Fusion Engineering and Design xxx (2015) xxx–xxx

Contents lists available at ScienceDirect

Fusion Engineering and Design

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

Integration of Simulink, MARTe and MDSplus for rapid development of real-time applications

G. Manduchi[a,*], A. Luchetta[a], C. Taliercio[a], A. Neto[b], F. Sartori[b], G. De Tommasi[b,c]

[a] Consorzio RFX (CNR, ENEA, INFN, Università di Padova, Acciaierie Venete SpA), Padova, Italy
[b] Fusion for Energy, Barcelona, Spain
[c] Consorzio CREATE/DIETI, Università degli Studi di Napoli Federico II, Via Claudio 21, 80125 Napoli, Italy

HIGHLIGHTS

- The integration of two frameworks for real-time control and data acquisition is described.
- The integration may significantly fasten the development of system components.
- The system includes also a code generator for the integration of code written in Simulink.
- A real-time control systemcan be implemented without the need of writing any line of code.

ARTICLE INFO

Article history:
Received 26 September 2014
Received in revised form 11 February 2015
Accepted 26 March 2015
Available online xxx

Keywords:
Real-time system
Plasma control
Software engineering
Software frameworks

ABSTRACT

Simulink is a graphical data flow programming tool for modeling and simulating dynamic systems. A component of Simulink, called Simulink Coder, generates C code from Simulink diagrams. MARTe is a framework for the implementation of real-time systems, currently in use in several fusion experiments. MDSplus is a framework widely used in the fusion community for the management of data. The three systems provide a solution to different facets of the same process, that is, real-time plasma control development. Simulink diagrams will describe the algorithms used in control, which will be implemented as MARTe GAMs and which will use parameters read from and produce results written to MDSplus pulse files. The three systems have been integrated in order to provide a tool suitable to speed up the development of real-time control applications. In particular, it will be shown how from a Simulink diagram describing a given algorithm to be used in a control system, it is possible to generate in an automated way the corresponding MARTe and MDSplus components that can be assembled to implement the target system.

1. Introduction

Real-time control is routinely applied in current fusion devices in order to control several plasma parameters such as shape, position and magneto hydrodynamic (MHD) perturbations. Modern computer technology allows the control system to react within milliseconds or even less, a timescale which is enough to control most plasma parameters. In addition to computer technology, the evolution of Operating Systems (OS), in particular Linux, toward a more efficient usage of the computer resources makes the use of general-purpose OS viable for real-time control without the need of specialized, and often expensive, solutions.

Even if it is in principle possible to develop from scratch a real-time control system targeted at a given application, this approach would represent a poor software engineering choice. It is in fact not easy to develop efficient code with real-time performance, and therefore re-using tested libraries and components becomes very important when dealing with real-world systems. For this reason, several frameworks for real-time applications in fusion devices have been developed and are currently in use, such as PCS [1], DCS [2] and MARTe [3]. Typically, a framework for real-time control provides the required management of data flow from sensors to control components, and from control components to actuators. The normal approach is a modular one; control algorithms are embedded in software components which are then “cabled” in the overall data flow. Its main advantage is that the control engineer can concentrate on control itself and neglect all the other aspects of the systems, such as the interaction with device drivers and

* Corresponding author. Tel.: +39 049 8295039.
E-mail address: gabriele.manduchi@igi.cnr.it (G. Manduchi).

http://dx.doi.org/10.1016/j.fusengdes.2015.03.050
0920-3796/

the protocols used to exchange data in a distributed environment. Practice is however not so bright and adapting a framework to the specific system will require an accurate configuration and possibly the development of new components to support the specific hardware devices and to interface the real-time application in the main control and data acquisition system.

