FUSION FOR ENERGY

Interfacing MARTe2 to the EPICS Channel Access and pvAccess protocols

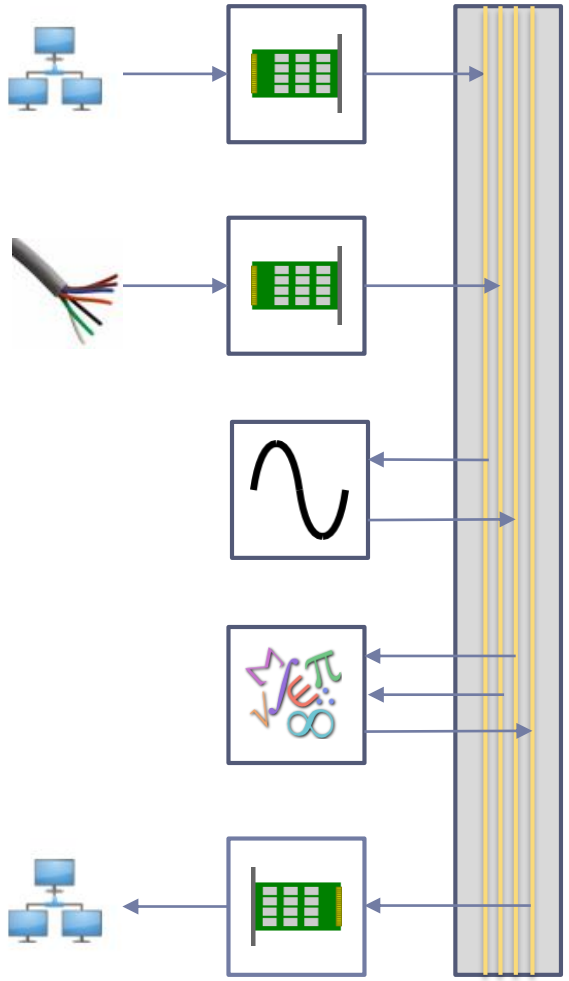
EPICS Collaboration Meeting

André Neto, Filippo Sartori, Bertrand Bauvir & MARTe community

June, 2019

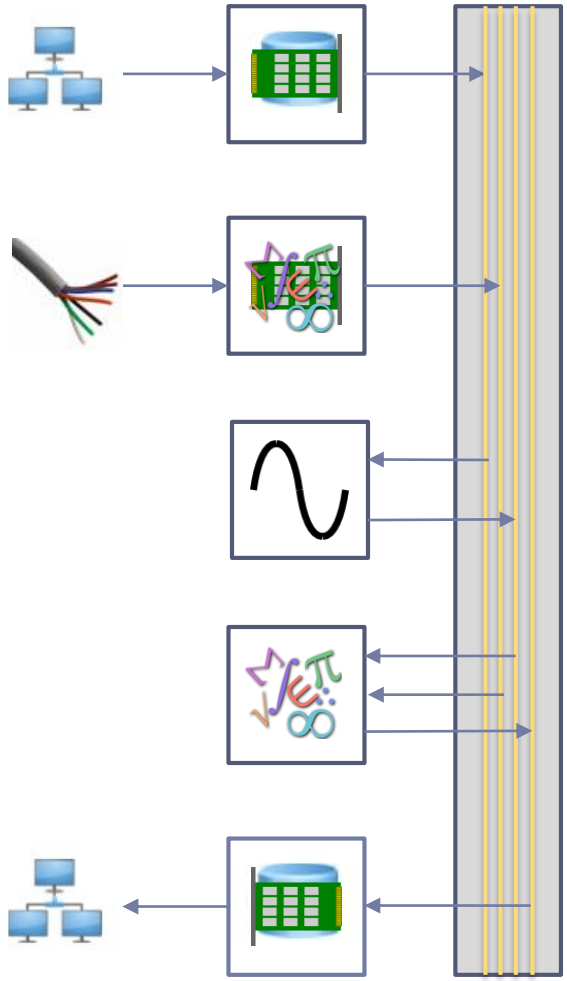
MARTe?

- Multi-platform C++ real-time control middleware
  - Simulink-like way of describing the problem
- Modular **development** and **execution** environment for control systems
- Ensures and monitors real-time
- Facilitates test & commissioning
- MARTe2 – the QA version
  - Documentation
  - Static code analysis – MISRA
  - Testing
    - Full functional
    - > 90 % coverage



Main features

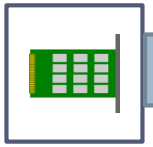
- Modular
- Clear boundary between algorithms, hardware interaction and system configuration
- Reusability and maintainability
- Simulation
- Agnostic of the operational environment
  - Different profiles of people can develop in parallel
    - Interfaces to hardware
    - Algorithms
    - Interfaces to CODAC
  - Develop and test in different platforms
    - Deploy the same source



https://vcis.f4e.europa.eu/marte2-docs/master/html/

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

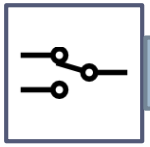

Hardware


Models


Persistence


Algorithms


Debug

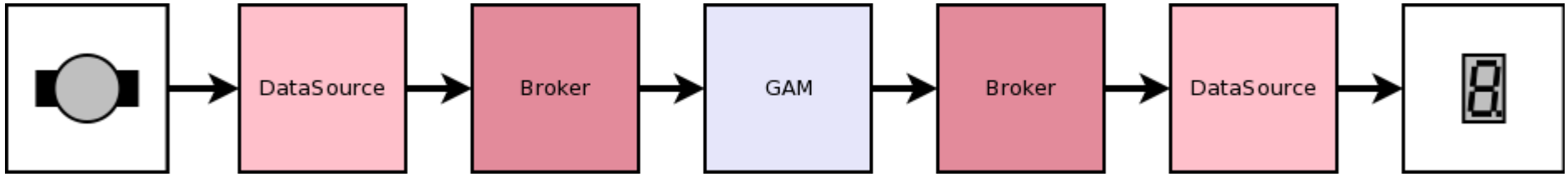

Decision


Scheduling


Information

DataSources

- Interface between GAMs and the outside world
  - Bridge data into/from DDB signals
    - Using MARTe high level drivers
- Connect to hardware
  - ADCs
  - DACs
  - DIOs
  - Networks
  - Filesystems
  - …



