MARTe Framework

Middleware for RT Control Development

André Neto*±, F. Sartori,
D. Alves, A. Barbalace,
L . Boncagni, G. De Tommasi,
G. Manduchi, R. Vitelli,
D.F. Valcárcel, L. Zabeo and
EFDA-JET PPCC contributors

*Instituto de Plasmas e Fusão Nuclear
Instituto Superior Técnico
Lisbon, Portugal
http://www.ipfn.ist.utl.pt

Framework important functions

- Provides **development** and **execution** environment for control systems
- Defines a way of designing/developing
  - Limits what you can do to what is needed!
  - Reduces mistakes
- Provides **standard interfaces** to outside world
- Facilitates test and **commissioning**
- Ensures and monitors **real-time**

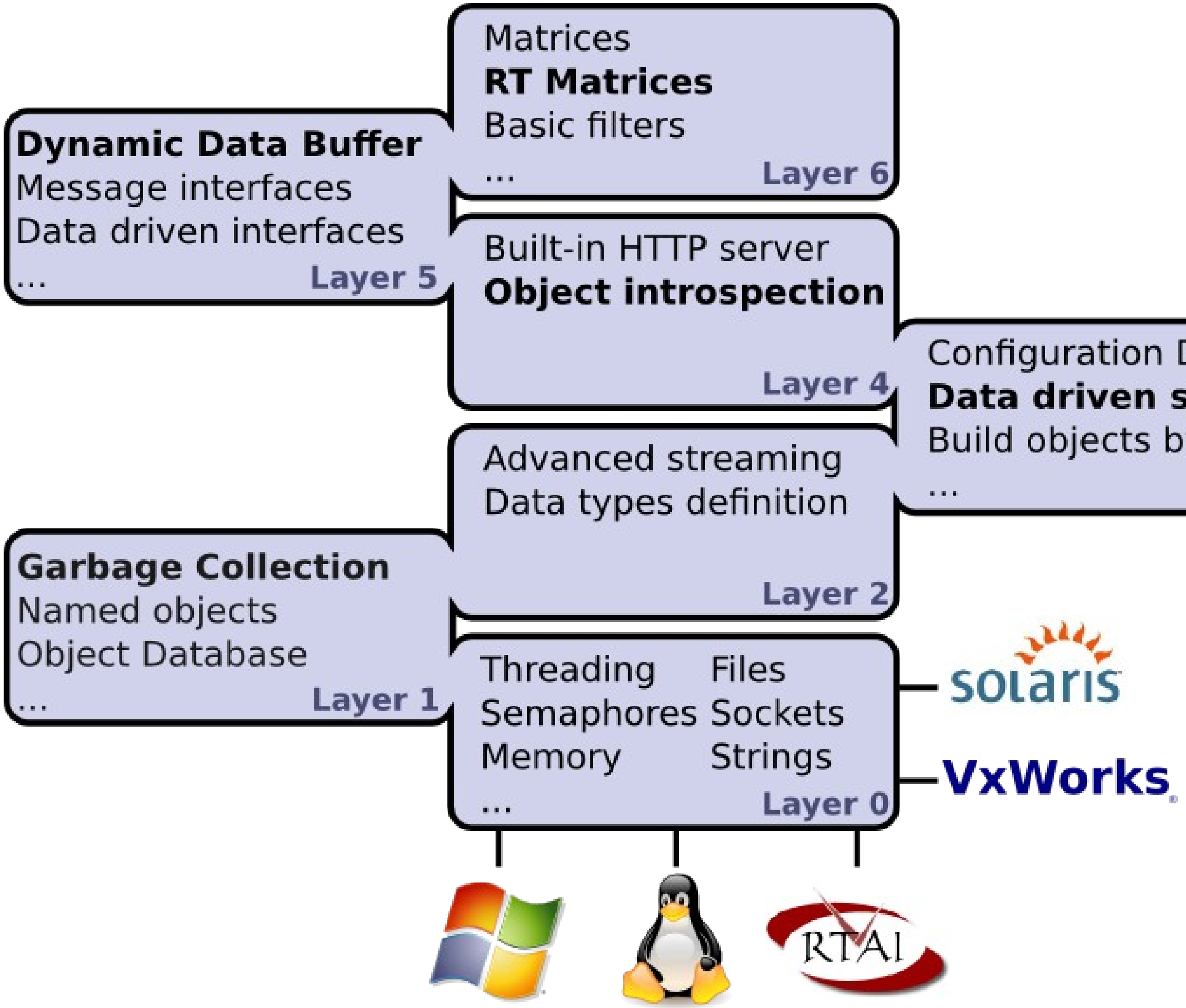
Main ideas

- Multi-platform C++ middleware
- Simulink-like way of describing the problem
- Modular
  - Clear boundary between algorithms, hardware interaction and system configuration
  - Reusability and maintainability
  - Simulation
  - Minimise constraints with the operational environments (portability)
- Data driven
- Provide live introspection tools
  - Without sacrificing RT

Multi-platform?

- **Why?**
  - **Debug** and develop in non RT targets
  - Eases the debugging process
  - Usually better developing environment
  - Debugger
  - IDE
- **How?**
  - Provide an **abstraction** layer/library which solves all the specificities of a given OS
  - Optimize code here
- **Possible?**
  - Yes, runs in Linux, Linux+RTAI, VxWorks, Solaris, MS Windows and Mac OS X

BaseLib2 – support library



Data driven components

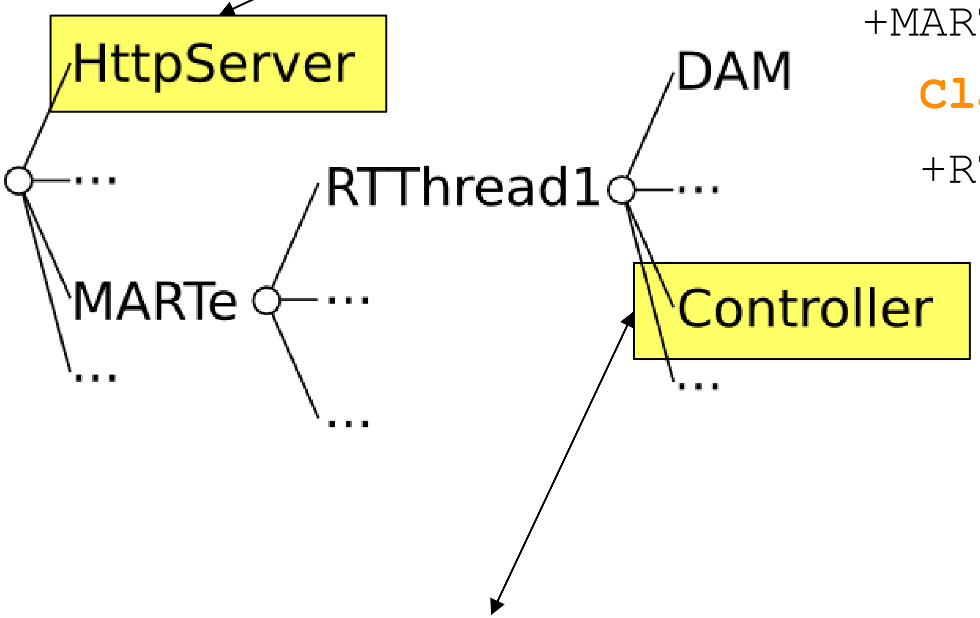
- Define **common** language
  - As **simple** as possible
  - But complete
  - Human understandable configuration
  - Should provide built-in **validation**
  - Should provide a clear way of expressing the problem
- Components are expected to be parsed only once per configuration request
  - Avoid unpleasant **surprises**

