Coupling DCS and MARTe: two real-time control frameworks in collaboration

Christopher Rapson[a], Pedro Carvalho[b], Klaus Lüddecke[c], André Neto[c], Bruno Santos[b], Wolfgang Treutterer[a], Axel Winter[d], Thomas Zehetbauer[a]

[a]*Max Planck Institute for Plasma Physics, Boltzmannstrasse 2, 85748 Garching, Germany*
[b]*Instituto de Plasmas e Fusão Nuclear, Instituto Superior Técnico, Universidade de Lisboa, 1049-001 Lisboa, Portugal.*
[c]*Unlimited Computer Systems GmbH, Seeshaupterstr. 15, 82393 Iffeldorf, Germany*
[d]*ITER Organization, Route de Vinon-sur-Verdon, 13115 St-Paul-Lès-Durance, France*

Abstract

Fusion experiments place high demands on real-time control systems. Within the fusion community two modern framework-based software architectures have emerged as powerful tools for developing algorithms for real-time control of complex systems while maintaining the flexibility required when operating a physics experiment. The two frameworks are known as DCS (Discharge Control System), from ASDEX Upgrade and MARTe (Multithreaded Application Real-Time executor), originally from JET.

Based on the success of DCS and MARTe, ITER has chosen to develop a framework architecture for its Plasma Control System which will adopt major design concepts from both the existing frameworks. This paper describes a coupling of the two existing frameworks, which was undertaken to explore the degree of similarity and compliance between the concepts, and to extend their capabilities. DCS and MARTe operate in parallel with synchronised state machines and a common message logger. Configuration data is exchanged before the real-time phase. During the real-time phase, structured data is exchanged via shared memory and an existing DCS algorithm is replicated within MARTe. The coupling tests the flexibility and identifies the respective strengths of the two frameworks, providing a well-informed basis on which to move forward and design a new ITER real-time framework.

Keywords: control system, real time, software framework, ITER, MARTe, DCS

1. Introduction

Fusion experiments, and in the future fusion power plants, require fast ($\tau < 10^{-2}$ s) real-time control and coordination of many complex subsystems. Experiments, by their nature, have the additional requirement of high flexibility to facilitate many different physics or technical experiments with minimal overhead. For the control software, a framework architecture - where applications are designed based on templates and generic base libraries encourage code re-use - is well suited to fulfil these requirements. The potential has been demonstrated by two control frameworks which have been developed independently within the fusion community - DCS (Discharge Control System) [1] was deployed at ASDEX Upgrade in 2003 and MARTe (Multithreaded Application Real-Time executor) [2] at JET in 2008. Both of these frameworks have been developed with emphasis on different aspects of tokamak control. DCS has implemented many supervisory functions and is designed to distribute controllers over multiple nodes, to facilitate the complete integrated control of a tokamak, whereas MARTe emphasized the easy creation of applications in a lean yet powerful infrastructure. ITER has recognised the benefits of both of these approaches, as they cover some of the fundamental aspects of the ITER control needs. One of the key differences of ITER compared to present-day devices is the pulse duration which can be up to 3000 seconds. This imposes a novel set of requirements on a potential framework, for example the possibility of event handling and the flexible adaptation of the pulse schedule based on the current state of the **machine [3, 4, 5].**

With this in mind, in early 2013 the ITER controls group has asked the fusion community to come together and help in designing a novel framework which is based on the experience and knowledge emerging from the MARTe and DCS developments and fulfils the addi-

Email address: `chris.rapson@ipp.mpg.de` (Christopher Rapson) tional requirements of ITER.

As a first step in the development of the new framework, DCS and MARTe have been brought together to operate side-by-side. This coupling is the main topic of this paper. The coupling tests the flexibility of DCS and MARTe, and provides a direct comparison of their features. This helps to identify the state-of-the-art, and showcase the concepts on which the ITER real-time framework will be built. Note that since the specifications for the new framework are still under discussion, they will not be addressed here.

The coupling is not intended as an optimal technical solution, since operating two frameworks inherently duplicates functionality and overhead. However, it has technical value in that it allows collaborators to directly apply algorithms at ASDEX Upgrade which were originally developed using MARTe at other experiments.

As an additional motivation, the human aspect should not be underestimated, since DCS and MARTe had previously been developed in relative isolation. Bringing the two teams together early in the development process facilitates an exchange of ideas, building working relationships and allows the new framework to leverage the shared expertise of a united community.

