MARTe Framework

a Middleware for Real-time Applications Development

André Neto*[±], F. Sartori,
D. Alves, A. Barbalace,
L. Boncagni, B.B. Carvalho,
P. J. Carvalho, G. De Tommasi,
H. Fernandes, G. Manduchi,
P. McCullen, A. Stephen,
D. Valcarcel, R. Vitelli,
L. Zabeo and JET EFDA Contributors*

*Associação EURATOM/IST
Instituto de Plasmas e Fusão Nuclear-Laboratório Associado
Instituto Superior Técnico, Universidade Técnica de Lisboa
Portugal
http://www.ipfn.ist.utl.pt

[+]JET-EFDA
Culham Science Centre, UK
http://www.jet.efda.org/

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 plant configuration and data retrieval
- 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
    - Optimise code here
- Possible?
  - Yes, runs in Linux, Linux+RTAI, VxWorks, Solaris and MS Windows

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

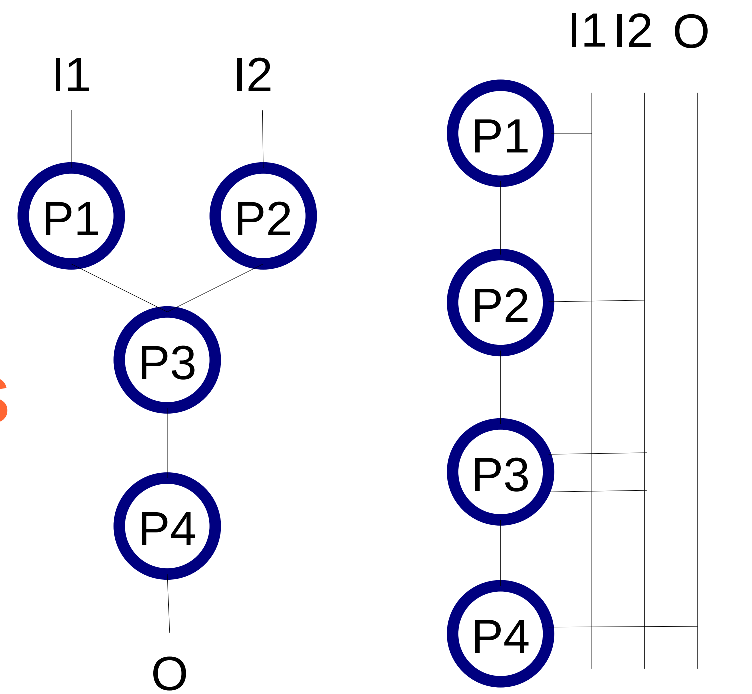
Object configuration

- Structured syntax
- Similar to XML
- Classes are automatically instantiated
- 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
      }
...
```

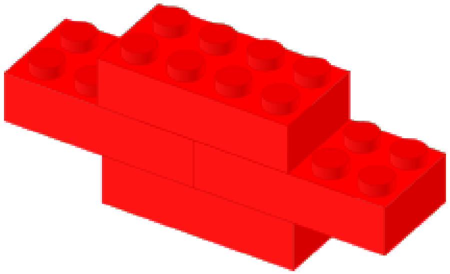
MARTe language

- Graph simulink like control schemes translates into serial execution
- Can it run in parallel? Yes
- Scheduling order is preset
- Distributed control (network or same machine)



Modularity (GAMs)

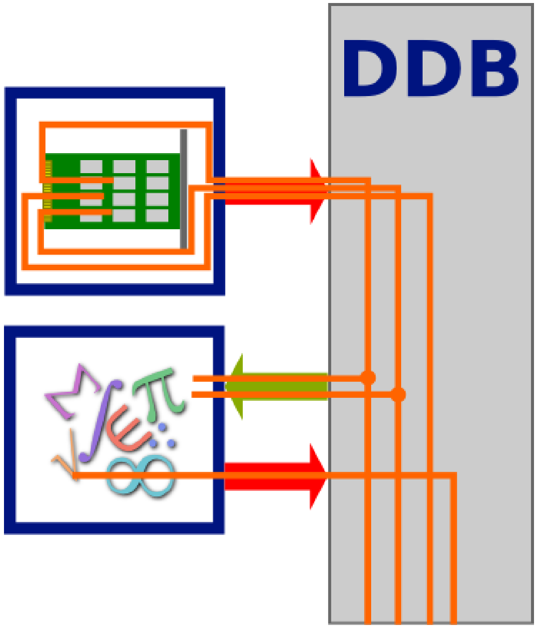
- Define boundaries
  - Algorithms and hardware don't mix!
