J Electr Eng Technol Vol. 9, No. 2: 755-761, 2014
http://dx.doi.org/10.5370/JEET.2014.9.2.755

ISSN(Print) 1975-0102
ISSN(Online) 2093-7423

An Effective Method of Sharing Heterogeneous Components of OPRoS and RTM

Andrey D. Salov*, Hong Seong Park†, Soohee Han and Dooam Lee****

Abstract – Heterogeneous components have different component models, which prevents such components from sharing the functionalities of other components based on the different models. As one of methods for linking heterogeneous components, this paper suggests a proxy component to construct a bridge between heterogeneous components of OPRoS (Open Platform for Robotic Service) and RTM (Robot Technology Middleware). The proxy component consists of two types of components called Adaptor and Interceptor, via which the heterogeneous components can exchange data and services easily. The proposed method enables adaptor and interceptor components to directly invoke the services of the latter and the former, respectively, in order to exchange data and services on a real-time basis. The proxy component can be implemented for OPRoS and RT (Robot Technology) component models to connect with RT and OPRoS ones, respectively. It is shown through a simple experiment that the proposed method works well for real-time control.

Keywords: Robot middleware, Component, Open Platform for Robotic Services (OPRoS), Robot Technology Middleware (RTM), Robot Technology (RT) components, Proxy component

1. Introduction

The cost of implementing numerical algorithms and controlling various HW devices accounts for a large part of the overall cost of developing a typical robot SW applications. Furthermore, the portion of such cost has been increasing since recent robot SW applications require more elaborate computations and controls for more advanced and intelligent behaviors. For these reasons, many attempts have recently been made all over the world to propose the specifications on components (or SW modules) and to develop common development platforms for robot SW applications [1-8]. Until now, there have been some well known robot SW platforms such as "OPRoS (Open Platform for Robotic Service) [6], RTM (Robot Technology Middleware) [7], and OROCOS (Open Robot Control Software) [8]. As in a general SW module, components provided by such robot SW platforms employ a black-box style and have well-defined interfaces. Additionally, the robot SW component models are required to meet the real-time constraints.

As mentioned earlier, components can be easily utilized in their platforms to make some applications quickly. However, it is not easy to involve the heterogeneous components in our platform in order to make the use of their functions that we don't have in our platform. In this regard, there have been some studies on how to share heterogeneous components with SW connectors, making applications more easily and quickly [9-10]. The studies focused on the general component models such as SOFA [11] and EJB (Enterprise JavaBeans). It would be meaningful to extend these studies to robot SW components for robot applications with real-time constraints. To the best of the authors' knowledge, there has been no trial for sharing the heterogeneous robot SW components with real-time constraints.

This paper considers two representative robot SW component models, OPRoS and RT component models among several others, which are well specified and standardized. OPRoS and RT components include a variety of useful functions, satisfy real-time constraints, and help users applying them do so with the best (or most cost-effective) performance in an efficient way. Especially, OPRoS and RT components also have features such as the periodical data transmission via the data port and the priority scheduling. It would be very beneficial to share such nice components and hence borrow useful functions from each other. For the sake of sharing two heterogeneous robot components, the connection (or data-exchanging) method between them should be developed.

This paper suggests an effective method of sharing the OPRoS components and the RT (Robot Technology) components for the robot SW applications with real-time constraints while maintaining the structures of OPRoS and RT component models without modifying any source codes. In order to share the RT components and the OPRoS components in an effective way, a proxy component is proposed, which consists of Adaptor and Interceptor: The

† Corresponding Author: Dept. of Electrical and Computer Engineering, Kangwon National University, Korea. (hspark@kangwon.ac.kr)
* Dept. of Electrical and Computer Engineering, Kangwon National University, Korea. (andrey@control.kangwon.ac.kr)
** Dept. of Electrical Engineering, Konkuk University, Korea. (shhan@konkuk.ac.kr, dlendka@nate.com)
Received: August 19, 2013; Accepted: November 18, 2013

former is the component employed in one component model and the latter is the component in the other component model. The proposed proxy component enables the OPRoS and RT components to utilize functionalities of each other. To illustrate the effectiveness of the proposed method, it is shown through experiments with an electrical power wheelchair that the performance measures of the steering control components such as latency time and control characteristics are almost the same for an OPRoS application, an RT one, and an OPRoS plus RT one.

