MARTe Live Tutorial

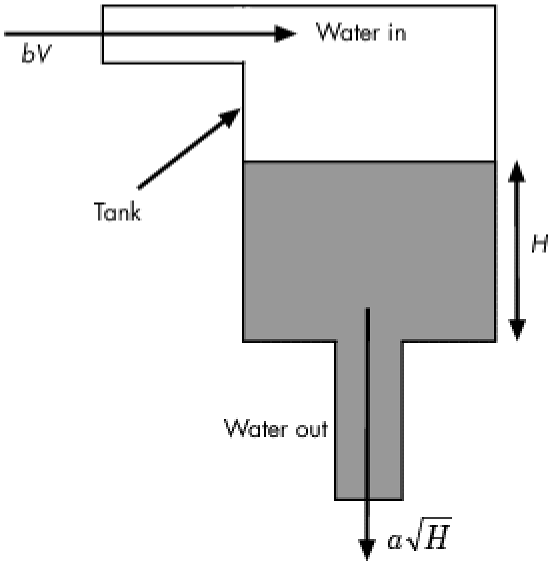
Building a MARTe application

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

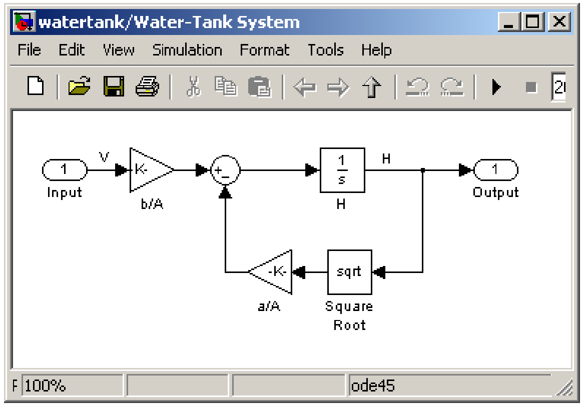
*Instituto de Plasmas e Fusão Nuclear
Instituto Superior Técnico
Lisbon, Portugal
http://www.ipfn.ist.utl.pt

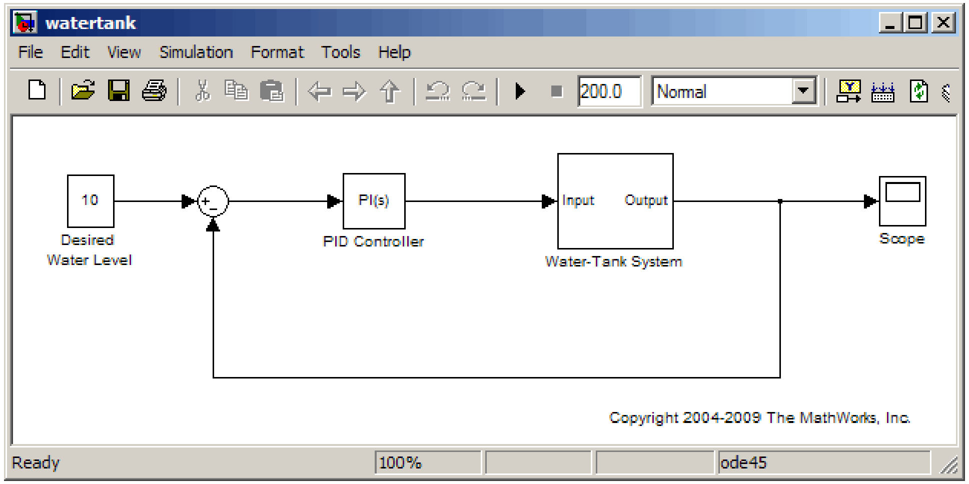
+JET-EFDA
Culham Science Centre, UK
http://www.jet.efda.org/

The water tank



$$\frac{d}{dt}Vol = A\frac{dH}{dt} = bV - a\sqrt{H}$$

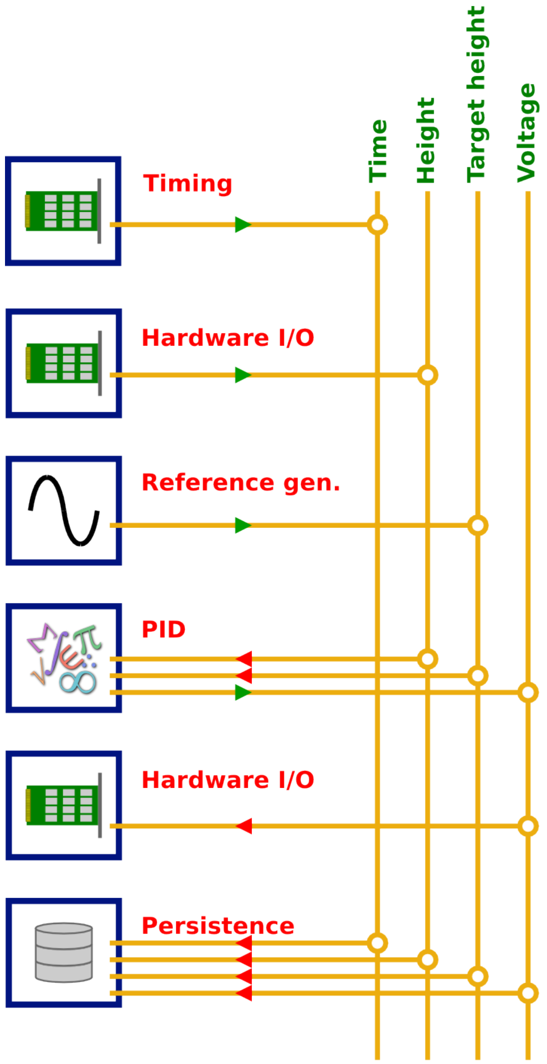

watertank/Water-Tank System
File Edit View Simulation Format Tools Help
1
Input
V
-K-
b/A
+
-
1/s
H
H
1
Output
-K-
a/A
sqrt
Square Root
100%
ode45

