Final Program & Digest Book

SICE-ICCAS 2006

SICE - ICASE International Joint Conference 2006

October 18 - 21, 2006

BEXCO(Busan Exhibition & Convention Center), Busan, KOREA

The Society of Instrument and Control Engineers (SICE), Japan (http://www.sice.or.jp)

The Institute of Control, Automation and Systems Engineers (ICASE), Korea (http://icase.or.kr)

SICE-ICASE International Joint Conference 2006
Oct. 18-21, 2006 in Bexco, Busan, Korea

RT(Robot Technology)-Component and its Standardization
– Towards Component Based Networked Robot Systems Development –

Noriaki Ando, Takashi Suehiro, Kosei Kitagaki and Tetsuo Kotoku

National Institute of Advanced Industrial Science and Technology (AIST)
AIST Tsukuba Central 2,Tsukuba,Ibaraki 305-8568, Japan
{n-ando, t.suehiro, k.kitagaki, t.kotoku}@aist.go.jp

Abstract: In this paper, we propose RT-Component architecture for robot system integration. We have studied modularization of RT (Robot Technology) elements and have developed RT-Middleware, which promotes application of RT in various field[1], [2], [3]. RT-Middleware is a software platform for RT systems and provides RT specific functionality for the component based system development. The component which constructs RT-systems in RT-Middleware is called the "RT-Compoment". We are also standardizing the architecture of RT-Component in the Object Management Group (OMG). Finally conclusion and future work will be described.

Keywords: RT (Robot Technology), component, OMG, standardization

1. INTRODUCTION

In recent years, the importance of research on system integration of robotic technologies bases on accumulated knowledge of robotics research is increased. The technologies used in intelligent systems, which can be applied not only to standalone robots but also to ubiquitous computing and other more intelligent electrical devices, we call "Robotic Technology" (RT).

From such backgrounds, the necessity for the knowledge of systematic robot system integration and the development of the implementation platform has increased. As shown in Figure 1, the methodology for robot system integration independent from persons' experience and knowhow, and robot system platform to support it are also needed for systematic RT-system development.

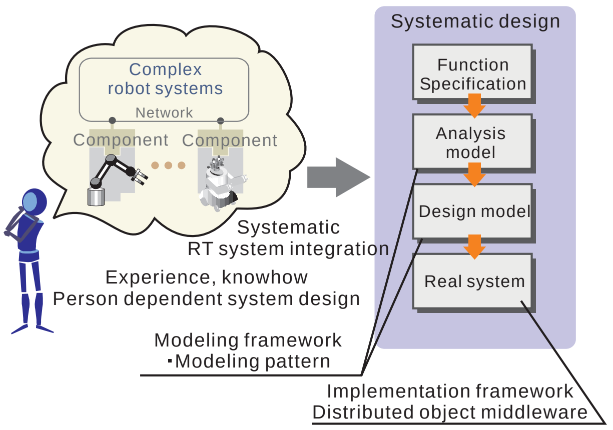


Fig. 1 A Robot Systems Modeling Flow: The RT system should be modeled and designed through systematic design flow independent from the persons' experience and knowhow.

We have studied integral elements for robot software platform, and the RT-Middleware has been proposed. The purpose of RT-Middleware is to establish basic technologies for integrating robot systems with a new function easily by using modularized software components.

If robot systems with new functions can be constructed more flexibly, it can satisfy every users' needs individually which cannot be satisfied now. Thus, it is expected that the conventional robot industry mainly restricted to the manufacturing field will be expanded to the nonmanufacturing field like support robots for daily life.

1.1 Related research

The research on software modularisation of robot functions and development of software libraries for robotics are performed actively in recent years.

