Fusion Engineering and Design 112 (2016) 942–945

Contents lists available at ScienceDirect

Fusion Engineering and Design

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

Fast development of real-time applications using MDSplus and MARTe frameworks

G. Manduchi[a,*], T.W. Fredian[b], J.A. Stillerman[b], A. Neto[c], F. Sartori[c]

[a] Consorzio RFX (CNR, ENEA, INFN, Università di Padova, Acciaierie Venete SpA), Padova, Italy
[b] Massachusetts Institute of Technology, 175 Albany Street, Cambridge, MA 02139, USA
[c] Fusion for Energy, Barcelona, Spain

HIGHLIGHTS

- The paper describes the integration of two different frameworks for control and data acquisition.
- It describes the way the two frameworks have been integrated.
- It describes the advantages of this combined approach.
- It presents a case study of the utilization of the two integrated frameworks.

ARTICLE INFO

Article history:
Received 20 May 2015
Received in revised form 14 January 2016
Accepted 8 April 2016
Available online 18 April 2016

Keywords:
Real-time control
Data acquisition
Software frameworks
Software integration

ABSTRACT

The recent long lasting fusion experiments introduced a change in paradigm for control and data acquisition. While formerly implemented by different systems, using different software and hardware solutions, new requirements, such as the need of handling a sustained data stream, the availability of powerful general-purpose computers and the evolution of Linux towards real-time responsiveness make an integrated solution nowadays feasible. In the fusion community several frameworks have been developed for control and data acquisition and some of them are shared among different experiments. In particular, MDSplus represents the most used framework for data acquisition and management and MARTe is a framework for real-time applications originally developed at JET, but then adopted in several other experiments. Neither system can be used alone to provide integrated real-time control and data acquisition but, since their functionality complements, they can be used in conjunction. To achieve this, an additional layer has been developed so that data handled in real-time by MARTe components can be redirected to pulse file for storage. At the same time, configuration data, typically defined in the MDSplus experiment model, can be seamlessly transferred to MARTe GAMs during system configuration.

1. Introduction

In former experiments, the short duration of the plasma discharge permitted the use of transient recorders to take signal snapshots during the discharge. The data acquisition system was not actively involved during the discharge itself, and only in a subsequent phase provided the readout of transient recorders memory and the storage of acquired signals in a database. On the other side, the management of data streams was required for real-time control, but the limitations in computer systems performance restricted the applicability of general solutions, leading to specialized solutions for real-time control, different both in hardware and in software from the data acquisition system.

Modern experiments are characterized by an extensive use of real-time control components. In addition, larger devices with long lasting plasma discharge require the management of streaming data acquisition so that information is available during the discharge itself. Even in this case, however, different solutions are required to manage data acquisition and real-time control. Considering Analog to Digital (ADC) devices, larger data buffers are required to improve data throughput in data acquisition, but smaller data buffers are required when ADC devices acquire sensor signals to be used in real-time control in order to reduce latency. Small data buffers are required in data acquisition for real-time control to account for occasional missed deadlines in the cycle time of the control system. The same dichotomy (large buffers, large

* Corresponding author.
E-mail address: gabriele.manduchi@igi.cnr.it (G. Manduchi).

http://dx.doi.org/10.1016/j.fusengdes.2016.04.014
0920-3796/

throughput vs. smaller buffers, smaller latency) holds also considering the hardware and software components of the computer systems used for data acquisition and real-time control, respectively.

While the technologies differ, the user's perspective of data in modern experiments is towards unification between data acquisition and control since the latter is becoming more and more an integrated component of the fusion device. Therefore data should be presented in the same way regardless of their origin and they may represent physical acquired signals as well as values computed by real-time control components.

The co-existence of different solutions sharing the same computer system hardware is made possible nowadays thanks to the rapid evolution of multicore computers that allow partitioning the computer into independent components sharing data in memory. The availability of multicore support with real-time responsiveness in Linux allows reserving a set of available cores for real-time operations without interfering with the rest of the system handling data acquisition and supervision [1].

