Functionality of DAQ-Middleware

Yoshiji Yasu, Kazuo Nakayoshi, Hiroshi Sendai and Eiji Inoue

***Abstract*—Functionality of DAQ-Middleware is presented with an introduction to DAQ-Middleware followed by its features, including a detailed description of DAQ Components with an example of the architecture, system configuration related to the configuration database and condition database, Web interface, and finally remote booting of DAQ Components.**

DAQ-Middleware is already used in experiments at J-PARC (Japan Proton Accelerator Research Complex), Tokai, Japan.

***Index Terms*— RT-Middleware, Database, Web**

I. Introduction

WHAT is DAQ-Middleware [1, 2, 3, 4, 5]? It is a software framework of a network-distributed DAQ system based on Robot Technology (RT) Middleware [6]. RT-Middleware (RTM) is an international standard of the Object Management Group (OMG) not only for Robotics but also for embedded systems. RT-Middleware provides the integration of RT systems as a Robotic Technology Component (RTC), which is the software unit. An RTC is a logical representation of a hardware and/or software entity that provides well-known functionality and services. The software package of RT-Middleware was developed by the National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Japan [7].

The package can run on several operating systems such as Linux and Windows. It also supports C++, Python, Java and .NET as programming languages. The RTC on the packages has several features as follows:

- Execution context, which has a simple state machine and behaves as the main thread for executing core logic.
- Data port, which is used for continuous data flow. It has InPort as input and OutPort as output. CORBA and TCP/IP socket as the communication protocol are available. Multiple ports can be implemented in a RTC. A data port behaves as a thread.
- Service port, which is defined by the user as the CORBA IDL for sending and getting its own parameters in the RTC.

Why did we adopt RT-Middleware? Many DAQ systems so far could share the device driver and the library. Thus, they could make their own DAQ system using the common software. However, they could not share the software on the DAQ Component level. Sometimes experimental groups could share a common DAQ system. We rather like to use not the level of the DAQ system, but the DAQ Component because of its flexibility and reusability. Why could we not share the DAQ Component? Because of there is no international standard of the DAQ Component level. DAQ-Middleware is one of the solutions.

DAQ-Middleware provides the integration of the DAQ system and the DAQ Component is the software unit. DAQ-Middleware in Fig. 1 is based on RT-Middleware and has been extended for the DAQ system. The main extensions are as follows:

- Extended state machine
- Run control mechanism
- XML-based configuration mechanism
- Web-based system interface

Those extensions are implemented on the DAQ Operator and DAQ Component.

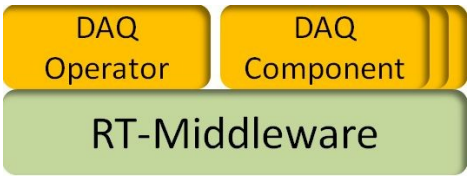

Fig. 1. DAQ-Middleware.

Following, the feature of the DAQ Component and DAQ Operator is explained.

- DAQ Component provides the following functionalities:
  - Data path for receiving and sending data
  - Command/status path for getting system configuration parameter
  - Command/status path for getting DAQ command
  - Command/status path for sending DAQ status
  - Reading, parsing and setting condition parameters
  - Core logic with handling both paths
- DAQ Operator provides the following functionalities:
  - Dynamic connection of DAQ Components of data path and command/status path
  - Controlling DAQ Components via command/status path
  - Reading, parsing configuration and sending the parameters to DAQ Components
  - System interface (Web interface)

Manuscript received June 7, 2009; revised September 28, 2009

Y. Yasu is with High Energy Accelerator Research Organization, 1-1 Oho, Tsukuba, Ibaraki, 305-0801, Japan (corresponding author to provide phone: +81298645383; fax:+81298642580; e-mail: Yoshiji.Yasu@ kek.jp).

K. Nakayoshi is with High Energy Accelerator Research Organization, 1-1 Oho, Tsukuba, Ibaraki, 305-0801, Japan (e-mail: Kazuo.Nakayoshi@ kek.jp).

H. Sendai is with High Energy Accelerator Research Organization, 1-1 Oho, Tsukuba, Ibaraki, 305-0801, Japan (e-mail: Hiroshi.Sendai@ kek.jp).

