An Intelligent Application Development Platform for Service Robots

Y. Sugawara, T. Morita, S. Saito and T. Yamaguchi

Abstract—**Robots should adjust their actions to suit the surrounding situation. This requires robots to be able to interchange signals to symbols and symbols to signals. We propose an intelligent application development platform based on the Service Oriented Architecture (SOA) and Semantic Web Service (SWS). Our platform has five layers: Service, Process, Module, Knowledge and Data. The first three layers are implemented by using OWL-based Web Service Ontology (OWL-S). In the Module layer, there are two kinds of module: modules for action and modules for recognition. By combining these modules, this architecture can handle symbols and signals. In this study, we applied our method to RobotCafe, which is a cafe where robots work as waiters, to enable the robots to change their greetings to suit the situation when customers come and to assist the services of the cafe.**

I. Introduction

Service robots are playing an increasing role in daily life and various fields. However, due to their restricted functions and movements, they can only undertake specific services; one robot cannot easily handle many services. Therefore, robots should cooperate with each other to provide services. This cooperation is called Multi-Robot Coordination (MRC), and we approached this issue by applying SWS for MRC[1].

However, if robots cannot sense the surrounding situation, they cannot provide good service like a human, even if they cooperate with other robots. So, to make better service robots, robots should be able to integrate symbols and signals in order to adjust their activities dynamically by interchanging symbols to signals and signals to symbols.

In this paper, we propose an intelligent application development platform based on SWS and OWL-S in order to complete a task given by a user by coordinating with other robots. By using our platform, users can create robot services easily by combining multiple software modules. Our platform has five layers: Service, Process, Module, Knowledge and Data. The first three layers are implemented by using OWL-S. Processes of OWL-S have inputs and outputs, so a process can easily be combined with other processes by binding a process's input to another's output. In the Module layer, there are two kinds of module: modules for action and modules for recognition. By combining these modules, this architecture can handle symbols and signals. This case study of RobotCafe shows how our proposed platform can be used to easily construct a service that can be changed dynamically to suit the situation.

Y. Sugawara and S. Saito are with the Graduate School of Science and Technology, Keio University, Kanagawa, Japan (phone: +81-90-7900-9236; e-mail: {y.sugawara, shunta.saito}@ keio.jp).

T. Morita and T. Yamaguchi are with the Faculty of Science and Technology, Keio University, Kanagawa, Japan (e-mail: {t_morita, yamaguti}@ae.keio.ac.jp).

II. Related Works

A. The Ubiquitous Network Robot Platform

The ubiquitous network robot (UNR)[2] is a fundamental technology for providing services by cooperating with robots, smartphone applications and environment sensors via a network. This aims to support human activities in various situations such as shopping malls, hospitals, homes and so on. The UNR platform is public open source software which helps users to make UNR systems. This study focuses on how robots collaborate on tasks with smartphone applications and environment sensors.

B. RoboEarth

RoboEarth[3] provides an infrastructure for sharing knowledge between robots. In this context, knowledge includes software components, environmental maps, knowledge about tasks and object recognition models. In RoboEarth, robot hardware, software and performance can be written in a form that software can read by using the Semantic Robot Description Language, which is based on the Web Ontology Language (OWL). RoboEarth focuses on not MRC but sharing knowledge with robots; just a single robot deals with tasks in RoboEarth.

C. OWL-S

OWL-S[4] is an ontology for services which is supported by W3C. Using OWL-S makes it possible for software agents to discover, invoke, compose and monitor web services. OWL-S has three main parts: the service profile, the service model and the service grounding. The service profile describes the outline of a service, the preconditions for a service, and the effects of a service. The service profile is used to advertise and discover services. The service model gives a detailed description of a service's operation. The service grounding provides details on how to interoperate with a service via messages, and relates OWL-S of a service to the web service's WSDL.

These previous works show how easily robot services can be made; our research focuses on this issue.

III. System Architecture

In order to make robot services easily, we propose an architecture which has five layers, and which allows users to make robot services easily by just connecting components.

A. System Architecture of Intelligent Application Development Platform

This section outlines our proposed system architecture. Figure 1 shows the hierarchy of the intelligent application development platform, which consists of five layers.

