

# Feasibility of Data Acquisition Middleware based on Robot Technology

Y.Yasu, E.Inoue, K.Nakayoshi, H.Fujii, Y.Igarashi, H.Kodama, KEK, Tsukuba, Japan,  
N.Ando, T.Kotoku, T.Suehiro, Intelligent Systems Research Institute, AIST, Tsukuba, Japan,  
S.Hirano, Information Technology Research Institute, AIST, Tsukuba, Japan

## ABSTRACT

In the situation that many programming languages and communication protocols are flooded, it is necessary to design DAQ system independent of programming languages, operating systems and communication protocols, because the rapidly growing Information Technology (IT) and the usage of many PCs in DAQ system lead us to increase necessary DAQ experts and the workload to make the software. In robot technology field, they also had same situation. It was necessary to make an infrastructure of robot systems. Then, **Robot Technology Middleware (RTM)** was born. AIST led the project and then developed a software package called OpenRTM-aist. From software point of view, the basic technologies are similar to that of DAQ and the model of software development process is also powerful for DAQ. Thus, we studied the feasibility of **Data Acquisition Middleware** to make a DAQ software framework.

## ROBOT TECHNOLOGY MIDDLEWARE

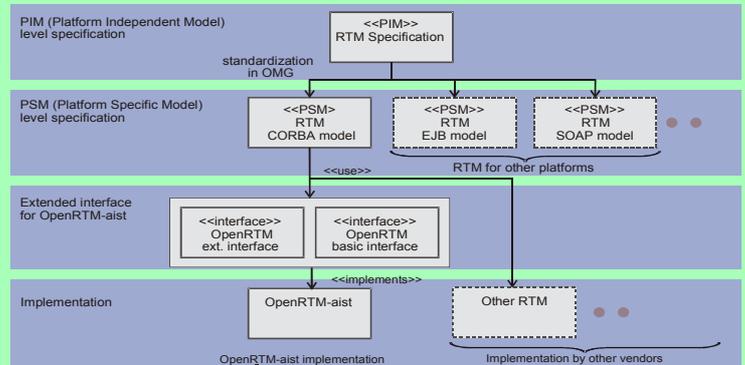


Figure 1: RTM development process model.

In the modeling flow of robot system, the system design was dependent of person's experience and know-how so far. However, RT system should be modeled and designed through systematic design flow independent from the person's ones. Object Management Group (OMG) has a model of software development process called **Model Driven Architecture (MDA)**. It has a hierarchy including **Platform Independent Model (PIM)** and **Platform Specific Model (PSM)** layers. RTM adopted the model. AIST now proposes a RTM development process model shown in Figure 1. It has four layers, namely, PIM, PSM, Extended interface for **OpenRTM-aist** and implementation. The PIM defines resource data model and the interfaces to access and manipulate resource data, described in **Unified modeling Language (UML)**. In the PSM, the interfaces and data structures used in the individual methods are mapped according to a **CORBA IDL** specification. It is still independent of programming languages and operating systems. In the OpenRTM-aist implementation, **C++** and **Python** were adopted as programming language.

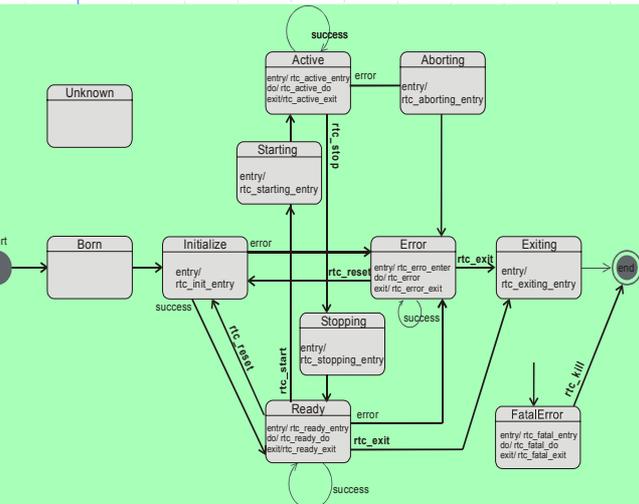


Figure 3: State Machine Diagram of RT-Component Activity.

A **RT-Component** is an autonomous object. It has an activity which always continues processing its task, and the activity serves as a subject of a device control, such as a robot. The **State Machine Diagram of RT-Component's Activity** and the **DAQ/RTM state mapping** are shown in Figure 3 and Figure 4, respectively. The "entry" methods and "exit" methods in Figure 3 are called only when changing the states while the "do" methods are continuously called in the steady states. DAQ component also has its own states in Figure 4. A configuration method is called when a configuration command is issued while run method is continuously called after a start command is issued. The state mapping between DAQ and RTM was successfully done in this case.