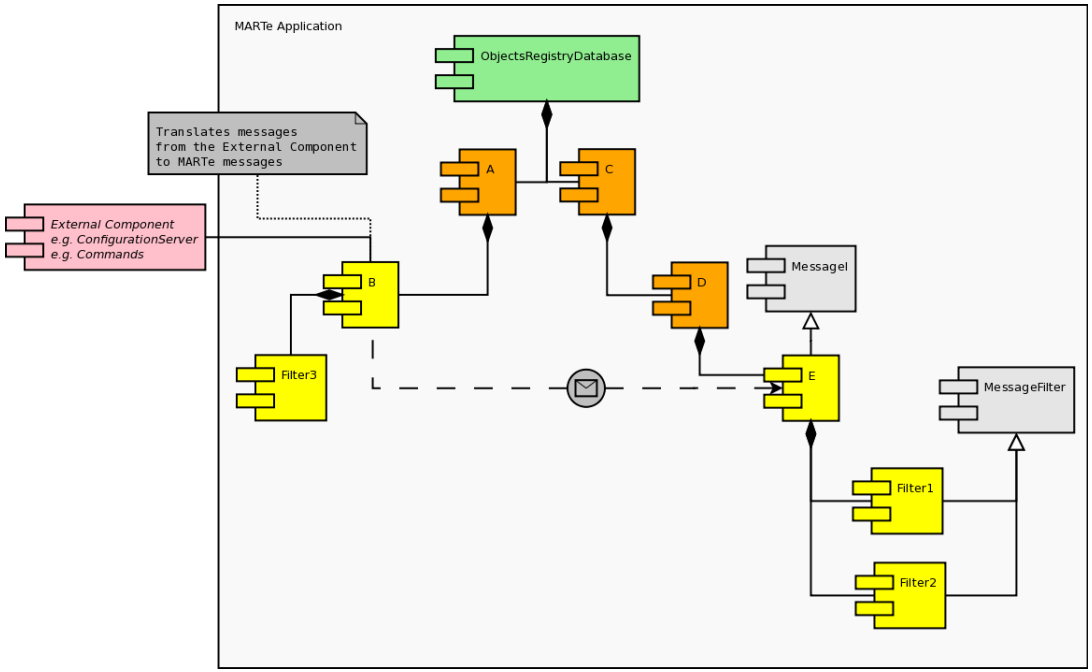
Data-driven components

- Structured syntax
  - Defines common language
  - Simple
    - Human readable configuration
  - Provides built-in validation
  - Allows for a clear way of expressing the problem
- XML and JSON also supported
- Classes are automatically instantiated
- Configuration is validated by the created object
- Asserting and parsing functions available
- Support for user-defined data types

