FUSION FOR ENERGY

MARTe2 Users Meeting

History, objectives and overview of the framework

Andre' Neto, Filippo Sartori, G. De Tommasi*

May, 2019

*Many slides from the presentation a *Brief history of the MARTe framework (2013)*

Tokamak continuous development

Tokamak are experimental machines:

- the process is the experiment!

Tokamak development is part of the experiment.

The development stops when the machine is closed.

This meant a continuous development of control systems at all levels: functions, performance, reliability.

Reliability vs complexity

Controlling a Tokamak means managing 100s MW of power with associated risks of 100s M€ of investment.

To measure control relevant Tokamak quantities one needs to **combine 100s of sensors using complex non-linear iterative algorithms**.

Failure of sensors is common, algorithms non-convergence is likely too.

Lack of resources

Tokamak control suffers from lack of resources:

Lack of commissioning time: Squeezed between delay of hardware deliveries and rigidness of start of experimental campaigns.

Low financing: Tokamak hardware is expensive and typically absorbs more resources than forecast

PPCC Control systems in the mid 90s

PPCC: Plasma Position and Current Control – plasma magnetic control

- Two of the main systems run at JET by the Plasma Operations Group:
  - Shape Controller (SC) C code deployed on a VxWorks/VME/Motorola68k platform (still is!)
  - Vertical Stabilization System (VS) C code deployed on 4 Texas Instruments DSPs

Main issues

- The code was tailored at the specific platform
- Lack of modularity
- Different software solutions to interface with the JET software infrastructure
  (pre-pulse system configuration, post-pulse data collection,. . .)

A new framework for RT applications

2001/2002 Opportunity for an upgrade

- Revamping of the SC was planned in order to add the eXtreme Shape Controller algorithm (XSC)
- Within the Plasma Operations Group, decision to move to a common framework for the development of real-time applications
    - With-in the organization new (and competing) integration solutions (frameworks) start to flourish

Aims (User Requirements)

- **Standardize** the development of real-time applications (style, language, …)
- Increase the code **reusability**
- **Separate** (as much as possible) the **user** application from the software required to interface with the **plant** infrastructure
- **Reduce** the time needed for **commissioning**

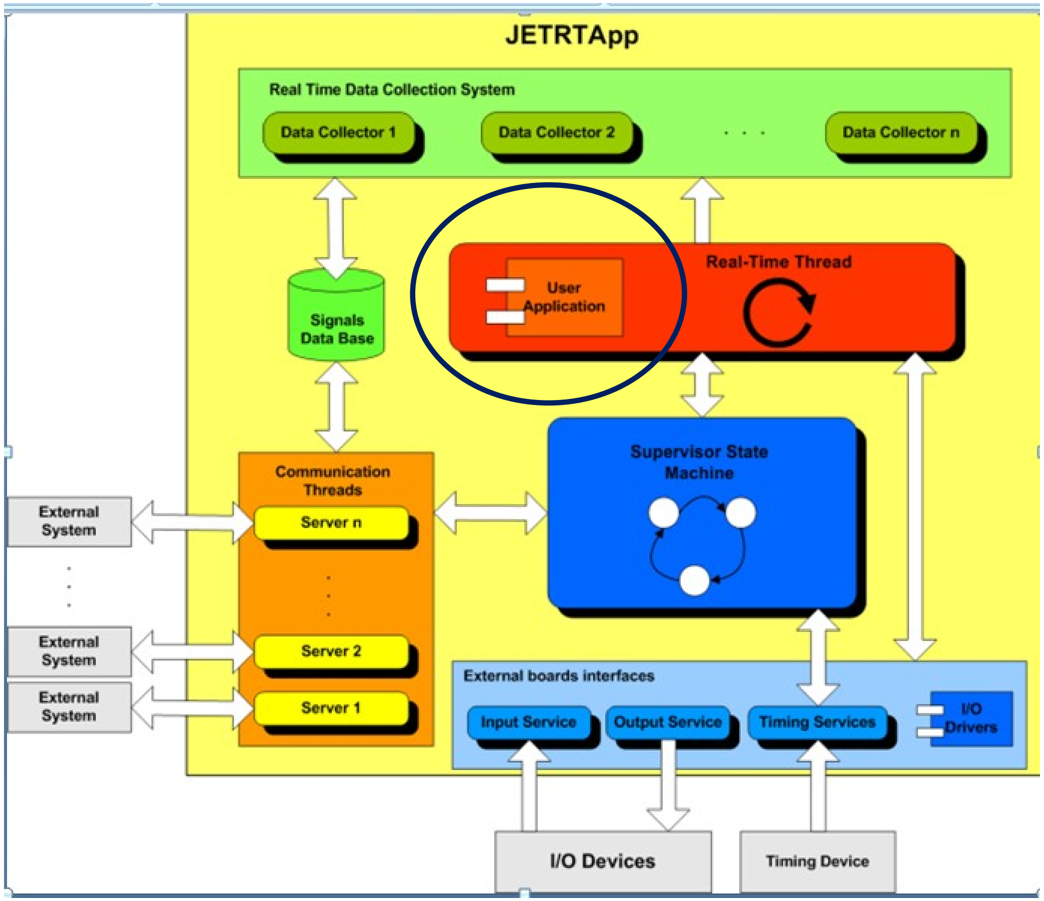
A new framework for RT applications – High level requirements

