https://doi.org/10.20965/jrm.2010.p0322

Paper:

Integration of Manipulation, Locomotion, and Communication Intelligent RT Software Components for Mobile Manipulator System Using Scenario Tools in OpenRT Platform

Natsuki Yamanobe*, Ee Sian Neo*, Eiichi Yoshida*, Nobuyuki Kita*, Kazuyuki Nagata*, Kazuhito Yokoi*, and Yosuke Takano**

*Intelligent Systems Research Institute, National Institute of Advanced Industrial Science and Technology (AIST)
1-1-1 Tsukuba Central 2, Umezono, Tsukuba, Ibaraki 305-8568, Japan
Email: n-yamanobe@aist.go.jp
**Service Platforms Research Labs, NEC Corp.
[Received September 30, 2009; accepted February 16, 2010]

The OpenRT Platform, an integrated development environment for component-based robot system development, is being constructed in order to enhance intelligent robot research and development efficiency. In this paper, a mobile manipulator system that can bring human indicated objects like a service dog is developed based on the OpenRT Platform. The system works with several components providing manipulation, locomotion, and communication functions developed as examples modularizing intelligent robotic functions. These components are integrated using scenario tools in the OpenRT Platform for achieving target tasks.

Keywords: OpenRT Platform, intelligent RT software component, system integration, application scenario

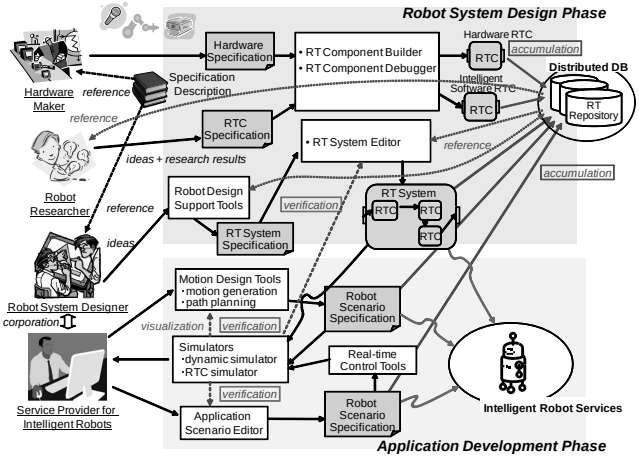

Fig. 1. RT-Component based robot development flow.

1. Introduction

Robots that can achieve various tasks in complex and dynamic environments are required for supporting daily life. In order to develop such a next-generation robot, robotic intelligence, especially the responsiveness to the change of tasks and environments and the reliability of operations, needs to be built efficiently.

So far, various robot hardware and software have been developed focusing on each application. It is expected to utilize existing knowledge for constructing a new robot system. Modularizing robot functions facilitates accumulating knowledge and integrating it in new robot systems. In order to enhance the efficiency of research and development in intelligent robotics, a common software platform is needed for implementing intelligent robot modules efficiently and constructing robot systems readily based on those modularized functions.

The Intelligent RT Software Project, supported by New Energy and Industrial Technology Development Organization (NEDO) and implemented in 2007, targets a common software platform for constructing robot systems and promoting the development and accumulation of modularized intelligent robotic functions [1]. The OpenRT Platform developed in this project as an integrated development environment is based on RT-Middleware [2], a middleware and software platform for component-oriented robot construction, developed at National Institute of Advanced Industrial Science and Technology (AIST). Basic functional units of an RT-Middleware based system called RT-Components have specifications defined and standardized by the Object Management Group (OMG) [3]. Thus, robot systems can be easily constructed just by connecting developed RT-Components. As shown in **Fig. 1**, there are many work processes for developing a new robot system. The OpenRT Platform comprehensively covers all the processes to realize seamless system development from robot design to application.

In this paper, we build a mobile manipulator system mimicking service dog functions. Several intelligent RT software components, which can provide manipulation, locomotion, and communication functions, are developed as examples of modularization of intelligent robotic functions. These components are then integrated by using scenario tools in the OpenRT Platform so as to realize target

tasks.

This paper is laid out as follows: Section 2 explains the OpenRT Platform, focusing on the scenario tools used for application development. Section 3 presents intelligent software components for constructing a mobile manipulator system. Section 4 discusses the mobile manipulator system, application scenario, component connection relationship, and experimental results. Section 5 evaluates robot system construction processes using the OpenRT Platform, and Section 6 presents conclusions.

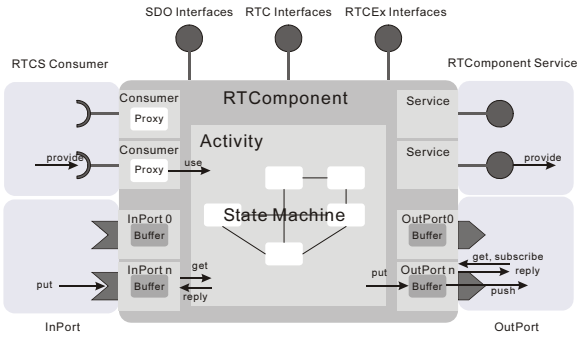

Fig. 2. RT-Component architecture [2].

2. OpenRT Platform

The OpenRT Platform is an integrated development environment for next-generation robot systems consisting of various modularized software and hardware components. For developing a new robot system, there are many work processes classified into the robot design and application development phases shown in **Fig. 1**. The platform provides various tools, which cover all the work processes, to improve each process and achieve seamless and efficient system development.

