論文 / 著書情報
Article / Book Information

Title	Study of Framework Based on Roles for Application Development of Service Robots
Author	Arturo E. Ceron Lopez, Edwardo F. Fukushima, Satoshi Kitano, Gen Endo
Journal/Book name	IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO), , , pp. 39-44
Issue date	2013, 11
DOI	http://dx.doi.org/10.1109/ARSO.2013.6705503
URL	http://www.ieee.org/index.html
Copyright	(c)2013 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other users, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works for resale or redistribution to servers or lists, or reuse of any copyrighted components of this work in other works.
Note	このファイルは著者（最終）版です。 This file is author (final) version.

Study of Framework Based on Roles for Application Development of Service Robots

Arturo E. Cerón López, Edwardo F. Fukushima, Satoshi Kitano and Gen Endo

***Abstract*— The modern approach for developing service robot applications is by the use of robotics middleware. Various issues are still present and several projects attempt to address them. However, an important issue that impacts the developers and integrators has not yet been thoroughly analyzed: usability. The usability of the development tools needs to be improved; it is required to access and work with service robots (and their components) with the least effort possible. This issue affects the introduction of service robots into the real world. In this article, we propose the Framework for Integration of Elements and Resources by Roles (FIERRo) to address the usability problem by making the items needed by developers and integrators as easily available and organized as possible. Description of example applications include: modeling of two independent robot systems, system development using and not using the FIERRo (hypotheses were set), and a simulated GUI. Social impact and future works are discussed thereafter.**

I. INTRODUCTION

Frameworks for developing service robot applications have passed through an intensive evolution process in recent years. Various models and abstractions have been proposed, each one of them aiming at particular goals for particular developer communities [1]. The modern approach for developing service robot applications is through the use of robotics middleware [1][2]. Examples include Robotics Technology Middleware (RT-Middleware) [3], Robot Operating System (ROS) [4] and Microsoft Robotics Developer Studio (MSRDS) [5]. By definition, robotics middleware is an abstraction layer that is found between the operating system and software applications; it sets the means for reusing and interconnecting distributed software programs in order to make them work in a collaborative and parallel way [6]; an example of a robot system built using robotics middleware is shown in Fig. 1.

However, the middleware approach still has many issues regarding the various application development stages. The stages may include: modeling, building, deployment, knowledge exchange and maintenance. Recent projects have proposed different approaches to address some of the existing issues, including the following:

1) RoboDB [7]: Is an attempt for categorizing robots (and related entities) by creating ontologies using robot's attributes, this is for reusing the information in a variety of environments. However, its reach is still limited to mainly linking attributes from robot to robot.

A. E. Cerón López, E. F. Fukushima, S. Kitano and G. Endo at Tokyo Institute of Technology, Department of Mechanical and Aerospace Engineering, 2-12-1, Ookayama, Meguro-ku, Tokyo 152-8552, Japan aceron at robotics.mes.titech.ac.jp, fukusima at mes.titech.ac.jp, kitano.s.ac at m.titech.ac.jp, gendo at mes.titech.ac.jp

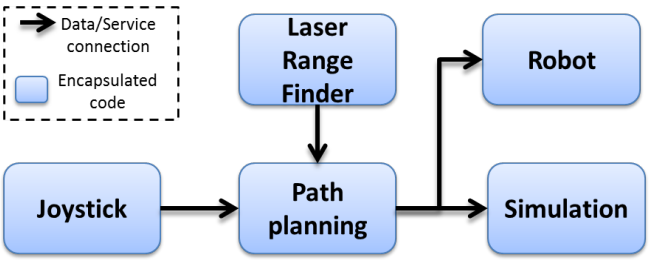

Fig. 1. Typical system diagram of an app. based in robotics middleware

2) Semantic Robot Space (SRS) [8]: A framework that builds semantic configurations for robots in order to discover and deploy context-aware services in a dynamic manner. Although the approach claims to reduce maintenance costs and allow an efficient data exchange among stakeholders in a robot space, it does not address the usability of such an environment when modeling and developing the system.

3) RoboEarth [9]: Project defining a language and database repository for exchanging knowledge about actions, objects and environments among robots. An API and engine for a computing environment on a cloud are provided. Despite this project having a complete repertoire of features, it still does not discuss in detail the usability of the proposed concepts for the developers and integrators.

