Journal of Ambient Intelligence and Smart Environments 6 (2014) 5–19
DOI 10.3233/AIS-130243
IOS Press

Knowledge-enabled decision making for robotic system utilizing ambient service components

Kun Qian [a,b,*], Xudong Ma [a,b], Xianzhong Dai [a,b] and Fang Fang [a,b]
[a] *Key Lab of Measurement and Control of Complex Systems of Engineering, Ministry of Education, China*
[b] *School of Automation, Southeast University, 2 Sipailou, Nanjing, 210096, China*

Abstract. Robotic and ambient intelligent environments are much more demanding regarding the knowledge a robot needs to have. A skilled robot performs human-scale manipulation tasks, interacts with a variety of objects, understands instructions given by humans and most importantly, requires the capability of interpreting ubiquitous resources and assembling them into a complex plan. In this paper, a novel Knowledge-enable Decision Making Framework (KDMF) is proposed for Component-Based Robotic System (CBRS). We exploit that the use of domain knowledge is pivotal to endow robots with higher degrees of autonomy and intelligence in CBRS. Ontology knowledge about classes, properties, relations is organized in OWL based conceptual map, which allows automated inference to derive new pieces of information. Knowledge about tasks is specified in a tree data structure, and knowledge about components' functions is formulated by a specific type of service specification profile. Using the knowledge representation, a task-oriented decision making method is proposed that integrates knowledge inference and service components utilization. In practical applications of ambient and robotic assisted living, robot's plans generated by the decision making software are based on the knowledge of components, rather than particular device instances, which improves the reusability and flexibility of the system. Experimental results validate the effectiveness of the proposed method.

Keywords: Component-based robotic system, semantic, knowledge, decision making, ambient intelligence

1. Introduction

In robotic and ambient assisted living applications [1–4], the ever increasing level of system complexity and autonomy is naturally demanding a more powerful system architecture that enables automatic utilization of ubiquitous services. Taking the heterogeneity of resources into account, Component-Based [5–8] robotic system (CBRS) approach provides a solution that shifts the emphasis from hardware-specific software programming to composing application systems from a mixture of components. Nowadays, middleware-based component platforms [8,9], such as ORCA [10], Miro [11], RT-Middleware [12], OPRoS [13], etc., have been introduced in a wide literature.

Component-based robotic system needs to make decisions in robot's perspective, about the selection and utilization of ambient services [14]. For example, in order to perform a daily object delivery task, a robot has to utilize a group of useful devices for perception and manipulation, such as embedded sensors (e.g., camera, sonar, and laser), actuators (e.g., wheels, gripper, and arm) and other surrounding or network resources. Unlike traditional robotic systems in which most tasks are executed according to hand-coded programs, the reusability nature of CBRS requires a new service computing discipline [15] that ensures reusable decision making method as well as a new framework that allows flexible ambient service utilization. Another important problem is that the social nature of robots demands not only interpreting general control commands that are explicitly de-

* Corresponding author. E-mail: kqian@seu.edu.cn.

scribed, but also the capability of inferring implicit information during task execution [16]. For example, when receiving the user's request "*I need some water*", robots usually need to infer and maintain a belief state about the person's intention, preference, object properties, place properties and other context information.

Therefore, the decision making of CBRS is critical for allowing ambient service utilization. We believe that the growing tendency to introduce high-level semantic knowledge into robotic systems envisions a world of knowledge-enable CBRS. Knowledge-enabled decision making improves the composability and reusability of CBRS. Composability is facilitated since the robot's control program does not need to be changed but the extended knowledge will introduce more solutions [16]. Reusability is improved because knowledge that has been described once can now be used multiple times for recognizing objects, inferring facts or parametrizing actions.

In traditional robotic systems, knowledge is usually hidden or implicitly described in terms of if-then statements. This makes it hard to re-use the knowledge and to apply it in different circumstances. A formal knowledge representation is needed to help the robot to make these aspects explicit and to increase the re-usability and flexibility. Formal methods such as Calculus of Communicating Systems (CCS) [17], Communication Sequential Processes (CSP) [18], PI-Calculus [19], Mobile Ambients [20], etc, are process algebra methods that are used to describe and theorise about concurrent systems. Of the formal methods, Mobile Ambients (MA) and their variants have been frequently applied to ubiquitous and pervasive systems, concerning the issues of requirement specification, runtime verification, etc. [21,22]. Although Mobile Ambients and their variants have inspired the applications in pervasive systems, they have not yet been adopted by the robotic society. One major concern is that most existing network-enabled autonomous robots, rather than those Web based tele-operation systems, are still in the early days of their network structure, i.e., distributed devices and resources are connected and managed via Local Area Networks (LANs). Though efforts have been made on Web tools for robots [23,24] and learning from Web [25], they are still far from the full capability of freely utilizing all kinds of Internet resources as humans do. Meanwhile, Mobile Ambients are designed for the mobile computation over the much larger scale of World-Wide Web and handling the administrative domains partitioned by firewalls [20]. Another possible reason that Mobile Ambients are not used in our CBRS is in the mobility per se. An ambient is a bounded placed where computation happens, so Mobile Ambients allow moving or nesting ambients. In contrast, the computation capability of a component is tightly fixed to a specific physical device in CBRS. Thus CBRS is typically featured by neither the property of moving computation, nor the boundaries of components.

Constructing a knowledge base for robot application is a challenging task since robotic applications have very specific demands. Generally, knowledge is usually encoded in the task planner in a suitable way. These include causal knowledge, that is knowledge about the effects of the robot's actions, and world knowledge, that is knowledge about the objects in the world, their properties and their relations. Related work on knowledge based robotics comprises three directions: robot description languages, semantic sensor specifications and semantic maps. An example of robot description is the Unified Robot Description Format (URDF) [27]. Kunze [28] proposes Semantic Robot Description Languages (SRDL), which improves URDF by explicitly representing the components of URDF descriptions in a symbolic knowledge base containing encyclopedic knowledge about robots and their components. Semantic sensor specification methods are mostly based on OWL ontology [29–32]. For example, Compton [33] proposes an OWL ontology that focuses on the composition of sensors. Semantic maps encode knowledge about robot workspace and the current state of the world. Galindo [34] proposes a specific type of semantic maps, which integrates hierarchical spatial information and semantic knowledge. Such semantic maps have two obvious benefits: extending the capabilities of the planner by reasoning about semantic information, and improving the planning efficiency in large domains.

In contrast to the traditional robotic system, CBRS requires not only knowledge representation of objects, relations and actions, but also the semantic properties of all types of components. Recently, Ontological knowledge is increasingly being used in distributed systems in order to allow automatic re-configuration in the areas of flexible automation and of ubiquitous robotics [16]. Ontological knowledge is also used recently to improve the inter-operability of robotic components developed for different systems. Pangercic [35] investigates semantic object maps (SOM+), which are a subcategory of maps that store information about the task-relevant objects in the environment, possibly including geometric 3D models of