High Level System Requirements

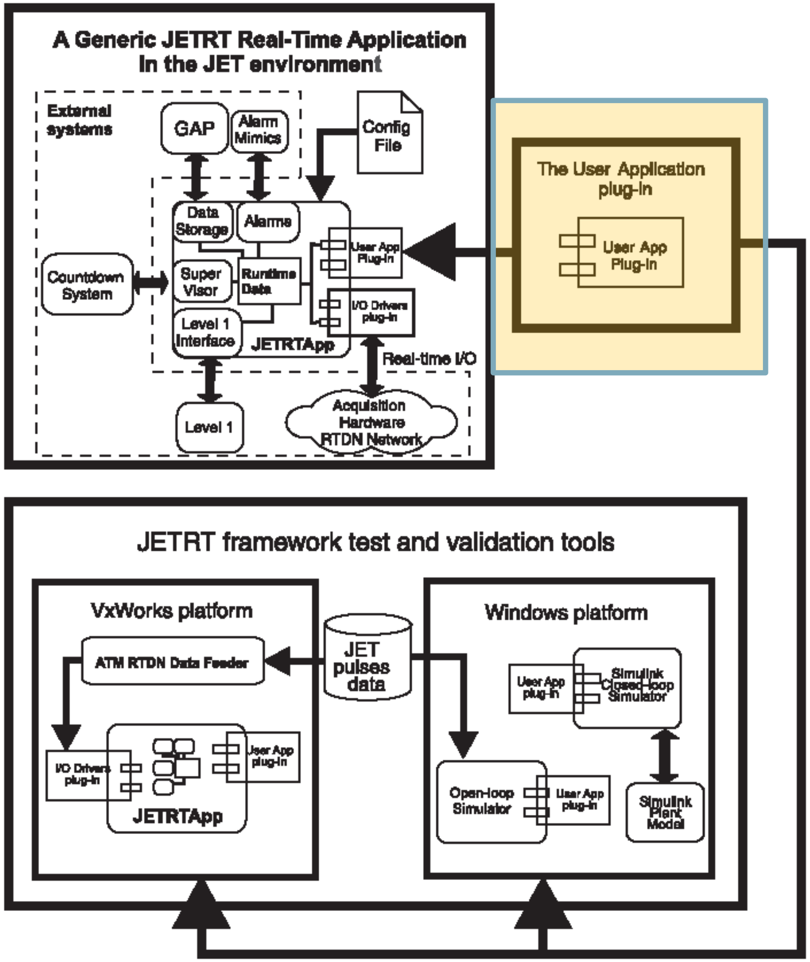
- Portable (multi-OS and multi-platform)
- Modular – the user application would have been easily plugged into an executor of real-time application
- Written in C++ (object oriented approach) – for embedded (2001!)
- Allow scientists (process experts) to abstract from the plant interfaces

JETRT

- JETRT framework developed in 2002/2003 to deploy the XSC
- Based on the cross-platform BaseLib library (developed within the POG group)