4) AutomationML (AML) [10]: A standard markup language that attempts to model and unify all kinds of information used by engineering tools. For instance, it can describe task information and spatial configurations in robots. Its current main use is on industrial applications.

- Knowledge Integration Framework for Robotics [11]: In this project, the AutomationML standard is used for categorizing and distributing knowledge among robots and their users. However, usability is not discussed.

It is to be noted that RoboDB, Semantic Robot Space, RoboEarth and Knowledge Integration Framework for Robotics had attempted to take advantage of the Semantic Web Technologies by adopting the Resource Description Framework (RDF) as modeling and categorization means, while AutomationML provides their own standard. A related project to the topic is BRICS [12], which is a large scale project aiming to structure and formalize the robot software development process. However, an important issue that impacts the developers and integrators has not yet

been thoroughly analyzed: usability. The usability of the development tools needs to be improved; it is required to access and work with service robots (and their components) with the least effort possible, as well as making a robot system usable to other developers.

II. USABILITY ISSUES

According to the ISO 9242 definition[13], usability is: the effectiveness, efficiency and satisfaction with which specified users achieve specified goals in particular environments. The poor usability of the currently available development and runtime tools is a key issue that can cause bottlenecks in the development of robot systems, hindering the development of real-life service robot applications. Every time a technological improvement comes out, new procedures, tools and other kind of instruments are introduced, usually in addition to the previous ones. This brings new levels of development for service robot experts, but at the same time it introduces longer learning curves and compatibility issues for software-hardware developers and integrators that are not robot specialists, making it more difficult for them to enter the service robotics world. They often require making use of such robots (and their components) with the least effort possible.

From the previously mentioned projects, a subjective comparison was made between them, taking into account 8 selected features considered to be important for usability. The comparison is based on the level of support given by the projects to the selected features. The purpose of this comparison is to get a general idea in terms of usability for the system developer and integrator.

TABLE I
COMPARISON OF PROJECT FEATURES RELATED TO USABILITY

Feature	RTM	ROS	MSRDS	RoboDB	SRS	RoboEarth	AML
Ontological desc.	X	△	X	O	O	O	O
Config./ Mgmnt.	O	O	O	X	△	△	△
Deployment	O	O	O	X	△	△	△
Modeling Lang.	X	X	O	△	X	O	O
Knowldg. distrib.	△	△	△	△	△	△	△
Connectivity	O	O	O	X	O	O	△
Independency	△	△	△	O	△	X	O
Execution/ Runtime	O	O	O	X	O	△	X

The marks are as follows: O = Supported, △ = Limited support, X = Poor support.

An approach for describing and organizing a service robot system, that helps to improve the usability of development and runtime tools, is required. We have proposed a novel framework that will serve for this purpose, as well as for performing usability tests in the future. This article first introduces the proposed framework, then an implementation with some example applications are shown, and finally conclusions and future work are discussed.

III. FRAMEWORK FOR INTEGRATION OF ELEMENTS AND RESOURCES BY ROLES (FIERRo)

With this framework, we intend to address the usability problem by making the items needed by developers and integrators as easily available and organized as possible (e.g. configuration files, manuals, 3D models, running applications, etc.). The general concept of this framework is to dynamically create a flexible relational database, which is based on elements and resources that are used during the service robot's application lifecycle. Items are mainly elements and resources.

A. Basic abstractions

A set of basic abstractions has been considered for referring to the involved components in this framework:

- Objects: Entity representing something in the robot's environment (internal and/or external).
- Elements: Constituents or building blocks of an entity (e.g. a system, an object, etc.).
  - Software Elements: Refers to a single software program or a group of them that perform specialized functions (e.g. accessing hardware or processing information) for a service robot.
  - Structured Data Elements: Semantically arranged set of data. The sets can be merged and associated to other sets for complementing the pertinent information (e.g. alternate representations or states). It can be used to represent hierarchical data, such as constructs (e.g. position, velocity, etc.).
  - Other Elements: May refer to a property, a file and/or raw data.
- Resources: Data supplies and supportive subsystems that can be drawn upon when required. Elements turn into resources when their information or functionality can be reached by other entities (e.g. an access point).