Object Configuration

- Structured syntax
- Similar to XML
- Classes are automatically created
- Configuration is validated by the created object
- Asserting and parsing functions available

```
+HttpServer = {
  Class = HttpService
  Port = 8084
}
...
+Control = {
  Class = ControlGAM
    Controller = {
      NoPlasmaVelocityGain = 0.0
      NoPlasmaCurrentGain = 40.0
      IPWaveform = {
        Times          = {0 120}
        Amplitudes = {0.5 0.5}
        Rounding     = 50
      }
...
```

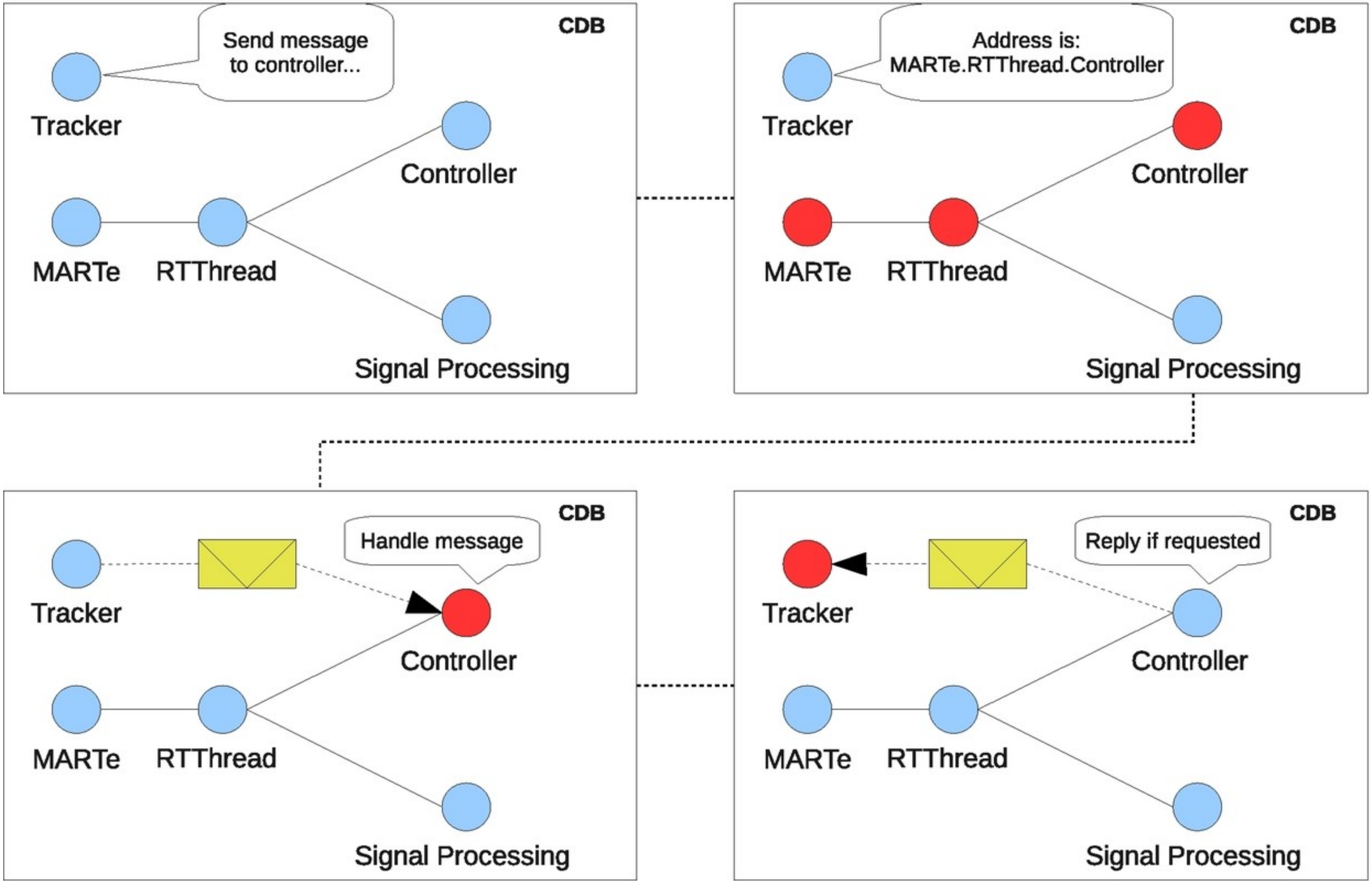
Configuration DB




```
+HttpServer = {
  Class = HttpService
  Port = 8084
}
+MARTe = {
  Class = MARTeContainer
  +RTThread1 = {
    Class = RealTimeThread
      +Controller = {
        NoPlasmaCurrentGain = 40.0
        IPWaveform = {
        Times            = {0 120}
        Amplitudes = {0.5 0.5}
    }
...
```

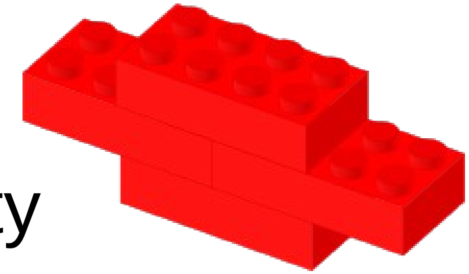
MARTe.RTThread1.Controller

Message mechanism



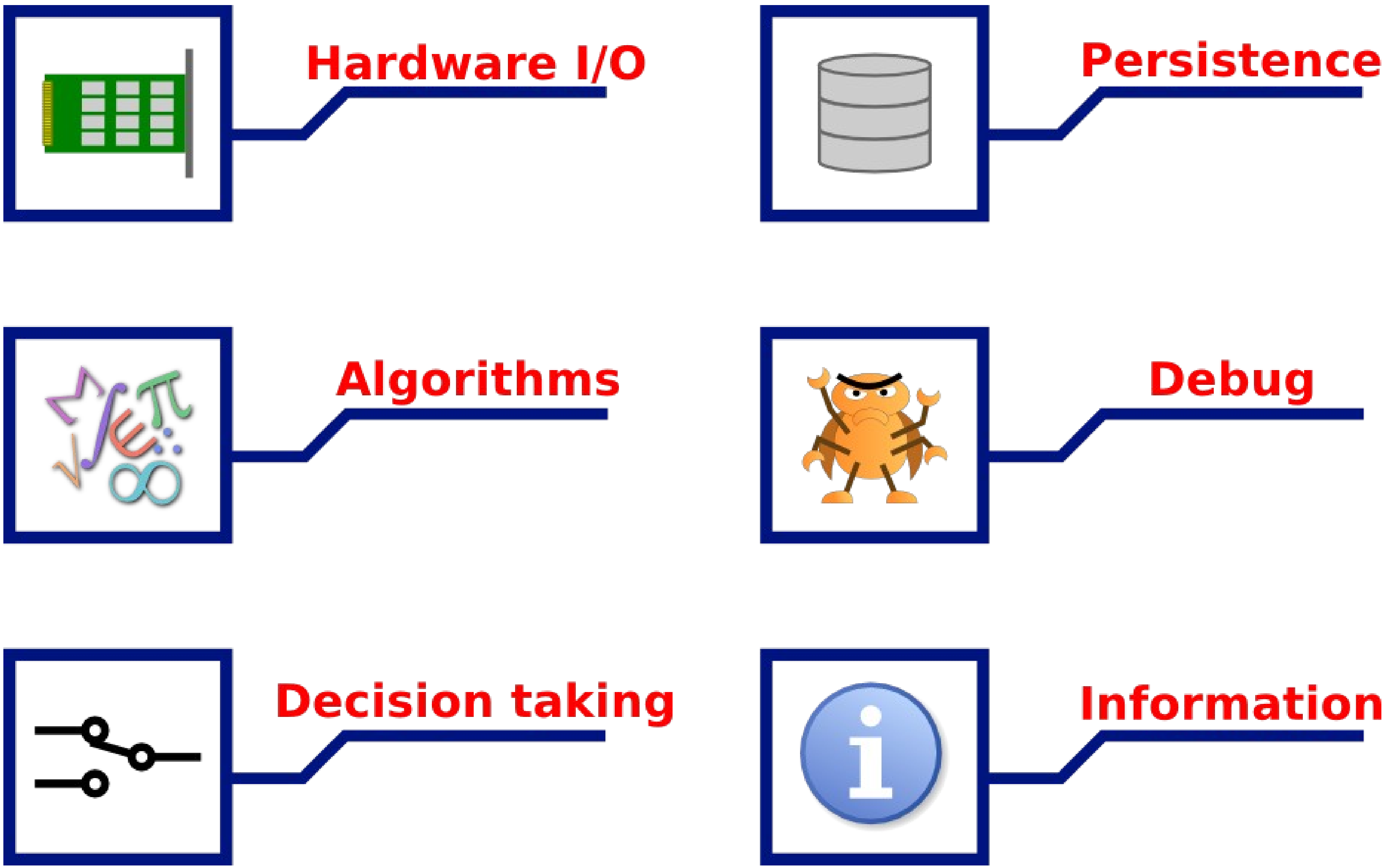
Modularity (GAMs)