The general configuration of a framework for real-time control is shown in Fig. 1. At the lowest level is the hosting OS, usually Linux. An abstraction of the OS is normally provided in order to simplify the interface with system resources and to allow the porting of the framework to other platforms. Data flow management components supervise the data transfer in real-time among system components. Configuration management and Control Algorithm integration components let the framework be adapted to the specific real-time control application. Monitoring and simulation tools provide the required components for the tuning and the validation of the system, respectively.

When a framework has been properly configured and the control components integrated in it, the work is normally not finished. It is in fact necessary to integrate the real-time control system into the general one for supervision and data acquisition in order to provide support for the synchronization of the control system activity in the overall discharge sequence, the transfer of the real-time control configuration from the central system and the readout and storage of the signals produced in operation. These signals may refer to physical quantities such as signals read by sensors or references sent to the actuators, but may also refer to internal values produced by the control algorithms which may prove necessary for the comprehension of the control behavior.

Typically in fusion research the development of control algorithms and of the real-time systems is not carried out by the same developers. Information engineers are normally in charge of the development and the maintenance of computers and system software, including real-time control framework, while control engineers or physicists with a different background are in charge of developing the optimal control strategies. Simulink is the preferred tool for the development of control algorithms because it offers a very powerful environment for the rapid development and simulation. Once a control algorithm has been designed, its integration in the real-time system requires its translation into a program, normally written in C or C++, and its integration in a system component. This process is error prone and may lead to a significant effort with consequent delay in the component release. As a consequence, the possibility of automatically generating the control components from Simulink blocks may lead to a drastic reduction, if not to the complete cancelation, of translation errors and therefore to a much reduced time-to-target. It is worth noting that Simulink supports the generation of C code carrying out block functionality, but in any case such code must be adapted to the specific framework interfaces.

This paper proposes a portable solution addressing the above facts using MARTe and MDSplus [4], two widespread frameworks for real-time control and data acquisition, respectively, and including an automated procedure for the translation of Simulink control blocks into MARTe components. After a brief description of the MARTe framework, representing the "driving wheel" of the integrated system, the integration of MDSplus functionality and the generation of MARTe components from Simulink are discussed.

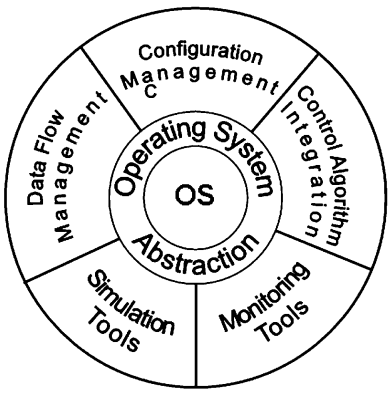

Fig. 1. The components of a framework for real-time applications.

2. The MARTe framework

MARTe has been originally developed at JET and is currently available under open source license [5] and in use in different laboratories such as JET, RFX, FTU. MARTe is written in C++ and provides the integration of a network of components which are represented by Generic Application Modules (GAMs) whose interface is known by the system and whose configuration and connections are specified by a configuration file parsed at system startup. GAMs represent the abstraction required for the integration of the control components, but other abstraction models are defined in MARTe as well. Generic Acquisition Modules (GACQM) represent the abstraction of device drivers and GACQM declared in the configuration file correspond to the instances of the hardware devices used for real-time control. GACQM are then associated with generic GAMs for input (InputGAM) and output (OutputGAM) which produce and consume, respectively, the data handled by the system. MARTe defines also the service abstraction to describe those activities which are not strictly related to real-time execution. For example, a MARTe service implements a Web Server so that every MARTe component can export its current configuration as a HTML page. More in general, MARTe services carry out operations in background, i.e., without compromising real-time performance, to complement the activity of the GAMs.