ORiN (Open Resource interface for the Network/Open Robot interface for the Network) is a middleware which offers the standard communication interface over various FA apparatus including robots [4], [5]. The specification of ORiN is released from the ORiN consortium, in which almost all the Japanese industrial-robots maker has affiliated. **Orocos** (Open Robot Control Software / Open Realtime Control Services) is a free software project for robotics [6], [7]. **ORCA** is a project to develop an Open-Source Robotic Control System derived from OROCOS project [8]. Another **ORCA** (Open Robot Controller Architecture) developed at Toshiba is a HORB (Java ORB developed in AIST) based robot controller architecture [9], [10]. In Toshiba's home robot "Apri-Alpha", functional components are integrated and controlled using ORCA. **OPEN-R** is the standard interface for the entertainment robot system that SONY is actively promoting. This interface greatly expands the capabilities of entertainment robots like AIBO, SDR-3X and QRIO[11]. **OSCAR** (Operational Software Components for Advanced Robotics) is an object-oriented framework for the development of control programs for robotic manipulators [12]. **URC** (Ubiquitous Robot Companion) Project is a fusion of robot, network and information technologies, in which software robot and embedded robot are also introduced as well as the conventional robot [13]. **Player/Stage** is a free software project for research in robot and sensor systems. The Player, which is a robot server with robot control interface, and its simulation backends, Stage and Gazebo, are very widely used [14].

1.2 Goals of RT-Middleware

The purpose of RT-Middleware project is research and development of the middleware for robots, which supports efficient development of robot systems. RT-Middleware aims at the spread of an open robot system architecture to contribute to robot market activation.

Buisiness model of robot market

Until now, only some makers with the synthetic technical capabilities of hardware and software have been able to participate in the robot market. If a standardized robot system architecture spreads, the makers with technical capabilities of software, can also get into the market as "a robot system integrator."

Wide application of RT

The software platform is an infrastructure to improve flexibility of robot system integration. Flexibility of system integration is needed to apply RT to a wide variety of market needs. Component based software development (CBSD) supported by RT-Middleware realizes flexible software and robot system integration.

Robotics research tool

A researcher has only to develop his/her logic or algorithm as a component, and he/she can build a new system by combining with other available components. If a researcher want to try some algorithms to specific part of the system, a researcher has only to replace a related module with new one implemented with his/her new algorithm. Therefore the efficiency of an experiment will improve. From academic view, robotics research can be shifted to research of integration technology by robot technology componentization.

1.3 Open Specification and Standardization

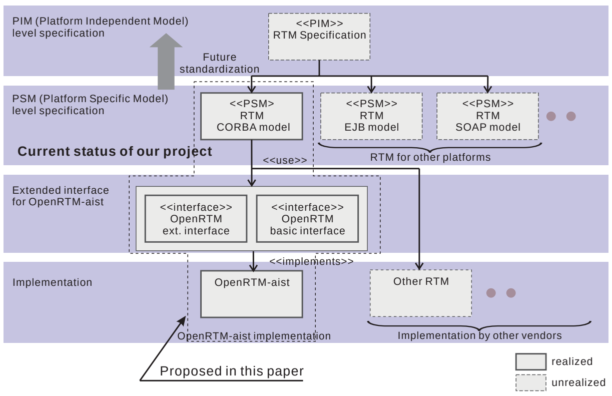

PIM (Platform Independent Model) level specification
<<PIM>> RTM Specification
Future standardization
PSM (Platform Specific Model) level specification
<<PSM>> RTM CORBA model
<<PSM>> RTM EJB model
<<PSM>> RTM SOAP model
Current status of our project
<<use>>
RTM for other platforms
Extended interface for OpenRTM-aist
<<interface>> OpenRTM ext. interface
<<interface>> OpenRTM basic interface
<<implements>>
Implementation
OpenRTM-aist
Other RTM
OpenRTM-aist implementation
Implementation by other vendors
Proposed in this paper
realized
unrealized

Fig. 2 The RT-Middleware (RTM) standardization process and the relation between RTM specification and our implementation of OpenRTM-aist.