2.1. Related Work

Research on software platforms for robot system construction have been performed actively in recent years. Player/Stage/Gazebo [4] is a client-server software platform for robot and sensor systems, developed mainly at the University of Southern California. Player is a robot server providing a network interface to a variety of robot, actuator, and sensor hardware. Stage, Player's simulation backends, simulates a population of mobile robots in a two-dimensional (2D) environment, whereas Gazebo does so in a 3D environment. OROCOS [5], a free software project in EU for developing a general-purpose modular framework for robot and machine control, supports a real-time software framework, a set of generic libraries of robots and machine tools, and a component-based application platform. ORCA/ORCA2 [6], an open-source framework for developing component-based robot systems and evolved from OROCOS, provides the means and tools for defining and developing software components to form systems ranging from simple robots to distributed sensor networks. SmartSoft [7], maintained at Hochschule Ulm, develops communication patterns and dynamic wiring of components on top of OROCOS and provides a toolchain for a model-driven software development in robotics. CLARAty [8], a framework for autonomous rovers developed at JPL, NASA Ames Research Center, CMU, and the University of Minnesota to promote the reuse of robotic software, provides interfaces for common robotic functionality, robot functions implemented based on the CLARAty architecture, and an integration framework. ORiN [9], a middleware providing standard communication interfaces over various factory automation apparatuses including robots, has specifications released by the ORiN consortium, with which most Japanese industrial-robot makers have affiliated. The Microsoft Robotics Developer Studio [10], a Windows-based software platform for creating robot applications, includes an asynchronous services-oriented runtime and a set of visual authoring and simulation tools. Willow Garage's ROS [11], an open-source software platform, consists of an original distributed framework and tools and libraries for intelligent robotics research and development including results of other robotics open-source projects.

The purpose of the OpenRT Platform is to provide a software framework enabling component-based robot system development based on standardized interface specifications and a complete toolchain increasing system development efficiency. The standardized interface specifications of software components is free and open, enabling vendors to implement specification-based software platforms and develop components using an appropriate platform and ensuring platform interoperability. This enhances the effectiveness of the developed components and robot systems, but other software platform projects have not considered standardization.

2.2. RT-Middleware and RT-Components

RT-Middleware underpinning the OpenRT Platform is a middleware and software platform for constructing robot systems by combining modularized robot functions. An RT-Component is a basic functional unit of an RT-Middleware based system, whose specifications are standardized in OMG [3]. In order for an RT-Component to be operating system, program language, and network independent, RT-Middleware is constructed based on a distributed object middleware such as Common Object Request Broker Architecture (CORBA) [12].

Figure 2 diagrams RT-Component architecture consisting of the following processes and interfaces:

- Activity – core logic execution entity implemented in the component.
- Component Profile – component information provided for dynamic component connection and disconnection.

- Data Port – used for data-centric interaction and connected to ports with same data-type. InPort is defined as data input port, and OutPort as data output port.
- Service Port – used for request/response interaction. Component developers define port functions. The Service Provider Port provides services to other components, and the Service Consumer Port consumes services of other components.
- Configuration Interface – manages internal component parameters.

If a developer determines specifications of an RT-Component, such as the component name, port type, port data type, and configuration parameters, template files for the component are automatically provided to generate the component by simply inserting program code into the component framework.

2.3. OpenRT Platform Tools

To improve system construction efficiency, almost all tools are implemented as Eclipse plug-ins for seamless system development.

Specification Description

To realize seamless, efficient system development and improve component and robot system reusability, the following specification descriptions are defined and used in common on the platform.

- Hardware specification description,
- Component specification description,
- Robot system specification description,
- Robot application scenario description.

RT-Middleware

The OpenRT Platform uses RT-Middleware as a robot system development and execution environment. To broaden the platform environment, several types of RT-Middleware are developed for different operating systems and programing languages.

RT-Component Development Support Tools

- The RT-Component Builder generates RT-Component template files based on component specifications written by the specification description and provides a development environment for RT-Components.
- The RT-Component Debugger verifies developed components, data flow in ports, internal component parameters, etc.

RT System Construction Support Tools

- The RT System Editor, a Graphic User Interface (GUI) for constructing a robot system based on RT-Components, manages InPort/OutPort or Service Port connection as a control-block diagram, defines configuration parameters, and activates and deactivates components.

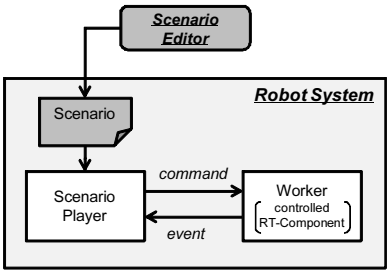

Fig. 3. OpenRT Platform scenario tools.

- The RT Repository operates as a distributed database to save developed RT-Components and hardware, component, and robot system specification definition files. The system integrator downloads these for use on the platform for new robot systems.
- The 3D dynamic simulator and RT-Component simulator verify developed robot systems, movements, and application scenarios, developed as OpenHRP3 [13].

Application Software Development Support Tools

- Robot Motion Design Tools plan movements and trajectories for robot applications.
- Robot Application Scenario Tools provide functions of editing application scenarios and controlling RT-Components based on the scenario. The details of the tools are shown in the next sub-section.

2.4. Scenario Tools

To achieve complex robot application tasks, a central control function is necessary so as to recognize task conditions and control the system according to the condition. On the OpenRT Platform, a central control function is implemented using the scenario tools developed based on RoboStudio [14], which is a scenario platform provided by NEC, with a robot scenario script language developed based on XML and its interpreter.

Figure 3 shows scenario tool functions. A scenario file generated in advance by the scenario editor is installed on the scenario player and the script is executed. The scenario player controls RT-Components by sending "command" messages and obtaining information from RT-Components in "event" messages. Using the scenario enables new tasks without changing or recompiling components.