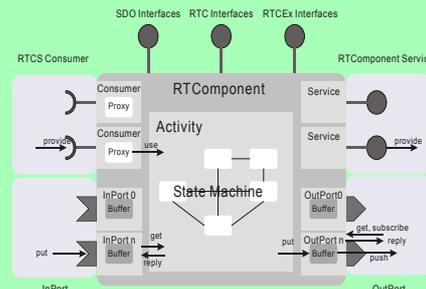


Figure 2: RT-Component Architecture.

Figure 2 shows RT-Component Architecture. RT-Component is basic functional unit of RT-Middleware based systems. RT-Component consists of several objects. **InPort** and **OutPort** are data stream input and output, respectively. RT-Component service (RTCS) and RTCS consumer are service interfaces for user command and reply to change internal attributes from outside without re-compiling. SDO interfaces, RTC interfaces and RTCEX interfaces are service interfaces for common command and reply to manage RT-Components.

## DATA ACQUISITION MIDDLEWARE

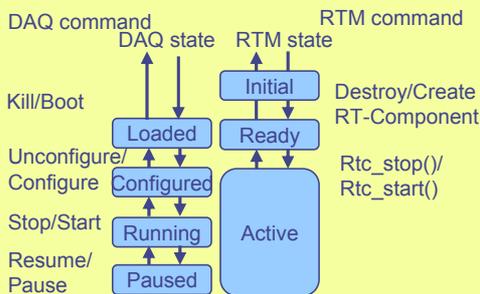


Figure 4: DAQ/RTM state mapping

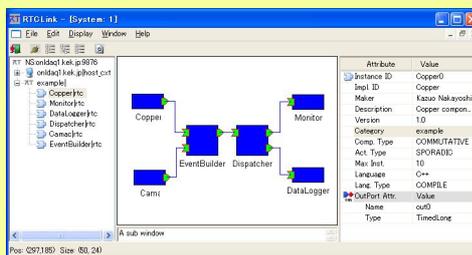


Figure 5: A snapshot of rtc-link

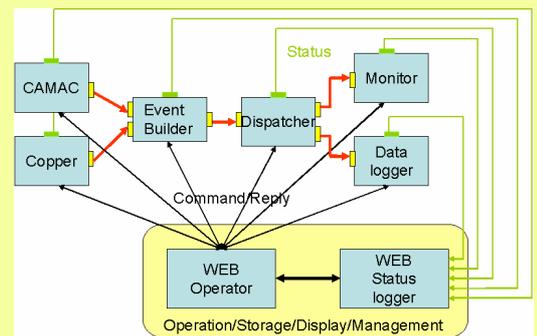


Figure 6: An example of DAQ Demonstrator

## DAQ DEMONSTRATOR

We made **DAQ demonstrators** to evaluate the OpenRTM-aist. Figure 5 shows a snapshot of a DAQ demonstrator used with a RTM system tool called rtc-link. The schematic diagram of DAQ demonstrator is shown in Figure 6. It consists of **data collectors** from CAMAC and Copper\*, a **2x1 event builder**, a **data dispatcher**, a **data logger** to a file system, a **monitor** (analyzer with ROOT), **WEB operator** and **WEB status logger**. The WEB operator can issue DAQ commands such as "start", via RTC service while the WEB status logger can get the status from the DAQ components via OutPorts. For developing the components, **rtc-template** to generate the C++ codes was used. Thus, we could concentrate on coding the contents of the DAQ function. As the result, the development time could be reduced.  
\* A new DAQ readout framework developed by KEK.

## DISCUSSION

- **Model of software development process** – RTM adopted the MDA as the model. It is a good way to design a platform independent DAQ system. RTM also adopted component-oriented approach. It is excellent from the reusability point of view.
- **Mapping DAQ components to RT components** – DAQ states could be mapped to the RTC states in the demonstrators.
- **Development time** – RTM has the system tools and the template program of RT-Components is also provided. It is useful to reduce the development time to construct DAQ system.
- **Dynamic creation and binding** – InPort and OutPort can be dynamically connected while RT-Component can be dynamically created.

## CONCLUSION

The feasibility of **Data Acquisition Middleware** was studied and discussed. **Robot Technology Middleware** was introduced. The **RTM-based DAQ demonstrator** was made and evaluated. In the demonstrators, the DAQ states were successfully mapped to the RT states. **The study showed that RTM had good features for DAQ and then Data Acquisition Middleware was feasible. More investigation is expected.**