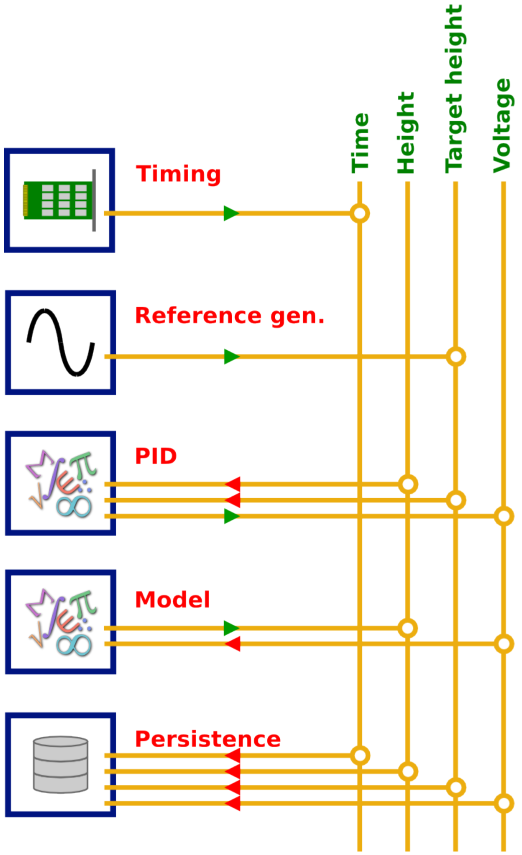


Copyright 2004-2009 The MathWorks, Inc.

- Vol – volume of water in tank
- A – cross-sectional area of water in tank
- b – constant related to flow rate into the tank
- a – constant related to flow rate out of the tank
- H – height of water

What GAMs for the water tank?