The paper is organized as follows. In Section 2, a proxy component is introduced together with its elements and the related class hierarchy. The effectiveness of the proposed method is illustrated through a simple experiment in Section 3. Finally, a conclusion is drawn in Section 4.

2. A Proxy Component: Adaptor & Interceptor

The simple hierarchical class structure for bridging OPRoS and RT components is depicted in Fig. 1. Two main components, an Adaptor component (or AdaptorComp) and an Interceptor component (or InterceptorComp), are inherited from OPRoS and RT basic components, respectively. Hence, they can be applied only to their inherent platforms. In other words, the adaptor to RTM and the interceptor to OPRoS are not possible. The architecture and the detailed hierarchical class structure of the proxy component, AdaptorComp and InterceptorComp, can be seen in Figs. 2 and Fig. 3. In this section, OPRoS components and RT components are denoted by OPRoSC and RTC, respectively, for simplicity.

The Adaptor component and the Interceptor component

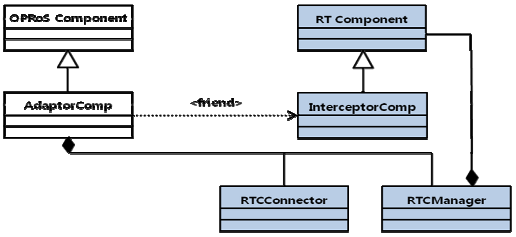


Fig. 1. The overall structure of a proxy component including the AdaptorComp class for an OPRoS component and the InterceptorComp Class for an RT component