An example of element can be a box, it is a constituent of the environment where the robot resides; although its existence is known, no additional information can be drawn from it. However, if it is known that the box can provide energy to the robot (e.g. a battery), and there is information on how to get its energy, then it becomes a resource.

The abstractions can be applied to different robotics middleware platforms, an example is provided in Table II.

B. Defining Roles

The framework is based on the concept of Roles. A Role is understood as a customary function. A Role can

TABLE II

ABSTRACTIONS USED IN DIFFERENT MIDDLEWARE PLATFORMS

Abstraction	Softw. Elem.	Str. Data Elem.	Other Elem.
RTM	RT-Comp.	Ports	Conf. files, etc.
ROS	ROS Nodes	ROS Topics	Conf. files, etc.
MSRDS	DSS Services	Messages	Conf. files, etc.

be formed by various robotic and environmental objects involved into an action that will modify the robot's behavior and/or environment. The robot system can be defined by a set of Roles. Then, a Role can be declared as in Fig. 2, where "A" is the subject object performing some "action", the "action" refers to a Software Element (with its corresponding configuration) which makes the necessary processing that leads to the intended activity, and "B" is the object to which the "action" is being applied (i.e. the direct object). The subject and the direct objects are a collection of elements and resources containing information which describes them in the most complete way possible (e.g. CAD files, config. files, data, etc.). In Fig. 3 an example of a robot system represented with Roles is shown.

1) Clause types: In order to identify the direction of information flow in a Role, three types of clauses are defined as follows:

- Deliver: Subject "delivers" info. to the action and is received (explicitly or implicitly) by the direct object.
- Receive: Subject "receives" (explicit or implicit) info. from the action which is delivered by the direct object.
- Process: Combination of a receive and deliver clause. Either the subject, direct object or both can "deliver" and/or "receive" information.

Explicit information is the one that is accessible and can be stored, while implicit info. is generated as consequence of the action but is not accessible and cannot be stored. Examples of clauses are given in Fig. 4, dashed arrows represent explicit information and dotted arrows represent implicit info.

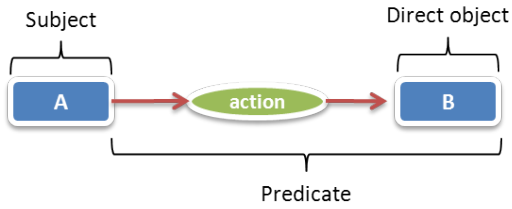

Fig. 2. Components and structure of a Role

2) Dependencies: There are cases where the performance of an action implies the performance of previous or posterior actions required to complete a known task. Two basic dependency rules are proposed:

- In order to: Defines the possible continuation Roles (which may depend on action context).
- Requires: Defines a previous Role that must be employed to perform a present one.

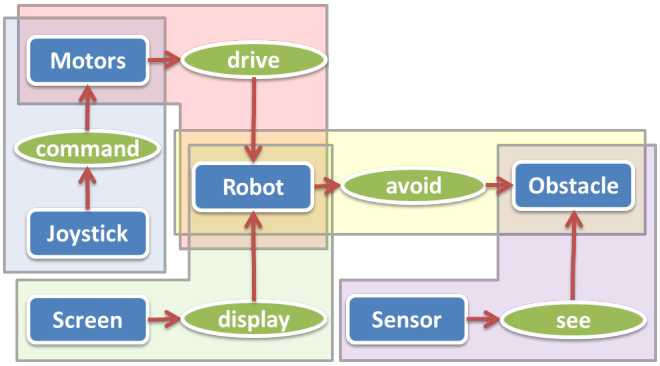

Fig. 3. Robot system represented with Roles (Roles are highlighted with the transparent shapes).

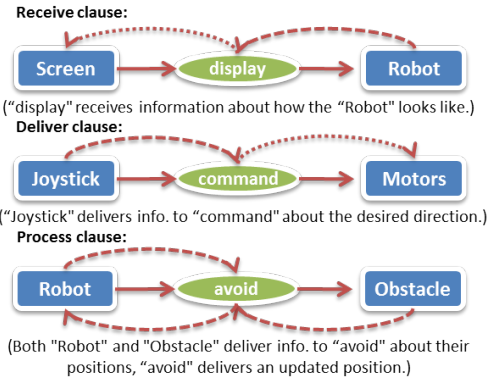

Fig. 4. Examples of Clause types

IV. IMPLEMENTATION

Details about prototype implementations of the FIERRo are described in the following subsections.