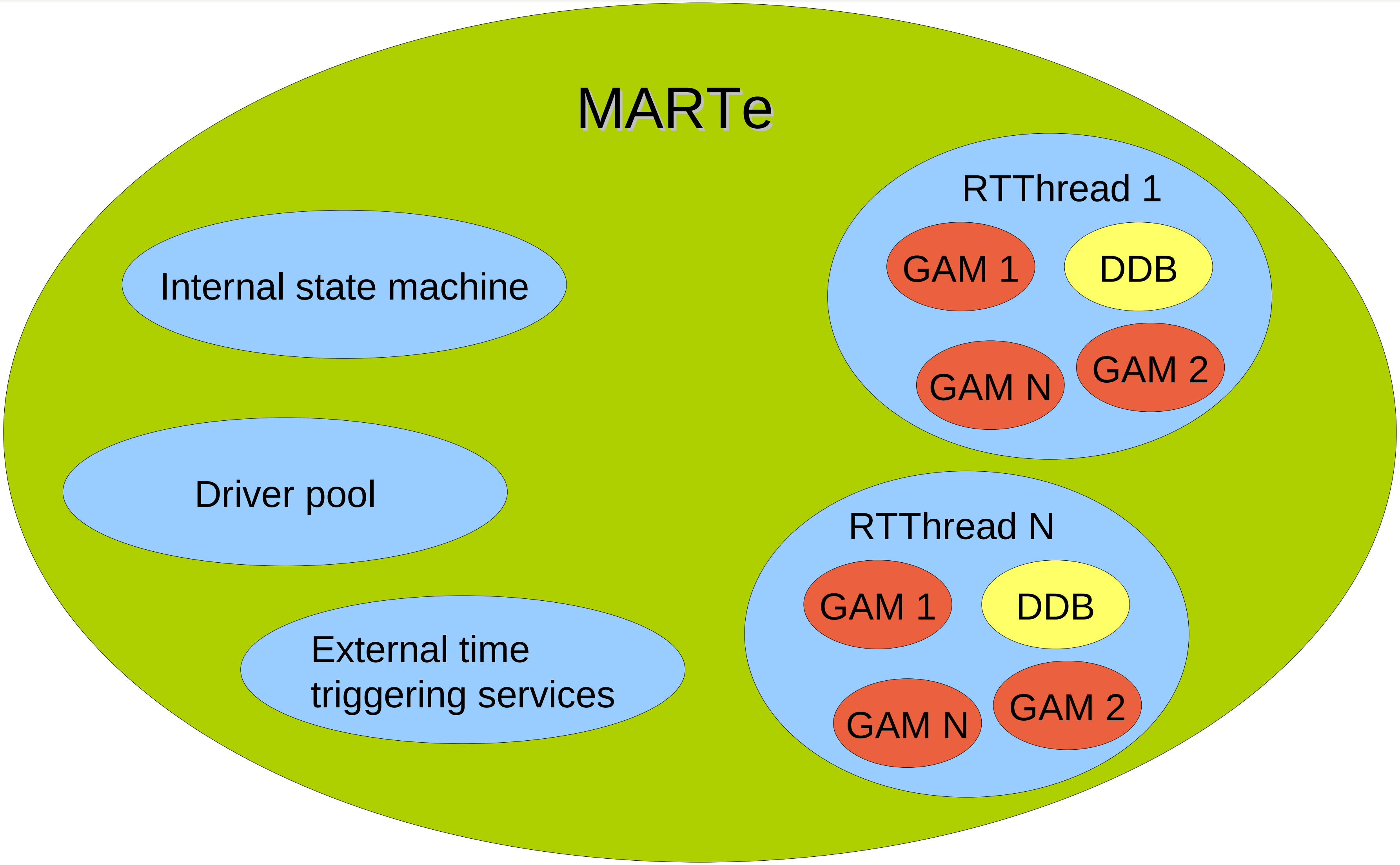




Development cycle

- What are my needs?
  - – Interfaces to hardware
  - – Algorithm execution
  - – Plant simulation
  - – Connection between components (DDB)
  - – Interfaces to outside world
- What do I have ready to be used?
  - – Recycle hardware interfaces
  - – Reuse algorithms

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

Requirements for our goal

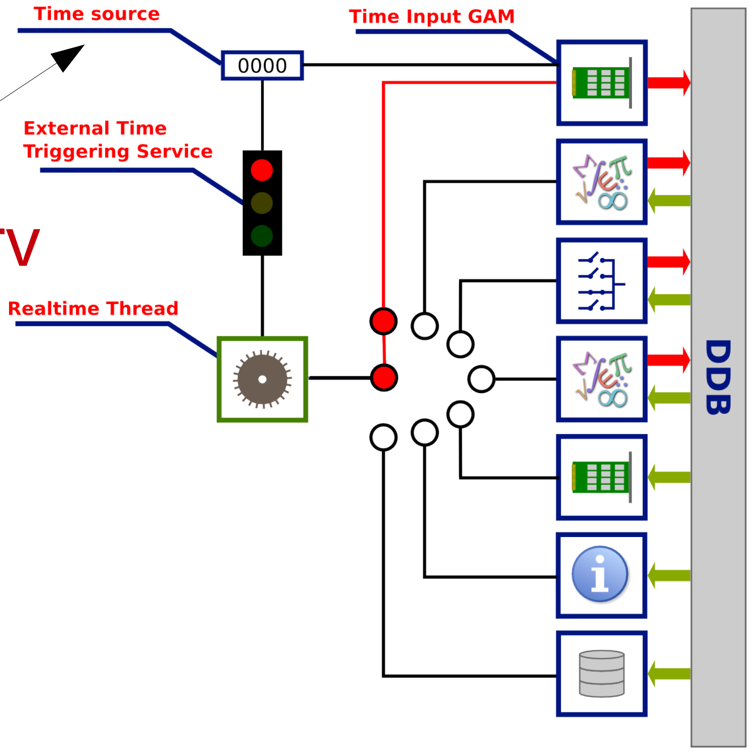
- Development of a water tank simulator
  - Time provider (timer)
  - Reference generation
  - A GAM with the model of the plant
    - water tank
    - pump power supply
  - PID
  - Data downloading
  - External triggering of the state machine

Skeleton configuration file

```
+MARTe = {
    Class = MARTeContainer
    StateMachineName = StateMachine
    MenuContainerName = MARTe
    +DriverPool = {...}
    +Messages = {...}
    +ExternalTimeTriggeringService = {...}
    +Thread_1 = {...}
}
```

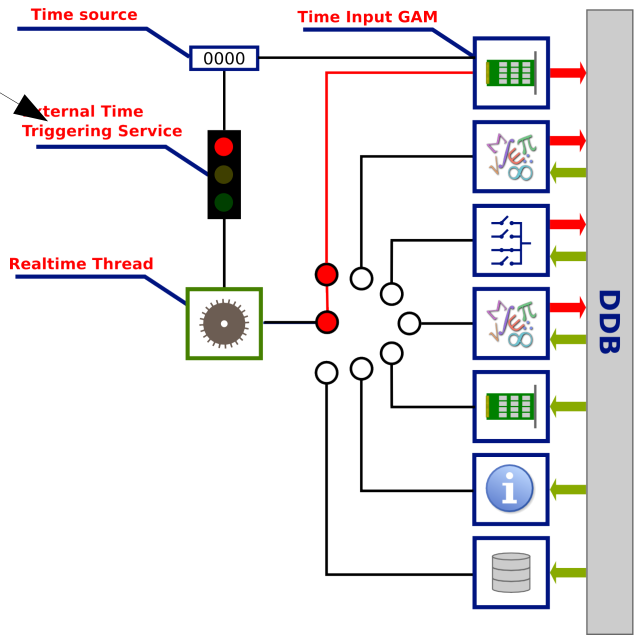
Generic Timer

```
+MARTe = {
    ...
    +DriverPool = {
        +TimerBoard = {
            Class = GenericTimerDrv
            NumberOfInputs = 2
            NumberOfOutputs = 0
            TimerUsecPeriod = 250
        }
    }
    ...
}
```