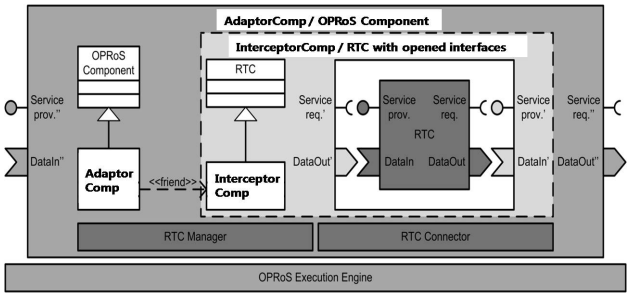


Fig. 2. The architecture of AdaptorComp and Interceptor-Comp

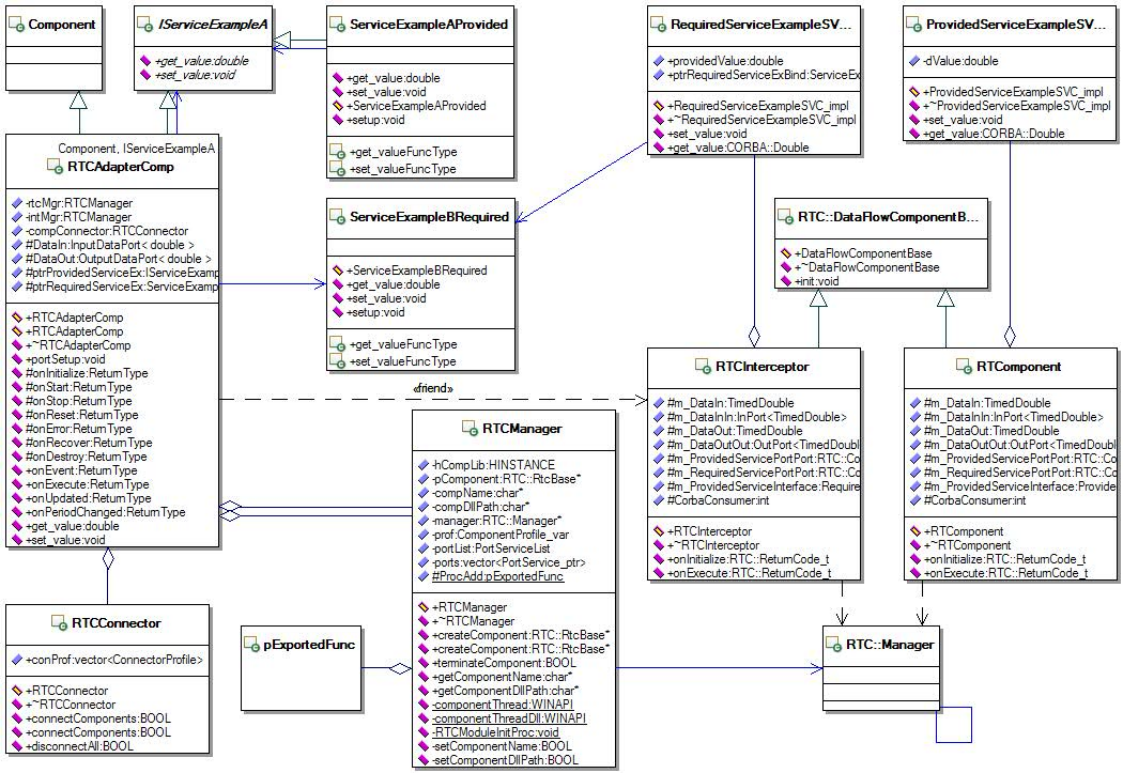


Fig. 3. The class hierarchy of AdaptorComp and InterceptorComp

have four kinds of interfaces for data input, data output, provided services, and required services, and hence make it possible to construct a communication channel between an OPRoSC and an RTC. To exchange data between an OPRoSC and an RTC, the OPRoSC connects to AdaptorComp and then InterceptorComp connects to the RTC. The sequences for the data exchange are shown in Figs. 4 and Fig. 5. It can be seen that data is sent from the OPRoSC to the RTC through a data port and it is requested from the RTC to make use of services provided by the OPRoSC.

Since AdaptorComp and InterceptorComp are inherited from the OPRoSC and RTC base classes, respectively, the OPRoSC (or the RTC) can access an RTC (or an OPRoSC) through AdaptorComp (or InterceptorComp). Interceptor-Comp (or AdaptorComp) allows the AdaptorComp (or InterceptorComp) to access all public, private, and protected types of interfaces of the RTC (or the OPRoSC), where the AdaptorComp class has a friend relationship with the InterceptorComp class. Both Adaptor-Comp and InterceptorComp have the same number of ports as the corresponding RTC and OPRoSC. Especially, Interceptor-Comp in Figs. 1 and Fig. 2 makes use of a Robot Technology Middleware (RTM) libraries in order to exchange data with the RTCs. Note that the RTM includes an RTC manager and an RTC Connector. The former is in charge of creating, activating, deactivating, and terminating the RTC, and the latter manages the connection between an RTC and InterceptorComp. RTCConnector and RTC-Manager have functions that manage RT components and make their connections to OPRoS through RTM libraries. It means that even OPRoS components can freely access RT components with RTCConnector and RTCManager. This is why AdaptorComp has a composition relationship with RTCConnector/RTCManager. InterceptorComp has open interfaces of an RT component and makes them accessed from AdaptorComp using RTCConnector/RTC Manager. AdaptorComp has a friend function to InterceptorComp using RTCConnector and RTCManager in order to communicate with RT components.

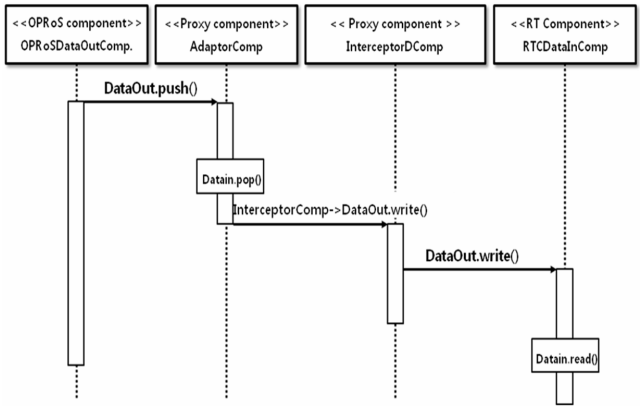


Fig. 4. An OPRoS component (OPRoSDataOutComp) sends data to an RT component (RTCDataInComp)

As shown in Figs. 4 and Fig. 5, AdaptorComp calls directly the data port-related function of the Interceptor-Comp to send the data from OPRoS component to RT component. InterceptorComp calls directly the service port-related function of the AdaptorComp to invoke the corresponding OPRoS method and get the value generated by this method. In Fig. 4, the OPRoS component sends data to the AdaptorComp through a data port. The AdaptorComp receives this data and sends it to the RTC through the data port-related function of InterceptorComp. Finally, the RTC receives data. Of course, the converse is possible as follows: the RTC sends data to Interceptor-Comp through a data port. InterceptorComp receives this data and sends it to the OPRoS component through the data port-related function of AdaptorComp.

