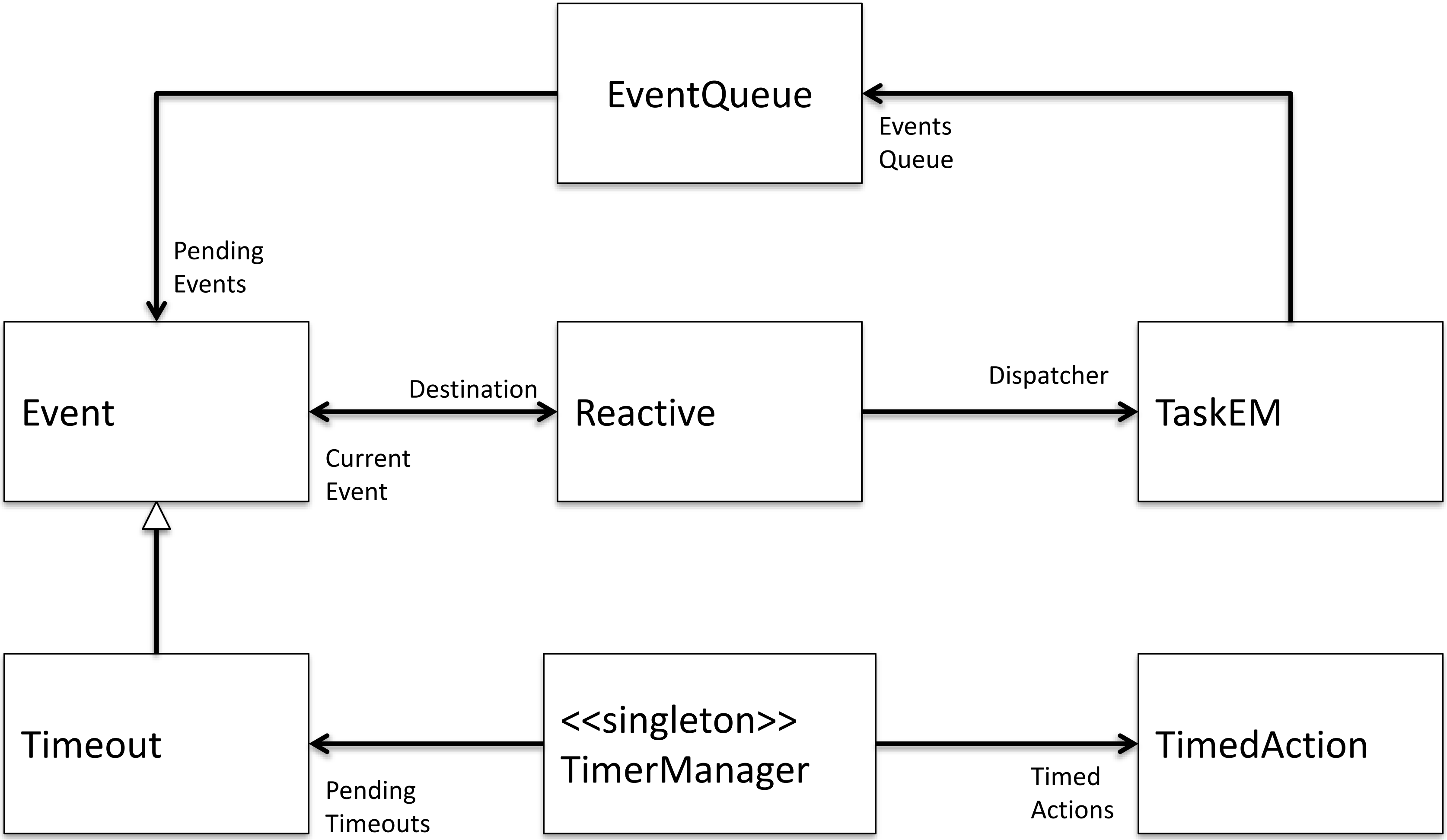
- Is **IBM Rational Rhapsody** an MDA tool: yes, but it is not according to the standard (e.g. is not possible to export the UML model by using the XMI)

- Main functionalities
  - Fully integration with IBM DOORS for the import and trace of requirements to design models/objects/functions
  - Possibility to convert a SysML project to a UML project(from version 8.3.x) switching the view from the same window
  - Support multiple profiles:
    - Project Type: Real-Time, DDS, NetCentril for Web Services, SysML, …
    - Project Setting: MISRA C/C++, Safety Critical, Code Centric (to use it as a blueprint)
  - Supported languages: C, C++, ADA, Java
  - Supported OS: Windows, Linux, VxWorks 6.5.3, QNX
  - Formal model checking
  - Schedulability analysis support