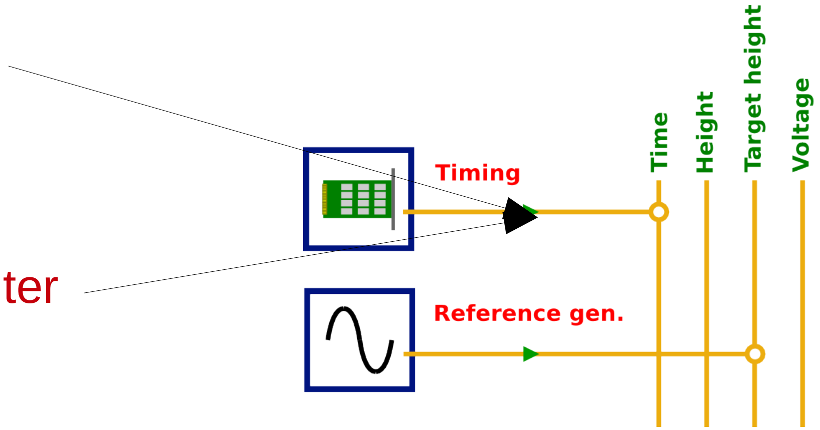
External Time & Trigger

```
+MARTe = {
    ...
    +ExternalTimeTriggeringService = {
        Class = InterruptDrivenTTS
        TsOnlineUsecPeriod = 250
        TsOnlineUsecPhase = 0
        TsOfflineUsecPeriod = 10000
        TsOfflineUsecPhase = 0
        TimeModule = {
            BoardName = TimerBoard
        }
    }
    ...
}
```



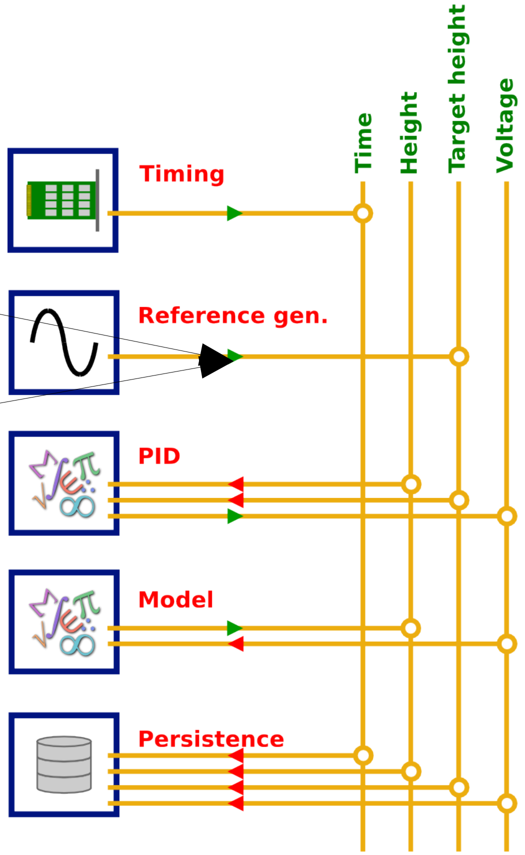
IOGAM (Timer)

```
+Thread_1 = {
        ...
        +Timer = {
                Class = IOGAMs::TimeInputGAM
                TriggeringServiceName = ExternalTimeTriggeringService
                Signals = {
                        time = {
                                SignalName = usecTime
                                SignalType = int32
                        }
                        counter = {
                                SignalName = timerCounter
                                SignalType = int32
                        }
                }
        }
        ...
}
```



Reference Generator

```
+Thread_1 = {
        ...
        +WaveformGen = {
                Class = WaveformGenerator
                UsecTime = usecTime
                +waterHeightReference = {
                        Class = WaveformClassSine
                        Frequency = 0.1
                        Gain = 1
                        Offset = 2.5
                }
                +zeroSignal = {
                        Class = WaveformClassPoints
                        TimeVector = {0 1}
                        ValueVector = {0 0}
                        Frequency = 1
                }
        }
        ...
}
```



PID GAM (1)

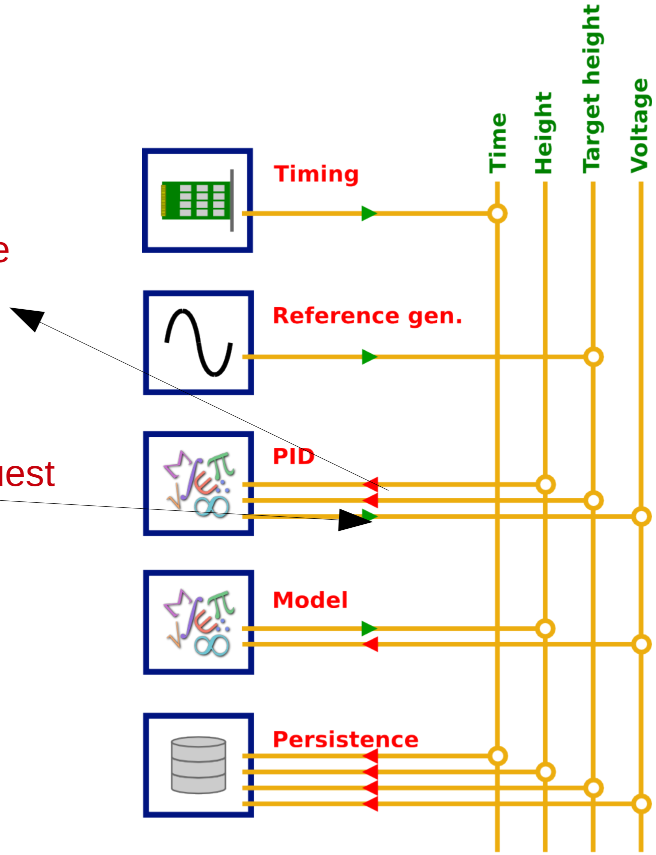
```
+Thread_1 = {
        ...
        +PIDGAM = {
                Class = PIDGAM
                TStart = 0.0
                TEnd = 10000.0
                InputSignals = {
                        PIDInput = {
                                SignalName = PIDIn
                                SignalType = PIDGAMInputStructure
                                FlatNamed = True
                        }
                }
                OutputSignals = {
                        PIDOutput = {
                                SignalName = PIDOut
                                SignalType = PIDGAMOutputStructure
                                FlatNamed = True
                        }
                }
                ...

        }
        ...
}
```

```
struct PIDGAMInputStructure {
        /** Time signal */
        int32                   usecTime;
        /** Reference signal to be
followed */
        float                   reference;
        /** Measurement signal */
        float                   measurement;
        /** Feedforward control */
        float                   feedforward;
};
```

