Check for updates

International Journal of Advanced Robotic Systems

OPEN ACCESS ARTICLE

A Utilization Framework of Ubiquitous Resources for Service Robots Using Semantic Matchmaking

Regular Paper

Kun Qian[1,2]*, Xudong Ma[1,2], Xianzhong Dai[1,2], Fang Fang[1,2] and Bo Zhou[1,2]

1 Key Lab of Measurement and Control of Complex Systems of Engineering, Ministry of Education, China
2 School of Automation, Southeast University, Nanjing, China
*Corresponding author(s) E-mail: kqian@seu.edu.cn

Received 11 July 2014; Accepted 06 March 2015

DOI: 10.5772/60504

© 2015 The Author(s). Licensee InTech. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract

The automatic and flexible utilization of ubiquitous service resources is a key prerequisite for autonomous robots in ambient intelligent environments. In this paper, a novel component-based robotic system (CBRS) framework is proposed in order to support the utilization of task-oriented service resources. In order to increase the reusability of robot control programs and software libraries, a framework that supports semantic modelling, discovery and dynamic selection of ubiquitous services is proposed, which employs semantic services computing technology to facilitate service representations, invocation and interactions. The system architecture is described with an emphasis on the matchmaking method. A new semantic similarity metric is proposed based on an extended OWL relation definition and a matchmaking algorithm is proposed that is compared to the Requirement and Advertisement in ontology using the conceptual span similarity measure. Hence, more reliable semantic matching is achieved compared with previous methods. By developing a rosbridge-based connection management module in the runtime system, the framework is favourable for developing web browser-enabled applications. Experimental results in home-care robot applications validate the effectiveness of the proposed framework.

Keywords Component-based robotic system, Ubiquitous robotics, Semantic, Matchmaking, Similarity

1. Introduction

Although networked robotics [1, 2] has a long history, one unachieved goal from the home service perspective is an inexpensive robot that can flexibly utilize ubiquitous resources. The idea of cloud robotics [3] has provided the potential to significantly improve robots working in human environments. Compared with cloud computing, which focuses on networked sources for computation and data about objects and knowledge, we are more interested in a more general concept of ubiquitous resources. These may exist in the local working environment of robots or on the Internet, providing various functions such as object recognition, environment perception, motion control, knowledge base, actuation, etc.

As a large number of ubiquitous devices are seamlessly integrated in robotic and ambient intelligent systems [4],the heterogeneous nature of resources requires great

effort to produce a robust robotic system; the largest obstacle is the application-level software [5].

Component-basedrobotic systems (CBRS) [6, 7], such as ORCA [8], Miro [9], RT-Middleware [10]and OPRoS [11], provide a solution to this, by shifting the emphasis from robot software programming to composing the system from a mixture of components. Thus, the reusability and interoperability of the system are improved.

Automatic ubiquitous service utilization is a critical aspect for CBRS. It requires a new computing discipline that empowers ubiquitous service utilization as the basic building blocks for the development of distributed applications in environments that contain heterogeneous resources. Services computing paradigms envision a world of cooperating services where application components are assembled with little effort into a network of services that can be loosely coupled to create flexible dynamic processes and agile applications that may span computing platforms [12]. Web service technologies, such as Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP) and Universal Description, Discovery and Integration (UDDI),were initially introduced to support ubiquitous service utilization. The semantic service discovery methodology [13, 14, 15], also known as matchmaking, provides a flexible, adaptable and maintainable means of representing and discovering services. Based on the OWL-S (OWL based on an ontology language for services) [16, 17]service description ontology, Klusch [14] proposes a hybrid approach which combines logic-based semantic matching with token-based similarity metrics. iMatcher [15] is another system that performs matchmaking by combining functional property matching and non-functional property matching. However, most published works focus on the matchmaking algorithm itself rather than on providing a unified and task-oriented service utilization framework.

With regard to the matchmaking algorithms, ontology matching is a crucial issue in the domain of semantic web and data interoperability. The similarity-based matching approaches seek to discover the equal relations between entities in ontology. Granitzer et al. [18] and Yan et al. [19]offer a comprehensive introduction to and comparison of different basic matching techniques and applications. In some approaches, more information about ontology is adopted, for example, the data type and value may be used to calculate the similarity [20]. While string metric is the most-commonly used technique [21], Song [22]proposes a core word-based method for measuring similarity from the semantic level of ontology entities.

Semantic matchmaking using IOPE parameters defined in OWL-S is a possible solution to service discovery for robotic application. [23]is a classic matching algorithm that uses all output parameters of the service *Requirement* and the candidate *Advertisement*. If only one pair of parameters is un-matched, the judgment can be made that the candidate services *Requirement* and *Advertisement* are unmatched. This is a very strict rule that may lead to match failure for service matching in robotic application, in which input and output parameters of the service *Requirement* and *Advertisement* are often defined by a set of parameters with a corresponding concept defined in taxonomy.

In this paper, a novel component-based robotic system framework is proposed. Compared with other component-based robotic systems, our contribution is a framework that supports semantic modelling, discovery and dynamic selection of ubiquitous services to accomplish complex tasks. Thus, it improves the reusability and interoperability of CBRS. As a result of the proposed framework, a robot can infer control decisions and policies to utilize ambient services, rather than requiring the decisions to be pre-programmed.

Our proposed system framework is favourable in two major aspects. Firstly, it provides a complete methodology of automatic utilization of ubiquitous resources for robots, such as component encapsulation, service/requirement modelling, service discovery and service composition. Secondly, an improved matchmaking algorithm is proposed based on IOPE parameters of the services *Advertisement* and *Requirement*.We have extended the original definition of relation in OWL by adding a new type of "*hasPart*" relation. The concepts of Requirement Tree (RT) and Advertisement Tree (AT) are proposed, which are organized by the extended set of relations, and a new semantic similarity between RT and AT, namely conceptual span similarity (CSS), is defined. Based on this, we have provided a semantic matching algorithm, which deals with IO matching using the semantic similarity, and handles the PE matching problem by representing PE parameters using RDF statements, and compare a pair of RDF statements according to their *subject*, *predicate* and *object* structure.

The development and deployment of the CBRS platform system is described using ROS middleware for low-level connection management and rosbridge for non-ROS node communication. This enables the use of web browser-based applications. In addition, the system benefits from interfacing with a higher-level knowledge processing system, such as KnowRob [24], which is employed for interpreting human-scale manipulation tasks in our current system implementation.

2. System Architecture

2.1 Service discovery framework in CBRS

The framework contains service providers (resources) that advertise ubiquitous services at one end and service requesters (robots) at the other end. The resources usually represent sensors, actuators and other devices in the robotic system. Atomic components are distributed and reusable software modules that run loosely coupled and independently. They make heterogeneous devices transparent to

upper-level applications using an abstract description of services.

Figure 1 depicts the structure of an atomic service component, which comprises basic functional module, service processing flow,component execution engine and service specification profile.