- **Service layer:** This layer is a task in the real world. Users can make a robot service by using a service or a combination of some services. A task is realized by a sequence of a number of processes according to a workflow.
- **Process layer:** This layer means a subtask composing a task in the real world. Processes are realized by some robots and some sensors. Each process has a flow which consists of some modules. Processes from more than one service can be used as a component that implements the service.
- **Module layer:** This layer is a minimum software module, which means a robot's activity or a sensor's recognition. Modules are divided into two kinds: modules for action and modules for recognition. Modules for action are software modules that robots perform. Modules for recognition are software modules for sensors that recognize signal information such as age, facial expression, poses and objects.
- **Knowledge layer:** This layer consists of some ontologies and rules which are used by modules in order to acquire knowledge about tasks, circumstances and so on. For example, in this layer, there are the Japanese Wikipedia Ontology (JWO)[5], OWL-S, domain ontologies and domain rules. JWO is a general ontology made from Wikipedia. JWO is used in dialog with users. The details of JWO are described in [6]. OWL-S is used in order to define services, processes, and modules. Domain ontologies define the input and output class of services, processes and modules. Domain rules can change robot activities based on the situation.
- **Data layer:** The data layer contains data used for module processing. For example, the data layer contains environmental maps, instance data, a gesture database and dialog history.

To realize this architecture, OWL-S is used. Modules are defined as atomic processes in OWL-S, while services and processes are defined as composite processes. Each service, process and module has inputs and outputs. Inputs and outputs correspond to classes of domain ontologies. Users can combine modules, processes and services with each other by binding before one's output to the next one's input. Some modules for action use domain rules written in the Semantic Web Rule Language (SWRL)[7] in order to modify a robot's activities according to the situation. SWRL is a rule language based on OWL, and enables domain ontologies to make new triples when conditions are satisfied by writing If-Then rules. This realizes the integration of symbols and signals.

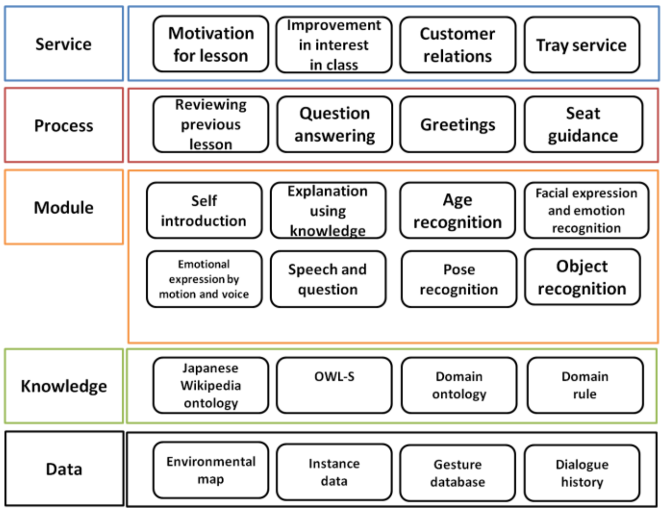


Figure 1. The hierarchy of the intelligent application development platform

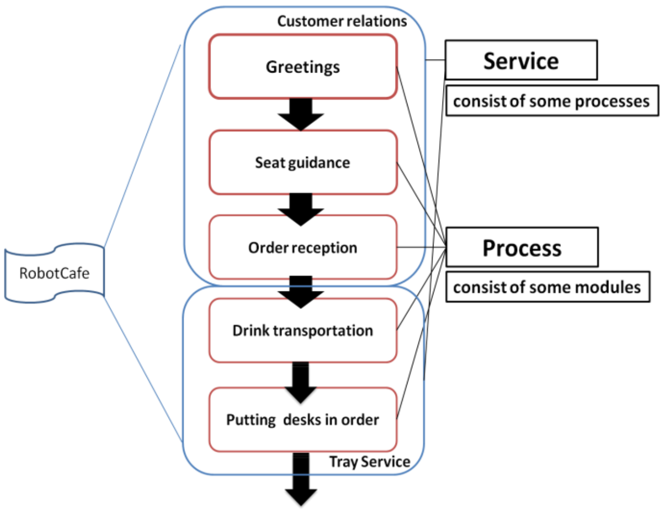


Figure 2. Outline of RobotCafe

B. Case study of Robot Service

This section explains in detail the proposed architecture based on a case study of the robot service: RobotCafe. In RobotCafe, we use a robot named "NaoTorso", which has an upper body of the humanoid robot "Nao" and a mobile robot "Turtlebot2". Figure 2 shows an overview of RobotCafe. RobotCafe has two services: customer relations and tray service. The output of the former service binds to the input of the latter.