PID GAM (2)

```
+Thread_1 = {
        +PIDGAM = {
                ...
                Remappings = {
                        InputInterface = {
                                usecTime = usecTime
                                reference = waterHeightReference
                                measurament = waterHeight
                                feedforward = zeroSignal
                        }
                        OutputInterface = {
                                controlSignal = pumpVoltageRequest
                                feedback = pumpVoltageRequest
                                error = pidHeightError
                                integratorState = pidIntState
                        }
                }
                Kp = 3.00
                Ki = 2.00
                Kd = 0.20
                SamplingTime = 0.001
                ControllerOn = On
        }
        ...
```



Water Tank model

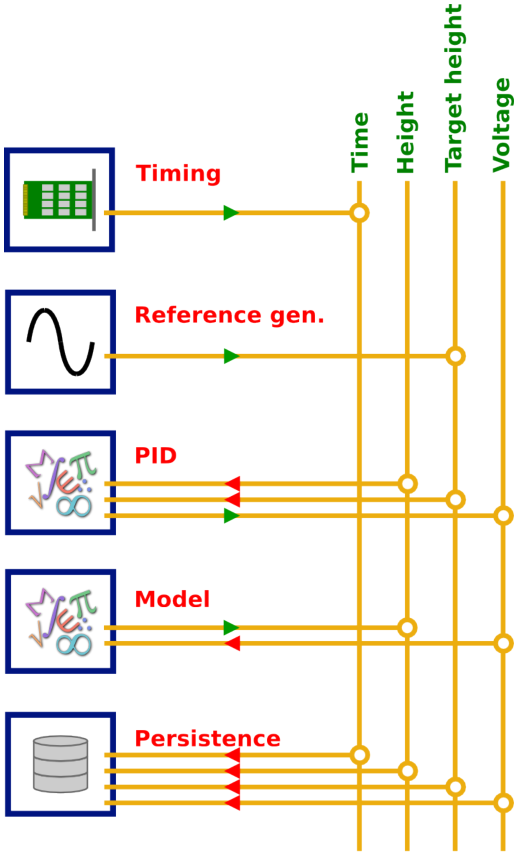
- Only GAM not readily available
- GAM development cycle
  - Design algorithm
    - Piece of paper
    - Software (matlab, octave, ...)
  - Decide inputs and outputs
  - What parameters are configurable?
    - What parameters are compulsory?

Water Tank variables

```
class WaterTank : public GAM, public HttpInterface {
...
// Parameters
private:
        /** Last usec time (for the integral) */
        int32                                    lastUsecTime;
        /** Last water height (for the integral) */
        float                                   lastHeight;
        /** Last voltage value after saturation*/
        float                                   lastVoltage;
        /** The input flow rate constant*/
        float                                   bFlowRate;
        /** The output flow rate constant */
        float                                   aFlowRate;
        /** Tank area */
        float                                   tankArea;
        /** Maximum voltage that can be requested */
        float                                   maxVoltage;
...
};
```

What input/output signals?

- Input
  - Time
  - Requested voltage from PID
- Output
  - Water height



Water Tank read config.

```
bool WaterTank::Initialise(ConfigurationDataBase& cdbData) {
        if(!AddInputInterface(input,"InputInterface")){
                AssertErrorCondition(InitialisationError, "WaterTank::Initialise: %s failed to add input
interface", Name());
              return False;
        }
...
        if(!cdb->Move("InputSignals")){
                AssertErrorCondition(InitialisationError, "WaterTank::Initialise: %s did not specify
InputSignals entry", Name());
              return False;
        }
...
        if(!cdb.ReadFloat(aFlowRate, "aFlowRate", 20)){
                AssertErrorCondition(Information, "WaterTank %s::Initialise: output flow rate not
specified. Using default %f", Name(), aFlowRate);
       }
...
        if(!cdb.ReadFloat(tankArea, "TankArea", 20)){
                AssertErrorCondition(Information, "WaterTank %s::Initialise: tank area not specified.
Using default %f", Name(), aFlowRate);
       }
}
```