In the case of service ports in Fig. 5, an RTC invokes a method provided by InterceptorComp. Subsequently, InterceptorComp invokes a method provided by an OPRoSC through AdaptorComp. In an opposite direction, return values generated from the OPRoSC are received by o the RTC.

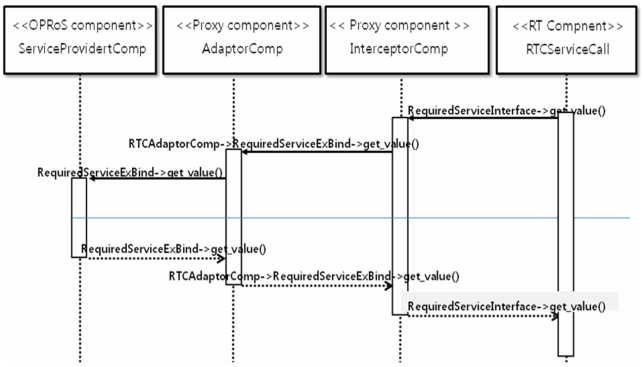


Fig. 5. An RT component invokes methods provided by an OPRoS component (ServiceProviderComp).

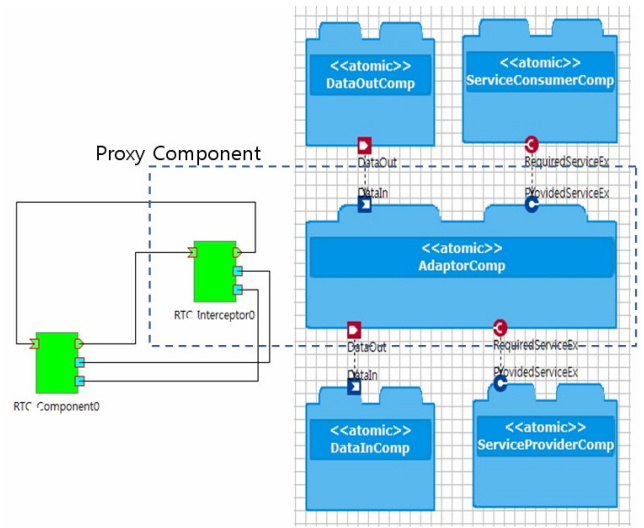


Fig. 6. A simple application example with a proxy component made up of AdaptorComp and RTCIntercetptor

Since the proposed method consumes negligible time in conversion of data of an OPRoSC to that of an RTC, it can be said that this method would not have a significant effect on periodic execution. It means that both OPRoS and RTM take charge of the real-time constraint of periodic execution.

As one example, a simple application with a proxy component is depicted in Fig. 6, in which both OPRoSC and RTC are used at the same time. Figs. 7 and Fig. 8 show the profiles of AdaptorComp and RTCInterceptor acquired at the runtime system in Fig. 6, respectively. Note that AdaptorComp is the OPRoSC and RTCInterceptor is the RTC.

Since both of OPRoS and RTM support a distributed environment, the proposed scheme also works well for a distributed environment.

```
<application_profile>
    <id>11f0efa.136c6a591d0.1.21ee</id>
    <name>testapp</name>
    <subcomponents>
      <subcomponent>
          <name>DataInComp</name>
          <id>121bfd6.136c6a5b2d6.9.1bb7</id>
          <type>atomic</type>
          <reference>DataInComp.xml</reference>
      </subcomponent>
      ......
      <subcomponent>
          <name>AdaptorComp</name>
          <id>246ceb.136c6a5cf08.10.15a1</id>
          <type>atomic</type>
          <reference>AdaptorComp.xml</reference>
      </subcomponent>
      ......
      <subcomponent>
          <name>ServiceProviderComp</name>
          <id>1f23c6d.136c6a5e792.17.bf1</id>
          <type>atomic</type>
          <reference>ServiceProviderComp.xml</reference>
          </subcomponent>
    </subcomponents>
    <port_connections>
      <port_connection port_type="data">
          <source component_name="AdaptorComp" port_name="DataOut"/>
          <target component_name="DataInComp" port_name="DataIn"/>
      </port_connection>
      <port_connection port_type="service">
          <source component_name="ServiceConsumerComp" port_name="RequiredServiceEx"/>
          <target component_name="AdaptorComp" port_name="ProvidedServiceEx"/>
      </port_connection>
      <port_connection port_type="data">
          <source component_name="DataOutComp" port_name="DataOut"/>
          <target component_name="AdaptorComp" port_name="DataIn"/>
      </port_connection>
      <port_connection port_type="service">
          <source component_name="AdaptorComp" port_name="RequiredServiceEx"/>
          <target component_name="ServiceProviderComp" port_name="ProvidedServiceEx"/>
      </port_connection>
    </port_connections>
</application_profile>
```