The configuration of the whole MARTe system is fully specified by a configuration structure in which all the GAMs, the GACQM and the services are declared, as well as the Real Time Threads (RTT) defined for the system and specifying how the system will behave in real-time. Every RTT represents an execution unit and is typically assigned to a given core in the system. A given RTT executes in sequence a set of GAMs, and the way data are exchanged among the GAMs is specified in the configuration file as well. A set of parameters is associated with every component in the configuration, and the specified values are assigned to components at system startup. An important fact about configurations in MARTe is that the configuration information, read at system startup from a configuration file, can be re-generated in runtime, recreating on the fly the whole set of components and associated parameters. In this way it is possible to handle dynamic reconfiguration of MARTe without restarting it, by letting a service regenerate the configuration when required, e.g., when the service receives the request for a new configuration from the supervision system. The main MARTe components are summarized in Fig. 2. The BaseLib library provides OS abstraction, and all the other MARTe components are layered on it. The GAM and GACQM layers provide the support for the development of user-defined Generic Applications Modules and I/O drivers, respectively. In addition to the user-defined ones, the input and output GAMs are provided in MARTe, which implement data sources and sinks, respectively, based on the associated driver. The service layer allows the integration of user-provided services, in addition to the Web service, already provided in MARTe. Real Time Threads host GAMs execution in real-time and are defined, in addition to the other components in the configuration file.

Please cite this article in press as: G. Manduchi, et al., Integration of Simulink, MARTe and MDSplus for rapid development of real-time applications, Fusion Eng. Des. (2015), http://dx.doi.org/10.1016/j.fusengdes.2015.03.050

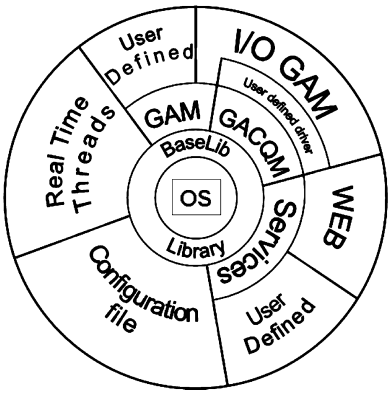

Fig. 2. The MARTe components.

3. MDSplus and its integration in MARTe

MDSplus is a widely used package for data acquisition and management in fusion experiments. It provides support for the management of a wide set of data types which represent the whole set of data classes handled during a discharge sequence. MDSplus pulse files contain both configuration data and acquired signals to represent a complete description of the discharge. In particular, configuration data are defined before the discharge sequence, using a variety of interface tools, and are contained in the experiment model, which represents the pulse file template. The pulse file created from this template is then filled during and after the discharge. Data relevant for real-time control can be categorized as follows:

- Configuration data, required for the proper configuration of the MARTe components, possibly changed from shot to shot.
- Waveform references describing waveforms to be generated in real-time. Examples of waveform references are the required output for open loop components, or the reference for a feedback control.
- Signals acquired from real-time components corresponding to physical signals acquired by sensors, actuator references and signals internally produced by control components.

Data related to real-time control is embedded in MDSplus into device instances. A MDSplus device represents the container of a set of related data. The class of the device describes how data are organized. For example, a given ADC hardware device will be characterized by a data set (including the ADC configuration and the acquired signals) described by the corresponding class. A number of hardware ADC devices of a given type will correspond to the same number of instances of that ADC device class.