For the above-mentioned purpose, we defined RT-Middleware object model as an open specification, which is a set of interface and its component model of distributed object middleware for RT functional element, and we are also trying to standardize this specification. If the "RT-Component" becomes an open standard, anyone can develop own implementation on any platform in any language,

Moreover, an open source implementation named "OpenRTM-aist" has been developed for the purpose of obtaining the feedback from many robot research fields. Figure 2 shows the relation between RTM (RT-Middleware) specification and our implementation of OpenRTM-aist.

In this paper, we propose new integration scheme, which is derived from component based software development (CBSD), for robotic systems. The functions required for the middleware which supports robot specific features are discussed and clarifies. To realize component based robotic system development efficiently, RT-Component and its architecture will be proposed. We have already proposed RT-Middleware and have implemented it, and they are redesigned for standardization and generalization. Finally conclusion and future work will be described.

2. MODULARIZATION OF RT ELEMENTS

In this section, structural differences in the modularization of the simple distributed object and the RT functional element are clarified, and the core architecture design of RT-module is discussed. We considered what kind of function required for a modularization of the RT element would be realized in the framework of distributed object middleware. As one of the views of a modularization of RT element, "RT-Component" was proposed. Required functions, a structure and a realization method based on distributed objects for the RT-Component were also reviewed.

2.1 RT specific functions

Granularity of modules

When modularizing an RT element, a module of various granularity size can be considered. Modules of fine granularity level, such as a motor, a sensor, and a controller. Modules of middle-fine granularity level, such as a vision system with some image processing algorithms, a several degrees of freedom manipulator arm and a mobile robot with some sensors. Modules of rough granularity level, such as a humanoid robot with legs, arms and vision system, an intelligent room with distributed sensors and robots. It is necessary to provide a framework which can choose such various granularity freely in RT-Middleware.

Active module

Usually, a general distributed object works as a passive object that sends back return values to a method invocation. In this case, an object is modeled as interfaces that contain operations with input and output parameters and a return value. An internal activity model of an object is not considered.

On the other hand, an RT element has its own tasks like real-time feedback control. Furthermore, it is necessary to collect required data RT-element itself, or to notify event to other elements when it happened.

Rrealtime capabilities

Realtime capabilities of module activity is a indispensable function in RT systems. RT-Middleware should support realtime capabilities in its software module as a framework.

Realtime capabilities between modules

RT system requires the high speed communication and close cooperation with other modules, such as a servo control.

Time management

For example, a servo control has to be performed under stable periodic real-time task. In order to make two or more modules cooperate in the real-time schedule, the time synchronization between modules, which may be running on distributed host machines, is needed.

Software reuse

Users are unwilling to use the framework which needs to remake all programs. In order to reuse a lot of software library created until now, it is necessary to provide the framework for modularizing the existing software library easily.

Platform independent middleware

In order to improve the reusability of software, the middleware has to be modeled on the platform (in this context, "platform" means operating systems) independent abstraction level.

Network independent middleware

RT-Middleware has to support various communication media and its model should have independent structure from them. If a real-time communication media is available, modules that depend on realtime features should use it.

3. RT-COMPONENT ARCHITECHTURE

RT-Component is basic functional unit of RT-Middleware based systems. Figure 3 shows the architecture block diagram of the RT-Component.

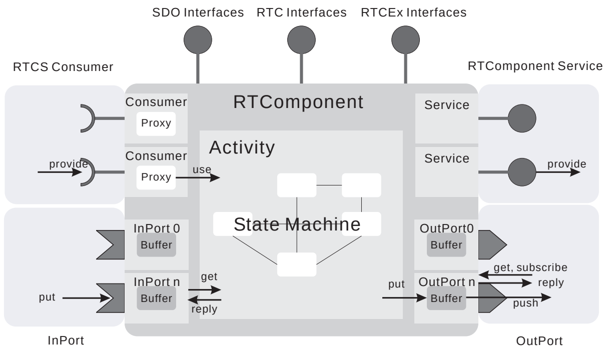


Fig. 3 A proposed architecture of RT-Component. RT-Component has a component object, command interfaces, an activity, InPorts and OutPorts.