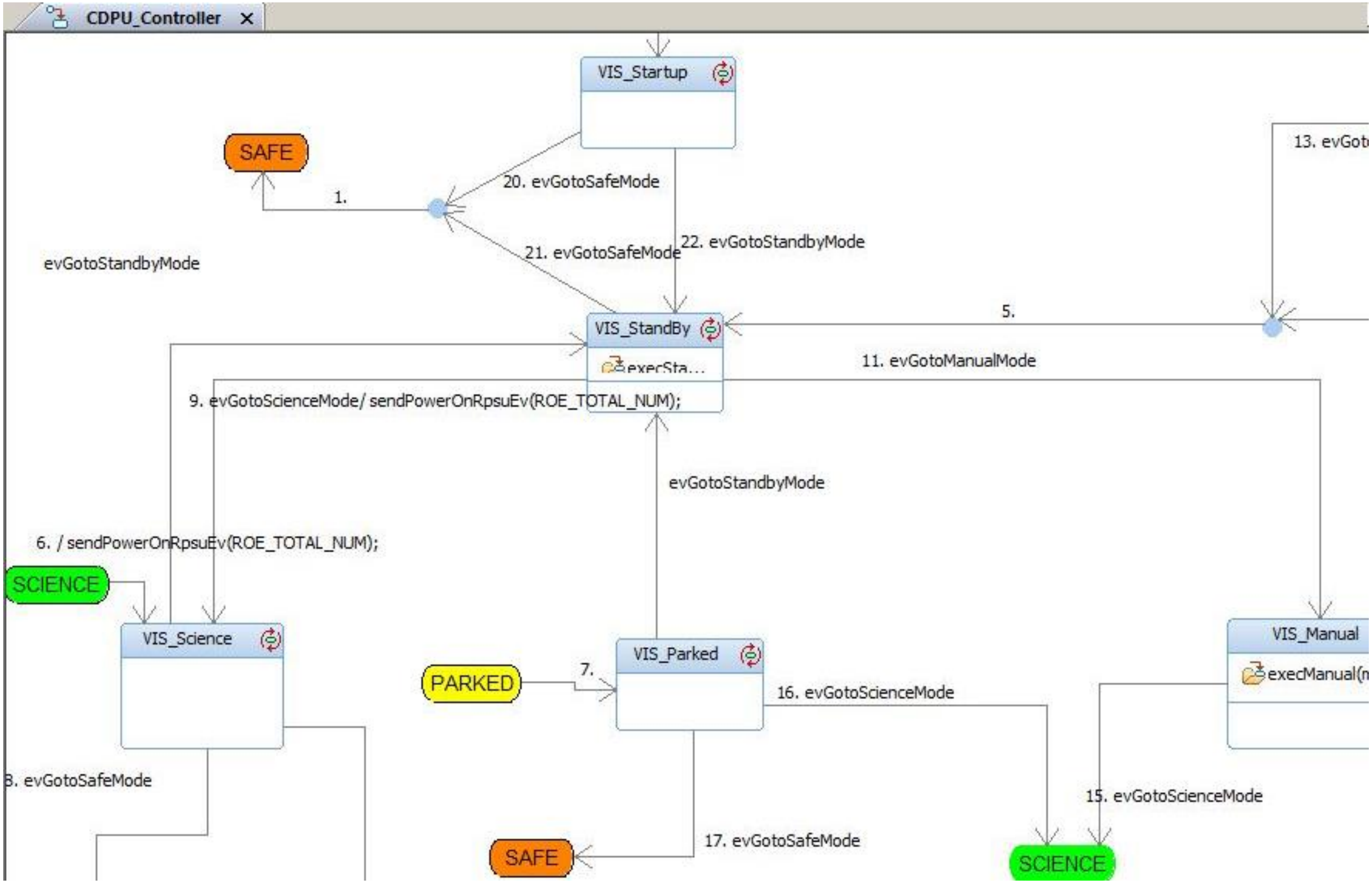
Event processing



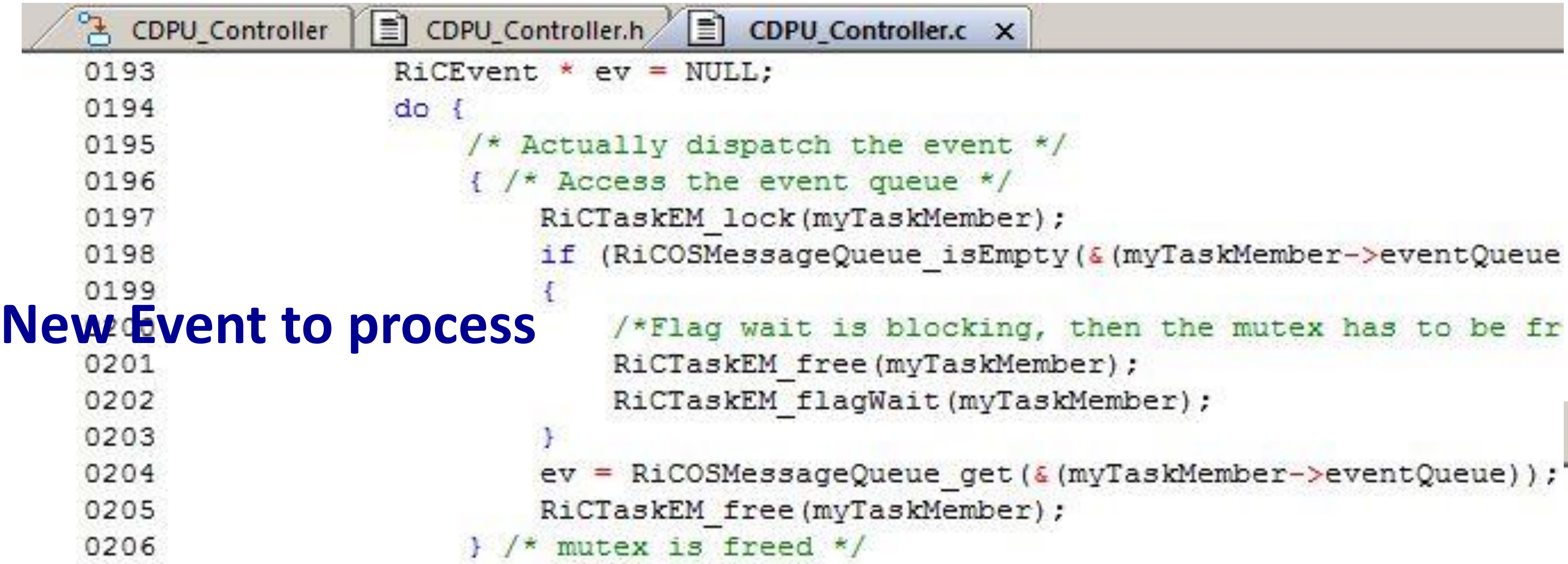
Type of events and tasks interaction



Example of use with State Diagram

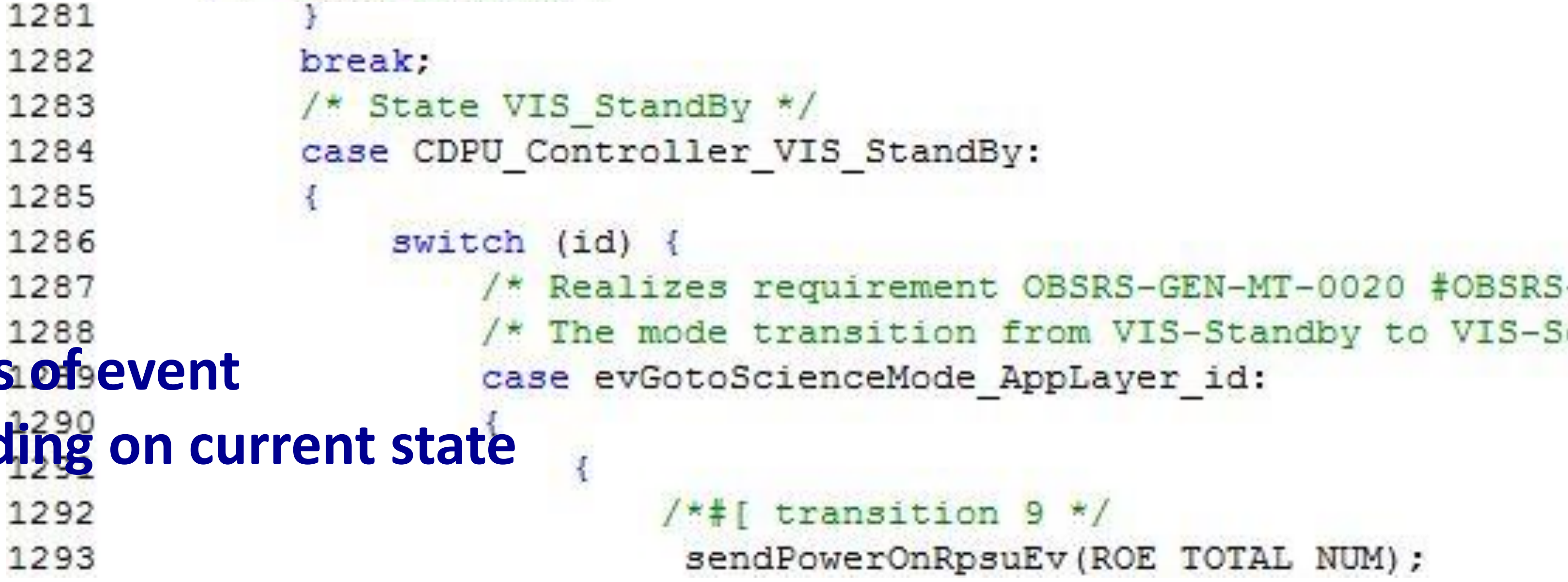


Example of use with State Diagram(2)



New Event to process

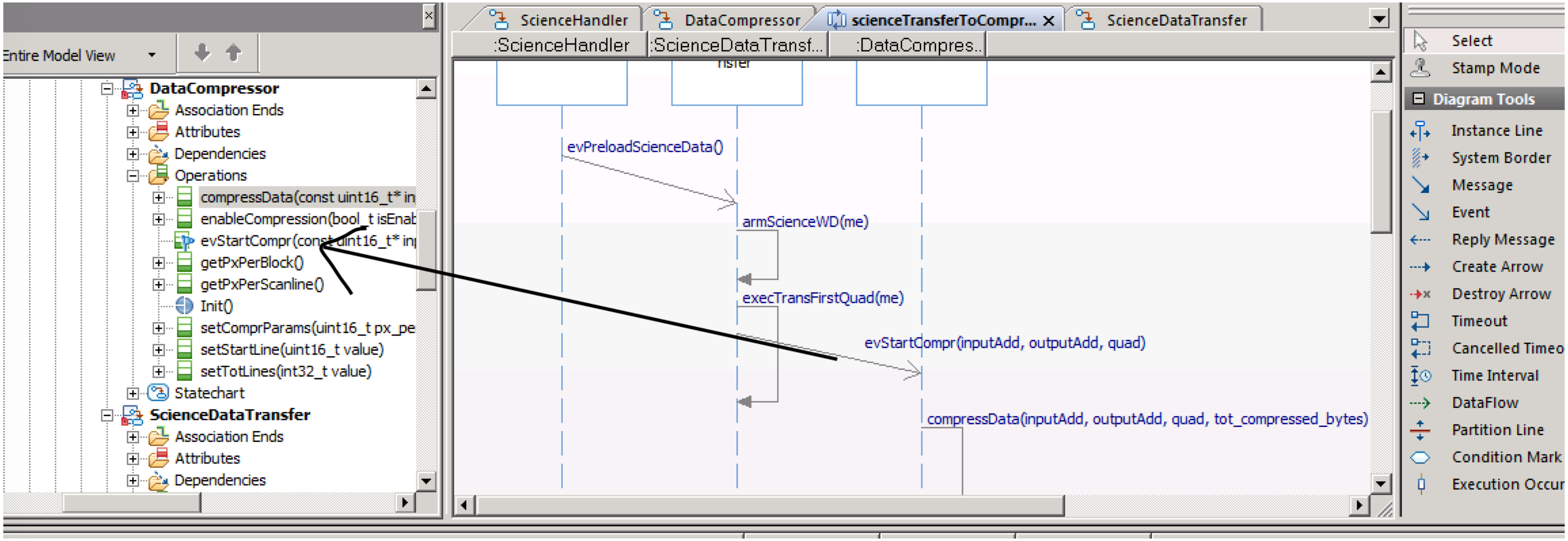
```
RiCEvent * ev = NULL;
do {
    /* Actually dispatch the event */
    { /* Access the event queue */
        RiCTaskEM_lock(myTaskMember);
        if (RiCOSMessageQueue_isEmpty(&(myTaskMember->eventQueue
        {
            /*Flag wait is blocking, then the mutex has to be fr
            RiCTaskEM_free(myTaskMember);
            RiCTaskEM_flagWait(myTaskMember);
        }
        ev = RiCOSMessageQueue_get(&(myTaskMember->eventQueue));
        RiCTaskEM_free(myTaskMember);
    } /* mutex is freed */
```



Process of event depending on current state

```
}
break;
/* State VIS_StandBy */
case CDPU_Controller_VIS_StandBy:
{
    switch (id) {
        /* Realizes requirement OBSRS-GEN-MT-0020 #OBSRS
        /* The mode transition from VIS-Standby to VIS-S
        case evGotoScienceMode_AppLayer_id:
        {
            {
                /*#[ transition 9 */
                 sendPowerOnRpsuEv(ROE TOTAL NUM);
```