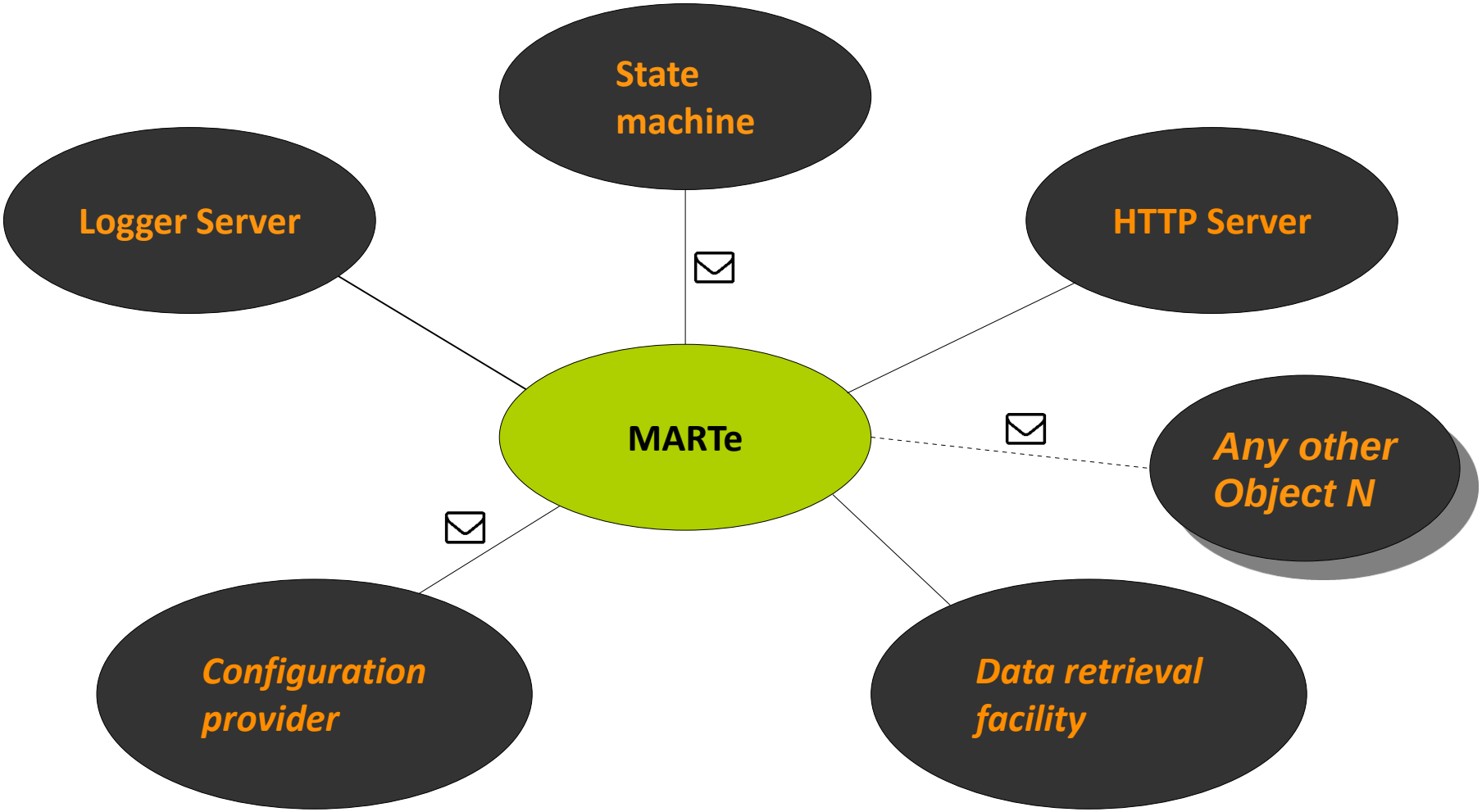
```
...
+ThreadStats1Struct = {
  Class = IntrospectionStructure
  Thread1C = {
...
+Control = {
  Class = ControlGAM
  Controller = {
    NoPlasmaVelocityGain = 0.0
    NoPlasmaCurrentGain = 40.0
    IPWaveform = {
      Times      = {0 120}
      Amplitudes = {0.5 0.5}
      Rounding   = 50
    }
    Type = ThreadStats1Struct
...
  }
  Ready = {
    Type = ThreadStats1Struct
  }
  ...
```

Messages

- Change the behaviour of an application based only on configuration data
  - i.e. without requiring any code recompilation.
- Provide a generic interface between MARTe components and any components and protocols that live outside a MARTe application
  - Deployment of applications into new plants without changing code
- Typically used for non real-time activities, such as configuration and state management