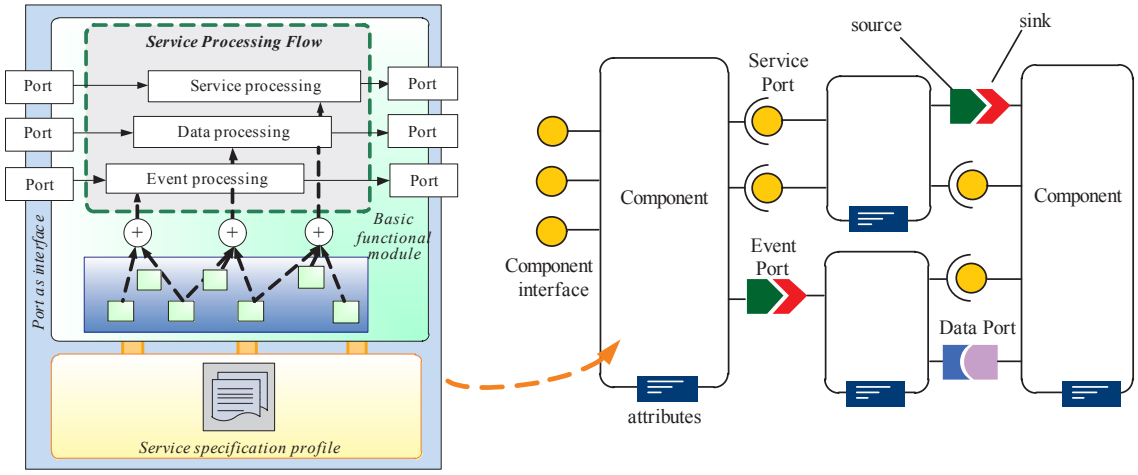

Figure 1. Structure of an atomic component and service composition

a. **Basic functional modules:** Functional modules are implemented and packaged as libraries to be invoked. In our system, typical functions of a service component include image processing, object tracking, human activity recognition, dual-arm control, localization, path planning, obstacle avoidance, etc. Most functional modules have been well developed in our previous research work [25, 27].

b. **Service processing flow:** A component sends or receives three types of information packages with others: method invocation, data and events. A specific processing flow is designed for each type of information, and components communicate with each other via the communication ports. In our system, three types of port are assigned to components, including *service port*, *data port* and *event port*.

A service port allows other components to invoke its method via a set of methods interfaces. A data port allows a component to exchange data with other components with data ports of the same data type. For example, a map server transmits map data via its data port, while a video surveillance service component transmits a stream of video data via its data port. An event port supports transmission and reception events that must be processed with high priority. For example, a vision-based human fall-down detection service component will send an alarm message via its event port to a robot inspection service component, if a fall-down event is detected during its real-time surveillance.

c. **Service specification profile**

The service specification profile of a component is a *service description document*,which is an XML-based profile that describes concrete attributes of services. More details about the service attributes will be given in Section 3.

d. **Component execution engine:** The component execution engine includes *plan execution* module, *life cycle management* module, *fault tolerance* module and *connection management* module, which will be described in detail in Section 5.

2.2 Service component container

Among the service attributes, the abstract descriptions of inputs, outputs, preconditions, effects (IOPEs) and the other non-functional properties (NFPs) are utilized for service discovery.

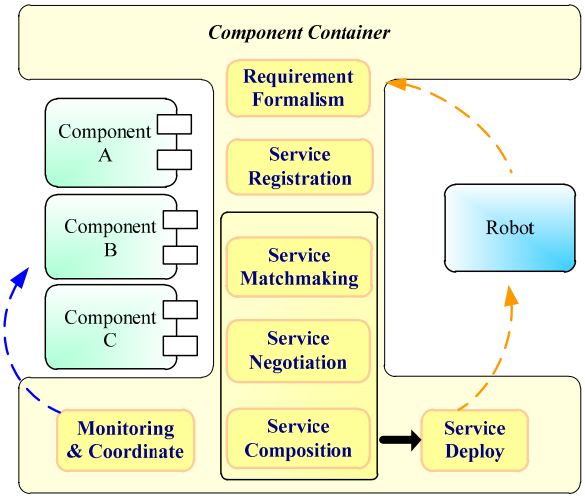

Figure 2. Component container

As shown in Figure 2, the component container stores and registers services in a repository. The component container extracts a requester's requirement and selects the appropriate service components by the matchmaking algorithm, negotiation (if any) and composition. It also monitors the execution of components and manages the selection of components from a global perspective. The selected service components for a specific task are then deployed and invoked by robots.

3. Semantic Service Modelling and Matching

3.1 Semantic modelling of ambient services and requirements

In our previous work [25], ambient and ubiquitous services in a typical robotic system are classified into four types:

- Dynamic perception components, which act as smart devices for providing people tracking, activity recognition, abnormal event detection and other perceptual functions.
- Static data components, which provide data inference about the environment, object, people and common sense for supporting the robot's decision making.
- Physical manipulation components, which consist of manipulative functions provided by different types of smart manipulator, e.g., flexible dual-arm, robotic gripper, etc.
- Method invocation components, which encapsulate various algorithm functions, such as path planning, obstacle avoidance, object recognition, etc., as software components so that they can be invoked by a robot or other components.

In order to support semantic matchmaking, we have extended the original OWL-S definition in two aspects. Firstly, we have incorporated three aspects for describing service functions, namely "OWLS Model", "SEAModel" and "QoSModel" [25],so that the service ontology (as shown in Figure 3) can better meet the requirements of robotic applications. Secondly, besides the original definition of the "*hasSubClass*" property in OWL, we have defined a new relation termed "*hasPart*" in our extended model. The "*hasPart*" property has many concrete instances, such as "*hasFamilyName*", "*hasGivenName*", "*hasStatus*", etc.

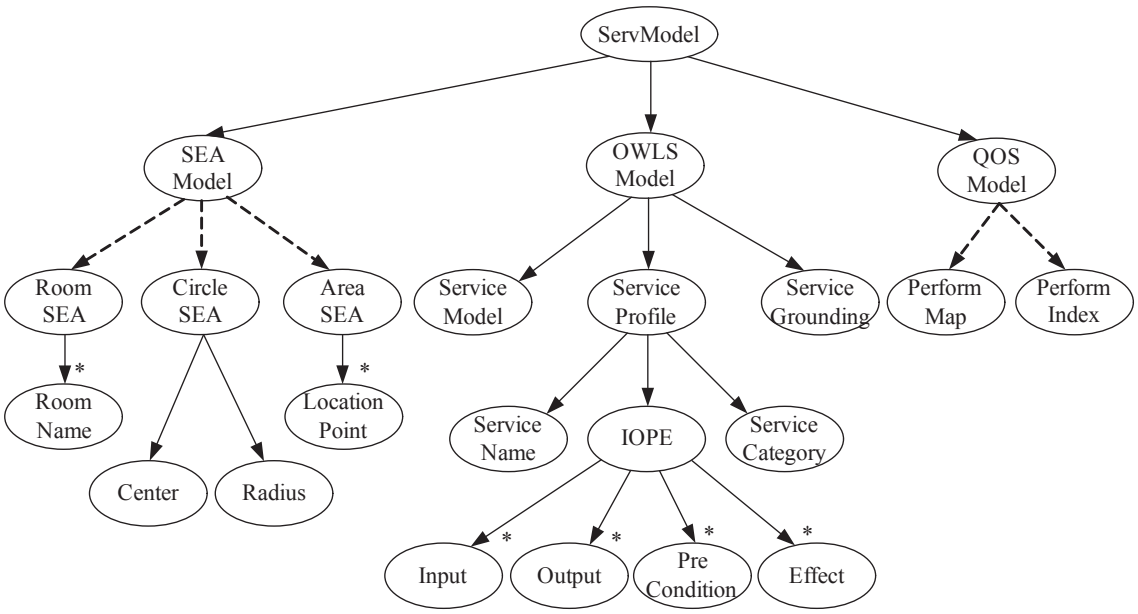

Figure 3. Semantic service ontology

A component's service is characterized using a six-tuple of <DN, SN, SC, IOPE, SEA, QoS>, in which DN stands for device name, and SN and SC stand for *ServiceName* and *ServiceCategory*, respectively. SEA describes the *Service Effective Area* of a component's service, and QoS describes the quality of service [25]. The IOPE parameters are the most important in semantic matchmaking:

- *Input/Ouput, IO*: Inputs(outputs) of a service resource are described by a set of concepts, each representing an input(output) parameter. IO concepts are represented by OWL classes and properties.
- *Precondition/Effect, PE*: RDF (Resource Description Framework) [26] is used as a general method for conceptual description or modelling of information that is implemented in web resources, using a variety of syntax notations and data serialization formats. The precondition and effect of service components are represented by RDF statements, each presented in a subject-predicate-

object structure. In each RDF triple, the subject and the object are expressed by OWL classes, and the predicate is expressed by an OWL property. The RDF statements describe the states and the changes between states caused by the P-E relation, which is the key to composing sub-actions. As an example, in Figure 4, the causal effect of the "*open a door*" action can be represented as:

E: (door, hasStatus, opened).

Table 1 shows an example of a "*LocateMobileRobot*" service component that is provided by a globally mounted camera for locating mobile robots. It uses the colour of a marker as the input and produces the robot location as the output. Figure 4 briefly depicts the IO and PE representation in our model. More details of service and requirement modelling can be found in our previous work [25].

A typical matchmaking algorithm takes an OWL-S *Query*(denoted by symbol "R") from the client as input and iterates over every OWL-S *Advertisement* (denoted by symbol "A") in its repository in order to determine a match. For the sake of clarity, we denote the *Advertisement* I/O/P/E parameters by AI/AO/AP/AE and the *Requirement* I/O/P/E parameters by RI/RO/RP/RE, respectively.

SN: LocateMobileRobot	SC: DynamicInfoService
I->O: RobotSignColor → RobotLocation	
SEA:	RoomSEA∩(∋hasRoomName "404"), CircleSEA ∩(∋hasCircleArea CircleArea∩ (∋hasCenter Center∩(∋hasX "123.45cm")∩(∋hasY "678.90cm"))∩(∋hasRadius "150.00cm")) ′
QOS:	PerformIndex∩(∋hasName "LocalizationError")∩(∋hasValue Value∩(∋hasAve "10cm")∩(∋hasMin "5cm")∩(∋hasMax "20cm"))

Table 1. Example of service model

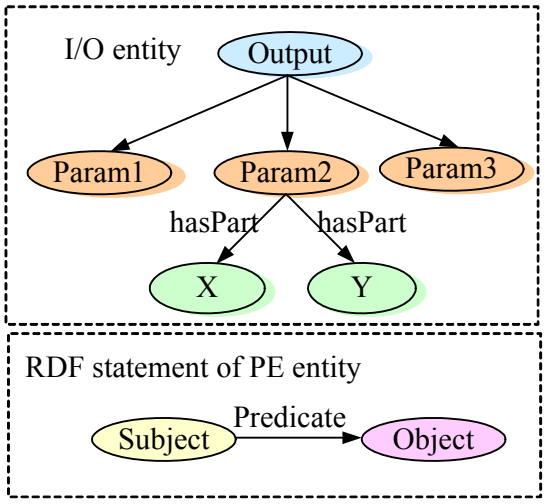

Figure 4. Formulated description of IO and PE

3.2 Extended matchmaking based on semantic similarity

Ontology matching rules are applied to discover services from a repository of registered services. Such rules seek to find semantic correspondences and the similarity between a pair of concepts in taxonomy. In this section, we will firstly define the Sem-Str similarity between a pair of single semantic entities, and then we will give the definition of conceptual span similarity between the Advertisement Tree (AT) and the Requirement Tree (RT), which provides a similarity metric to evaluate a candidate pair of Advertisement and Requirement.

a. Sem-Str similarity of a pair of entities

Semantic similarity refers to the similarity between two concepts in a taxonomy such as the WordNet or CYC upper ontology.

Denote $\{e_1, e_2\}$ as pair of semantic entities to be matched. Since the IOPE parameters are all represented by OWL classed with "*is-a*" properties or any instance of "*hasPart*" properties, the ontology of concepts as well as textual information can be utilized to measure the semantic similarity between e_1 and e_2. A general metric combining semantic with text information (denoted as "Sem-Str matching") is applied.

Definition 1 (Sem-Str similarityin taxonomy):

Given a fragment in a taxonomy such as WordNet, the semantic similarity between a pair of semantic entities $\{e_1, e_2\}$ is:

$$Sim(e_1,e_2)=\begin{cases}\dfrac{1}{1+N_{e_1e_2}} & if\ Situation1\\ \dfrac{2N_r}{N_{e_1}+N_{e_2}+2N_r} & if\ Situation2\\ Jaro-winkler(e_1,e_2) & if\ Situation3\end{cases} \quad (1)$$

Situation1:

In a special situation, such that e_1 node is a leaf of e_2 node, or their most specific common superclass is the root node, denote that $N_{e_1e_2}$ is the number of single-direction "*is-a*" links or "*hasPart*" links from e_1 to e_2, and the semantic similarity equals $1/(1+N_{e_1e_2})$. For example, this avoids the case that their similarity equals zero, e.g., $Sim(e_B, e_A)$ in Figure 5. In a special case such as $Sim(e_E, e_D)\approx 0$, since $N_{e_Ee_D}=\infty$.

Situation2:

In a general situation other than Situation1, denote that N_{e_1} and N_{e_2} are the number of "*is-a*" links or "*hasPart*" links from the corresponding entity of e_1 and e_2 to their most specific common superclass C. N_r is the number of "*is-a*" links or "*hasPart*" links from C to the root of the taxonomy. Thus, the similarity is computed as $2N_r/(N_{e_1}+N_{e_2}+2N_r)$; for example, in Figure 5 $Sim(e_C, e_E)$=0.67. The result is actually the semantic similarity proposed by Wu and Palmer [28].

Situation3:

When no taxonomy information is available, a broad edit distance metric [22] (Jaro-Winkler distance) is applied.

