A Task-oriented Service Composition for Internet of Things

Jang-Ho Choi, Han-Gyu Ko, Jae-Hyun Cho, Angel Jimenez-Molina, Jorge Gaete, In-Young Ko
Department of Computer Science
Korea Advanced Institute of Science and Technology
Daejeon, Republic of Korea
[jangho87, kohangyu, skswwho, anjimenez, jgaete, iko]@kaist.ac.kr

Abstract—Service-oriented computing has enabled dynamic composition of Web services to help users meet their task goals. However, one of the emerging issues in Urban Computing (UrbComp) environment is to seamlessly utilize the new concept, Internet of Things (IoT) to compose and provide IoT-based services of local smart objects. This challenge is about enabling services that are actuated by Smart Objects (SmObs), supported and delivered from the perspective of users. In order to accomplish this goal, it is essential to integrate IoT-based services into the service composition framework to discover nearby SmObs and utilize their operations to deal with the high-level goals of users. We use the task-oriented computing approach to utilize the concept of IoT in an UrbComp environment. This approach represents users’ goals in tasks, which are then bound to available service instances. We describe an implemented prototype of our framework and demonstrate a scenario to show its use in a real life to illustrate its feasibility.

Keywords—Internet of Things; Urban computing; User centricity; Task-oriented computing; Service composition

1. INTRODUCTION

Urban computing (UrbComp) is a new paradigm extended from ubiquitous computing (UbiComp). Urban environments are denser in terms of the number of users, larger regarding their physical settings, and more diverse in relation to the type of users and social groups. The main goal of UrbComp is to enable users to access networked services embedded in the infrastructure or in the Web anytime, anywhere [1]. We are already experiencing an increasing availability of smart objects (SmObs) in UrbComp environments. Examples of SmObs are smart phones, smart TVs, robots, intelligent appliances, public displays, smart vehicles, and traffic sensors, among others actuators. These SmObs are connected via standard protocols so that they are always accessible through the Internet. The concept is called as Internet of Things (IoT), which refers to a concept in an urban computing that different types of network-capable smart objects are able to communicate one another to accomplish common goals. Therefore, we can expect new applications that interact with online services and IoT-based services for the spontaneous service provision [2].

Assume that there is a user who drives his car in the downtown to buy a digital camera. Since the streets are crowded, finding a parking spot is a hard task to him. Fortunately, his smart phone shows a suitable area to park by interacting public cameras, parking meters, and information from other users looking for parking spots. After he parks the car, his smart phone recommends the electronic stores that he might be interested in, according to the information on the social network applications of users co-located and the product reviews on cameras satisfying his preferences.

However, in reality, there is a gap between low-level functions of those SmObs and high-level goals of the user. In order to satisfy user goals utilizing the IoT, users need to recognize available SmObs and understand their functionalities even the place they have never been before. Moreover, there may be users with various goals in the UrbComp environment with a large number of SmObs, increasing the complexity of utilizing the appropriate set of SmObs significantly.

Hence, to discover and provide appropriate operations of SmObs and deal with the complexity problem, we need to explicitly represent user goals and make IoT-based services efficiently arranged to meet the goals. The task-oriented service composition framework aims to represent user goals in an explicit task definition that is machine-processable. Common tasks are predefined in our repository as a form of template in BPEL that each task template is comprised of a flow of abstract services. The abstract services are instantiated by orchestrating available services including the operations actuated by local SmObs presented (e.g. smart phones, public displays, public speakers, laptops, etc.) in an UrbComp environment.

In this paper, we present the task-oriented service composition framework to provide IoT-based services that are actuated by operations of SmObs. In order to support tasks of the user, the proposed framework firstly discovers possible services based on SmObs around users. The proposed framework then finds a set of available tasks given the possible services. Important technical requirements for realizing the proposed framework are efficient descriptions of SmObs and a discovery mechanism that corresponds to the proposed framework. Hence, we explain how we setup SmObs and integrate them into the proposed framework as an example, showing their use in a real life to illustrate its feasibility.

The rest of this paper is organized as follows. In Section 2, we state related works of task-oriented frameworks and the IoT. Then, in the next section, we introduce our task-oriented service framework, its overall architecture and core mechanisms. In Section 4 and 5, we describe implementation details and a demonstrated scenario in our experimental
environment in which real SmObs are installed, followed by the conclusion and future work in Section 6.

