Proposal of Distributed Architecture Based on JAUS and RT-Middleware

INAMURA Wataru : Control & Communication Technology Department, Products Development Center, Corporate Research & Development
FUJII Masakazu : Control & Communication Technology Department, Products Development Center, Corporate Research & Development
MURAKAMI Hiroki : Senior Researcher, Control & Communication Technology Department, Products Development Center, Corporate Research & Development
TOMIZAWA Masaaki : Manager, Control & Communication Technology Department, Products Development Center, Corporate Research & Development

For development of a robot system, a modular approach by standardized technologies is expected to improve the compatibility and reusability of robotic functions. It is expected to contribute to industry progress because robot systems will be developed more rapidly and flexibly by integrating various modules in the near future. This paper proposes a distributed architecture for robot systems using standardized technologies. We constructed the distributed architecture based on both JAUS (Joint Architecture for Unmanned Systems) as a specification for the application layer and RTM (Robot Technology Middleware) as a development framework. The architecture was applied to mobile robots. The experimental results show the effectiveness and reliability of the proposed architecture.

1. Introduction

RT (Robot Technology) is a general term for "any intelligent networked system using robotic technology and which can perform real world tasks." [(1)] RT is expected to support human lives and activities and contribute to the creation of an affluent, safe, and secure society in the near future. RT systems that can meet diverse needs of society are required. To develop RT systems capable of meeting these diverse needs, we need development techniques that enable us to reduce system development, maintenance, and expansion costs. RT systems were conventionally developed by designing different software for individual RT systems due to computer capacity limitations and so on. We can now integrate RT systems by assembling modular software components, because remarkable advances in computer and communication system performance have been achieved in recent years. It is expected that new methods and procedures will be established in the not too distant future. RT systems with diverse functions will be developed in a short period of time by combining these modular software components or new functions and functional enhancements will be added by developing only the software components required and replacing existing components with them. Given this, various activities are being conducted to standardize the RT software infrastructure; that is, to establish common ground on which to construct RT systems.

Although everyone agrees that module-based development technology is effective, it is far from being widely used. This is because each R&D group has software assets that it has accumulated over the years through RT system development activities, and the efficiency of system development will temporarily drop if new technology is introduced. In addition, usable software components are hard to come by under the present circumstances. If module-based development technology is to be introduced then, software components must initially be developed one by one in accordance with a standardized modularization procedure since usable software components are in short supply. If we suppose, though, that the required software components have been made easily obtainable, development efficiency will increase and the number of users will also increase by prompting the development of new software components and further strengthening the lineup of software components. To start this beneficial cycle, it is important that the software infrastructure for RT system development conforms to internationally standardized specifications. Effective ways of achieving the use of standardized technology are to use the evolving standard technology in practice, demonstrate the effectiveness of the technology, publish the results of applying the technology and the results of evaluations of systems built with the technology, point

out problems that need to be resolved, make continual improvements, and accumulate know-how.

In this paper, we propose distributed architecture to be constructed by using software components for RT systems. [2], [3] The proposed distributed architecture was formulated by mutually complementing two different specifications that are still in the process of standardization. In developing a mobile robot system based on the proposed distributed architecture, we added some functional specifications that were not defined. The experimental results described in the following chapters show the effectiveness and reliability of the proposed architecture.

2. Standard software technology

2.1 Outline of RT-Middleware

RT-Middleware (hereinafter called RTM) is a standard technology that "aims at establishing a common platform based on the distributed object technology and which would support the construction of various networked robotic systems by the integration of various network enabled robotic elements called RT-Components." [1] The 21st Century Robot Challenge Program was directed by NEDO (New Energy and Industrial Technology Development Organization). As part of this program, RT-Middleware project was started, and a prototype implementation of RTM was developed. A process of international standardization of RTM is underway at the Robotics-DTF (Domain Task Force) [4] of the OMG (Object Management Group), which is an industry organization for promoting software standardization. In Japan, a committee has also been established in the Japan Robot Association. The authors are members of this committee and are contributing to this standardization activity.

RTM is a standard specification to ensure interconnectivity between robotic software components. State transition was defined to determine the basic workings of RT-Components. Furthermore, communication channels were defined to realize easily between RT-Components.