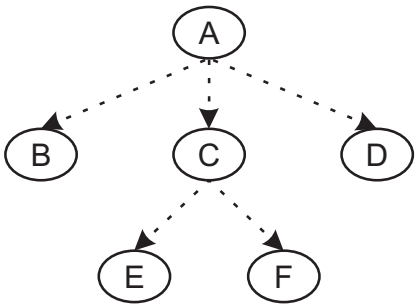

Figure 5. Illustration of the tree structure of ontology

Denote m as the number of matching characters and t as half the transportation number. Two characters are matched only when the distance is not beyond the matching window, i.e., taking a_i and b_j (i, j denotes the sequence in the string),the character from e_1 and e_2, if $a_i=b_j$ and $j-g<i<j+g$, where $g=\max(|s_1|, |s_2|)/2-1$.

$$MV=\frac{max(|e_1|,|e_2|)}{2}-1, \tag{2}$$

$$Jaro(e_1,e_2)=\frac{1}{3}\cdot\left(\frac{m}{|e_1|}+\frac{m}{|e_2|}+\frac{m-t}{m}\right). \tag{3}$$

The Jaro-Winkler distance adds a weight for a common prefix, as defined in Eq. (4). P is the length of the longest common prefix of e_1 and e_2. The coefficient $\min(P, 4)/10$ is to ensure the similarity function is greater than 1.

$$Jaro-winkler(e_1,e_2)= Jaro(e_1,e_2)+\frac{\min(P,4)}{10}\cdot\left(1-Jaro(e_1,e_2)\right) \tag{4}$$

Denote that the identified core words of e_1 and e_2 are $\{cw_1, cw_2\}$, and the complimentary words are $\{ci_1, ci_2\}$, respectively. Thus, the overall semantic similarity is the evaluation result of the weighted sum of the two parts, in which α and β are two weights that satisfy $\alpha>\beta$.

$$Sim(e_1,e_2)= \begin{cases} \alpha Sim(cw_1,cw_2)+\beta Sim(ci_1,ci_2) & cw_1\neq cw_2 \\ 1 & cw_1=cw_2 \end{cases} \tag{5}$$

b. IO matching with conceptual span similarity (CSS)

The matching between the Input and Output field is performed by matching between the Advertisement Tree (AT) and the Requirement Tree (RT).

Suppose we have an Advertisement Tree, modelled as a tree structure with a root node N^{Sup} and a set of leaf nodes $\{L_i^{Sup}\}$, $i=0, ..., K$. Similarly, we also have a Requirement Tree, modelled as a tree structure with a root node N^{Dem} and a set of leaf nodes $\{L_i^{Dem}\}$, $i=0, ..., K$. By using the above-mentioned Sem-Str matching metric with a certain threshold, we have obtained a set of matched entity pairs $[N_i^{Sup}, N_i^{Dem}]$, $i=0, ..., M$.

In a general situation where the matched entity pairs cover both leaf and non-leaf node types, the conceptual span similarity (CSS) between the Advertisement Tree (AT) and the Requirement Tree (RT) is computed according to Equation (6), in which the $Sim(\cdot)$ function takes the form of Equation (5).

$$CSS(AT,RT)=Sim(N^{Sup},N^{Dem})\prod_{i=1}^{M}Sim(L_i^{Sup},L_i^{Dem}),\quad \text{s.t. } (N^{Sup}\ \text{hasPart}\ L_i^{Sup})\cap(N^{Dem}\ \text{hasPart}\ L_i^{Dem}) \tag{6}$$

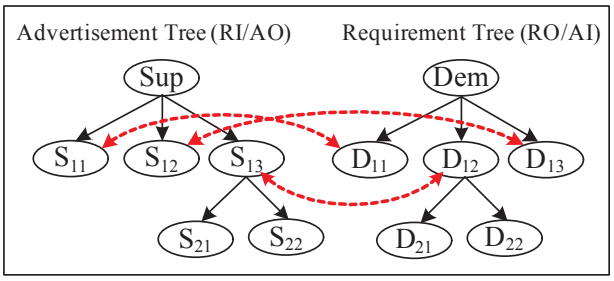

Figure 6. Matching between AT and RT

As in the example shown in Figure 6, the matched entity pairs between the AT and RT are the leaf nodes $[S_{11}, D_{11}]$, $[S_{12}, D_{13}]$ and the non-leaf nodes $[S_{13}, D_{12}]$. Thus, the conceptual span similarity of the AT and the RT equals:

$$CSS(AT,RT)=Sim(S_{11}, D_{11})Sim(S_{12}, D_{13})\cdot Sim(S_{13}, D_{12})Sim(S_{21}, D_{21})Sim(S_{22}, D_{22}) \tag{7}$$

In the special case where the root node of AT and RT matches, the conceptual span similarity of the AT and the RT can be simply computed as the $Sim(\cdot)$ function defined in Equation (5).

c. PE matching with RDF statement

The PE fields are represented by sets of RDF statements, so that the PE matching is performed between the RDF statement of an Advertisement and that of a Requirement. The PE matching makes use of the <subject, predicate, object> structure of RDF statement pairs, as in the example shown in Figure 7.

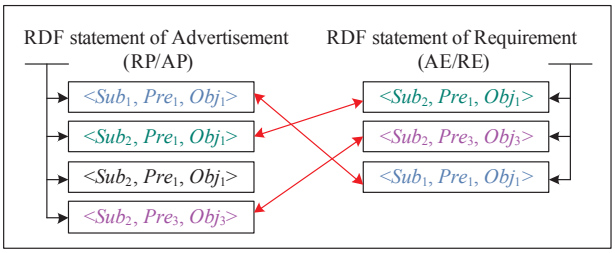

Figure 7. Matching with RDF statement

Take a pair of RDF statements R^{Sup} and R^{Dem}, representing the RDF statement of Advertisement and the Requirement, respectively. The RDF similarity is computed according to Equation (8):

$$RDFSim(R^{Sup}, R^{Dem}) = Sim(R^{Sup}.Sub, R^{Dem}.Sub) \cdot Sim(R^{Sup}.Pre, R^{Dem}.Pre) Sim(R^{Sup}.Obj, R^{Dem}.Obj) \quad (8)$$

4. Service Composition

A service composition method is applied for assembling atomic components according to their source-sink correspondences, and connecting each pair of "matched" components. In this paper, we improve the method and argue that it can also be applied to decomposing actions into a sequence of sub-actions.

The method formulates the fundamental structure of a directed graph $G(V, E)$ called a service resource graph (SRG). The graph consists of a finite set of vertices V, which represent available atomic service resources discovered by the robot, and a set of directed edges E, each of which associates a pair of atomic service resources. In this context, an edge can be linked between an atomic service resources model AM and an atomic service requirement model RM if their I-O or P-E parameters "*match*".

$$\text{MatchAccept}(AM.IO, RM.IO) \cup \text{MatchAccept}(AM.PE, RM.PE). \quad (9)$$

The service composition algorithm essentially makes use of a bidirectional breadth-first search (BBFS) [29]strategy for finding paths from an initial vertex to a goal vertex in the service resource graph. A virtual root node S_{RI} which requires RI as input is simulated to play the role of the maze entrance. Correspondingly, the exit of the maze is also simulated by a virtual node S_{RO} or S_{RE}. The matching proceeds according to the *Qualified Pre-action*/pre-service criteria and the *Qualified Post-action*/post-service criteria, as defined in [30].

When matching two concepts, four degrees of relations are defined according to an OWL-S reasoning:

- Exact(≡)
- SubClassOf/SubPropOf(⊂)
- HasSubClass/HasSubProp(⊃)
- Fail(≠)

In this paper, a semantic matching is accepted according to the threshold which is chosen as:

$$\text{MatchAccept}_{IO \cup PE}(x, y) \in threshold \quad (10)$$

in which the threshold is {"Exact", "SubClassOf", "HasSubClass"}.