- **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

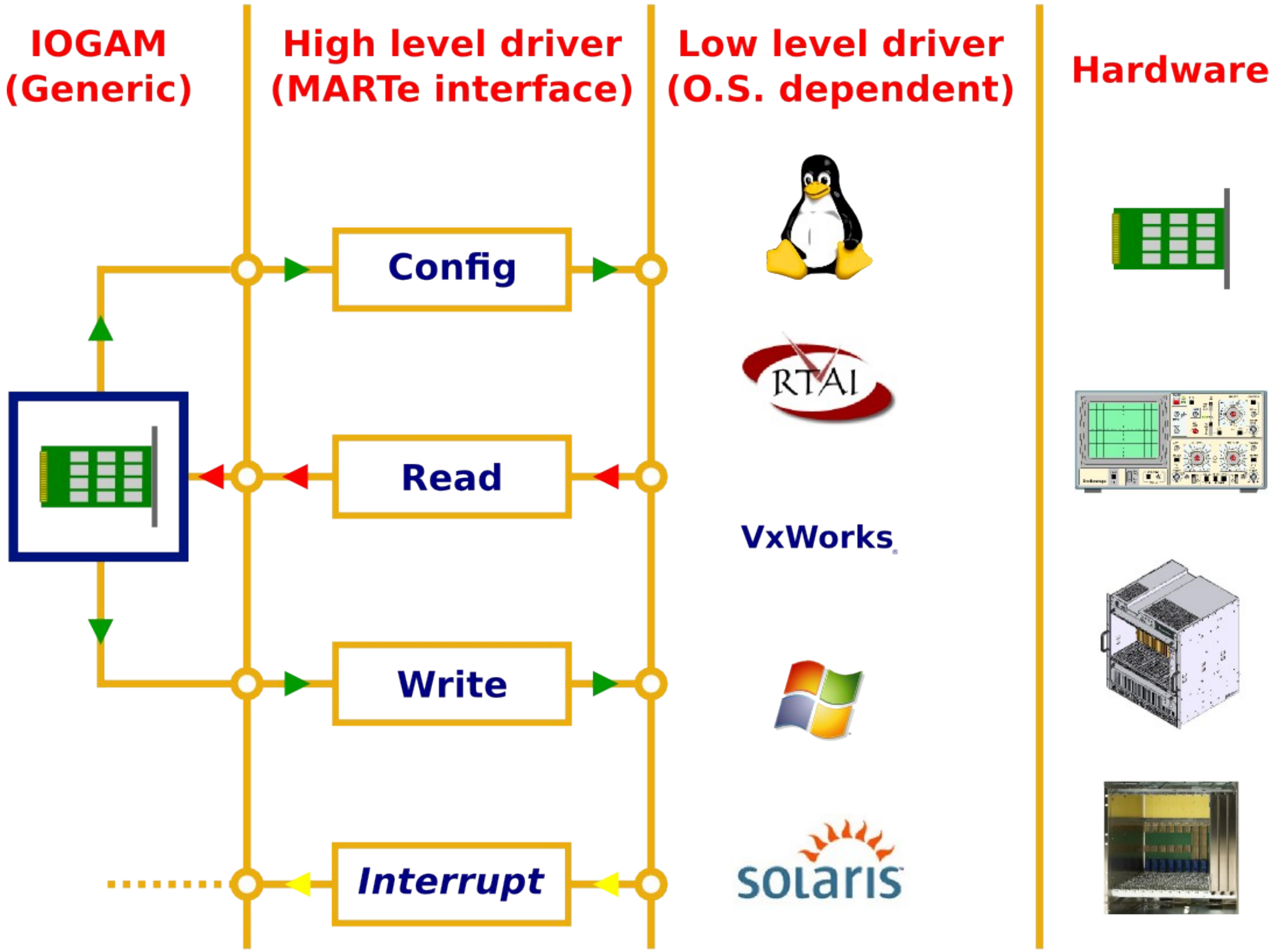


- **Simulation**
  - Replace actuators and plants with models
  - Keep all the other modules untouched

Common GAMs

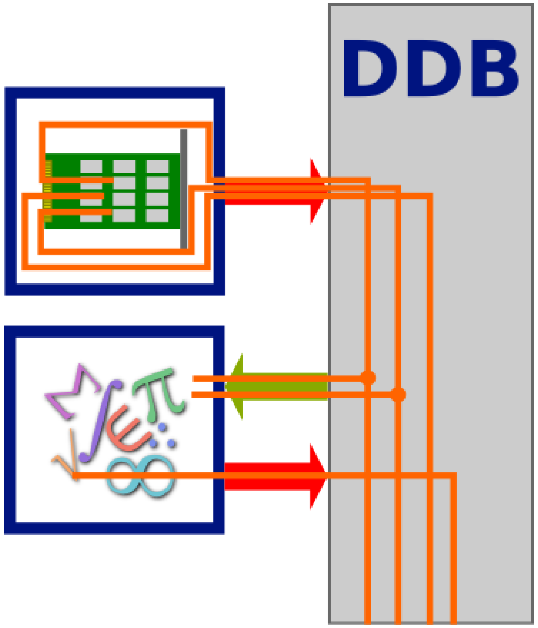

Hardware I/O
Persistence
Algorithms
Debug
Decision taking
Information

IOGAM


IOGAM
(Generic)
High level driver
(MARTe interface)
Low level driver
(O.S. dependent)
Hardware
Config
Read
Write
Interrupt
RTAI
VxWorks
solaris

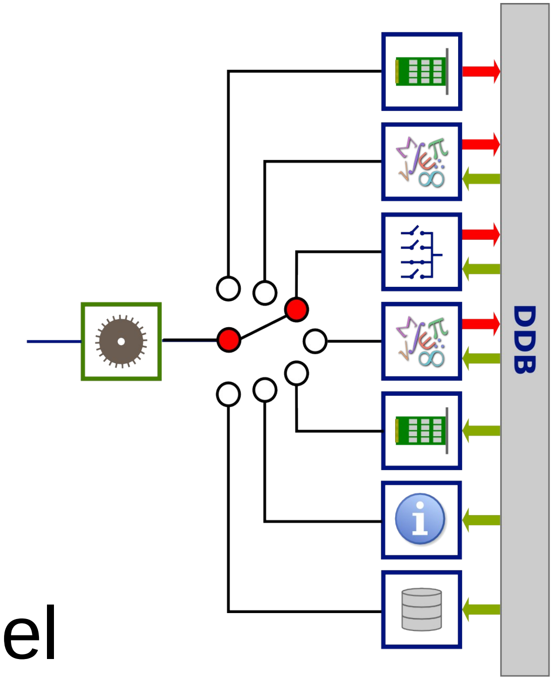
Dynamic Data Buffer

- 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



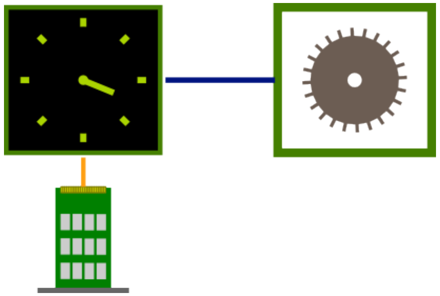
RT-Thread

- 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
  - synchronously or asynchronously