Each service has several processes and the output of the previous process binds to the input of the next process. Each process has a sequence of some modules. In this paper, we focus on the process of greetings in customer relations. The greeting process has nine modules as shown in Fig. 3, where rectangles, rounded rectangles and ovals represent the modules comprising the greeting process, arrows mean flows, and labels of arrows mean the input and output of the modules. In the blue rounded rectangles, the signals sensed by cameras are changed to symbols such as age and the kind of object. In a red

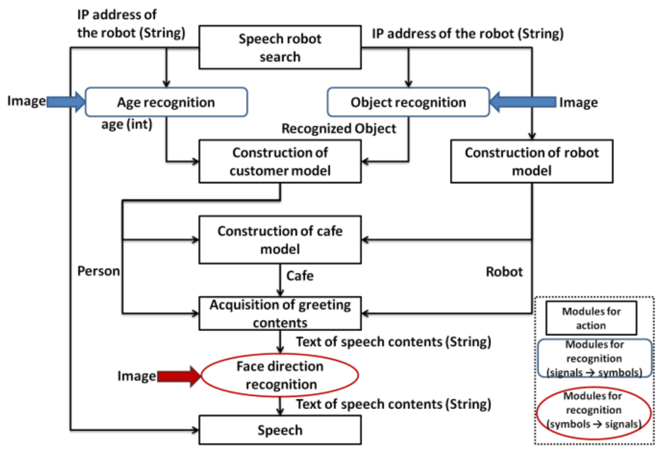

IP address of the robot (String)
Speech robot search
IP address of the robot (String)
Image
Age recognition
Object recognition
Image
age (int)
Recognized Object
Construction of customer model
Construction of robot model
Construction of cafe model
Person
Cafe
Robot
Acquisition of greeting contents
Text of speech contents (String)
Image
Face direction recognition
Text of speech contents (String)
Speech
Modules for action
Modules for recognition (signals → symbols)
Modules for recognition (symbols → signals)

Figure 3. Details of the greeting process

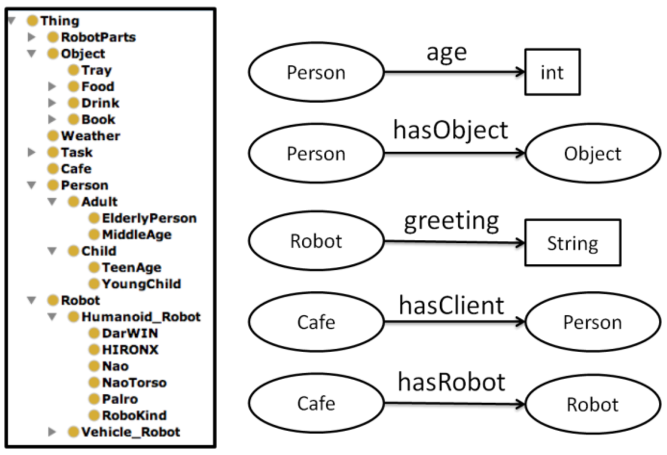


Figure 4. Class hierarchy and properties of RobotCafe ontology

oval, symbols such as face direction are changed to signals which go forward to the next module.

In the greeting process, a speech robot search is done first and returns the IP address of the searched robot. According to this IP address, the robot performs age recognition and object recognition. In age recognition, the robot estimates the customer’s age. In the age recognition module, we use the NAOqi API[8] of Aldebaran Robotics to enable the robot to estimate the customer’s age. In object recognition, the robot recognizes the objects which a customer has. This method can recognize 200 objects such as umbrellas and suitcases. Then, in the module which constructs the customer model, a person model is made according to rules for judging the type of person based on age by using the estimated age and the recognized objects. Simultaneously, robot models are constructed in the “Construction of robot model” module based on the result of the speech robot search method. By using person models and robot models, the cafe model is constructed. In the “Acquisition of greeting contents” module, the contents of greeting speech are determined by referring to the cafe model and the rules for judging the contents of greeting speech. Then, in the “Face direction recognition” module, the processing waits until the customer’s face is directed at the robot. Finally, in the speech module, the robot searched in the first module speaks the content of the greeting speech by using speech synthesis to convert from text to voice.