OpenRTM-aist is a prototype implementation of RTM developed by AIST(National Institute of Advanced Industrial Science and Technology). OpenRTM-aist is designed based on OS-independent distributed object technology called CORBA (Common Object Request Broker Architecture). Since RT-Components and ports that serve as communication channels are implemented as CORBA objects, communication between RT-Components can be easily established. In addition, basic types of data to be sent or received through ports were defined: integer-type data and floating-point-number-type data, and so on. Data arrays were also defined. In addition, unique data types other than these types can also be defined, as required by an application.

As described above, RTM makes it possible to develop versatilely, application-independent RT-Components. [5], [6] But, if there is no consistency in RT-Components, it is difficult to integrate an RT system with the RT-Components. We need standardized specifications that gives RT-Components consistency at the level of application to integrate RT systems by effectively using that components.

2.2 Outline of JAUS

JAUS (Joint Architecture for Unmanned Systems) [7] is a standard specification for unmanned systems that the United States Department of Defense promotes in order to reduce the system life cycle cost, curtail the development period, make it easy to introduce technology, and realize expansion of new capabilities. An unmanned system is typically a mobile robot such as a UGV (Unmanned Ground Vehicle), UAV (Unmanned Air Vehicle), or UUV (Unmanned Underwater Vehicle). Comprised of numerous research organizations, manufacturers, and the like, the JAUS Working Group is working to establish JAUS specifications. The principle of JAUS is that unmanned systems are independent of computer hardware, software implementation technology, and operations executed by a system.

Elements of modularization as indicated by JAUS are called components. A system is integrated by combining components, and these components work in coordination with each other while exchanging messages. Components are defined by assigning each function for unmanned systems. Although JAUS specifies the many useful components and messages for mobile robots, there are some parts that remain undefined since JAUS is still in the process of standardization.

Because JAUS does not specify the implementation technology, we must employ some kind of implementation technology to integrate RT systems.

3. Proposal for the distributed architecture

3.1 Basic configuration of the distributed architecture

As has already been explained, RTM is a specification that supports the modularization of an RT system which is not dependent on application-level specifications. OpenRTM-aist is a prototype in which the RTM implementation technology is included. On the contrary, JAUS is an application-level specification for the domain of mobile robots that is not dependent on hardware or implementation technology.

We propose that the basic configuration of the distributed architecture for RT systems should be constructed by these mutually complementing related standard technologies. Specifically, functions of software components to be implemented by OpenRTM-aist should be defined based on JAUS specifications, and message communication defined in JAUS specifications should be implemented by using the ports of the RT-Components. So we will be able to make

RT-Components work in coordination with each other and integrate them into an RT system.

We can used some components and messages already defined in JAUS. If necessary, we defined some new components and messages as complying with the JAUS design concept to integrate RT systems.

3.2 Basic component design

To implement the JAUS functions in RTM, the RT-Components must function as basic components of JAUS. JAUS defines a state transition as common basic operations to be performed by components, while RTM also defines a state transition as operations to be performed by RT-Components. This section describes the designs for the state transition and message communication that were adopted for the distributed architecture under discussion in this paper.

(1) State transition

A state transition is defined in the RT-Component specifications in more detail than in the JAUS specifications, because consideration is given to implementation in the RT-Component specifications. **Figure 1** shows the results of the mapping of state transition carried out with care taken to maintain consistency with the state transition defined by the JAUS specifications. Since state transitions can be mapped as defined by the JAUS specifications, it was verified that RT-Components can perform the state-transition functions of JAUS components.

Furthermore, this state transition allow basic operations of individual components to be unified, making it possible to integrate individually developed RT-Components into a system.

(2) Message communication

For OpenRTM-aist, the ports working as channels of communication between RT-Components are implemented as CORBA objects. Therefore, structures defined in accordance with CORBA specifications can be handled as communication data types. A port is provided for each data type, and communication can be established only between input and output ports that are designated to handle the same data type.