  - Modules do only what they advertise
  - No interdependence or *a priori* knowledge
- Generic by design
  - Same goals, same module
  - Reusability and maintainability



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

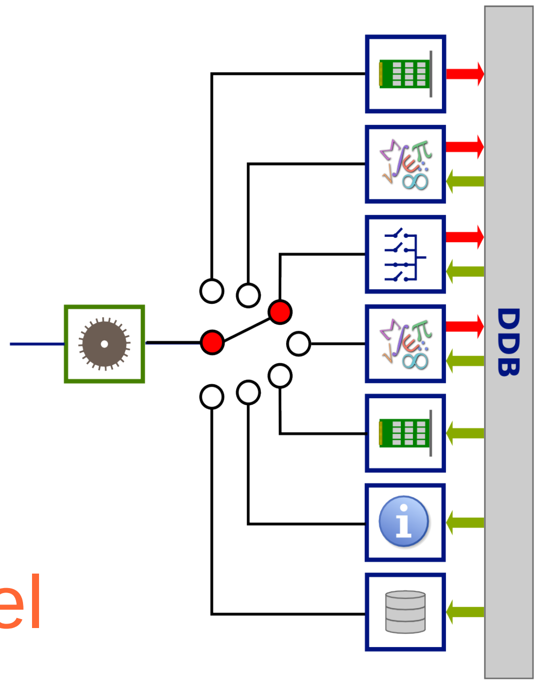
Dynamic Data Buffer

- 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



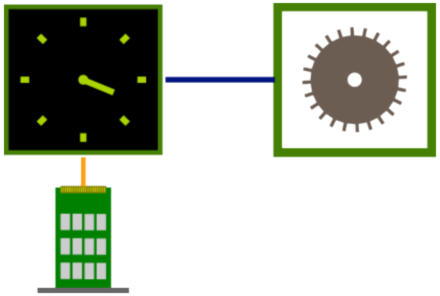
Real-time thread

- 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



Introspection

- Probe the system
  - Without sacrificing RT