In the following, Section 2 outlines the common features as well as the differences between the frameworks, and how they affected the coupling. Section 3 describes the interfaces which were designed to adjust for the differences, and allow DCS and MARTe to operate together. Section 4 describes the implementation of a specific algorithm, which was used to test the interfaces and then demonstrated live at the ITER offices. Section 5 presents the conclusions and an outlook on the next steps to develop a real-time control framework for ITER.

2. Similarities and Differences between DCS & MARTe

DCS and MARTe share many common features. First and foremost is their modular design. Generic Application Modules (GAMs) in MARTe, or Application Processes (APs) in DCS are computation modules which are chained together to perform the real-time control. Modules inherit much of their functionality from a library of base classes, which naturally suggest the use of an object-oriented language. For performance reasons, both DCS and MARTe use C++. Communication between GAMs, or between APs, is done by passing signals. This is analogous to block diagrams [6] as commonly used in control systems design. The signals and modules in both frameworks are configuration driven, allowing them to be modified without having to recompile source code. This provides the flexibility to modify the control function between two plasma pulses (i.e. less than 20 minutes for ASDEX Upgrade and JET).

In addition to their architecture, the frameworks also have similar functional requirements. Both distinguish between real-time and non-real-time functionality, and co-ordinate the non-real-time tasks using a state machine. As expected, control algorithms run in real-time, and raise an appropriate exception when execution is delayed beyond an acceptable time limit. Both frameworks provide a message logging mechanism and data archiving which can be used to understand the pulse from a physics perspective, or to analyse the control system itself and improve the algorithms. So it is clear that there are many common concepts, which differ only in name or other details.

Nevertheless, some differences exist, arising mainly from the details of their respective origins. DCS was conceived from the start to be a holistic system, capable of performing all real-time relevant actions on ASDEX Upgrade. For example, DCS parses all commands from experiment leaders and checks boundary conditions from plant systems before a plasma pulse. During the pulse DCS calculates references and co-ordinates all plant systems. MARTe is an evolution of a real-time framework developed by the plasma operations group at JET [7]. Taking advantage of a major upgrade to the JET vertical stabilisation system [8], it was decided to create a new version of the framework that would ease the development and the commissioning of the new system and enable a large team of people to develop and contribute in parallel. Hence MARTe can be deployed as an independent module controlling a single plant system, requiring only reference trajectories as input. Reference trajectories can be set either through configuration before the experiment, or driven by an external entity, such as the real-time network. This reduced scope makes MARTe easy to understand and easy to deploy incrementally, which has led to its wide adoption in the fusion community. The coupling in this project makes use of the central control capabilities intrinsic to DCS, providing references and input data to MARTe.

In MARTe, one thread can consist of any number of GAMs. GAMs pass information from one to another by writing to and reading from a Dynamic Data Buffer (DDB). Each signal has a reserved space, whose address is communicated to the GAMs as part of their configuration. The configuration and initialisation processes check and guarantee that all signal subscribers have an existing and valid source. Since GAMs are executed sequentially, read/write conflicts are avoided. MARTe can use IOGAMs as interfaces to exchange data with different threads or instances of itself and with other hardware or software frameworks like DCS. Since DCS envisages that every module could potentially exchange data with every other module executed in a separate thread or even on a remote computation node, it uses a reflective memory approach in combination with a local buffer and semaphores to exchange signals. During configuration, a publish and subscribe network is configured to connect all requests for signals with their relevant producers and memory buffers are allocated for each signal. A buffer is only allocated as required, for instance two APs on the same node require only one copy. When a producer publishes a signal, it will be copied to all subscribers' buffers via reflective memory and the subscribers receive a semaphore to indicate that new data is available. The infrastructure for transferring signals in DCS is collectively referred to as the Signal Agent.

Since within DCS, all signals produced by an AP are available to any downstream process, it was found to be important to add metadata which indicates whether these signals can be trusted. This is useful information for all algorithms, but in particular for safety-relevant systems [9]. The metadata attached to all DCS signals consists of a timestamp, a confidence state and a production state. The confidence state indicates the integrity of the data, and the production state indicates the intention of the sample producer to generate new samples . MARTe modules traditionally exchange signals as scalars and multi-dimensional arrays of primitive data types without metadata. Since ITER requirements specify that all signals will carry metadata, it was one of the goals for this project to demonstrate how MARTe would potentially handle such structured data.