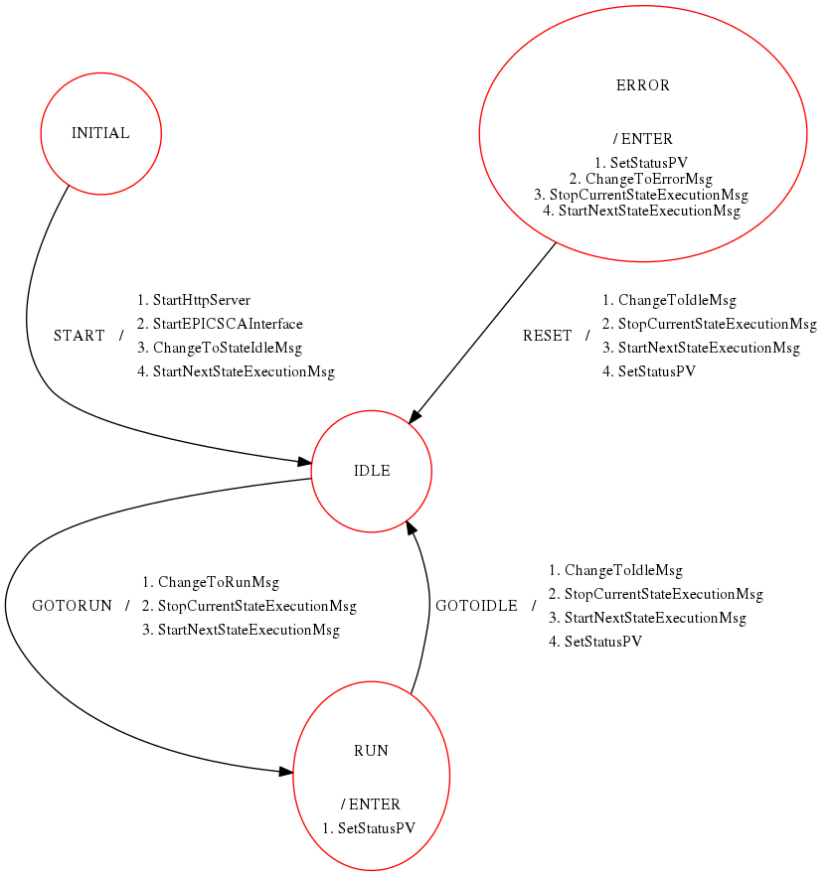


Interfacing with MARTe

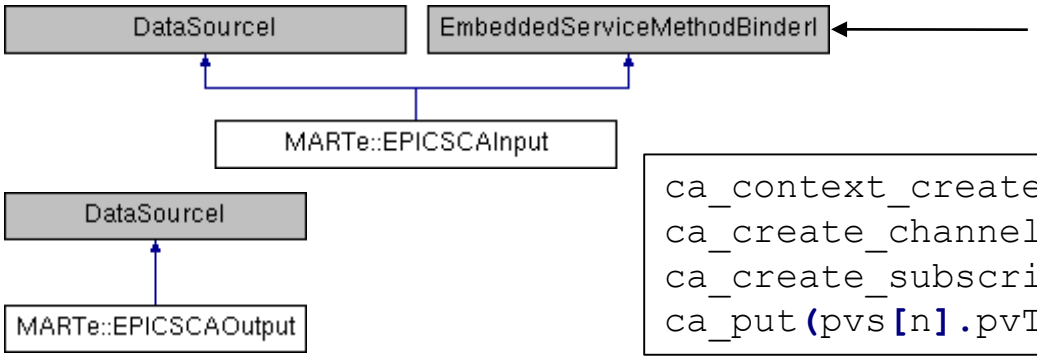


Channel Access - Objectives

- Monitoring
  - Asynchronously query value of a given set of variables
- Commands
  - Proxy PV value change into Messages
- E.g. change the state-machine state
- Input/output data source
  - Typically non-real-time
  - ITER => SDN for real-time networking



EPICSCA DataSources



Asynchronous w.r.t. to real-time threads

```
ca_context_create(ca_enable_preemptive_callback) ...
ca_create_channel(&pvs[n].pvName[0], ...
ca_create_subscription(pvs[n].pvType, ...
ca_put(pvs[n].pvType, ...
```

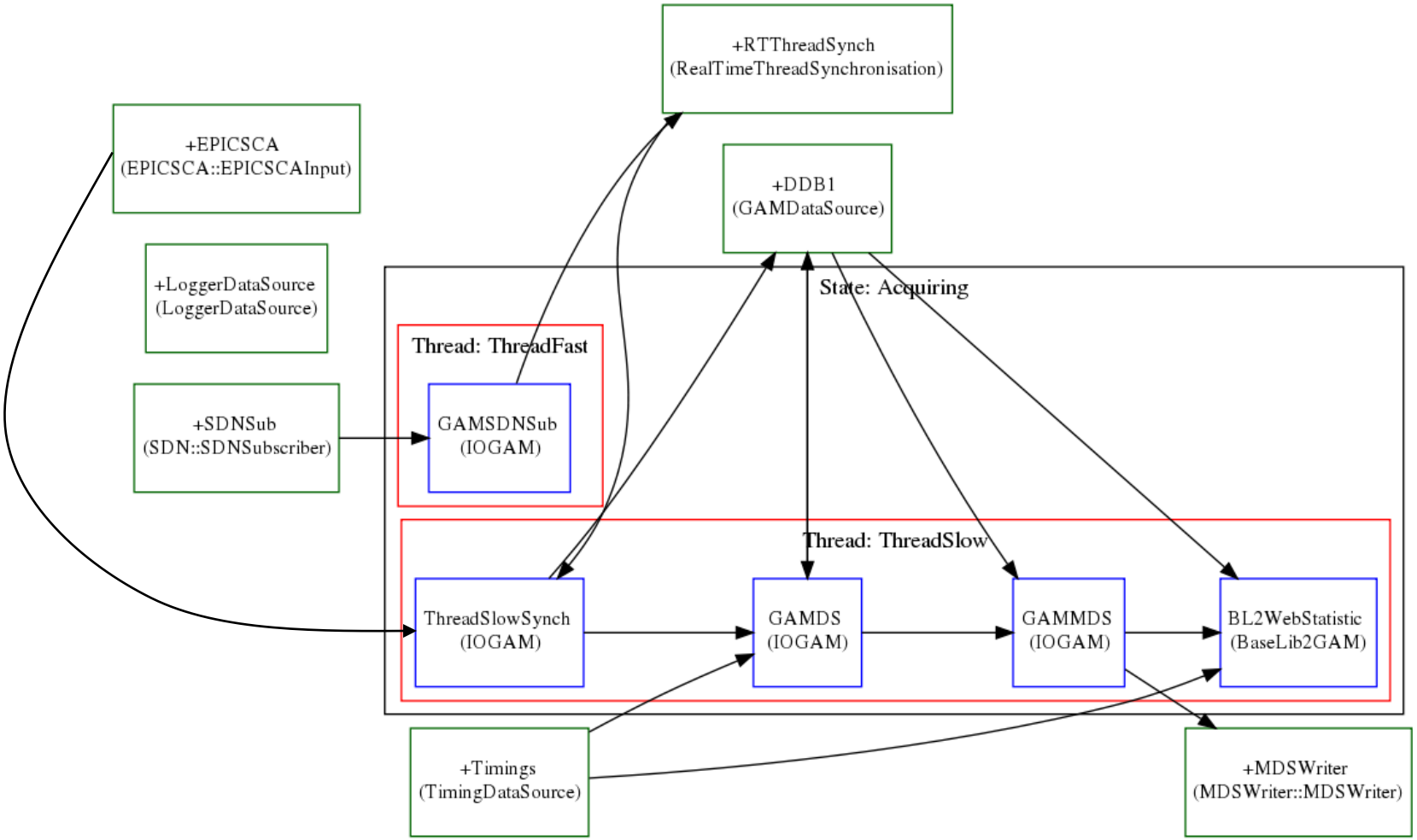
```
+EPICSCAInput_1 = {
    Class = EPICSCA::EPICSCAInput
    StackSize = 1048576
    CPUs = 0xff
    Signals = {
        PV1 = {
            PVName = My::PV1
            Type = float32
            NumberOfElements = 10
        }
        ...
    }
}
```

```
+EPICSCAOutput_1 = {
    Class = EPICSCA::EPICSCAOutput
    StackSize = 1048576
    CPUs = 0xff
    IgnoreBufferOverrun = 1
    NumberOfBuffers = 10
    Signals = {
        PV1 = {
            PVName = My::PV1
            Type = uint32
        }
        ...
    }
}
```

