JET Plasma Control System Upgrade using MARTe2

A.V. Stephen[a], A. Goodyear[a], J. Waterhouse[a], E. Jones[a], P.A. McCullen[a], C. Boswell[a], N. Petrella[a], P. Fox[a], M. Lennholm[a], M. Wheatley[a], H. Baker[a], A. Parrott[a], E. Miniauskas[a], K. Purahoo[a], M. Anderton[a], C. Stuart[a], D. Collishaw-Shepman[a], H. Harmer[a], R. Padden[a]

[a] *United Kingdom Atomic Energy Authority, Culham Campus, Abingdon, OX14 3DB, Oxfordshire, United Kingdom*

Abstract

JET real-time plasma control has been delivered with a heterogeneous collection of control systems linked by a dedicated low-jitter, low-latency network. To provide a high degree of flexibility in tuning plasma control algorithms to experimental requirements, the Real-Time Central Controller (RTCC) has been available since 1997. RTCC provides a sandboxed execution environment where experimental algorithms can be deployed with a rapid development workflow. New control laws can be developed by operators during the course of an experimental session. The potential impact of a defect in algorithms evolved without full lifecycle quality assurance can be bounded by clipping feedback control requests at the actuator managers. The likelihood of such defects is reduced in the first place by constraining the algorithms to be composed from reusable blocks and trusted real-time signals. Although this system operated successfully for a long time, limitations in compute capacity of the legacy hardware on which the application was deployed constrained algorithm development.

Motivated by the need to provide physics operators with a more performant system, an upgrade project was carried out to port the RTCC application to a modern high performance PC platform. The architecture selected was to use the MARTe2 framework. Development was able to reuse existing MARTe2 data sources to connect the application to the JET environment using the ITER SDN protocol. RTCC blocks were converted to MARTe2 functions. Python tooling was created to automatically convert previously deployed RTCC algorithms to MARTe2 configuration form.

This paper describes the techniques used to demonstrate system correctness prior to deployment in the JET operating environment. This was particularly important given that it was deployed around the time of the DT campaigns. It explains how the system was used to demonstrate some novel control methods which delivered useful experiments in the final JET campaigns. It also outlines how the JET legacy data combined with this MARTe2 application can offer future value, even in the absence of the JET machine itself.

Keywords: real-time, control, MARTe2, JET, QA

1. Introduction

Plasma Control Systems[1] are large and complex. There are variations in approach, but most fusion projects distinguish between the disciplines of physics design and real-time system implementation.

Physics design is informed by, but usually separate from, first principles high-fidelity simulation. It works with macroscopic plasma quantities and involves reduced order models with sufficient simplification to admit practical use in a real-time environment. This discipline combines physics understanding of the main plasma and machine processes, along with engineering control theory. Development typically uses modelling environments and may make use of experimental or synthetic data.

Real-time system implementation is tasked with converting the physics design into a robust and resilient deterministic control system. It must reproduce the expected behaviour from the design system, in the real-world environment. One challenge is to prove that the algorithm outcomes do not diverge from the predicted path when running on different hardware, implemented in different languages and frameworks. The performance of the code (in all possible paths) must be shown to be compatible with the dynamics time scales. Delay is an enemy of control, time jitter effectively injects noise into a process. The code must be free of race conditions, memory or other resource leaks and defects. If generated by tools, then those tools must be verified. If generated by manual coding, adequate unit and integration tests must be performed.

In both cases, it is necessary to provide tooling and processes which support evolution of the design, and the implementation. This means tracking versions, configuration, input data sets experienced by the codes, and the results and verification of the correctness of these. The lifecycle must also define for how to iterate, when to migrate new features from design to implementation, and how to commission and deploy.

Email address: adam.stephen@ukaea.uk (A.V. Stephen)

[1] PCS - Plasma Control System

Finally, management of off-normal events (often termed *exceptions*) must be addressed. From the design point of view, exceptions include both failure of the design to achieve control of the process dynamics (error of method) as well as failure of equipment which prevents control (error of measurement or actuator). Mitigations defined in the design must be implemented in real-time. In addition the real-time system must be able to degrade gracefully in the event of internal exceptions (hardware faults or system corruption).