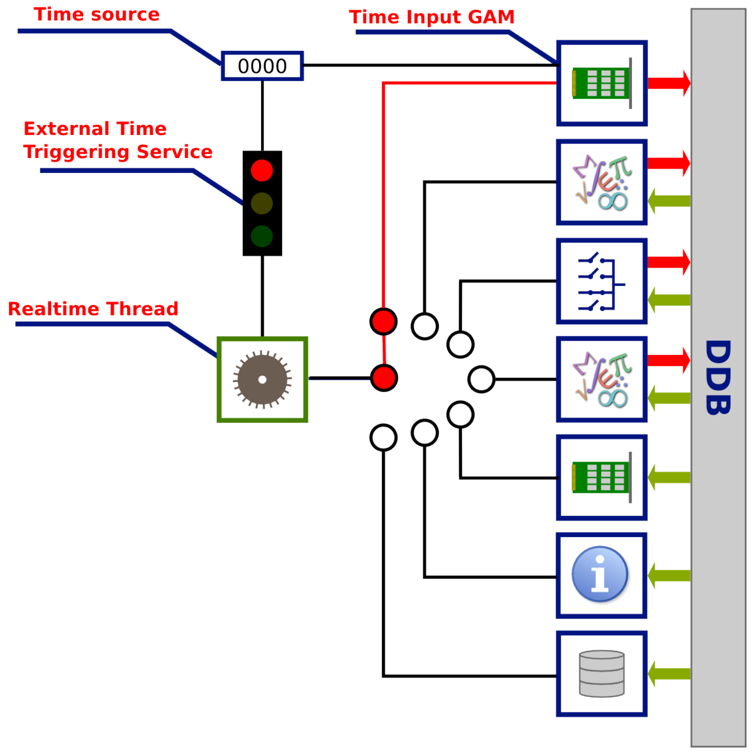
Synchronisation

- Asynchronous
  - Get latest available value
  - Verify acceptable latency (sample too late?)
- Synchronous
- Routinely used both schemes
- ADC, time input, ...
- Network
- From other control loop

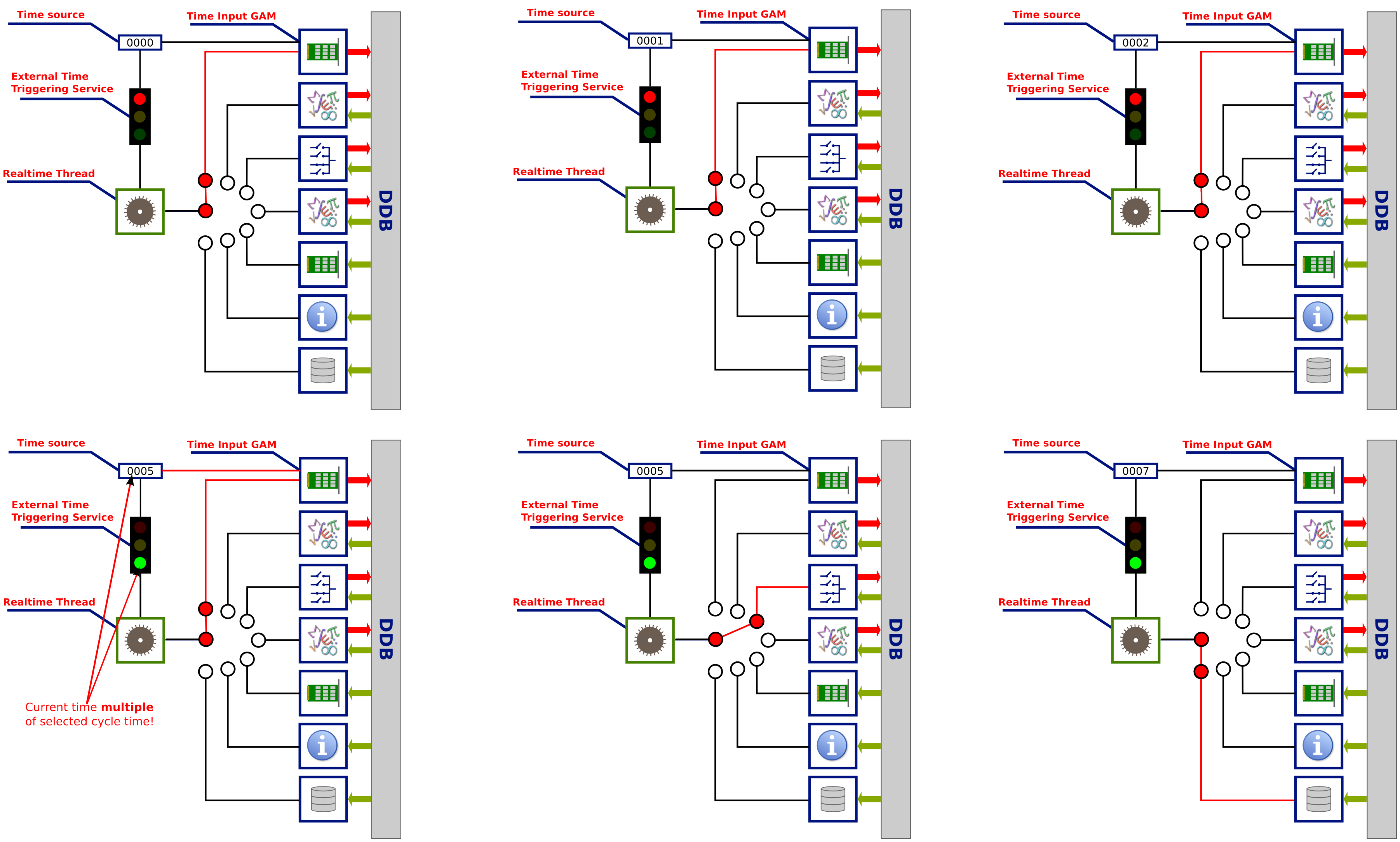


Synchronisation demo (1)

- ETTS waits for trigger from time source
- Current time multiple of cycle time?
- If so, unlock realtime thread and execute GAMs
- ETTS can be configured to exit after timeout
  - Trigger an error



Synchronisation demo (2)



Interfacing with MARTe (1)

- **Why?**
  - Send configurations
  - Retrieve acquired data
  - Query status
- **How?**
  - Component acts as a proxy to the outside world
    - Extended to implement the desired protocol
- MARTe is **interface agnostic**
  - No predefined GUI
  - No predefined high level protocols

Interfacing with MARTe (2)

- **Price?**
  - Requires the development of a module which translates your language to MARTe's language
  - MARTe forwards the configuration internally
  - A message server is provided
- HTTP interaction is widely used for retrieving information
  - Can also be used to change values
  - GAMs configuration
  - State machine
  - ...

MARTe Internal State Machine

- MARTe has its internal state machine
- It can be triggered by
  - External events
    - Has its own message interface
  - Internal events
    - e.g. errors while executing
- Capable of sending messages upon state changing

Introspection

- Probe the system
- Without sacrificing RT
- Crucial for an expedite debugging
- Does this still makes sense?
- New data streaming concepts, leverage concept?
- Stream your probes?