- Crucial for an expedite debugging
- Network continuous data streaming
  - No impact in RT performances

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

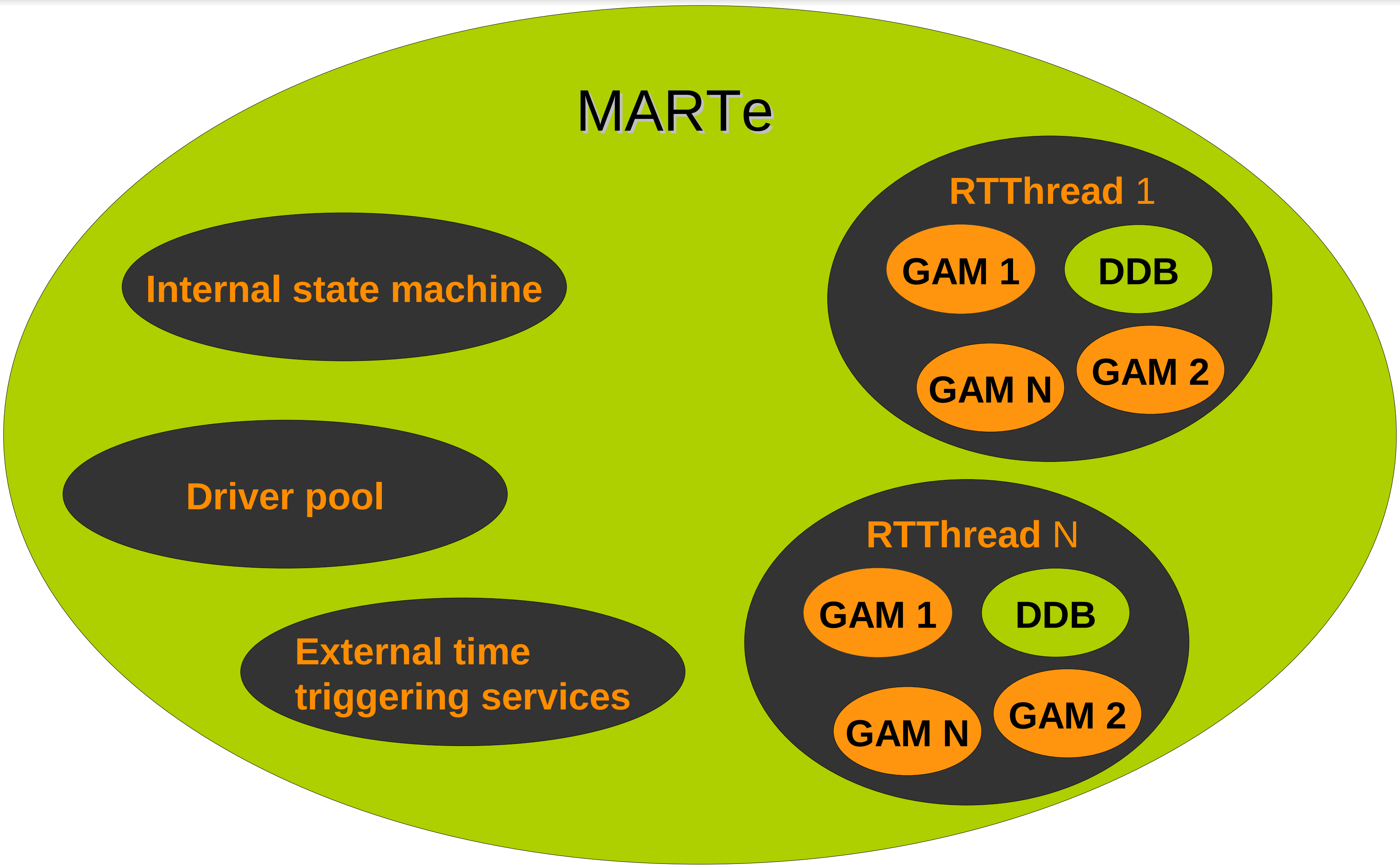
Parameter	Value
Voltage delta threshold	[illegible]
High gain	[illegible]
Low gain	[illegible]
Keep low gain f[illegible]	12000 usecs

PCU2 current adaptation parameters

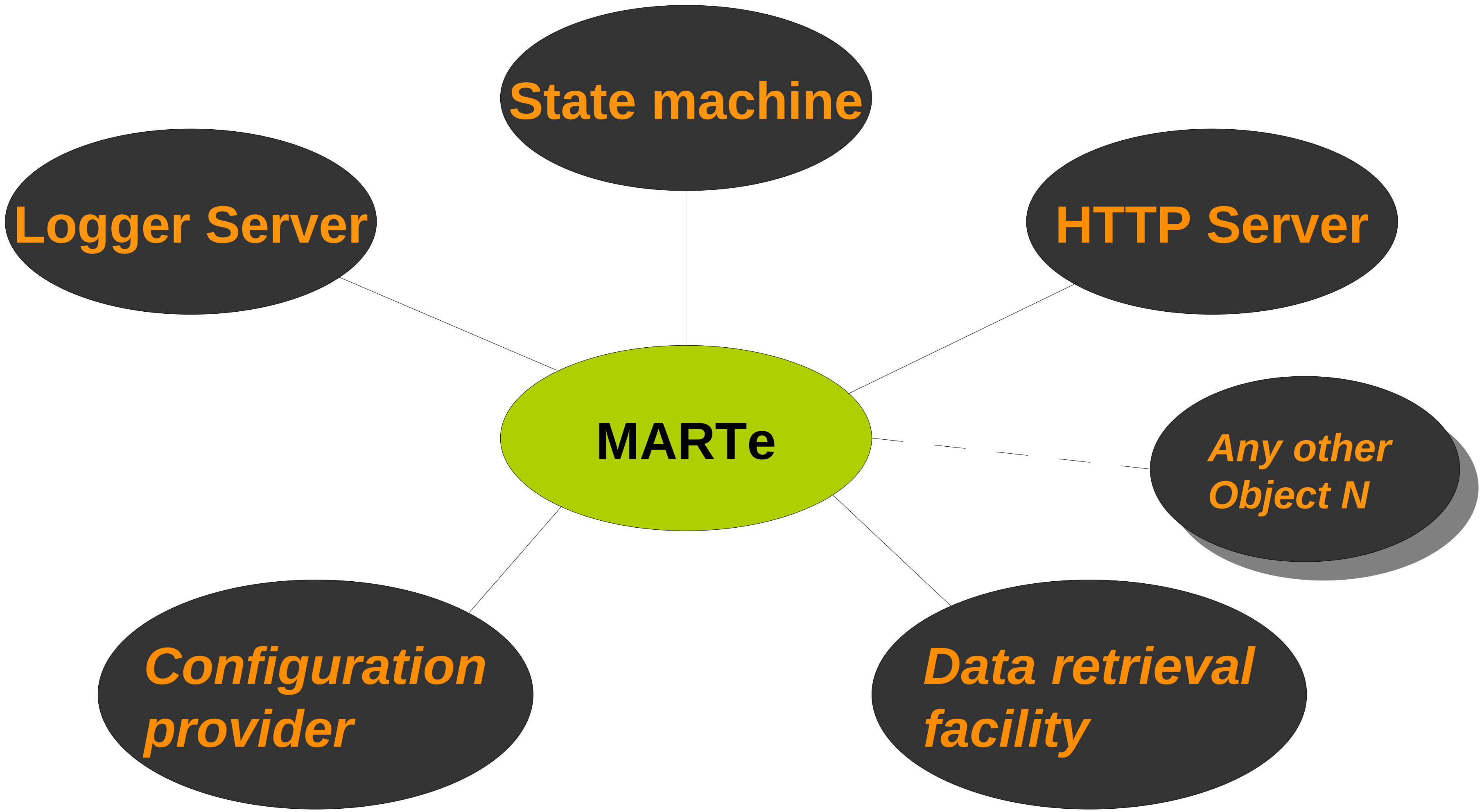
Parameter	Value
Amplifier current saturation index threshold	2500.000000
High gain	-15000.000000
Low gain	-5000.000000
Alpha	0.500000