One of the design strategies for the JET plasma control system was to include a flexible real-time central controller (RTCC). This was capable of addressing both engineering aspects directly, for a subset of the overall design space. This enabled very rapid prototyping of new control algorithms during the course of JET experimental sessions.

This paper describes why it was necessary to upgrade JET RTCC as part of the final JET operations campaigns. It describes how the JET design principles and system architectures reduced the risk and cost of this project to a level that made if feasible. It describes the outcomes of the project, including the additional benefits that the work leaves for the future exploitation of the JET data.

2. Infrastructure and Architecture

2.1. JET PCS Infrastructure

The JET PCS infrastructure was distributed and heterogeneous[2]. It was divided into around 80 subsystems spanning measurement, control, and protection functions. This separation was required partly for organisational reasons: to divide the responsibility, and the work. It was also beneficial to achieve sufficient decoupling to permit evolution without excessive risk.

The variability in technologies in implementation arose partly from differences in optimal design choices and constraints across the project. It was also enforced as the project lifetime extended and as generations of software and hardware evolved through industry cycles.

Where collections of subsystems were required to work collectively, real-time data was shared as required. As one example, magnetics sensor data was acquired by the KC1D diagnostic, and provided to the plasma shape and position controller. During early phases of the project, these real-time links were dedicated point to point connections. The infrastructure became more fully connected with the introduction of a comprehensive real-time data network. This was initially implemented using Asynchronous Transfer Mode (ATM) telecomms technology (effective, but highly bespoke). A later upgrade extended the real-time network using standard ethernet equipment, and adopting the ITER Synchronous Databus Network (SDN[1]) protocol.

[2]JET has entered decommissioning, and although the PCS systems still exist, they no longer operate.

2.2. JET PCS Architecture

The JET PCS architecture is shown in figure 1. It comprises four main groups: measurements, local feedback, global feedback and actuators. The arrows indicate information flows, including pre-configured parameter values, and real-time data. Some of the core functions operate mostly within the inner local feedback loop. This includes the main magnetic, heating and fuelling loops. More sophisticated control, and protection, are provided through the real-time central controller (RTCC) and auxiliary real-time signal server (RTSS).

Each of the individual systems may be implemented using a variety of technology. The unifying aspects of the architecture are a standard real-time data exchange protocol[2], and a common parameter management system (Level-1[3]).

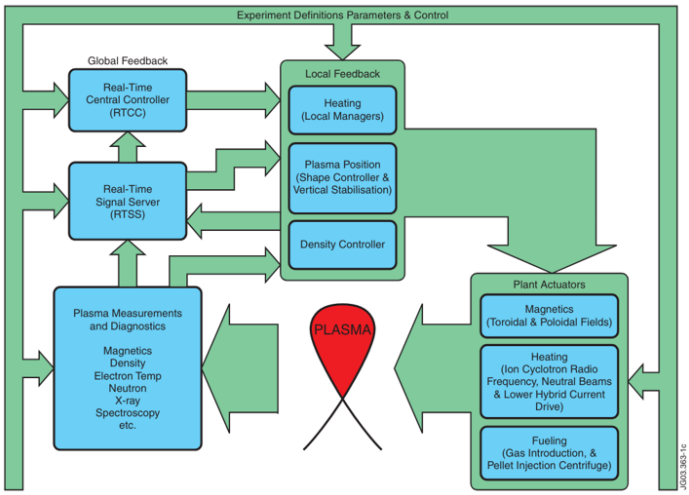

Figure 1: The JET Plasma Control System (PCS) architecture from [2].

2.3. JET Real-Time Central Controller

The function of the standard local control systems was to establish common plasma operating states (scenarios), mostly based on pre-calculated feed-forward references (with feedback control to stabilise around particular operating points). As such, these control systems had a stable development lifecycle, with modifications and enhancements generally being implemented during long shutdown periods.