[illegible]	[illegible]
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 pa[illegible]meters

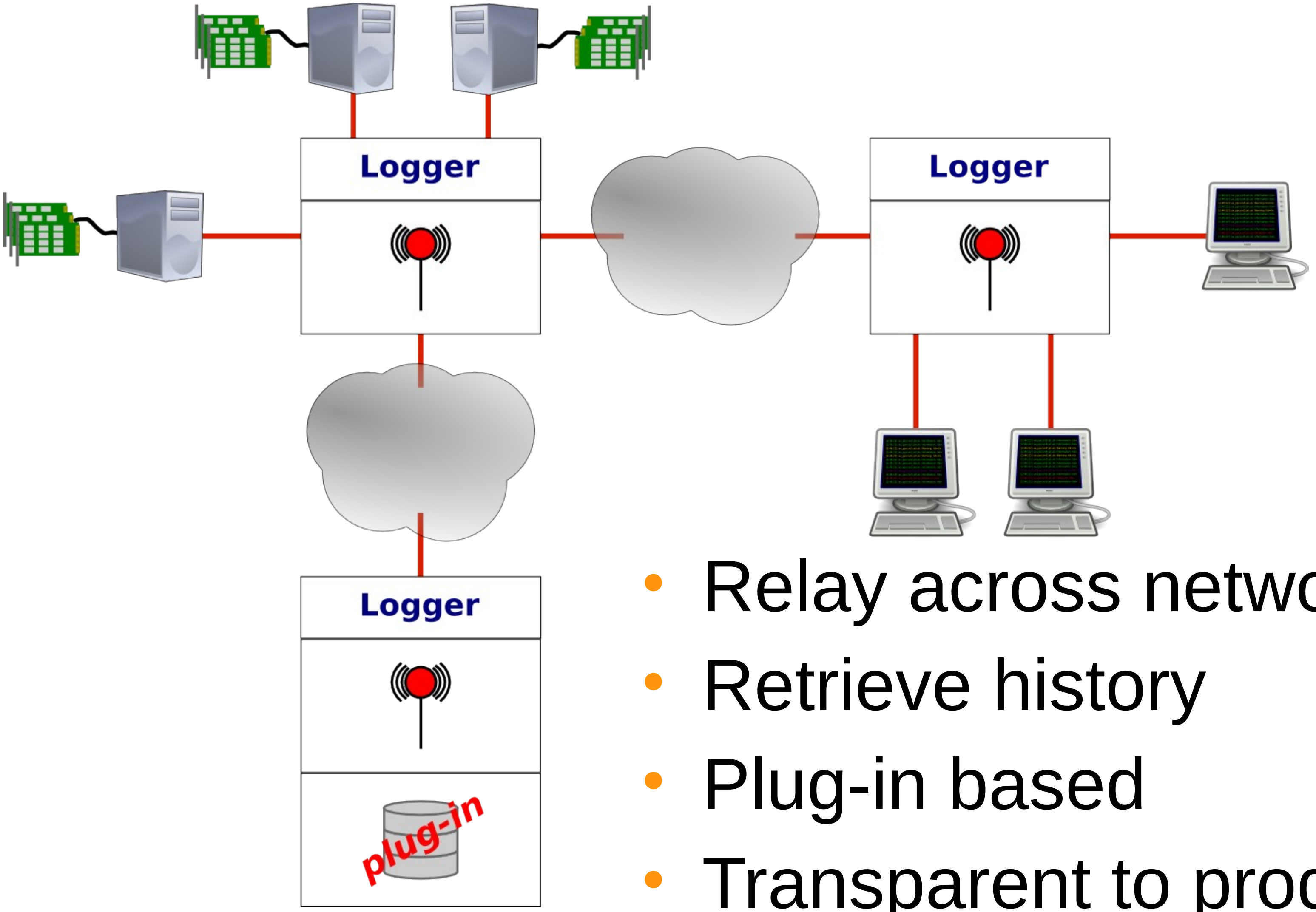
Parameter	[illegible]lue
Voltage delta threshold	[illegible].000000
High gain	10000.000000
Low gain	-5000.000000
Keep low gai[illegible]	12000 usecs