We chose CORBA (Common Object Request Broaker Architecture) as distributed object middleware for its platform independency, and tried modeling of the RT-Component on CORBA.

An RT-Component consists of the following objects and interfaces.

- Component object.
- Activity.
- InPort as input port object.
- OutPort as output port object.

The general distributed object model can be described as some interfaces that contain operations with parameters and a return value. On the other hand, the RT-Component model has a component object as a main body, activity as a main process unit, input ports (InPort) and output ports (OutPort) as data stream ports.

3.1 Activity and state transition

An RT-Component itself has an activity which always continues processing its tasks, and activity serves as a subject of a device control, such as a robot.

The activity of RT-Component has eleven states: `BORN, INITIALIZE, READY, STARTING, ACTIVE, STOPPING, ABORTING, ERROR, EXITING, FATALERROR, UNKNOWN.` Figure 4 shows the state transition chart (UML state chart) of RT-Component's activity.

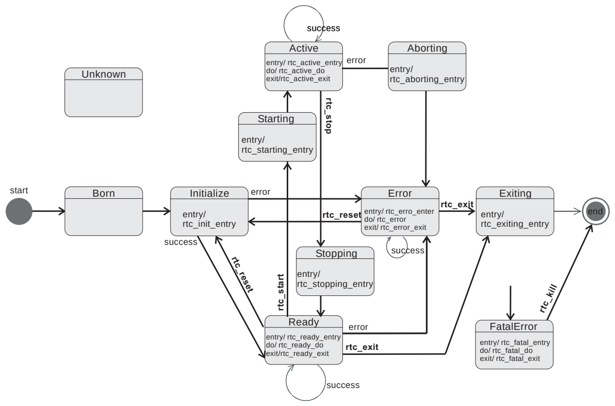


Fig. 4 RT-Component statechart diagram.

The states and state transition of the component was defined so that various type of RT-Components could be treated as common software parts. By giving a common state transition to RT-Components, and specifying the meaning of states, it is possible to control the action of many components similarly. A component developer has only to map his/her algorithm's process or library's process into each RT-Components state, and just insert his/her code to the RT-Component framework.

3.2 InPort/OutPort

In the low level real-time control layer, if a component is considered as the functional unit which consists of inputs, processing, and outputs like a control block diagram, it will be easy to perform a system configuration.

This input/output model is not so suitable for general usage of the distributed object model. Because an object which sends its data to other objects has to know all objects' complete interface definition. On the other hand, in such low level control layer, data type, number of data and unit of data are more important than interface definition. RT-Component adopted the publisher/subscriber model and defines it as InPort/OutPort.

3.2.1 InPort object

An input port of RT-Component. An InPort receives data from an OutPort that calls method of "InPort::put()". This is basic function of the InPort.

Other functional InPorts, that raise a signal or invoke a callback method etc., can be implemented as subclasses of the InPort.

3.2.2 OutPort object

An output port of RT-Component. An OutPort sends data to InPorts that "subscribes" this OutPort, calling "InPort::put()" as "push" type data exchange. "pull" type data exchange calling "OutPort::get()" method is also provided.

OutPort supports some subscription types, "New", "Once", "Periodic", "Periodic New", "New Periodic", "Triggered", "Triggered Priodic", "Periodic Triggered".

For example, the "New" subscription type means that an OutPort sends data to InPorts which subscribe it when a new data come from the activity. Other subscription type can also be defined if user needs.

3.3 RTComponent Service

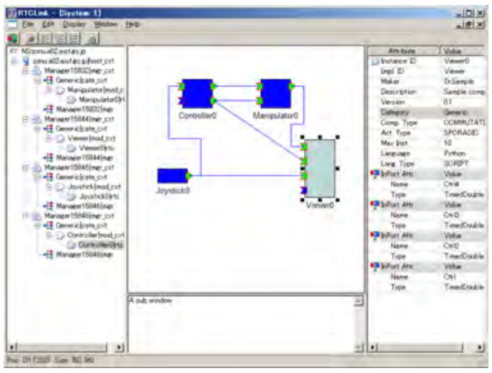
The software component should have enough interfaces to change internal attributes from outside without re-compileing. The object which provides the function attached to the RT-Component is called "RT-Component Service (RTCS)" in RT-Middleware. Component developer can provide his functional object interfaces to other RT-Components by using RT-Component Service feature.