Example with Sequence Diagram

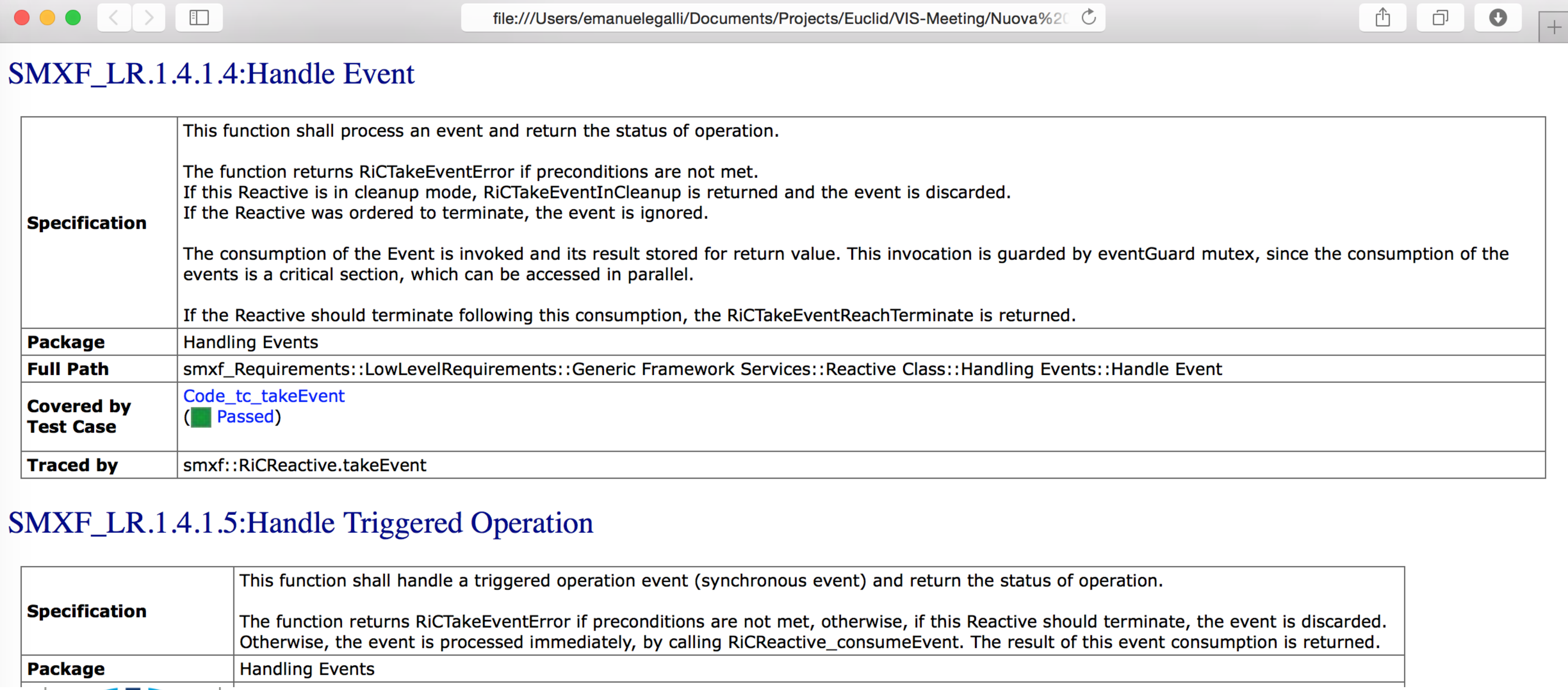


Events are translated to messages exchanged between tasks

Messages are translated to function calls

Requirements covering

- For EUCLID, the IBM Gateway tool was adopted:
  - is a gateway between the Software Requirement Document and the design as well as the code itself
  - Necessity to create a parser
- For PLATO, import of requirements directly from DOORS



file:///Users/emanuelegalli/Documents/Projects/Euclid/VIS-Meeting/Nuova%2

SMXF_LR.1.4.1.4:Handle Event

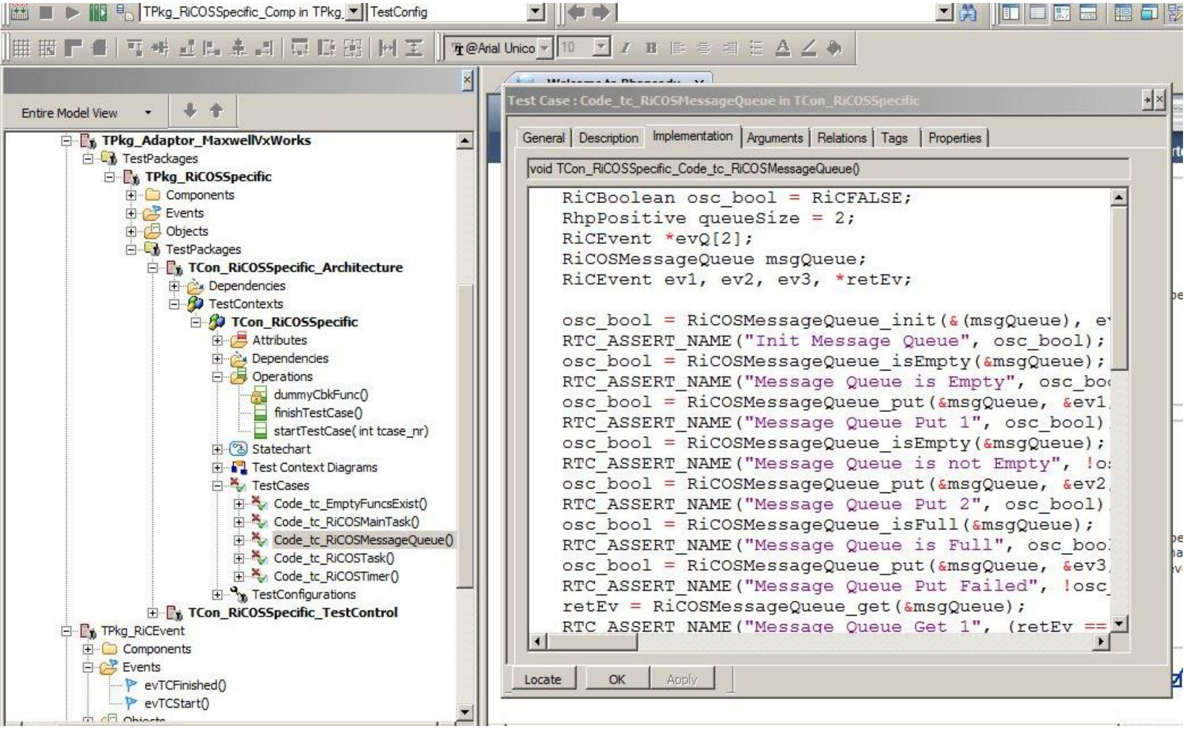
Specification	This function shall process an event and return the status of operation. The function returns RiCTakeEventError if preconditions are not met. If this Reactive is in cleanup mode, RiCTakeEventInCleanup is returned and the event is discarded. If the Reactive was ordered to terminate, the event is ignored. The consumption of the Event is invoked and its result stored for return value. This invocation is guarded by eventGuard mutex, since the consumption of the events is a critical section, which can be accessed in parallel. If the Reactive should terminate following this consumption, the RiCTakeEventReachTerminate is returned.
Package	Handling Events
Full Path	smxf_Requirements::LowLevelRequirements::Generic Framework Services::Reactive Class::Handling Events::Handle Event
Covered by Test Case	Code_tc_takeEvent (🟩 Passed)
Traced by	smxf::RiCReactive.takeEvent

SMXF_LR.1.4.1.5:Handle Triggered Operation

Specification	This function shall handle a triggered operation event (synchronous event) and return the status of operation. The function returns RiCTakeEventError if preconditions are not met, otherwise, if this Reactive should terminate, the event is discarded. Otherwise, the event is processed immediately, by calling RiCReactive_consumeEvent. The result of this event consumption is returned.
Package	Handling Events

Model Testing

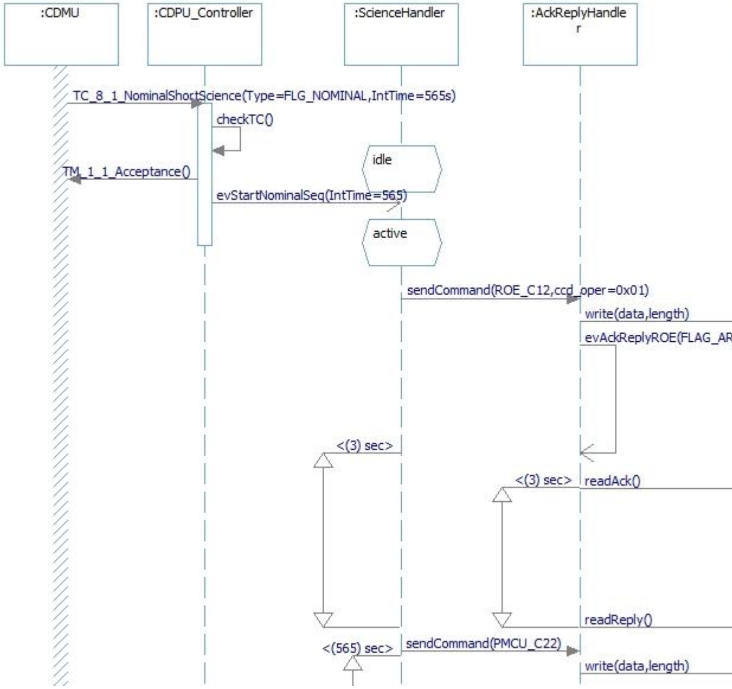
- Model and Unit testing can be performed directly inside the IBM Rational Rhapsody using the optional tool **IBM TestConductor**
  - Events can be configured and generated at a specific tick-time
  - Sequence diagrams can be generated to see if the behavior was as expected
- Static Analysis performed with external tools (i.e. **Parasot C++Test**)