2.4.1. Scenario

A robot application scenario is written using scenario script language based on an event-driven state transition model. All possible task states are presented in the scenario. Robot actions in a state, internal and external events, and subsequent destination states are also described. According to the scenario, the scenario player

```
<?xml version="1.0" encoding="EUC-JP"?>
<!DOCTYPE interface SYSTEM "./worker_interface.dtd">
<interface module_name="RtcSrComponent">
  <command name="Start" response="true" type="int">
    <arg name="RuleName" type="string" default="Main"/>
  </command>
  <event name="RecogResult" duration="10000">
    <arg name="Word" type="string"/>
    <arg name="Score" type="int" default="100"/>
  </event>
</interface>
```

Fig. 4. Worker interface definition example. This worker sends one command and receives one event. Command "Start" has one argument and event "RecogResult" two.

sends commands to other RT-Components to execute corresponding actions, waits for events to recognize task conditions, and forwards state transitions.

2.4.2. Scenario Editor

A scenario is written on the scenario editor and compiled into an executable file for the scenario player. The scenario editor is implemented as an Eclipse plug-in just like other OpenRT Platform tools.

2.4.3. Scenario Player and Worker

A worker is an RT-Component controlled by the scenario player, e.g., as shown in **Fig. 3**. One scenario player manages multiple workers, which may be connected to scenario players and conventional RT-Components, thorough two data ports sending command and event messages. A worker interface definition file, shown in **Fig. 4**, determines commands and events sent and received by workers.

2.4.4. Scenario Framework

The scenario framework, an execution environment of scenario players and workers, is developed on RT-Middleware. The framework provides basic communication functions and simplifies user programs in communication between scenario players and workers. Command and event messages including command or event name, type, address, and arguments are automatically generated by worker interface definition files by the framework. Corresponding callback function is also automatically executed by the framework when a command or event is called, enabling the component developer to generate a worker simply by defining the worker interface and implementing callback functions.

3. Intelligent RT Software Components for Mobile Manipulator System

To verify OpenRT Platform effectiveness, we are developing several intelligent RT software components and the mobile manipulator system shown in **Fig. 5**. The components realize manipulation, locomotion, and communication functions required for a service robot supporting daily life. The mobile manipulator system integrates these functions to communicate vocally with users, move around in known environments, search visually for objects, and fetch objects from elsewhere.

We modularize such intelligent functions as one or more RT-Components and determine input/output interfaces between the functions to maximize their reusability. Specifications of those RT-Component are defined based on the RT-Component specification description, and the all components are developed using the RT-Component Builder in the OpenRT Platform. In the following subsections, the components developed for the mobile manipulator system are briefly presented.

3.1. Manipulation Intelligence

Environment and object recognition components

The components visually obtain environment and object information required for a robot to manipulate an object while avoiding obstacles, such as the position of a known-shape object and the depth map around the object.

Environment and object information management components

Environment and object information in the database, such as name, shape, color, and existing position, is managed based on data obtained by the recognition components. The database saves such data based on an ontology for information management, which is defined using Web Ontology Language (OWL).

Motion planning components

A collision-free trajectory connecting given initial and goal configurations is generated by sampling-based approaches such as PRM and RRT based on the environment and object information saved in the database.

Grasp motion planning components

A variety of grasping-related movements such as approach, grasping, and pickup are planned based on object and environment models abstracted using primitive shapes for rapid planning.

Manipulator control components

The components control real and simulated manipulators based on robot reference configurations fed from the motion planning components.

3.2. Locomotion Intelligence

Localization components

Current robot position and orientation are estimated based on odometer and vision sensor data using a Kalman filter.

Map building and maintenance components

A 2D grid map is built using the environment model saved in the database.

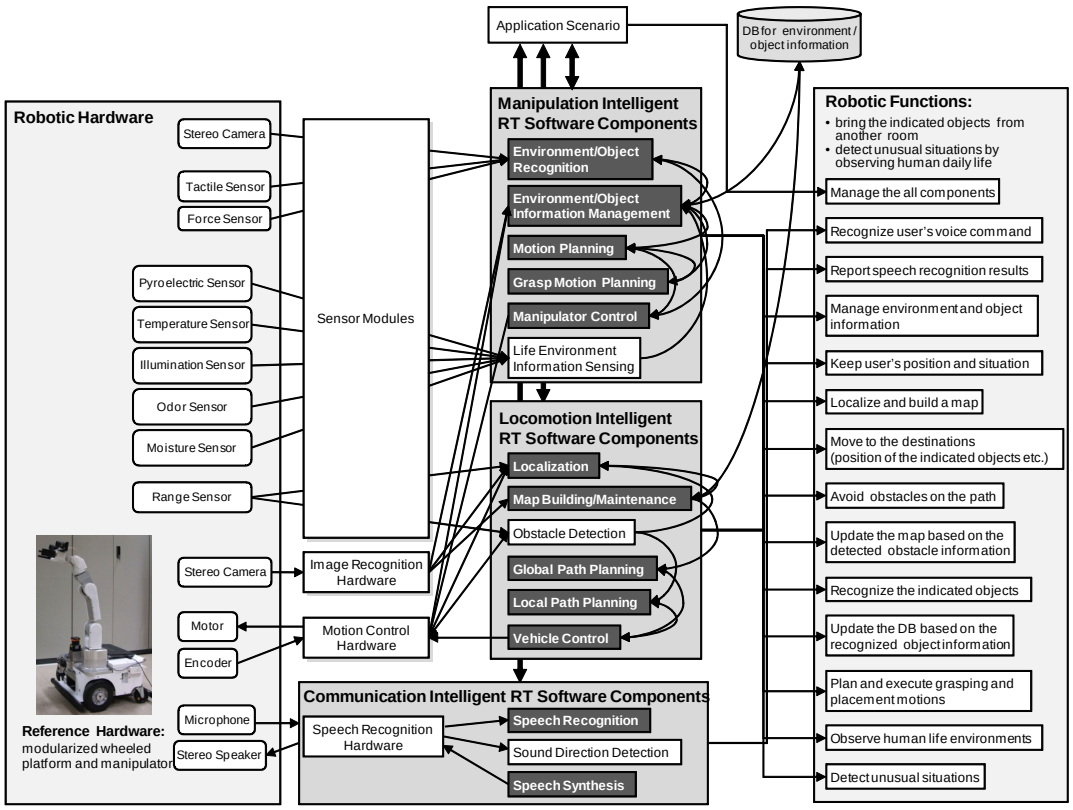

Fig. 5. Intelligent RT software components for a mobile manipulator system. Those in gray were used in our experiments.