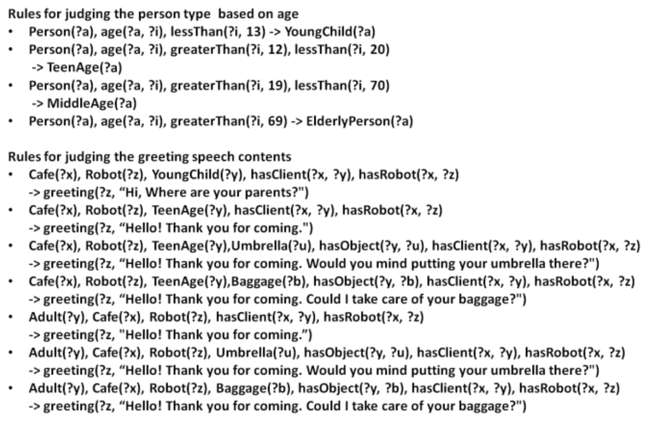
In this flow, the class hierarchy and properties of the RobotCafe ontology shown in Fig. 4 are used as inputs and outputs of modules. Related to the greeting process, the Person class, Object class, Robot class and Cafe class are determined in this ontology. The Person class has subclasses such as Adult class and Child class, which are used to judge the person type of the customer. The Object class has subclasses such as Umbrella class and Baggage class, which are used to determine the greeting comment based on the customer’s belongings.

The Person class has two properties: age property and hasObject property. The age property’s domain is the Person class and its range is int type. The hasObject property’s domain is the Person class and its range is the Object class. The Robot class has the greeting property, which represents the contents of greeting speech. The greeting property’s domain is the Robot class and its range is String type. The Cafe class has two properties: hasClient property and hasRobot property. The hasClient property represents the relationship between the cafe and the client, while the hasRobot property represents the relationship between the cafe and the robot which is in the cafe. The hasClient property’s domain is the Cafe class and its range is the Person class. The hasRobot property’s domain is the Cafe class and its range is the Robot class.

Figure 5 shows a part of the SWRL rules for RobotCafe. In Fig. 5, there are rules for judging the person type based on age and rules for judging the greeting speech contents. For example, in the former rules, there are rules that a person has the YoungChild class when the person is 12 years old or younger, and has the TeenAge class when the person is 13 to 19 years old. In the latter rules, there are rules such as that a robot says “Hello. Thank you for coming.” when the customer belongs to the TeenAge class or Adult class and that a robot says “Hi. Where are your parents?” when the customer belongs to the Child class.

In addition, there are rules related to the customer’s belongings in rules for judging the greeting speech contents. For instance, there are the rules that a robot says “Would you mind putting your umbrella there?” in addition to the usual comments when the customer has an umbrella, and rules that a robot says “Could I take care of your baggage?” in addition to the usual comments when the customer has baggage. Therefore, by using these rules, the greeting comments can be tailored to the situation such as the customer’s age and belongings.

Table 1 shows the types, names and details of inputs and outputs of modules composing the greeting process. In this table, modules whose input or output class is null such as the IP (Internet Protocol) address of the robot and recognized age have no RDF graph input or output but basic data type input or output.



Rules for judging the person type based on age

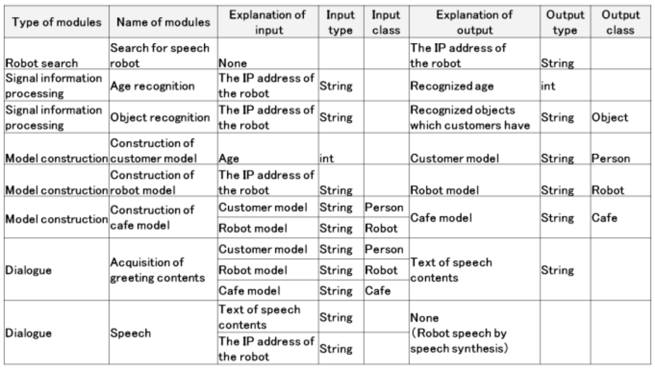
- Person(?a), age(?a, ?i), lessThan(?i, 13) -> YoungChild(?a)
- Person(?a), age(?a, ?i), greaterThan(?i, 12), lessThan(?i, 20)
  -> TeenAge(?a)
- Person(?a), age(?a, ?i), greaterThan(?i, 19), lessThan(?i, 70)
  -> MiddleAge(?a)
- Person(?a), age(?a, ?i), greaterThan(?i, 69) -> ElderlyPerson(?a)