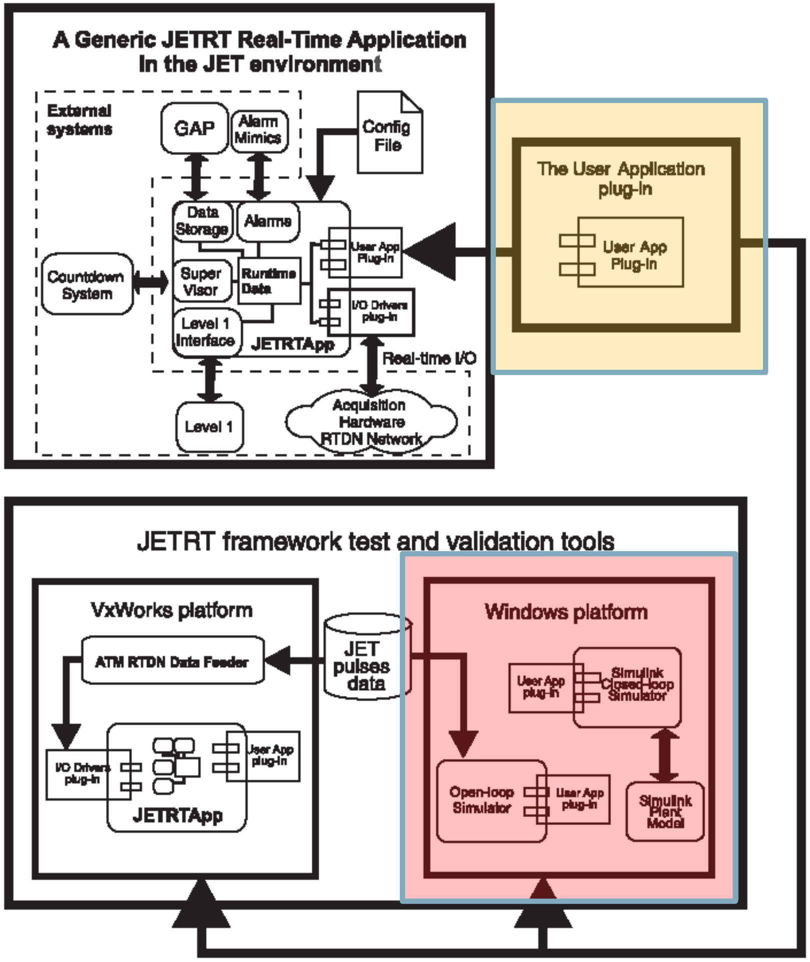
Use Application plug-in



Use Application plug-in

- The user supplied code

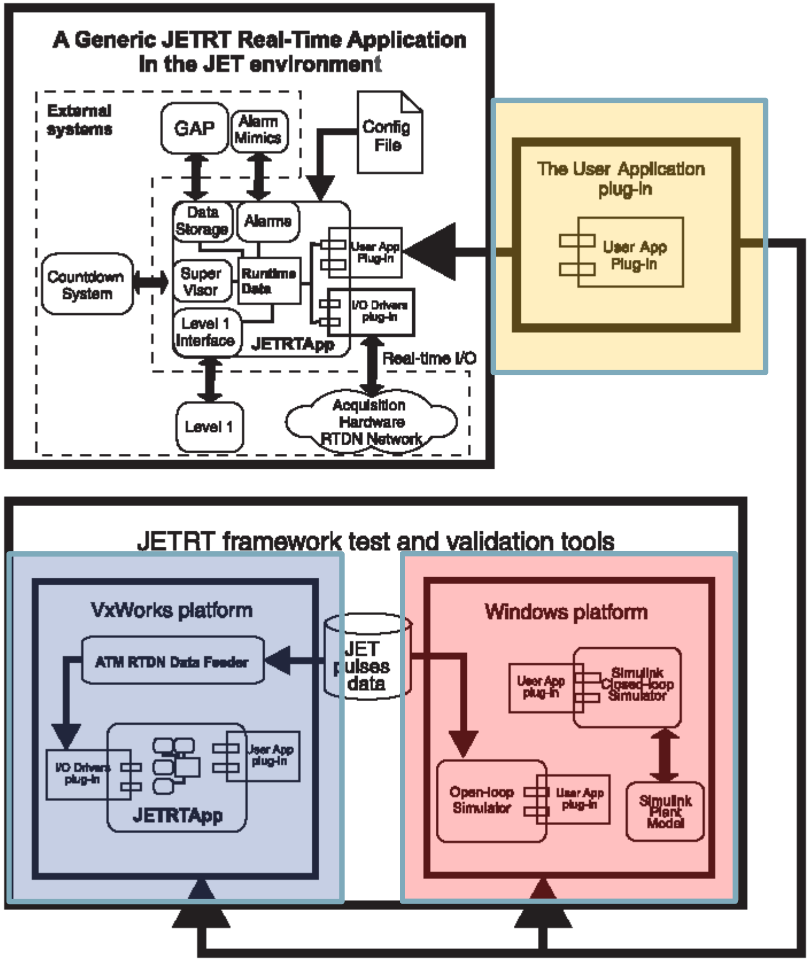
Use Application plug-in



Use Application plug-in

- The user supplied code
- Perform offline validation against a plant model

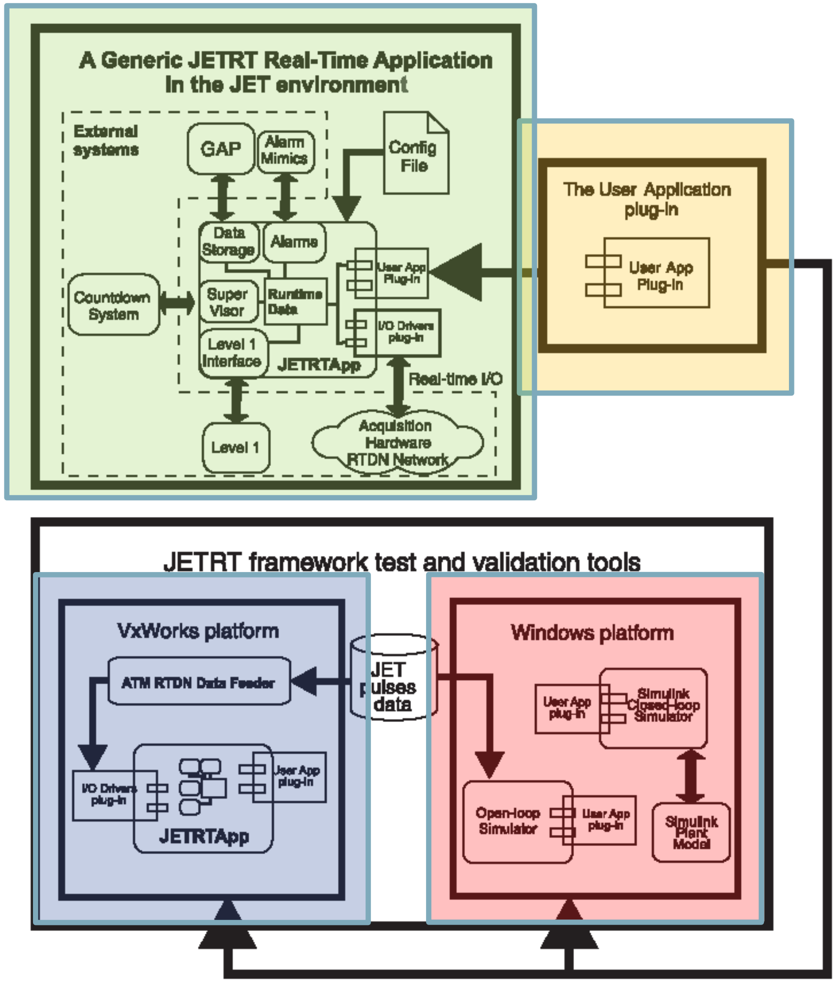
Use Application plug-in



Use Application plug-in

- The user supplied code
- Perform offline validation against a plant model
- Perform real-time validation with hardware-in-the-loop

Use Application plug-in



Use Application plug-in

- The user supplied code
- Perform offline validation against a plant model
- Perform real-time validation with hardware-in-the-loop
- Run the real-time system on the plant

XSC

XSC + JETRT

- New SC (including the XSC) was deployed on a 400 MHz G4 PowerPC running VxWorks