Global path planning components

A collision-free path connecting the initial location and given destination is planned as a grid trajectory using a potential-based method.

Local path planning components

A local path is replanned based on the global grid trajectory, current position and orientation, and obstacle information to return the wheeled robot to the reference trajectory.

Vehicle control components

To move the robot steadily and efficiently on the planned path, the robot's reference velocity is calculated. Both a real and simulated wheeled robots are controlled.

3.3. Communication Intelligence

Speech recognition component: Japanese language utterances are recognized from voice input with a dictionary implemented based on the subset of Speech Recognition Grammar Specification (SRGS).

Speech synthesis component: Input Japanese sentences are uttered based on language analysis and phoneme composition results.

4. Mobile Manipulator System Development

RT-Components for manipulation, locomotion, and communication intelligence presented in the above section are integrated using the robot application scenario tools in the OpenRT Platform for a task fetching indicated objects for users.

4.1. Intelligent RT Software Component Integration Using Scenario Tools

To develop an efficient system in which system reconstruction or task process change are simple, the component groups used in experiments were integrated as follows:

- Connect component groups through the scenario player as often as possible to reconstruct the system easily.
- Connect component groups directly when large amounts of data or complex format data are sent in the connection.

The component groups below were connected to the scenario player with the following functions (only the main functions are shown):

Manipulation intelligence

- Environment and object information management components
  command: `QueryKnownLocation(object_name)`
  command: `QueryDetectedLocation(object_name)`
  event: `DetectingObject(object_name)`
  event: `DetectionReport(object_name,result)`
  event: `MobilePlatformGoToGoal(x,y,orientation)`
- Motion planning components
  command: `StartPlannedMotion()`
- Grasp motion planning components
  command: `GraspObject(object_name)`
  command: `PlaceObject(destination_name)`
  event: `PlannerReport(command_name,result)`

Locomotion intelligence

- GUI for managing locomotion intelligence
  command: `GoToGoal(x,y,orientation)`
  event: `ArrivedAt(x,y,orientation)`

Communication intelligence

- Speech recognition component
  command: `StartRecognization()`
  event: `SpeechRecognized(string)`
- Speech synthesis component
  command: `StartSpeak(string)`

In **Fig. 6**, a scenario developed for the target task is illustrated. The scenario player manipulates workers in turn based on an input scenario, first requesting the communication components to recognize the user voice indication, querying the location where the indicated object seems to exist to the environment and object information management component, and ordering the locomotion components to move there. After the robot reaches the goal, the player requires the environment and object information management component to recognize the actual position and orientation of the target object. The player orders the grasp motion planning components to plan grasping, and requests that the motion planning component implements the plan. The task process branches based on task conditions as shown in **Fig. 6**.

The connection relationship of the components is presented in **Fig. 7**. The blue box is an RT-Component, the connection between small green boxes is a ServicePort connection, and other connections are DataPort connections. The colored line is the interaction between the scenario player and worker, the blue line sending command messages and the green line event messages. As shown in **Fig. 7**, this system is simulated only by changing the manipulator control components. The mobile manipulator system was constructed by integrating 26 RT-Components excluding the scenario player.

4.2. Experimental Setup

The experimental environment for the target task and a mobile manipulator used for the experiments are presented in **Figs. 8** and **9** respectively. As shown in **Fig. 9**, the mobile manipulator RH1 consists of a 6 Degrees of Freedom (DOF) serial link manipulator unit and a 2-wheeled vehicle unit developed in the Intelligent RT Software Project as reference hardware for RT-Component research applications. The modularized manipulator and vehicle enable users to use each module alone. The 755 mm high manipulator has a 2 kg payload and picks up an object from the floor and puts it on a 700 mm high table. The 500 mm wide and 543 mm long wheeled vehicle turns in a 800 mm wide corridor. The stereo camera system equipped at the wrist of the manipulator enables the robot to recognize its work environment and objects.

4.3. Experimental Results

The experimental results are shown in **Fig. 10**. The experiments were done assuming the size of the room and the information of the objects present in the room to be saved in advance in the database. The operator vocally orders a red drink can. The robot then recognizes the command and determines from the environment and object information management component that the can is on the table. The robot goes to the table, locates the can, and planes and executes a collision-free trajectory for picking the can up. After that, the robot takes it to the indicated table and sets it down. The target task was successfully executed based on the developed scenario shown in **Fig. 6**.

5. Estimation of Robot System Development Using OpenRT Platform

In this paper, we aimed to show it is practicable to develop a complex robot system and its application using the OpenRT Platform by constructing the mobile manipulator system. We evaluated the platform from the following perspectives: RT-Component development; robot system and application construction, verification, and modification.

The mobile manipulator system used in the experiments was constructed by 5 developers, starting with the modularization of intelligent robotic functions, and determining only mutual interfaces. Individual developers were responsible for one or more intelligent functions and implemented RT-Components independently for their own intelligent functions. RT-Components developed for the mobile manipulator system numbered 26, and all were integrated based on the target task scenario using the RT System Editor. Simulation using OpenHRP3 was conducted in both verification processes for the RT-Components and integrated mobile manipulator system. For this system development, RT-Component Builder, RT System Editor, the robot application scenario tools, and OpenHRP3 in the OpenRT Platform were utilized.