PCU[illegible]rrent 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

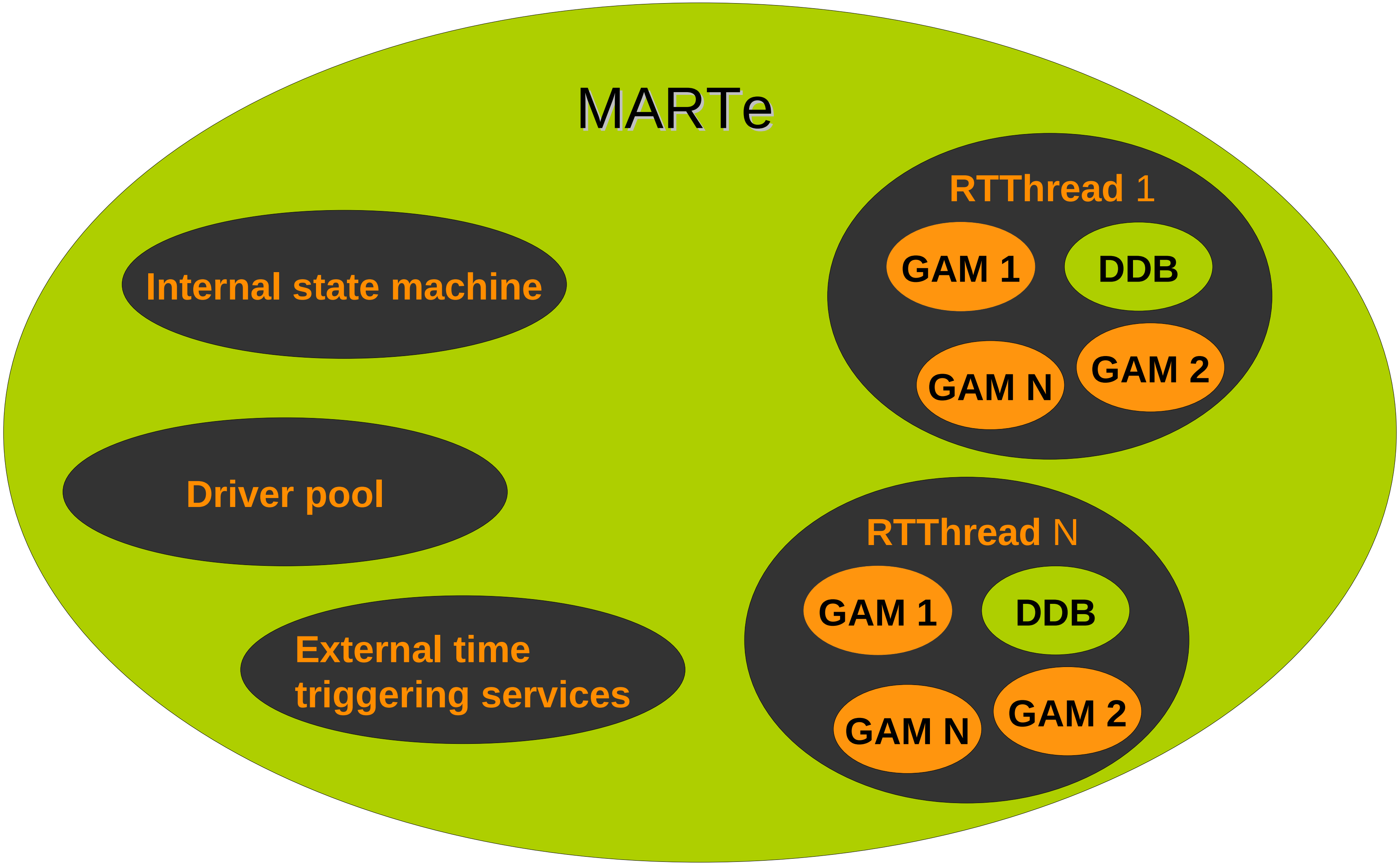
Automatically built

MARTe - logger

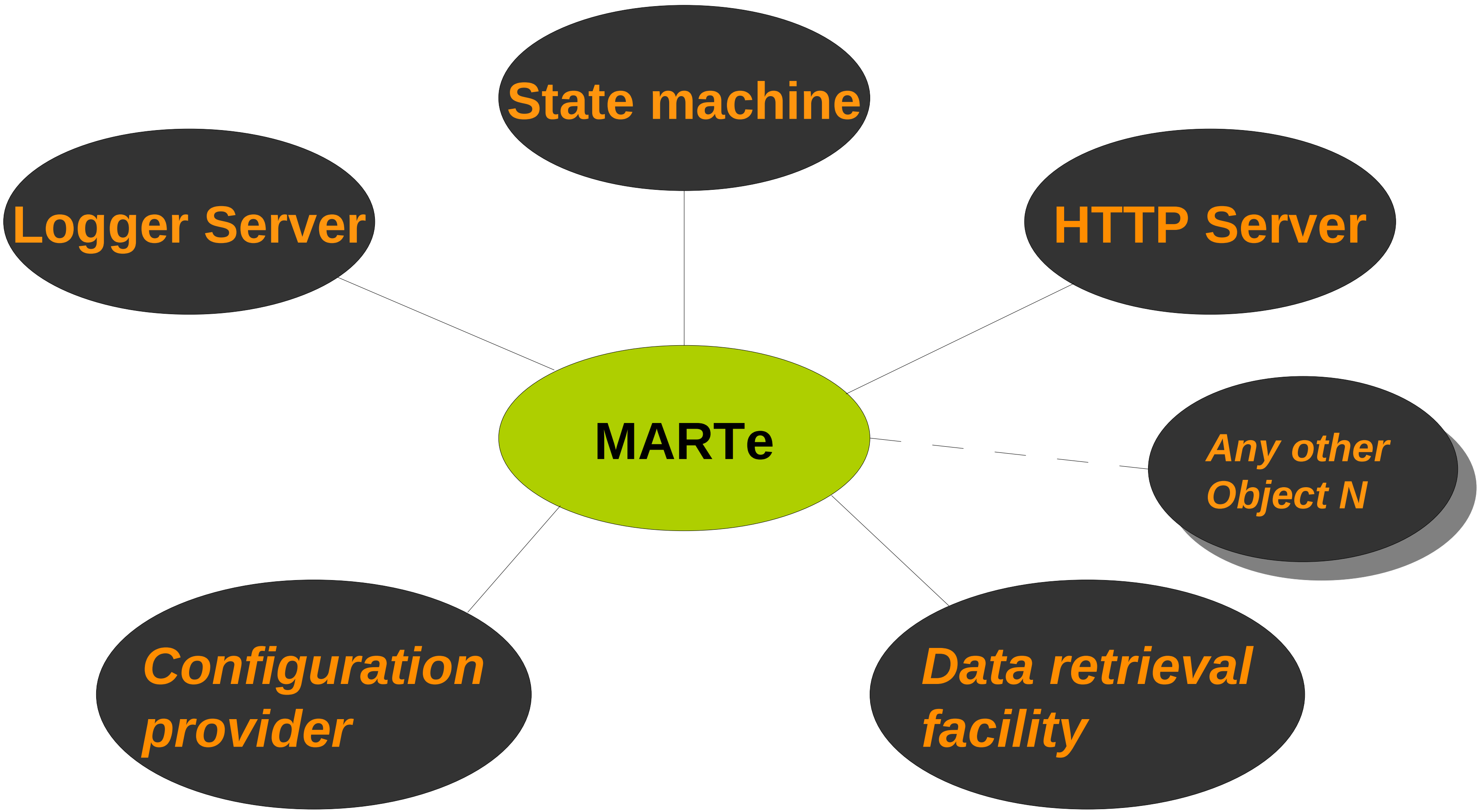


- Relay across networks
- Retrieve history
- Plug-in based
- Transparent to producers

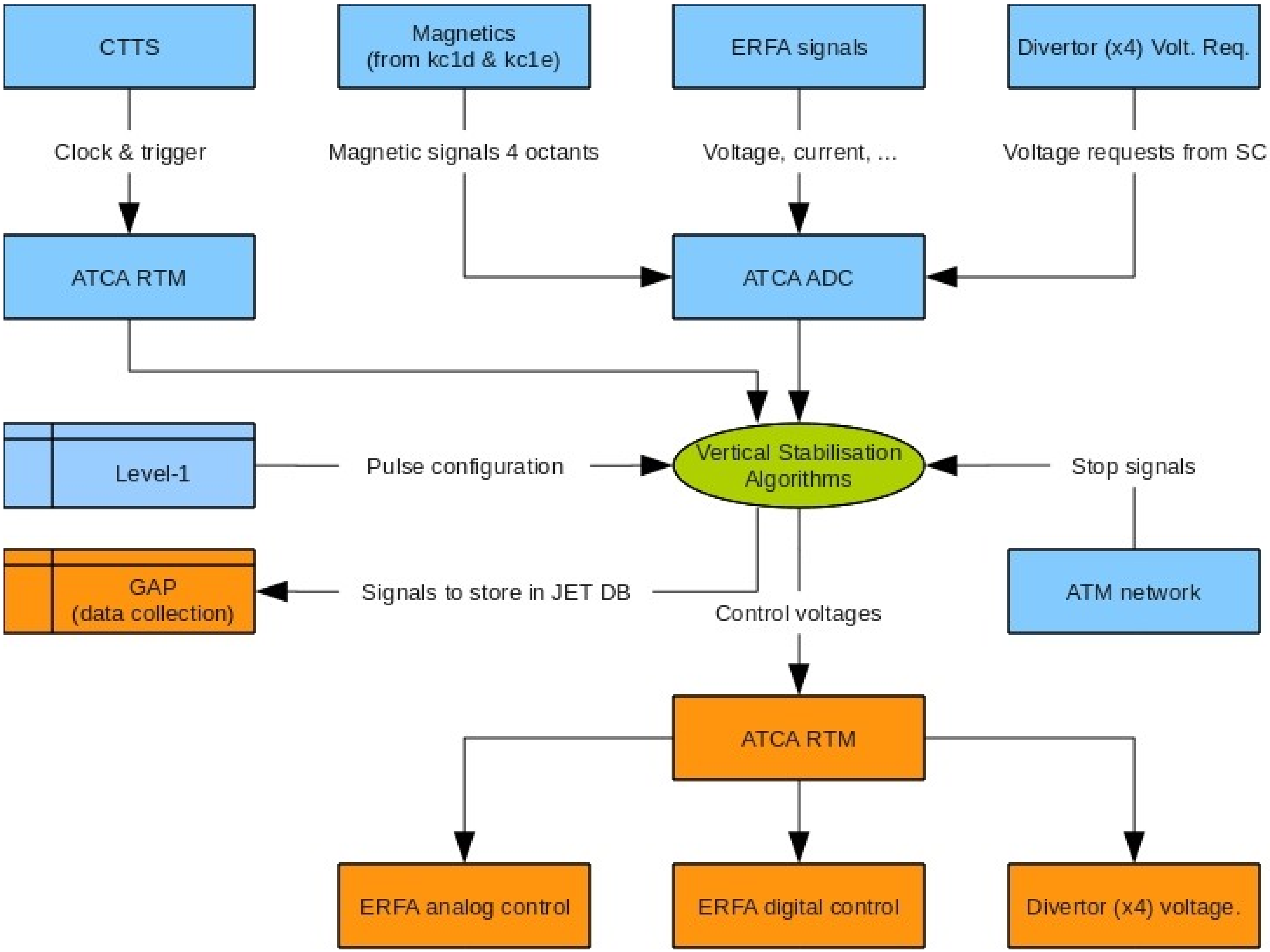
MARTe World


MARTe
Internal state machine
Driver pool
External time
triggering services
RTThread 1
GAM 1
DDB
GAM N
GAM 2
RTThread N
GAM 1
DDB
GAM N
GAM 2