We defined data types as structures for JAUS messages, so the messages can be communicated via ports. In the JAUS component specifications, a group of core messages (command-type messages used to operate components) to be commonly handled by all components, as well as messages specific to the functions of each component, are defined. With this in mind, we defined a group of core messages as one union, which is a group of structures, because a component with a little ports was more simple and useful. In this distributed architecture, RTM ports of each component should be designed to handle both a group of core messages for JAUS and component-specific messages.

3.3 Design of the RT system for mobile robots

We designed one of the RT systems for mobile robots equipped with force-controllable manipulators based on the distributed architecture proposed in this paper. **Figure 2** shows the configuration of this RT system. Of the various components defined in the JAUS specifications, the components necessary to integrate a mobile robot were selected and used. Since a Force Sensor component is not yet defined in the JAUS specifications, we designed it by assigning functions based on reference data on other sensor component products. We also designed the messages to be output by this Force Sensor component by referring to messages

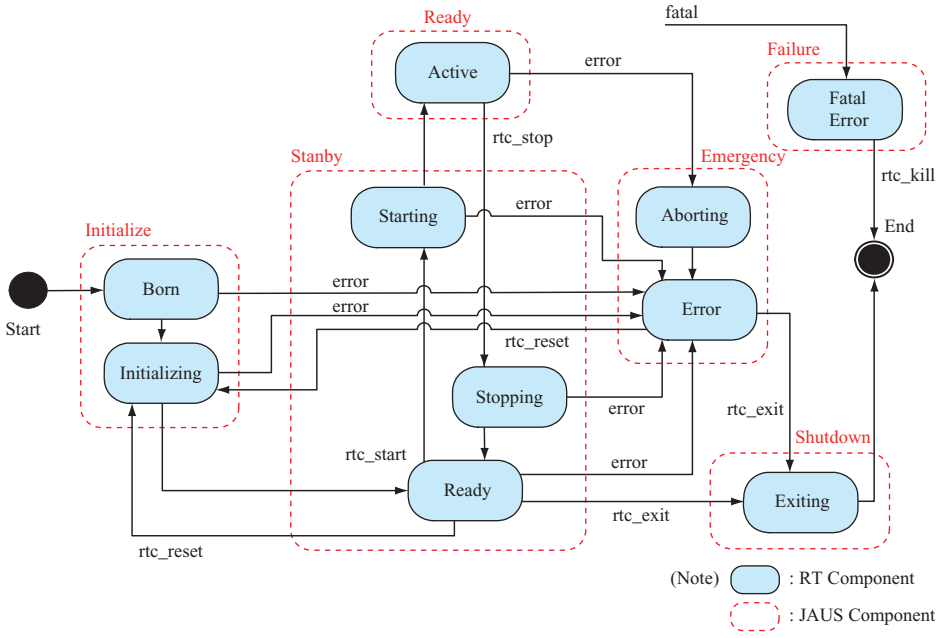

Fig. 1 State chart of components

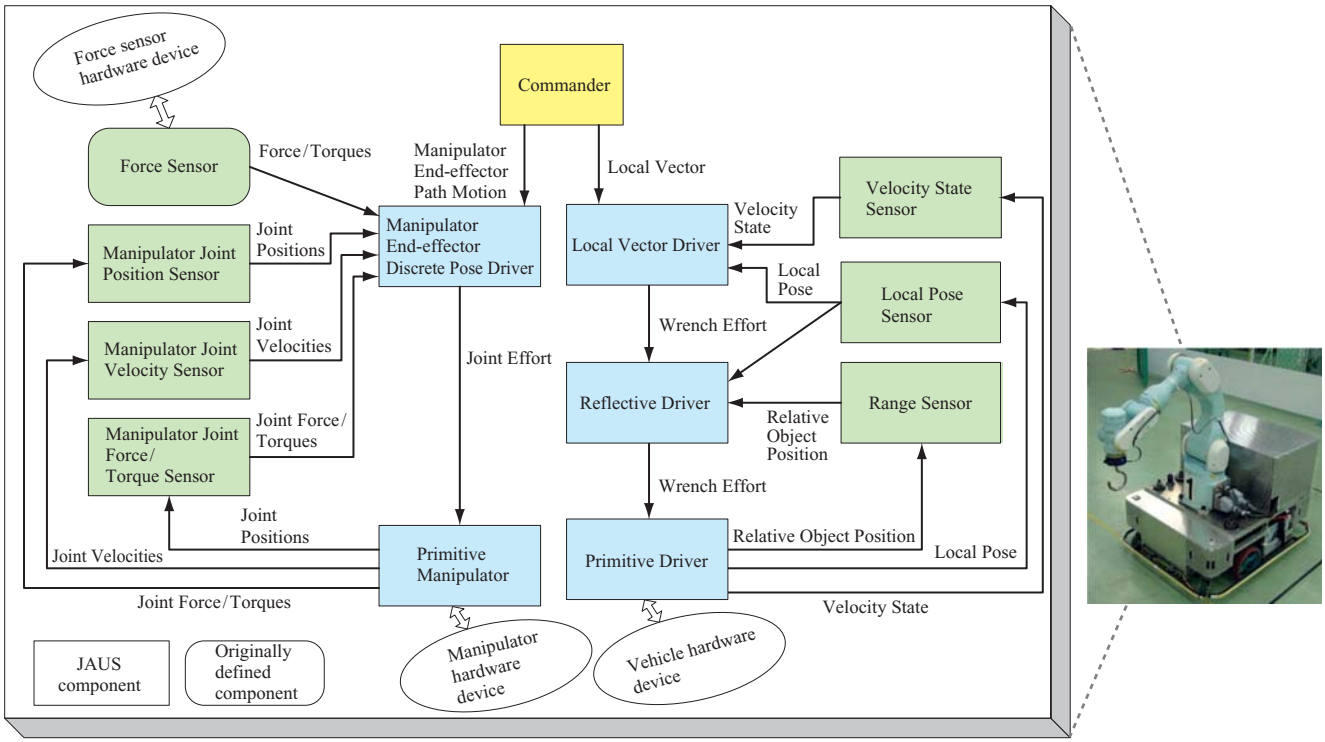

Fig. 2 System structure of mobile robot

regarding existing sensors. The newly designed component and messages can be handled in the same way as existing JAUS components and messages are.