Once interfaces between RT-Components are determined, developers need not consider other components, enabling us to concentrate on implementing and debugging of our own RT-Components and easily exchanging effective information about system integration. We could

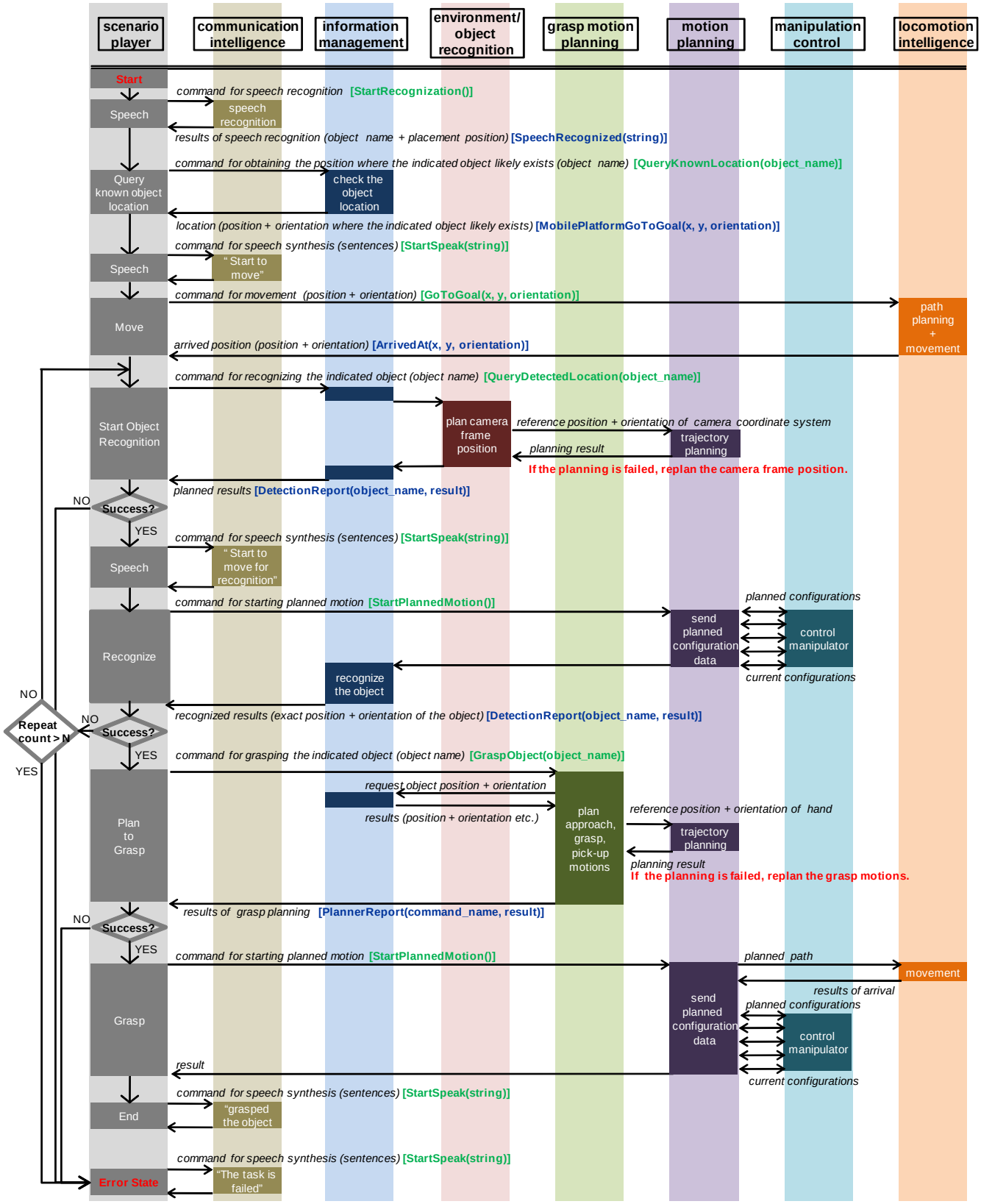

Fig. 6. Object pickup scenario as the first task process. Subsequent scenarios for bringing the object and putting it on indicated table are similar.