3. Interfaces

3.1. Existing Interfaces

A controller is only relevant in the context of a control system, so both frameworks have previously developed interfaces to several other systems. MARTe comes pre-packaged with interfaces to ethernet, shared memory and PCIe. There are also interfaces to EPICS (Experimental Physics and Industrial Control System) and ITER SDN (Synchronous Databus Network) **streams [10, 11, 12].** MARTe can run on different environments such as Gentoo®, Red Hat® and Fedora® core Linux distributions, Solaris®, VxWorks®, Mac® or Windows® and has been successfully deployed to control several aspects of different fusion devices across the world, e.g control of magnetohydrodynamic (MHD) instabilities at FTU [13], plasma position at COMPASS [14], and control of multiple subsystems at ISTTOK [15] and JET [16]. It is also a candidate for use in ITER Fast Plant System Controllers, with a prototype having been integrated into the ITER CODAC (COntrol, Data Access and Communication) Core System and connected to the EPICS framework [11].

DCS draws in measurement data directly from various analogue-to-digital converters, and drives some actuators using digital-to-analogue converters [17]. In addition, data can be transferred via an interface to or from a number of real-time diagnostics which can run on Linux®, Solaris® or LabVIEW® operating systems [18]. DCS itself has so far been tested on VxWorks® and RedHawk® Linux operating systems at the ASDEX Upgrade experiment. DCS will soon be adapted for use at a second experiment - the upgrade of Tore Supra known as WEST.

3.2. Non-Real-Time Coupling

Despite both frameworks being focussed on real-time control, the coupling of DCS and MARTe requires some functionality which is not time-critical. Before the pulse, configuration files are automatically synchronised to avoid maintaining duplicate settings. During the pulse, DCS notifies MARTe of any relevant state transitions via HTTP and MARTe confirms that the state machines are aligned. If DCS does not receive the appropriate confirmation within a 2 s timeout, DCS will abort the pulse and launch a "cleanup" to try and return to a well-defined state. The "abort" function in MARTe fulfils the same function, so a DCS "cleanup" was used to trigger a MARTe "abort". At all times, an extension to the MARTe relay logger reformats messages and forwards them to the DCS logger server, such that log messages from both frameworks can be visualised at the same place.

3.3. Real-Time Data Transfer

A central component of this collaboration was the ability to exchange information in real-time between DCS and MARTe. As noted above, DCS already includes a well-defined interface to "real-time diagnostics" which is relatively agnostic regarding the hardware and operating system to which it interfaces. It would have been possible to couple DCS & MARTe via this interface, with both systems well separated and even running on separate computers. Indeed, this would have been similar to how MARTe is deployed at other experiments, e.g. JET. However, in order to demonstrate a closer collaboration of the two systems, it was decided to run both frameworks on the same computer and

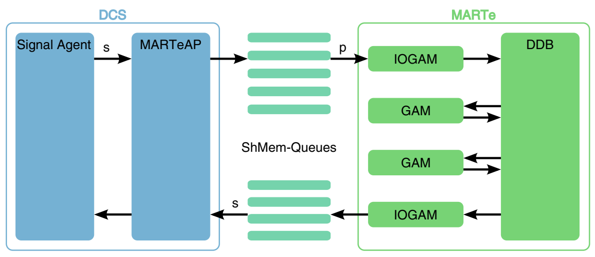


Figure 1: The path used to transfer signals in real-time from the DCS signal agent to MARTe GAMs for computation, and to return the results. Data transfer marked with 's' are triggered by semaphores, those marked with 'p' are polled. Unmarked arrows indicate that the data transfer happens sequentially, as soon as the previous step is completed.

exchange signals via shared memory (ShMem) queues. DCS typically uses ShMem to pass signals between APs running on the same computer, and MARTe was able to simply include the library in order to use the same access methods. It was decided to have one queue per signal, and for the queue management to be performed by MARTe. DCS only checks whether the queues are present during configuration. Queues have a limited size, and overflow detection. Locking is implemented for all reading and writing operations. Since there is only one "reader" per queue, data is removed after reading. Also, since data will be read almost instantly after being written (DCS waits for a semaphore and MARTe polls at 1 MHz) the queues are usually empty. Attempting to read an empty queue returns *false*, which MARTe uses as an indication to keep polling. Note that aside from data transfer, the ShMem queues provide a method to synchronise the two frameworks in real-time, since both wait for data from the other before proceeding.

