Fusion Engineering and Design 146 (2019) 277–280

Contents lists available at ScienceDirect

Fusion Engineering and Design

journal homepage: www.elsevier.com/locate/fusengdes

Robust configuration of the JET Real-Time Protection Sequencer

James S. Edwards[a,*], I.S. Carvalho[b], R. Felton[a], C. Hogben[a], D. Karkinsky[a], P.J. Lomas[a], P.A. McCullen[a], F.G. Rimini[a], A.V. Stephen[a]

[a] UKAEA-CCFE, Culham Science Centre, Abingdon, Oxon, OX14 3DB, UK
[b] Instituto de Plasmas e Fusão Nuclear, Instituto Superior Técnico, Universidade de Lisboa, P-1049-001, Lisboa, Portugal

ARTICLE INFO

Keywords:
Configuration
Machine protection
Real-time framework
Domain-specific language
Software validation
Operations

ABSTRACT

The JET Real-Time Protection Sequencer (RTPS) co-ordinates responses for magnetic and kinetic actuators to protect the ITER-Like Wall from possible melting events and other undesirable scenarios. It allows programmable stop responses per pulse, based on alarms raised by other systems.

The architecture combines a modular run-time application developed using MARTe (Multithreaded Application Real-Time executor) with the top-level JET supervisory and configuration software, Level-1. Operational experience since 2011 drove a requirement to refactor the system in 2017, moving the maximum degree of functionality from compiled code to configuration data, providing more flexibility, maintainability and verifiability of action(s) to be taken during a pulse.

This paper discusses the features of the architecture that made this clean separation of rule-based logic and real-time signal processing possible and practical, including how functions and interfaces between MARTe and Level-1 are organised. It also explains management of configuration data to address development, testing, commissioning and operations, each with individual ownership, responsibility and lifecycles. The core technology enabling this is the Level-1 domain specific language, able to manipulate, validate and load into plant configuration parameter sets. The language also enables implementation of advanced user interfaces, providing operators with the tools to focus on essentials tasks for their area of responsibility. It exemplifies this with recent verification and validation of the refactored protection system: unit/low-level integration tests defined by core developers and integration/behavioural tests defined by JET's Plasma Operations Group, respectively, ensuring robust and consistent behaviour. We show how the wide scope and power of this language has enabled evolution of JET operations efficiently and correctly over decades of operational experience.

1. Context

The JET Real-Time Protection Sequencer (RTPS) receives signals from diagnostic and real-time modelling systems to provide protection of the JET vessel [1]. These signals can include alarms indicating an exceptional state such as wall overheating or magnetic instability onset. Session leaders can define protection responses to these exceptional states, deviating from the nominal pulse schedule to safely terminate a pulse. RTPS performs real-time processing and filtering of input signals to identify exceptions and implements protection responses by calculating alternative output commands for magnetic and kinetic actuators.

The JET Level-1 software provides an operations setup and configuration tool for machine operators [2]. In the context of RTPS, it provides an interface for session leaders to specify control schemes for actuators and overall response actions for exceptions that may arise during a pulse. Pulses are split into several timed "phases". Responses can be defined that vary by phase and category of alarm for each pulse. Scenarios group types of response appropriate for different classes of exceptions. The operation of RTPS, magnetic and kinetic actuators and other JET PCS systems (such as the Real-Time Central Controller, a programmable physics experiment controller) are inherently coupled by the plasma [3]. Level-1 processes configuration parameters, ensuring plant system consistency and conformance to JET Operating Instructions (JOIs). For RTPS, the resulting configuration is transformed to a MARTe format (Multithreaded Application Real-Time executor) [4].

MARTe is a modular framework for handling I/O and processing logic. It supports dynamic configurations specifying logical blocks, separating the abstractions of what to do and how to do it. No recompilation of the code is required and behavioural variability is expressed in the configuration language as far as possible.

RTPS was first introduced for protection of the ITER-Like Wall (ILW) in 2011. Operational experience since then has driven requirements

* Corresponding author.
E-mail address: james.edwards@ukaea.uk (J.S. Edwards).

https://doi.org/10.1016/j.fusengdes.2018.12.045
Received 9 October 2018; Accepted 16 December 2018
Available online 08 February 2019

change for two key reasons: 1) systems it interfaces with have changed and 2) session leaders have learnt how to use RTPS more successfully, leading to improved and revised specifications. These changes initially resulted in increased complexity of the real-time code. During the operations break in 2017, refactoring the system to migrate functional change back into the configuration language was undertaken. Expected outcomes were improved maintainability and better capability to validate the system without consuming expensive operational machine time.