Each component performs the functions described below. The Commander component takes general control of the robot. It provides the Manipulator End-effector Discrete Pose Driver or Local Vector Driver component with a target trajectory. The Manipulator End-effector Discrete Pose Driver and Local Vector Driver calculate the control commands for tracking a target trajectory every time based on feedback information provided by sensor components and output them. The Primitive Manipulator and Primitive Driver convert a received control command to a hardware-specific format and output it in that format. The Reflective Driver modifies the command value to avoid obstacles and stabilize attitude and forwards the modified command value.

In the RT system discussed in this paper, sensor components other than the Force Sensor component perform only the function of relaying messages. Their presence, therefore, is not a prerequisite for operation of the RT system. However, if the RT system is designed this way, functional expansion becomes easier. For example, in a situation where system functions are to be enhanced by using higher-performance sensors, they can be installed without any other part in the RT system having to be modified if they have the same ports as those of existing sensor components.

In this distributed architecture, each component is assigned a single and simple function. RT systems are able to deliver these specified performance, as these components, each performing a given single function, operate in coordination with each other. Based on this distributed architecture, it is possible to integrate RT systems that range from simple RT systems to complex, high-performance RT systems. Furthermore, this distributed architecture allows for easy RT system modification and/or expansion.

4. Experiment and results

To verify that system functions can be modified easily, an experiment was conducted with two mobile robots. Specifically, the two robots were operated to perform cooperative transportation task. [8] The two robots