In particular, a MDSplus device class has been defined to describe MARTe related data and an instance of this class will be defined for every MARTe component interacting with MDSplus. This class defines room for a generic set of configuration parameters, each identified by a pair of name and associated value, a set of reference waveforms and a set of acquired signals. Reference waveform will be stored in MARTe before entering real-time operation, and acquired signals will be written during real-time operation. The uploading of the configuration data and of the reference waveforms is handled by a specific service in MARTe for the interaction of MDSplus. This service synchronizes with the main discharge sequence via MDSplus events. A MDSplus event represents an asynchronous event which may bring data. Using the publish–subscribe pattern, the service registers for a class of MDSplus events which will bring information about the current MARTe configuration. These MDSplus event will be generated by the code associated with the MARTe-specific MDSplus devices executed during the discharge preparation sequence. When such an event is received, the associated data will specify its configuration, expressed as a set of (name, value) pairs, to the target component and the service will regenerate the MARTe configuration on the fly so that the new configuration is made available to the target component. In addition, the service will store in memory reference waveforms data to be used in real-time operation. The management of waveform generation in real-time as well as the storage of the generated signals is carried out by a specific GACQM. Generation of reference waveforms can be seen as an input from a source of data, as from an ADC device, and storage of produced signals can be seen as a data sink, not different from a DAC device receiving reference waveforms. For this reason reference waveform generation and signal storage will be handled in MARTe by InputGAM and OutputGAM components, respectively, associated with the MDSplus specific GACQM. While reference waveforms are stored in memory before entering real-time operation, it is normally necessary to handle a data stream for the storage of signals in long lasting discharges. Storing output signals in memory and reading it afterwards would in fact prevent the possibility of knowing what is happening to the system during the discharge itself, unacceptable for long lasting discharges. In order to avoid interference with real-time operations, it is necessary that the online storage of produced signals is managed by a separate thread running on a core that is not involved in real-time operation. This is handled by the MDSplus MARTe service which receives data packets from the running MDSplus GACQM, queues them and let a separate thread of the MDSplus MARTe service write (possibly remotely) the data packets into the pulse file. The MDSplus GACQM is involved whenever data are written in real-time into OutputGAM for storage and acts as a bridge between the real-time thread and the thread writing data in pulse files.

The modular organization of MARTe combined with the data access management of MDSplus lets the overall system be simulated to test both its correctness and its real-time constraints. To simulate the system it is necessary to replace the MARTe components (GACQM) that acquire data from sensors and send data to actuators with components which read stored signals from previous discharges and store the system response, respectively, so that it is possible to check whether the system behaves as expected. Even if in this case the system is being executed offline, it is also possible to collect useful information for real-time behavior, provided the same computer system is used for off-line and real-time operation. In fact, MARTe collects at any cycle the execution time of every component, which is then and stored by MDSplus along the other saved signals.

4. MARTe Simulink integration

As stated before, the possibility of automated translation from Simulink control blocks developed by control engineers into components for the real-time system represents an important factor in the overall quality of the system. The components for control in MARTe are represented by GAMs, and therefore the specific solution is a tool which translates a given Simulink block into a GAM. It is worth noting that there is a strong conceptual similarity between Simulink blocks and MARTe GAMs. A Simulink block defines parameters and input/output signals. Input/output signals will correspond to the GAMs real-time arguments which are connected to the other GAMs as specified in the configuration file. Simulink parameters will correspond to the GAM configuration parameters, possibly updated before the discharge. The Simulink Coder component of Matlab [6] provides the generation of C code implementing the functionality of a given block. Basically, what

Please cite this article in press as: G. Manduchi, et al., Integration of Simulink, MARTe and MDSplus for rapid development of real-time applications, Fusion Eng. Des. (2015), http://dx.doi.org/10.1016/j.fusengdes.2015.03.050

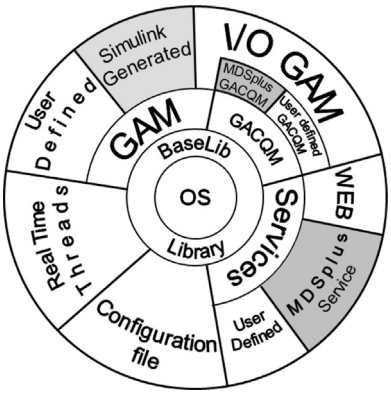

Fig. 3. The added MARTe components for MDSplus and Simulink.