To provide more advanced control, of plasma parameters for which the relationship between direct measurement and fixed actuator response, it was necessary to add a more flexible and dynamic controller. This real-time central controller (RTCC) provided a library of reusable calculation blocks. Using a custom editor, the physics operator was permitted to design a control algorithm as a sequential graph (or *network*) of up to 200 blocks. Up to four concurrent RTCC networks could be executed.

Figure 2 shows the documentation for one of these calculation blocks (a first order transfer function). Each block received a variable number of analogue inputs (continuously varying signals) as well as one digital input. Blocks to express logical combinations, as well as signal transformation were provided. The RTCC networks could therefore encompass control flow as well as signal processing.

Provision was made for all of the usual mathematical operators, as well as for higher level functions such as waveform generation, and PID control. One limitation was that only scalar valued signals could be treated. Vector oriented control schemes were possible, but had to be expressed as sets of scalar expressions.

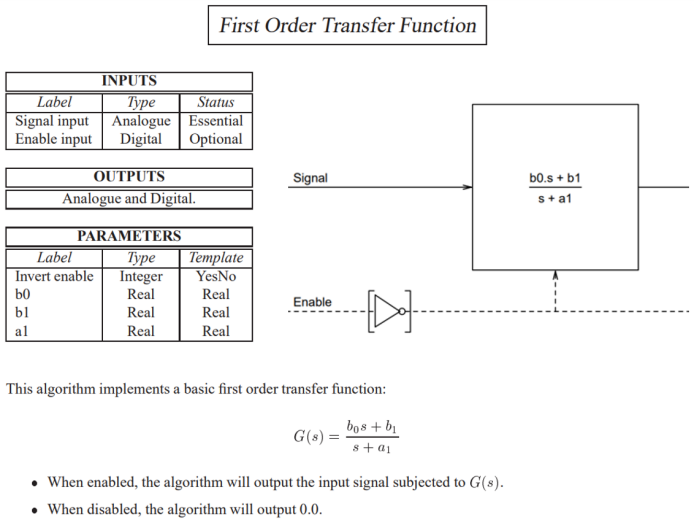

Figure 2: The first order transfer function block. One of approximately 40 reusable components available for physics operators to design control schemes on-the-fly in RTCC.

Using this toolkit, a specialist operator—the Plasma Duty Officer (PDO) would take instruction from the Session Leader (SL) as to the required control schemes needed to support an experimental session. The PDO team built an extensive library of RTCC networks. Typical responsibilities for the PDO could be to modify the networks to probe and tune gains, to replace diagnostic signals by alternatives or to develop entirely new schemes.

The software tools to support the role were sufficient, but required training and had some idiosynchrasies and limitations. Proposals to improve the applications were put forward, but ease of use was not seen as a sufficiently high priority to allocate resources. Figure 3 shows the tabular graph editor. Each row on the left instantiates one of the blocks, and the output is assigned to the identifier on the right. While a supporting viewer which could render the defined network as a graph was available, the graphical view did not support editing.

The outputs from RTCC networks could be routed to the actuator systems to provide references overriding the pre-configured local control. In view of the flexibility, control systems generally offered a restricted window within

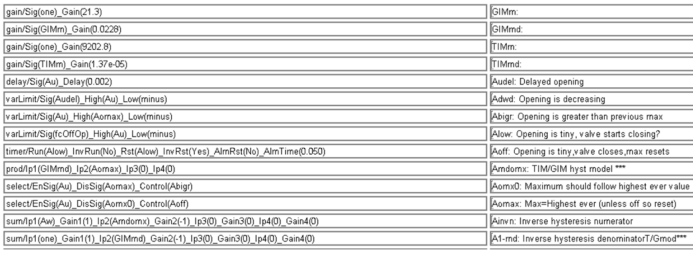

Figure 3: The tabular RTCC network editor, implemented in JET Level-1 software.

which demands would be accepted. Some RTCC networks were also allocated to computing protection signals, and could be used to trigger protective actions if limits were breached.