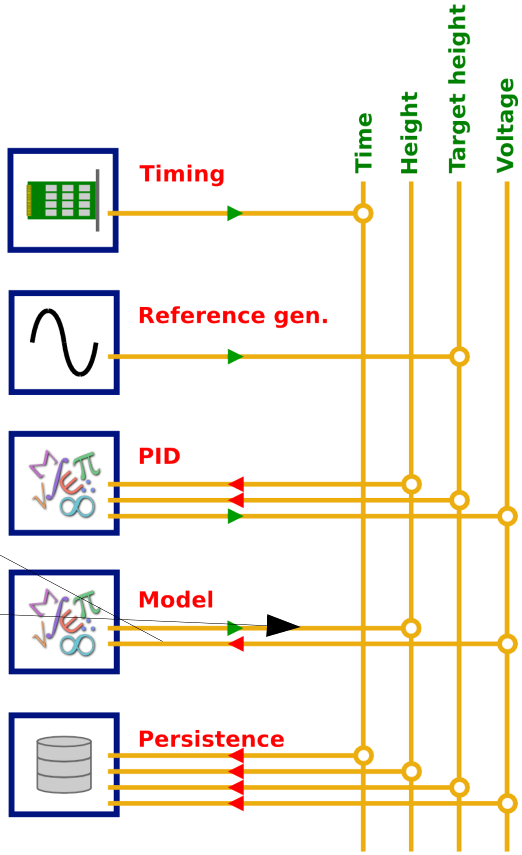
Input signals from DDB

Water Tank execution

```
bool WaterTank::Execute(GAM_FunctionNumbers functionNumber) {
        // Get input and output data pointers
        input->Read();
        int32 usecTime  = *((int32*)input->Buffer());
        float voltage       = ((float *)input->Buffer())[1];
        float *outputBuff = (float*)  output->Buffer();
        float height          = 0;
...
        //Saturate voltage
        if(voltage > maxVoltage){
                voltage = maxVoltage;
        }
        if(voltage < minVoltage){
                voltage = minVoltage;
        }
        //simple Euler method
        height   = (voltage * bFlowRate - aFlowRate * sqrt(lastHeight)) / tankArea * (usecTime -
lastUsecTime) * 1e-6 + lastHeight;
        lastHeight   = height;
...
        *outputBuff  = height;
...
        // Update the data output buffer
        output->Write();
}
```

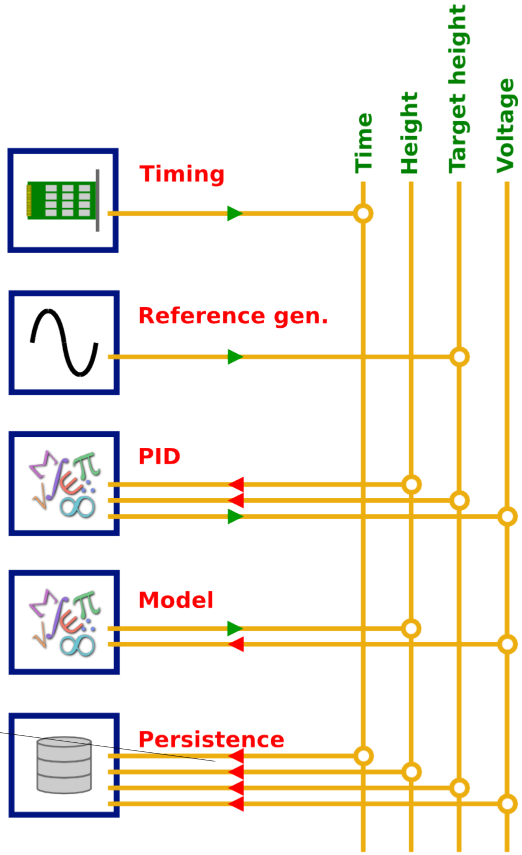
Water tank config.

```
+Thread_1 = {
        +WaterTank = {
                Class = WaterTank
                InputSignals = {
                        usecTime = {
                                SignalName = usecTime
                                SignalType = int32
                        }
                        voltage = {
                                SignalName = pumpVoltageRequest
                                SignalType = float
                        }
                }
                OutputSignals = {
                        height = {
                                SignalName = waterHeight
                                SignalType = float
                        }
                        pumpVoltage = {
                                SignalName = pumpVoltage
                                SignalType = float
                        }
                }
                aFlowRate = 20.0
                TankArea = 20.0
                …
```



Data collection

```
+Thread_1 = {
        ...
        +Collection = {
                Class = CollectionGAMs::DataCollectionGAM
                EventTrigger = {
                        TimeWindow0 = {
                                NOfSamples = 40000
                                UsecPeriod = 250
                        }
                }
                Signals = {
                        CLOCK = {
                                SignalName = usecTime
                                JPFName = “TIME”
                                SignalType = int32
                        }
                        WaterHeight = {
                                SignalName = waterHeight
                                JPFName = “WaterHeight”
                                SignalType = float
                        }
                        ...
```