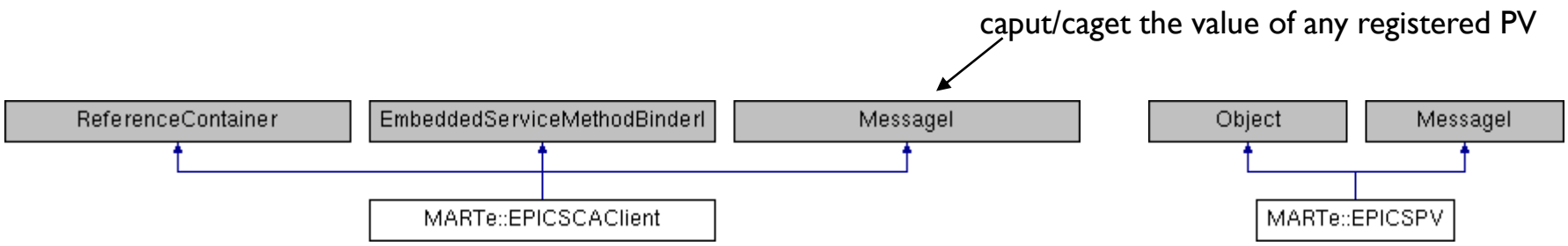
https://vcis-jenkins.f4e.europa.eu/job/MARTe2-Components-docs-master/doxygen/classMARTe_1_1EPICSCAInput.html
https://vcis-jenkins.f4e.europa.eu/job/MARTe2-Components-docs-master/doxygen/classMARTe_1_1EPICSCAOutput.html

e.g. Falcon PON sampler

- Synchronously store the value of all plant PVs based on SDN tick



EPICSCAClient




```
+EPICS_CA = {
  Class = EPICS::EPICSCAClient
  StackSize = 1048576
  CPUs = 0x2
  AutoStart = 0
  +PV_STATUS = {
    Class = EPICS::EPICSPV
    PVName = "FDAQ:Fast_Status"
    PVType = uint32
  }
  +PV_COMMAND = {
    Class = EPICS::EPICSPV
    PVName = "FDAQ:Fast_Status_CMD"
    PVType = uint32
    Event = {
      Destination = StateMachine
      PVValue = Function
      FunctionMap = {{"1", "MAKEREADY"},
{"0", "GOOFFPULSE"}}
```

```
+StateMachine = {
  Class = StateMachine
  ...
  +ONLINE = {
    Class = ReferenceContainer
    +ENTER = {
      Class = ReferenceContainer
      +SetStatusPV = {
        Class = Message
        Destination = "EPICS_CA.PV_STATUS"
        Function = CAPut
        +Parameters = {
          Class = ConfigurationDatabase
          param1 = 1
        }
      }
    }
    +StartNextStateExecutionMsg  = {
      ...
```

https://vcis-jenkins.f4e.europa.eu/job/MARTe2-Components-docs-master/doxygen/classMARTe_1_1EPICSPV.html
https://vcis-jenkins.f4e.europa.eu/job/MARTe2-Components-docs-master/doxygen/classMARTe_1_1EPICSCAClient.html

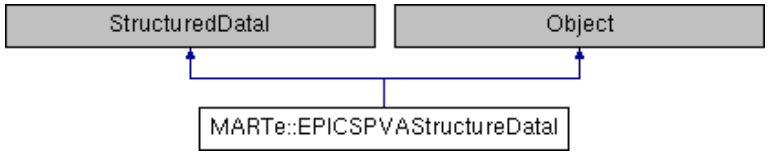
pvAccess - Objectives

- Monitoring
  - Asynchronously query value of a given set of (structured) variables
- PV Database
- (RPC) Messages
  - Commands
  - Configuration changes
- Input/output data source
  - Typically non-real-time
  - ITER => SDN for real-time networking

PVAStructuredDataI

- Wraps StructuredDataI as a PVStructure
  - Allows to use PVStructures directly with any MARTe components
- Can be used to e.g.
  - Serialise/deserialise MARTe over the network
  - Bootstrap MARTe applications from a PVStructture
- Perfect match with MARTe structured types
  - Navigation of arrays of structures were difficult to implement

```
bool MARTe::Object::Initialise(StructuredDataI &data)
```




```
EPICSPVAStructureDataI test;
test.InitStructure();
bool ok = test.CreateAbsolute("R");
ok &= test.CreateAbsolute("R.A");
ok &= test.CreateAbsolute("R.A.C[0]");
ok &= test.Write("a", 0);
ok &= test.CreateAbsolute("R.A.C[1]");
ok &= test.Write("a", 0);
ok &= test.CreateAbsolute("R.B");
ok &= test.Write("d", 1.0);
```

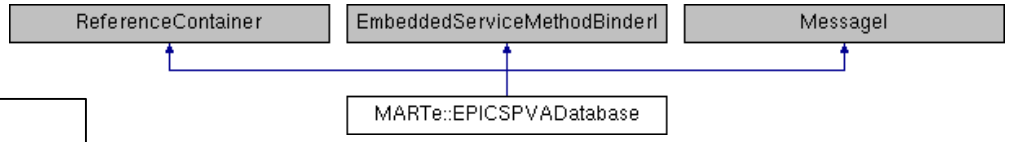
```
structure R
  structure A
    structure[] C
      structure
        int a
  structure B
    float d
```

```
...
virtual bool Initialise(MARTe::StructuredDataI &data) {
  bool ok = Object::Initialise(data);
  if (ok) {
    ok = data.MoveAbsolute("R.A.C[0]");
  if (ok) {
    ok = data.Read("a", gain1);
...
```

https://vcis-jenkins.f4e.europa.eu/job/MARTe2-Components-docs-master/doxygen/classMARTe_1_1EPICSPVAStructureDataI.html

PVA Database

- EPICSPVARecord server.