Following the major ITER-like wall project in 2011, the original RTCC system was duplicated so that one instance be dedicated to running protection networks, while the other was used for experimental functions[4]. This reduced the likelihood that a protection network be misconfigured.

2.3.1. RTCC Limitations

As JET enhancements delivered ever more real-time diagnostic signals, and expanded the capability of the control systems, the demands on the RTCC system increased. Larger networks, consuming more data, required more CPU and memory, or risked skipping control cycles and losing information.

An upgrade to RTCC was required to be able to support the needs of the JET TaskForce planned experiments. Planning the design, implement, testing and commissioning of the upgrade required considerable care given the criticality of the system to operations.

2.3.2. Upgrade Project Requirements and Constraints

The minimum project requirements were to upgrade the system so as to deliver the following.

REQ-1 The required increased capacity in processing and storage.

REQ-2 Full backwards compatibility to reuse any previous control scheme.

REQ-3 Minimised risk to operations

The project team were encouraged to find a design solution which if possible would in addition

REQ-4 Support rapid development of new components.

REQ-5 Improve the physics user experience.

The original implementation assumed a particular hardware platform and operating system. It was also written as a single, large C application. The configuration of the data structures was complex and relied on automatically generated intermediate files through scripts developed with legacy tools.

3. Methodology and Design Approach

One direct route to increased capacity and backwards compatibility (REQ-1 and REQ-2) would have been to port the existing C code to run on a more performant platform. In theory, a good solution. In practice, there were risks in terms of moving the code to a new operating system, or finding new platform support for the existing operating system. While technically achievable, with no operational experience of the alternative platform, it was worth looking for a different solution, to better meet REQ-3. This is not least because operational experience shows that new platforms often require a long period to eliminate edge case defects. This approach would not have offered any potential to meet the stretch targets (REQ-4 and REQ-5).

One of the most successful platforms for developing real-time applications both in JET and other fusion facilies is the Multi-threaded Application for Real-Time Execution (MARTe[5]). This software framework is extremely modular and decouples data management from signal processing. Figure 4 sketches some aspects of the framework. An application consists of one or more real-time threads, each of which marshals data from external sources and performs a series of calculations. Supervisory objects deliver services including data collection, state management, and live introspection.

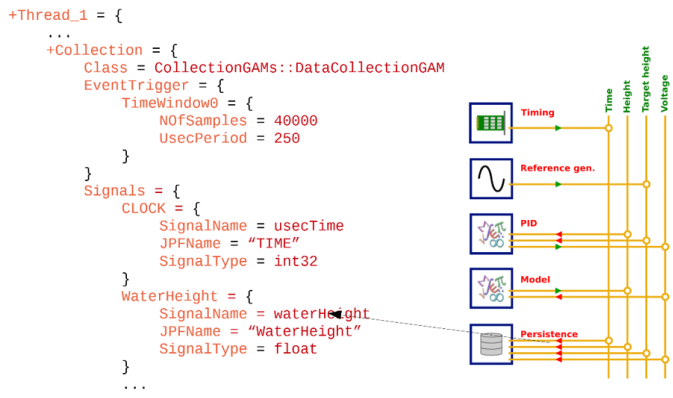

Figure 4: The MARTe C++ real-time framework permits rapid composition of a new application from a deployment file which describes the component instances, attributes and links. Each application is a tree of cooperating objects. The components have been designed to be compatible with robust, reliable, deterministic operation.

Fully proven on JET, for both control and protection applications this would meet REQ-1 and REQ-3. Indeed, it was possible to select the latest release of the framework, MARTe2. This has been developed under the management of F4E to provide integration for a number of ITER systems. It includes some behavioural upgrades which make development more efficient, and has been industrially hardened[6].

A further beneficial feature arose from the JET real-time network update in 2016[7] to add support for the ITER Synchronous Databus Network (SDN). This made MARTe2 directly compatible with the available real-time signals.