Different frameworks have been developed in the fusion community to address streamed data acquisition and real-time control, using different techniques to increase data throughput in the former and to reduce latency in the latter and a fully integrated system could be built by integrating two frameworks addressing the two different functions. If we compare it to a new implementation from scratch, this approach has the clear advantage of retaining the experience and solutions gained in years of development and operation and ensures therefore both better availability and reduced time to target. In the presented solution, MDSplus [2,3] and MARTe [4,5], two frameworks used in fusion community for data acquisition and real-time control, respectively, have been integrated to provide a flexible and powerful framework which can be adapted to laboratory experiments running on a single machine as well as to large and distributed systems. The integration of MARTe and MDSplus has been first introduced in [6]. In that paper the advantages of a combined approach have been presented. This paper provides more insight into the way the two systems interact, and presents, as a proof of concept, the first application integrating such frameworks.

2. MARTe and MDSplus

The MARTe framework for real-time applications has been developed at JET and is currently in use also in other machines (RFX, FTU, Compass). MARTe is not the only available framework for real-time control and other solutions are adopted elsewhere, such as DCS in use at ASDEX [7] and foreseen in WEST, and PCS in use at DIID, EAST and KSTAR [8]. The main architectural design choices of MARTe are:

- The abstraction of the underlying platform, which makes the system portable to other non-Linux systems.
- The component abstraction which clearly separates the management of data flow from control computation.

MARTe is implemented in C++ and its modular organization is reflected in the underlying C++ class hierarchy. The integration of new components is carried out by implementing classes that extend a set of given interfaces (abstract classes). These interfaces represent the only required interaction between user components and the rest of the framework, hiding in this way system internals and allowing a rapid development of new components.

MDSplus is a framework for data acquisition and management widely used in the fusion community. It provides a common data interface to all the data types dealt with in a scientific experiment. The data abstraction is pushed further by the concept of expression. All data, ranging from simple scalars to whole python programs are represented in MDSplus expressions that are evaluated on the fly when required. Evaluating an expression means returning its value for scalars and arrays, or carrying out the computation specified in the expression itself. It is possible in this way to express in a natural way the data dependencies which characterize the possible complex structure of scientific experiments. MDSplus provides also support for high speed streaming data acquisition in distributed, multi reader and multi writer environments, and includes several support tools and Application Programming Interfaces (API) to data in several languages.

The functionality of MDSplus and MARTe has very little overlap, but their union covers most requirements for integrated real-time control and data acquisition. As a consequence, the two frameworks can be integrated in an effective way since:

- The lack of overlapped functionalities helps in the definition of an unambiguous definition of what the integration components have to merge.
- It is not necessary to implement new major components from scratch, and the integration mostly consists in providing wrappers to make the components from the different frameworks compatible each other.

The integration addresses two main features: configuration and data storage. Configuration in MARTe consists in the definition of a possibly large set of parameters required both by the core and the user components. In its default configuration, MARTe parameters are read during system startup from a configuration file, and they can be changed on the fly during offline (i.e. non real-time) operation. The MDSplus experiment model represents the most natural place for storing MARTe configuration data, in addition to the other experiment parameters. Data storage is required during real-time execution in order to save the streams of data handled by MARTe, both coming from sensors and produced by online computation.

A peculiar aspect of both MARTe and MDSplus is their highly modular approach that allows the installation of only those components that are required for the specific requirements. This eased their integration, mainly consisting in assembling components from both systems with a minimal development effort for interface adapter components.