4. SYSTEM INTEGRATION USING RT-COMPONENTS

Since the software component is based on some isolation principles such as cleanly specified interfaces, interoperability and glue mechanism, it is one of the great merits of component based software development that user can integrate application software using GUI tools.

"rtc-link" is a GUI tool that manages a connection of InPort/OutPort between RT-Components like a control-block diagram and performs activation/deactivation of an RT-Component (Figure 5). "rtc-link" is the powerful tool which can be used in case development of RT-Component and debugging are performed. Moreover, it can use also in verification and an experiment of a robot system, performing the low level integration of components.

The Connection information among RT-Components can be saved as XML file. RT-Components connection can be also rebuilt from a saved XML file.



Attribute	Value
Instance ID	Viewer0
Impl ID	Viewer
Maker	DrSample
Description	Sample comp
Version	0.1
Category	Generic
Comp Type	COMMUTATI
Act Type	SPORADIC
Max Inst	10
Language	Python
Lang Type	SCRIPT
InPort Attr	Value
Name	Ctrl4
Type	TimedDouble
InPort Attr	Value
Name	Ctrl3
Type	TimedDouble
InPort Attr	Value
Name	Ctrl2
Type	TimedDouble
InPort Attr	Value
Name	Ctrl1
Type	TimedDouble

Fig. 5 GUI tool: "rtc-link". The rtc-link is a graphical user interface for RT-Components manipulation.

4.1 OpenRTM-aist

"OpenRTM-aist" is prototype implementation based on RT-Middleware interface specification and RT-Component object model. "OpenRTM-aist" consists of an RT-Component development frame work, a manager and some set of tools.

4.1.1 RT-Component frame work

RT-Component frame work provides a managed state transition and InPort/OutPort management as frozen spots. A component developer inherits the base class of the RT-Component, and can create his new component class. The state transition logic is implemented in the base class, and a developer only needs to map process to be executed in each state by the specific method override.

4.1.2 RT-Component manager

RT-Component manager manages a life cycle of an RT-Component and provide access to CORBA naming service. RT-Component can exist as a loadable module. The loadable module is loaded by a component manager and an instance of an RT-Component is created by the manager. A manager also activates a component as a CORBA object and binds it to a name server. Clients can obtain RT-Component object references from the name server.

4.1.3 RT-Component template generator

OpenRTM-aist provides a template source code generator for RT-Component development. The template generator creates C++ and Python source code from given component profiles and InPort/OutPort profiles. (Here, "template" is not C++ template metaprogramming.) C++ source code includes a header file, component source, executable component source code, Makefile and specification file of the RT-Component. A standalone RT-Component executable and a loadable RT-Component module are created from these files.

5. STANDARDIZATION OF RTC

In this section, standardization of RTC (RT-Component) in Object Management Group (OMG) will be described. The standardization in OMG is char-

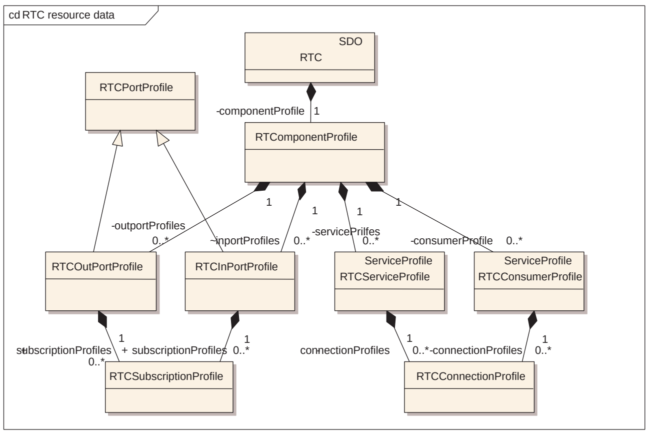


Fig. 6 The RTComponent's PIM (Resource data model).

acterized about "Model Driven Architecture", "Platform Independent Model (PIM)" and "Platform specific Model PSM)." RTC's PIM will be shown and described here.