Two alternative methods for porting the RTCC network calculation system to MARTe2 were considered. A monolithic approach, to wrap the calculation engine from the heart of RTCC as a single MARTe2 component would have had the least complexity in terms of testing. However, this approach would have provided no benefit in terms of evolving the system. Instead, each of the function blocks provided for RTCC were individually ported as MARTe2 function components. The benefit of this scheme was that to extend the capability would only need additional MARTe2 functions to be added, and this was a simple task.

The implementation therefore consisted of four tasks. To create the MARTe2 application outline. To port and test each of the 40 function blocks. To create a translation script to convert RTCC network descriptions into MARTe2 format. To run integration tests based on historical JET data to verify backwards compatibility.

4. From Validation to Operation

As the software engineering work progressed, each of the deliverables was progressively integrated with operations systems. Wherever possible, full coverage unit and integration tests were defined in the UKAEA gitlab continuous integration (CI) system. This included extensive regression tests of the integrated application against recorded data. This tested not only the correctness of the results, but the real-time performance. To achieve this required some innovation in the use of containers as well as the Yocto environment for building custom linux distributions.

JET machine management rules require strict adherence to procedures for defining and executing commissioning procedures for critical systems[8]. These procedures document the risks that have been identified for each system, the supporting mitigations that have been carried out prior to deployment, and the integrated online end to end checks which must be demonstrated before a system can be used in full JET operation.

Figure 5 lists the types of checks which were applied to each of the software components.

Function	Config Component	Software Components	V&V Method
Communication	Database Definition	Protocol Header Generator SDN Library + MARTe2 Datasource	Manual review / Live Checks
Configuration	Level-1 Database / New Config	Xpsedit + XMARTe Python Application Builder	Regression tests to prove translation Runtime validation to prove new cfg.
RTCC2 Supervisor	Deployment file YML	Python application	Manual checks
Runtime	MARTe2 Config	DataSources + Functions	Unit and Integration Tests in CI/CD

Figure 5: Validation and verification applied to each of the engineering categories related to delivering the upgrade.

4.1. Deployment - Modes of Operation

JET operations had entered a particularly sensitive period at the time when RTCC2 was ready for deployment.

To further minimise the risk, senior management requested that the deployment plan allow for three modes of operation as shown in figure 6.

The first mode was to retain the possibility to operate the full control system with RTCC2 in parasitic mode. RTCC2 would carry out all of the normal operations, would run connected to the real-time network, but *only* as a passive consumer. This eliminated almost all risk and was appropriate for any experiment which only needed RTCC1 capability. This mode was also an opportunity to exercise RTCC2 in the production environment and prove readiness for use.

The second mode would permit RTCC2 to send control requests arising from executing control networks, but only to RTCC1. RTCC1 would then be able to select from the RTCC2 results to directly control the actuator systems. This opened up the extended capacity and capability of RTCC2, but eliminated any possibility of unexpected interactions between RTCC2 and the actuator systems, at the cost of one extra timestep delay.

The third mode would enable full RTCC2 operation, to execute control networks, and to directly control the actuator systems.

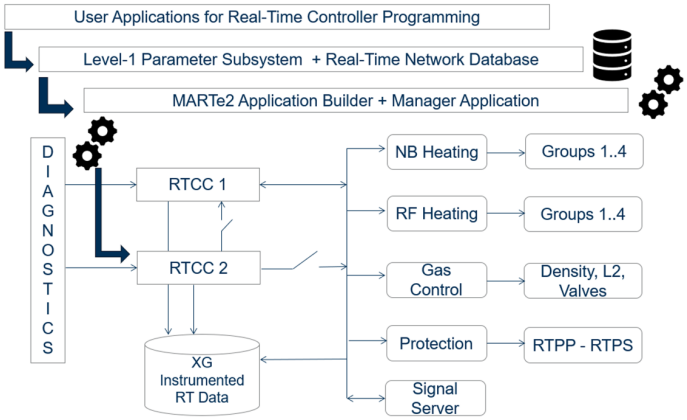

Figure 6: Multiple modes of operation were requested to reduce the risk of updating the plasma control system at a particularly sensitive time for JET operations.