Fig. 1 illustrates the path taken by data. On the DCS side of the queue, an AP called "MARTeAP" subscribes to signals from within the DCS real-time network. As an AP, it can make use of DCS' base functions to synchronise the signals and start execution as soon as all signals are available. It then writes the signals to the corresponding ShMem queue. On the MARTe side, an Input-Output GAM (IOGAM) polls the queue for a new data sample, and, when available, fetches all data from the ShMem queue to the DDB. The IOGAM implements the DCS driver that reads and converts DCS signals into MARTe's signals. DCS signals are a structure with 4 attributes (timestamp, confidence state, production state and value(s)) which are split into separate signals. The IOGAM defines one of the incoming signals as a timing source to synchronize the internal timing of both frameworks. The mapping between DCS signals and MARTe DDB signals is previously defined in MARTe configuration file as well as which signals will be used by which GAMs. To return signals to DCS, a MARTe IOGAM writes data from the DDB to other shared memory queues. The IOGAM is configured to combine 4 attributes which are separate signals in MARTe into one data structure for each DCS signal. The IOGAM includes the data structures definitions via a DCS header file. There are different structures for each of the different value types used: uint32, uint64, float, vector<float> and double. Metadata always has the same type.

As soon as MARTe writes its outputs to the queues, the MARTeAP receives a semaphore for each queue, reads in the signals, and distributes them over the DCS real-time network. If a response is delayed, the MARTeAP will wait indefinitely. The DCS alarm system detects the delay and will raise an alarm if real-time constraints are violated.

This design is generic and scalable; in principle it would allow several MARTeAPs to communicate with several IOGAMs.

4. Implementation and Results

To demonstrate the interface described above, it was decided to replicate an AP from ASDEX Upgrade within MARTe. Then, both the AP and its replica can run in parallel and the results can be compared. The AP chosen was the MHD_Evaluator, which accumulates information on MHD instabilities from several diagnostics and co-ordinates actuator responses. This is one of the most active areas of development within DCS, as part of the program to stabilise Neoclassical Tearing Modes (NTMs). In the context of this demonstration, the details of NTM control is not important, and a large body of literature is available to the interested reader [e.g. 19]. The demonstration used a "Replay Mode" to reproduce the input data that DCS received during a specific ASDEX Upgrade pulse. The Replay Mode is useful to conduct tests without being connected to a real experiment, and allows software development under fully reproducible and realistic conditions. Note that all DCS processes (in particular those required as part of the framework infrastructure) are executed as normal, although only the outputs from the MHD_Evaluator AP and its replica in MARTe are shown here.

The MHD_Evaluator algorithm was implemented in MARTe, with the functionality modularised to use 4 GAMs: The first is a low pass filter which is applied to Mirnov coil measurements ~~**of NTM amplitudes for mode numbers 2/1 and 3/2**~~ (one GAM is used twice),

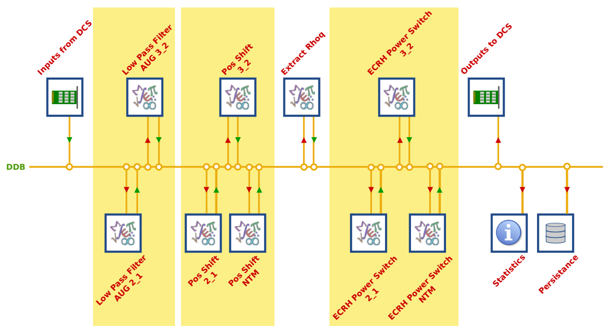


Figure 2: Diagram depicting the interactions of GAMs with the DDB. GAMs are executed sequentially from left to right. Yellow shading indicates where one GAM has been used multiple times with different inputs and outputs.

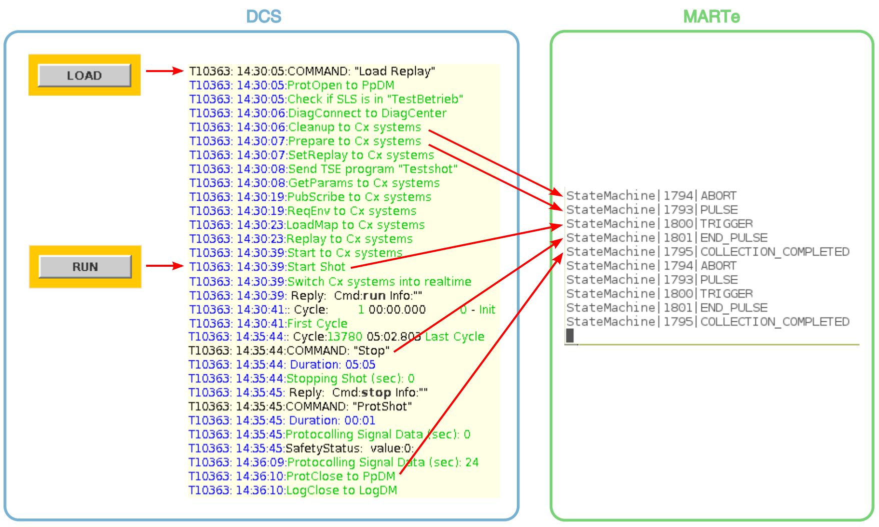
the second performs a shift on **a** detected position ~~**of the 2/1 and 3/2 NTM, as well as a separate shift on whichever one of these has the higher amplitude**~~ (one GAM is used 3 times), the third extracts two ~~**rho**~~ elements ~~**(corresponding to q=1.5 and q=2.0)**~~ from a vector $rho(q)$ and finally power switching is performed depending on the proximity of the Electron Cyclotron Resonance Heating (ECRH) deposition to the shifted positions from the second GAM (again, one GAM is used 3 times). Fig. 2 depicts the GAMs and their sequential execution in MARTe.