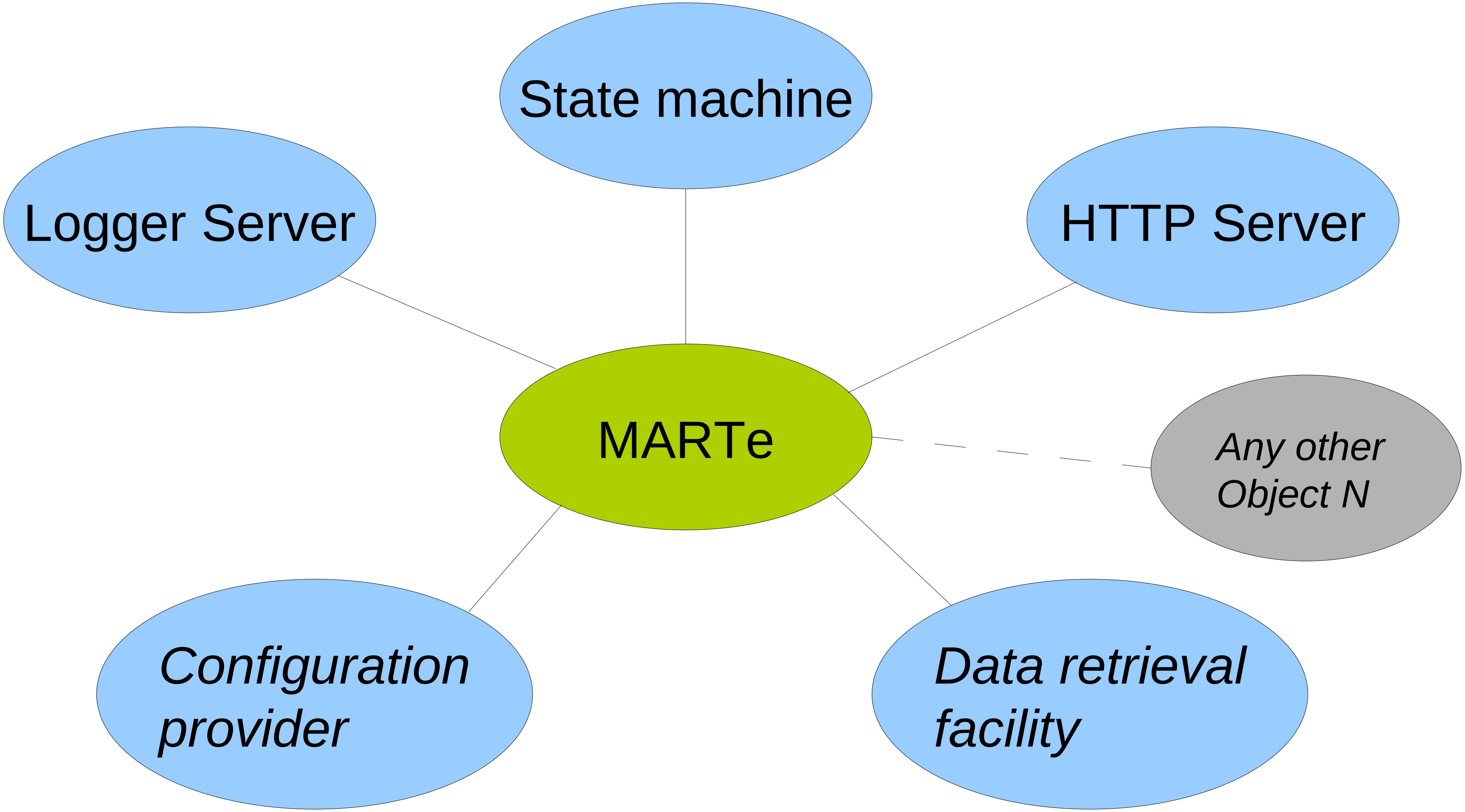
Execution order

- MARTe has two runtime cycles
- Online is associated with the real-time cycle, whereas offline is the stand-by mode
- GAMs can be Online *forever*

Execution order

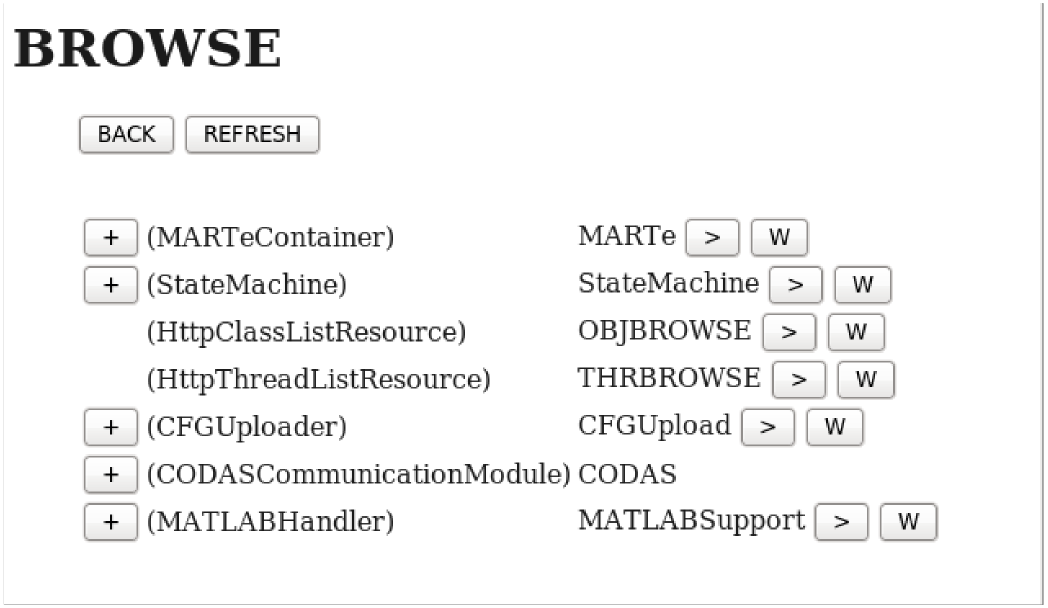
```
+Thread_1 = {
        ...
        Online = "Timer WaveformGen PIDGAM WaterTank Statistic Collection"
        Offline = "Timer Collection"
}
```