A. Using the Resource Description Framework (RDF)

An encoding standard for Roles, objects and actions is used for implementing FIERRo. The RDF standard has been selected. The RDF is a standard for encoding metadata and other knowledge on the Semantic Web [14]. In this way, structured information can be spread in a distributed and decentralized manner. By using RDF, it is possible to break down knowledge into discrete pieces, as well as making inferences from the stated facts. In RDF, resources are stated in the form of Uniform Resource Identifiers (URI); for instance, the subject and action can be URIs, while the direct object can be either a URI or a literal value. For this implementation, the RDF/XML syntax specification was used. As for the query engine, a database supporting the SPARQL language or equivalent was suggested.

B. Diagram construction method

The following method is proposed for constructing a system model diagram by using FIERRo:

1) List the objects that compose the robot and which are present in robot's environment (physical and virtual).

2) From the listed objects, list their composing elements, if any. If an element has composing elements too, list them as well.
3) From all the listed elements, list the ones that are resources, if any.
4) From the listed resources, identify the Software Elements (if any) and list the actions performed by them.
5) Make associations among the objects using the listed actions and store them as Roles ("A action B").
6) From all the listed elements, list the ones that represent data which may be transferred among objects and describe it in Structured Data Elements form.
7) For the listed objects, construct a hierarchical tree diagram using their composing elements. The same applies for the elements that have composing elements. In both cases, don't include the Software Elements.
8) For the listed actions, construct a hierarchical tree diagram using their related Software Elements.
9) Associate the Structured Data Elements with the objects and/or actions (that are interested in such data) by putting them on their respective hierarchical trees.
10) Serialize the Roles, objects and actions into RDF files.
11) Merge all RDF files to build the entire system model.
12) In case of model update/refinement, iterate the method.

The method can be implemented by an algorithm that automates this process in a computer program. A figure showing the key parts for building the diagram is provided in Fig. 5. The rounded rectangles represent objects, ovals represent actions, darker rectangles represent Software Elements, lighter rectangles represent Structured Data Elements, and white rectangles can represent other elements such as files, properties or raw data. As for the connections, arrowed connectors represent action flow and plain connectors represent a relationship between an object or an action with their respective elements. Since one of the purposes of this framework is providing the modeling guidelines for improving usability, level of detail among models may change from modeler to modeler. Despite this, models can be further detailed on the fly by other modelers that may want to state or know more specific information about the system through the definition and merging of Roles, and they can use the same query mechanism independently from the detail level or abstractions used. Additionally, when using RDF files, it becomes compatible with the Semantic Web.

V. EXAMPLE APPLICATIONS

In order to validate the proposed framework (and its implementation), it was applied into three example applications.

A. Example 1: Independent robot systems

In this example, two independent robot systems available at our laboratory were modeled using the FIERRo. They were the TITAN XIII and GRYPHON robot systems shown in Fig. 6. Each of the robots has different purposes and architectures (software and hardware). TITAN XIII is a quadruped robot intended for gait experimentation on irregular terrains

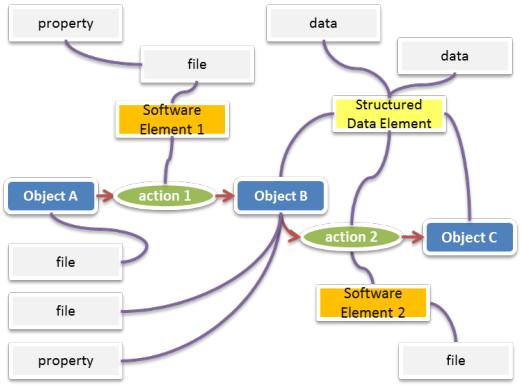

Fig. 5. Diagram using Roles.

[15]. GRYPHON is a robotic arm equipped with various sensors and mounted on a buggy intended for humanitarian demining operations [16]. The resulting diagrams for each of the systems are shown in Fig. 7 and Fig. 8. The framework provides the freedom of choosing the level of detail, in this example the TITAN XIII system model has a greater level of detail than the GRYPHON system model.

