A framework for Intelligent Service Environments based on Middleware and General Purpose Task Planner

Wenshan Wang, Qixin Cao, Xiaoxiao Zhu, Shuang Liang
Research Institute of Robotics
Shanghai Jiao Tong University
Shanghai, China
amigo@sjtu.edu.cn

***Abstract*—Aiming at providing various services for daily living, a framework of Intelligent Service Environment of Ubiquitous Robotics (ISEUR) is presented. This framework mainly addresses two important issues. First, it builds standardized component models for heterogeneous sensing and acting devices based on the middleware technology. Second, it implements a general purpose task planner, which coordinates associated components to achieve various tasks. The video demonstrates how these two functionalities are combined together in order to provide services in intelligent environments. Two different tasks, a localization task and a robopub task, are implemented to show the feasibility, efficiency and expandability of the system.**

Keywords—intelligent service environment; middleware; task planning;

I. INTRODUCTION

Aiming at providing various services for daily living, there are increasing efforts on combining the robotic research with the ambient intelligence research. One consequent field is the ubiquitous robotics [1][2]. In ubiquitous robotic systems, sensors and actuators are distributed as equivalent modules in the environments, where these modules are able to communicate and cooperate with each other through the network. Compared to the traditional monolithic robot, the ubiquitous sensing and acting network enables the system to complete more complex service tasks, as well as decreases the development expense.

In the video, we demonstrate a system, taking advantages of heterogeneous networked sensors and actuators in the environment, and providing services following user's commands. This system is based on ISEUR framework. One of the most notable features of this framework is that it combines Robot Technology Middleware (RTM) with a general purpose task planner. Not only does the middleware enable the communication and interoperation between each pair of the devices, but it also provides standardized interfaces to the up-level planner, which is able to resolve various tasks in different domains. The video demonstrates this system with two different domains, the localization domain and the robopub domain. It is shown that this framework enables easy extension of new devices as well as the easy transition to new task domains.

This research has been supported by Yaskawa Electric Corporation, and the National High Technology Research and Development Program of China (No. 2012AA041403, No.2012AA041401).

II. SYSTEM OVERVIEW

The ISEUR framework mainly addresses two issues. First, the distributed robotic devices may be highly heterogeneous both with regard to hardware platform and software implementation. It is infeasible for these devices to communicate and collaborate with each other. As a result, the first problem is how to integrate these large amounts of heterogeneous devices and allow painless modification, expansion and deletion. Besides, there are a variety of tasks and different situations for each task in the day-to-day service scenarios. Users' commands are usually vague and complex, such as 'give me a cup of water', 'clean the room', etc., which need further decomposition and organization before being understood and executed by the system. So the second problem is how to plan on these complex problems in optimization of time and resource consumption.

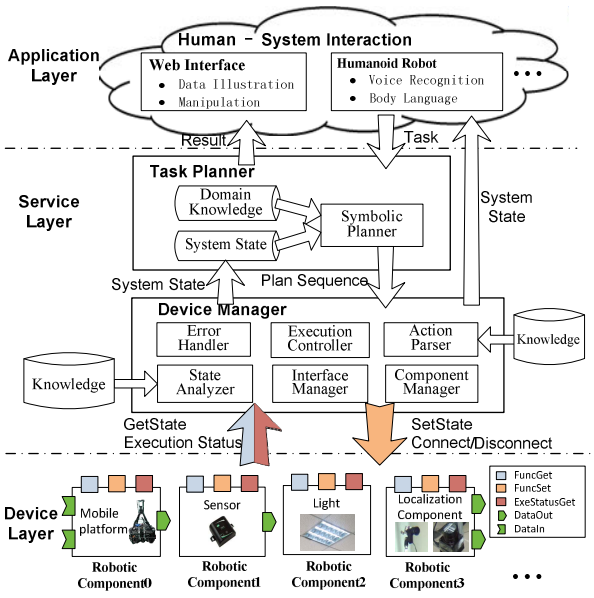

Fig. 1. The system framework of ISURE

According to these issues, the proposed framework is composed of three layers. They are the device layer, the service layer and the application layer, as Fig. 1 shows.

In the device layer, a middleware technology is introduced into the system. It encapsulates heterogeneous robotic devices into uniform standardized components. The components communicate with each other through

978-1-4673-6654-0/15 $31.00 © 2015 IEEE
DOI 10.1109/IE.2015.40

uniform standardized ports. This also brings benefits to the easy modification of existing devices and the expansion of new ones. It will be detailed in Section 3.

The service layer contains a device manager and a general purpose task planner. Device manager is responsible for transferring up-level tasks and monitoring low-level status. The task planner turns users' abstract commands into sub-task sequences, which can be directly carried out by corresponding components. We will further describe this in Section 4.