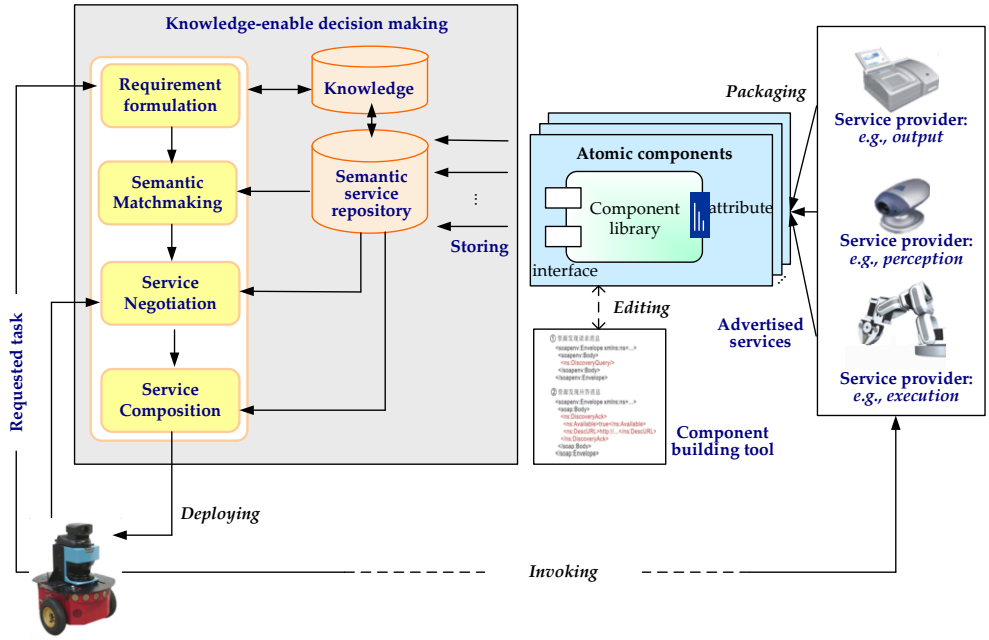


Fig. 1. Knowledge-enable decision making framework for CBRS.

the objects, their position and orientation, their appearance, and object category. In addition, ambient service resources, which are packaged as ubiquitous components, are needed for on-line operation of robots. They should also be assembled into plans in order to perform complex tasks. Hence the component-based robotic system also demands suitable mechanism that interprets knowledge about services and selects useful services according to task requirements.

So in this paper we choose Description Logics (DL) as formalism to represent robot's knowledge in component-based robotic system. Description logics are a family of logical languages. They are expressive, capable of describing relational knowledge and provide formal semantics that allow to draw conclusions from the available knowledge [26]. We build a Knowledge-enabled Decision Making Framework (KDMF) for CBRS. In particular, Ontology knowledge about classes, properties, relations are organized in a OWL based *conceptual map*, which can be further used for automated inference in order to derive new facts by combining different pieces of information. Knowledge about tasks is specified in a tree data structure for decomposition, and knowledge about components' functions is formulated by a specific type of service specification profile. Such knowledge representation provides models of robots and ambient services, respectively, and then software components can be designed and implemented based on the models rather than particular robot instance. The knowledge-enabled decision making of CBRS contains two major aspects: knowledge inference and task-oriented service component utilization. Firstly, since OWL is a file format for storing and exchanging description logic formulas, the formally represented knowledge allows drawing conclusions using deterministic logical inference. Secondly, a search based service composition method is proposed using knowledge description of service components.

2. A brief introduction of component-based robotic system

The Component-Based Robotic System (CBRS) contains service providers (resources) that advertise Web Services at one end, and service requesters (robots) at the other end. Figure 1 shows the service discovery framework of CBRS. The resources usually represent sensors, actuators and other devices in robotic system. Atomic components are distributed and reusable software modules that run loosely coupled and independently. They make the heterogeneity of devices transparent to upper-level applications using an abstract description of services, inputs, outputs, preconditions, and effects (IOPEs) as well as the non-functional properties (NFPs), according to the defined domain ontology [14].

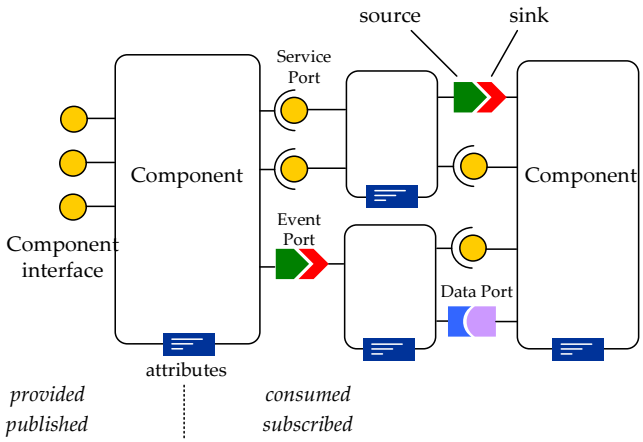


Fig. 2. Illustration of assembling components.

In the CBRS architecture, the service repository aggregates services of atomic components. When a request from a robot is initiated, the requirement is formulated by the requirement formulation module. Requirement is specified by a six-tuple similar to the service model as described our previous work [36]. The candidate services in the repository are matched with the formulated requirement according to the IOPE parameters. In the situation that service resources are in different runtime states, or providing different qualities of services, at a specific time, the service negotiation module is invoked to interact with the candidate providers to access dynamic information, if any. The service negotiation module accesses the Finite State Machine (FSM) of the dynamic resources and obtains their states for computing state dependent cost consumption, as described in [36]. Service selection and composition is then carried out, according to the requester's preferences and/or based on NFP considerations. Finally, the selected service is invoked by the robot and the results are returned.

An atomic component provides a specific service. The structure of an atomic component contains four major parts: basic functional module, service processing flow, component execution engine and service specification profile. The basic functional module implements functions and packages them into libraries. The service processing flow handles three types of information (method invocation, data, and events) and provides corresponding ports as interface for inter-component communication. The component execution engine provides lifecycle management, fault tolerance and other important modules. The service specification profile specifies the function of the service resource using a *Service Interface Definition Document* and a *Service description document*, as will be explained in Section 3.2. In general, an atomic component can be illustrated in Fig. 2 as a block with attributes and different types of interface. Figure 2 shows an example of assembling atomic components according to their source-sink correspondences and connecting each pair of "matched" components. A pair of matched and connected components communicates with each other through their service port, data port, or event port. A chain of components connected in sequence forms a plan to be invoked by robots.

3. Knowledge representation and inference for CBRS

In order to match between robots and actions, the concept of "capabilities" is introduced. An action depends on a set of capabilities and a set of components equips the robot with a certain capability. A capability is defined as a class in capability taxonomy and its properties are specified as will be explained in this section.

To store or query knowledge, existing description logic reasoners, such as Racer [37], Pellet [38] and Hermit [39] are not well-suited to robotics applications. One major reason is that robot's knowledge is not static. Existing description logic reasoners can be costly to re-classify the complete knowledge base whenever a robot acquires new knowledge from new senor data or human's teaching and updates its knowledge base.

So the system uses the Web Ontology Language OWL for modeling knowledge about *conceptual map*

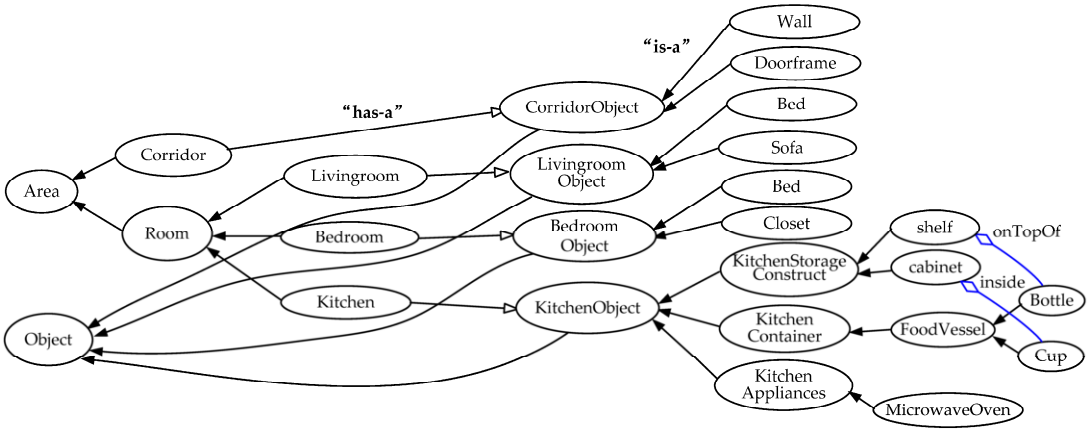


Fig. 3. Example of class relation and spatial relation.