Beta	0.800000

Automatically built

MARTe World

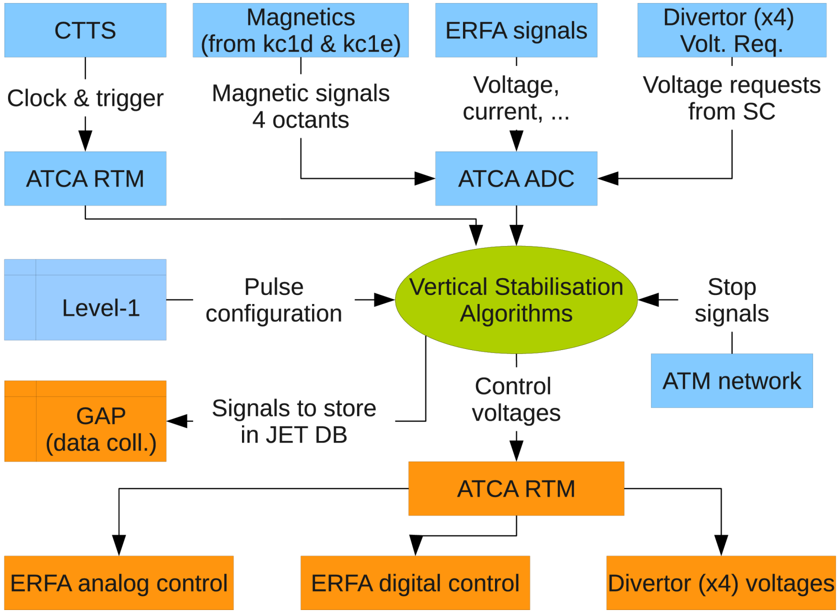

MARTe
Internal state machine
Driver pool
External time
triggering services
RTThread 1
GAM 1
DDB
GAM N
GAM 2
RTThread N
GAM 1
DDB
GAM N
GAM 2

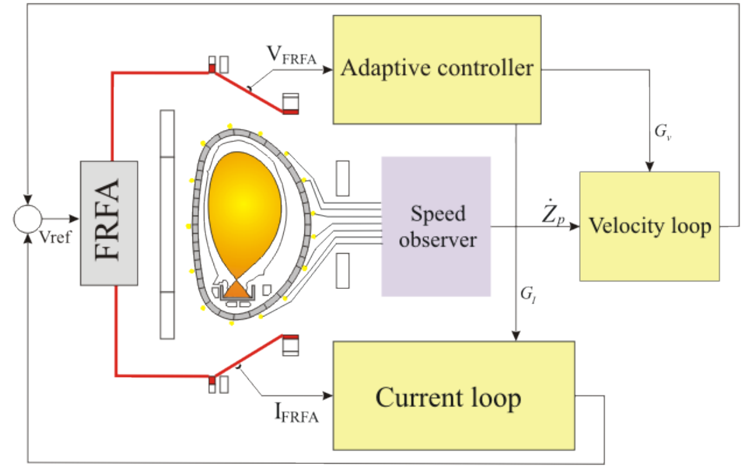
MARTe Universe



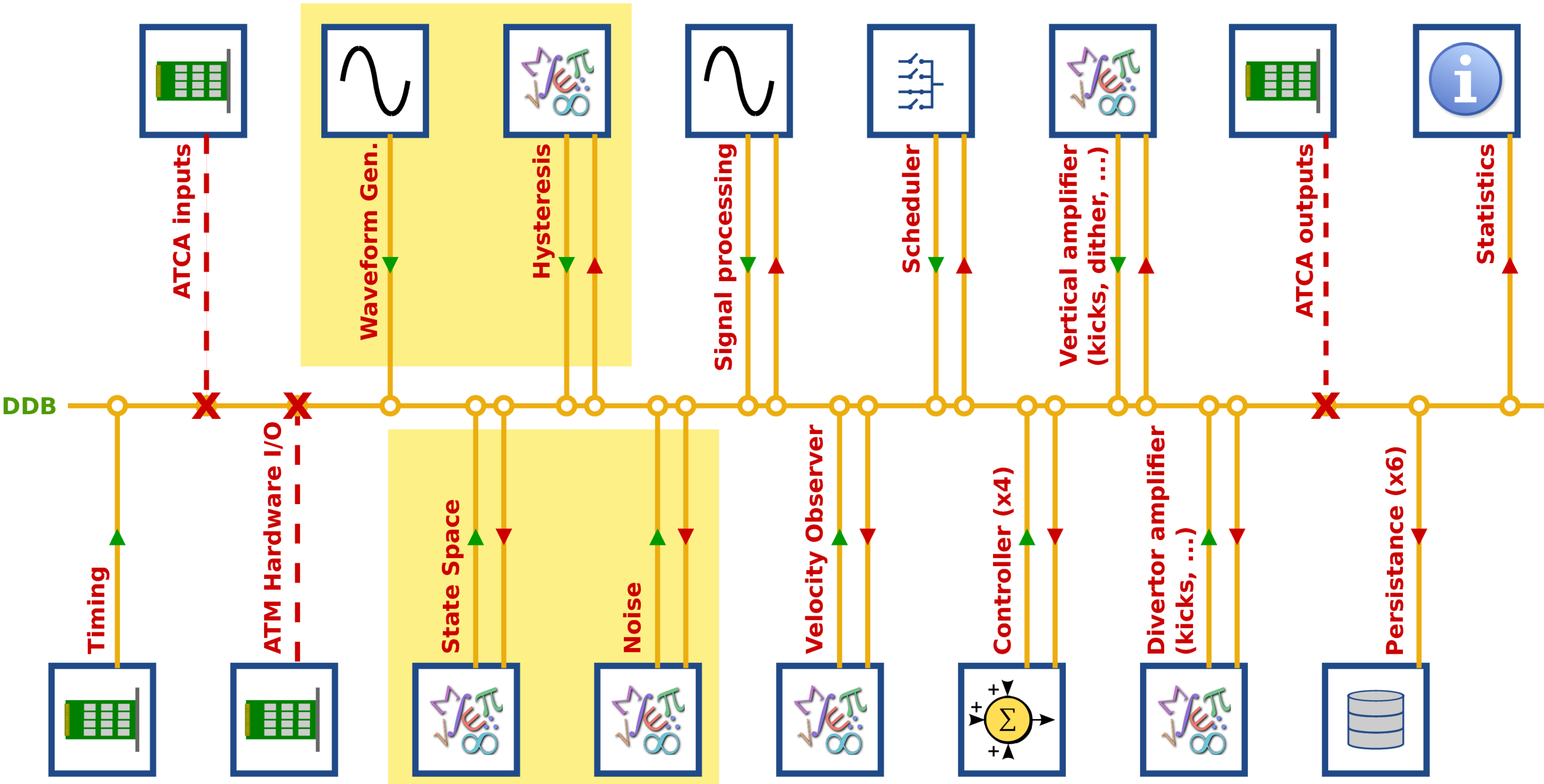
VS – An example

- Elongated tokamak plasmas are susceptible to a vertical axisymmetric instability
- Dedicated Vertical Stabilisation System required
- Essential system for operation
- Growth rate of 1000s-1