Fig. 7. The OPRoS application profile with AdaptorComp for system in Fig. 6.

Fig. 8. The RTCInterceptor's profile (runtime)

3. Experiment

To illustrate the effectiveness of the proposed method, an experiment with an electric power wheelchair [12] was conducted. As shown in Fig. 9, the electric power wheelchair has DSP and IO boards for driving two rear motors and acquiring information on motor speeds from encoders. When a wheelchair is turning, the speed

Fig. 9. An electric power wheelchair

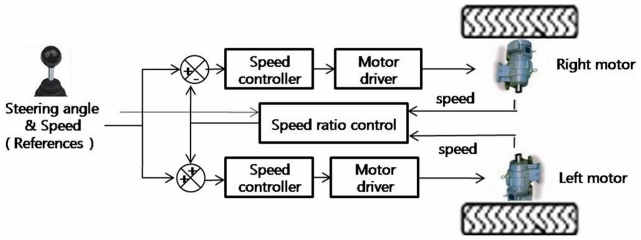


Fig. 10. The structure of a steering control system [12, 13]

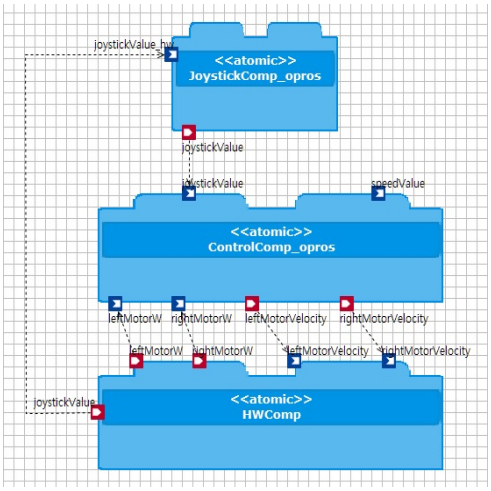


Fig. 11. OPRoS components and their composition

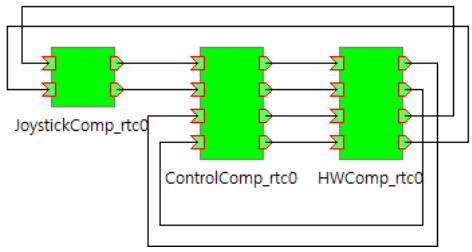


Fig. 12. RT components and their composition

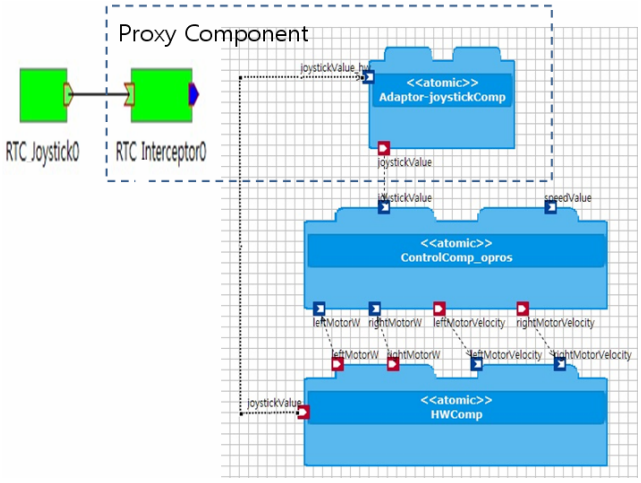


Fig. 13. Integration of one RT component and OPRoS components.

difference between the left and the right driving wheels is needed for smooth driving without slipping. The speed ratio of the left and right wheels should be properly controlled for smooth steering. The so called speed ratio control depicted in Fig. 10 is used with two DC motors for both drive wheels [12, 13].