as well as robot task. The *Conceptual Map* integrates ontology knowledge of facts and semantic knowledge of ambient services, as will be described in Sections 3.1 and 3.2. Meanwhile, logical programming language PROLOG [40] is employed for implementing the inference algorithms. OWL and PROLOG interact in a rather shallow way: knowledge is internally stored in terms of PROLOG predicates and implemented by a mapping from the OWL properties onto the first-order expressions in PROLOG [26]. For example, one class is assumed to be a sub-class of another one, so that the classification of instances can be performed by applying standard PROLOG inference methods. In this system, Amzi!, a prolog inference engine [41] is used to implement the inference predicates for reasoning about OWL classes, properties, restrictions and individuals.

3.1. Conceptual knowledge representation and parsing

Conceptual knowledge is used to specify knowledge about robots, sensors, actuators and others. In order to support knowledge storing and parsing, the static pieces of knowledge in the conceptual map are stored in OWL files using the standardized RDF/XML syntax format [42]. The implementation is based on the PROLOG Semantic Web Library for storing RDF triples and the OWL parser library for reasoning based on these representations. The representation is extended with predicates that compute complex relations by combing single pieces of knowledge into hierarchies of classes and properties. This extension enables the more advanced features of OWL such as hierarchies of class and properties. Figure 3 shows an example of class hierarchy which describes spatial and object knowledge required in a home-care application context.

- **Classes and properties**

All classes are organized in a taxonomic structure, from general classes like *Object* to specific ones like *Bottle*. The inherit property of an object class to its super-classes is reflected by "*is-a*" relations that use multiple inheritance. The location property is reflected by "*has-a*" relations that describe the fact that certain objects are often located at certain places. The taxonomy is able to describe object properties, so that knowledge about objects can be represented at different abstraction levels. Other properties of objects are also described by the RDF/XML syntax format. As an example of object properties shown in Fig. 4, an object has the properties of *location*, *label*, *place*, *manipulation*, *features*, *affiliation*, *name*, etc. These properties describe knowledge as concise and specific as possible. Note that although Figs 3 and 4 only illustrate part of important classes and properties, other parts of the ontology are also organized by similar representations.

In OWL individuals and their properties can be queried by using query predicates such as *rdf(?Subj, ?Pred, ?Obj)* to inspect the descriptions of classes and properties in the PROLOG database. For example, *rdf(?S, ?P, ?O)* returns only exactly matching triples, *owl_has(?S, ?P, ?O)* returns the OWL inference of "has-a" properties. Exploiting the hierarchical structure of classes enables those queries with generic relations to return all and more concise relations asserted for any of the sub-classes.

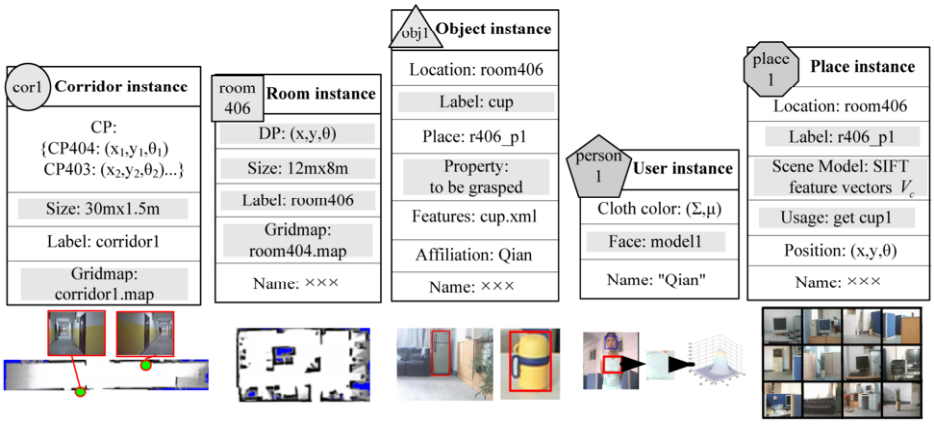


Fig. 4. Example of object properties representation.

- **Qualitative spatial relations**

Apart from metric positions, objects are usually also described by qualitative spatial relations to other objects. Such qualitative spatial relations are in accordance with the semantic knowledge of human. In this system we define a group of spatial relations, including *leftOf*, *rightOf*, *behind*, *onTopOf*, *inside* and *connectedTo*. The first five relations describe directional or topological relations between two objects, while the last relation can describe articulated objects and hinges, e.g., an instance of door is *connectedTo* an instance of cabinet. Figure 3 represents an example that a cup is placed inside of a cabinet and a bottle is placed on top of a shelf.

The qualitative spatial relations help to generalize spatial knowledge in robot application. For instance, if the system infers that an object is inside a cupboard whilst a cupboard is usually placed in the kitchen, the robot will firstly navigate to a room which is classified as kitchen, and then approach to the cupboard. Qualitative spatial relations of an object are obtained by PROLOG query, which will list all spatial relations corresponding to this object.

3.2. Service resource function representation

Service specification profile A *Service Interface Definition Document* based on WSDL describes services as a set of endpoints operating on messages and defines interface model for service component which allows for inter-component communication. Such a profile contains five elements of <types>, <message>, <portType>, <binding> and <service>. The first three elements describe the functionality and the invoking method of a service, as well as the required/provided information of a service. The latter two elements provide details on how abstract message parts map into the concrete protocol and data formats of the binding.

A *Service description document* is an XML based profile that describes concrete attributes of services by factors: “SEAModel”, “QoSModel” and “OWLSModel”, as described in our previous work [36]. The OWLSModel inherits the OWL-S definition in “Profile”, “Model” and “Grounding”. Nevertheless, the “Profile” section models four aspects:

- *ServiceName, SN*: It provides an access for accurate service discovery according the exact name.
- *ServiceCategory, SC*: Semantic searching according to the service category can help reducing scope of searching.
- *Input/Ouput, IO*: Inputs(outputs) of a service resource are described by a set of concepts, each representing an input(output) parameter.
- *Precondition/Effect, PE*: The PE model consists of several RDF statements, each presented in a Subject-Predicate-Object structure. The subject and the object can be express by OWL classes, but the predicate can only be express by an OWL property. The RDF statements describe the states and the changes between states caused by the P-E relation.

According to the above definition, the abstract service model characterizes a component’s service using a six-tuple of <DN, SN, SC, IOPE, SEA, QoS>, in which DN stands for *Device Num*, SN is *Service Name*, SC is *Service Category*, SEA is *Service Effective Area* and QoS describes the *Quality of Service*. Table 1 sums up the content of the service description document.