- 2 ms control loop (but it can run at 1 ms)
- **Portability** enabled
    - Exhaustive debug of both the JETRT framework and the XSC **offline**, on a Windows platform
    - Testing in the lab, with a mockup of the JET timing system and of the I/O
    - Only 3 days of testing on the plant were needed for the commissioning of the new system

MARTe1 – the origins

Fall 2004/Winter 2005

MARTe1 – the limitations of JETRT

Limitations of JETRT

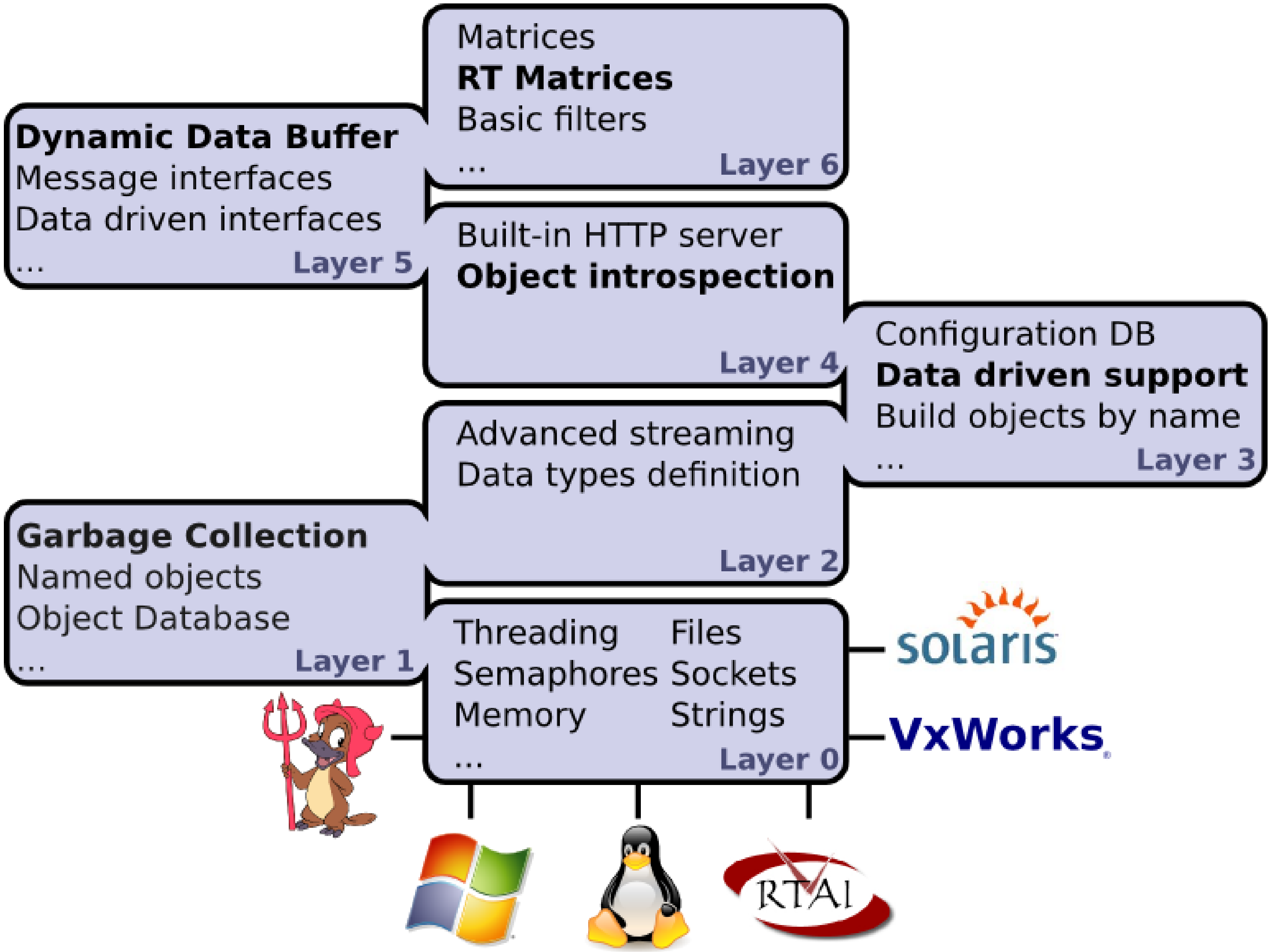
- No real separation between the user application and the plant-interface software
  - Some *soft-dependencies* on JET libraries and JET operational philosophy
- In 2011, about 1 ppm was needed to include the Current Limit Avoidance system in SC!
  - Lack of confidence driven by difficulties on testing and porting to newer platforms and machines