The application layer provides the interaction between human and the ISURE system. Two kinds of user interfaces are implemented. One is a web interface that can be accessed from remote computers or mobile phones. The other is a humanoid robot that acts as a communication interface with both vocal and body languages.

III. MIDDLEWARE-BASED DEVICE LAYER

As mentioned above, the distributed devices are highly heterogeneous with respect to platforms such as operating system, programming language and communication media. Thus the middleware is employed to generalize the devices into uniform standardized components, which enables the dynamic communication and cooperation between any two of the modules. Fig. 2 explains how the middleware transparentizes the hardware and software platforms, and offers standardized access to the robotic devices.

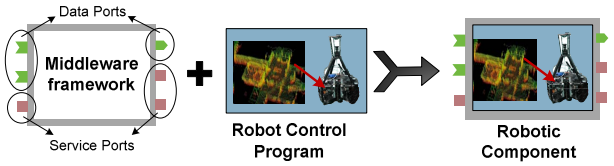

Fig. 2. Conceptual Model of Middleware

Many research efforts have been expended on the middleware technology for heterogeneous robotics system [3], such as Player [4], Lime [5], Miro [6], RTM [7] and etc. We argue that the middleware technology needs to have the capacity to provide not only the communication model but also the interfaces for managing component states. With this in mind, a distributed localization system using RTM is proposed. We implement our system based on RTM owing to its two functionalities.

- The Data Ports and Service Ports for data exchange and service invoking.
- Component execution context for lifecycle management.

The ports are categorized into data ports and service ports. The data port is responsible for the continuous exchange of data, while the service port provides the command based communication. Each component can have any number of data InPorts and OutPorts. A data OutPort sends the data to a corresponding InPort which receives the data. The component with a service port, offering a set of services, listens for requests upon those services via a connector. Fig. 3 depicts the simplified UML model of the RT-component.

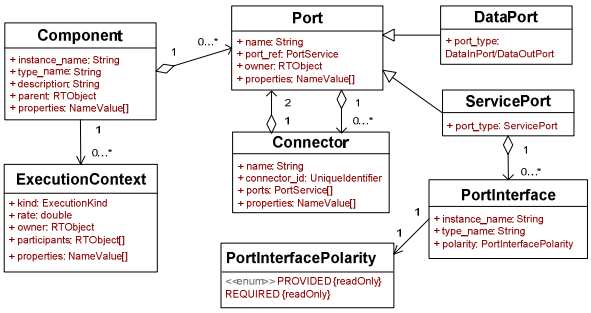

Fig. 3. Simplified UML component model

Based on that component model, Fig. 4 illustrates some of the components in our system. Each component is responsible for some specific functionality. For instance, the environmental camera is responsible for robot localization and object recognition tasks; the mobile table is in charge of transportation. Furthermore, the decentralized modules are able to collaborate through flexible combination, which increases the reusability of the components. For example, the environmental camera can provide localization information for multiple robots; the path planning module can also serve for multiple mobile platforms.

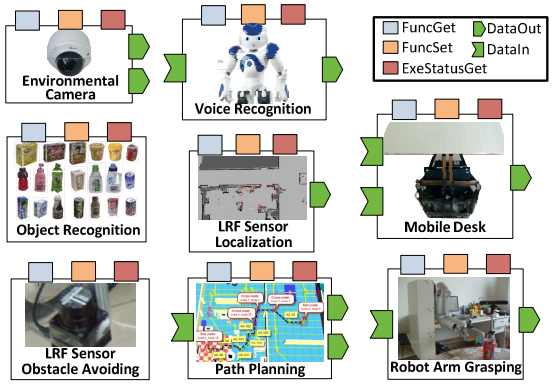

Fig. 4. Some of the components for the experiments of this study. The distributed components are in the uniformed structure using middlware technology.

Each component has three service ports, namely FuncGet, FuncSet and ExeStatusGet. These service ports are responsible for the interaction with the upper layer. FuncGet port reports to the service layer about the components' state. For example, the environmental camera provides the index of current robots within the field of vision; the mobile table feeds back its coordinates, etc. FuncSet port provides the functionality invoking, such as setting the target location for the mobile table, setting the localizing target for the environmental camera, etc. ExeStatusGet port returns the execution status, such as whether the mobile table has reached its destination, or whether the environmental camera is locking on its target.

The data ports allow the communication and cooperation between components. For instance, the localization information is transferred from the data out-port of environmental camera to the data in-port of the path planning component. Once two data ports are connected,

those two components are able to perform real time communication to accomplish the task collaboratively.