In normal operation, DCS uses separate computers for real-time and non-real-time processes. For the demonstration, DCS was adapted such that all software ran on just one computer, with an 8-core 2.4 GHz Intel Xeon $E5-2609$ processor. The RedHawk® operating system was used because it is the operating system of choice for DCS, and MARTe is flexible enough to run on a wide range of operating systems. Of the 8 available cores, 6 were shielded from interrupts and used specifically for real-time processes. Of the 6 shielded cores, 4 were allocated to DCS and 2 to MARTe. The remaining 2 unshielded cores were shared between DCS non-real-time processes and the operating system. MARTe GAMs are assigned a specific core and an execution order by the configuration. DCS APs are scheduled by the operating system on any of the 4 available cores, according to their priorities. For example, data acquisition APs have high priority since several other APs are waiting for their output. Hyper-threading was not used, and all other kernel parameters were left at their default values.

After approximately four months of development, the coupled DCS-MARTe system was presented in a live demonstration. Within this demonstration, it was shown how to start up both frameworks, and operate their user interfaces. Notably, developers with a MARTe background showed that they could operate DCS and vice versa. Then, the DCS state machine was directed to transition through its states via the GUI. MARTe state transitions were synchronised as explained in section 3.2. Fig. 3 depicts buttons at the left triggering state transitions in DCS, which in turn trigger the appropriate state transitions in MARTe. The states for two complete pulses are shown on the MARTe side, since it was important to demonstrate multiple pulses, replaying several different ASDEX Upgrade pulses. In order to start the next pulse from a well-defined initial state, the “cleanup” function in DCS is called before each pulse.

During the real-time phase, MARTe output could be displayed using live introspection tools. Some notable log messages were also highlighted. This demonstrated that it was still possible to use the computer for other tasks, without disrupting the real-time functionality. Performance was robust, despite the hardware limitations and the additional overhead of running two real-time frameworks and in particular the communication between them. As shown in Fig. 4, the cycle time was still well within the real-time limit of 1.5 ms. The diagram shows the cumulative time spent in different tasks each cycle. Firstly, the time for the MARTeAP to acquire signals as a group from the Signal Agent and write them to the ShMem queues. Secondly, the time used by MARTe to read in the signals, compute the results and write the results to the queues. And thirdly the time for the MARTeAP to read the queues and publish the signals back to the Signal Agent. The process requires some additional time for logging, a watchdog timer, and signal management threads. Note that the completion time for each of these tasks does not necessarily correspond to the execution time, since processes must wait for the scheduler and are data-driven. That is, they must wait for inputs from other processes. The figure shows that in most cycles, this wait time (included in the magenta area) dominates. All measurements were made within the DCS MARTeAP **by requesting a timestamp from a dedicated timing board.** So the task “MARTe” refers to the time between writing MARTe inputs to the queue and receiving a semaphore that MARTe outputs are available. Even though MARTe’s performance is not affected by the scheduler, the measurement will be. Most importantly, the process is always completed in a maximum time of 1.4 ms. The excerpt shown in Fig. 4 is the period where this process was under the highest load.