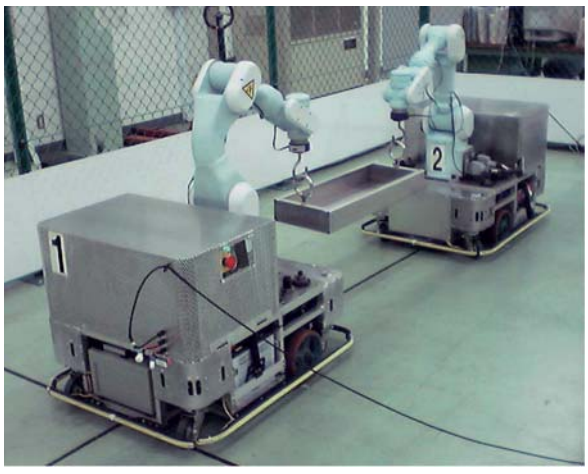

Fig. 3 Experiments of cooperative transportation

hold one load together, as shown in **Fig. 3**. The robot in front is called leader and the robot at the back called follower. The cooperative transportation task is carried out as the follower follows the leader. We replaced some components of mobile robot described in **Section 3.3** with new components to perform the task.

To carry out this cooperative transportation task, we developed a new component called a Transportation Driver. This component estimates a target trajectory for the leader from data on the deviation of the end-effector of the manipulator, and then calculates its own travel command. This component was designed to have the same input and output ports as those of the Local Vector Driver, and an input port was added to receive data on the deviation of the end-effector of the manipulator. In addition, a port for outputting data on the deviation of the end-effector was added to the Manipulator End-effector Discrete Pose Driver.

A conventional Local Vector Driver is a component that tracks a target trajectory for travel given as vector data. The mobile robots were able to perform cooperative transportation task when the Local Vector Driver in the follower was replaced with the previously mentioned Transportation Driver. It was verified from the results of this experiment that robot functions can be modified and expanded.

As described above, it was verified that in this distributed architecture, a new function can be added to an RT system by modifying only a target component or replacing an existing component with a component that has a desired function without modification of any other part in the RT system. If a new control algorithm must be verified or a new experiment must be conducted for any reason, an existing RT system can be used by updating only part of the components used in the system.

5. Conclusion

In this paper, we described the methodology for developing components in accordance with standardized specifications and combining developed components to integrate RT systems. We proposed a distributed architecture that can be constructed based on two standard technologies : (1) JAUS application specifications and (2) OpenRTM-aist, which is the RTM prototype implementation technology. We applied the proposed architecture to two mobile robots and demonstrated that the functions of a module-based RT system can be modified or expanded easily and flexibly.

We now stand at the dawn of an age in which RT will become widespread as a technology for supporting society. As RT standardization is already underway, we believe that the actual use of RT is significant and that it sets a precedent for how RT can be put to practical use. We would be delighted if this paper proves to be of some help to the development of the RT industry.

REFERENCES

(1) RT-Middleware Project Home Page : http://www.is.aist.go.jp/rt/ (cf. 2007.4.10)

(2) W. Inamura, M. Fujii, H. Banno and K. Tanaka : Proposal of Distributed Architecture based on JAUS and RT Middleware, Proceedings of the 6th SICE System Integration Division Annual Conference (SI2005) (2005.12) pp. 967-968

(3) W. Inamura, M. Fujii, H. Banno and K. Tanaka : Proposal of Distributed Architecture based on JAUS and RT Middleware –Application of Cooperative Control System by multiple mobile robots–, Proceedings of the 11th Robotics Symposia (2006.3) pp. 538-543

(4) OMG Robotics DTF Home Page : http://robotics.omg.org/ (cf. 2007.5.10)

(5) N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku and W-K Yoon : System Development Method using RT Components –R&D of RT Middleware Fundamental Functions Part 14–, Proceedings of 2005 JSME Conference of Robotics and Mechatronics (2005.6) 2A1-N-072(1)-(4)

(6) H. Kitano, T. Kuwata, T. Nakahara, H. Araki, T. Ishikawa and M. Komiyama : Home helper robotic system using PT-Middleware, Proceedings of 2005 JSME Conference of Robotics and Mechatronics (2005.6) 2P1-N-068 (1)-(4)

(7) JAUS Working Group Home Page : http://www.jauswg.org/ (cf. 2007.5.10)

(8) M. Fujii, W. Inamura, H. Murakami, K. Tanaka and K. Kosuge : Cooperative Control of Multiple Mobile Robots Transporting a Single Object with Loose Handling, Proceedings of the 2007 IEEE International Conference on Robotics and Biomimetics (2007.12) pp. 816-822