IV. Task Planning Module in the Service Layer

The task planning module is a crucial part in the service layer. In ISURE, the tasks are complicated and the situations are dynamic. It is unlikely to predefine all the possible states. As a result, a flexible and robust planning method is needed. What's more, it is supposed to be a domain-independent general approach for solving a variety of problems.

A. Task Modeling

Task modeling is the precondition of the task planning. The quality of the planning result greatly depends on the expressivity of the task model. On the other hand, the more complicated of the model, the more difficult for the planner to solve the problem.

This paper follows the techniques in automated planning field. The Task planning problem is modeled as a state transition system. Formally, it is modeled as a five-tuple $\prod = (S, A, c, I, G)$, where:

- $S = \{s_1, s_2, \cdots\}$ is a finite set of world states;
- $A = \{a_1, a_2, \cdots\}$ is a finite set of actions, each $a \in A$ is a triple $(name_a, pre_a, eff_a)$ referred to the action's name, parameters, precondition and effects respectively. The action's name includes the parameters associated with the action.
- $c : A \mapsto \mathbb{R}_0^+$ is the cost function;
- $I \subseteq S$ is a set demotes the initial state;
- $G \subseteq S$ is a set denotes the goal state.

To further depict the planning domain and planning problem, the Planning Domain Definition Language (PDDL) [8] is employed. Some sample actions are shown below, representing the moving capability of the mobile robot and the grasping capability of a robot arm.

```
(:action drive
    :parameters (?r - mobile ?start - place ?dist - place)
     :precondition (and (at ?r ?start) (can-locate ?r))
     :effect (and (at ?r ?dist) (not (at ?r ?start))))
(:action pickup
     :parameters (?a - arm ?o - object ?p - plane)
     :precondition (and (beside ?a ?p) (on ?o ?p))
     :effect (and (in ?o ?a) (not (on ?o ?p))))
(:action putdown
     :parameters (?a - arm ?o - object ?p - plane)
     :precondition (and (beside ?a ?p) (in ?o ?a))
     :effect (and (on ?o ?p) (not (in ?o ?a))))
```

B. Task Planning

Inspired by International Planning Competition (IPC), the automated planning technology has been significantly improved these years. The increase was mainly due to three fundamental approaches, which are the Graphplan approach [9], satisfiability method [10] and heuristic search method [11]. The last one shows a better performance in recent years.

This paper employs the heuristic search based algorithm to solve the planning problem we defined above, referring to the Fast Downward (FD) planner [12]. The PDDL files are translated to build a search space, which can be seen as a directed graph, where the nodes denote the states of the system, and the links denote the actions that make the system transfer from one state to another. FD searches the shortest path that starts from the initial state and reaches the goal state. The links on the path compose an action sequence, which is the planning solution. We improve the FD planner by adapting it to the online planning system. The detailed algorithm is shown below.

Algorithm 1: Task planning

```
while exists task T uncompleted:
    for each alive component C_i :
        s_i ← readState(C_i)
        if S_i is ERROR_STATE: reset(C_i) endif
    endfor
    S_init ← analyzeState(s_0, s_1, ···)
    S_goal ← analyzeTask(T)
    P_task ← taskModelPDDL(S_init, S_goal)
    T_result ← FDplanner(P_task, P_domain)
    for each sub-task t_i in T_result :
        while(r_i ← execute(t_i) not complete) endwhile
        if r_i is SUCCESS: continue
        else: break with failure
        endif
    endfor
    if not failure: mark T as completed
```

C. Combining middleware and task planner

As Fig. 1 illustrates, the service layer and the device layer communicate through 3 kinds of service ports, namely FuncGet, FuncSet and ExeStatusGet. The Device Manager is developed as the bridge between these two layers.

Firstly, the FuncGet service ports are used by readState(C_i) function with respect to Algorithm 1. Each component C_i provides the functionality of reporting its own states. For instance, the object recognition component reports the name of objects that are currently in its view. All these states are translated by the Device Manager, and then form the initial state fed to the task planner.

Secondly, the planning result is in the form of action sequences, such as (drive robo1 door2 table1), (pickup arm0 coke table1), etc. Notice that the first item is the action name, and the second item is configured as the component name, of whom is in charge of this action. Device Manger compiles each action into a method-call through associated FuncSet service port. For example, the above two actions are compiled as robo1.move_to(table1_x, table1_y) and arm0.pickup(coke_id) respectively. These methods are defined in an interface definition language (IDL) file for each service port. One action is bound to one method of the service port. The action could also be bound to the connection or disconnection of two data ports. For example the action (localized-by robo1 cam1) means

connecting the data out-port of the camera component and data in-port of the robot component.