II. RELATED WORK

A. Task-oriented Frameworks

Many existing frameworks such as Gaia [8], Aura[9], and Amigo [10] have tried to identify user activities in an accurate and efficient manner by taking into account the contextual information of the UbiComp environment. Gaia provides a mechanism for building new applications and for making use of a predefined application model. It allows applications to change their structure dynamically given inter-space user migration. One of the limitations of this framework involves the static binding of the user information within the predefined and available applications in the UbiComp environment.

In Aura, users can interact with the system in terms of high-level tasks instead of individual services or applications as stated by Wang and Garlan. Although Aura is a representative of a task-oriented framework, it assumes a given knowledge of the relationships between virtual services and concrete service suppliers. Amigo is an example of a task-oriented service provision in ambient system domain. Ambient systems are of interest as they pertain to the present study, as they share a number of identical features with the UbiComp environments. A drawback of Amigo is that workflows are directly mapped to available services without any mediating process and without abstract layers. This leads to a lack of flexibility and reusability of compound services.

B. IoT-based Service Provision

There are also researches that tried to connect services and real world using various approaches. SECE (Sense Everything, Control Everything) [11] is an event-driven system that allows user to make a new service that connects/combines communication, social, devices in the real world. In communication between SmObs and SECE, SECE uses gateways and servers to control and connect between sensors and actuators. However, it has some drawbacks of scalability or interoperability problem between services and system since it is a kind of nonstandard system. Our framework uses Web Services standards (BPEL1, SOAP2, WSDL3) that can be scalable and easy to integrate new SmObs. Using WS-* standards has several advantages in flexibility and interoperability.

SOCRATES [12] is a service integration system that uses Web service standards to combine services and SmObs. They propose the approach that also uses Web-oriented pattern such as RESTful APIs to connect service with SmObs. The main contribution of SOCRADES is Real-world Service Discovery and Provisioning Process (RSDPP). RSDPP is to help the developers at implementation time in the discovery of real-world services to be contained in composite applications. When a new device joins to SOCRADES, the device should multicast a message to introduce itself. However, the subject of service discovery is a mobile client in our framework. The mobile client actively discovers available SmObs and orchestrates the composition of them.

III. SERVICE COMPOSITION FRAMEWORK FOR IoT

A. Overall Architecture of the Proposed Framework

In order to support the user in his/her tasks in UrbComp environment, our framework help utilize IoT-based services offered by available SmObs. To achieve this goal, we have reflected three important technical requirements in the system architecture of our framework: (1) users’ needs and contextual information need to be gathered, (2) users’ tasks need to be represented in a form that is machine processable, and (3) the gap among tasks and SmObs needs to be semantically connected.

Figure 1. Overall architecture for task-oriented service framework

As shown in Figure 1, our client module collects context information of user, such as his/her needs and environmental context. Our repository system consists of task ontology manager and task definition manager. Common tasks and activities are predefined and stored in the repository system to enable automated service discovery. The major function of the task manager consists of selecting and filtering those activities from the repository, matching IOPE (inputs, outputs, preconditions and effects) of each SmOb with the contextual information that characterizes the UrbComp environment and the users. The gap between IOPE and user context is eliminated by using hierarchical task ontology.

B. Task Selection Mechanism

The task selection mechanism follows a bottom-up view of the task composition process, by firstly considering SmObs that are available in the space to construct only the locally supported tasks. It starts by identifying the building blocks of tasks, that is, the feasible activities. This is done by semantically matching the SmObs IOPE to the ones required by the activities in the task repository. Once a set of feasible activities is identified, they are tailored into tasks by semantically matching the activities coordinated in the task templates available in the task repository, against the obtained filtered activities [1]. During the composition process, the context information of user is considered as a restriction for the binding of activities into tasks. This results on a set of
tasks that are feasible in the given UrbComp environment by both, the given contextual information and the existing SmOb.

While the task was inferred with a bottom-up focus, its deployment is decided using a top-down approach. Our framework deploys the task by a series of services actuated by the available SmOb, orchestrating them into a composition flow in consideration of the activities being performed in the physical environment.