E. Inoue is with High Energy Accelerator Research Organization, 1-1 Oho, Tsukuba, Ibaraki, 305-0801, Japan (e-mail: Eiji.Inoue@ kek.jp).

II. Autonomous DAQ Component

The DAQ Component inherits the InPort/OutPort of RTC for data port and Consumer/Provider of RTC for the service port [6, 7]. Fig. 2 shows the autonomous DAQ Component. The data port and the service port are used for data path and command/status path, respectively. The main thread is used for core logic including management of command/status flow and data flow. The DAQ Component does not work just in the client/server model, but is autonomous with its own state machine. The state machine is shown in Fig. 3.

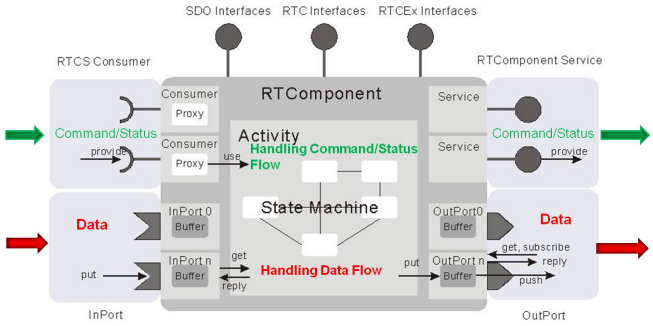

Fig. 2. Autonomous DAQ Component

RTC has a simple state machine. The DAQ state machine is defined in the active state of RT-Middleware. There are four states and eight transitions. The state machine is tightly coupled with the run control on the DAQ Operator and DAQ Component. DAQ-Middleware does not provide the framework of easily adding a new state and removing a previous state.

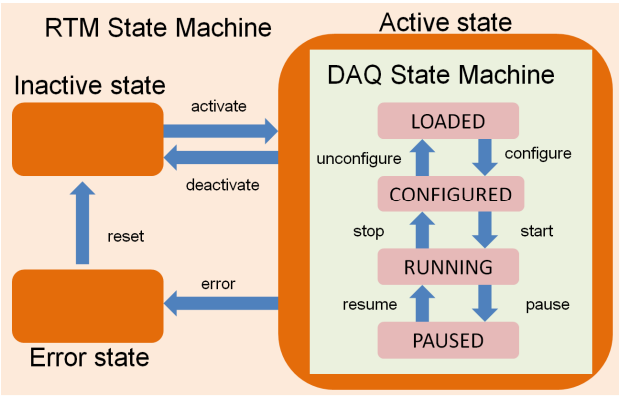

Fig. 3. DAQ state machine

III. Types Of DAQ Components

There are several types of DAQ Components shown in Fig. 4.

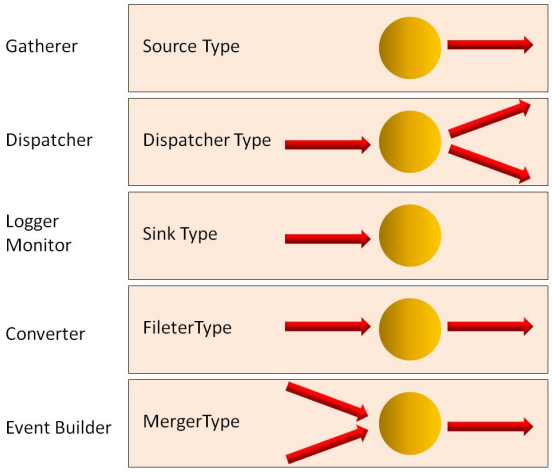

Fig. 4. Types of DAQ Components

One is source type, which is the source of the data to be sent to another Component. An example of this type is the Gatherer Component for data readout. Another one is sink type. Data received from another Component will be processed, but will not be sent to other Component. Logger Component for logging data and Monitor Component for data analysis belong to this type. Dispatcher type is necessary to send data to multiple paths. By using those types of Components, a basic DAQ unit was developed for the DAQ system of J-PARC/MLF experiments [3, 4]. J-PARC stands for Japan Proton Accelerator Research Complex. This is a high intensity proton accelerator facility using MW-class high power proton beams at both 3 GeV and 50 GeV. MLF stands for Materials and Life Science Facility, which is aimed at promoting materials science and life science using the world's highest intensity pulsed neutron and muon beams produced using 3 GeV protons. The requirement of total throughput on data path is about 30 MB/sec and DAQ-Middleware meets the requirement.