void TCon_RiCOSSpecific_Code_tc_RiCOSMessageQueue()

```
   RiCBoolean osc_bool = RiCFALSE;
   RhpPositive queueSize = 2;
   RiCEvent *evQ[2];
   RiCOSMessageQueue msgQueue;
   RiCEvent ev1, ev2, ev3, *retEv;

   osc_bool = RiCOSMessageQueue_init(&(msgQueue), e
   RTC_ASSERT_NAME("Init Message Queue", osc_bool);
   osc_bool = RiCOSMessageQueue_isEmpty(&msgQueue);
   RTC_ASSERT_NAME("Message Queue is Empty", osc_bo
   osc_bool = RiCOSMessageQueue_put(&msgQueue, &ev1
   RTC_ASSERT_NAME("Message Queue Put 1", osc_bool)
   osc_bool = RiCOSMessageQueue_isEmpty(&msgQueue);
   RTC_ASSERT_NAME("Message Queue is not Empty", !o
   osc_bool = RiCOSMessageQueue_put(&msgQueue, &ev2
   RTC_ASSERT_NAME("Message Queue Put 2", osc_bool)
   osc_bool = RiCOSMessageQueue_isFull(&msgQueue);
   RTC_ASSERT_NAME("Message Queue is Full", osc_boo
   osc_bool = RiCOSMessageQueue_put(&msgQueue, &ev3
   RTC_ASSERT_NAME("Message Queue Put Failed", !osc
   retEv = RiCOSMessageQueue_get(&msgQueue);
   RTC_ASSERT_NAME("Message Queue Get 1", (retEv ==
```



Use of Rhapsody in IAPS

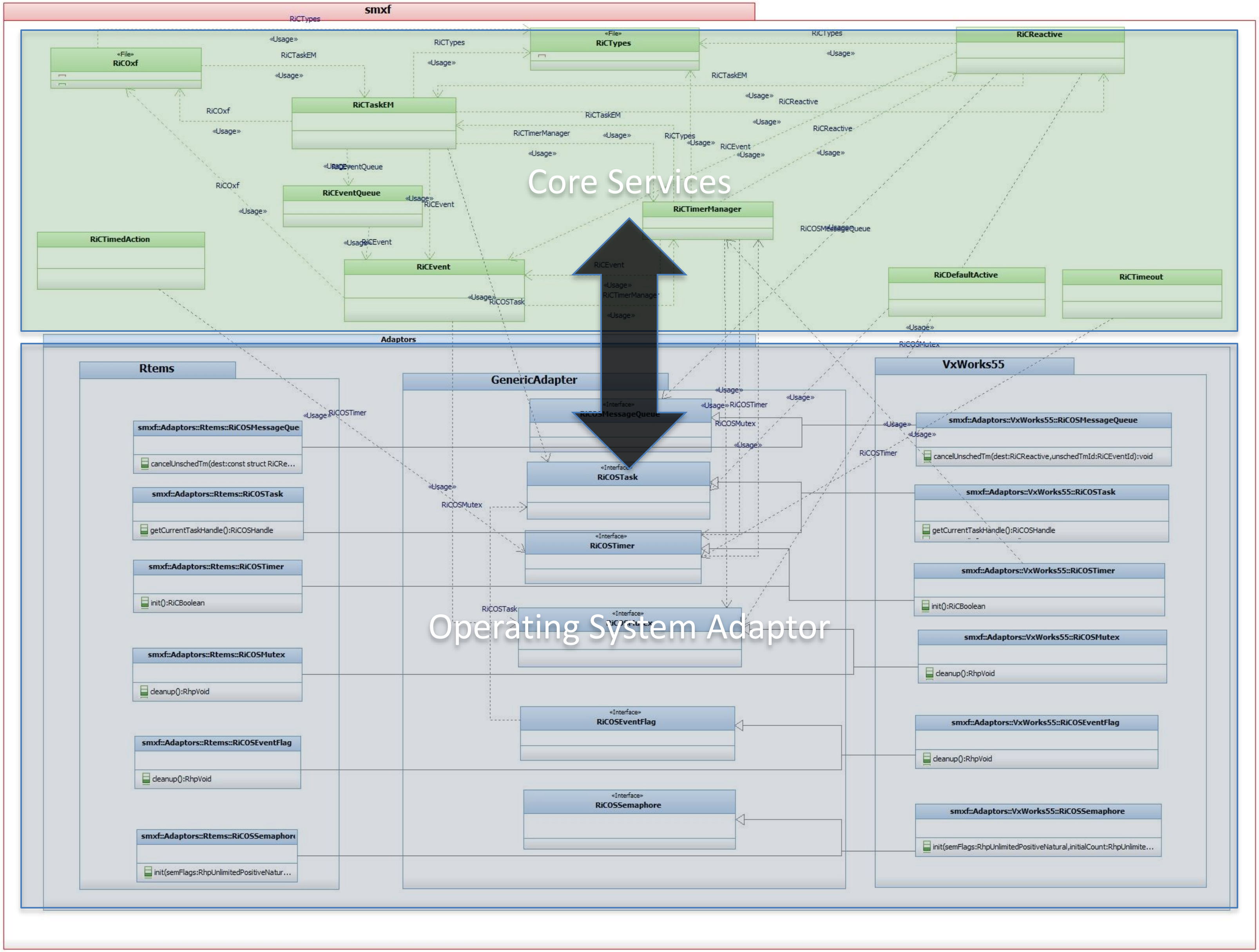
- The IBM Rhaposdy tool has been used for the developing of:
  - EUCLID VIS CDPU Application Software (SW QAR ongoing)
  - PLATO ICU Application Software (Delivered v0.4 for the EM#1)

- EUCLID VIS CDPU is based on a Maxwell Board SCS750 with the VxWorks 5 as RTOS

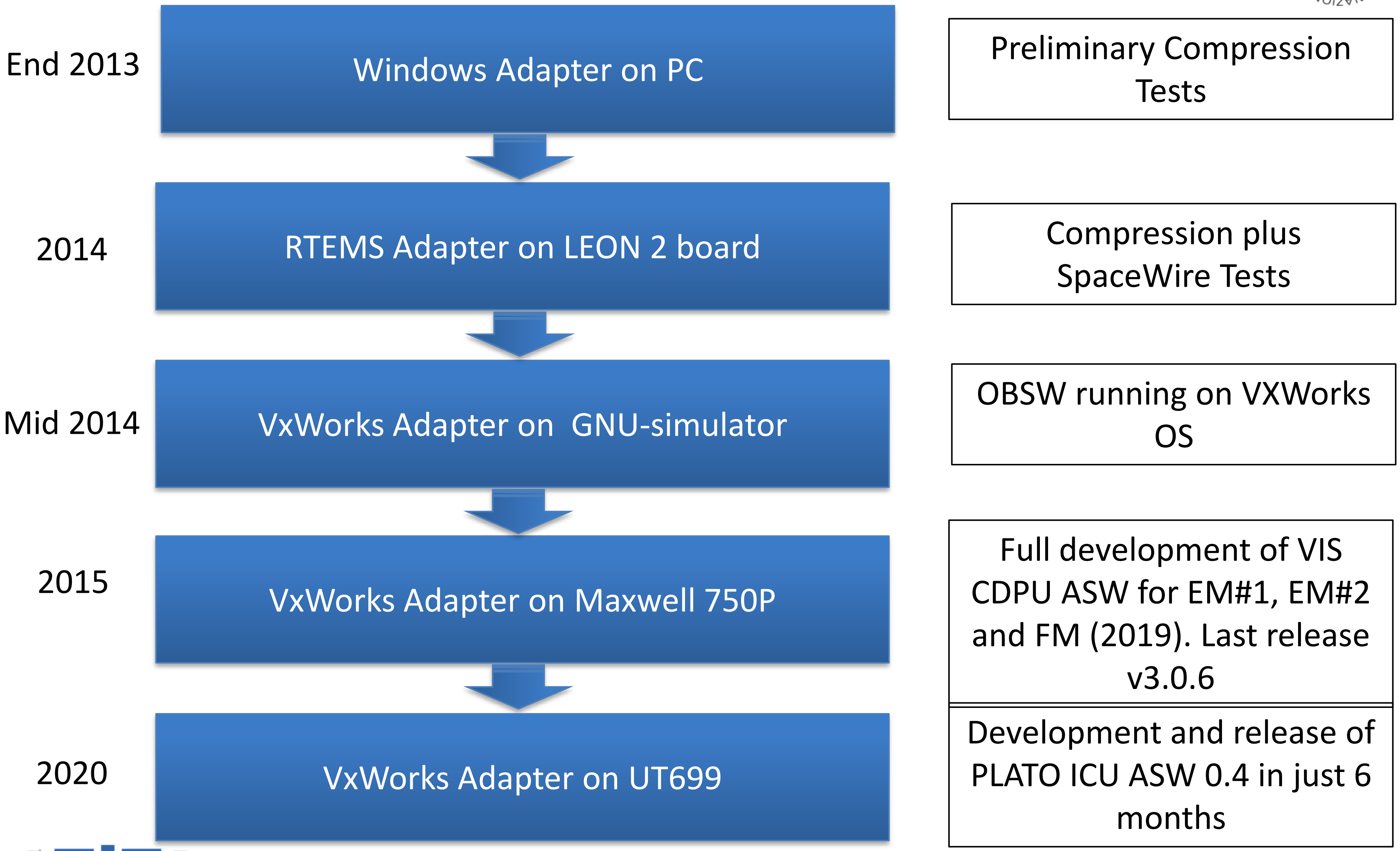
- PLATO ICU is based on a LEON3 with the RTEMS 4.8 (pre-qualified) as RTOS