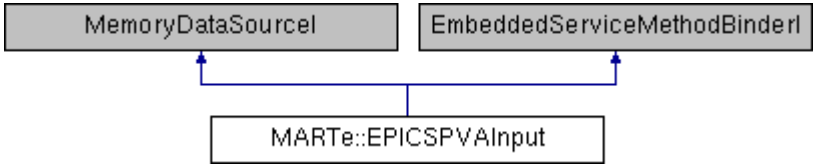

```
+EPICSPVADB = {
  Class = EPICSPVA::EPICSPVADatabase
  StackSize = 1048576
  CPUs = 0x2
  AutoStart = 0
  +FalconFastControllerStatistics = {
    Class = EPICSPVA::EPICSPVARecord
    Alias = "Falcon:Fast:Statistics"
    Structure = {
      value = {//name for pvget
      Type = FalconAppStatsStruct
    }
  }
  ...
```

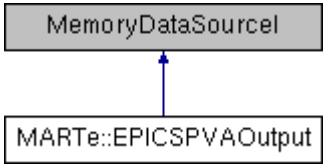
```
[netoa@next-4 Configurations]$ pvget Falcon:Fast:Statistics
Falcon:Fast:Statistics structure
    FalconAppStatsStruct value
        ThreadStats1Struct Offpulse
            uint Thread1C 1447
            uint[] Thread1H [0,0,0,0,0,0,0,0,0,37628]
            uint Thread2C 0
            uint[] Thread2H [37628,0,0,0,0,0,0,0,0,0]
        ThreadStats1Struct Ready
            uint Thread1C 0
            uint[] Thread1H [37628,0,0,0,0,0,0,0,0,0]
…
```

```
...
+ThreadStats1Struct = {
  Class = IntrospectionStructure
  Thread1C = {
    Type = uint32
  }
  Thread1H = {
    Type = uint32
    NumberOfElements = 10
  }
  ...
+FalconAppStatsStruct = {
  Class = IntrospectionStructure
  Offpulse = {
    Type = ThreadStats1Struct
  }
  Ready = {
    Type = ThreadStats1Struct
  }
  ...
```

https://vcis-jenkins.f4e.europa.eu/job/MARTe2-Components-docs-master/doxygen/classMARTe_1_1EPICSPVADatabase.html

PVADataSource





- As EPICSCA DataSources but with structured data

```
+EPICSPVAInput_1 = {
  Class = EPICSPVADataSource::EPICSPVAInput
  StackSize = 1048576
  CPUs = 0xff
  Signals = {
    RecordIn1Value = {//Record name if no Alias
      Alias = "alternative::channel::name"
      Field = "value" //If not set "value" is assumed
      Type = MyStruct1
    }
    RecordIn2 = {
      Field = "test"
      Type = uint32
      ...
```

```
+AGAM = {
  Class = MyGAM1
  InputSignals = {
    MySignal1 = {
      DataSource = EPICSPVAInput_1
      Alias = "RecordIn1Value.A.B"
      Type = float32
    }
    RecordIn2 = {
      DataSource = EPICSPVAInput_1
    }
…
```

https://vcis-jenkins.f4e.europa.eu/job/MARTe2-Components-docs-master/doxygen/classMARTe_1_1EPICSPVAInput.html
https://vcis-jenkins.f4e.europa.eu/job/MARTe2-Components-docs-master/doxygen/classMARTe_1_1EPICSPVAOutput.html

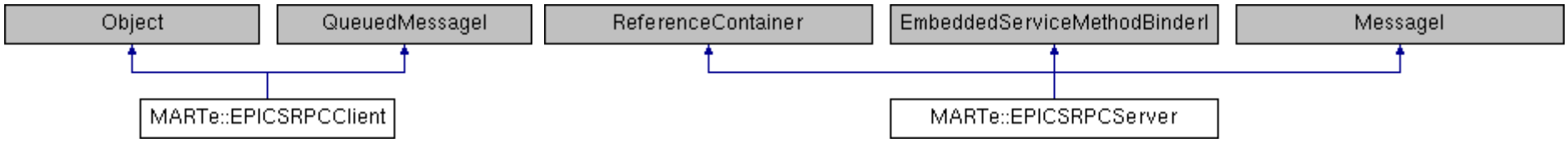
e.g. protocol proxy

- EPICS CA to PVA
- EPICS PVA to CA
- EPICS PVA to OPCUA
- OPCUA to EPICSPVA
- …

```
+EPICSPVAInput_1 = {
  Class = EPICSPVADataSource::EPICSPVAInput
  StackSize = 1048576
  CPUs = 0xff
  Signals = {
    RecordIn1Value = {//Record name if no Alias
      Alias = "alternative::channel::name"
      Field = "value" //If not set "value" is assumed
      Type = MyStruct1
    }
...
```

```
+OPCUAOut_1 = {
  Class = OPCUADataSource::OPCUADSOutput
  Address = "opc.tcp://localhost.localdomain:4840"
  Signals = {
    RecordIn1Value = {
      NamespaceIndex = 1
      Path = "value"
      Type = MyStruct1
    }
...
```

EPICSRPCClient & EPICSRPCServer



- Relay messages
  - EPICSPVAStructuredDataI serializes messages as PVStructures
  - Structure sent using an epics::pvAccess::RPCClient
- Can be used to e.g.
  - Command a remote application
  - Key component in SUP demo