C. Smart Object Discovery Mechanism

The IoT Connection Manager detects available SmOb around the user, and receives its description in the Web Services Description Language (WSDL) format. It includes the semantic interface information, which consists of available operations with its IOPE and required parameters. Furthermore, the description also has a network address for its SmOb for direct communication.

In order to find SmOb, the IoT Connection Manager exploits Multicast DNS, which is a method to register a service and find it on the network. Using the Multicast DNS, a client notifies its behavior in local network using multicast. Then, other SmOb can recognize client’s behavior through the ServiceListener. In this time, the SmOb can response specific information such as name of SmOb, port of operation, description, and etc. Thus, clients can receive available SmOb and their descriptions by using multicasting between SmOb and clients. Obtained descriptions are delivered to the Task Manager, and they are utilized by the unit-task selection mechanism described previously.

IV. IMPLEMENTATION

To perform a task in an IoT environment, we have built a simulation environment including SmOb, client-side application and task manager.

A. Smart Objects

SmOb are built by implementing agent systems on the existing electronic devices of physical space. The architecture of SmOb is shown in Figure 2.

![Smart Object Diagram](image)

**Figure 2. Architecture of a Smart Object**

In order to transform existing electronic devices into SmOb, which can be accessed by the Web, we have implemented an agent for each device. The agents are developed in Java, and deployed on *Beagle Board*, which is a low-power open source hardware single-board computer. *Beagle Board* plays important roles such as a communication medium and a ‘brain’ that it is able to collect and compute context data autonomously. Hence, each agent can detect context changes in the network, receive SOAP message from a client, and execute functions in physical devices. One of the challenges is that each physical device uses a different medium for its control, so that an appropriate adapter must be utilized to support the interface between the physical device and its agent.

To set temperature of an air conditioner, for instance, the client sends a SOAP message to its agent via HTTP protocol. The message contains an operation for setting preferred temperature in the air conditioner with a desired value as a parameter. The agent extracts the operation name and the desired temperature through SOAP Message Engine upon receiving the message. The associated method is called to generate necessary signal information to control. The object adapter then operates the device’s function based on the signal information from the agent method.

B. Client-side application

The client discovers available SmOb, and notifies the activity selector with their IOPE. This is realized by the IoT discovery, and the IoT description manager respectively, being both of them modules of the IoT connection manager. In addition, the recommended tasks from the task manager are presented to the users by the visualizer. This visualization involves both, the XML parser to extract the coordinated information from the template, and the user interface itself. Also, if the user chooses one of the tasks, the client requests its execution flow to the task manager, and actuates the SmOb through the modules of the task executer. The execution flow for the client is executed by dynamically binding the abstract services of the tasks to concrete and available operations. We express each service with a visual icon in addition to text description, providing the user intuitive experience. Activities already executed fade out and slide up for the next activities that gradually appear from the bottom.

C. Task Manager

Task managers are implemented and managed locally to increase performance and accuracy of discovery. Although the required computing resource of a user request to detect available SmOb would be insignificant, it may lead congestion at the server if a large number of users send the requests at the same time. In performance aspect, it may leads execution delays in spontaneous task selection, decreasing usability of the composition framework. Hence, we utilize task managers at local-level (i.e. at local network) so that task managers can react on user requests in a faster fashion. Moreover, another requirement for SmOb discovery is to detect available SmOb based on the user location. However, in reality, finding user’s indoor location is very difficult and the result would often be inaccurate. Hence, by using local connection with a local task manager, we could guarantee the only SmOb that are physically close to the user to be detected.
D. Repository System

We have utilized a standard BPEL (business process execution language) to describe a coordination of operations. In order to access each SmOb through the Web, a WSDL file for each SmOb is created and stored for local repository, where each WSDL file defines operations of a SmOb. Common tasks are predefined and stored as execution process templates in BPEL, so that operations of web services are invoked according to its process logic. One of the advantages of using BPEL is that we can express not only sequential flow, but also control flows such as conditions and loops.

V. A DEMONSTRATED SCENARIO

In this section, we demonstrate how our framework satisfies a user goal by using a scenario. Within the scenario, we also show the performance of SmOb discovery as an evaluation of our application framework.