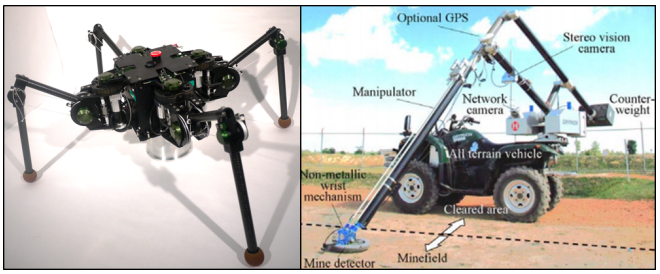

Fig. 6. Robots for modeling examples (left: TITAN XIII,right: GRYPHON)

B. Example 2: Working with the system

For this example, the TITAN XIII system was used. This robot system can be described by the following relevant items as shown in Fig. 7:

- Executable program containing a GUI for controlling the robot (GUI.bat).
- Executable program that reads a Gamepad for controlling the robot (GameCtrl.exe).
- Simulation (using V-REP simulator [17]) (vrep.exe).
- Microcontroller program (uController).
- Serial ports (using virtual and real ports) (COM2, COM4, COM5).
- Configuration files (config.ini).
- Project notes (describing some configurations) (parameter.txt, conf.txt).
- CAD Files (scene.stl, TITANXIII.stl).
- Source code files (main.cpp, control.c, main.py, TITANXIII.ttt).
- Data (Position: Leg1, Leg2, Leg3, Leg4).

The developer needs to perform certain maintenance activities, such as editing files, running programs, configuring