By the reverse-directional PE matching originating from the requirement node S_{RE}, *Qualified Pre-actions* are revealed and assembled in sequence. As an example, in Figure 8, the service component *"PuttingDownAnObject"* is linked with *"CarryingWhileLocomoting"*. In a similar way, the four sub-steps of the action *"PuttingSomethingSomewhere"*can be connected in order, such as "*ApproachToObject*", "*PickingUpAnObject*", "*CarryingWhileLocomoting*" and "*PuttingDownAnObject*". Compared with KnowRob system, our method only requires specifying the coarse sub-actions, rather than the full nested hierarchy of a complex action. In addition, by the reverse-directional IO matching originating from the requirement node S_{RO}, qualified pre-services are connected, as in the example shown in Figure 9.

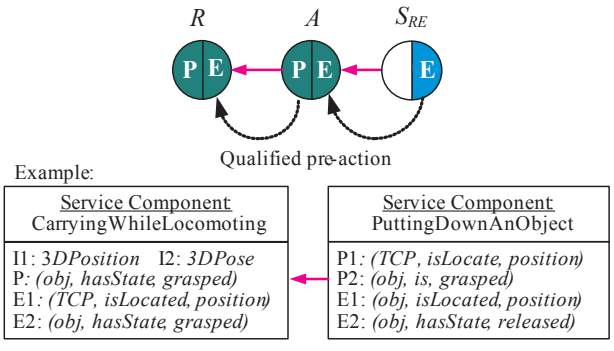

Figure 8. Qualified pre-action

The service composition algorithm begins with a loop for a bidirectional breadth-first traversal search. The forward search process inspects all the neighbouring nodes for

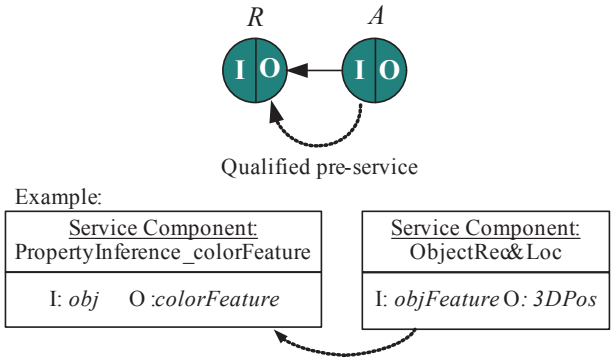

Figure 9. Qualified pre-service

finding *Qualified Pre-services* and *Qualified Pre-actions*.The backward search uses the *Qualified Post-service* and *Qualified Post-action* criteria. If a pair of matches is accepted, their similarity is measured by $Sim(e_1, e_2)$. All candidate qualified services and actions are registered and ranked with their similarity value. In case more than one connecting path is discovered, the one with the highest accumulated similarity or minimum cost value is selected as the best plan.

5. Implementation and Web Applications

In our work, a web-enabled robotic system is developed that integrates robots and ambient services for home-care applications. In the system, different robot platforms (e.g., ActivMedia Pioneer 3 DX, Peoplebot, and Robai Bi-handed Cyton 14D-2G robot) are designed as the service requesters. A group of embedded sensors are integrated, including video cameras, global cameras, a laser scanner, a Kinect sensor, and sonar sensors. These sensors work as the processing units for people detection, activity recognition or environmental perception. In addition, several Cortex-M3-based environmental sensor boards are also integrated as nodes, which measure temperature and other environmental parameters.

In the current system's implementation, a component execution and management engine is developed using an open source robot middleware ROS(robot operating system) [31], which administrates the low-level robot controllers, sensor processing, planning, navigation and manipulation. All nodes are encapsulated as modular, runtime, configurable software components, which are fed to the component container for allocation of ubiquitous resources according to the task requirement.

In the connection management module, Non-ROS nodes (e.g., smart nodes) communicate with the ROS nodes (e.g., Pioneer 3 DX robot and Cyton dual-arm robot) using rosbridge and *topic* publishing/subscribing or *service* request/response as serialized JSON objects. Figure 10 shows the structure of the application system and Figure 11 illustrates the overview of architectural design of a web-equipped ROS system.

The component execution and management engine contains the *plan execution* module, *life cycle management* module and *fault tolerance* module, as shown in Figure 11. The *life cycle management* module manages the component's lifecycle and states.The executor is activated when the component comes to its run by a thread scheduler of the engine. After a component's instance is created, it runs through a sequence of states during its lifecycle. The *fault tolerance* module detects faults or anomalies, and repairs them autonomously in order to recover from faults. When a violation is detected at a component, the tasks will be switched to other components providing similar functions, and thus the execution will proceed.