MARTe1 – a new framework was born

Aims (User Requirements)

- Clear boundary between algorithms, hardware and with the operational environments (portability)
    - **Run exactly the same application as in the RTOS**
- Data driven
- Instropection
- Without sacrificing RT
- Promote simulink-like way of describing the problem

BaseLib2 – the multiplatform support library



Modularity (GAMs)

GAM: Generic Application Module

Define boundaries

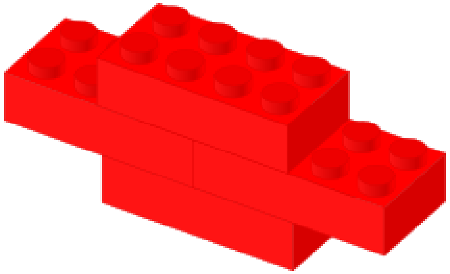
- Algorithms and hardware don't mix!
- Modules do only what they advertise
- No interdependence or a priori knowledge

Generic by design

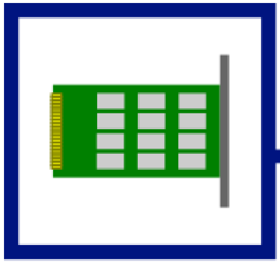
- Same goals, same module
- Reusability and maintainability

Simulation

- Replace actuators and plants with models
- Keep all the other modules untouched



Common GAMs!

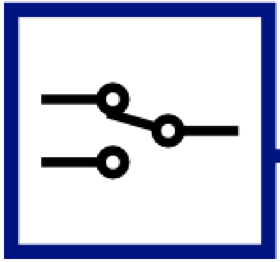


Hardware I/O

Persistence

Algorithms

Debug



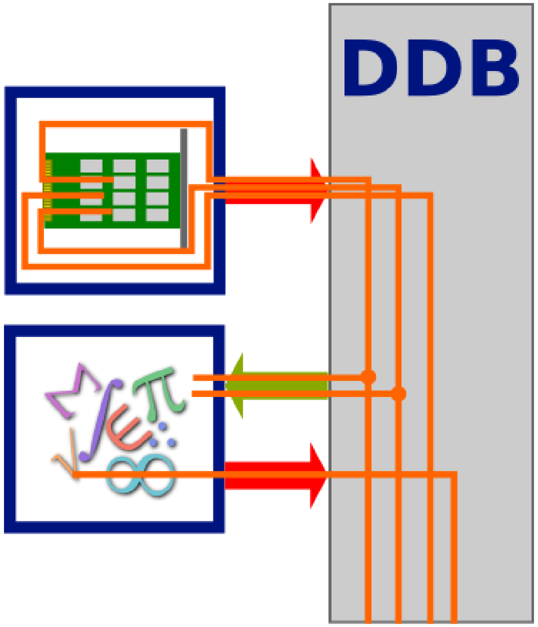
Decision taking

Information

Sharing data between GAMs

DDB: Dynamic Data Buffer

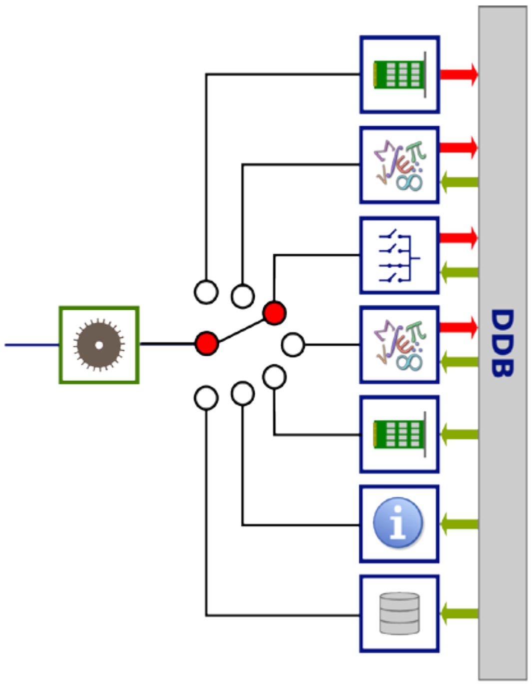
- GAMs share data through a memory bus
- MARTe guarantees coherency between requested and produced signals
- Set of GAMs allow to stream data to different MARTe systems