After the pulse, it was demonstrated that the MARTe outputs exactly replicated those from the MHD_Evaluator. A selection of these outputs are com-



```
T10363: 14:30:05:COMMAND: "Load Replay"
T10363: 14:30:05:ProtOpen to PpDM
T10363: 14:30:05:Check if SLS is in "TestBetrieb"
T10363: 14:30:06:DiagConnect to DiagCenter
T10363: 14:30:06:Cleanup to Cx systems
T10363: 14:30:07:Prepare to Cx systems
T10363: 14:30:07:SetReplay to Cx systems
T10363: 14:30:08:Send TSE program "Testshot"
T10363: 14:30:08:GetParams to Cx systems
T10363: 14:30:19:PubScribe to Cx systems
T10363: 14:30:19:ReqEnv to Cx systems
T10363: 14:30:23:LoadMap to Cx systems
T10363: 14:30:23:Replay to Cx systems
T10363: 14:30:39:Start to Cx systems
T10363: 14:30:39:Start Shot
T10363: 14:30:39:Switch Cx systems into realtime
T10363: 14:30:39: Reply:  Cmd:run Info:""
T10363: 14:30:41:: Cycle:        1 00:00.000          0 - Init
T10363: 14:30:41:First Cycle
T10363: 14:35:44:: Cycle:13780 05:02.803 Last Cycle
T10363: 14:35:44:COMMAND: "Stop"
T10363: 14:35:44: Duration: 05:05
T10363: 14:35:44:Stopping Shot (sec): 0
T10363: 14:35:45: Reply:  Cmd:stop Info:""
T10363: 14:35:45:COMMAND: "ProtShot"
T10363: 14:35:45: Duration: 00:01
T10363: 14:35:45:Protocolling Signal Data (sec): 0
T10363: 14:35:45:SafetyStatus:  value:0:
T10363: 14:36:09:Protocolling Signal Data (sec): 24
T10363: 14:36:10:ProtClose to PpDM
T10363: 14:36:10:LogClose to LogDM
```

```
StateMachine|1794|ABORT
StateMachine|1793|PULSE
StateMachine|1800|TRIGGER
StateMachine|1801|END_PULSE
StateMachine|1795|COLLECTION_COMPLETED
StateMachine|1794|ABORT
StateMachine|1793|PULSE
StateMachine|1800|TRIGGER
StateMachine|1801|END_PULSE
StateMachine|1795|COLLECTION_COMPLETED
```

Figure 3: A selection of log messages depicting state transitions. Buttons from the DCS GUI are shown at left, log messages within the same GUI are shown in the centre, and log messages from MARTe are shown at right. The red arrows indicate the cause of the state transition.

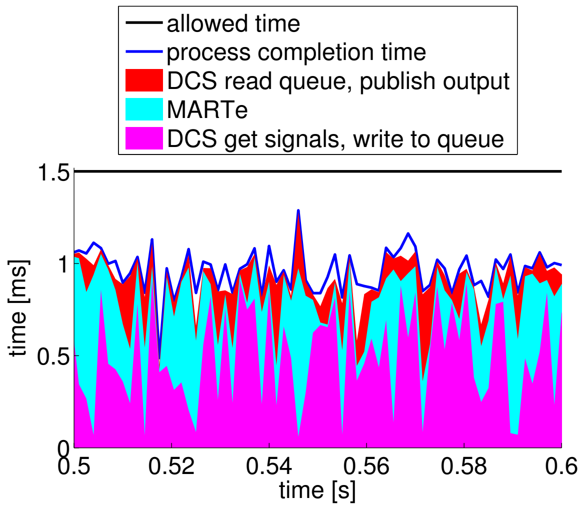


Figure 4: Time allowed for each cycle, and the time required to execute various tasks within the real-time process.

pared in Fig. 5, with those from MHD_Evaluator on the left in blue, and those from MARTe on the right in green. The light blue and light green symbols represent the metadata (confidence and production state), where these deviate from the best case.

The top frames show a measurement of the NTM amplitude, after being passed through a low-pass infinite impulse response filter. Until t=0.26s, the sensor data is tagged as INVALID and STOPPED. This data must be ignored, otherwise it would corrupt the output of the filter for infinite time. The output quality depends on the input, so during this time it is also tagged as INVALID (marked with an X) and STOPPED (marked with a circle).

The second row shows the target for Electron Cyclotron Current Drive (ECCD) in the co-ordinates of the normalised poloidal flux (ρ). The target is calculated by adding an offset trajectory set by a physicist to a measurement of the NTM position. Measuring the NTM position is difficult, so many datapoints are tagged as INVALID (marked with an X). If the position cannot be estimated at all, the value is set to -1, but then the signal must be tagged INVALID since this is not a physical value for ρ. Again, the signal quality of the input (NTM position) is reflected in the output.