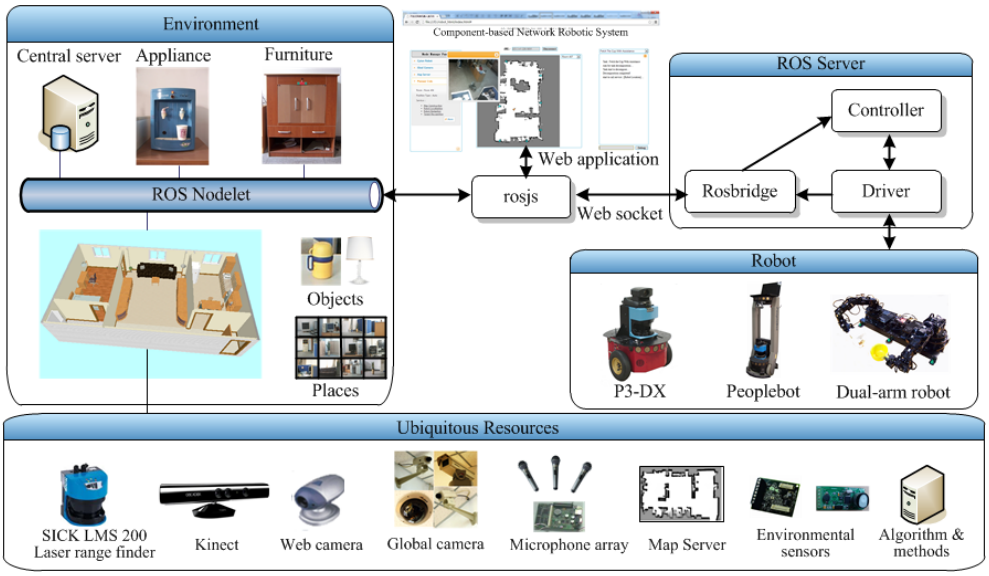

Figure 10. Application system example

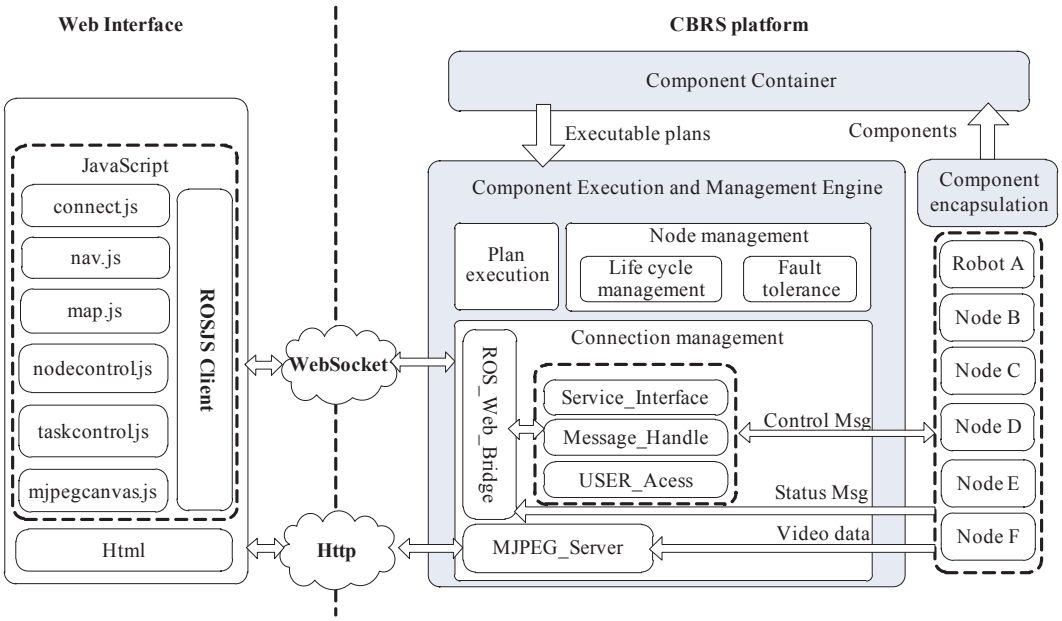

Figure 11. Using ROS to implement the connection management module

The *connection management* module provides a communication interface to allow robot components to communicate across networks. The module allocates a connector to establish the connection between two interacting ports. The Service_interface and Message_interface are designed to be service ports and event ports.MJPEG_server is designed for data ports that transmit video streams.Rosjs handles rosbridge connection to a web browser application through HTML5 websockets.Hence,a user-friendly web interface is implemented.

Moreover, the system benefits from the ROS-based connection management module in the interface with the higher-level knowledge processing system.By encapsulating a manipulative function as an atomic action-type component, the system is also favourable for automatic decomposition of human-scale manipulation tasks, in which the relative pair-wise ordering among sub-actions can be revealed by the service composition procedure. Hence the method is beneficial for less manual work of specifying detailed task instructions.

6. Evaluation

6.1 Semantic service matchmaking

We firstly show the example of semantic service matchmaking in a scenario that involves navigation of the service robot to different places with the support of global surveillance cameras for globally localizing robots in a room environment. It is assumed that the robot is not equipped with any sensors (e.g., sonar, laser) for self-localization. Hence the robot firstly needs to discover the camera network resource in the environment for localizing its self-position by detecting a colour marker mounted on the robot.

To demonstrate the performance of semantic service matchmaking, we focus on the robot's requirement of subscribing its self-position from other service resources. To be more specific, Table 2 shows the formalization of the corresponding robot requirements. Assume that we have a set of available candidate service providers, as listed in Table 3, and a fragment of the ontology used for the semantic matching example, as shown in Figure 12. According to the proposed semantic service matchmaking algorithm that compares the IO parameters of the Advertisement and Requirement, the similarity scores can be computed as listed in Table 3, in which a best match can be discovered (marked with shadow in the table).

In our experiment, we generated a testing database including 20 requirements and 100 service components. We have selected some testing pairs of Advertisements and Requirements in which their IOPE entities were not given with exactly the same strings. For example, "*Position*" and "*Location*" may have the same meaning that both contain (x, y, θ) in a 2D coordinates framework, but the two words cannot be matched according to a pure string matching standard.In such situations, the proposed semantic matching standard matters. In general, a correct matching rate of about 89 % can be achieved. In comparison, the pure string matching method performs poorly (lower than 20 % success rate).

The proposed method also outperforms the Wu and Palmer [28] method, since their method often outputs a zero similarity score in the situation where the number of "*is-a*" links or "*hasPart*" links from the most specific common

superclass to the root of the taxonomy is zero. Moreover, the information-theoretical definition of similarity proposed by Lin [32] requires that the probability that a randomly selected object belongs to a certain concepts is given by WordNet. By contrast, our method can work without such information and can thus be applied to a smaller predefined set of ontology.

SN: ServiceRequired2	SC: DynamicInfoService∩MonitorService
I->O:	SubscribeInfo, RGBTypeRobotSignColor → NoticeInfo∩(∋hasTopic "RobotLocation")∩(∋hasVariable "RobotLocation")
SEA: SEA ∩ ∋ coverLocationPoint RobotLocation	
QOS: PerformIndex∩(∋hasName "LocalizationError")∩(∋hasValue Value∩(∋hasMax 20cm))	

Table 2. Example of robot requirement

ID	I	O(hasVariable)	Similarity score
1	RobotFeature	RobotLocation	0.25
2	RobotMovingSpeed	RobotLocation	0
3	RobotApperance	RobotLocation	0.50
4	RobotSize	RobotLocation	0
5	RobotSignColor	RobotLocation	0.80
6	**RGBTypeRobotSignColor**	**RobotLocation**	**1.00**
7	EnumTypeRobotSignColor	RobotLocation	0
8	RedValue	RobotLocation	0.67
9	RGBTypeRobotSignColor	RobotPose	0
10	RGBTypeRobotSignColor	RobotHeading	0
11	RGBTypeRobotSignColor	XCoordinate	0.67
12	RGBTypeRobotSignColor	YCoordinate	0.67

Table 3. Similarity score in the example

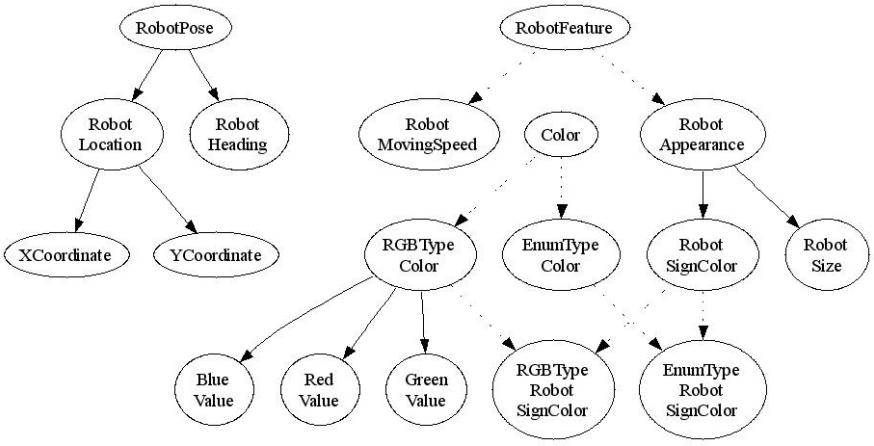

Figure 12. Example of ontology used for semantic matching

6.2 Two testing scenarios

The system is tested in several application scenarios, but due to the page limit, only two experimental instances are shown in this paper.

In the flexible robot grasping scenario (as shown in Figure 13), the dual-arm robot is equipped with a calibrated Kinect sensor for localizing the recognized object and transforming the position into the robot's base coordinate. To demonstrate the run-time configuration performance of the system, a complex action example of "*RobotArmServeA-Cup*" is manually decomposed into a coarse level which contains three sub-actions. Similar to the method used in KnowRob, the ordering of sub-actions is specified by the partial ordering constraints (partly shown in Figure 14).

Our proposed method is used to further decompose the sub-actions type "*PutSomethingSomewhere*" by finding competentservice components. For instance, the second sub-action "*PutCupBelowOfTapOfWaterDispenser*" is modelled as a semantic requirement represented by two inputs ("*cup*","*belowOfTap*") and an RDF effect (*cup*,*isLocated*,*belowOfTap*). The service composition procedure and the resulting service resource graph is shown in

Figure 15, in which magenta arrows indicate the backward search, blue arrows indicate the forward search and black dot lines indicate how pairs of components are matched.