Table 1

Content of service description document

SN: ServiceProfile/ServiceName	SC: ServiceProfile/ServiceCategory
I –> O: ServiceProfile/IOPE/Input/datatype → ServiceProfile/IOPE/Output/datatype	
P –> E: ServiceProfile/IOPE/Precondition/datatype → ServiceProfile/IOPE/Effect/datatype	
SEA: {RoomSEA, CircleSEA, AreaSEA}	
QOS: {PerformIndex, PerformMap}	
Others: Other user defined data and additional property information about the service	

4. Knowledge representation and inference for CBRS

4.1. Task tree

A task consists of several steps or sub-tasks which result in a hierarchical structure and are formulated as a knowledge tree. The task tree is specified by nodes and links a pair of nodes, in which nodes stand for tasks or sub-tasks and the links stand for the property *hasSubTask*. A task can be decomposed into multiple layers of sub-tasks. Sub-tasks arranged in each layers represent a sequence of actions which require services that are loosely coupled. Figure 5 shows an example for task *FetchTheCup* consisting of 5 sub-tasks and one of them consists of three sub tasks. Actions in the third layer, e.g. *RobotNavToGoal* can not be further decomposed because it requires the collaboration of tightly coupled services, such as *localization*, *path planning*, and *obstacle avoidance*.

4.2. Inferring and composing required service component

After a complex task is decomposed into a sequence of sub-tasks, we now explain how to compose service components that are jointly able to perform a given sub task. This procedure is called service composition in our context.

Generally, in the field of Web Service a typical service matchmaking algorithm [31,32] takes an OWL-S *Query* from the client as input and iterates over every OWL-S *Advertisement* in its repository in order to determine a match. An Advertisement and a *Query* match if their *Outputs* and *Inputs* match. The algorithm returns a set of matching advertisements sorted according to four degrees of match.

In this paper, service composition is achieved by a search based service composition algorithm that entails the concept of requirements and capabilities. An action depends on a set of service components, but it's difficult to compare a sub-task with a component

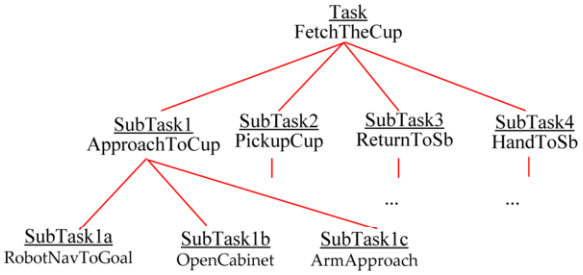


Fig. 5. Hierarchical task tree for FetchTheCup.

directly. Hence we explicitly specify the requirements and capabilities of tasks and service components in an abstraction level to act as a bridge. This is achieved by the abstract description of service components as well as tasks by the inputs, outputs, preconditions, and effects (IOPEs) and the non-functional properties (NFPs), according to the defined domain ontology. Table 2 shows some examples of IOPE parameters. 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.

The service composition algorithm firstly formulates the fundamental structure of a directed graph $G(\boldsymbol{V},\boldsymbol{E})$ called Service Resource Graph (SRG) to represent the "match" relations among a given task and candidate service components. The graph consist of a finite set of vertices $\boldsymbol{V}$, which represents available atomic service resources discovered by robot, and a set of directed edges $\boldsymbol{E}$, each of which associates a pair of atomic service resources. In this context, an edge can be linked between an atomic service resources A and an atomic service requirement R if their inputs and outputs "*match*" according to the following definitions:

Definition 1 (Qualified Pre-composite Service)**.** An atomic service resource A is a qualified pre-composite service for another atomic service resource R, iff A meets the need of R:

$$\mathrm{A} \mapsto \mathrm{R} \equiv (\mathrm{Same}(\mathrm{RO}, \mathrm{AI}) | (\mathrm{Contain}(\mathrm{AI}, \mathrm{RO})) . \quad (1)$$

Table 2

Examples of component specification using IOPE

Service component	Type	Input	Output	Precondition	Effect
PlaceInference	Static data	Object	Place		
QuerryDatabase	Static data	Place Object	Position Properties		
PathPlanner	Method invocation	Position	Path		
RobotLocalization	Dynamic perception	SensorData	RobotPosition		
ObjectRecognition & Localization	Dynamic perception	Object Properties	3D position		
RobotNavigateToGoal	Actuator manipulation	Path, robotPos			goalReached
OpenCabinet	Actuator manipulation			goalReached	cabOpened
ArmControl	Actuator manipulation		3D position	cabOpenned	armReached

Definition 2 (Qualified Post-composite Service)**.** Similarly, A is a qualified post-composite service for R, iff the following condition is satisfied:

$$\mathrm{R} \mapsto \mathrm{A} \equiv (\mathrm{Same}(\mathrm{RI}, \mathrm{AO}) | (\mathrm{Contain}(\mathrm{AO}, \mathrm{RI})) . \quad (2)$$

In the above definitions, Same(•) and Contain(•) are two types of relations defined in OWL-S. Note that not only I/O but also P/E parameters can be utilized to compare R with A because they represent causality relations between actions.

The service composition algorithm essentially makes use of a Bidirectional breadth-First Search (BBFS) [43] strategy for finding paths from an initial vertex to a goal vertex in the Service Resource Graph. We chose the BBFS strategy because of two concerns: First, a bidirectional search can reduce the search time by searching forward from the start and backward from the goal simultaneously. Second, breadth-first search in both directions would be more likely to be guaranteed to meet, compared with a depth-first search in both directions.

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}. Note that the requirement of a given sub-tasks can also be specified using P/E parameters. The structure of the Service Resource Graph is formulated automatically as follow.

Firstly, sub-tasks that can not be further decomposed are organized in action sequence by a backward searching process. It starts from the exit node S_{RO} according to the *Qualified Post-composite Service* standard performed on P/E parameters and discovers qualified service components to meet the need of *R*, as shown in Fig. 6. In the "*FetchTheCup*" example, sub-tasks *RobotNavToGoal, OpenCabinet* and *ArmApproach* are organized in the sequence of action and are linked to S_{RO}. They are represented by node_4, node_5 and node_6 in Fig. 7a and the link between each pair of nodes represents the match between the effect of an antecedent node and the precondition of a subsequent node.

Secondly, since these sub-tasks can not be further decomposed into other sub-tasks, their requirement and capability dependencies are analyzed. This involves matching the I/O parameters of more service components to find those qualified services that don't belong to actions. As shown in Fig. 7a, a query is conducted to find the required input of the node_4. The matchmaking between node_3 and node_4 reports a qualified match and thus a link is established between them.

Then, while the above search performed backwardly, a similar search procedure runs forwardly from the entrance node according to the *Qualified Pre-composite Service* standard as shown in Definition 1. The two full searching procedures execute in all candidate service components, resulting in a well organized Service Resource Graph.

Finally, the Service Resource Graph is compared to a maze, in which typical elements are entrance, exit, channel, crossroad and dead path. The solution is to solve the maze by finding paths to connect the entrance and the exit with maximum accumulated fitness value and minimum cost. Here the Bidirectional breadth-First Search (BBFS) algorithm is employed based on the established SRG structure starting from both the entrance and the exit. The bidirectional search stops when the two meet in the middle. Consequently, the resulting optimal path contains a sequence of atomic service resources that can be invoked by robot in sequence for accomplishing a specific sub-task, as shown in Fig. 7b. More details about the search algorithm can be found in our previous work [36].

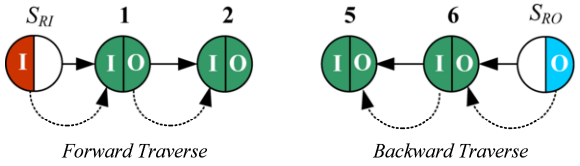


Fig. 6. The forward and backward search.

5. Experiments and results

In our previous work [4,36], we have built an instance of robotic and ambient intelligent environment for assisting the elderly and disabled. The system aims to provide daily care services such as object delivery, accompany, health monitoring, etc. In order to facilitate robot's manipulation, furniture and home appliances in the testing environment have been rebuild with simple sensing and reacting capabilities. For example, we have developed an automatic drinking water machine that can fill a cup automatically and a proximity-sensing cabinet that opens and closes the doors automatically. We believe that the assistive furniture and appliances can effectively relieve the manipulation burden of our robots, since they are designed as low-cost service robots with less complex manipulators. As such, the necessity of utilizing ubiquitous resources for accomplishing complex tasks is obvious.