The third row indicates when the ECCD is on target so that power can be switched on. This takes the output from the second row as an input, and compares it to an estimate of the ECCD deposition location (ρ_{ECCD}). In this pulse ρ_{ECCD} was constant, so it only aligns with ρ_{target} by chance for two short intervals. The confidence

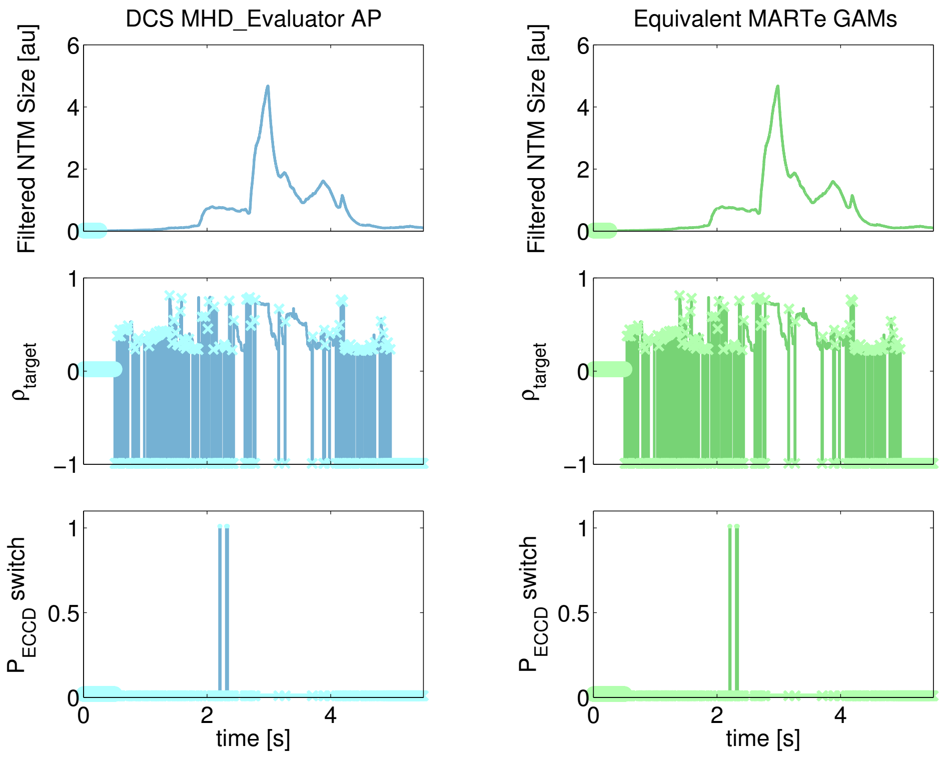


Figure 5: Time traces of some of the output signals from the original DCS MHD_Evaluator AP (left) reproduced by the MARTe GAMs (right).

state is INVALID (marked with an X) whenever the NTM position estimate and hence ρ_{target} are INVALID. Where ρ_{target} is VALID (no symbol), the power switch confidence state depends on the state of ρ_{ECCD}. ρ_{ECCD} depends in turn on a measurement of the density profile, which in this pulse was degraded to CORRECTED. Hence, the confidence state of the power switch is CORRECTED (marked with a $\cdot$) at best. Again, the MARTe GAMS produce identical outputs to the original algorithm in terms of both the numeric output and the related metadata.

5. Conclusions and Future Work

In conclusion, the MARTe and DCS frameworks were successfully adapted to operate side by side, exchanging data and co-operating on a control task. The functionality was demonstrated live in a range of different scenarios for the ITER CODAC team.

For ASDEX Upgrade, the ability to execute MARTe in parallel with DCS adds the possibility to use algorithms for diagnostics and control from other experiments which have already been developed using MARTe. This would be possible with minimal extra development, whenever **an** experiment would require it.

Looking towards the new framework, the successful coupling of DCS and MARTe indicates that the collaboration is off to a good start. As noted previously, coupling two frameworks is not an optimal design, but is extremely useful in terms of stimulating an exchange of ideas and experience. Following this project, the next step is to converge on the specifications for the new framework. Once these are settled, the new code base can be started. In addition to support from a core team of DCS and MARTe developers, there is also interest from the wider community. Good progress up to this point gives confidence that the result will be a flexible and powerful tool for tokamak control in the future.

6. Acknowledgements

This project has received funding from the Euratom research and training programme 2014-2018.

[1] W. Treutterer, et al., ASDEX Upgrade discharge control system - a real-time plasma control framework, Fusion Eng. Des. (2014).