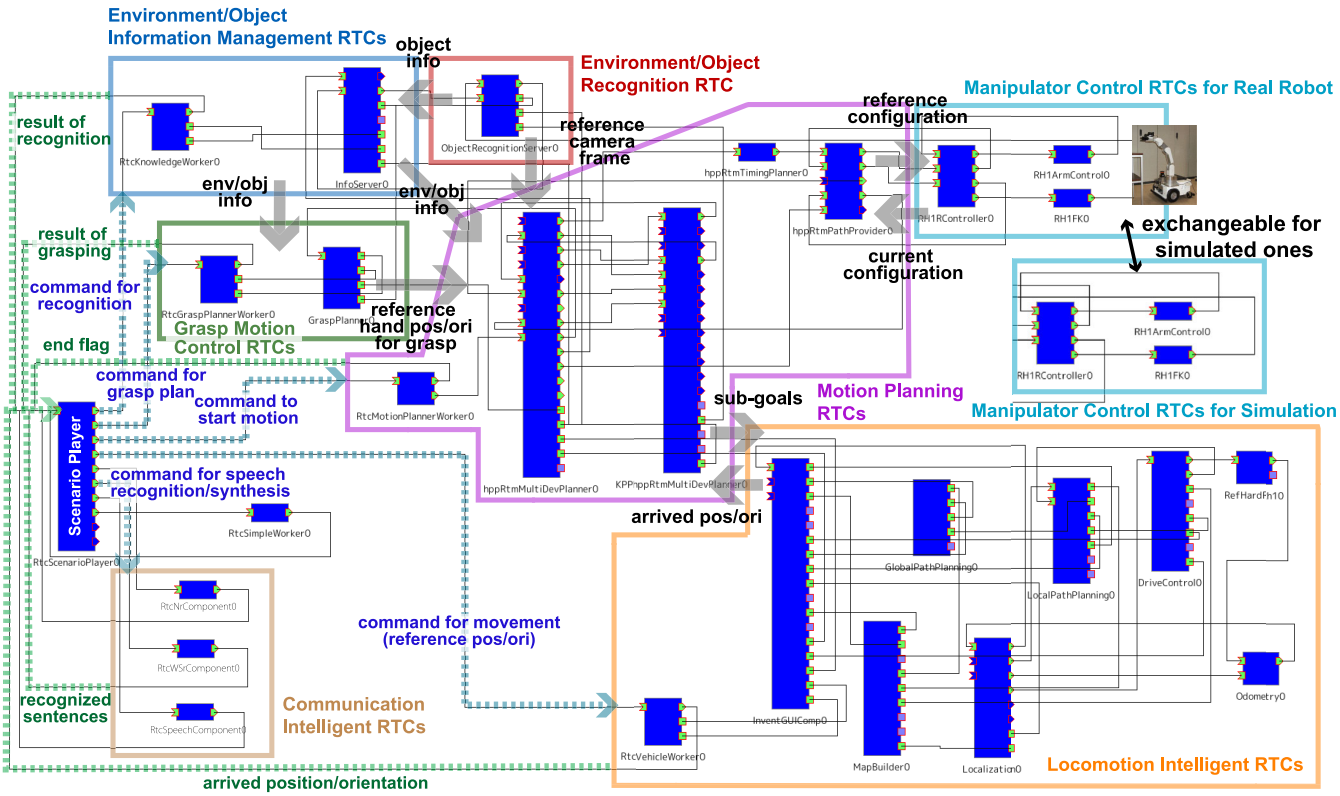

Fig. 7. RT-Component and scenario player connection for the mobile manipulator system depicted based on an RT System Editor screenshot.

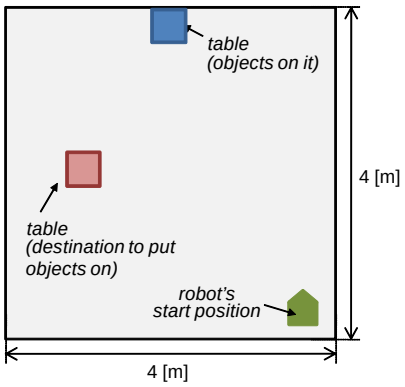

Fig. 8. Experimental environment.

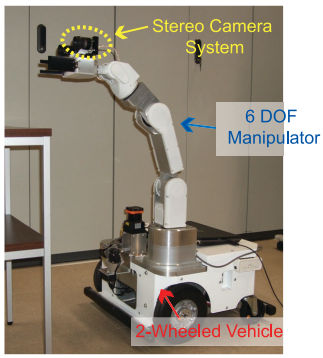

Fig. 9. Mobile manipulator used for experiments: reference hardware RH1 consisting of a modularized 6 DOF manipulator and a 2-wheeled vehicle, and a stereo camera system.

spend almost all time on core logic development because the RT-Component Builder automatically generates template files for an RT-Component based on component specifications. If component specifications are changed during development, the builder automatically inserts and deletes program code, which is related to the change, for RT-Component program being developed.

Using RT System Editor, the mobile manipulator system was constructed easily by connecting ports of RT-Components via a real-time GUI. After ports are connected, the editor checks connection compatibility, enabling simple and reliable system construction. The RT System Editor manipulates RT-Components, such as connection of ports, definition of component parameters, activation/deactivation of components, etc., so all system construction, modification, and verification processes are completed seamlessly by the same editor. RT-Components and the mobile manipulator system are simulated by exchanging the hardware RT-Components for the simulated ones on the editor using OpenHRP3. A developed system's structure and setting are saved and restored using the editor, enhancing reusability of constructed systems.

In the application development processes, a scenario facilitates system integration and promotes understanding of the developing system. Once the minimum task pro-

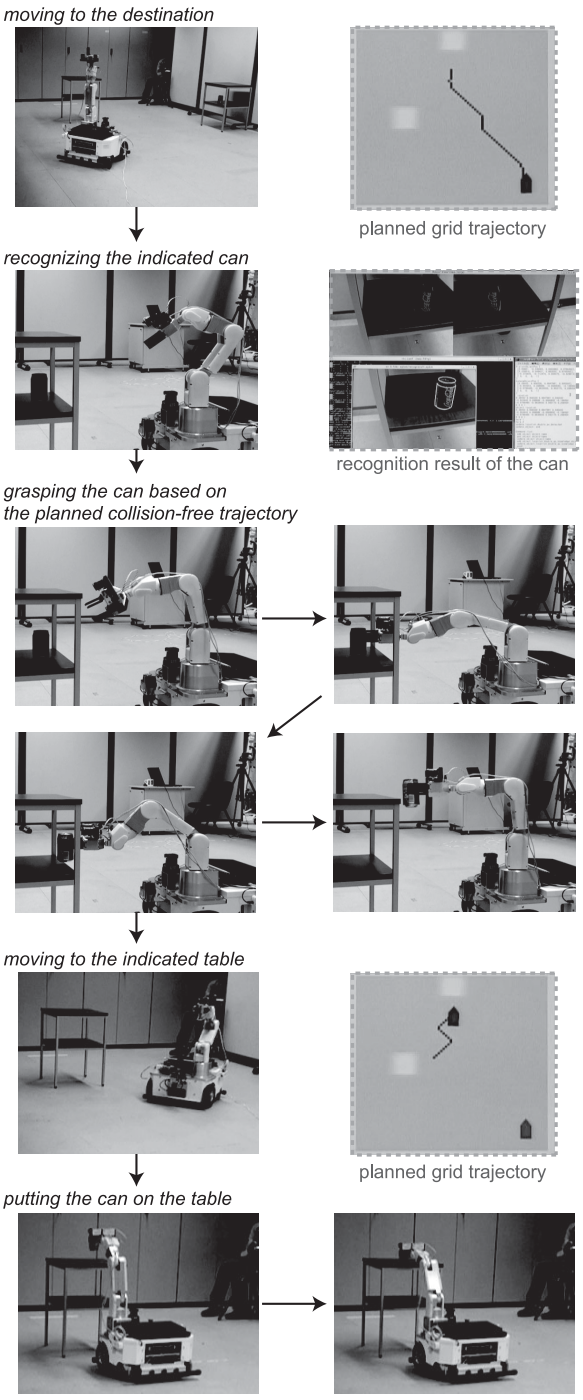

Fig. 10. Experimental results based on the developed scenario (**Fig. 6**).