Executing GAMs

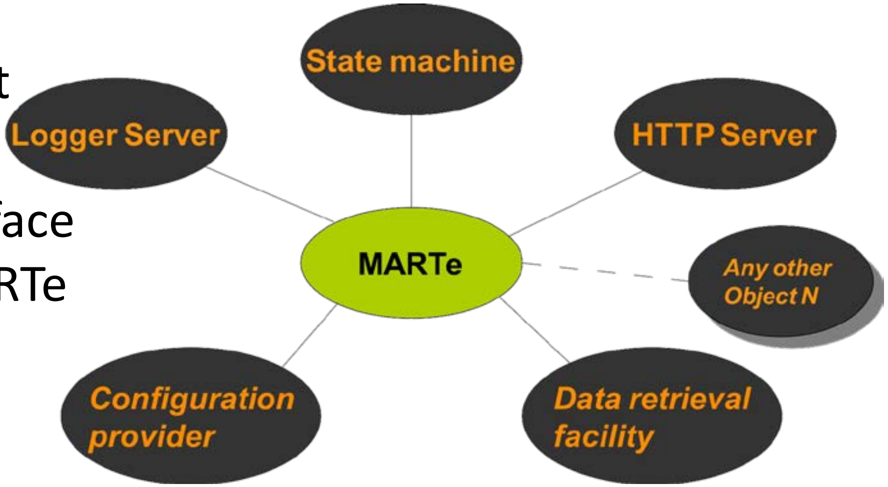
RealTimeThread

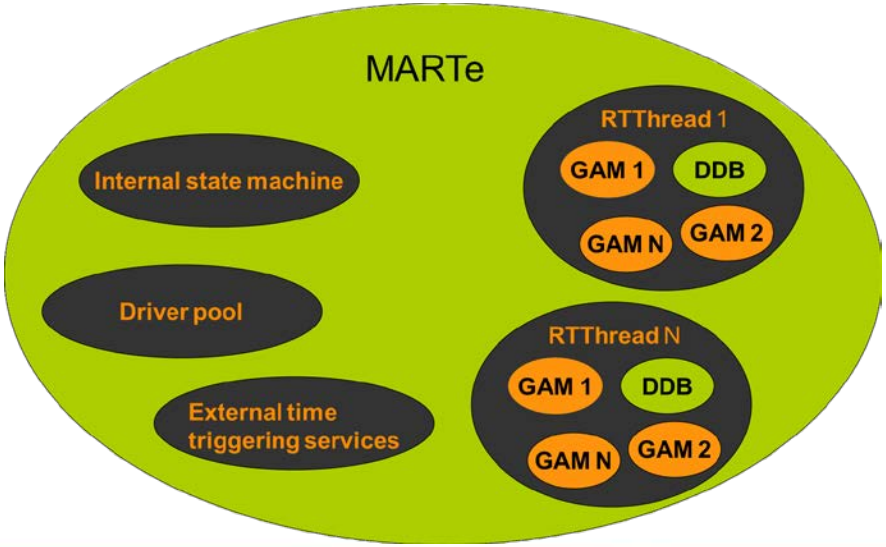
- Sequentially executes GAMs
- Works as micro-scheduler
- Can be allocated to specific CPUs
- Keeps accurate information about execution times
- Requires an external time and triggering mechanism
- Multiple RTThreads can run in parallel
- Synchronisation
    - Synchronous to hardware
    - Asynchronous
        - Get latest available value
        - Verify acceptable latency (sample too late?)
    - ADC, network, time card, another loop, ...



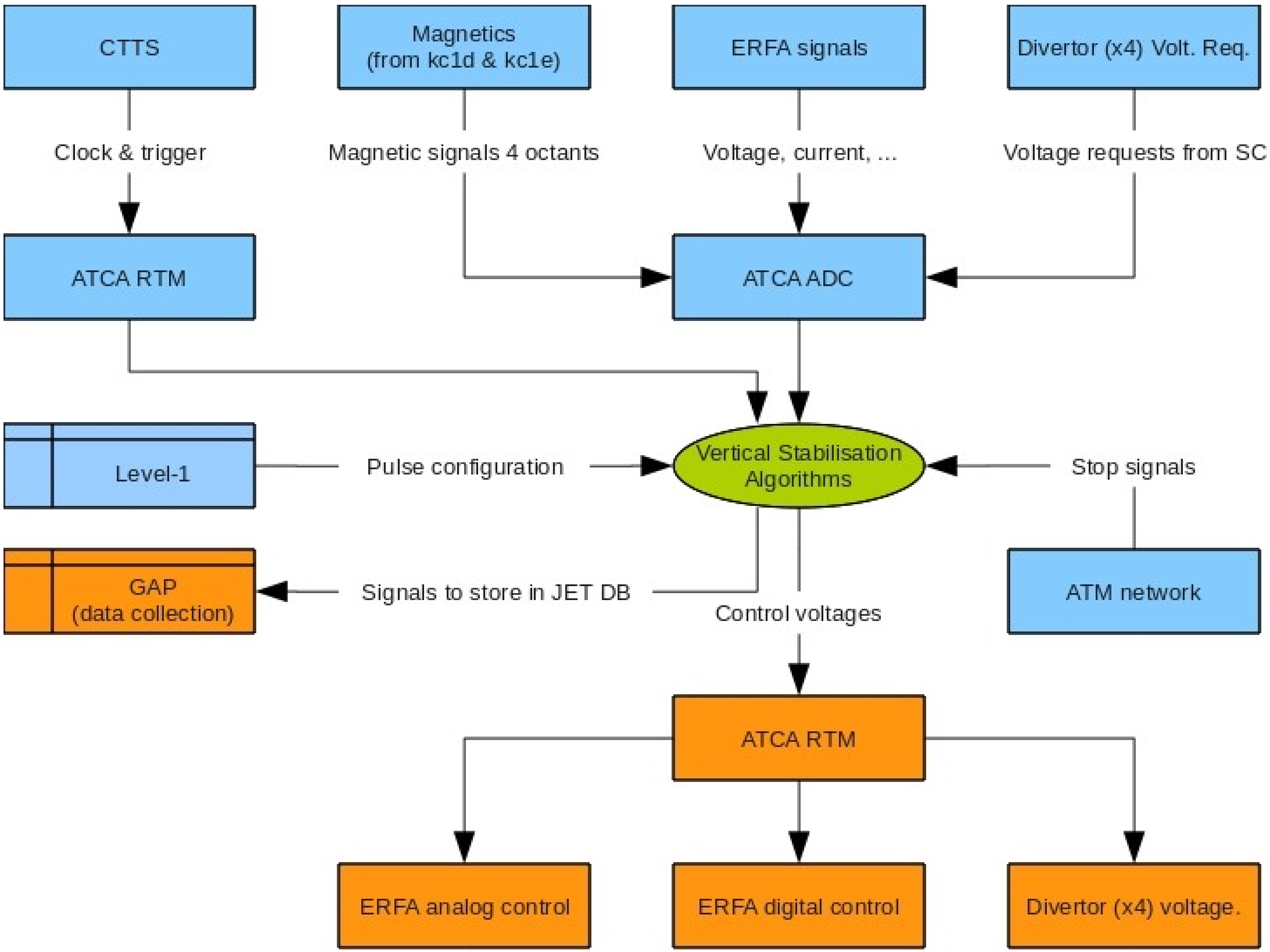
MARTe1 was already interface agnostic

- No predefined GUI
- No predefined high level protocols
- Allowed to easily deploy in many different environments and machines
- Translation components implement interface between site specific environment and MARTe