4.2. Outcomes

All of the engineering to develop the new RTCC2 system were completed. In addition to the new MARTe2 components and the tooling for automatic conversion of RTCC control schemes to MARTe2 format, a graphical user interface was created. This provides a more intuitive way to both view, and edit, these control graphs.

The extensive quality assurance evidence (described in § 4) was sufficient to justify a commissioning procedure which had a minimal requirement for machine time to verify system conformity. This is an important point as machine time is very expensive.

The new system entered parasitic testing from JPN 101487. The upgraded functionality was communicated to the task force experiment teams in January 2023. New components were requested to support implementation of an X-point radiation tracker control. The benefits of MARTe2 as an effective environment in which to develop new modules was proven as vector processing and a peak detector components were quickly added. The vector processing significantly reduced the complexity of the networks. These features were used to succcessfully support detachment control experiments.

5. Discussion

It is useful to analyse the architecture of the RTCC2 solution in the context of modern trends in systems design.

Top-down design approaches look to start from formal requirements. High-level designs are then developed using Model-Based Systems Engineering (MBSE). Models are represented in formal modelling languages. Computer aided tools for MBSE support textual and graphical descriptions. Formerly limited to static documents, the current generation of applications can include quantitative, executable simulations. The focus is on modelling the system domain, independent of implementation issues.

JET predated the rise of model based design tools. However, the core parameter management system (Level-1) can be seen as a system which captures the main domain concepts and relationships. It provides the interfaces within which the science and engineering experts encode the target behaviour to be delivered by the control system.

ITER is more directly applying modern methods and the PCS design is developed using formal Systems Engineering. These models document the control models which are developed in Matlab/Simulink so that they can be quantitatively be demonstrated to work correctly.

Turning to the implementation side, software engineers must design a framework suitable to achieve the goals of the models. Design models in UML are typically produced along with the software. In some contexts, this is similar to the systems engineering. More typically though, the lead architects will seek to leverage abstraction to achieve a modular design of components. This methodology may value reusability across projects above ruthless traceability to requirements.

JET software frameworks in general have had a strong tradition of reusability, initially using data-driven patterns. The original RTCC system is a good example of this. As JET became increasingly complex, it was recognised that factoring out the recurring engineering costs common to classes of application was essential. This resulted in the MARTe concept which strongly decouples reusable code from modifiable configuration.

ITER has followed this pattern. It has adopted and enhanced the original JET framework, now MARTe2. It has also commissioned an independent Real-Time Framework (RTF) specifically for the core plasma control system. The provision of dual redundant software systems gives advantages in terms of resilience for the protection system.

Both domain and implementation designers seek to deal with the complexity of large systems through phases of analysis to deconstruct followed by sythensis to compose. Each seeks to balance investment in flexible elements which yield resilience against scope creep which may add cost and risk.

Traditional approaches to bring the domain design and implementation into alignment rely on teams of experts iterating manually. As with other fusion design problems, the limitation of this approach is that early assumptions tend to make the dependencies between the two worlds fragile. The additional cost of adding a late change to the high level model can be severe for the implementation team. Conversely, ensuring that modest changes in the implementation find adequate representation and traceability in the system engineering can also be challenging. Maintaining design integrity across both domains efficiently demands a holistic approach.

One route is to use modelling tools which incorporate code generation capability. This is starting to become possible, and indeed the ITER approach expects to do exactly this. The technique is not without risk, since the challenge is not simply to generate production code that directly reproduces the control models. Practical aspects like operating system interfaces, data acquisition drivers, timing and hardware constraint management must be considered.