- Both of them are not supported natively by Rhapsody ☹

Implemented Wrapper to RTOS



What we were able to do…



End 2013 — Windows Adapter on PC — Preliminary Compression Tests

2014 — RTEMS Adapter on LEON 2 board — Compression plus SpaceWire Tests

Mid 2014 — VxWorks Adapter on GNU-simulator — OBSW running on VXWorks OS

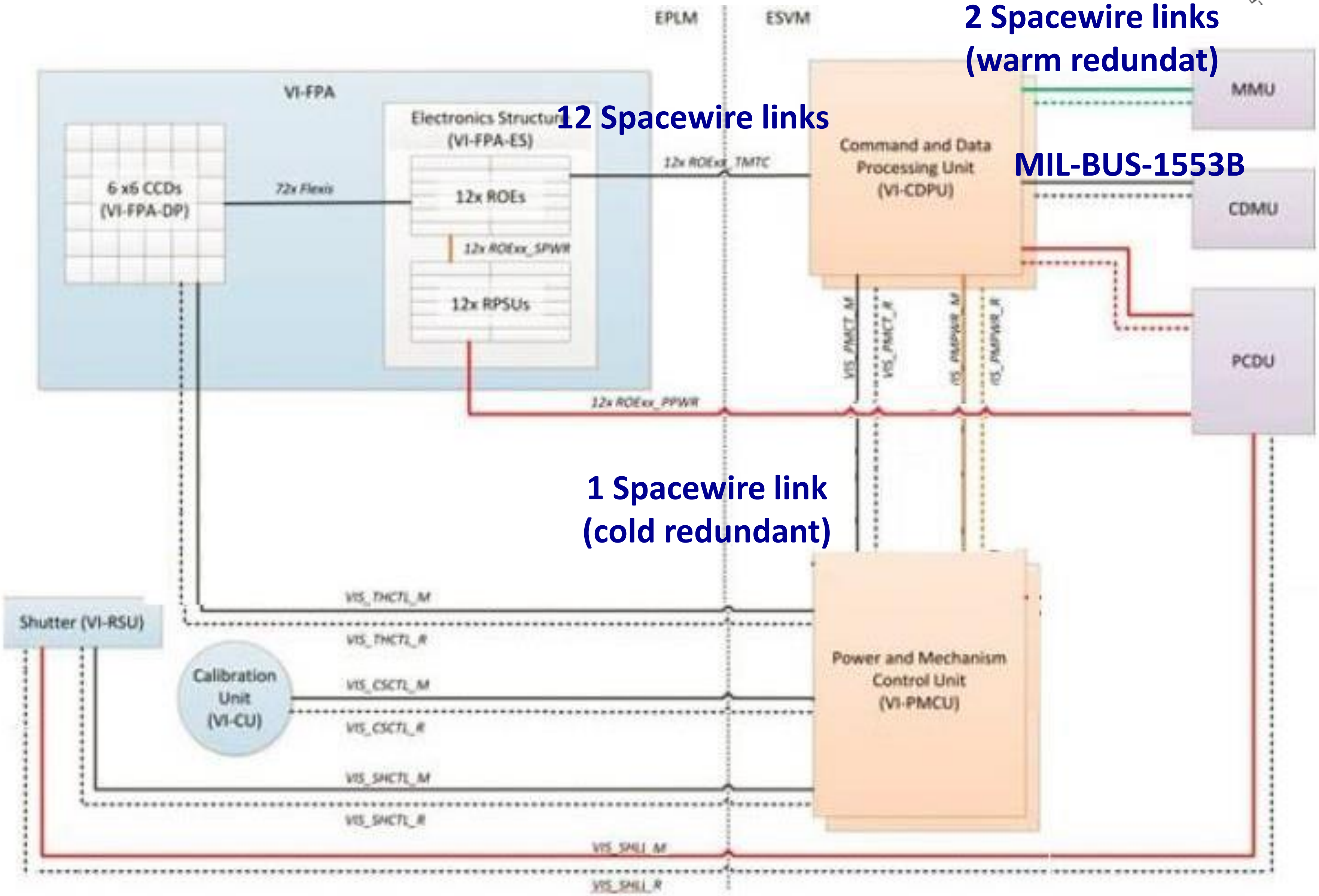
2015 — VxWorks Adapter on Maxwell 750P — Full development of VIS CDPU ASW for EM#1, EM#2 and FM (2019). Last release v3.0.6

2020 — VxWorks Adapter on UT699 — Development and release of PLATO ICU ASW 0.4 in just 6 months

EUCLID VIS instrument

- VIS instrument consists of:
  - A **Focal Plane (FPA)** with 36 CCDs and 12 Readout (ROE) each one handling 3 CCDs. Each CCD is 4238*4132 px
  - A **Calibration Unit (CU)** which provides uniform illumination for calibration purposes
  - A **Shutter Unit (RSU)** for on demand occultation of telescope light
  - A **Payload Mechanism Control Unit (PMCU)** which oversees distributing power to CU, driving RSU and monitoring temperature of FPA
  - A **Control Data Processing Unit (CDPU)** which is in charge of monitoring and controls the instruments, performs data processing and transfer Science Data to the Space Craft

VIS Electrical Architecture



2 Spacewire links
(warm redundat)

12 Spacewire links

MIL-BUS-1553B

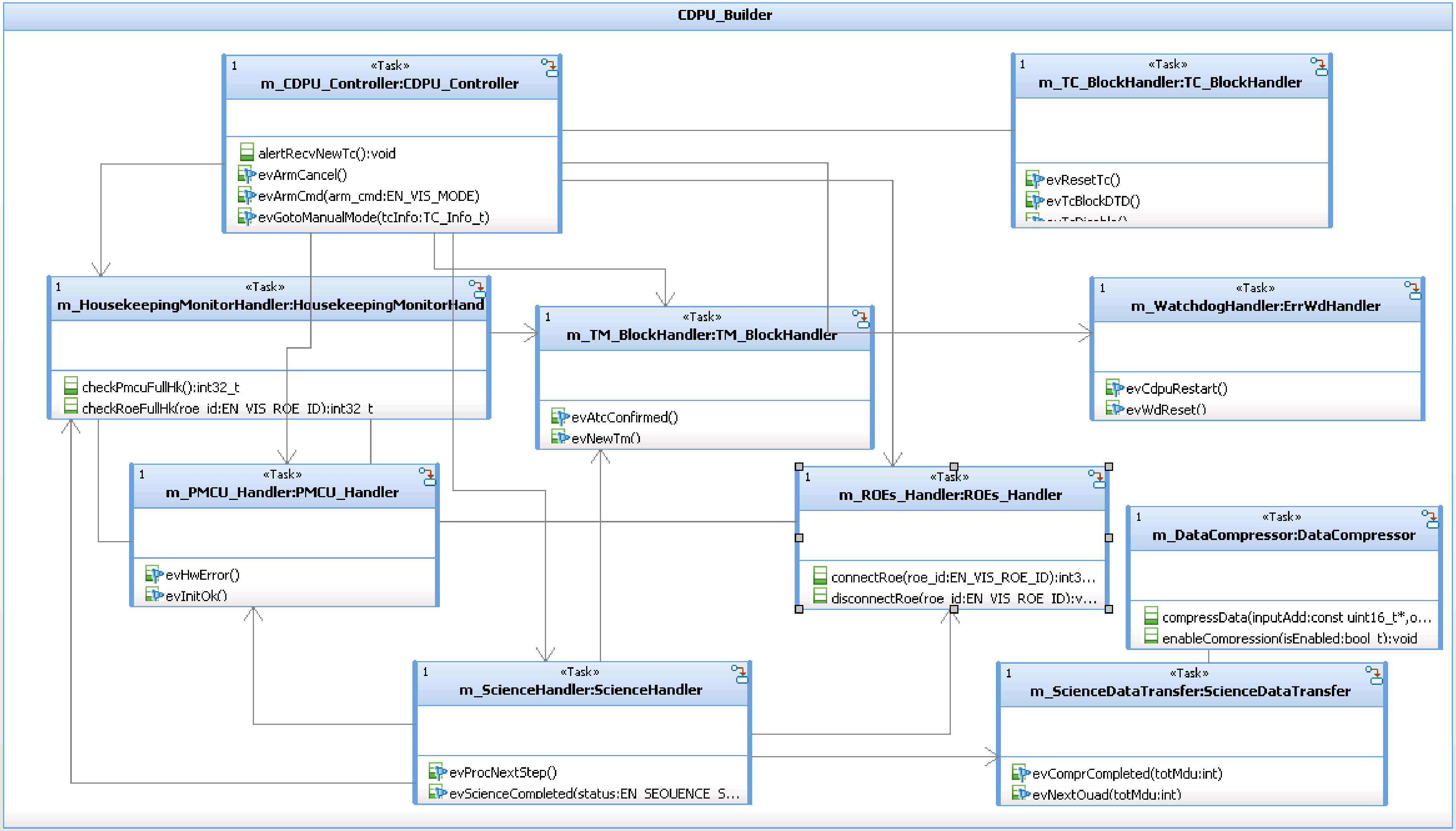
1 Spacewire link
(cold redundant)