Rules for judging the greeting speech contents

- Cafe(?x), Robot(?z), YoungChild(?y), hasClient(?x, ?y), hasRobot(?x, ?z)
  -> greeting(?z, "Hi, Where are your parents?")
- Cafe(?x), Robot(?z), TeenAge(?y), hasClient(?x, ?y), hasRobot(?x, ?z)
  -> greeting(?z, "Hello! Thank you for coming.")
- Cafe(?x), Robot(?z), TeenAge(?y),Umbrella(?u), hasObject(?y, ?u), hasClient(?x, ?y), hasRobot(?x, ?z)
  -> greeting(?z, "Hello! Thank you for coming. Would you mind putting your umbrella there?")
- Cafe(?x), Robot(?z), TeenAge(?y),Baggage(?b), hasObject(?y, ?b), hasClient(?x, ?y), hasRobot(?x, ?z)
  -> greeting(?z, "Hello! Thank you for coming. Could I take care of your baggage?")
- Adult(?y), Cafe(?x), Robot(?z), hasClient(?x, ?y), hasRobot(?x, ?z)
  -> greeting(?z, "Hello! Thank you for coming.")
- Adult(?y), Cafe(?x), Robot(?z), Umbrella(?u), hasObject(?y, ?u), hasClient(?x, ?y), hasRobot(?x, ?z)
  -> greeting(?z, "Hello! Thank you for coming. Would you mind putting your umbrella there?")
- Adult(?y), Cafe(?x), Robot(?z), Baggage(?b), hasObject(?y, ?b), hasClient(?x, ?y), hasRobot(?x, ?z)
  -> greeting(?z, "Hello! Thank you for coming. Could I take care of your baggage?")

Figure 5. SRWL rules for RobotCafe

Table 1. Details of modules comprising the greeting process



Type of modules	Name of modules	Explanation of input	Input type	Input class	Explanation of output	Output type	Output class
Robot search	Search for speech robot	None			The IP address of the robot	String	
Signal information processing	Age recognition	The IP address of the robot	String		Recognized age	int	
Signal information processing	Object recognition	The IP address of the robot	String		Recognized objects which customers have	String	Object
Model construction	Construction of customer model	Age	int		Customer model	String	Person
Model construction	Construction of robot model	The IP address of the robot	String		Robot model	String	Robot
Model construction	Construction of cafe model	Customer model	String	Person	Cafe model	String	Cafe
		Robot model	String	Robot			
Dialogue	Acquisition of greeting contents	Customer model	String	Person	Text of speech contents	String	
		Robot model	String	Robot			
		Cafe model	String	Cafe			
Dialogue	Speech	Text of speech contents	String		None (Robot speech by speech synthesis)		
		The IP address of the robot	String				

Figure 6 shows the detailed processing of the acquisition by the greeting contents module. This module inputs RDF graphs merging the Person model, Robot model and Cafe model. In this module, rules for judging the greeting speech contents are used as inputs. Then, this process makes a triple which defines the greeting contents by using the greeting property and outputs the triple. Figure 6 shows an example in which the greeting content is "Hello. Thank you for coming. Would you mind putting your umbrella there?" for a customer who belongs to the MiddleAge class and has an umbrella.

As mentioned above, RobotCafe can be implemented by using the proposed architecture based on modules, processes, services, ontologies and rules and can be dynamically changed according to the circumstances.

IV. Discussion

As an example of the robot service using symbols and signals, we describe the greeting process in RobotCafe. In this process, we use a robot called "NaoTorso", which is a combination of a humanoid and the mobile robot "NaoTorso with Kobuki". This robot has an upper body of the humanoid robot "Nao" and a mobile robot "Turtlebot2". This robot is good at dialogue and stable movement.

