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/
• 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**

```java
+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
MARTe Universe

State machine

Logger Server
HTTP Server
Any other Object N

Configuration provider
Data retrieval facility
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)

A. Neto | ICALEPCS 2011, Oct 13 | MARTe
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
      - $50 \pm 0.10 \mu s$
    - Always running

**Working systems**

<table>
<thead>
<tr>
<th>System</th>
<th>OS</th>
<th>Cycle Time (μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JET VS</td>
<td>Linux-RTAI</td>
<td>50</td>
</tr>
<tr>
<td>JET EFCC</td>
<td>VxWorks</td>
<td>200</td>
</tr>
<tr>
<td>COMPASS SC</td>
<td>Linux*</td>
<td>500</td>
</tr>
<tr>
<td>COMPASS VS</td>
<td>Linux*</td>
<td>50</td>
</tr>
<tr>
<td>ISTTOK Tomography</td>
<td>Linux-RTAI</td>
<td>100</td>
</tr>
<tr>
<td>FTU RT</td>
<td>Linux-RTAI</td>
<td>500</td>
</tr>
<tr>
<td>RFX MHD Control</td>
<td>Linux*</td>
<td>125</td>
</tr>
<tr>
<td>JET RTPS</td>
<td>VxWorks</td>
<td>2</td>
</tr>
<tr>
<td>JET VTM</td>
<td>Linux*</td>
<td>10</td>
</tr>
<tr>
<td>JET WALLS</td>
<td>Linux*</td>
<td>10</td>
</tr>
<tr>
<td>IST FPSH</td>
<td>Linux*</td>
<td>100</td>
</tr>
</tbody>
</table>