EUCLID VIS ASW Tasks

- 10 Tasks (5 **P**eriodic and 5 **S**poradic) have been defined + 1 NAP task to idle the CPU when is not used
  - Controller(S): to process and execute all telecommand (except science telecommands)
  - The SVM TM and TC handler tasks (P): to handle the communication with the SVM on the MIL BUS 1553
  - Science Handler(S): to process and executed science sequence telecommands
  - Science Data Transfer(S): to handle buffered science data, pass data to compressor, transfer data to the external mass memory unit
  - Housekeeping and Monitoring handler (P): to read HK from all units and to monitor hard limits
  - PMCU Handler (P): to handle communication with the PMCU
  - ROE Handler (P): to handle communication with the 12 ROEs
  - Data Compressor (S): to compress data
  - Error and WD handler (P): to check and execute FDIR, reset the HW WD

- 2 extra tasks are creaed by the framework :
  - to handle the internal tick of the framework
  - to check if there is an event/timer-event to be processed by a task

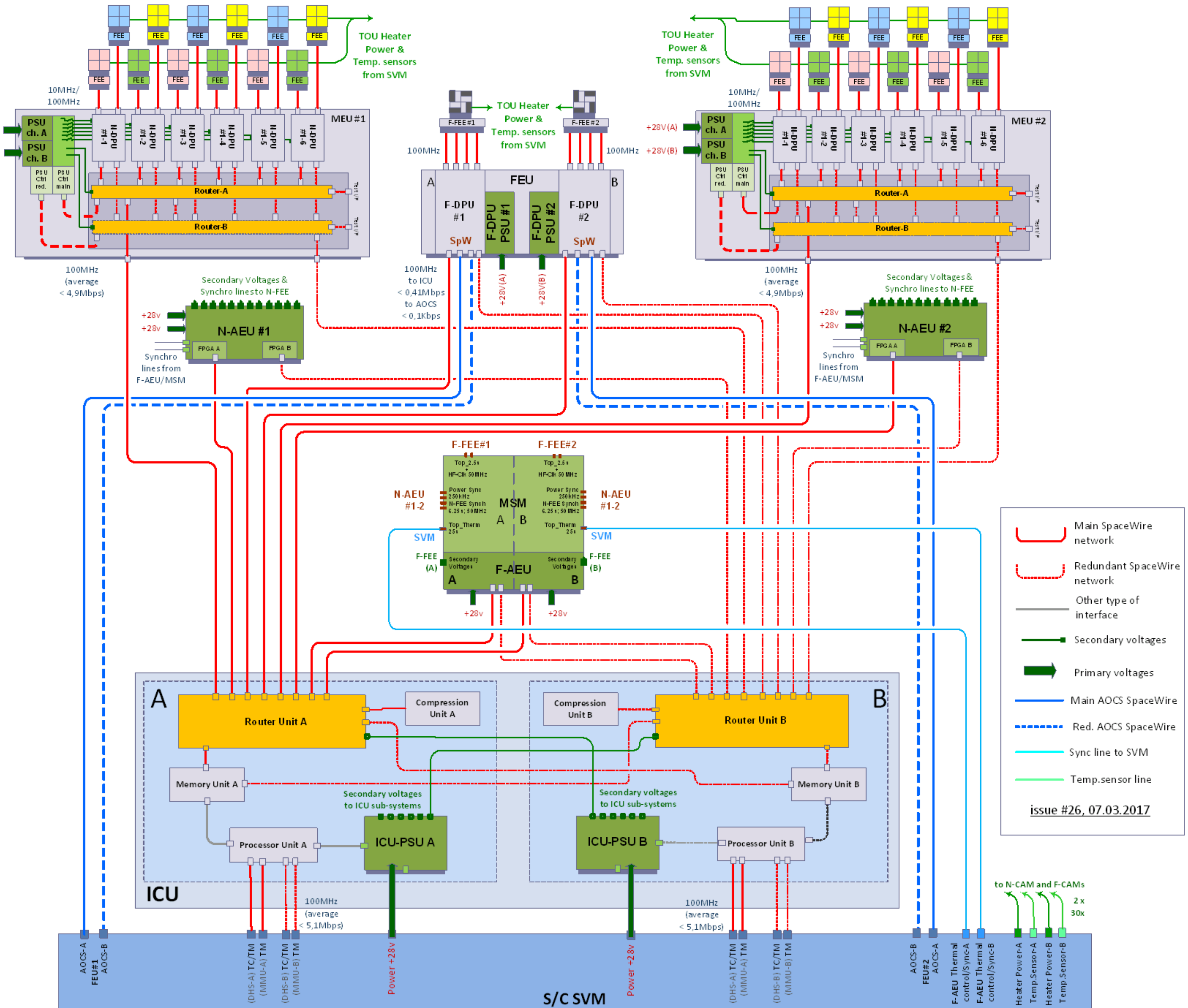
EUCLID VIS ASW Tasks(2)


CDPU_Builder
1
«Task»
m_CDPU_Controller:CDPU_Controller
alertRecvNewTc():void
evArmCancel()
evArmCmd(arm_cmd:EN_VIS_MODE)
evGotoManualMode(tcInfo:TC_Info_t)
1
«Task»
m_TC_BlockHandler:TC_BlockHandler
evResetTc()
evTcBlockDTD()
evTcDisable()
1
«Task»
m_HousekeepingMonitorHandler:HousekeepingMonitorHand
checkPmcuFullHk():int32_t
checkRoeFullHk(roe_id:EN_VIS_ROE_ID):int32_t
1
«Task»
m_TM_BlockHandler:TM_BlockHandler
evAtcConfirmed()
evNewTm()
1
«Task»
m_WatchdogHandler:ErrWdHandler
evCdpuRestart()
evWdReset()
1
«Task»
m_PMCU_Handler:PMCU_Handler
evHwError()
evInitOk()
1
«Task»
m_ROEs_Handler:ROEs_Handler
connectRoe(roe_id:EN_VIS_ROE_ID):int3...
disconnectRoe(roe_id:EN_VIS_ROE_ID):v...
1
«Task»
m_DataCompressor:DataCompressor
compressData(inputAdd:const uint16_t*,o...
enableCompression(isEnabled:bool_t):void
1
«Task»
m_ScienceHandler:ScienceHandler
evProcNextStep()
evScienceCompleted(status:EN_SEQUENCE_S...
1
«Task»
m_ScienceDataTransfer:ScienceDataTransfer
evComprCompleted(totMdu:int)
evNextQuad(totMdu:int)