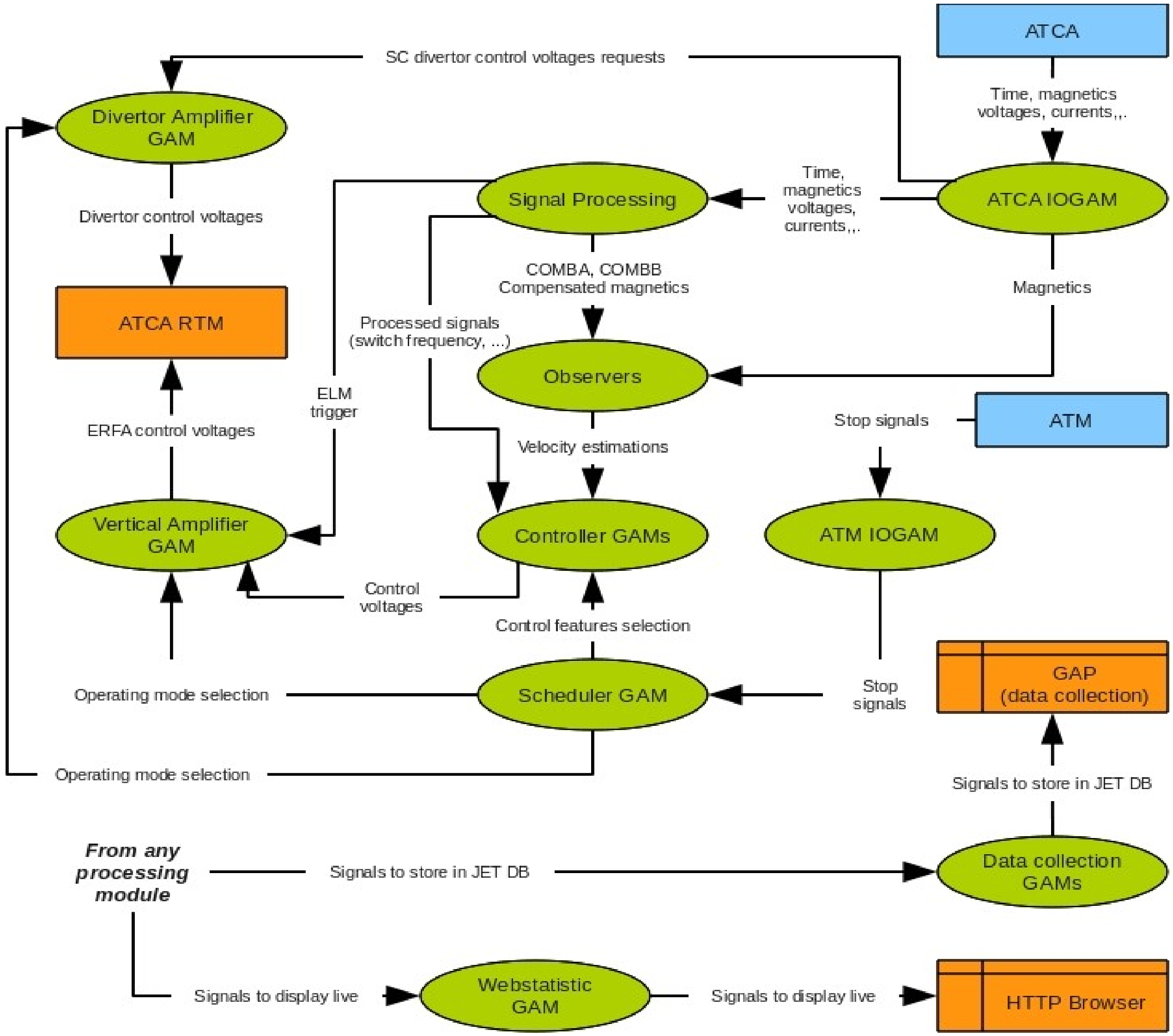
MARTe Universe



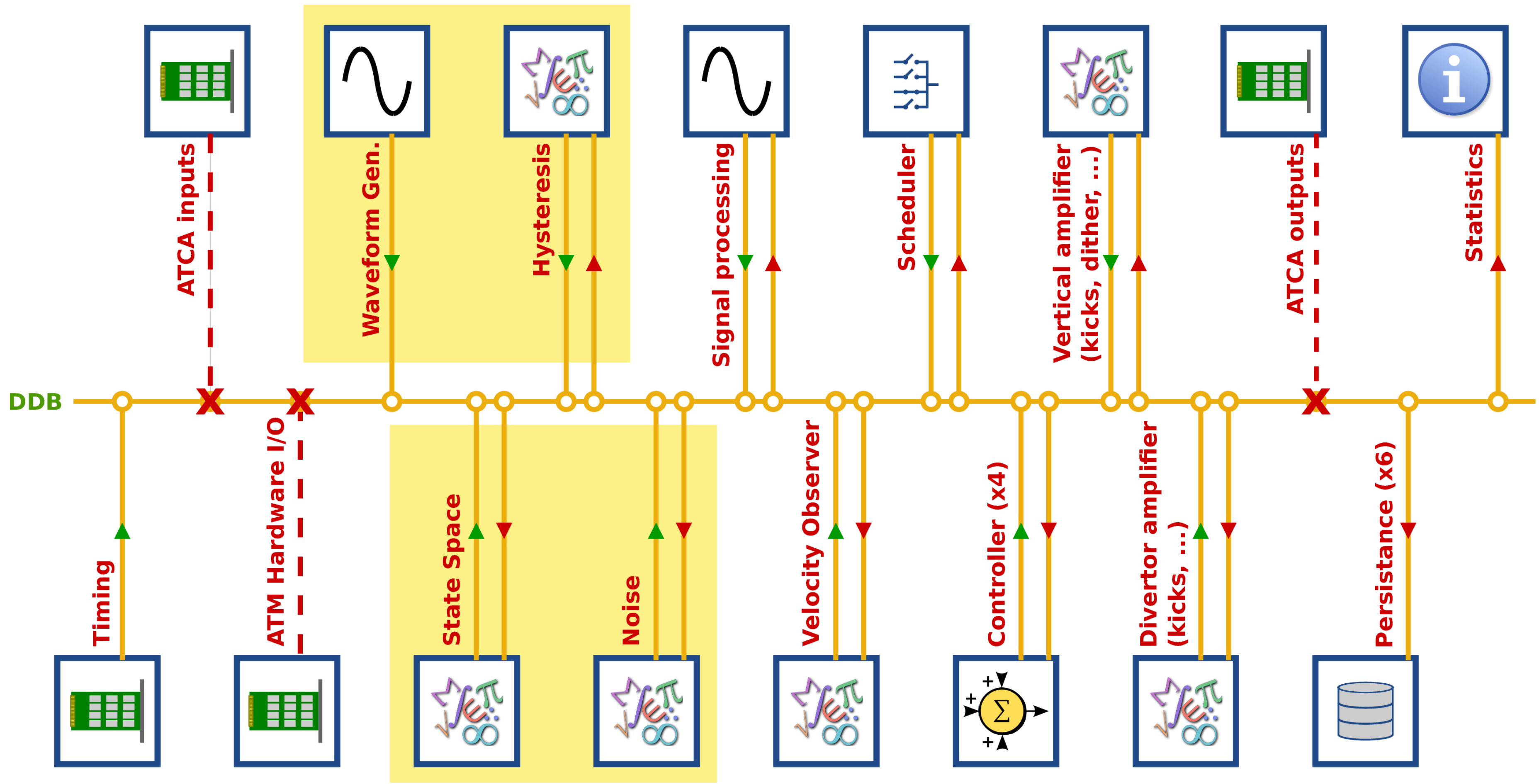
VS: a case study (1/3)



VS: a case study (2/3)



VS: a case study (3/3)



Does it work?