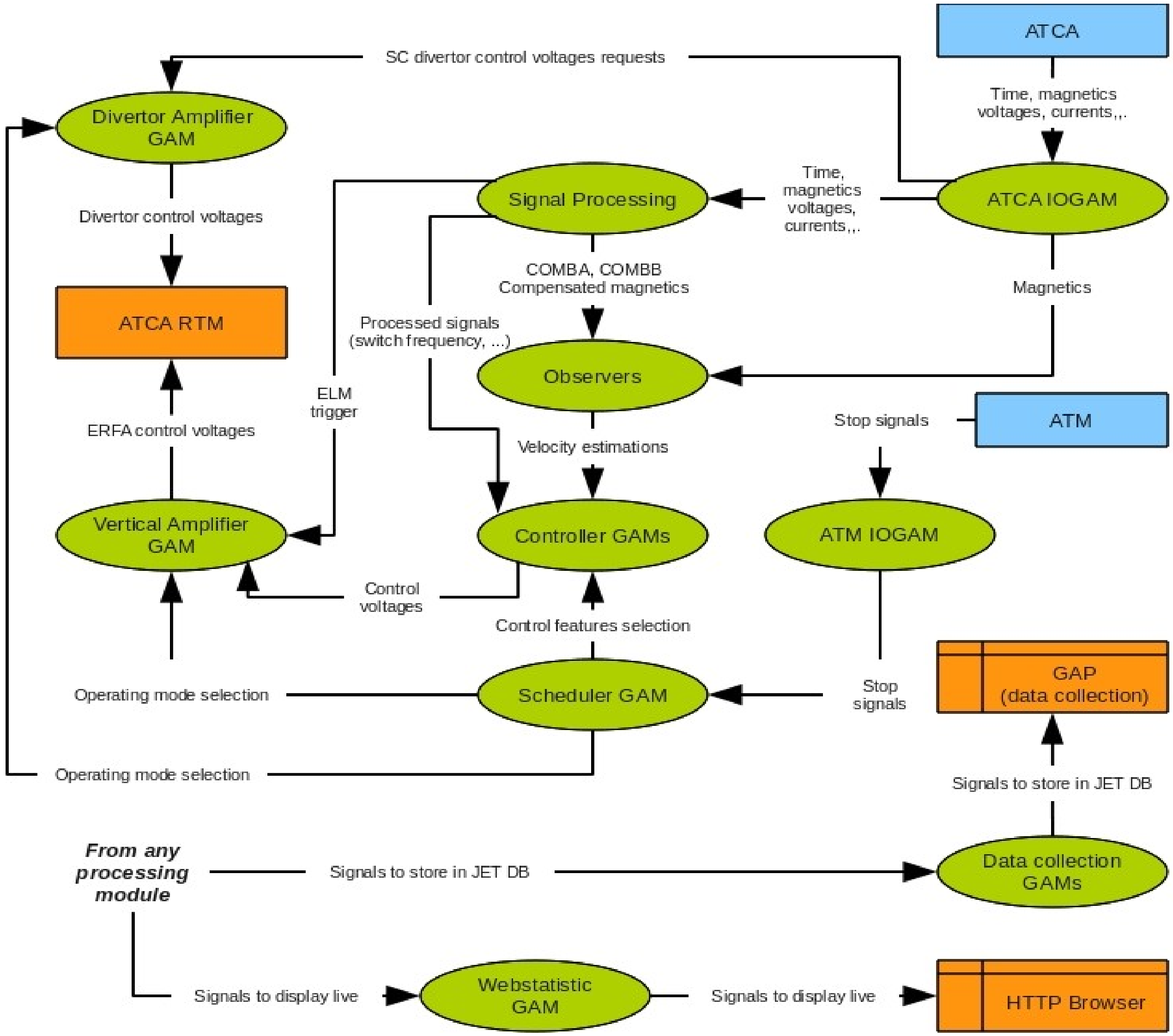




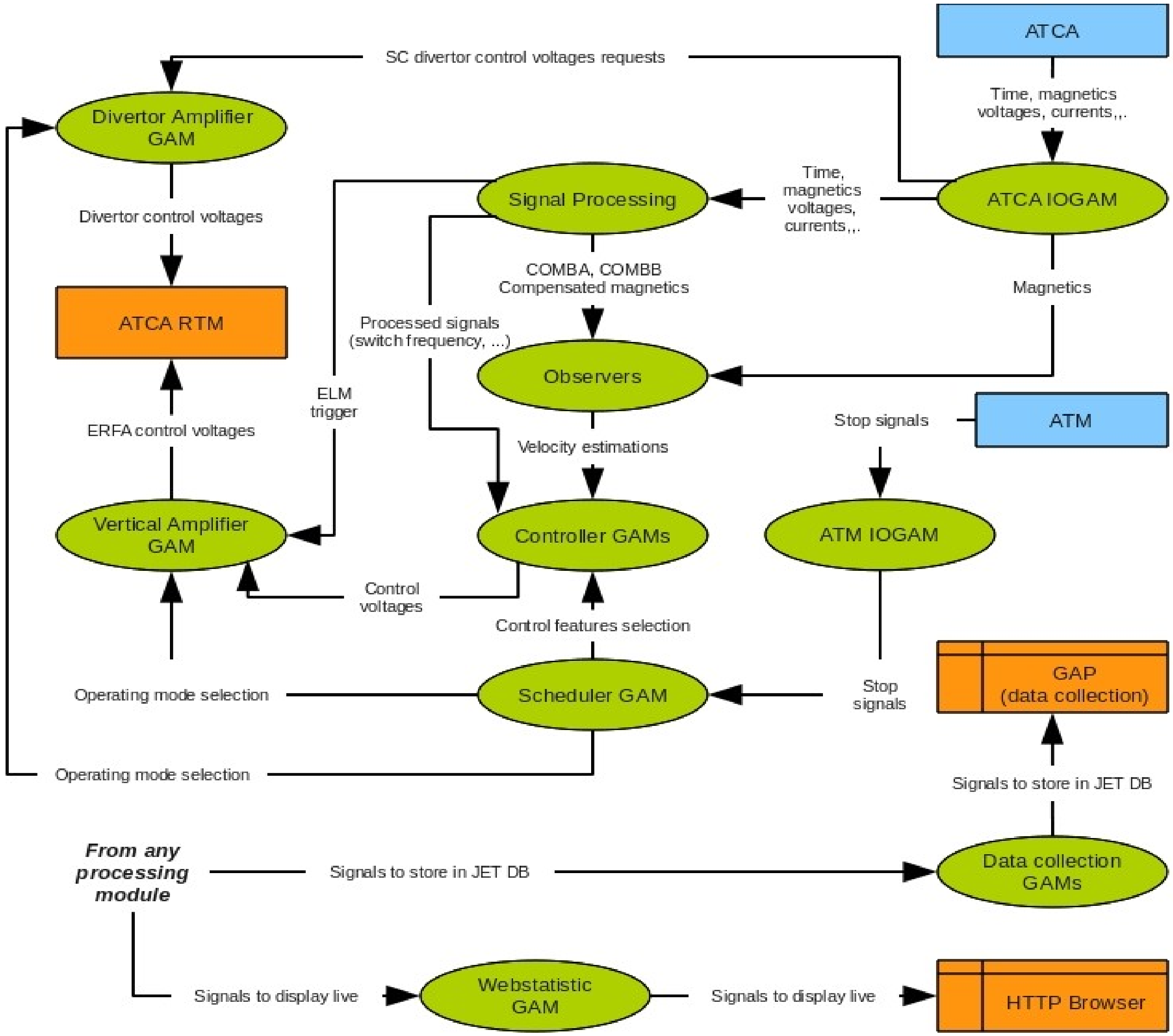
VS5: a case study



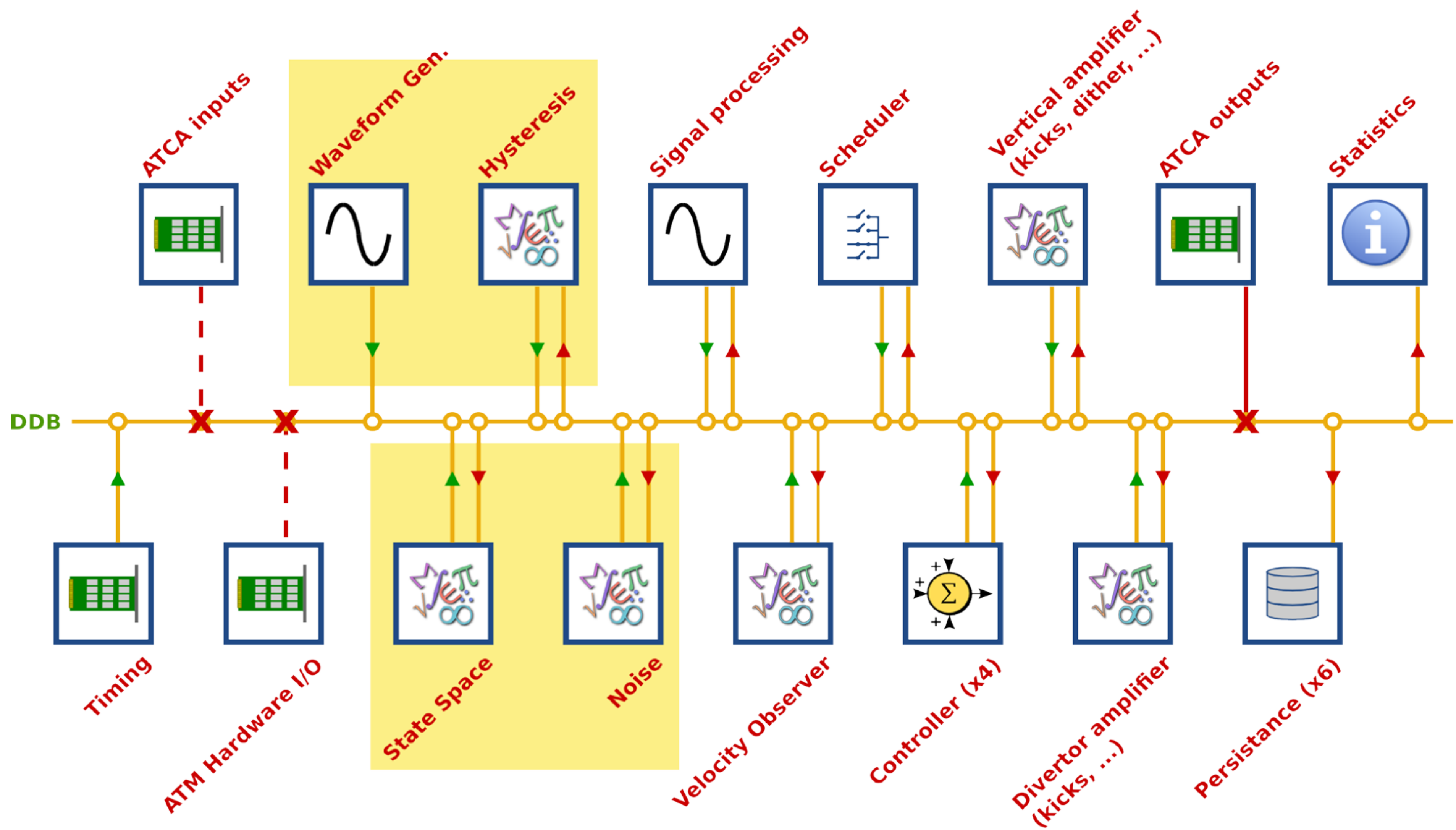
VS5: a case study



VS5: a case study



VS5: a case study



2008 – VS5 deployed

Modular

Data driven

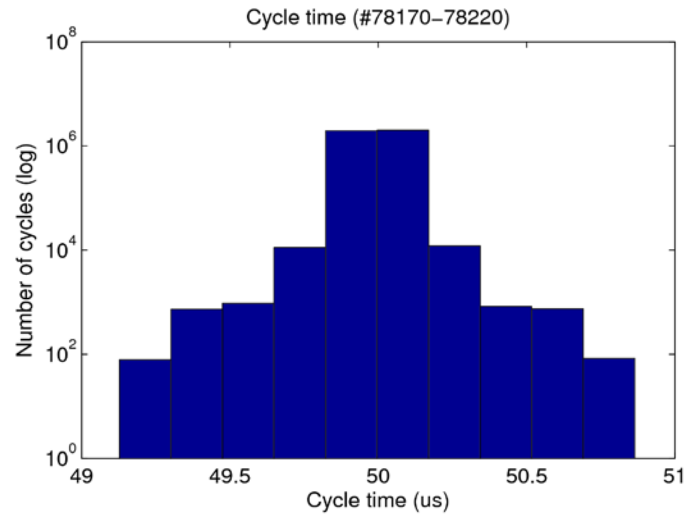
Introspection

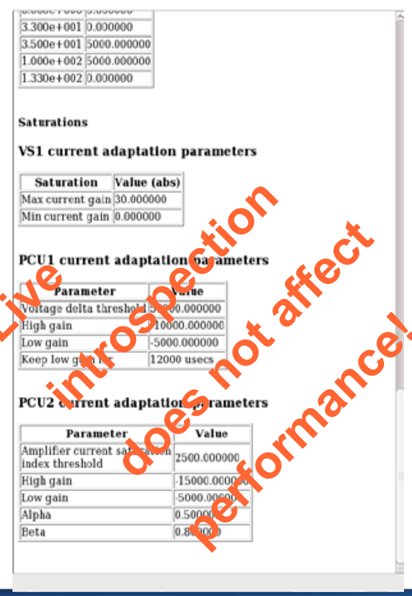
Reliable

Performance

Low jitter

VS Achieved:
50 ± 0.10 μs
(max jitter of 0.80 μs)





3.300e+001	0.000000
3.500e+001	5000.000000
1.000e+002	5000.000000
1.330e+002	0.000000

Saturations

VS1 current adaptation parameters

Saturation	Value (abs)
Max current gain	30.000000
Min current gain	0.000000

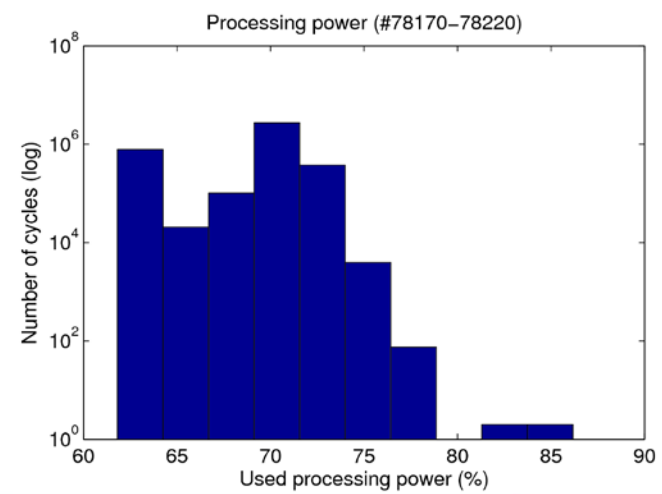
PCU1 current adaptation parameters

Parameter	Value
Voltage delta threshold	[illegible]0.000000
High gain	10000.000000
Low gain	-5000.000000
Keep low gain for	12000 usecs

PCU2 current adaptation parameters

Parameter	Value
Amplifier current saturation index threshold	2500.000000
High gain	-15000.000000
Low gain	-5000.000000
Alpha	0.500000
Beta	0.800000

Live introspection does not affect performance!



MARTe1: what is it being used for?

- Real-time control of tokamak systems
  - Many required to the operate the machine

JET
- Vertical Stability
- Real-time Protection Sequencer
- Vessel Thermal map
- Walls
- Equinox
- Beta-Li
- As part of post-processing codes

COMPASS
- Shape control
- Vertical Stability

ISTTOK
- Integrated plasma control

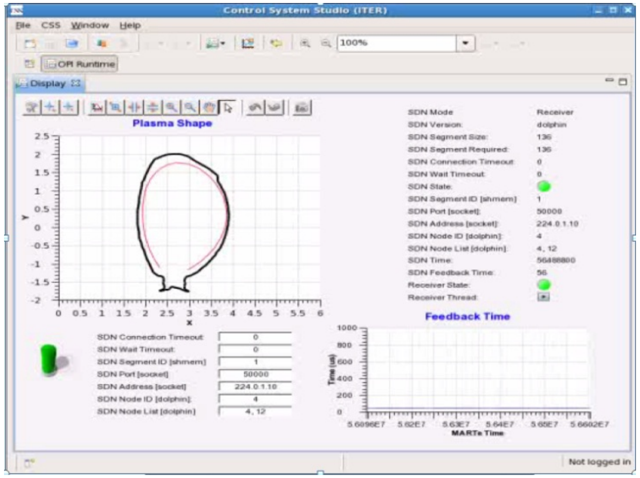
FTU
- Integrated plasma control
- Additional heating control

RFX
- Performance studies of the framework