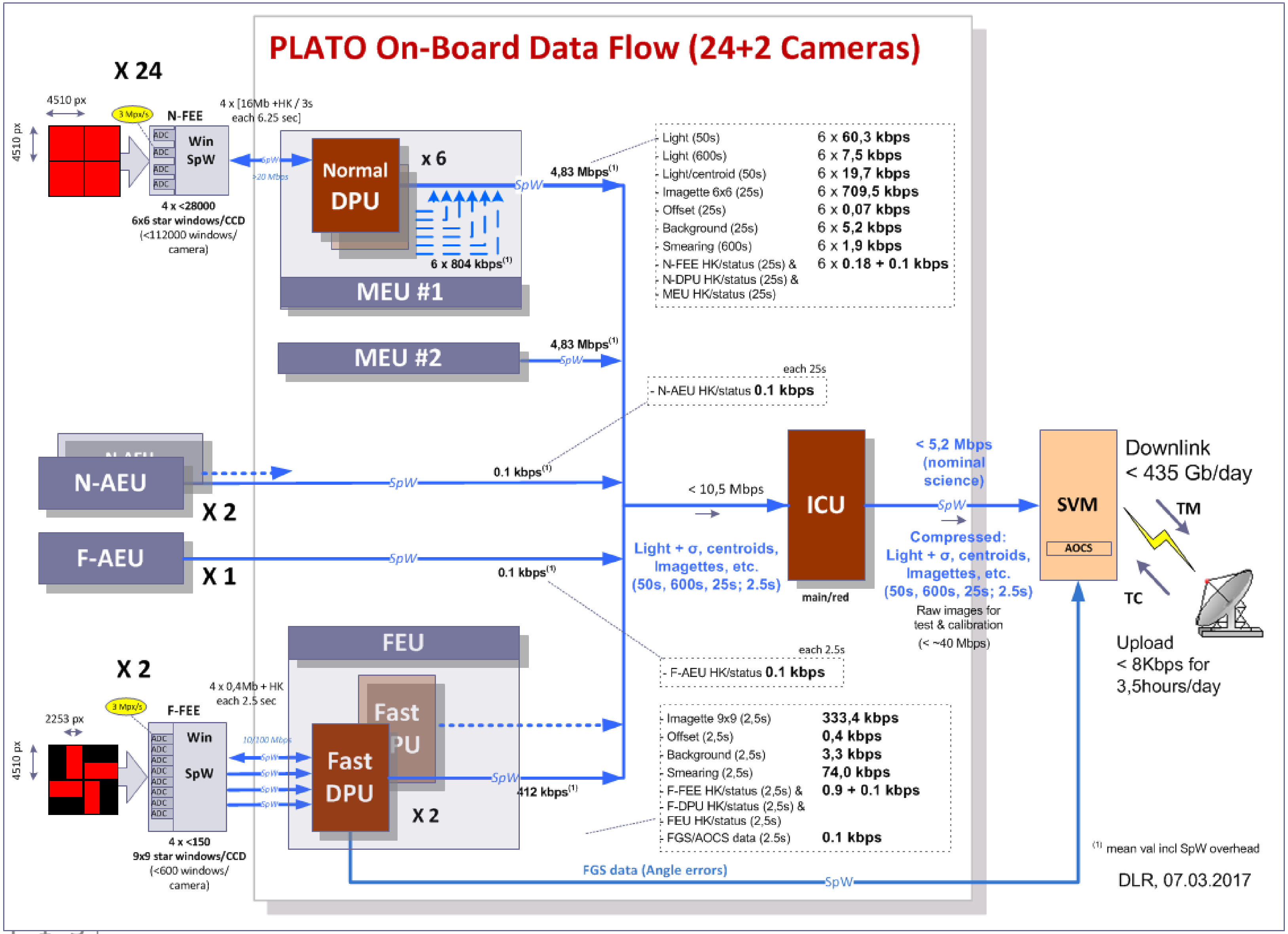
PLATO mission

- PLATO “Planetary Transits and Oscillations of stars” aims to characterise exoplanetary systems of all kinds and identification of suitable targets for future more detailed characterization
- Instrument consists of
  - 24 normal cameras
  - 2 fast cameras
  - each camera has 4 CCDs
  - 12 DPUs (N-DPU) for Normal cameras hosted in 2 MEUs
  - 2 DPU (F-DPU) for fast cameras and to provide information as Fine Guidance System (FGS) to SVM hosted in 1 FEU
  - 2 ICU in cold redundancy. Each one with 1 RDCU (router data compressor unit)

PLATO Payload Electr. Arch.



PLATO on-board Data Flow



- Light (50s)	6 x **60,3 kbps**
- Light (600s)	6 x **7,5 kbps**
- Light/centroid (50s)	6 x **19,7 kbps**
- Imagette 6x6 (25s)	6 x **709,5 kbps**
- Offset (25s)	6 x **0,07 kbps**
- Background (25s)	6 x **5,2 kbps**
- Smearing (600s)	6 x **1,9 kbps**
- N-FEE HK/status (25s) & - N-DPU HK/status (25s) & - MEU HK/status (25s)	6 x **0.18 + 0.1 kbps**