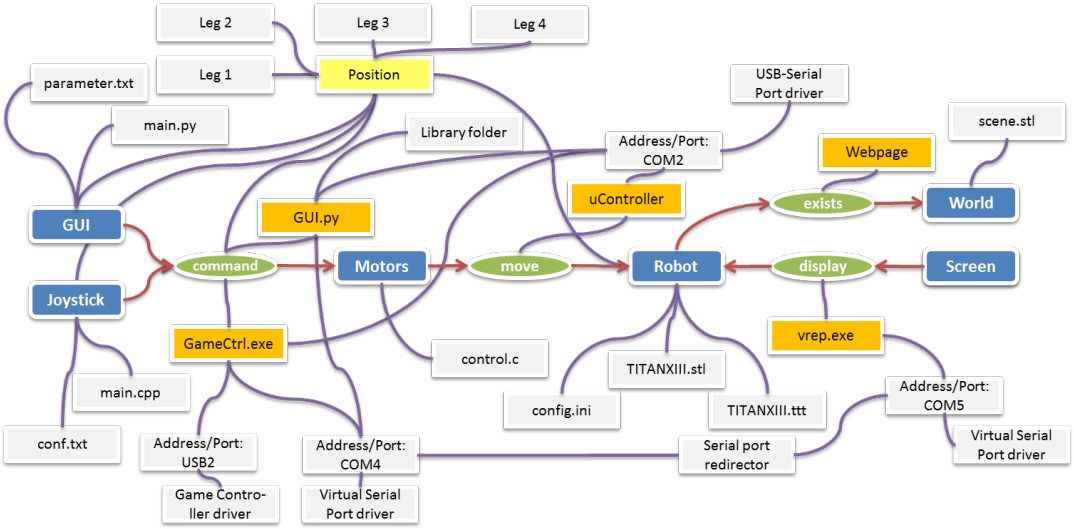

Fig. 7. TITAN XIII system model

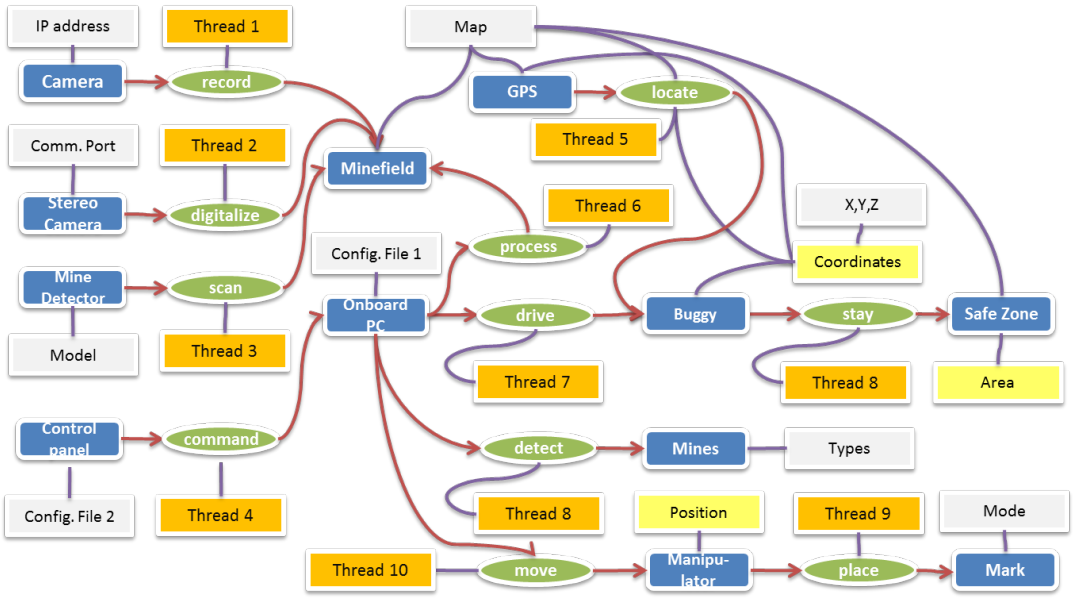

Fig. 8. GRYPHON system model

them and troubleshooting the system. For this, the developer needs to organize the project in a way that becomes easier to perform the previously mentioned activities. Here two cases were considered: working without FIERRo and with FIERRo, where we are stating the hypotheses for each of the cases that will serve as starting points for the future experiments:

1) Working without FIERRo: Without the proposed framework we expect a developer, who is not robot specialist, to have a difficult time building a model which integrates all the details of the robot system. It is expected from them to do something that resembles the following: creating a series of folders to place the files, trying to use a development framework (e.g. Eclipse IDE with some additional plugins) for organizing and manage the project, or taking a look at one of the specialized solutions (e.g. RT-Middleware and ROS). Our hypothesis for this part is that only one of those solutions may not be enough as discussed previously, and probably the users will end combining by themselves the available solutions in the best of the cases, where the resulting usability of the project is poor.

2) Working with FIERRo: By using FIERRo, the process for modeling the system will be simplified and will have the merit of identifying the objects and their actual resources (e.g. the path of an executable file, an ftp or http address, a port, etc...), letting other resources and modelers locate them and make use of them with less detours. The hypothesis here is that less time (measured in seconds) and effort (measured in steps) are required for modeling the system by using FIERRo than without using it, increasing usability.

C. Example 3: Simulation of a Graphical User Interface (GUI) to access parts of the system

Once the system was modeled using FIERRo like in the example 1, a simulated GUI was made using the generated RDF files, which were merged using an RDF engine. Such GUI displays the diagram that represents the model as in Fig. 7. Since all the resources in the model point to some sort of resource, when clicking with the mouse on one of them, the pertinent file, executable program, or webpage is opened, giving the user access to such items. Moreover, through the GUI, data can be queried using an SPARQL engine. Query format is in the form of "Obj. A + relation + Obj. B", where any of them can be the item to list, e.g.:

- "X" command Motors = Who/What commands the Motors?
  Answer: "GUI" and "Joystick"
- Joystick command "X" = Who/What is the Joystick commanding?
  Answer: "Motors"
- command resource "X" = Which are the resources of command?
  Answer: "GUI.bat", "GameCtrl.exe" and "Position"

A refined version of the simulation is now being developed in order to test the hypotheses stated in example 2.

VI. CONCLUSION

In this article, we have discussed usability issues regarding modern development and integration tools for service robots. Projects like RobotDB, Semantic Robot Space, RoboEarth, AutomationML and robotics middleware solutions were studied and compared in order to get an image of the current situation of their usability as development/runtime tools. The usability of such tools needs to be improved; developers and integrators need to access and work with service robots (and their components) with the least effort possible. We have proposed the Framework for Integration of Elements and Resources by Roles (FIERRo) as an attempt for increasing the usability of the development and integration tools for service robots, and for performing usability tests in the future. Description of example applications include: modeling of two independent robot systems, system development using and not using the FIERRo (hypotheses were set for the possible outcome), and a simulated GUI.