2. Level-1/MARTe architecture

2.1. Level-1

JET's primary high-level tokamak operations software, Level-1, was introduced in 1997 and has seen more than 500 software upgrades since and the definition of approximately 100k parameters. Configuration-driven concepts in MARTe were initially exploited to provide a generic interface, exposing parameters pertinent to operational goals. This permits responsibility for an application to be divided between domain experts (for functional requirements) and engineering implementation (for non-functional requirements). Parameter metadata supports powerful dependency analysis and permits complex shared management of central information.

This approach has been adopted by several MARTe applications critical to protecting the ILW, including vertical stabilisation, protection response co-ordination (RTPS) and vessel wall temperature monitoring. Level-1 ensures strong version control of configuration data at a range of granularities, permitting re-use even as the system evolves. Parameters from different domains of expertise are cross-checked, ensuring coherent configurations conforming to JOIs and further supporting correct and consistent application semantics.

2.2. MARTe

MARTe applications consume configuration files to build applications from a tree of interconnected blocks called Generic Application Modules (GAMs). GAMs encapsulate a reusable unit of functionality and can be built and tested independently of the application.

In RTPS, MARTe uses Level-1 generated GAM configurations to instantiate objects which perform I/O, compute state machine changes as a function of alarms, and map state changes to action responses resulting in override commands to actuators. The configuration drives object creation and behaviour specification. For example, the RTPS engine moves through a state machine when responses are triggered. Available states are defined by Level-1 so no hard-coded states are required. A significant advantage of this approach is eliminating recompilation. Each pulse has bespoke protection configurations, loaded into the run-time at pulse setup.

3. Level-1 DSL

3.1. Definition

“A Domain Specific Language (DSL) is a programming language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain.” [5].

3.2. Benefits of DSLs

DSLs are limited in their scope compared with general purpose languages (GPLs) and hence feature constrained. These constraints often make them less likely to contain semantic bugs than GPLs, as testing the logic of a DSL is often simpler than generic source code.

Errors arising during execution of a DSL can use vocabulary from the problem domain, making them more intuitive for domain experts over general purpose programming language errors.

Knowledge about the domain is encapsulated in the abstractions of the DSL. Thus, programs can be understood by domain experts, ensuring semantic correctness. Systems built using this approach capture expert knowledge in a reusable and exploitable form.

Level-1 is built with two semantic layers:

1) An external DSL providing primitives that model the generic components of JET control systems. These include managed parameters, JOIs, user interface (UI) widgets, plus standard code to connect these components.
2) An internal DSL, emerging from the implementation of instances of the internal DSL’s primitives used to model specific plasma control entities, such as ShineThroughDensityLimit or TfValidation.

Combined, these two DSLs provide a tool kit to define concepts and entities suitable for solving problems in the domain of tokamak operations. In the case of RTPS, algorithms for coordinated actuator control for vessel protection can be expressed using a vocabulary of the domain. The external DSL is essential, but the greatest value emerges from the internal DSL.

3.3. Internal DSL

This DSL encapsulates concepts such as parameters that exist for a range of pulses and whose names do not change or blocks of code corresponding to a JOI.

A key principle when creating new concepts within the internal DSL is that they must abstract at the appropriate level and be well named. To achieve this, it is critical that definitions are created by a process whereby the Level-1 system engineer elicits knowledge from domain experts. The concepts are agreed by both parties to be readable, pronounceable and comprehensible at a glance. They are chosen to be semantically important to improve understanding, longevity and maintainability of the source code.

3.4. Benefits of the Level-1 DSLs

Used widely throughout the Level-1 source, the internal DSL makes development of interfaces to view/manipulate data simple and quick. Sets of commands can be grouped together much like standard functions/methods of imperative programming languages, providing translation and transformation of underlying data sets and encapsulating operational experience.

Operations tools have requirements and design elements arising from three domains:

- Regulator: defining machine and operational limits.
- Operator: demanding powerful and flexible control room tools, expressive of their goals.
- Control system: requiring configuration from the operator, consistent with achieving the desired execution.

The DSLs support collaboration between experts from each of these domains with maximum communication efficiency: minimising ambiguity, ensuring consistent vocabulary and maximising the separation of concerns.

Data and domain dependencies are mapped using the DSL, including any domain rules that apply to the data. Expanding on section 2.2, complex stop response logic can be built up from parameters entered by session leaders, rather than only populating a predefined set of parameters. The DSL not only specifies parameter values, but also enables dynamic creation of parameter subtrees and their values from within the protection domain. This capability of generating structure as well as populating it is important.