All components required for driving a wheelchair are depicted using the OPRoS components and the RT components in Figs. 11 and Fig. 12, respectively. We can see three components for joystick handling, steering control, and HW access to DSP and IO boards in Fig. 9, which are named as JoystickComp, ControlComp, and HWComp. All components run periodically while sending or receiving data. They have the same period of 10 ms. Since this application is developed for executing a control program periodically, data ports are employed for fast communication. A component for joystick handling receives the positional data of the joystick from the component for HW handling. Computing the reference direction and speed, the joystick component sends them to the control component that computes a proper speed ratio and sends the corresponding reference voltages to the HW component. In this way, a control loop is formed to control the direction and the speed difference between the left and the right driving wheels. Among three OPRoS components in Fig. 11, the joystick component is replaced with an RTC in Fig. 13.

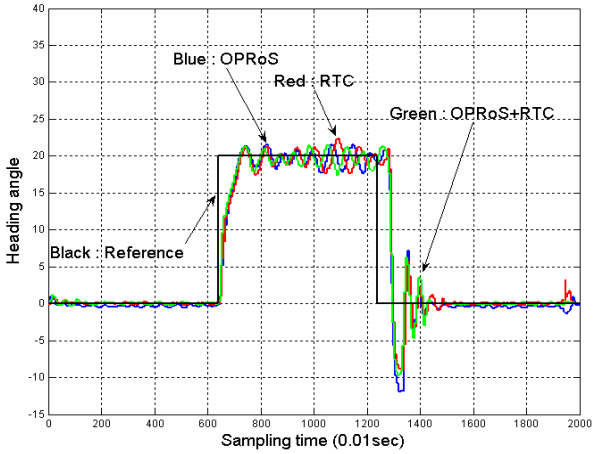


Fig. 14. Controlled steering angles for three kinds of applications

Table 1. Latency time between two components

Application Example	Fig. 11	Fig. 12	Fig. 13
Average(us)	4.916	4.907	4.909
Standard Deviation(us)	0.037	0.108	0.085
Min/Max Value(us)	4.9/5.0	4.4/5.0	4.6/5.0

When the wheelchair abruptly turns at a certain time point with the steering angle 20 degrees, the performance is compared in Fig. 14. Since the reference signal is electronically given, it is possible to make its abrupt change. In both the steady and transient states, the performance of the steering control remains almost unchanged even though the RT component runs in tandem with the OPRoS components. It means that the proposed method of integrating the RT and the OPRoS components does not affect the overall performance significantly and hence the RT components can be used together with the OPRoS components without any concern on the performance degradation.

Table 1 shows the latency time measured in application examples of Figs. 11, Figs. 12, and Fig. 13. The latency time means the elapsed time from data transmission of the JoystickComp component to data reception of the ControlComp component. From Table 1, we can see that the proxy component consumes a negligible processing time compared with that of OPRoS and RT components from the real-time viewpoint in ms time unit. The latency time were measured 100 times in each case and averaged in Table 1.

4. Conclusion

In this paper, a new proxy component for sharing the RT and OPRoS components has been proposed and developed, which consists of Adaptor for OPRoS components and Interceptor for RT components. For the proxy component to satisfy the real-time constraints, the Adaptor component

invokes directly the services of the Interceptor component to transmit data from OPRoS components to RT components. The converse is possible for transmitting data from RT components to OPRoS components.

The experiment with an electric power wheelchair has showed that the proposed method using a proxy component to share the RT and OPRoS components does not affect the performance significantly.

It is believed that the proposed method easily helps development of the robot SW applications by utilizing the various types of components available in OPRoS and RTM.

Acknowledgements

This work was supported by the Ministry of Trade, Industry, and Energy.

References

[1] S. Cousins, "Exponential Growth of ROS," *IEEE robotics & automation Mag.*, vol. 18, no. 1, pp. 19-20, Mar. 2011.

[2] D. Brugali, and A. shakhimardanov, "Component-Based Robotic Engineering (Part II)," *IEEE robotics & automation Mag.*, vol. 17, no. 1, pp. 100-111, Mar. 2010.