As shown in Fig. 8, in the system different robot platforms (e.g., ActivMedia Pioneer 3 DX, Peoplebot, and Southeast University SeuBot) are designed as the service requesters. A group of embedded sensors are integrated, including DSP-based video cameras, global cameras, laser scanner, Kinect sensor, sonar and other environmental sensors. They work as processing units for people detection, activity recognition or environmental perception. An ARM-based smart device is developed to play the role of map server that provides the occupancy grid map of the environment for robot navigation. All these devices are encapsulated as software components in the CBRS. Besides, for supporting complex computational tasks, a set of method invocation components are developed, such as path planning, obstacle avoidance, etc.

Knowledge specifications are developed in the following way. In each component, the Service specification profiles are developed using WSDL/XML documents, as described in Section 3.2. In the central server, Amzi! PROLOG is used for specifying conceptual knowledge about classes, properties, relations and tasks. SQLite database is also used for storing

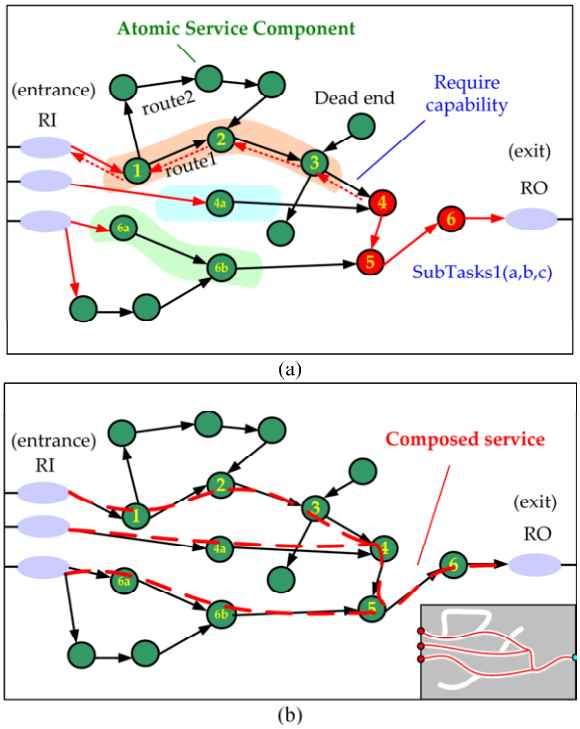


Fig. 7. Service composition.

and querying extensional knowledge such as objects and places properties. The central server runs the decision making software that generates plans for robot's task accomplishment.

The Knowledge-enabled Decision Making Framework (KDMF) was tested in several application scenarios, such as smart device assisted robot navigation for fallen person rescue, people-following and so on. In this paper, the scenario of "CupFetching" application is described in detail to validate the system performance.

In the "CupFetching" scenario, the decision making software firstly decomposed the task into subtasks as shown in Fig. 5. We take the first sub-task "ApproachToCup" as an example to illustrate the knowledge-based decision making. The requirement of "ApproachToCup" can be parametrized using two inputs ("cup" and global camera sensory data) and one casual effect (that the robot has reached the cup). The inputs and outputs (effect) of the sub-task are formulated as S_{RI} and S_{RO} respectively.

Firstly, the sub action nodes that can not be further decomposed are organized in sequence by a backward searching process that starts from the exit node S_{RO}. So that $node_4$, $node_5$ and $node_6$ in Fig. 7a are

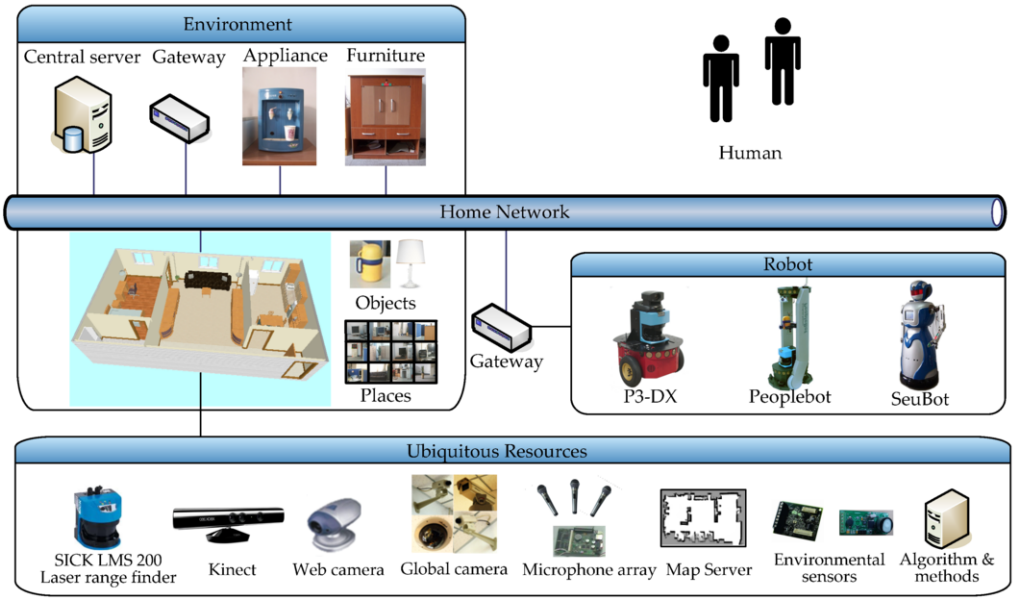


Fig. 8. The experimental system for ambient assisted living application.

Table 3

Examples of service description document of PlaceInference component

SN: QueryPlace	SC: StaticDataService
I –> O: Object → Place	
Data Topic: DataTopicList ⋂ (∍ hasDataTopic "ConceptualMap")	

connected. Similarly, $node_1$ and $node_2$ can also be connected because the input of $node_1$, *obj*, matches one required input of S_{RI}, "cup". Table 3 shows the service description of the *PlaceInference* component that corresponds to $node_1$.

Secondly, the requirement and capability of sub action nodes $node_4$, $node_5$ and $node_6$ are further analyzed to find more necessary components, if they have more than one input. As shown in Fig. 9, a query is conducted to find the other required input of the $node_4$, *robotPos*. By doing this, $node_{4a}$ is discovered since it connects S_{RI} and $node_{4.}$ Table 4 shows the service description of the *RobotLocalization* component that corresponds to $node_{4a.}$ Similarly, the matchmaking procedure also connects $node_{6a}$ and $node_{6b}$ with $node_6$.

The basic function of the *RobotLocalization* component is implemented using global video cameras for detecting robots in its field of view. Data measured by the global cameras have previously been calibrated onto the ground plane in the 2D occupancy grid map coordinates, so that the component can directly report the global *x*–*y* position of targets.

In the above mentioned matchmaking procedure, the search based service composition algorithm ensures that the chain of connected components has the highest matching likelihood and meets certain optimization criteria, such as minimum energy costs or time costs. More details about the service composition algorithm can be found in our previous work [36].