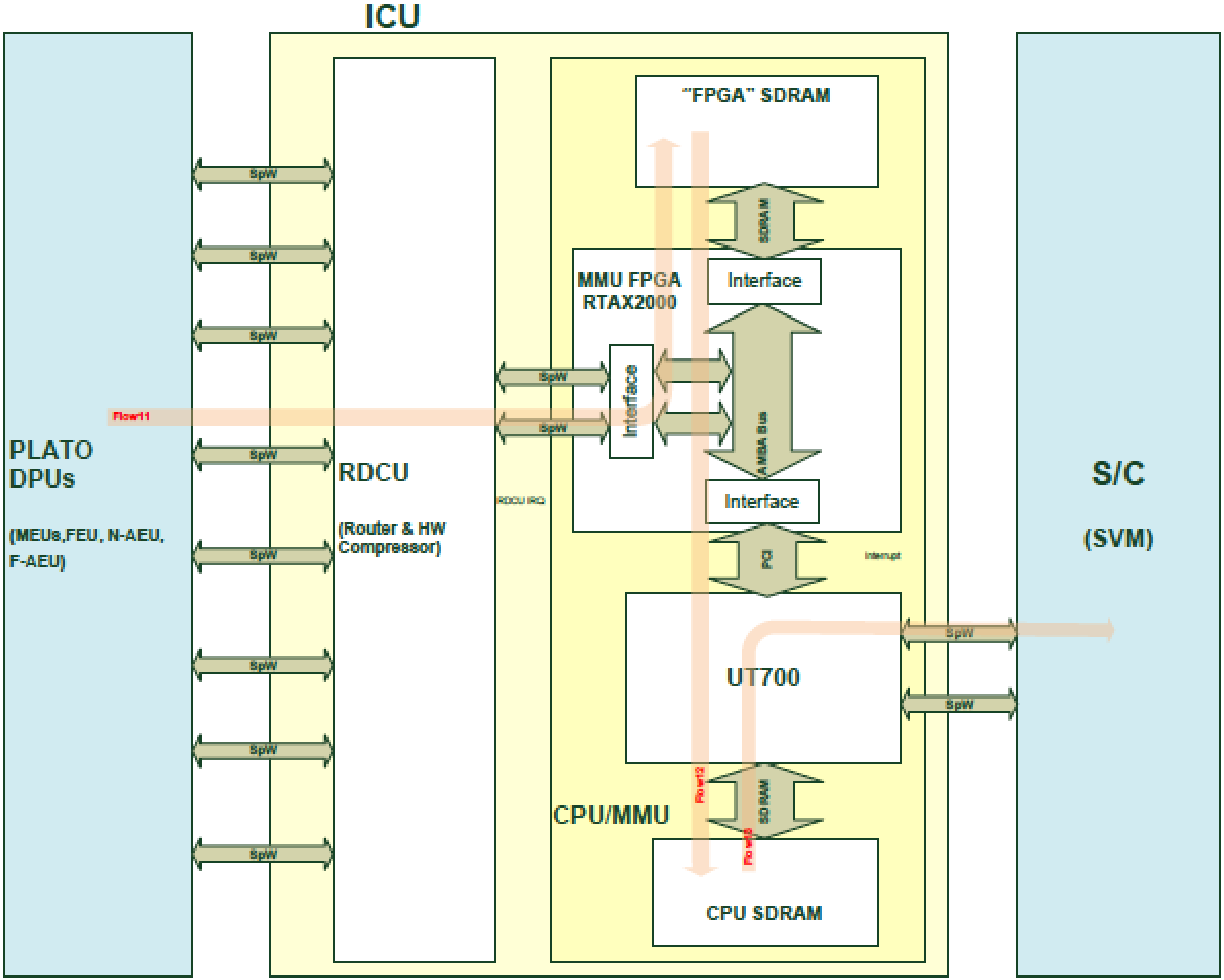
- N-AEU HK/status **0.1 kbps**

- F-AEU HK/status **0.1 kbps**

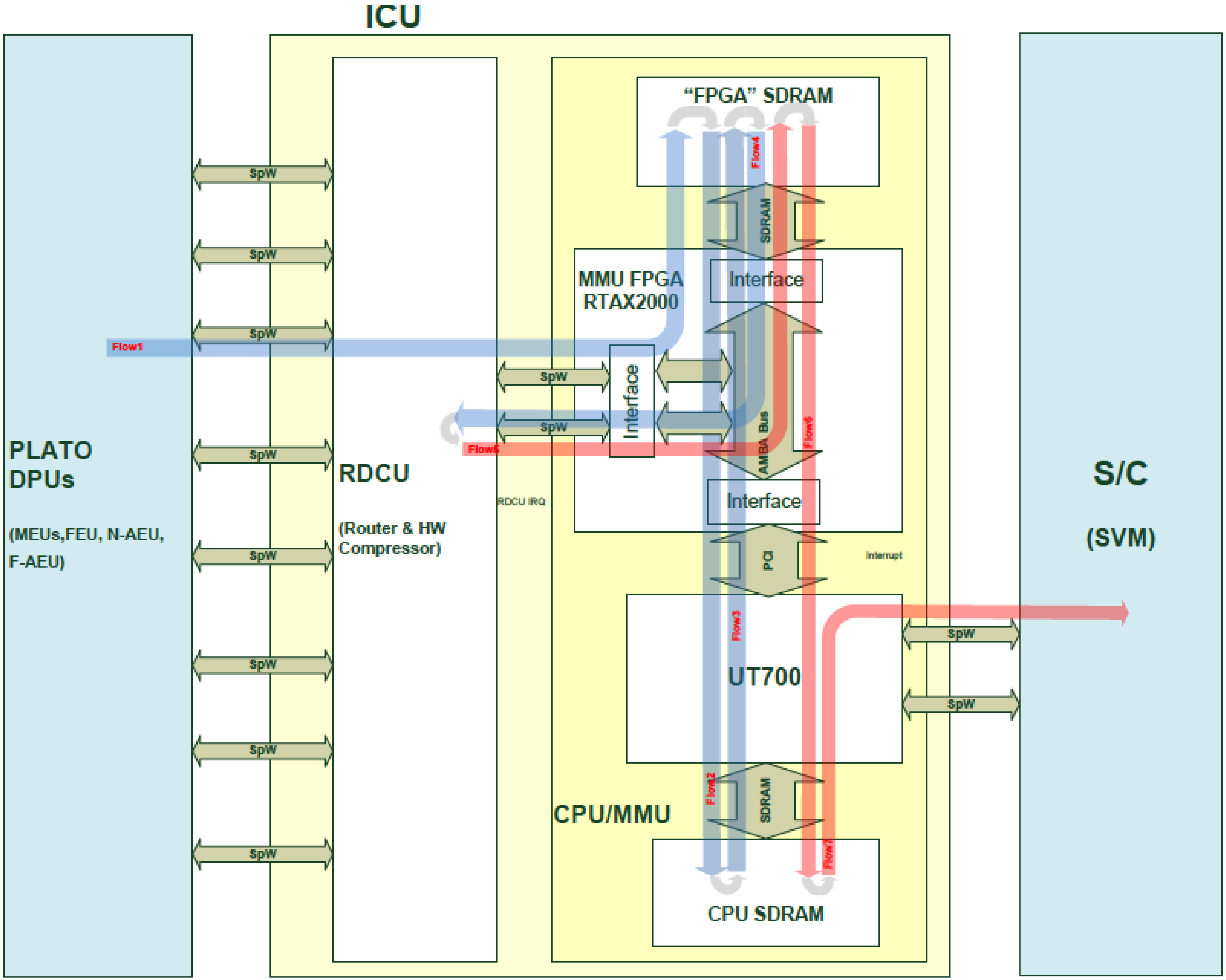
- Imagette 9x9 (2,5s)	**333,4 kbps**
- Offset (2,5s)	**0,4 kbps**
- Background (2,5s)	**3,3 kbps**
- Smearing (2,5s)	**74,0 kbps**
- F-FEE HK/status (2,5s) & - F-DPU HK/status (2,5s) & - FEU HK/status (2,5s)	**0.9 + 0.1 kbps**
- FGS/AOCS data (2.5s)	**0.1 kbps**

[1] mean val incl SpW overhead

HK data flow of RMAP units



TM Data Flow of CPTP/PUS packets



PUS Services comparison

EUCLID

- APID: 1 for the BSW , 1 for the ASW
- Limit set of PUS services:
  - PUS 1,3,5,6,8,9,17
  - Monolithic approach. Service 8 is used almost for everything

PLATO

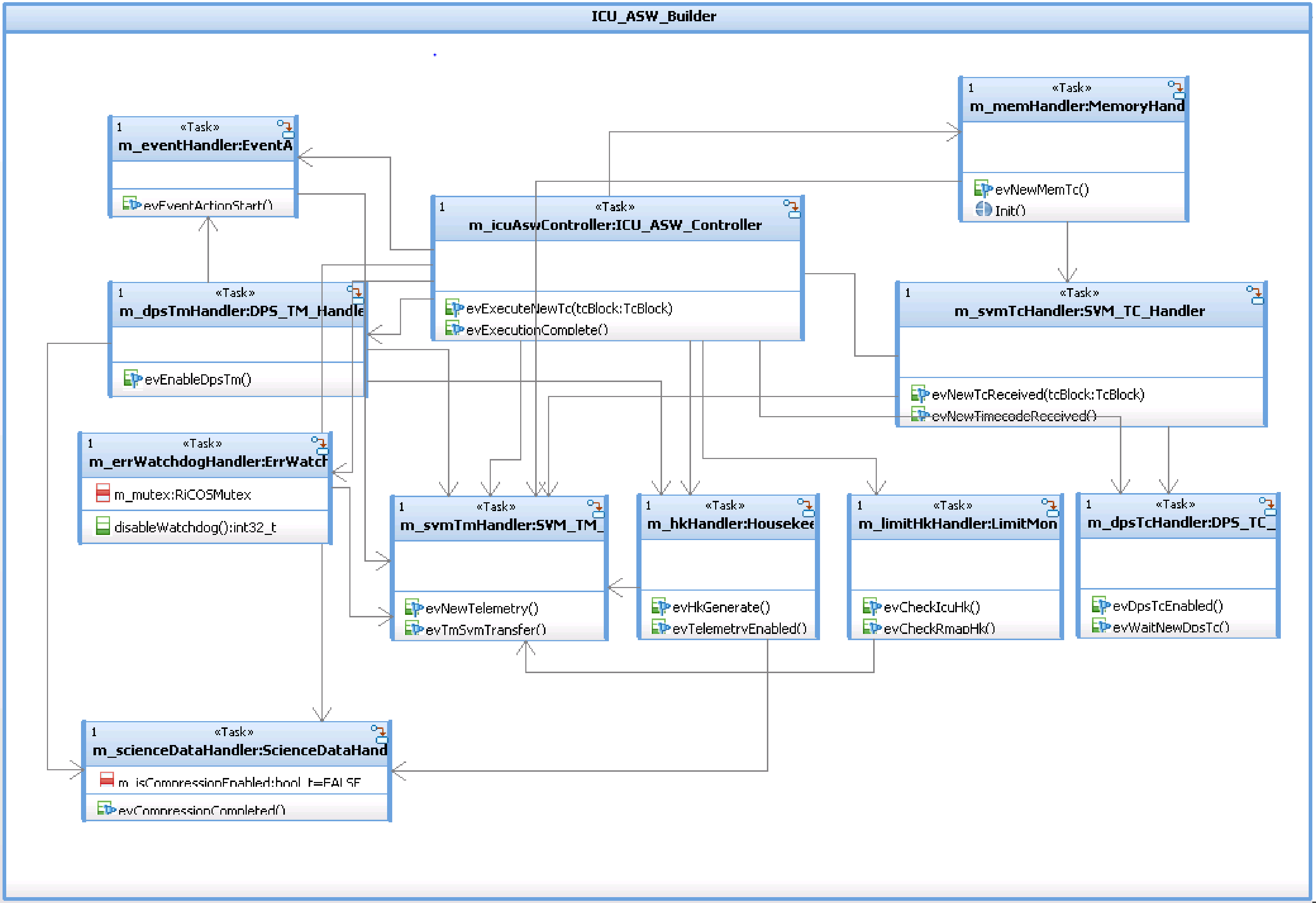
- APID: 1 for the BSW, 4 for the ASW. Moreover it has to handle directly 16 APIDs of RMAP units and indirectly >70 APIDs
- Many and many services:
  - 12 Basic service including OBCP→ PUS 1,2,3,5,6,9,12,14,18,17, 19,20
  - 6 Private service from 190 to 196
  - Common Private service 245 (Heartbeat)
  - 2 extra service for RMAP units (F-AEU and N-AEU)

PLATO ICU ASW Tasks

- 10 Tasks (8 **P**eriodic and 3 **S**poradic) have been defined
  - Controller(S): to process and execute all telecommand (except memory and the time sync telecommand)
  - The SVM TM and TC handler tasks (P): to handle the communication with the SVM on the SPW link (DHS and MMU)
  - Science Data Handler(P): to handle buffered science data, compress data, transfer data to the external mass memory unit
  - Housekeeping (P) to read HK from all units
  - Monitoring handler (P): to monitor hard limits
  - DPU TC and TM Handler(P): to process PUS TM coming from DPUs Data and to send TC
  - Memory TC Handler (S): to process memory telecommands (PUS 6)
  - Error and WD handler (P): to check and execute FDIR, reset the HW WD
  - Event Action Handler (S): execute action triggered by external events (PUS 5 events)
  - Idle task (P): scrub memory and idle CPU

- 2 extra tasks are created by the framework :
  - to handle the internal tick of the framework
  - to check if there is an event/timer-event to be processed by a task

PLATO ICU ASW Tasks(2)



ICU_ASW_Builder

1 «Task»
m_eventHandler:EventA
evEventActionStart()

1 «Task»
m_icuAswController:ICU_ASW_Controller
evExecuteNewTc(tcBlock:TcBlock)
evExecutionComplete()

1 «Task»
m_memHandler:MemoryHand
evNewMemTc()
Init()

1 «Task»
m_dpsTmHandler:DPS_TM_Handle
evEnableDpsTm()

1 «Task»
m_svmTcHandler:SVM_TC_Handler
evNewTcReceived(tcBlock:TcBlock)
evNewTimecodeReceived()

1 «Task»
m_errWatchdogHandler:ErrWatch
m_mutex:RiCOSMutex
disableWatchdog():int32_t

1 «Task»
m_svmTmHandler:SVM_TM_
evNewTelemetry()
evTmSvmTransfer()

1 «Task»
m_hkHandler:Housekee
evHkGenerate()
evTelemetryEnabled()

1 «Task»
m_limitHkHandler:LimitMon
evCheckIcuHk()
evCheckRmapHk()

1 «Task»
m_dpsTcHandler:DPS_TC_
evDpsTcEnabled()
evWaitNewDpsTc()

1 «Task»
m_scienceDataHandler:ScienceDataHand
m_isCompressionEnabled:bool_t=FALSE
evCompressionCompleted()

TEchnologies for Telescopes and Instrument control Software

Thank You!