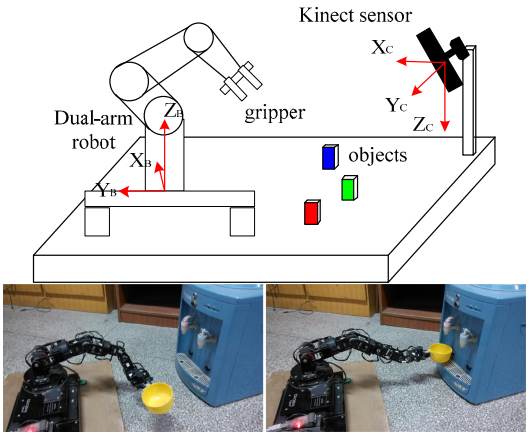

Figure 13. Application scenario 1

```
Class: RobotArmServeACup
        SubClassOf: ActOnObject
        EquivalentTo:
    subAction some ApproachToCup
    subActionsome        PutCupBelowOfTapOfWaterDis-
    penser
    subAction some WaitForPredefinedTimeInterval
    subAction some PutCupOnRobotDeck
    orderingConstraints value Ordering1
    orderingConstraints value Ordering2
    orderingConstraints value Ordering3
…
```

Figure 14. Action description

To find the plan, the backward search process starts from the S_{RE} node to connect $node_4$, $node_3$, $node_2$ and $node_1$ in proper order, according to the *Qualified Pre-action* composition criterion. The matchmaking between a pair of action-type components is based on the semantic matching between their PE parameters. The connected nodes are illustrated by the graph structure shown in Figure 16. After the four action-type components are connected, the requirements of each are further analysed to discover more necessary components, until the search procedure reaches the entrance node S_{RI}.Similarly, the forward search procedure starts from S_{RI} according to the *Qualified Post-service* composition criterion. The plan is executed as follows. The robot firstly infers the property of the cup and obtains its colour feature. Then the robot invokes the object recognition component to compute its 3D position. Meanwhile, the manipulation property of a cup is also queried, so that a proper grasping pose is determined. The two inputs are fed to the action-type component "*Approach-ToObject*", followed by other three sub-actions.

Another experimental scenario was an object delivering application. When the P3 robot was assigned with a task "*FetchMyPen*" via the web browser interface (Figure 17), it firstly invoked a place inference component to query the place and feature of the pen. It then invoked a map server to get the 2D grid map and a path planner to compute a trajectory to follow. Since the robot was not equipped with a flexible arm for accurately grasping a small-size object such as a pen, it sought the assistance of another robot (dual-arm), which picked up the pen and loaded it on the platform of the P3 robot. After the pen was loaded on board, the robot sought a visual perception component (global camera) for person tracking and localization. With the help of the path planner component and the robot localization component, the robot navigated to the human so that the pen was returned to him/her. Figure 18 shows some results of the trial.

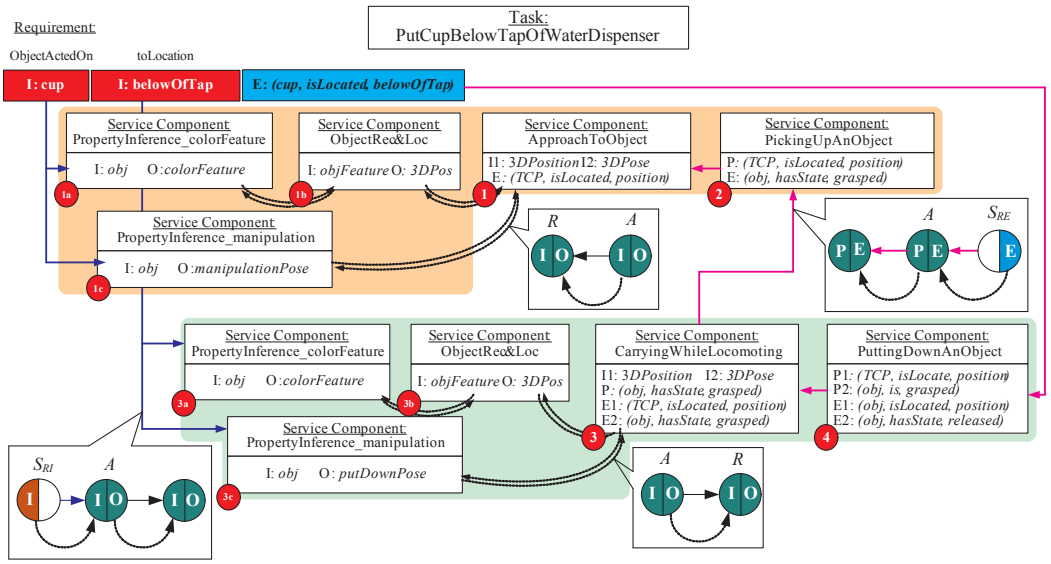

Figure 15. Service discovery and action decomposition

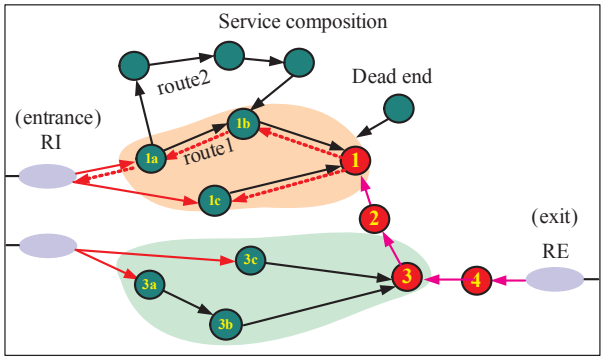

Figure 16. SRG of scenario 1

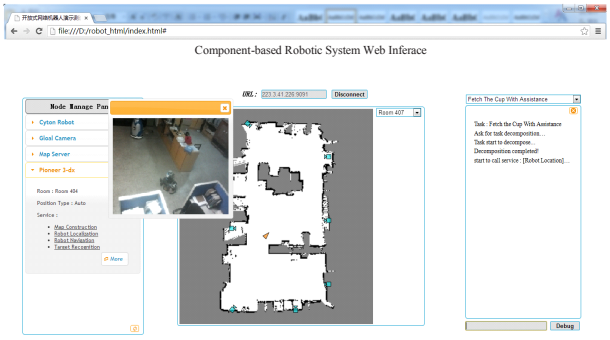

Figure 17. Web browser interface to use to initiate a service robot's task

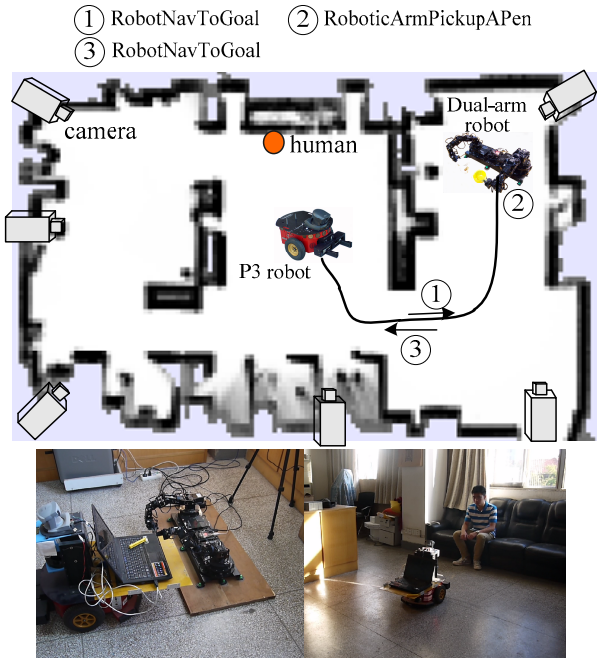

Figure 18. Application scenario 2

7. Conclusion

In this paper, a component-based robotic system framework is proposed for ubiquitous service utilization. The system tackles the problem of flexible service discovery by semantic service modelling and matching. In order to avoid the drawback of overly strict matching found in most existing matchmaking algorithms, we have extended the relation definition in OWL and proposed a new entity matching metric with a conceptual span similarity method that compares the Requirement Tree (RT) and Advertisement Tree (AT). Hence, more reliable semantic matching is achieved compared with previous methods.