3. MARTe-MDSplus integration

MDSplus pulse files are hierarchically organized and therefore sets of related data items can be grouped into subtrees in the main experiment tree. This represents the usual way data are organized in MDSplus. Pushing this concept further, groups of data items describing specific hardware devices can be grouped into subtrees. Different instances in the experiment of a given piece of hardware such as an ADC device are reflected in the experiment model and pulse file (the pulse is an experiment model filled with data acquired in a given pulse discharge) by different subtrees with the same structure. This approach has a strong similarity with Object Oriented programming, where device types represent the classes (i.e. describe the internal data structure) which can be instantiated during program execution. In general, MDSplus devices can be used whenever related data may appear in different instances, and this is the case for the MARTe components. For example, a MARTe module for PID control may be defined in several instances into the MARTe configuration, and each instance may require individual configuration parameters. The natural choice is therefore to define MDSplus devices to contain the configuration of the corresponding components in MARTe, originally defined in MARTe in a configuration file.

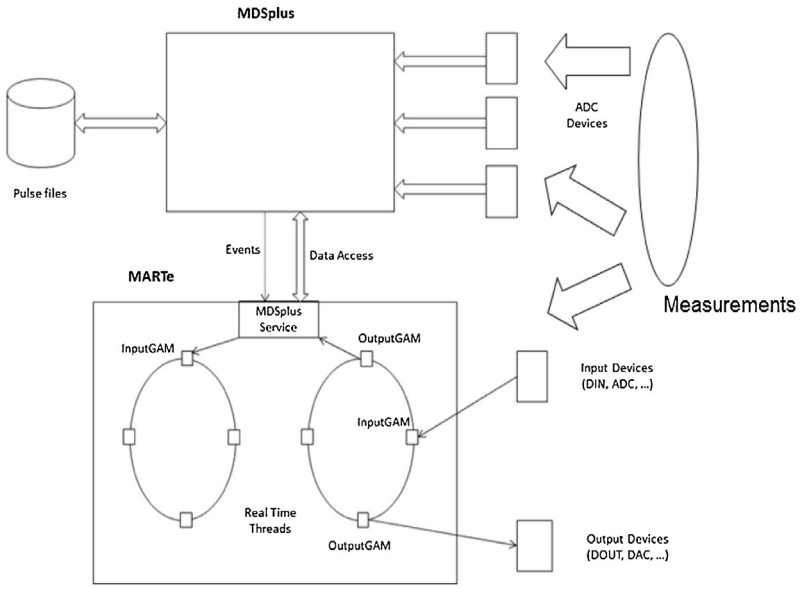

Fig. 1. Block diagram of MARTe-MDSplus integration.

MDSplus allows specific graphical interfaces be developed and associated with devices types using Java Beans aware IDEs. As a consequence, the specific graphical interfaces for MARTe configuration data can be easily developed to provide a user friendly interface to MARTe configuration.

The MARTe framework provides three main classes of components:

- Generic Application Modules (GAMs) which host the computation components involved in the control system. GAM instances are connected in the MARTe configuration to form a set of Real Time Threads each carrying out cyclical computation for a set of GAMs.
- Generic Acquisition Modules (GACQMs) which provide generic driver support. GACQMs are associated with two kinds of GAMs provided in MARTe core: InputGAM() and OutputGAM() that represents the origin and the sink on the real-time data flow. The origin of a data flow typically is represented by some kind of sensor, that is, by some hardware device which needs to be read during execution. The adaptation of the generic concept of data origin with the specific hardware and software driver is carried out by the GACQM instance associated with the corresponding InputGAM. The same concept applies to the data sinks which typically correspond to the generation of physical outputs via some kind of hardware device.
- Services, describing generic support activity not necessarily related to data flow in real-time operation. An example is the Web service which exports the internal MARTe components configuration to Web browsers.

The full integration of MDSplus in MARTe required the development of only two new components: a GACQM and a Service. When MARTe starts, the MDSplus Service activates two threads communicating with MDSplus via mdsip, the TCP/IP based protocol of MDSplus for distributed data access, and via MDSplus asynchronous events. The first thread listens to requests for configuration updates. Whenever a new configuration has to be uploaded, typically at the beginning of a new discharge sequence, an event is generated by MDSplus and recognized by the service thread which will read the configuration parameters. The received configuration is uploaded on the fly into MARTe and overrides the previous one. It is worth noting that MARTe components (GAMs and GACQMs) are not aware of the fact that their configuration is received by MDSplus, as they access parameters via the internal MARTe API for parameter retrieval. This means that the parameters of every MARTe component can be stored in MDSplus, and no MDSplus specific issues need to be considered when developing new GAMs. The second thread acts as a dispatcher for data streaming towards MDSplus pulse files. Such data streams originate from OutputGAM instances which are associated with a MDSplus GACQM instance, and are queued to this thread, typically running in a core which is separated from the cores used by real-time control, in order not to affect real-time performance.