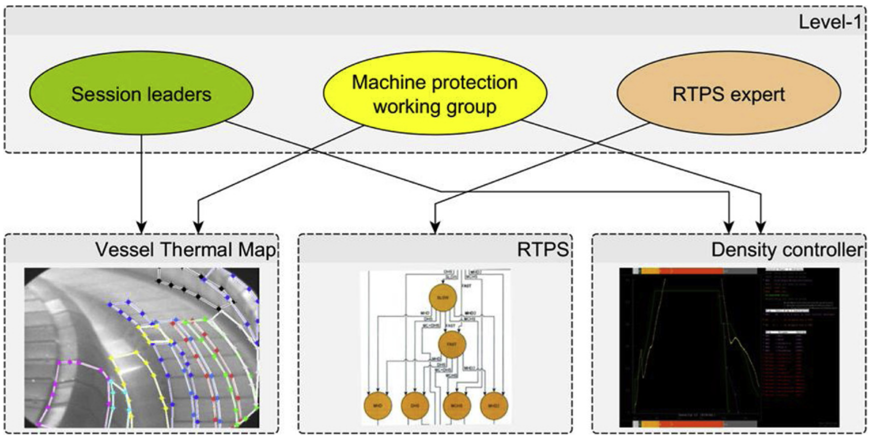

Fig. 1. Separation of concerns by Level-1 when protecting the ILW against hot spots.

In the context of MARTe systems, embedded knowledge made accessible by the internal DSL determines which blocks of configuration can be defined and *how* they can be defined. The glue layer of Level-1 joins many of the rules together with the user interface. It adds knowledge from the domain and so critically, injects operations related insight.

For RTPS, this means functional behaviour can evolve but ensures interface requirements and internal logic remain invariant while the user interface is upgraded. This avoids needing to modify the low-level C++ code in most cases. This is essential as there is high variability per experiment that may include single use or rare parameter usage. This model is analogous to the ITER PCS simulation platform (able to generate code for real-time use), but with the added advantage that no compilation step which may introduce run-time issues is required [6].

3.5. Domain knowledge benefit example

A specific example of the presented architecture benefitting a functional goal and decoupled systems is the implementation of hotspot protection, illustrated in Fig. 1. Configuration is managed on a need-to-know basis. Level-1 comprises an internal model, mapping regions of interest used directly by cameras and how those regions of interest map to tiles around the entire machine. If alarms are raised based on this camera data, protective actions can then be taken. Level-1 programs both RTPS and the vessel thermal map (VTM) such that:

- RTPS needs to know only what it must do to start protective actions.
- The VTM needs to know only temperature thresholds for the regions of interest, which alarms to use and other data in the domain of *detecting* hotspots.

Neither of these systems require extensive knowledge of the other as this has been abstracted to Level-1, which provides them only with data within their specific domain and functionality. The system coupling is reduced to a simple synchronisation of state, yielding a more robust overall system with simple communications protocols.

4. Configuration of RTPS

During the 2017 reengineering process, much of the state machine logic implemented in RTPS GAMs was migrated to Level-1 such that the central tool now:

- encapsulates rules that process control system semantics from parameter lists,
- encodes operating instructions and limits and
- enforces consistency of constraints on parameters across many systems and roles at the point of final plant configuration.

4.1. RTPS domain configuration parameters and values

The RTPS-specific DSL provides an ontology to describe signal mappings, alarms, responses, control actions and a core state machine model. With this vocabulary, Level-1 can specify what should happen when an alarm occurs under given conditions as a function of pulse phase and state history. Importantly, Level-1 maps higher level concepts of control systems and operations down to the detailed domain of real-time protection by means of the MARTe DSL, reducing the semantic gap between the domains and their expert teams.

The blocks of configuration arising from this DSL pertain to interaction with many different control systems, including controllers for density, plasma position and current, neutral beam and radio frequency heating systems and others. These blocks (which can also be considered as layers of a DSL) are described in 4.2, 4.3 and 4.4.

4.2. External configuration

This layer expresses the interoperation of RTPS with standard high-level operations and data acquisition systems, including JET standard communications protocols and Level-1 itself.

4.3. Internal configuration

This layer expresses the interoperation of RTPS with other control systems, mostly comprised of configuration describing real-time data input and output. The main benefit of this layer is the ability to perform drop-in replacements for data source/sink GAMs, meaning model-based testing can be performed by using simulation data GAMs instead of real data sources.

4.4. Application logic

The data-driven layer is primarily generated by Level-1, providing actuator control schemes and responses for segments and states based on session leader choices.

4.5. Overall benefits

Critically, the internal configuration layer has been defined by the RTPS engineers and the plasma operations group (POG) and is considered invariant as it is tested and reused for both online pulsing and offline commissioning.