A variation on the theme is neatly encapsulated by the example of RTCC/RTCC2 and underpins the design goals of MARTe. This is to use the modelling tool to perform code generation, but with a domain specific target language. For the RTCC application, this target language described the control algorithms in terms of abstract operators (the function blocks), real-time plasma signals as inputs, and parameters values (gains, coefficients). It was this abstraction that enabled a simple and low risk approach to migrating the application to a new software framework. By maintaining the conceptual interface between the physics operators and the run-time engine, it was possible to have high confidence that the replacement system would deliver equivalent functionality. The additional testing to prove the confidence was warranted because of the financial and project risks, but it was important in order to approve the project from the outset.

When debating design approaches, it is recognised that in theory, there is no difference between high-level and low-level languages. As long as they are Turing complete they are functionally equivalent. In practice, the effectiveness of using different types of language for different engineering workflows is important. As Feynman said, "notation is powerful. Invent it.".

6. Conclusion

The RTCC system enabled JET real-time control experiments to adapt rapidly to new information and ideas because the round trip time from concept to implementation did not require external software engineering. This was made possible by offering a flexible, but constrained programmable environment.

The project to upgrade RTCC successfully delivered the minimum project goals of increased capacity with full backwards compatibility to reuse all previous control scheme definitions. This was done without loss of operational time. In addition, the stretch goals of increasing capability and improving usability for end users were addressed.

This change to a critical part of the JET plasma control system, implemented at one of the most sensitive times in the programme life was enabled by sound historical architecture choices. These were both based on abstracting the definition of real-time data flow and processing into machine processable formats. In respect of inter-system communication, this was the real-time data network database, implemented using CODAS Configuration Language (CCL) with automatic code generation. In respect of intra-system algorithm representation, this was via the RTCC network graph, and the equivalent RTCC2 object graph.

The project feasibility rested not solely on these architectural patterns, but on the MARTe framework and the improvements and extensions made to this software in preparation for ITER. The JET real-time network extension to ethernet and use of SDN in 2016 provided a solid foundation. The quality assurance and functional updates delivered by the F4E MARTe2 project reduced the RTCC upgrade risk to an acceptable level.

JET innovations in control system architecture have proved their value in many contexts. The power of inventing effective programming tools to cope with the growing complexity in modern systems is well recognised in other fields. It is hoped that this project will make the case for continued investment in fusion control systems software. To quote the motto of the Software Sustainability Institute, "Better Software, Better Research".

The JET real-time algorithm executor has been ported to a future-proof and future-relevant framework. The implementation leaves the possibility to rerun former JET shots against a virtualised version of RTCC2. This opens the possibility to running virtual JET experiments. These can test alternative state estimation concepts just by replaying the historic data. By replacing dynamic elements with models, a closed loop virtual JET emulator could be demonstrated. This could have a number of valuable uses.

7. Abbreviations

ATM Asynchronous Transmission Mode

DSL Domain Specific Language

JET Joint European Tokamak

MARTe Multi-threaded Application for Real-Time execution

MBSE Model Based System Engineering

PCS Plasma Control System

RTCC Real-Time Central Controller

SDN Synchronous Databus Network

UDP User Datagram Protocol

8. Acknowledgements

The MARTe software has been developed over a long time by members of the fusion community. Much of the software engineering that delivered the RTCC system and many other JET real-time systems was the work of Quentin King.

9. Funding

JET, which was previously a European facility, is now a UK facility collectively used by all European fusion laboratories under the EUROfusion consortium. It is operated by the United Kingdom Atomic Energy Authority, supported by DESNZ and its European partners. This work, which has been carried out within the framework of the Contract for the Operation of the JET Facilities up to 31 October 2021, has been funded by the Euratom Research and Training Programme. Since 31 October 2021, UKAEA has continued to work with the EUROfusion Consortium as an Associated Partner of Max-Plank-Gesellschaft zur Förderung der Wissenschaft e.V represented by Max-Plank-Institut fur Plasmaphysik ("IPP") pursuant to Article 9.1 of the EUROfusion Grant Agreement for Project No 101052200. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