As a result, a plan is generated. In this example of sub-task "ApproachToCup", the plan is executed in the sequence as indicated in Figs 7b and 9. The whole procedure of the "CupFetching" scenario was executed as follow. Firstly, a global camera was utilized for initially localizing the robot. Secondly, the robot inferred about the fact that the cup was positioned inside the cabinet, and then queried the fact that the cabinet is located in the kitchen at a position (10.6 m, 12.8 m). Then the robot used the *PathPlanner* component to obtain a trajectory that led from its current position to the goal position. In the following step, the robot used the *RobotNavToGoal* component to reach the goal place, and at the same time the *RobotLocalization* component is also employed for updating the belief about robot's position using an appropriate global camera. Once the robot had reached the goal place, the cabinet opened and the robot used the *ArmApproach* component to get closer to the target object. In order to approach to the target cup, the

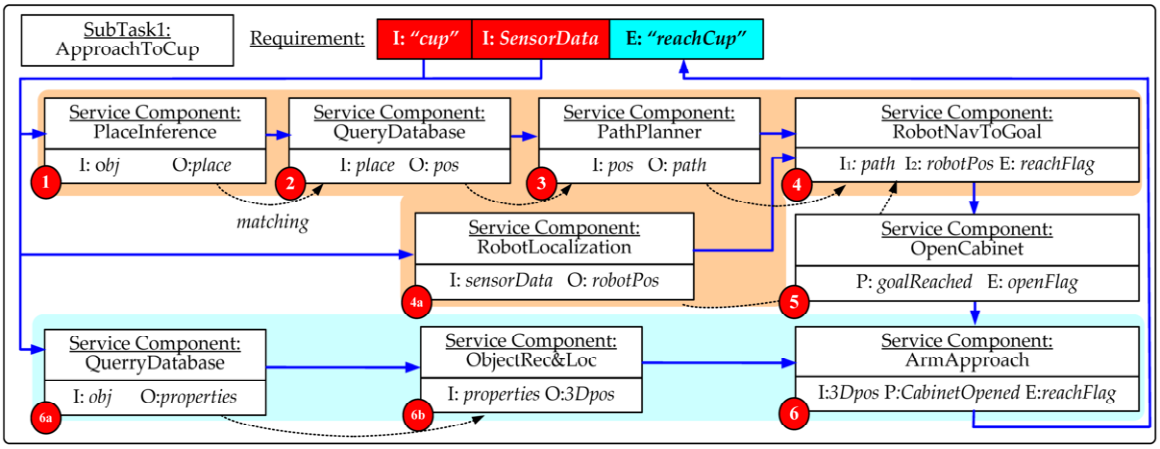


Fig. 9. Example of assembling services SubTask1 of the "CupFetching" scenario.

Table 4

Examples of service description document of RobotLocalization component

SN: RobotLocalization	SC: DynamicPerceptionService
I –> O: Sensor data → Robot position (x, y, θ)	
SEA:	RoomSEA ⋂ (∋ hasRoomName "*room404*"), CircleSEA ⋂ (∋ hasCircleArea CircleArea ⋂ (∋ hasCenter Center ⋂ (∋ hasX "7.8m") ⋂ (∋ hasY "12.9m")) ⋂ (∋ hasTheta "30.0°")) ,
QOS: PerformIndex ⋂ (∋ hasName "LocalizationError") ⋂ (∋ hasValue Value ⋂ (∋ hasAve "10cm") ⋂ (∋ hasMin "5cm") ⋂ (∋ hasMax "20cm"))	
Others: Supported environmental map type: occupancy grid map. Supported map resolution: 10cm, Supported sensor: SICK LMS 200 laser range finder.	

robot queried about the color appearance feature of the cup, before it invoked the *ObjectRecognition&Localization* component to compute a 3D position of the target using its eye-on-hand camera. After that, the robot used visual servo algorithm and kinematic control algorithm to grasp the cup and finally returned the cup to the person. Figure 10 illustrates some key steps of the experimental scenario. Figure 11 shows the results of robot navigation and object grasping.

The decision-making system is also tested in several other scenarios, two of which are described briefly here.

The second scenario is a person-following application. A Pioneer 3 DX robot was assigned with the task of following a person and keeping a certain distance behind him/her, which was decomposed into three subtasks: acquiring person appearance, person feature extraction and motion velocity control. The robot first queried the knowledge base about the visual appearance model of the target person, then sought a component for tracking people using RGB-D sensor (Kinect) and finally used a component for motion control that drove the wheel towards the target person. Figure 12 shows four snapshots of the person-following scenario, in which the executable plan was generated by the developed decision-making software system.

The third scenario is a remote object fetching application that is similar to the first scenario but entails the assistance of external manipulation functions. During the task execution, the Pioneer 3 DX robot inferred about its capability, the object location, the human position as well as the environmental occupancy grid map. Since the robot involved in the third scenario was not equipped with a flexible arm for accurately grasping small-size objects such as a pen, it sought the assistance of another robot (Robai Bi-handed Cyton 14D-2G), which picked up the pen and loaded on the platform of the Pioneer 3 DX robot. Figure 13 shows the process of the trial.

The computational performance of the system was evaluated by performing tests on executing the decision-making system with variations on scenarios and

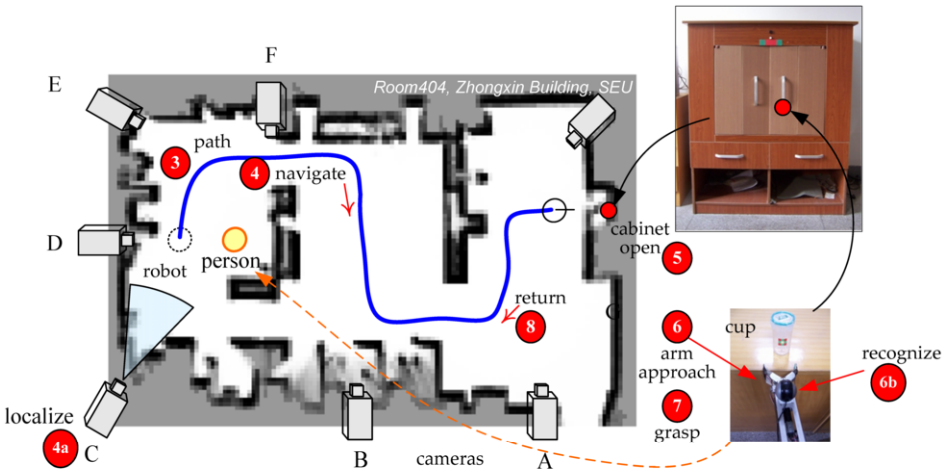


Fig. 10. Example scenario of "CupFetching".

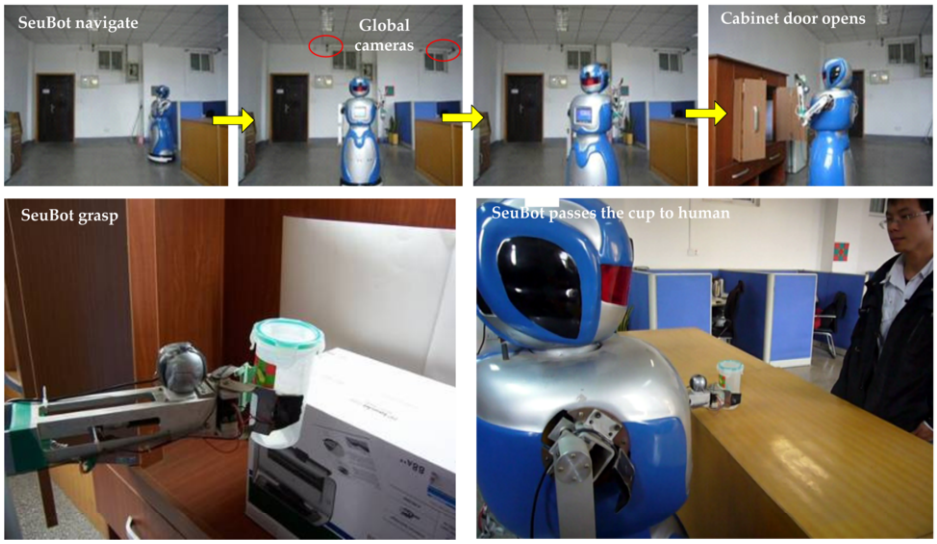
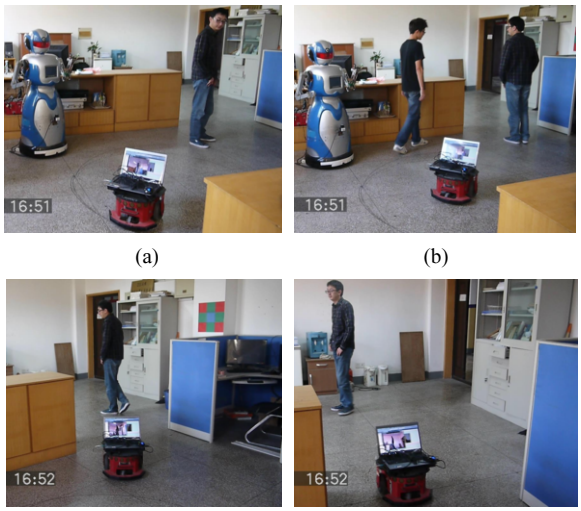