The MDSplus GACQM provides the driver support to InputGAM and OutputGAM instances used as data origins and sinks, respectively. When a MDSplus GACQM instance is associated with an InputGAM, configuration data stored in MDSplus pulse files is used and made available to MARTe GAMs. Such data are typically represented by reference waveforms prepared before the plasma discharge. As it is not possible to provide data readout in real-time, reference waveforms are read and stored in memory by the MDSplus Service during configuration upload, so that interpolated values (depending on the current time) can be provided in real-time by the MDSplus GACQM. Data sinks towards MDSplus pulse files are managed by enqueueing output data to the data storage thread which will then efficiently buffer and store data in segments. In this way it is possible to store not only data representing physical quantities (i.e. sensor readout) but also data internally computed. This feature proved extremely useful for debugging real-time applications by analyzing the intermediate results of the real-time computation performed during plasma discharges. Observe that acquired data are not stored in memory but streamed during control. It is therefore possible to inspect (e.g. to display) the status and the performance of the control system during long lasting discharges.

Fig. 1 illustrates the interaction between MARTe and MSplus. MDSplus is responsible of data acquisition from devices not involved in real-time control. The MARTe MDSplus Service orchestrates communication with MDSplus and the Input and Output GAMs reading waveform references from or streaming signals to MDSplus. Duplication in functionality is avoided: the ADC devices involved in real-time control are not accessed directly by MDSplus

for data acquisition, nor MARTe components directly write data to disk. In both cases, the specific MDSplus-aware GACQM will supervise the data flow between the real-time environment and the data storage.

4. A case study: the control and data acquisition system of the SPIDER experiment in the ITER NBI test facility

The ITER NBI test facility is currently in construction at Consorzio RFX, Padova (ITALY). SPIDER represents the first experiment in the facility for the study of the negative ion source to be then used in the Full Size Neutral Beam construction [9]. During the experiment, a negative ion beam is generated and accelerated. Since the beam generation can last to up to an hour, continuous data streams will be handled by the data acquisition system implemented in MDSplus. Real-time operation is also requested in order to protect the system against grid discharges. Discharges represent unavoidable events in accelerators which require in any case a prompt response from the system in order to avoid damaging power supplies. When a discharge is detected, the signals generated by the control system and providing the reference of the power supplies must be turned off for a given amount of time and then reset to their previous level via a predefined evolution in time. This fast control is carried out by MARTe that reads the reference waveforms from the MDSplus pulse file and acquires the discharge detection signal via a digital input (DIN) device. The computed reference waveforms are sent in real-time to the power supplies via Digital to Analog (DAC) converters. The computed waveforms are also acquired by MDSplus via the MARTe-MDSplus GACQM and Service. Data acquisition and interface for SPIDER is handled by MDSplus. The configuration of the experiment is defined in the MDSplus experiment model, including the configuration of the used MARTe components. During the beam generation data not handled by MARTe are acquired via National Instrument ADC devices and streamed to MDSplus which stores them in the pulse file in segments and made available as soon as the corresponding segment has been stored. From the user's perspective there is no difference between signals acquired by MDSplus via the ADCs and those computed in MARTe and acquired by MDSplus via the MARTe-MDSPlus GACQM and Service.