Additionally, filter type is necessary for conversion of data. Merger type will be used for building events from event fragments or simply gathering data, and then sending them to another Component. Those types of DAQ Components are not used for MLF now, but generally useful.

The combination of DAQ Components will make the DAQ configuration flexible. For example, Gatherer and Monitor Components are dependent on the detectors. Thus, Gatherer and Monitor were developed for each detector. However, Dispatcher and Logger Components are commonly used for the MLF DAQ system. Each experiment can choose the combination.

IV. Architecture

The architecture of DAQ-Middleware is illustrated in Fig. 5. In Fig. 5, the DAQ Operator Component and System Interface are described in a frame for DAQ Operator. This means that the DAQ Operator including the System Interface runs on a PC. Similarly, a unit in which a Gatherer, a Dispatcher, a Logger and a Monitor run on a PC, is defined as a basic DAQ unit.

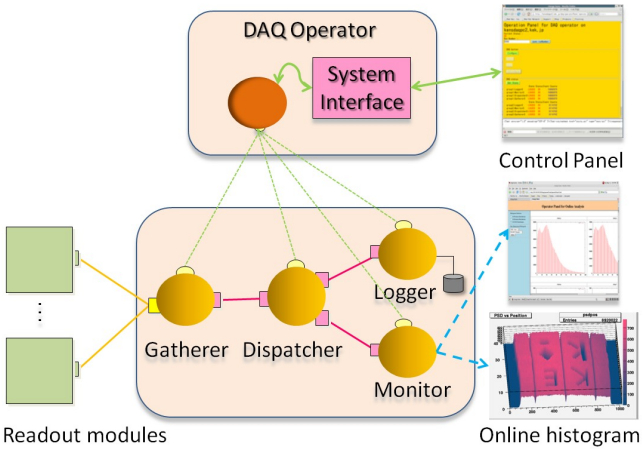

Fig. 5. The architecture of DAQ-Middleware

The DAQ Operator reads and parses the system configuration written in XML and then stores the parameters as pairs of key and value. When the control panel issues a "configure" command, the DAQ Operator receives the command via the System Interface and then sends the parameters to the DAQ Components. In the same way, the control panel issues "start" and "stop" commands to control the readout, the data logging and the online data analysis. The DAQ Operator can manage multiple basic DAQ units. The readout module uses the hardware-based TCP [11] processor called SiTCP [12]. The SiTCP is a general purpose front-end interface. From a network-distributed DAQ system point of view, it is better that the readout protocol of the front-end is TCP/IP while the transfer speed of SiTCP can reach the hardware limitation of Ethernet. On the other hand, DAQ-Middleware is independent of the readout protocol. The online histogram using ROOT [13] and online histogram displaying on the Web browser are now available. There are two types of display methods. One is to use X window. Another is to browse image files from a Web browser. In Fig. 5, the online histogram using ROOT (bottom in Fig. 5) uses the former while the ROOT libraries are linked to the Monitor Component. Another one (middle in Fig. 5) uses the latter.

V. Configuration & Condition Databases

System configuration and condition parameters are written in XML documents as text files. The system configuration file is not often modified after the configuration is decided. On the other hand, the condition file describes the parameters of the readout hardware module or analysis. The parameters may be often changed.

In Fig. 6, the configuration is parsed by the DAQ Operator and the parameters are sent to the DAQ Components.

In Fig. 7, the condition is automatically converted to a JSON document by Xalan [14] and is then parsed by the DAQ Component. JSON document [15] was adopted instead of XML document because the DAQ Component should be as lightweight as possible. After parsing in both cases, a parameter is stored as a pair of key and value in the DAQ Component.