It is possible!

Modular

Data driven

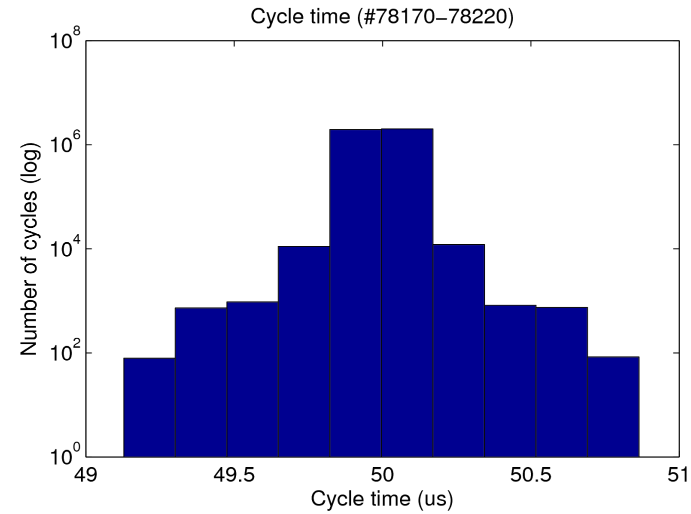
Introspection

Reliable

Performance

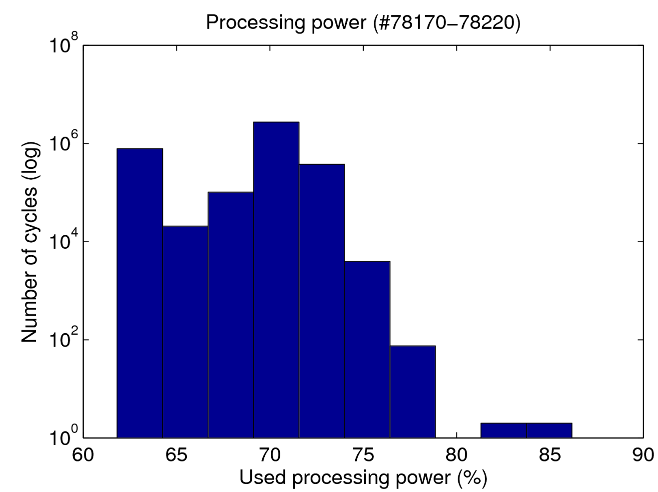
Low jitter

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

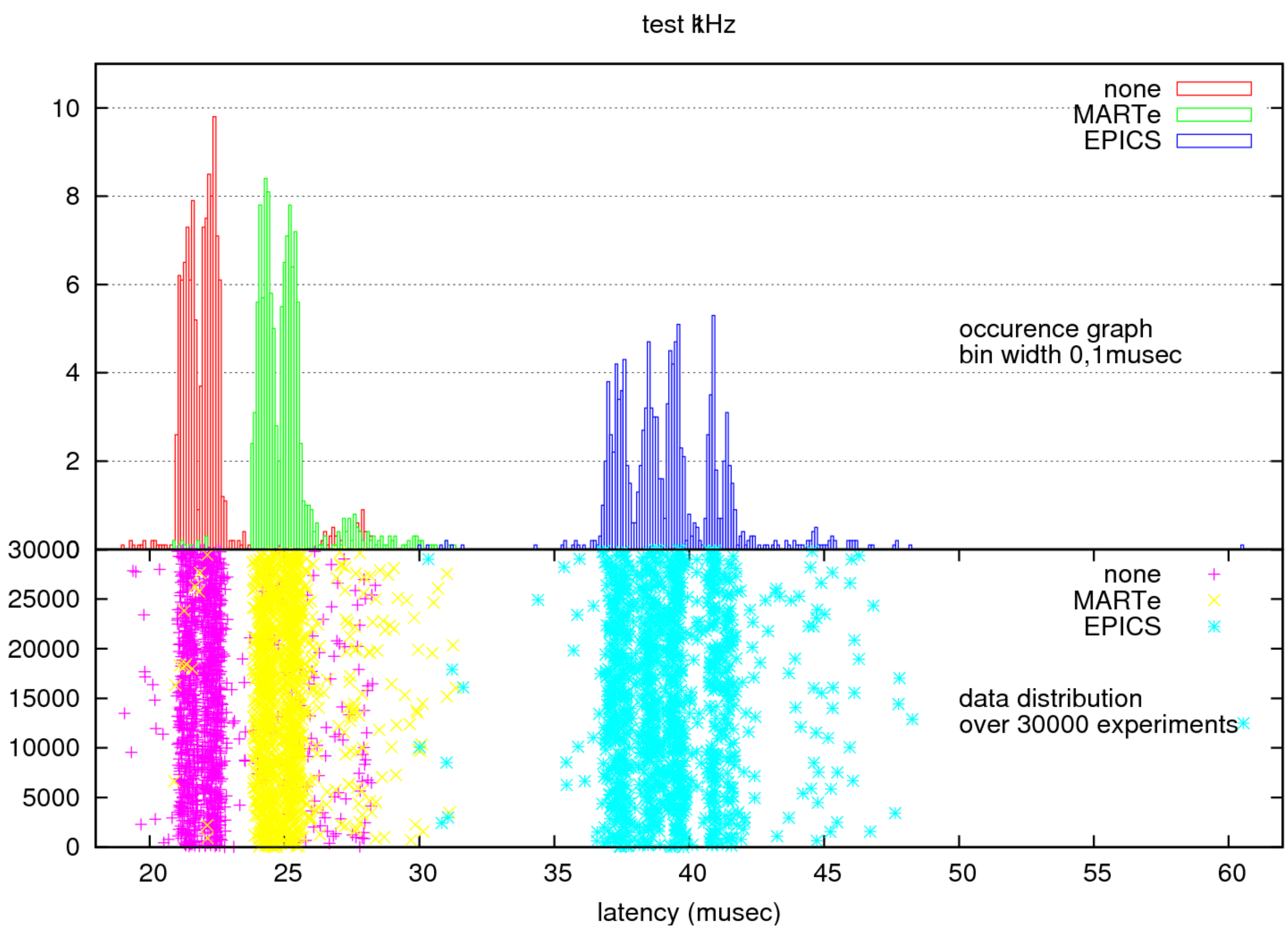


Working systems

JET VS	Linux-RTAI	50 μs
JET EFCC	VxWorks	200 μs
COMPASS SC	Linux*	500 μs
COMPASS VS	Linux*	50 μs
ISTTOK Tomography	Linux-RTAI	100 μs
FTU RT	Linux-RTAI	250 μs



How does it compare with EPICS? (for *Hard*-real-time?)



See: A. Barbalace, et. al, Performance comparison of EPICS IOC and MARTe in a Hard Real-Time Control Application, in IEEE-NPSS RT 2010

Future work

- **MARTe is interface agnostic...**
  - Would be good to have standard tools which help on the development and deployment of new systems
    - Simulink, Ptolemy
    - EPICS
- **MARTe has its own language**
  - Would be good to have a meta-language with builtin validation features
    - XML
- **More and better documentation**
  - Practically none targeted at the end user
    - Deployment and installation manual, GAM development manual
    - Configuration file writer manual, Real world examples
    - Tutorials

Backup slides