5.1 OMG

Object Management Group (OMG) is a one of the biggest software standardization organization that is founded in 1989. CORBA (Common Object Request Broader Architecture), which is distributed object middleware, and UML (Unified Modeling Language), which is a modeling language, used for software modeling and design language standard, are the typical standardized specification which OMG led and specified. In addition, many standards for various domain are discussed and specified according to the open process as a neutral nonprofit organization independent of a specific software company.

5.2 MDA and PIM, PSM

MDA (Model Driven Architecture) is the reference architecture for realizing development of the software system led by modeling, and management of a software life cycle which OMG advocates. It consists of two classes of modeling layers, which is PIM (Platform Independent Model) and PSM (Platform Specific Model).

PIM which is described with UML (Unified Modeling Language) is the model of the system independent of a platform, which is not dependent on a specific language, OS, middleware, etc.

PSM, which is automatically or semi-automatically created from PIM, is a model that depends on specific language or OS, and implementation is performed according to this PSM.

5.3 Super Distributed Object (SDO)

RTC (RT-Component) will be standardized based on SDO (Super Distributed Object), which is already standardized in OMG. SDO is a advanced distributed object model which has unified description for distributed resources in the network. SDO supports interoperability, interconnectivity and dynamic integration of distributed devices. RTC is redesigned and RTC's object model inherits SDO's object model to standardize in OMG.

5.4 RT-Component PIM

RTC's PIM consists of the following two parts.

Resource Data Model The data model of profile information description that RTC has.

Interfaces Interface model to access to the resource data model.

Figure 6 shows the RTC resource data model. Figure 7 shows RTC class diagram that describes the interfaces.

5.4.1 Resource Data Model

As shown in Figure 6, RTC's profile information inherits SDO's profile information. This is Robot Domain specific profile information for SDO. RTCOmponent profile has the Activity's status, InPort profile, OutPort profile, RTCS profile and Consumer's profile. Moreover, Port's connection has profile as connection profile.

5.4.2 Interfaces

Figure 7 shows the interface definition for access to the resource data and other operations. The RTComponent class is a main body of RT-Component, and all resource data and other object references are obtained from this object. RTCActivityAdmin interface and RTCActivity interface are service interfaces that provide operations to the Activity of RTC.

RTCInPortAny and RTCOutPortAny are example mapping to Any type data for InPort and OutPort. InPort and OutPort for each data type are implemented inheriting RTCOutBase.

RTCService is base class of RTC's service interface that is provided from RTC. RT-Component developer can implement his/her own service inheriting it. RTCConsumer is a interface to use other RTC's service, and at starting the RTC, dependent RTCService object references are given from outside and business logic inside RTC uses the service reference.

The Activity's state machine diagrams, sequence diagrams that describe interaction between each object are also defined in PIM. Because of the reason of the space, the details are not described here.

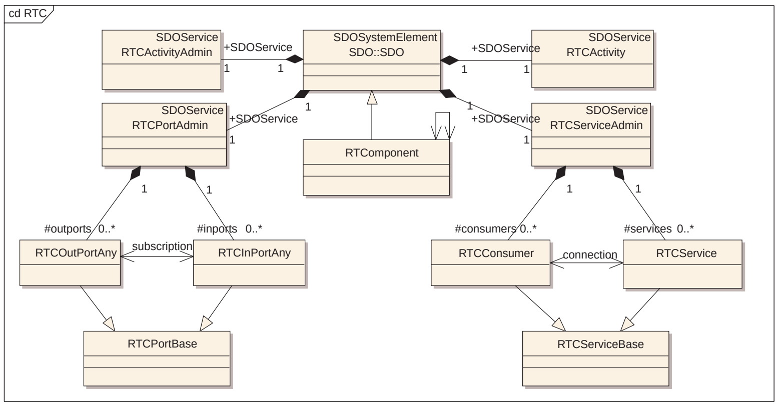


Fig. 7 The RTComponent's PIM (Class diagram).

