Available online at www.sciencedirect.com
SCIENCE DIRECT®
Control Engineering Practice 14 (2006) 1387–1393
CONTROL ENGINEERING PRACTICE
www.elsevier.com/locate/conengprac

A flexible software for real-time control in nuclear fusion experiments

G. De Tommasi[a,*], F. Piccolo[b], A. Pironti[a], F. Sartori[b]

[a]Associazione EURATOM/ENEA/CREATE, Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II, Via Claudio 21, I-80125 Napoli, Italy

[b]Euratom/UKAEA Fusion Ass. Culham Science Centre, Abingdon OX14 3EA, UK

Received 8 February 2005; accepted 7 October 2005
Available online 10 November 2005

Abstract

JETRT is a software framework particularly suited for implementation of both real-time control and data acquisition systems. It is especially designed to work in a complex experimental environment such as the JET nuclear fusion facility. This new architecture maximizes the software reusability. The project-specific algorithm is compiled into a separate software component, in order to achieve a separation from the plant interface code. *JETRT* provides a set of tools to perform most of the validation phase on a Windows running desktop PC. Thanks to these design choices, both the development costs and the commissioning time have been reduced and even non-specialist programmers can easily contribute to the deployment of a new real-time system.

Keywords: Real-time systems; Object-oriented design and programming; Software tools

1. Introduction

Tokamaks are the most promising confinement devices in the field of controlled nuclear fusion: their principal objective is to contain a thermonuclear plasma by means of strong magnetic fields.

JET tokamak (Wesson, 2000) is the world's largest pulsed operated fusion experiment, where several computers interact in order to perform real-time control and monitoring services (Lennholm et al., 1999; Puppin et al., 1996).

Due to the complex environment it is important to standardize the coding practice as well as to separate the application software from its interfaces to the external system. Such an approach is the key to minimize development time, cost and to maximize reusability and efficiency.

JETRT is a new software framework used at JET to develop both real-time control and data acquisition systems. *JETRT* differs from the hardware and software architectures used in other nuclear fusion experiments (Behler et al., 1999; Luchetta and Manduchi, 1999; Moulin et al., 1998) because it has been developed to separate the algorithmic part of a real-time application (*User Application*) from the plant-interface software (*JETRTApp*). This design choice has been done to standardize the application development, to achieve portability among the different computer platforms, and to increase the code reusability.

JETRT framework has been successfully used to develop and test many systems among the *Real-Time Data Network* processing nodes (RTDN, Felton et al., 1999). For example the eXtreme Shape Controller (Ambrosino et al., 2003; Ariola et al., 2003) runs on a reliable PowerPC/VxWorks architecture with a latency time of 1 ms. Non-safety critical data acquisition systems run on a cheaper INTEL/WinNT4 platform with a latency time of 2–10 ms.

This paper gives an overview of the whole *JETRT* framework and describes in more details the architecture of real-time executor *JETRTApp*. The next section introduces the JET experiment. Section 3 deals with design choices and carries out a comparison between our approach and other design methodologies and technologies for real-time systems. In Section 4 an overview of the whole framework

*Corresponding author. Tel.: +39 0817683853; fax: +39 0817683816.
E-mail addresses: detommas@unina.it (G. De Tommasi), fpiccolo@jet.uk (F. Piccolo), apironti@unina.it (A. Pironti), fisa@jet.uk (F. Sartori).

0967-0661/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.conengprac.2005.10.001

is given. Sections 5, 6 introduce *JETRTApp* architecture, and timing issues. A short overview about how *JETRTApp* works with different I/O boards is given in the next section. Section 8 deals with the *User Application* plug-in, while the following section introduces the internal debugger feature available within the *JETRT* framework. Eventually, some concluding remarks are presented.

2. The JET experiment

In a fusion experiment the main aim is to obtain a plasma (a fully ionized gas) with the desired characteristics. This result cannot be achieved by simply pre-programming the actuators. Because of the various types of instabilities shown by the plasma, several corrective actions have to be taken.

The constraints on the latency times of the control systems have always to be met: the systems containing these tasks can be classified as hard real-time systems, in accordance with the operational definition given in Liu (2000). At JET typical cycle times for control loops range between 50 μs (Vertical Stability System, Lennholm et al., 1997) and 1 ms (eXtreme Shape Controller).

JET is a pulsed machine: this means that an experiment is performed every 20–60 min, during which the plasma is formed and sustained for about 1 min. As a consequence of such a cycle, all the real-time applications operate in two different modalities: ON-LINE and OFF-LINE. During the experiment, the systems are in the ON-LINE mode and they perform real-time measurement, control or protective actions. In this phase all the communications are disabled and the hard real-time constraints on the latency time have to be met, while there is no need to check such constraints in the OFF-LINE mode.

When a system is in the OFF-LINE mode it can interact with the other JET subsystems. These interfaces are the main sources of technical complexity for the applications. Before and after every pulse the *Plant Supervisor* sends to all JET subsystems a change-of-state request to synchronize their evolution. As soon as the scientists have finished to programme a new experiment, the *Level*-1 plant management system sends a message to each real-time node, containing the new set-up parameters. Eventually, the information collected during the pulse is sent to General Acquisition Program (*GAP*), which is the data management system.

JETRT framework, helps working in this environment, answering the need for a fast and reliable deployment of new systems.

3. JETRT design choices

Along the last 20 years, several real-time systems have been deployed at JET. Since the very beginning, most of a project code was recycled during the implementation of the next one, hoping to save development and testing time. While this practice proved to be very helpful in reducing programming costs, it eventually appeared to have too many shortcomings:

- the hardware-related details where mixed with the application ones. In order to test the same application on a different platform, it was necessary to emulate the target hardware.
- Once a specific hardware platform had run out of its commercial life, the migration to a new platform required an almost complete re-writing of the code.
- Only the people with enough knowledge of the platform could re-use the existing software.

Since JET is an experimental environment, each real-time system is realized only once, therefore, the costs of a prototype cannot be diluted as in a mass production. For this reason the system architecture needs to satisfy two additional requirements:

(1) use components on the shelf as much as possible;
(2) ensure enough processing power, such as to satisfy the timing constraints.

At the same time it was observed that, in the ON-LINE mode, all of the deployed systems, despite their complexity, could actually be reduced to the simple iterative model shown in Fig. 1.

Once the standard hardware architecture has been chosen, the real-time application has to be designed to realize the iterative cycle introduced above, when in the ON-LINE mode. In the OFF-LINE mode, the application has to accomplish all the communications and ancillary tasks. These tasks are the same for all the JET applications and therefore this part of the code can be re-used when a new system is developed.

It follows that the definition of the ON-LINE processing task is the only thing to do when creating a new real-time application.

The scheduling policy adopted in the *JETRT* framework is the *priority-based pre-emptive* (Liu, 2000) with the granularity equals to the minimum allowed by the platform (10 ms for INTEL/WinNT4 and 200 μs for PowerPC/

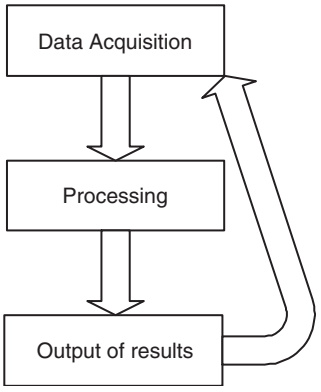

Fig. 1. JETRTApp iterative model.