Fig. 11. Results of the experimental scene: scenario 1.

the number of candidate components. In the tests, a number of 25 components providing different services were developed. 95 additional simulated components were also implemented in the tests. The computational performance tests were conducted on the central server, which was a laptop computer with a 2.6 GHz Intel dual core, 2 GB RAM, and Ubuntu 2.04 OS, fixed on board of the Pioneer 3 DX robot. Table 5 shows the average results of the tests with 20 repeats, each with all subtasks executed. As expected, two major computationally expensive steps are semantic matchmaking and service composition. The case including more candidate components takes larger mount of time for matching. Due to the fact that registered components are managed at the central server, most of the time consumed by the semantic matchmaking procedure is cost by reading the registered component file stored on the hard disk. But the number of candidate components basically doesn't affect the service composition processing, because the structure of the established Service Resource Graph (SRG) is determined by the matched components. Note that during the runtime execution of the tasks, remote service port communication takes much more time, mainly because of the remote procedure call over the network.



(a) (b) (c) (d)

Fig. 12. Scenario 2, a person-following application.

The third scenario is a remote object fetching application that is similar to the first scenario but entails the assistance of external manipulation functions. During the task execution, the Pioneer 3 DX robot inferred about its capability, the object location, the human position as well as the environmental occupancy grid map. Since the robot involved in the third scenario was not equipped with a flexible arm for accurately grasping small-size objects such as a pen, it sought the assistance of another robot (Robai Bihanded Cyton 14D-2G), which picked up the pen and loaded on the platform of the Pioneer 3 DX robot. Figure 13 shows the process of the trial.

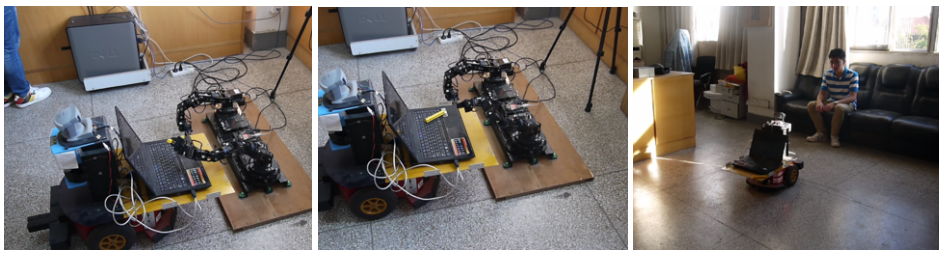
The computational performance of the system was evaluated by performing tests on executing the decision-making system with variations on scenarios and the number of candidate components. In the tests, a number of 25 components providing different services were developed. 95 additional simulated components were also implemented in the tests. The computational performance tests were conducted on the central server, which was a laptop computer with a 2.6 GHz Intel dual core, 2 GB RAM, and Ubuntu 2.04 OS, fixed on board of the Pioneer 3 DX robot. Table 5 shows the average results of the tests with 20 repeats, each with all subtasks executed. As expected, two major computationally expensive steps are semantic matchmaking and service composition. The case including more candidate components takes larger mount of time for matching. Due to the fact that registered components are managed at the central server, most of the time consumed by the semantic matchmaking procedure is cost by reading the registered component file stored on the hard disk. But the number of candidate components basically doesn't affect the service composition processing, because the structure of the established Service Resource Graph (SRG) is determined by the matched components. Note that during the runtime execution of the tasks, remote service port communication takes much more time, mainly because of the remote procedure call over the network.

The system features are also analyzed and compared with other robotic software platforms [12,13,44,45], as shown in Table 6. KDMF proposed in this paper is based on distributed component architecture including remote procedure calls and inter-component communication via various types of ports. KDMF supports automatic and semantic service discovery and service composition, which are partially supported in many other platforms. Compared with our previous system [36], KDMF also integrates knowledge representation, query and inference, which ensures future expansion of knowledge acquisition for robots.

6. Conclusions

In this paper, we exploit that the use of deeper domain knowledge is pivotal to endow a robot with higher degrees of autonomy and intelligence in component-based robotic system. We propose a knowledge-enable decision making framework that not only integrates ontology knowledge and service knowledge but also provides an effective solution to assembling qualified components for accomplishing complex tasks.

As a result of knowledge-enabled decision making, the robot is capable of utilizing ubiquitous services in its daily task execution. The essential problem solved in our research work has three aspects. First, robot is empowered by the ability to understand instructions given by humans and translate them into effective task specification. Second, robot can adapt its actions in the correct way depending on the current context and infer new knowledge during task execution. Most importantly, the knowledge representation of services and the automatic ambient service utilization method improve the reusability and flexibility of the system. As heterogeneous resources are wrapped as software components, they can be composed to complete different tasks without any additional development efforts.



(a) (b) (c)

Fig. 13. Scenario 3, a dual arm grasping application.

Table 5

Average elapsed time of executing decision-making

	Number of candidate components	Semantic Matchmaking	Service Composition
Scenario 1: CupFetching	25	105 ms	40 ms
(4 subtasks)	120	478 ms	
Scenario 2: PersonFollowing	25	78 ms	25 ms
(3 subtasks)	120	329 ms	
Scenario 3: PenPicking	25	202 ms	83 ms
(8 subtasks)	120	917 ms	

Table 6

Comparisons of robotic software platforms

	OROCOS	OPRoS	RT-middleware	Player & stage	Our pervious system	KDMF
OS	Windows	Windows/Linux	Windows/Linux	Windows/Linux	Linux	Linux
Remote procedure call control	Yes	Yes	Yes	No	Yes	Yes
Semantic service discovery	No	No	No	No	Yes	Yes
Service composition	No	Manual	No	No	Automatic	Automatic
Knowledge representation and inference	No	No	No	No	No	Yes
Interface with ROS	Yes	Yes	No	Yes	Yes	Yes

By developing knowledge-enabled CBRS, we believe that we empower more skilled robots that operate in ambient and smart environment for assisting human living.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (Grant No. 61075090, No. 61005092, No. 61105094 and No.60805032), and the open fund of Key Laboratory of Measurement and Control of Complex Systems of Engineering, Ministry of Education (No. MCCSE2012B02).

References

[1] F. Cardinaux, D. Bhowmik, C. Abhayaratne and M. Hawley, Video based technology for ambient assisted living: A review of the literature, *Journal of Ambient Intelligence and Smart Environments* **3** (2011), 253–269.