Thirdly, when executing each action, it is important to monitor its status. If it's successful, it will move on to the next action, while if it's failing, it will start that over again. The ExeStatusGet service ports are responsible for reporting the execution status. There are 4 types of status, namely idle, running, success and failure.

Two major benefits of this approach are that it allows an easy extension with new components and allows an easy transition to new task domains. Adding new components has few side effects on the existing ones and the planning module. All that we need to take care is to define the three kinds of service ports or other necessary data ports. Besides, same set of components can be used for different task domains, as long as the PDDL files are provided.

V. SCNARIOS SHOWN IN THE VIDEO

In the video, there are two task domains. One is the localization domain, and the other one is the robopub domain.

In the localization scenario, one mobile robot is moving around in the environment with the localization info provided by three environmental cameras. The mobile robot and three cameras are in the form of components implemented with service and data ports. The working area of each camera is depicted in a domain definition file.

The system receives user commands of destinations via a web interface. Then the planner works out the plans and sends them to the components. In this scenario, the camera switching process is automatically performed, promised by the cooperation of the device layer and service layer. Besides, it is quite convenient to add new localization components. In our previous work, three kinds of localization components, which are based on laser, camera, RGB-D sensors respectively, have been developed for robot localization [13].

The second scenario is in a pub, where customers can order drinks. This task comprises many technologies, such as object recognition, motion planning, localization, path planning, object avoiding, voice recognition, etc. A number of components were employed for this scenario. There are four environmental cameras, one mobile table with laser sensor and one robot arm. Software components are also implemented, including object recognition, path planning, etc. In this scenario, a humanoid robot works as the human-system interface. Compared to the monolithic service robot, our system is more efficient and robust.

This scenario is an upgrade version of the first scenario. They are under the same framework and use same piece of planning code. The camera components and the mobile robot components from the first scenario are reused. And no modification is needed when deploying them to the second one. All that is needed is adding some new components, and upgrading the domain description file. The components and the planning module are reusable for different domains.

VI. SUMMARY AND FUTURE WORK

This paper has proposed a framework of ISURE, aiming at providing services to common day-to-day deployment scenarios. The middleware-based device layer and the service layer with a general purpose task planning module have been discussed in detail. The experiments signified that this framework significantly improves the system's expandability and reusability.

The framework is able to be enhanced with new technologies, such as other task planning approaches and other human-system interaction methods. We have been working on task planning algorithm with probabilistic feature. And advanced knowledge sharing algorithms are under investigation for the better automation and higher intelligence.

VIDEO LINK

The video can be viewed at the following link:
http://youtu.be/fXLKMSKQKYs

ACKNOWLEDGMENT

This research has been supported by Yaskawa Electric Corporation.

REFERENCES

[1] A. Saffiotti, et al., "The PEIS Ecology project: vision and results," Proc. of the IEEE/RSJ Int Conf on Intelligent Robots and Systems (IROS), 2008.

[2] G. Amato, et al., "Robotic ubiquitous cognitive network," in Ambient Intelligence-Software and Applications, ed: Springer, 2012, pp. 191-195.

[3] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "A review of middleware for networked robots", *International Journal of Computer Science and Network Security*, Vol.9 No.5, pp. 139-148.

[4] T.H.J. Collett, B.A. MacDonald, and Gerkey, B.P. "Player 2.0: Toward a practical robot programming framework", In *Proc. ACRA*, 2005.

[5] A.L. Murphy, G.P. Picco, and G.-C. Roman, "LIME: A coordination model and middleware supporting mobility of hosts and agents", *ACM TOSEM*, Vol.15 No.3, pp. 279-328.

[6] H. Utz, S. Sablatnog, S. Enderle, and G. Kraetzschmar, "Miro-middleware for mobile robot applications", *IEEE Transactions on Robotics and Automation*, Vol.18 No.4, pp. 493-497.

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

[8] D. McDermott, "PDDL–the planning domain definition language," *the AIPS'98 Planning Competition Committee*, 1998.

[9] A. L. Blum, M. L. Furst, "Fast planning through planning graph analysis," *Artificial Intelligence*, 2005, pp.279-298.

[10] H. A. Kautz, B. Selman, "Pushing the envelope: Planning, propositional logic, and stochastic search," In *Proc. AAAI-96*, pp. 1194-1201.

[11] B. Bonet, H. Geffner, "Planning as heuristic search," *Artificial Intelligence*, 2001.

[12] M. Helmert, "The Fast Downward Planning System," *Journal of Artificial Intelligence Research*, 2006, pp.191-246.

[13] W. Wang, et al., "An automatic switching approach of robotic components for improving robot localization reliability in complicated environment", *Industrial Robot: An International Journal*, Vol. 41 No.2, pp. 135-144, 2014.