- Container of MARTe Objects that implement the epics::pvAccess::RPCService interface
- Can be used to e.g.
  - Reconfigure an application based on a PVStructure
  - Query the current configuration as a PVStructure
  - Send messages to a running application

https://vcis-jenkins.f4e.europa.eu/job/MARTe2-Components-docs-master/doxygen/classMARTe_1_1EPICSRPCServer.html
https://vcis-jenkins.f4e.europa.eu/job/MARTe2-Components-docs-master/doxygen/classMARTe_1_1EPICSRPCClient.html

Conclusions & lessons learned

- CA & PVA data sources and interfaces integrated in the MARTe official release
  - Fully tested (coverage > 90 %)
  - Static code analysis
  - Documentation
- Components are already deployed in operational plants
  - Everything very stable once deployed
- Structures are extremely useful and key to the design of complex/scientific ITER plant systems
- Concerns about the PVA API
  - Not always clear what are the best practices to implement a given functionality
  - Arrays of structures were painful to implement

FUSION FOR ENERGY

Thank you for your attention

Follow us on:

www.f4e.europa.eu

www.twitter.com/fusionforenergy

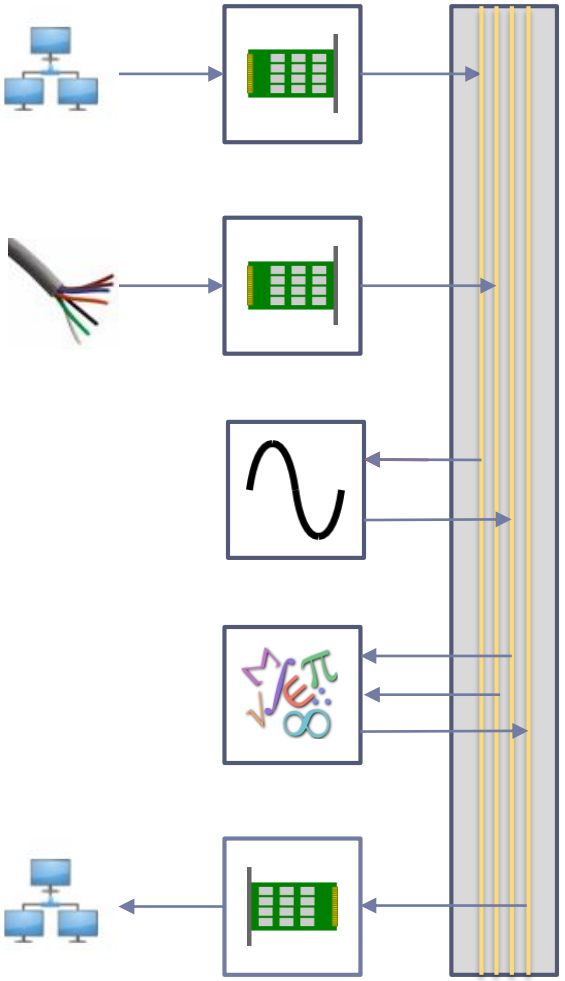
www.youtube.com/fusionforenergy

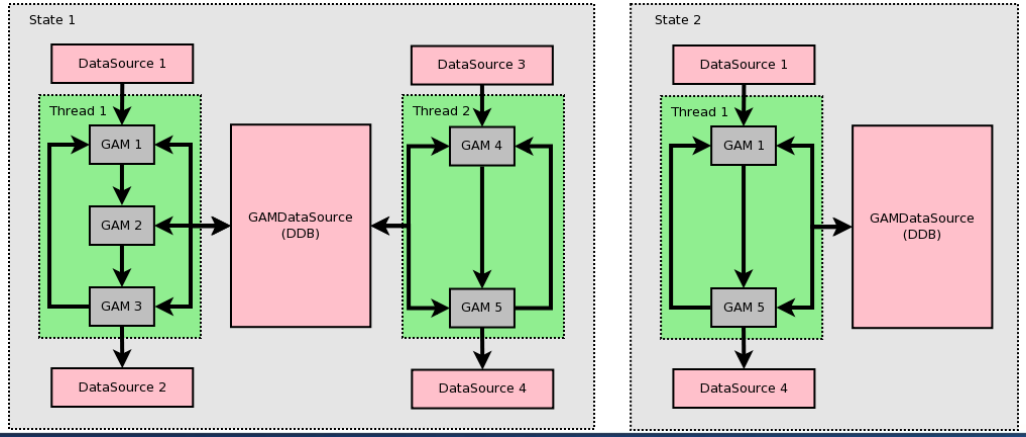
www.linkedin.com/company/fusion-for-energy

www.flickr.com/photos/fusionforenergy

GAMDataSource (aka DDB)