6. CONCLUSION

In this paper, we proposed component based robotic system integration scheme RT-Middleware and RT-Component. The functions required for the RT-Middleware which supports robot specific features were discussed and clarifies. To realize component based robotic system development efficiently, RT-Component and its architecture was proposed.

Since RT-Component can be a software platform and its standardization is important, we are standardizing RT-Component specification in OMG. RT-Component object model was redesigned inheriting SDO that already standardized in OMG.

In the future work, newly proposed object model's validation will be performed by using newly implemented software, and we will go forward the standardization of the RTC specification in OMG.

REFERENCES

[1] Noriaki ANDO, Takashi SUEHIRO, Kosei KITAGAKI, Tetsuo KOTOKU, Woo-Keun 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

[2] Noriaki ANDO, Takashi SUEHIRO, Kosei KITAGAKI, Tetsuo KOTOKU, Woo-Keun YOON, "RT-Component Object Model in RT-Middleware - Distributed Component Middleware for RT (Robot Technology) -", 2005 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA2005), p.We-B2-5, 2005.06

[3] Noriaki Ando, Takashi Suehiro, Kousei Kitagaki, Tetsuo Kotoku and Woo-Keun Yoon, "Composite Component Framework for RT-Middleware (Robot Technology Middleware)", IEEE/ASME International Conference on Advanced Intelligent Mechatronics(AIM), 2005

[4] M.Mizukawa, H.Matsuka, T.Koyama, T.Inukai, A.Noda, H.Tezuka, Y.Noguchi, N.Otera, "ORiN Open robot Interface for the Network – The Standard Network Interface for Industrial robots and its Applications –", ISR2002, No.45

[5] Makoto Mizukawa, Hideo Matsuka' Toshihiko Koyama, Toshihiro Inukai, Akio Noda, Hirohisa Tezuka, Yasuhiko Noguchi, Nobuyuki Otera, "ORiN: Open Robot Interface for the Network – The Standard and Unified Network Interface for Industrial Robot Applications –", SICE Annual Conference 2002, pp.1160-1163, Osaka

[6] Orocos: Open Robot Control Software. http://www.orocos.org

[7] C. Schlegel, R. Worz, "The Software Framework SmartSoft for Implementing Sensorimotor Systems", IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS '99, pp.1610-1616, Kyongju, Korea, October '99

[8] http://orca-robotics.sourceforge.net/about.html

[9] Fumio OZAKI, "Open Robot Controller Architecture (ORCA)", IROS2004 Workshop on Robot Middleware toward Standards,

[10] Fumio OZAKI, "Open robot controller archtecture (ORCA)", AIM2003 Workshop: Middleware Technology for Open Robot Architecture

[11] Kohtaro SABE, "Open-R : An Open Architecture for Robot Entertainment" , AIM2003 Workshop: Middleware Technology for Open Robot Architecture

[12] Kapoor, C. Tesar, D., "A Reusable Operational Software Architecture for Advanced Robotics", Proceedings of the Twelfth CSIM-IFToMM Symposium on theory and Practice of Robots and Manipulators, Paris, France, July 1998.

[13] Minsu Jang, Jaehong Kim, Meekyoung Lee and Joo-Chan Sohn, "Ubiquitous Robot Simulation Framework and Its Applications", 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2005), pp.3213-3218, 2005.08

[14] Brian P. Gerkey, Richard T. Vaughan, Kasper Stoy, Andrew Howard, Gaurav S. Sukhatme, and Maja J Mataric, "Most Valuable Player: A Robot Device Server for Distributed Control", Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001), pp.1226-1231, 2001.11.