- Loss of control can produce forces in the order of the 100's of tonnes





VS-GAMs

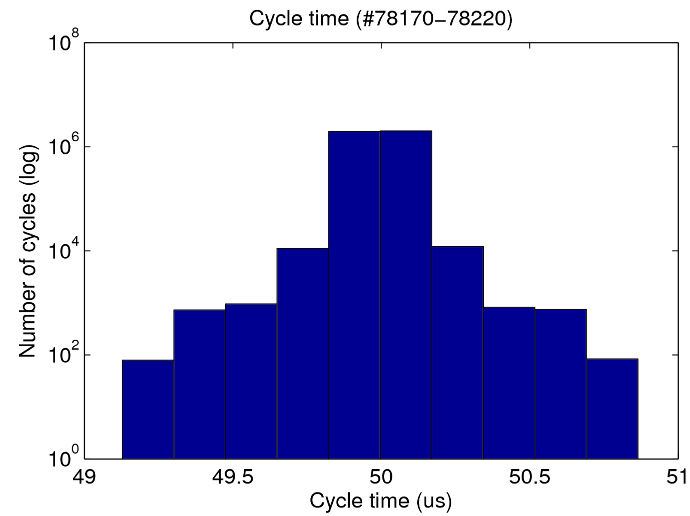

ATCA inputs
Waveform Gen.
Hysteresis
Signal processing
Scheduler
Vertical amplifier
(kicks, dither, ...)
ATCA outputs
Statistics
DDB
Timing
ATM Hardware I/O
State Space
Noise
Velocity Observer
Controller (x4)
Divertor amplifier
(kicks, ...)
Persistance (x6)

Conclusions

- MARTe
  - Designed for real-time systems
  - Multi-platform
    - Key for simulation and commissioning
  - Modular
    - Reusability and maintainability
    - Clear boundary between algorithms, hardware interaction and system configuration
  - Portable
  - Data-driven
  - Live introspection

Does it work?

- e.g. Vertical Stabilisation
  - Essential to operation
    - Loss of control can produce forces in the order of the 100's of tonnes
  - 50 ± 0.10 μs
    - Always running



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	500 μs
RFX MHD Control	Linux*	125 μs
JET RTPS	VxWorks	2 ms
JET VTM	Linux*	10 ms
JET WALLS	Linux*	10 ms
IST FPSH	Linux*	100 μs