The usability problem affects the introduction pace of service robots into the real world. We have the belief that by studying the development of a framework targeting the application development of service robots, that increases the usability for the developers and integrators (especially for the ones that are non-software professionals), has a social impact because these kind of robots can be introduced at an accelerated pace, and at the same time the quality of the applications can increase, since greater time will be spent on developing the actual application rather than dealing with usability problems of the employed tools, platforms or frameworks. Thus, the quality of life of people can be increased by using well-developed service robot applications.

Future works include framework improvements, experiments to quantify the usability, implementing the framework on the Intelligent Cross-Platform Interface (ICPI) [18] (interface for connecting and managing various robotics middleware platforms), and defining a higher level hardware-oriented framework, which automates software management, targeted for non-software professionals.

ACKNOWLEDGMENT

This work was partially supported by JSPS KAKENHI Grant Number 25303012. The first author acknowledges support from Instituto de Innovación y Transferencia de Tecnología (I^2T^2), Consejo Nacional de Ciencia y Tecnología (CONACYT) and Roberto Rocca Education Program.

REFERENCES

[1] W. Smart, "Is a Common Middleware for Robotics Possible?", IROS 2007 workshop on Measures and Procedures for the Evaluation of Robot Architectures and Middleware (2007).

[2] N. Mohamed, J. Al-Jaroodi and I. Jawhar, "Middleware for Robotics: A Survey", 2008 IEEE International Conference on Robotics, Automation, and Mechatronics, pp. 736-742 (2008).

[3] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku and Y. Woo-Keun, "RT-middleware: distributed component middleware for RT (robot technology)", 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp.3933-3938 (2005).

[4] M. Quigley et al., "ROS: an open-source Robot Operating System", ICRA Workshop on Open Source Software (2009).

[5] Microsoft Robotics: Microsoft Robotics Developer Studio 4. http://www.microsoft.com/robotics/. Accessed on Feb. 2013.

[6] A. Elkady and T. Sobh, "Robotics Middleware: A Comprehensive Literature Survey and Attribute-Based Bibliography", Journal of Robotics, vol. 2012, Jan. 2012, Article ID 959013, 15 pages.

[7] A. Juarez, J. Hu and L. Feijs, "RoboDB: An Application of Semantic Web Technologies to Robotics", Eindhoven University of Technology, (2011).

[8] M. Jang, J. Sohn and Y. Cho, "Building Semantic Robot Space based on the Semantic Web", 16th IEEE International Conference on Robot & Human Interactive Communication, pp.499-504 (2007).

[9] M. Tenorth, A. Perzylo, R. Lafrenz and M. Beetz, "The RoboEarth language: Representing and Exchanging Knowledge about Actions, Objects, and Environments", 2012 IEEE International Conference on Robotics and Automation (ICRA), pp.1284,1289 (2012).

[10] AutomationML: Home. https://www.automationml.org. Accessed on February 2013.

[11] J. Persson et al., "A Knowledge Integration Framework for Robotics", Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK) , pp.1068-1075, (2010).

[12] R. Bischoff et al., "BRICS - Best practice in robotics, Robotics (ISR)", 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK), pp. 968-975, (2010).

[13] W3C: Usability - ISO 9242 definition. http://www.w3.org/2002/Talks/0104-usabilityprocess/slide3-0.html. Accessed on January 2013.

[14] W3C: RDF Primer. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/. Accessed on February 2013.

[15] S. Kitano, G. Endo and Hirose, S.; "Development of Light Weight Sprawling-type Quadruped Robot TITAN-XIII and its Dynamic Walking", Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (2013), in-press.

[16] E. Fukushima, M. Freese, T. Matsuzawa, T. Aibara and S. Hirose, "Humanitarian Demining Robot Gryphon: Current Status and Objective Evaluation", International Journal on Smart Sensing and Intelligent Systems, vol.1(3) pp. 735-753, (2008).

[17] Coppelia Robotics: V-REP virtual robot experimentation platform. http://www.coppeliarobotics.com/. Accessed on May 2013.

[18] A. Ceron Lopez and E. Fukushima, "Proposal of Intelligent Cross-Platform Interface for Robotics Middleware", Proceedings of the 2012 IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems, pp.382-387 (2012).