is produced is the code for a set of routines receiving the inputs and producing the outputs by means of a set of C structures. An init() routine will initialize the environment, based on the passed parameters, and subsequent calls to the step() routine will advance the system computation. In order to use the code generated by the Simulink tool, it is necessary to develop some wrapper code which exports the block functionality as a MARTe GAM, that is a C++ class which inherits from a generic GAM interface class. The wrapper code depends in turn on the data structures defined for the inputs, the outputs and the parameters. Therefore, even if the code for the block functionality is generated by a Simulink tool, some manual coding is still necessary to make it a GAM. In order to completely avoid any manual coding, a python tool has been developed for the automatic generation of the wrapper code which interfaces the code generated by Simulink Coder with the contained MARTe GAM. This tool parses the C structure definition for inputs, outputs and parameters and produces the C++ wrapper code along with the include and configuration files needed to generate the shareable library expected by MARTe to manage the corresponding GAM. Fig. 3 highlights the added MARTe components for the integration of MDSplus and Simulink. The MDSplus driver provides support for real-time storage and generation of reference signals, carried out by Output and InputGAMs, respectively. The MDSplus service supervises all the MDSplus components, that is, all the instances of GACQM components storing signals in the pulse file and the generation of the reference waveforms. The Simulink generated GAMs, over the GAM layer, are generated by the python generator wrapping the Simulink generated code.

An important information for Simulink computation is the cycle time. In MARTe the cycle time, can be derived either from internal timers or via the clock given the ADC converters used to acquire sensor inputs. In both cases this period can be defined as a parameter in the MDSplus experiment model and then passed to MARTe in the system configuration phase. Changing the cycle period requires however the re-generation of Simulink code since it does not allow a dynamic cycle time definition.

5. Conclusions

The presented approach represents a ready-to-use solution, viable for both small and large systems. The combined usage of MARTe and MDSplus represents a possible solution for the rapid prototyping of new control systems which can be developed in the following steps:

- The algorithms to be used in real-time are developed using the Simulink environment for modeling and simulation.
- System component are described in the MARTe configuration file.
- MARTe components for control computation are generated from their Simulink definition.
- The set of signals of interest for every component is defined and reported in the MDSplus configuration.
- The resulting system is validated using stored input signals from previous discharges. This is achieved by replacing input/output components with others using MDSplus for data access.

This system is currently in use at RFX [7], making extensive usage of real-time MHD control, and in the ITER Neutral Beam Test Facility [8].

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] A. Hyatt, D.A. Humphreys, A. Welander, N. Eidietis, J.R. Ferron, R. Johnson, et al., Designing, constructing, and using plasma control system algorithms on DIII-D, IEEE Trans. Plasma Sci. 42 (2014) 421–426.
[2] W. Treutterer, R. Cole, K. Lüddecke, G. Neu, C. Rapson, G. Raupp, et al., ASDEX upgrade discharge control system—a real-time plasma control framework, Fusion Eng. Des. 89 (2014) 146–154.
[3] A.C. Neto, F. Sartori, F. Piccolo, R. Vitelli, G. De Tommasi, L. Zabeo, et al., MARTe: a multi-platform real-time framework, Trans. Nucl. Sci. 57 (2010) 479–486.
[4] G. Manduchi, T. Fredian, J. Stillerman, Future directions of MDSplus, Fusion Eng. Des. 89 (2014) 775–779.
[5] MARTe Web site at http://efda-marte.ipfn.ist.utl.pt/marte/
[6] G. Manduchi, A. Luchetta, A. Soppelsa, C. Taliercio, The new feedback control system of RFX-mod based on the MARTe real-time framework, Trans. Nucl. Sci. 61 (2014) 1216–1221.
[7] A. Luchetta, G. Manduchi, C. Taliercio, Fast control and data acquisition in the neutral beam test facility, Fusion Eng. Des. 89 (2014) 663–668.
[8] Simulink coder at http://www.mathworks.com/products/simulink-coder/

Please cite this article in press as: G. Manduchi, et al., Integration of Simulink, MARTe and MDSplus for rapid development of real-time applications, Fusion Eng. Des. (2015), http://dx.doi.org/10.1016/j.fusengdes.2015.03.050