Figure 7 shows the greeting process. When a customer comes, NaoTorso approaches the customer and speaks the greeting dialogue.

Figure 8 shows the implementation of person detection.

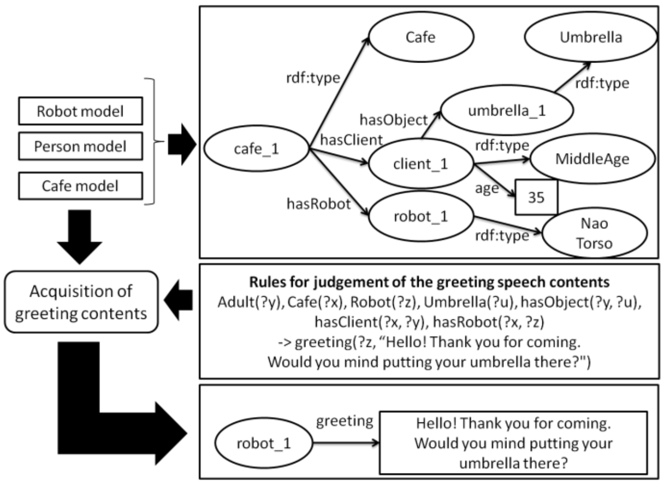

Cafe
Umbrella
rdf:type
rdf:type
Robot model
Person model
Cafe model
hasObject
umbrella_1
cafe_1
hasClient
rdf:type
client_1
MiddleAge
age
35
hasRobot
robot_1
rdf:type
Nao
Torso
Acquisition of
greeting contents
Rules for judgement of the greeting speech contents
Adult(?y), Cafe(?x), Robot(?z), Umbrella(?u), hasObject(?y, ?u),
hasClient(?x, ?y), hasRobot(?x, ?z)
-> greeting(?z, "Hello! Thank you for coming.
Would you mind putting your umbrella there?")
robot_1
greeting
Hello! Thank you for coming.
Would you mind putting your
umbrella there?

Figure 6. Acquisition processing by the greeting contents module

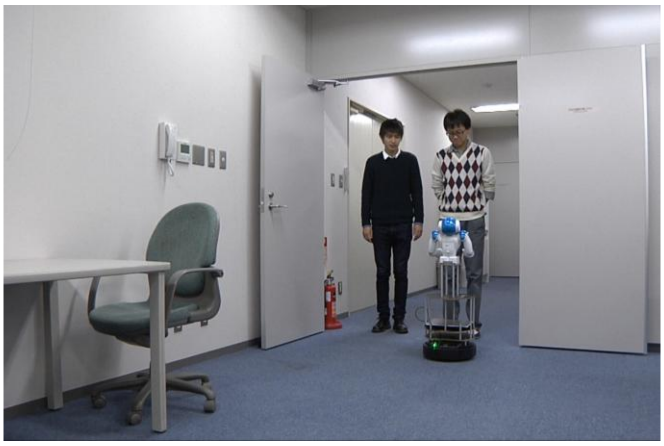


Figure 7. The greeting process

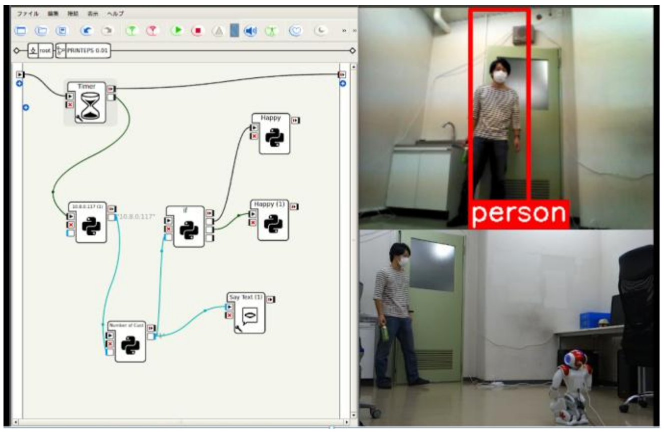


Figure 8. Implementation of person detection

The left side of Fig. 8 shows an example of person detection, which is implemented by using Choregraphe[9] of Aldebaran Robotics. In Choregraphe, boxes mean each robot activity such as saying some text, person recognition and so on. Users can create a robot service by connecting some boxes. We made RobotCafe by implementing boxes which use web services corresponding to modules in our intelligent application development platform.