[2] A. Saffiotti and M. Broxvall, PEIS ecologies: Ambient intelligence meets autonomous robotics, in: *Proc. of the Int. Conf. on Smart Objects and Ambient Intelligence*, 2005, pp. 275–280.

[3] E. Aarts and F. Grotenhuis, Ambient intelligence 2.0: Towards synergetic prosperity, *Journal of Ambient Intelligence and Smart Environments* **3** (2011), 3–11.

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

[5] K. Qian, X.D. Ma, X.Z. Dai and F. Fang, Socially acceptable pre-collision safety strategies for human-compliant navigation of service robots, *Advanced Robotics* **24**(13) (2010), 1813–1840.
[6] A. Brooks, T. Kaupp, A. Makarenko and S. Williams, Towards component-based robotics, in: *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2005, pp. 163–168.
[7] G.T. Heineman and W.T. Councill, *Component-Based Software Engineering: Putting the Pieces Together*, Addison-Wesley, Boston, 2001.
[8] S.A. Blum, Towards a component-based system architecture for autonomous mobile robots, in: *Proc. IASTED Int. Conf. Robotics and Applications*, 2001, pp. 220–225.
[9] T. Niemueller, A. Ferrein, D. Beck and G. Lakemeyer, Design principles of the component-based robot software framework fawkes, in: *Proc. of the Second International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, Lecture Notes in Computer Science, 2010, pp. 300–311.
[10] F. Mosch, M. Litza, A.E. Auf et al., ORCA – Towards an organic robotic control architecture, in: *Proc. of on Self-Organizing Systems*, Vol. 4124, 2006, pp. 251–253.
[11] H. Utz, S. Sablatnog, S. Enderle et al., Miro – Middleware for mobile robot applications, *IEEE Transactions on Robotics and Automation* **18**(4) (2002), 493–497.
[12] N. Ando, T. Suehiro, K. Kitagaki et al., RT-middleware: Distributed component middleware for RT (robot technology), in: *Proc. of International Conference on Intelligent Robots and Systems*, 2005, pp. 3555–3560.
[13] C. Jang, S. Lee, S. Jung et al., OPRoS: A new component-based robot software platform, *ETRI Journal* **32**(5) (2010), 648–656.
[14] M. Tenorth, U. Klank, D. Pangercic and M. Beetz, Web-enabled robots: Robots that use the web as an information resource, *IEEE Robotics & Automation Magazine* **18**(2) (2011), 58–68.
[15] L.D. Ngan and R. Kanagasabai, Semantic Web service discovery: State-of-the-art and research challenges, *Personal and Ubiquitous Computing* (2012), 1–12.
[16] M. Tenorth, Knowledge Processing for Autonomous Robots, Ph.D. Dissertation, Technical University of Munich, 2011.
[17] R. Milner, *A Calculus of Communicating Systems*, Lecture Notes in Computer Science, Vol. 92, Springer-Verlag, 1980.
[18] C.A.R. Hoare, Communicating Sequential Processes, *Programming Languages* (1983), 306–317.
[19] R. Milner, J. Parrow and D. Walker, A calculus of mobile Processes, Part l, *Information and Computation* **100**(1) (1992), 1–40.
[20] L. Cardelli and A.D. Gordon, Mobile ambient, in: *Proc. the First international Conference on Foundations of Software Science and Computation Structure*, M. Nivat, ed., Lecture Notes in Computer Science, Vol. 1378, Springer-Verlag, 1998, pp. 140–155.
[21] A. Coronato and G.D. Pietro, Formal specification and verification of ubiquitous and pervasive systems, *ACM Trans. on Autonomous and Adaptive* **6**(1) (2011).
[22] A.E.K. Sobel and M.R. Clarkson, Formal methods application: An empirical tale of software development, *IEEE Transactions on Software Engineering* **28**(3) (2002), 308–320.
[23] S. Cousins, Is ROS good for robotics?, *IEEE Robotics and Automation Magazine* **19**(2) (2012), 13–14.
[24] A. Brandon, H. Kaijen, J. Chad, S. Bener and T. Russel, Robot web tools, *IEEE Robotics and Automation Magazine* **19**(4) (2012), 20–23.
[25] C. Penaloza, Y. Mae, K. Ohara, T. Takubo and T. Arai, Web-enhanced object category learning for domestic robots, *Intelligent Service Robotics* **6** (2013), 53–67.
[26] M. Tenorth and M. Beetz, KNOWROB: Knowledge processing for autonomous personal robots, in: *Proc. 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 4261–4266.
[27] URDF, http://www.ros.org/wiki/urdf.
[28] L. Kunze, T. Roehm and M. Beetz, Towards semantic robot description languages, in: *Proc. IEEE International Conference on Robotics and Automation*, 2011, pp. 5589–5595.
[29] S.A. McIlraith, T.C. Son and H.L. Zeng, Semantic web services, *IEEE Intelligent Systems & Their Applications* **16**(2) (2001), 46–53.
[30] OWL Web Ontology Language: Overview, http://www.w3.org/TR/owl-features/.
[31] M. Paolucci et al., semantic matching of web service capabilities, in: *International Semantic Web Conference*, LNCS, Springer Verlag, 2002, pp. 333–347.
[32] U. Bellur and R. Kulkarni, Improved matchmaking algorithm for semantic web services based on bipartite graph matching, in: *Proc. IEEE International Conference on Web Services*, 2007, pp. 86–93.
[33] M. Compton, H. Neuhaus, K. Taylor and K. Tran, Reasoning about sensors and compositions, in: *Proc. Semantic Sensor Networks*, 2009, pp. 33–48.
[34] C. Galindo, J.A. Fernández-Madrigal, J. González and A. Saffiotti, Robot task planning using semantic maps, *Robotics and Autonomous Systems* **56** (2008), 955–966.
[35] D. Pangercic, B. Pitzer, M. Tenorth and M. Beetz, Semantic object maps for robotic housework – Representation, acquisition and use, in: *Proc. IEEE International Conference on Intelligent Robots and Systems*, 2012, pp. 4644–4651.
[36] K. Qian, X.D. Ma, X.Z. Dai and F. Fang, Flexible ambient service discovery and composition for component-based robotic system, *Journal of Ambient Intelligence and Smart Environments* **4**(6) (2012), 547–562.
[37] V. Haarslev and R. Müller, RACER system description, *Automated Reasoning* (2001), 701–705.
[38] E. Sirin, B. Parsia, B.C. Grau, A. Kalyanpur and Y. Katz, Pellet: A practical OWL DL reasoned, *Software Engineering and the Semantic Web* **5**(2) (2007), 51–53.
[39] R. Shearer, B. Motik and I. Horrocks, Hermit: A highly-efficient owl reasoned, in: *Proc. the 5th International Workshop on OWL: Experiences and Direction*s, 2008, pp. 26–27.
[40] L. Sterling and E. Shapiro, *The Art of Prolog: Advanced Programming Techniques*, MIT Press, 1994.
[41] Amzi!, http://www.amzi.com/.
[42] RDF/XML Syntax Specification, http://www.w3.org/TR/REC-rdf-syntax/.
[43] D.E. Knuth, *The Art of Computer Programming,* Vol. 1, 3rd edn, Addison-Wesley, Boston, 1997.
[44] H. Bruyninckx, Open Robot Control Software: The OROCOS Project, in: *Proc. IEEE Int. Conference on Robotics and Automation*, 2001, pp. 21–26.
[45] B.P. Gerkey, R.T. Vaughan and A. Howard, The Player/Stage project: Tools for multi-robot and distributed sensor systems, in: *Proc. International Conference on Advanced Robotics*, 2003, pp. 317–323.