[3] A review of robotics SW platforms http://www.linuxfordevices.com/c/a/Linux-For-Devices-Articles/A-review-of-robotics-software-platforms

[4] C. Jang, S.-I. Lee, S.-W. Jung, B. Song, R. Kim, S. Kim, and C.-H. Lee, "OPRoS: A New Component-Based Robot Software Platform," *ETRI Journal*, vol. 32, no. 5, pp. 646-656, Oct. 2010.

[5] S. Han, M. Kim, and H. S. Park, "Open Software Platform for Robotic Services," *IEEE Trans. Automation Sci. and Eng.*, vol. 9, No. 3, pp. 482-495, Sep. 2012.

[6] Open Software Platform for Robotic Services (OPRoS) http://www.ropros.org

[7] Robot Technology Component (RTC) http://www.is.aist.go.jp/rt/OpenRTM-aist

[8] OROCOS, http://www.orocos.org

[9] L. Bulej, T. Bures, "Using Connectors for Deployment of Heterogeneous Applications in the Context of OMG D&C Specification", Proceedings of Interoperability of Enterprise Software and Applications, Feb., 2005, pp. 349-360.

[10] O. Galik and T. Bures,"Generating connectors for heterogeneous deployment," Proceedings of the 5th international workshop on Software engineering and middleware, Sep., 2005, pp. 54-61.

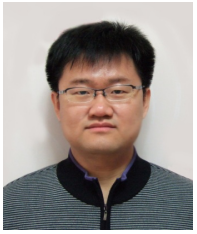
[11] SOFA, http://sofa.ow2.org/#overview

[12] S. You, H. Lee, D. Lee, H. Mok, Y. Lee, and H. Han, "Speed ratio control for electronics differentials," IEE Electronics Letters, vol. 47, No. 16, pp. 933-934, Aug., 2011.

[13] D.-A. Lee, D.-G. Jung, K.-S. Woo, L.-K Kim, H. Mok, and S. Han, "Orientation Compensation for Initially Misaligned Caster Wheels," International Journal of Control, Automation, and Systems, vol. 11, No. 5, pp. 1071-1074, Oct., 2013.

Andrey D. Salov He received the Specialist Degree in Engineering from Khabarovsk State University of Technology, Khabarovsk, Russia, in 2005. He is currently a Ph.D. candidate in the Department of Electronic and Communication Engineering, Kangwon National University, South Korea. His research interests are wireless networking, including wireless sensor networks, mobile ad hoc networks, routing protocols, topology control algorithms, power saving techniques, multicasting, radio propagation models, and real-time systems, distributed systems, embedded systems, control systems, robot middleware, and component-based software.

Hong Seong Park He received the B.S., M.S., and Ph.D. degrees from Seoul National University, Seoul, Korea, in 1983, 1986, and 1992, respectively. Since 1992, he has been with the Department of Electrical and Computer Engineering, Kangwon National University, Korea. Since 2007, he has been the Director of the project called "OPRoS (Open Platform for Robotic Services)" developing in Korea. His research interests include design and analysis of robot software platforms, communication networks and mobile/wireless communication, discrete-event systems, and network-based control systems.



Soo-hee Han He received the B.S. degree in electrical engineering from Seoul National University (SNU), Seoul, Korea in 1998. He received the M.S. and Ph.D. degrees in School of electrical engineering and computer science from SNU in 2000 and 2003, respectively. From 2003 to 2007, he was a researcher at the Engineering Research Center for Advanced Control and Instrumentation of SNU. In 2008, he was a senior researcher at the robot S/W research center. Since 2009, he has been with the Department of Electrical

Engineering, Konkuk University, Seoul, Korea. His main research interests are in the areas of computer aided control system design, distributed control system, time delay system, and stochastic signal processing.
He is currently an Associated Editor of International Journal of Control, Automation, and Systems, and Journal of Electrical Engineering & Technology. In 2012, he was a Guest Editor of Mathematical Problems in Engineering for optimal control problems in engineering. He served as the financial chair of IFAC and ICCAS for 2008 and 2012, respectively.

Doo-Am Lee He received the B.S. degree in electronic engineering from Mokpo National Maritime University, and M.S. degree in Electrical Engineering from Konkuk University. He is currently working toward the Ph.D. in Konkuk University. His current research interests include teleoperation control systems and electric vehicles.