cess scenario was implemented and the developing system was verified, the scenario was improved by adding useful processes such as error management. The scenario tools enables task processes to be readily changed without any modification or recompiling of RT-Components.

All the aboves are the advantages of robot system development using the OpenRT Platform. However, one difficulty remaining is debugging a whole robot system. While there is the RT-Component Debugger for testing one RT-Component, additional functionality for checking data flow in a whole system would be also required. OpenRT Platform tools are being developed to provide a complete toolchain, and our verification results are being fed back for improvement. Determining RT-Component modularization boundaries is also a difficult and important problem. Too big decreases reusability. Conversely, too small makes system development cumbersome. We plan to show more examples of modularized intelligent RT-Components in the future.

6. Conclusions

We have developed a mobile manipulator system that fetches indicated objects for users based on the OpenRT Platform to verify its effectiveness. RT-Components for manipulation, locomotion, and communication functions have been developed and integrated using the scenario tools. The mobile manipulator system executed the target task in both real and simulation world based on the developed scenario, demonstrating OpenRT Platform practicality in applications for complex robot systems.

Acknowledgements

This work was supported by the Intelligent RT Software Project of NEDO.

References:

[1] T. Sato and H. Hirukawa, "Intelligent RT Software Project," Proc. of Annual Conf. of the Robotics Society of Japan, AC1F1-01, 2008. (in Japanese)
[2] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, and W. K. Yoon, "RT-Middleware: Distributed Component Middleware for RT (Robot Technology)," Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 3555-3560, 2005.
[3] Object Management Group, "Robotic Technology Component Specification Version 1.0," formal/2008-04-04.
[4] T. H. J. Collett, B. A. MacDonald, and B. P. Gerkey, "Player 2.0: Toward a Practical Robot Programming Framework," Proc. of the Australasian Conf. on Robotics and Automation, 2005.
[5] H. Bruyninckx, "Open Robot Control Software: the OROCOS Project," Proc. of IEEE Int. Conf. on Robotics and Automation, pp. 2523-2528, 2001.
[6] A. Makarenko, A. Brooks, and T. Kaupp, "Orca: Components for Robotics," Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Workshop on Robotic Standardization, 2006.
[7] C. Schlegel, "A Component Approach for Robotics Software Communication Patterns in the OROCOS Context," Autonome Mobile Systeme (AMS), Informatik aktuell, Springer, pp. 253-263, 2003.
[8] I. A. Nesnas et al., "CLARAty: Challenges and Steps Toward Reusable Robotic Software," Int. J. of Advanced Robotic Systems, Vol.3, No.1, pp. 023-030, 2006.
[9] M. Mizukawa et al., "ORiN: Open Robot Interface for the Network. The Standard and Unified Network Interface for Industrial Robot Applications," Proc. of SICE Annual Conf., pp. 1160-1163, 2002.
[10] J. Jackson, "Microsoft Robotics Studio: A Technical Introduction," IEEE Robotics and Automation Magazine, Vol.14, pp. 82-87, 2007.
[11] M. Quigley et al., "ROS: An Open Source Robot Operating System," Proc. of IEEE Int. Conf. on Robotics and Automation, Workshop on Open Source Software, 2009.
[12] Object Management Group, Common Object Request Broker Architecture (CORBA) Specification Version 3.1, formal/2008-01-04.
[13] S. Nakaoka, S. Hattori, F. Kanehiro, S. Kajita, and H. Hirukawa, "Constraint-based Dynamics Simulator for Humanoid Robots with Shock Absorbing Mechanisms," Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 3641-3647, 2007.
[14] Y. Takano, A. Uda, and M. Ishida, "The Design of Robot Software development kit RoboStudio," Proc. of JSME Conf. on Robotics and Mechatronics, 1A1-L1-10, 2004. (in Japanese)

Name:
Natsuki Yamanobe

Affiliation:
Researcher, Intelligent Systems Research Institute, National Institute of Advanced Industrial Science and Technology (AIST)

Address:
1-1-1 Tsukuba Central 2, Umezono, Tsukuba, Ibaraki 305-8568, Japan

Brief Biographical History:
2004 M. Eng Degree (Precision Engineering, University of Tokyo)
2007 Dr. Eng Degree (Precision Engineering, University of Tokyo)
2007- Researcher, Intelligent Systems Research Institute, National Institute of Advanced Industrial Science and Technology (AIST), Japan

Main Works:
- "Motion Generation for Clutch Assembly by Integration of Multiple Existing Policies," IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 3218-3223, 2008.

Membership in Academic Societies:
- The Institute of Electrical and Electronics Engineers (IEEE)
- The Japan Society of Mechanical Engineers (JSME)
- The Japan Society of Precision Engineers (JSPE)
- The Robotics Society of Japan (RSJ)

Name:
Ee Sian Neo

Affiliation:
Research Scientist, National Institute of Advanced Industrial Science and Technology
(Current) Invention Specialist, Intellectual Ventures

Address:
1-1-1 Tsukuba Central 2, Umezono, Tsukuba, Ibaraki 305-8568, Japan
(Current) 150 Beach Road, #08-05/08, Gateway West 189720, Singapore

Brief Biographical History:
2006- Joined National Institute of Advanced Industrial Science and Technology
2009- Joined Intellectual Ventures

Main Works:
- "Whole-Body Motion Generation Integrating Operator's Intention and Robot's Autonomy in Controlling Humanoid Robots," IEEE Trans. on Robotics, Vol.23, No.4, pp. 763-775, 2007.