The following facts have been confirmed by our experience:

- No changes have been required in the existing code for any used MARTe and MDSplus component.
- The interface components described in section 3 represent the only ones required for integrating the two systems in a variety of applications.
- With the above components, integration is just a matter of configuration.
- The thread/core segregation provided in MARTe configuration allowed optimizing computer resource assignment making it possible to let a real-time system co-exist with components originally not intended for real-time applications.

The only specific MARTe component developed for SPIDER has been the GAM for the check for discharge and the consequent actions in the generation of the power supply reference waveform. Even if the configuration of this GAM is stored in the MDSplus experiment model and its reference signals will be stored in MDSplus pulse files, no MDSplus specific issues have been included in its integration, exclusively based on the MARTe interfaces for GAM development. This means that this GAM can be used in other MARTe systems regardless of the fact they use also MDSplus and, at the same time, that other components originally developed in MARTe systems not using MDSplus can be interfaced to the latter as well.

5. Conclusions

The paper has presented a new approach for real-time control and data acquisition based on the integration of two frameworks used in the fusion community. MDSplus and MARTe turned out to be very well suited for integration because the overlap in supported functionality is minimal but at the same time the union of their capabilities covers all the requirements for control and data acquisition in long lasting experiments, which can be summarized as follows:

- User interface: for system configuration and visualization of acquired data (MDSplus).
- Pulse scheduling: to carry out the states in a pulse discharge sequence (MDSplus in step with MARTe).
- Data distribution: to let configuration data be available in any system participating to control and data acquisition (MDSplus + MARTe-MDSplus integration components).
- Real-time computation: to carry out active experiment control and other real-time activities (MARTe).
- Data acquisition streaming: to carry out the continuous acquisition of data generated by devices not involved in real-time control (MDSplus).
- Real-time data streaming: to carry out the continuous acquisition of data produced by real-time control components (MDSplus + MARTe-MDSplus integration components).

The presented case study confirmed that the combined approach allows a rapid development of control and data acquisition systems relying on well-established solutions thanks to the experience gained in both frameworks after years of operation in many scientific experiments. A quality improvement process is currently (and independently) in course for both MDSplus [10] and MARTe, driven by the need of relying on well-tested components to reinforce reliability in critical control applications.

Disclaimer

This publication reflects the views only of the author, and Fusion for Energy cannot be held responsible for any use which may be made of the information contained therein.

References

[1] G. Manduchi, A. Luchetta, A. Luchetta, C. Taliercio, From distributed to multicore architecture in the RFX-mod real time control system, Fusion Eng. Des. 89 (2014) 224–232.
[2] G. Manduchi, T. Fredian, J. Stillerman, Future directions of MDSplus, Fusion Eng. Des. 89 (2014) 775–779.
[3] MDSplus Web site at http://www.MDSplus.org.
[4] A.C. Neto, et al., MARTe: a multi-platform real-time framework, Trans. Nucl. Sci. 57 (2010) 479–486.
[5] MARTe Web site at http://efda-marte.ipfn.ist.utl.pt/marte/.
[6] G. Manduchi, A. Luchetta, C. Taliercio, A. Neto, F. Sartori, G. De Tommasi, Integration of Simulink, MARTe and MDSplus for rapid development of real-time applications, Fusion Eng. Des. 96–97 (2015) 645–648.
[7] W. Treutterer, et al., ASDEX upgrade discharge control system—a real-time plasma control framework, Fusion Eng. Des. 89 (2014) 146–154.
[8] A. Hyatt, et al., Designing, constructing, and using plasma control system algorithms on DIII-D, IEEE Trans. Plasma Sci. 42 (2014) 421–426.
[9] A. Luchetta, G. Manduchi, C. Taliercio, Fast control and data acquisition in the neutral beam test facility, Fusion Eng. Des. 89 (2014) 663–668.
[10] T. Fredian, J. Stillerman, G. Manduchi, A. Rigoni, K. Erickson, MDSplus quality improvement project, in: 10th IAEA Technical Meeting on Control, Data Acquisition and Remote Participation for Fusion Research, 20–24 April 2015, Ahmedabad, Gujarat, India, 2016.