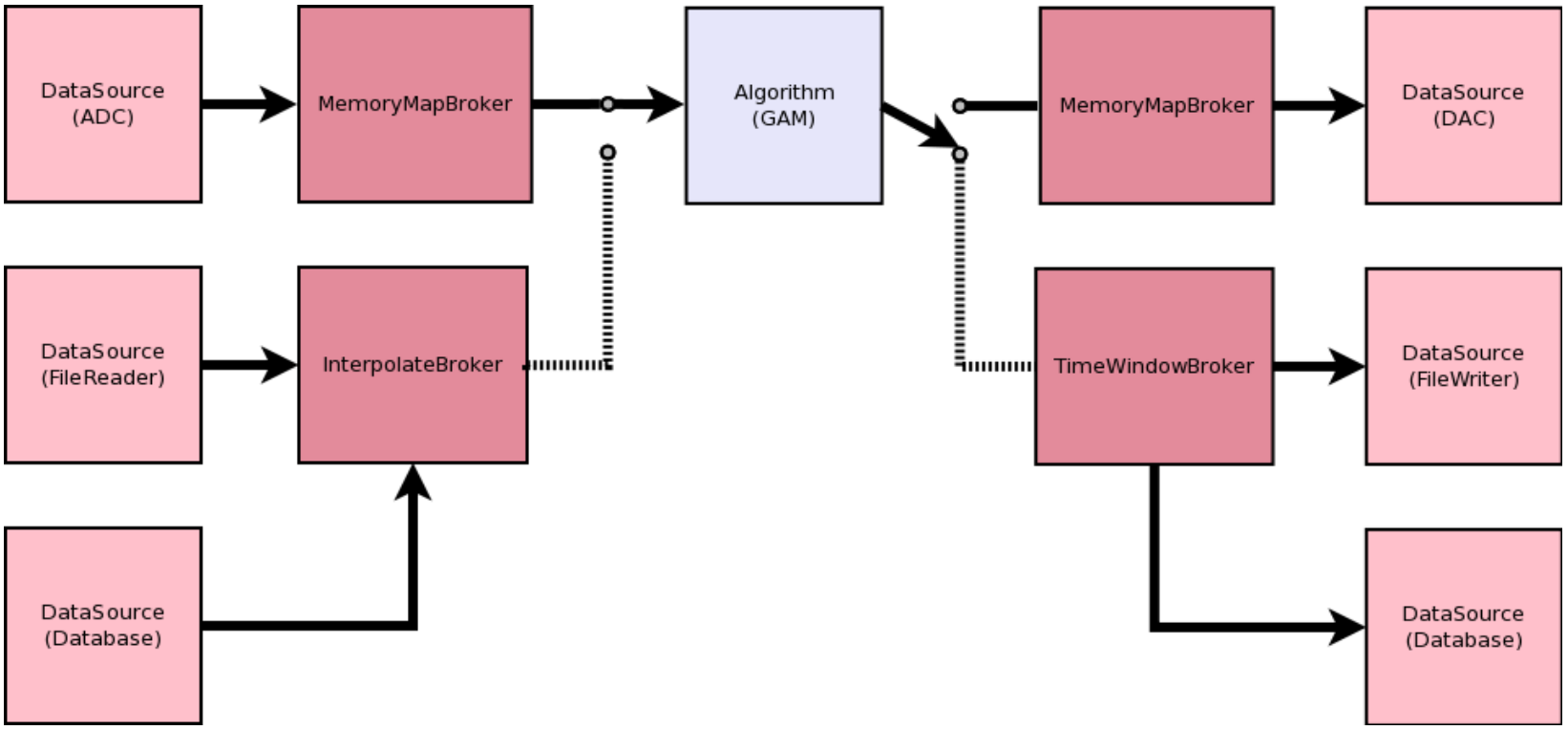
- GAMs share data through a memory bus
- MARTe guarantees coherency between requested and produced signals
- Set of GAMs allow to stream data to different MARTe systems
  - Distributed control systems





Brokers

- Interface between the GAMs memory and the DataSource hardware data (typically memory).
- Broker selected by DataSource based on GAM requirements (e.g. time base period)



Source code ptrs

https://vcis-gitlab.f4e.europa.eu/aneto/MARTe2-components/blob/master/Source/Components/Interfaces/EPICSPVA/EPICSPVAStructureDataI.cpp

https://vcis-gitlab.f4e.europa.eu/aneto/MARTe2-components/blob/master/Source/Components/Interfaces/EPICSPVA/EPICSPVAHelper.cpp

https://vcis-gitlab.f4e.europa.eu/aneto/MARTe2-components/blob/master/Source/Components/Interfaces/EPICSPVA/EPICSRPCClientMessageFilter.cpp

https://vcis-gitlab.f4e.europa.eu/aneto/MARTe2-components/blob/master/Source/Components/DataSources/EPICSPVA/EPICSPVAInput.cpp

https://vcis-gitlab.f4e.europa.eu/aneto/MARTe2-components/blob/master/Source/Components/DataSources/EPICSPVA/EPICSPVAOutput.cpp

https://vcis-gitlab.f4e.europa.eu/aneto/MARTe2-components/blob/master/Source/Components/Interfaces/EPICSPVA/EPICSRPCServer.cpp

https://vcis-gitlab.f4e.europa.eu/aneto/MARTe2-components/blob/master/Source/Components/Interfaces/EPICSPVA/EPICSPVADatabase.cpp

https://vcis-gitlab.f4e.europa.eu/aneto/MARTe2-components/blob/master/Source/Components/Interfaces/EPICS/EPICSPV.cpp

https://vcis-gitlab.f4e.europa.eu/aneto/MARTe2-components/blob/master/Source/Components/Interfaces/EPICS/EPICSCAClient.cpp

https://vcis-gitlab.f4e.europa.eu/aneto/MARTe2-components/blob/master/Source/Components/DataSources/EPICSCA/EPICSCAInput.cpp

https://vcis-gitlab.f4e.europa.eu/aneto/MARTe2-components/blob/master/Source/Components/DataSources/EPICSCA/EPICSCAOutput.cpp

And the numbers are…

Item	Lines of code
Core	47 k
Core (test)	138 k
Official components	28 k
Official components (test)	122 k

- For every unit of development expect:
  - ~4.5x of QA
  - ~0.3x of QA review
- For every new release expect:
  - 1 day of QA