In addition, the embedded real-time infrastructure configuration is decoupled from the signal processing algorithms. The final data sent to

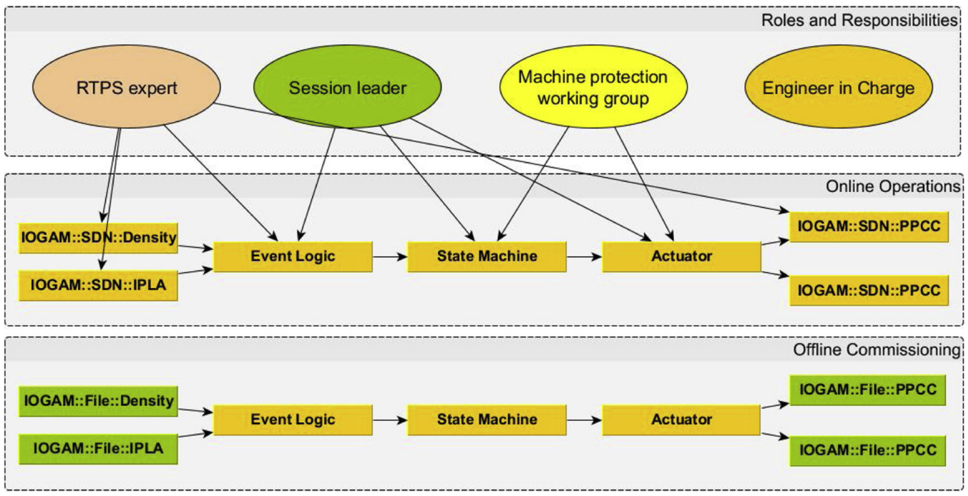

Fig. 2. Domains mapped to RTPS logic, showing online and offline interchangeable GAMs.

RTPS is produced by Level-1 via a generation process that imports several templates from these different layers, patching parameter values and entire blocks of configuration as needed.

5. Commissioning and validation of RTPS

RTPS falls within JET's integrated operations and protection system commissioning process and as such requires robust formal quality assurance prior to operation achieved by a combination of process and tools.

5.1. Validation

Pair programming between the RTPS engineer and POG minimised errors during the initial refactor of key algorithms. The RTPS engineers implemented unit tests for each component of the algorithm and involved POG in validating a subset of these tests in an iterative manner.

Integrated validation was achieved by POG, who defined 71 pulse schedules with Level-1 to use as tests. This is possible due to the interchangeable GAMs mentioned in 4.3, with a template for testing on offline systems automatically used by Level-1.

5.2. Commissioning

Commissioning previously (2011–2013) used five integrated tests carried out on the offline VxWorks (real-time) system. This was expanded to nine tests to complement other reengineering activities (2013–2015). Although this did provide some confidence in the system, it was deemed insufficient in comparison with the full control space available during operations. Hence, a much-expanded set of the validation tests developed by POG were adopted in the 2017 work plan. These 71 cases ensure complete reproducibility between offline and online versions of RTPS.

For both parts of the process, Level-1 was adapted to provide a regression test design feature, built using the same commands a human would need to use (via the interface) to achieve the desired result. This guarantees that the feature is consistent and correct once validated. The validity of the system was thoroughly checked by domain experts, as it no longer requires a full test cycle of a pulse using an offline system, illustrated in Fig. 2.

Using the same tooling for commissioning, offline tests and normal operations lets POG define behavioural tests in the same way as when designing a normal pulse, considerably reducing effort required for validation.

6. Conclusion

Our experiences of the software and operations engineering aspects of separating complexity of protection and configuration systems of JET has shown:

- Level-1 system engineers can produce control room ready solutions at a speed comparable to eliciting the information from domain experts, enabled by DSLs highly targeted at the fusion operations domain.
- The DSLs are designed to be understood at the domain level and hence minimise cognitive friction, increasing confidence of correctness of the overall system behaviours.

Hence small, incremental changes to the tool are lower risk than larger, infrequent changes would be and permits continuous change to Level-1, even during operations.

We conclude that having such a tool kit in place allows very fast transfer of understanding of the problems within the operations domain between domain experts and the system engineer experts. This enables knowledge to be gained quickly within the domain, mitigating the risk of knowledge being lost as system engineer experts change over prolonged periods.

Acknowledgments

This work has been carried out within the framework of the Contract for the Operation of the JET Facilities and has received funding from the European Union's Horizon 2020 research and innovation programme. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

References

[1] A.V. Stephen, et al., Centralized coordinated control to protect the JET ITER-like wall, Contributions to the Proceedings of ICALEPCS 2011, (2012), p. 1423.
[2] P.A. McCullen, J.W. Farthing, The JET level-1 software, Contributions to the Proceedings of the 20th Symposium on Fusion Technology 1998, (1998), pp. 575–578.
[3] R. Felton, et al., Real-time measurement and control at JET experiment control, Fusion Eng. Des. 74 (2005) 561–566.
[4] A.C. Neto, et al., MARTe: a multiplatform real-time framework, IEEE Trans. Nucl. Sci. 57 (2010) 479–486.
[5] A. Deursen, et al., Domain-Specific Languages: An Annotated Bibliography, Technical Report, CWI, Amsterdam, 2000.
[6] M.L. Walker, et al., A simulation environment for ITER PCS development, Fusion Eng. Des. 89 (2014) 518–522.