The upper right side of Fig. 8 shows the result of person recognition. When a man enters the room, the robot recognizes him like this. We use *Regions with Convolutional Neural Network features(R-CNN)*[10] to extract object labels and their locations simultaneously from an input image. R-CNN preliminary learns feature extractors by training a deep convolutional neural network (CNN), AlexNet[11], on the ImageNet ILSVRC dataset[12]. In the inference stage, R-CNN firstly obtains object proposal regions by a selective search[13], then extracts features for each region by using the pre-trained deep CNN. Finally, those features are classified to one of the classes of interest or background by pre-trained Support Vector Machines (SVMs)[14]. We use R-CNN as an object detector used by NAO.

The lower right side of Fig. 8 shows the appearance of the person and the robot.

The greeting process is performed as follows, which shows the greeting comments spoken by NaoTorso according to the customer's situation.

- A child (whether with belongings or not)

  NaoTorso: Hi. Where are your parents?

- An adult without belongings

  NaoTorso: Hello! Thank you for coming.

- An adult with an umbrella

  NaoTorso: Hello! Thank you for coming. Would you mind putting your umbrella there?

- An adult with some baggage

  NaoTorso: Hello! Thank you for coming. Could I take care of your baggage?

V. CONCLUSIONS

In this paper, we proposed an intelligent application development platform based on SWS and OWL-S. This architecture enables users to integrate symbols and signals and to make robot services easily which can be tailored to the circumstances. The architecture has five layers: Service, Process, Module, Knowledge and Data. The upper layer consists of the combination of components in the lower layer. In the case study, we showed a greeting service in a cafe which can be changed according to the customer's situation.

In the future, we will create many modules for actions and modules for recognition in order to achieve different and difficult tasks. Then, we will create modules for affect/emotion processing. In addition, we will build the architecture in order to manage robots and sensors and a workflow editor which enables users to use this intelligent application development platform easily. Finally, we will do more testing in complex scenarios involving many processes.

ACKNOWLEDGEMENT

This work has been supported by JST/CREST research titled A Framework PRINTEPS to Develop Practical Artificial Intelligence.

REFERENCES

[1] Y. Mori, Y. Ogawa, A. Hikawa, and T. Yamaguchi, "Multi-robot Coordination Based on Ontologies and Semantic Web Service," in *Knowledge Management and Acquisition for Smart Systems and Services*, Gold Coast, 2014, pp. 150–164.
[2] M. Sato, K. Kamei, S. Nishio, and N. Hagita, "The Ubiquitous Network Robot Platform: Common platform for continuous daily robotic services," in *2011 IEEE/SICE International Symposium on System Integration (SII),* Kyoto, 2011, pp. 318–323.
[3] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussemann, R. Janssen, J. M. M. Montiel, A. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle, and R. van de Molengraft, "RoboEarth," *IEEE Robotics & Automation Magazine*, vol. 18, no. 2, pp. 69–82, June 2011.
[4] 4. OWL-S: Semantic Markup for Web Services, http://www.w3.org/Submission/OWL-S/#tex2html2
[5] Japanese Wikipedia Ontology, http://www.wikipediaontology.org.
[6] S. Tamagawa, S. Sakurai, T. Tejima, T. Morita, N. Izumi, and T. Yamaguchi, "Learning a Large Scale of Ontology from Japanese Wikipedia," *IEEE/WIC/ACM International Conference on Web Intelligence*, pp. 279–286, September 2010.
[7] SWRL: Semantic Web Rule Language, http://www.w3.org/Submission/SWRL/
[8] NAOqi API, http://doc.aldebaran.com/2-1/naoqi/index.html
[9] Choregraphe, http://doc.aldebaran.com/2-1/software/choregraphe/index.html
[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," *Computer Vision and Pattern Recognition*, pp. 580–587, June 2014.
[11] K. Alex, I. Sutskever, and E. H. Geoffrey, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems 25*, pp. 1097–1105, 2012.
[12] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, pp. 1–42, April 2015.
[13] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective Search for Object Recognition," *International Journal of Computer Vision*, no. 2, vol. 104, pp. 154–171, 2013.
[14] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.