KSTAR
- *Density control

ITER
- ECRH**
- Diagnostics**

F4E
- Falcon test facility**
- IFMIF**



*Evaluation only
**MARTe2

Why a new MARTe project?

- Fast controller prototype project
  - Integration of fast plant systems in ITER
- Development of ITER specific integration components
- Imposes the implementation of a Quality Assurance (QA) strategy that is appropriate for ITER

Development and upgrading of framework components will be distributed to different teams

F4E internal resources	IO resources	Academic institutions	Industrial suppliers

Guarantee consistency of

Implementation of coding and application of coding standards	Documentation	Testing standards

Safely integrate contributions from a large and heterogeneous development community

Manage changes to the configuration items and baselines

Continuous Integration

SonarQube
C++ Code Quality Inspection
Initially: CppCheck
Future: PC-lint (FlexeLint)

Googletest
Coverage Tests
Gcovr

Jenkins
Git Plugin
Cobertura

Version Control System

GIT

Eclipse + EGit

MARTe What and Why

FUSION FOR ENERGY

Tokamak Control System

- Rapid development
- Low budget
- High reliability
- High complexity

MARTe has been developed in this environment and offers:

- Modularity & Configurability → rapid dev
  → low budget
- Recycle & Standardize → reliability
- Modularity & Code Integration → complexity

Thank you for your attention

Follow us on:

www.f4e.europa.eu

www.twitter.com/fusionforenergy

www.youtube.com/fusionforenergy

www.linkedin.com/company/fusion-for-energy

www.flickr.com/photos/fusionforenergy