MARTe Universe



MARTe Universe components

```
LoggerAddress = localhost
DefaultCPUs = 8
+HTTPSERVER= {
        Class = HttpService
        Port = 8084
        Root = WEB
}
+WEB= {
        Class = HttpGroupResource
        +BROWSE = {
                Class = HttpGCRCBrowser
                Title = ”Http Object browser”
                AddReference = ”MARTe
StateMachine OBJBROWSE THRBROWSE
CFGUpload MATLABSupport”
        }
}
+MATLABSupport = {
    Class = MATLABHandler
}
+CFGUpload = {
    Class = CFGUploader
}
+StateMachine = {
...
```

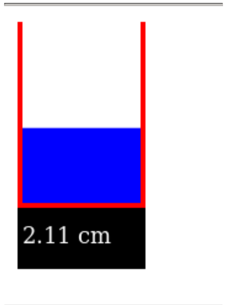


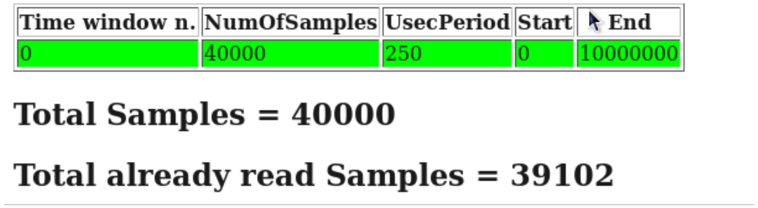
BROWSE

+	(MARTeContainer)	MARTe	>	W
+	(StateMachine)	StateMachine	>	W
	(HttpClassListResource)	OBJBROWSE	>	W
	(HttpThreadListResource)	THRBROWSE	>	W
+	(CFGUploader)	CFGUpload	>	W
+	(CODASCommunicationModule)	CODAS		
+	(MATLABHandler)	MATLABSupport	>	W

GAMs and HTTP interface

- GAMs may expose information about themselves using the HTTP interface
  - Write to a stream facility which is provided every time an HTTP request for their URL is performed

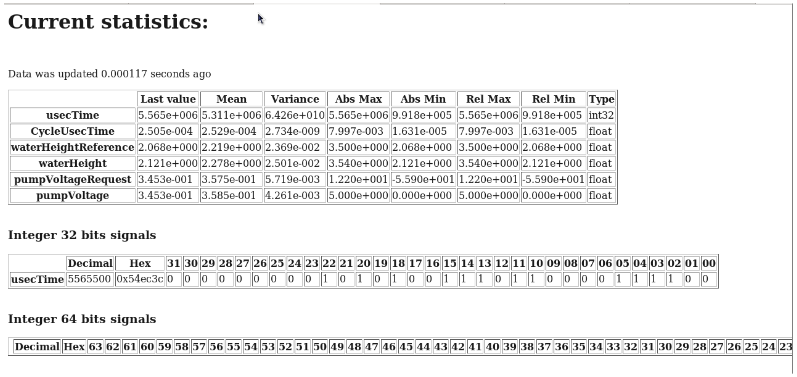




Time window n.	NumOfSamples	UsecPeriod	Start	End
0	40000	250	0	10000000

Total Samples = 40000

Total already read Samples = 39102



Current statistics:

Data was updated 0.000117 seconds ago

	Last value	Mean	Variance	Abs Max	Abs Min	Rel Max	Rel Min	Type
usecTime	5.565e+006	5.311e+006	6.426e+010	5.565e+006	9.918e+005	5.565e+006	9.918e+005	int32
CycleUsecTime	2.505e-004	2.529e-004	2.734e-009	7.997e-003	1.631e-005	7.997e-003	1.631e-005	float
waterHeightReference	2.068e+000	2.219e+000	2.369e-002	3.500e+000	2.068e+000	3.500e+000	2.068e+000	float
waterHeight	2.121e+000	2.278e+000	2.501e-002	3.540e+000	2.121e+000	3.540e+000	2.121e+000	float
pumpVoltageRequest	3.453e-001	3.575e-001	5.719e-003	1.220e+001	-5.590e+001	1.220e+001	-5.590e+001	float
pumpVoltage	3.453e-001	3.585e-001	4.261e-003	5.000e+000	0.000e+000	5.000e+000	0.000e+000	float

Integer 32 bits signals

	Decimal	Hex	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
usecTime	5565500	0x54ec3c	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	1	1	1	0	1	1	0	0	0	0	1	1	1	1	0	0

Integer 64 bits signals

Decimal	Hex	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23

Compiling the example

- Go to MARTe directory and run:
  - make -f Makefile.<os>
  - Where <os> is the operating system to compile (linux, vx5100, msc, ...)
- Go to the WaterTank example and compile
  - cd <MARTe directory>/GAMs/WaterTank

Running the example

- Depends on the operating system
- For linux a bash script is provided
  - Points to all the shared libraries and exports the LD_LIBRARY_PATH
  - Starts MARTe with the desired configuration file
- Other operating systems
  - Collect all required binaries
  - Load in memory (if required)
  - Start MARTe with the desired configuration file

The example running (LIVE)

- Linux
  - http://pc-rtdn-off-09.jet.uk:8084/BROWSE/
- RTAI
  - http://pc-rtdn-off-08.jet.uk:8084/BROWSE/
- VxWorks
  - http://vx-rtdn-off-00.jet.uk:8084/BROWSE/
- Run a sequence
  - GAMs HTTP output
  - Download data
  - Look at data with octave

Backup slides