References

[1] L. Boncagni, C. Centioli, F. Iannone, C. Neri, M. Panella, L. Pangione, M. Riva, M. Scappaticci, V. Vitale, L. Zaccarian, Synchronous Databus Network in ITER: Open source real-time network for the next nuclear fusion experiment, Fusion Engineering and Design 83 (2) (2008) 504–510. doi:10.1016/j.fusengdes.2007.10.007.
URL https://www.sciencedirect.com/science/article/pii/S092037960700508X

[2] R. Felton, E. Joffrin, A. Murari, L. Zabeo, F. Sartori, F. Piccolo, J. Farthing, T. Budd, S. Dorling, P. McCullen, J. Harling, S. Dalley, A. Goodyear, A. Stephen, P. Card, M. Bright, R. Lucock, E. Jones, S. Griph, C. Hogben, M. Beldishevski, M. Buckley, J. Davis, I. Young, O. Hemming, M. Wheatley, P. Heesterman, G. Lloyd, M. Walters, R. Bridge, H. Leggate, D. Howell, K. D. Zastrow, C. Giroud, I. Coffey, N. Hawkes, M. Stamp, R. Barnsley, T. Edlington, K. Guenther, C. Gowers, S. Popovichef, A. Huber, C. Ingesson, D. Mazon, D. Moreau, D. Alves, J. Sousa, M. Riva, O. Barana, T. Bolzonella, M. Valisa, P. Innocente, M. Zerbini, K. Bosak, J. Blum, E. Vitale, F. Crisanti, E. de la Luna, J. Sanchez, Real-time measurement and control at JET experiment control, Fusion Engineering and Design 74 (1) (2005) 561–566. doi:10.1016/j.fusengdes.2005.06.286.
URL https://www.sciencedirect.com/science/article/pii/S0920379605003352

[3] H. van der Beken, B. Green, C. Steed, J. Farthing, P. McCullen, J. How, Level 1 software at JET: a global tool for physics operation, in: IEEE Thirteenth Symposium on Fusion Engineering, 1989, pp. 201–204 vol.1. doi:10.1109/FUSION.1989.102207.
URL https://ieeexplore.ieee.org/document/102207

[4] J. S. Edwards, I. S. Carvalho, R. Felton, C. Hogben, D. Karkinsky, P. J. Lomas, P. A. McCullen, F. G. Rimini, A. V. Stephen, Robust configuration of the JET Real-Time Protection Sequencer, Fusion Engineering and Design 146 (2019) 277–280. doi:10.1016/j.fusengdes.2018.12.045.
URL https://www.sciencedirect.com/science/article/pii/S092037961

[5] A. C. Neto, D. Alves, L. Boncagni, P. J. Carvalho, D. F. Valcarcel, A. Barbalace, G. De Tommasi, H. Fernandes, F. Sartori, E. Vitale, R. Vitelli, L. Zabeo, A Survey of Recent MARTe Based Systems, IEEE Transactions on Nuclear Science 58 (4) (2011) 1482–1489, conference Name: IEEE Transactions on Nuclear Science. doi:10.1109/TNS.2011.2120622.
URL https://ieeexplore.ieee.org/abstract/document/5742792

[6] A. C. Neto, F. Sartori, R. Vitelli, L. Capella, G. Ferro, I. Herrero, H. Novella, An agile quality assurance framework for the development of fusion real-time applications, in: 2016 IEEE-NPSS Real Time Conference (RT), IEEE, Padova, Italy, 2016, pp. 1–7. doi:10.1109/RTC.2016.7543176.
URL http://ieeexplore.ieee.org/document/7543176/

[7] J. Waterhouse, A. Stephen, N. Petrella, Introduction of ITER CODAC relevant technologies on JET and MAST, Fusion Engineering and Design 161 (2020) 111858. doi:10.1016/j.fusengdes.2020.111858.

[8] A. S. Kaye, M. Cox, J. W. Farthing, M. Hitchin, P. Lomas, Operation of JET as a user facility, Fusion Engineering and Design 66-68 (2003) 175–179. doi:10.1016/S0920-3796(03)00213-8.
URL https://www.sciencedirect.com/science/article/pii/S092037960