Membership in Academic Societies:
- The Robotics Society of Japan (RSJ)
- The Institute of Electrical and Electronics Engineers (IEEE)

Name:
Eiichi Yoshida

Affiliation:
Co-Director, CNRS-AIST JRL (Joint Robotics Laboratory), UMI3218/CRT, Intelligent Systems Research Institute, National Institute of Advanced Industrial Science and Technology (AIST)

Address:
1-1-1 Tsukuba Central 2, Umezono, Tsukuba, Ibaraki 305-8568, Japan

Brief Biographical History:
1990-1991 Research Student in Swiss Federal Institute of Technology at Lausanne (EPFL)
1993 M.Eng Degree (School of Engineering, University of Tokyo)
1996 Dr.Eng Degree (School of Engineering, University of Tokyo)
1996- Joined former Mechanical Engineering Laboratory
2001- Senior Research Scientist, Intelligent Systems Research Institute, National Institute of Advanced Industrial Science and Technology (AIST), Japan
2004-2008 Co-Director of AIST/IS-CNRS/ST2I Joint French-Japanese Robotics Laboratory (JRL) at LAAS-CNRS, Toulouse, France
2009- Co-Director of CNRS-AIST JRL (Joint Robotics Laboratory), UMI3218/CRT, AIST, Japan

Main Works:
- "Pivoting based manipulation by a humanoid robot," Autonomous Robots, Vol.28, No.1, pp. 77-88, 2010.
- "Planning 3D Collision-Free Dynamic Robotic Motion through Iterative Reshaping, IEEE Trans. on Robotics, Vol.24, No.5, pp. 1186-1198, 2008.

Membership in Academic Societies:
- The Institute of Electrical and Electronics Engineers (IEEE)
- The Japan Society of Mechanical Engineers (JSME)
- The Society of Instrument and Control Engineers (SICE)
- The Japan Society of Precision Engineers (JSPE)
- The Robotics Society of Japan (RSJ)

Name:
Nobuyuki Kita

Affiliation:
Senior Researcher, Humanoid Research Group, Intelligent Systems Research Institute, National Institute of Advanced Industrial Science and Technology (AIST)

Address:
1-1-1 Tsukuba Central 2, Umezono, Tsukuba, Ibaraki 305-8568, Japan

Brief Biographical History:
1981- Joined Electrotechnical Laboratory (ETL)
2009- Humanoid Research Group

Main Works:
- Active vision, robotics vision.

Membership in Academic Societies:
- The Robotics Society of Japan (RSJ)

Name:
Kazuyuki Nagata

Affiliation:
Senior Research Scientist, Intelligent Systems Research Institute, AIST

Address:
1-1-1 Tsukuba Central 2, Umezono, Tsukuba, Ibaraki 305-8568, Japan

Brief Biographical History:
1986- Joined Tohoku National Industrial Research Institute (TNIRI) at former AIST of MITI
1991- Joined Electrotechnical Laboratory (ETL) at former AIST of MITI
2001- Assigned to Planning Headquarters of AIST
2002- Intelligent Systems Research Institute of AIST

Main Works:
• "Manipulation by a Parallel-Jaw Gripper Having a Turntable at Each Fingertip," Proc. of 1994 IEEE Int. Conf. on Robotics and Automation, pp. 1663-1670, 1994.
• "Grasping Operation Based on Functional Cooperation of Fingers," J.of Robotics and Mechatronics, Vol.19, No.2, pp. 134-140, 2007.

Membership in Academic Societies:
• The Japan Society of Mechanical Engineers (JSME)
• The Robotics Society of Japan (RSJ)
• The Society of Instrument and Control Engineers (SICE)

Name:
Kazuhito Yokoi

Affiliation:
Deputy Director, Intelligent Systems Research Institute (IS), National Institute of Advanced Industrial Science and Technology (AIST)
Research Group Leader of Humanoid Research Group, IS/AIST
Adjunctive Member, CNRS-AIST JRL UMI3218/CRT
Adjunctive Professor, University of Tsukuba
Visiting Professor, Fukuyama University

Address:
1-1-1 Tsukuba Central 2, Umezono, Tsukuba, Ibaraki 305-8568, Japan

Brief Biographical History:
1986- Joined Mechanical Engineering Laboratory, Ministry of International Trade and Industry, Japan
2001- Senior Researcher, Intelligent System Research Institute (IS), National Institute of Advanced Industrial Science and Technology (AIST)
2009- Deputy Director, IS/AIST

Main Works:
• "Planning 3D Collision-Free Dynamic Robotic Motion through Iterative Reshaping," IEEE Trans. on Robotics, Vol.24, No.5, pp. 1186-1198, Oct. 2008.

Membership in Academic Societies:
• The Japan Society of Mechanical Engineers (JSME)
• IEEE, Robotics and Automation Society
• The Robotics Society of Japan (RSJ)

Name:
Yosuke Takano

Affiliation:
Senior Manager, Services Platforms Research Laboratory, NEC Corporation

Address:
1753 Shimonumabe, Nakahara, Kawasaki, Kanagawa 211-8666, Japan

Brief Biographical History:
1989- Joined NEC Corporation
2000- Engaged in research of the software platform of personal robots

Main Works:
• "Field trial of Asynchronous Communication Using Network-based Interactive Child Watch System for the Participation of Parents in Day-care Activitie," IEEE Int. Conf. on Robotics & Automation, 2008.

Membership in Academic Societies:
• Information Processing Society of Japan (IPSJ)