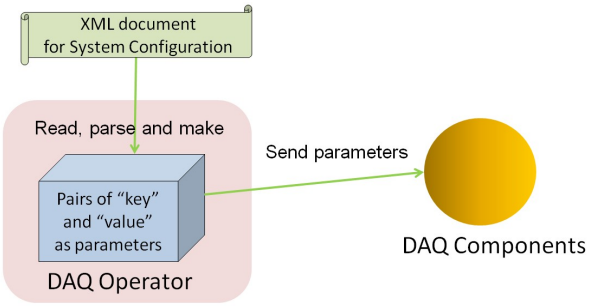

Fig. 6. System configuration mechanism

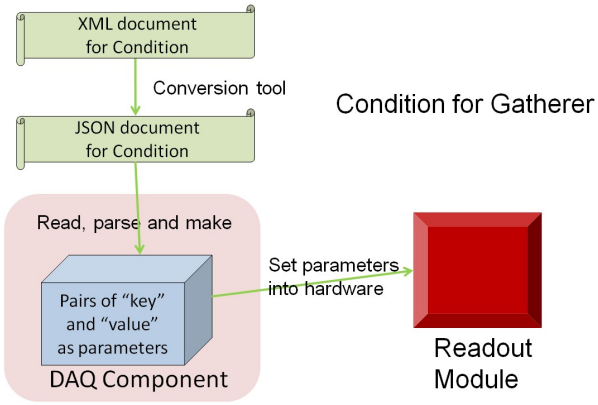

Fig. 7. Condition mechanism

VI. Communication With Web Interface

DAQ-Middleware uses the XML/HTTP protocol [16] to communicate with a Web client or other external world.

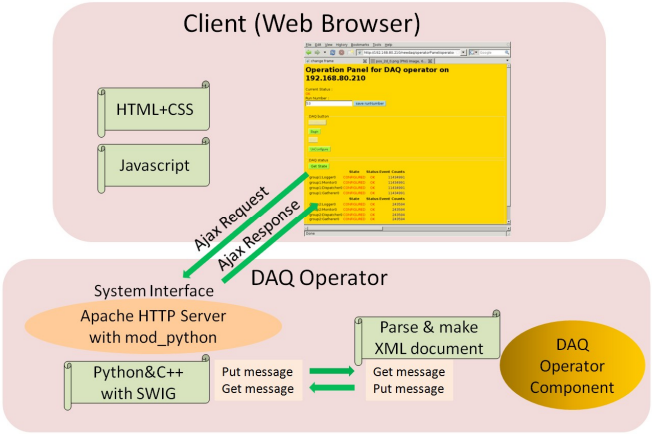

Fig. 8. Web interface

In Fig. 8, the DAQ Operator receives commands and sends statuses from/to the Client (Web Browser) via the System interface, which consists of an Apache HTTP server with a mod_python module and message passing program to communicate with the DAQ Operator Component. Ajax technology [17] has an important role in HTML [18] and JavaScript (ECMAScript) [19] programming to reduce the workload. For example, when "Get Status" button in Fig. 8 is

clicked, Ajax Request written in JavaScript gets DAQ status as XML document (Ajax Response) asynchronously and then changes only status items in HTML on the client side instead of changing the whole page.

VII. Remote Booting Mechanism

DAQ Components will run on multiple PCs. For the booting mechanism, xinetd [20] was adopted. The remote booting mechanism is shown in Fig. 9. The script to run DAQ Components on a PC will be sent to the PC, and then xinetd daemon running on the PC will execute the script to run the DAQ Components. After the DAQ Operator and DAQ Components run, the DAQ Operator connects data paths and command/status paths of DAQ Components each others according to the configuration, by using RT-Middleware's naming service daemon, in which the object references of the DAQ Components are already registered.

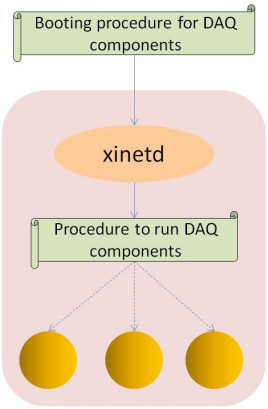

Fig. 9. Remote booting mechanism

VIII. Conclusion

DAQ-Middleware provides a minimum set of functions for a network-distributed DAQ system with autonomous DAQ Components configured by databases in cooperation with a Web system.