[2] A. C. Neto, et al., MARTe: A Multiplatform Real-Time Framework, IEEE T. Nucl. Sci. 57 (2010) 479–486. 16th IEEE/NPSS Real-Time Conference, Beijing, PEOPLES R CHINA, MAY 10-15, 2009.

[3] J. A. Snipes, Y. Gribov, A. Winter, Physics requirements for the ITER plasma control system, Fusion Eng. Des. 85 (2010) 461–465. 7th IAEA Technical Meeting on Control, Data Acquisition and Remote Participation for Fusion Research, Aix-en-Provence, FRANCE, MAY 15-JUN 19, 2009.

[4] A. Winter, et al., Status of the ITER plasma control system conceptual design, in: Proceedings of ICALEPCS2009, Kobe, Japan, 2009. URL: `http://accelconf.web.cern.ch/accelconf/icalepcs2009/papers/tup110.pdf`.

[5] G. Raupp, et al., Real-time exception handling - use cases and response requirements, Fusion Eng. Des. 87 (2012) 1891 – 1894. Proceedings of the 8th {IAEA} Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research.

[6] W. Treutterer, et al., Management of complex data flows in the ASDEX Upgrade plasma control system, Fusion Eng. Des. 87 (2012) 2039 – 2044. Proceedings of the 8th IAEA Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research.

[7] G. D. Tommasi, F. Piccolo, F. Sartori, A flexible and reusable software for real-time control applications at JET, Fusion Eng. Des. 74 (2005) 515 – 520. Proceedings of the 23rd Symposium of Fusion Technology SOFT 23.

[8] F. Sartori, et al., The JET PCU project: An international plasma control project, Fusion Eng. Des. 83 (2008) 202 – 206. Proceedings of the 6th IAEA Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research.

[9] G. Raupp, et al., Control processes and machine protection on ASDEX Upgrade, Fusion Eng. Des. 82 (2007) 1102–1110. 24th Symposium on Fusion Technology (SOFT-24), Warsaw, POLAND, SEP 11-15, 2006.

[10] D. Valcàrcel, et al., EPICS as a MARTe configuration environment, Nuclear Science, IEEE Transactions on 58 (2011) 1472–1476.

[11] B. B. Carvalho, et al., The ITER fast plant system controller ATCA prototype real-time software architecture, Fusion Eng. Des. 88 (2013) 541–546. Proceedings of the 27th Symposium On Fusion Technology (SOFT-27); Lige, Belgium, September 24-28, 2012.

[12] S. Yun, A. C. Neto, M. Park, S. Lee, K. Park, A shared memory based interface of {MARTe} with {EPICS} for real-time applications, Fusion Eng. Des. 89 (2014) 614 – 617. Proceedings of the 9th {IAEA} Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research.

[13] C. Galperti, et al., Specifications and implementation of the rt mhd control system for the ec launcher of ftu, in: EPJ Web of Conferences, volume 32, 2012. URL: `www.scopus.com`, cited By (since 1996):2.

[14] F. Janky, et al., Determination of the plasma position for its real-time control in the COMPASS tokamak, Fusion Eng. Des. 86 (2011) 1120–1124. Cited By (since 1996):2.

[15] I. S. Carvalho, et al., ISTTOK control system upgrade, Fusion Eng. Des. 88 (2013) 1122–1126.

[16] D. Alves, et al., A new generation of real-time systems in the JET tokamak, in: Real Time Conference (RT), 2012 18th IEEE-NPSS, 2012, pp. 1–9. doi:10.1109/RTC.2012.6418367.

[17] K. Behler, et al., Deployment and future prospects of high performance diagnostics featuring serial I/O (SIO) data acquisition (DAQ) at ASDEX Upgrade, Fusion Eng. Des. 87 (2012) 2145 – 2151. Proceedings of the 8th IAEA Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research.

[18] W. Treutterer, et al., Real-time signal communication between diagnostic and control in ASDEX Upgrade, Fusion Eng. Des. 85 (2010) 466–469. 7th IAEA Technical Meeting on Control, Data Acquisition and Remote Participation for Fusion Research, Aix-en-Provence, FRANCE, MAY 15-JUN 19, 2009.

[19] M. Reich, et al., ECCD-based NTM control at ASDEX Upgrade, in: Europhysics Conference Abstracts, Proceedings of the 39th EPS Plasma Physics Conference, Stockholm (2012) Paper P1.076, 2012. URL: `http://ocs.ciemat.es/epsicpp2012pap/pdf/P1.076.pdf`.