The system implementation makes use of ROS middleware for low-level connection management among robots and nodes, which enables the advantages of our system in interfacing with an open source robot knowledge base as well as with a web browser client. The framework empowers robot developers to build a reusable ambient intelligent software system, in which the capabilities of robots can be greatly extended by the task-oriented utilization of ubiquitous services.

8. Acknowledgements

This work is supported by the National Natural Science Foundation of China (Grant No. 61105094, No. 61075090, No.61005092and No.61175091), the Fundamental Research Funds for the Central Universities(No.2242013K30004), and it's also funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.

9. References

[1] Jia S, Takase K. Network distributed monitoring system based on robot technology middleware. International Journal of Advanced Robotic Systems. 2007; 4(1):69-72.

[2] Tenorth M, Klank U, Pangercic D, Beetz M. Web-enabled robots: Robots that use the Web as an information resource. IEEE Robotics & Automation Magazine. 2011; 18(2):58-68.

[3] Kamei K, Nishio S, Hagita N, Sato M. Cloud networked robotics. IEEE Network. 2012; 26(3): 28-34.

[4] HaY G., SohnJ C, Cho Y J, et al. Towards a ubiquitous robotic companion: Design and implementation of ubiquitous robotic service framework. ETRI Journal. 2005; 27(6):666-676.

[5] Brooks A, Kaupp T, Makarenko A, Williams S. Towards component-based robotics. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems; 2005. p. 163-168.

[6] Blum S A. Towards a component-based system architecture for autonomous mobile robots. In: Proceedings of IASTED International Conference on Robotics and Applications; 2001. p. 220-225.

[7] Chang H. A study on intelligent system design of U-business service by using ubiquitous component technologies. International Journal of Advanced Robotic Systems. 2012; 9.

[8] Mosch F, Litza M, Auf A E, et al. ORCA - Towards an organic robotic control architecture. In: Proceedings of on Self-Organizing Systems; 2006. p. 251-253.

[9] Utz H, Sablatnog S, Enderle S, et al. Miro-middleware for mobile robot applications. IEEE Transactions on Robotics and Automation. 2002;18(4): 493-497.

[10] Ando N, Suehiro T, Kitagaki K, et al. RT-middleware: Distributed component middleware for RT (robot technology). In: Proceedings of International Conference on Intelligent Robots and Systems;2005. p. 3555-3560.

[11] Jang C, Lee S, Jung S, et al. OPRoS: A new component-based robot software platform. ETRI Journal. 2010; 32(5): 648-656.

[12] Ngan L D, Kanagasabai R. Semantic web service discovery: State-of-the-art and research challenges. Personal and Ubiquitous Computing. 2013;17(8): 1741-1752.

[13] Kuster U, et al. On the evaluation of semantic web service frameworks. International Journal on Semantic Web and Information Systems. 2004;4(4): 31-55.

[14] Klusch M, Fries B, Sycara K. Automated semantic web service discovery with OWLS-MX. In: Proceedings of 5th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS); 2006. p. 915-922.

[15] Kiefer C, Bernstein A, The creation and evaluation of iSPARQL strategies for matchmaking. In: Proceedings of 5th European Semantic Web Conference (ESWC). Spain: Springe;2008. p. 463-477.

[16] McIlraith S A, Son T C, Zeng H L, Semantic web services. IEEE Intelligent Systems & Their Applications. 2001; 16(2): 46-53.

[17] OWL web ontology language: Overview. Available from:http://www. w3. org/TR/owl-features, Accessed on 20 Jun 2014.

[18] Granitzer M, Sabol V, Onn K W, et al. Ontology alignment – a survey with focus on visually supported semi-automatic techniques. Future Internet. 2010; 2(3): 238-258.

[19] Yan W, Zanni-Merk C, Rousselot F. Matching of different abstraction level knowledge sources: The case of inventive design. In: Proceedings of 15th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems, Kaiserslautern Germany: Springer; 2011. p. 445-454.

[20] Euzenat J, Valtchev P. Similarity-based ontology alignment in OWL lite. In: Proceedings of 16th European Conference on Artificial Intelligence, Valencia, Spain: IOS Press; 2004. p. 323-327.

[21] Stoilos G, Stamou G, Kollias S. A string metric for ontology alignment. In: Proceedings of 4th International Conference on the Semantic Web Galway, Ireland: Springer; 2005. p. 624-637.

[22] Song F, Zacharewicz G, Chen D. Pattern-based core word recognition to support ontology matching. International Journal of Knowledge-based and Intelligent Engineering Systems. 2013; 17:167-176.

[23] Paolucci M, Kawamura T, Payne T R, et al. Semantic matching of web services capabilities. First International Semantic Web Conference Proceedings. Berlin, Germany: Springer-Verlag; 2002. p. 333-347.

[24] Tenorth M, Beetz M. KNOWROB: Knowledge processing for autonomous personal robots. In: Proceedings of 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems; 2009. p. 4261-4266.

[25] Qian K, MaX D, Dai X Z, Fang F. Flexible ambient service discovery and composition for component-based robotic System. Journal of Ambient Intelligence and Smart Environments. 2012; 4(6): 547-562.

[26] W3C Resource Description Framework (RDF). Available from: http://www. w3. org/RDF Accessed on 20 Jun 2014.

[27] Qian K, Ma X D, Dai X Z, Fang F, Zhou B. Decision-theoretic navigation of service robots using POMDPs with human-robot co-occurrence prediction. International Journal of Advanced Robotic Systems. 2013; 10: 1-12.

[28] Wu Z, Palmer M. Verb semantics and lexical selection. In Proceedings of the 32nd Annual Meeting of the Associations for Computational Linguistics; 1994. p. 133-138.

[29] Knuth D E. The Art of Computer Programming. 3rd ed., Boston: Addison-Wesley; 1997.

[30] Qian K, Ma X D, Dai X Z, Fang F. Knowledge-enabled decision making for robotic system utilizing ambient service components. Journal of Ambient Intelligence and Smart Environments. 2014; 6(1): 5-19.

[31] Cousins S. Is ROS good for robotics? IEEE Robotics and Automation Magazine. 2012; 19(2): 13-14.

[32] Lin D K. An information-theoretic definition of similarity. In: Proceedings of the Fifteenth International Conference on Machine Learning; 2008. p. 296-304.