Acknowledgement

We wish to thank Prof. J. Haba at KEK for his support of this project. This work was performed in the next generation DAQ sub-group of the KEK Detector Technology Project at KEK. We also wish to thank N. Ando, T. Kotoku and S. Hirano at AIST for RT-Middleware support. We thank the staff of experimental groups at J-PARC/MLF and KEK electronics system group for their help.

References

[1] Y. Yasu, E. Inoue, K. Nakayoshi, H.Fujii, Y.Igarashi, H. Kodama, N. Ando, T. Kotoku, T. Suehiro, S. Hirano, "Feasibility of data acquisition middleware based on robot technology," CHEP2006 Macmillan Advanced Research Series p458-461

[2] Y. Yasu, K. Nakayoshi, E. Inoue, H. Sendai, H. Fujii, N. Ando, T. Kotoku, S. Hirano, T. Kubota, and T. Ohkawa, "A Data Acquisition Middleware," 15th IEEE NPSS Real Time Conference, 2007, doi:10.1109/RTC.2007.4382850

[3] K. Nakayoshi, Y. Yasu, E. Inoue, H. Sendai, M. Tanaka, S. Satoh, S. Muto, N. Kaneko, T. Otomo, T. Nakatani, T. Uchida, "Development of a data acquisition sub-system using DAQ-Middleware," Nucl. Instr. and Meth. A 600 (2009) 173-175.

[4] K. Nakayoshi, Y. Yasu, H. Sendai, E. Inoue, M. Tanaka, S. Sato, S. Muto, J. Suzuki, T. Otomo, T.Nakatani,, T. Ito, Y. Inamura, M. Yonemura, T. Hosoya, T. Uchida, "DAQ-Middleware for MLF/J-PARC," presented at The 1st international conference on Technology and Instrumentation in Particle Physics, March12-17, 2009, Tsukuba, Japan

[5] Y. Yasu, K. Nakayoshi, H. Sendai, E. Inoue, M. Tanaka, S. Suzuki, S. Satoh, S. Muto, T. Otomo, T. Nakatani, T. Uchida, N. Ando, T. Kotoku, S. Hirano, "Development of DAQ-Middleware," presented at 17th International Conference on Computing in High Energy and Nuclear Physics, 21 - 27 March 2009 Prague, Czech Republic

[6] Object Management Group, "Specification of Robotic Technology Component (RTC)," Version 1.0, Available: http://www.omg.org/spec/RTC/1.0/

[7] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, W. K. Yoon, "RT-Middleware: Distributed Component Middleware for RT (Robot Technology) ," 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2005), pp.3555-3560, 2005.08, Edmonton, Canada

[8] World Wide Web Consortium, "Extensible Markup Language (XML) ," [Online]. Available: http://www.w3.org/XML/

[9] Apache Software Foundation, Apache HTTP Server Project, [Online]. Available: http://httpd.apache.org/

[10] Apache Software Foundation, "Apache/Python Integration," Available: http://www.modpython.org/

[11] Internet Engineering Task Force, "TRANSMISSION CONTROL PROTOCOL," RFC 793

[12] T. Uchida, "Hardware-Based TCP Processor for Gigabit Ethernet," IEEE Trans. Nucl. Sci. NS 55 (2008) 1631.

[13] The ROOT Team, "ROOT," [Online]. Available: http://root.cern.ch/drupal/

[14] Apache Software Foundation, "Xalan," [Online]. Available: http://xalan.apache.org/

[15] Crockford D, "The application/json Media Type for JavaScript Object Notation (JSON) ," RFC 4627

[16] World Wide Web Consortium, "HTTP - Hypertext Transfer Protocol," [Online]. Available: http://www.w3.org/Protocols/

[17] World Wide Web Consortium, "Ajax," [Online]. Available: http://www.w3.org/TR/2006/ WD-XMLHttpRequest-20060405/

[18] World Wide Web Consortium, "HTML," [Online]. Available: http://www.w3.org/html/

[19] ISO/IEC, "ECMAScript," ISO/IEC 16262:2002

[20] Panos Tsirigotis , "xinetd," [Online]. Available: http://www.xinetd.org/