1. As Sam enters a seminar room, his smartphone client multicasts a greeting message to all smart objects in the room: projector, door, light, smartboard, and printer. As a reply, each smart object returns its information such as IP address of its network and descriptions of its operations, which is stated in Table 1. Furthermore, as shown in the table, the average time of SmOb discovery is less than 0.4 seconds, which is quite reasonable for dynamic service composition.

2. Then, Sam’s mobile sends his context information with SmOb information to local Task Manager. SmOb information consists of operations and IOPE information that each SmOb has, whereas context information consists of his personal profile, preference, tasks and also environmental context. The seminar room has multiple place potentials, which are latent placeness. That is, the seminar room is capable of executing tasks such as academic or business. By using urban computing environment context and Sam’s preference, our framework finds the most appropriate place potentials of the place for Sam.

3. Based on the user context and descriptions of available smart objects, Task Manager finds a feasible set of unit tasks from the repository. The matching algorithm compares the effects of operations of smart object services using hierarchial effect ontology. In this specific case, the local task manager finds tasks such as ‘Having a meeting’, ‘Watching a movie’, and ‘Taking an online lecture’ by matching effects of discovered smart objects—i.e. projector and smart board—with description of tasks using task and effect ontologies. Then, these tasks are ranked based on Sam’s preference and shown on his mobile.

<table>
<thead>
<tr>
<th>[Smart Object]</th>
<th>[Operations]</th>
<th>[Avg. Response Time]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projector</td>
<td>Power On/Off, Input URL, Adjust Lamp Up/Down, Adjust Lamp Left/Right, Adjust Lamp Brightness, Adjust Lamp Contrast</td>
<td>0.269 s</td>
</tr>
<tr>
<td>Door</td>
<td>Lock On/Off</td>
<td>0.234 s</td>
</tr>
<tr>
<td>Light</td>
<td>Power On/Off, Brightness Setting</td>
<td>0.252 s</td>
</tr>
<tr>
<td>Smart Board</td>
<td>Up, Down, Stop</td>
<td>0.327 s</td>
</tr>
<tr>
<td>Printer</td>
<td>Power On/Off, Input URL, Cancel</td>
<td>0.358 s</td>
</tr>
</tbody>
</table>

4. As Sam selects ‘Having a meeting’ task from the task list, the client requests its task template to the server. The task template defines a sequential flow of abstract services: setting up the meeting environment, setting up the communication, presenting material, and closing the meeting environment. These services are displayed on the client screen.

5. The services are then automatically executed according to their pre-condition, dynamically bound with concrete operations of SmObs by the local task manager. During the execution, the current activity is located at the center and highlighted. In this case, the projector, light, and smartboard are turned on as the first service, ‘Setting up the meeting environment.’ When the environmental setup is finished, the client waits for Sam’s input on presentation material selection. As the user chooses a file, the file is transferred to the computer and displayed on the screen by the projector. While presenting the slides, the user can remotely control the presentation on his/her mobile.

6. When the meeting is over, Sam directly inform it to the client application so that the client application can command the projector, smartboard, and light to be power off. Or alternatively, Sam can simply leave the decision to the client that it may detect the pre-conditions of ‘closing the meeting environment’: no one left in the meeting room.

VI. CONCLUSION

In order to provide user-centric services utilizing the IoT in UrbComp environments, we introduced and described our task-oriented service framework. Since SmOb, which are
building blocks of IoT, can make the synergy between computationally augmented objects and services in general such as Web services, we presented how the proposed framework could include SmObS with a semantic representation model of unit-task and SmOb description model.

More specifically, in the proposed framework, users’ goals and SmObS are represented by semantic models and users’ goals can be satisfied by executing SmObS. A runtime infrastructure provides the spontaneous composition among SmObS through the description of the task template.

The major contribution of this work is to design a task-oriented service framework in which the IoT is utilized in the perspective of users. The proposed framework seamlessly integrates online services and IoT-based services to meet tasks that represent users’ goals. In addition, an implemented prototype with an example of its use in a real life scenario Have a Meeting is introduced to illustrate its feasibility. Within the experimental environment organized with SmObS such as vim projector, smart TV, light equipment and curtain, we could describe how the SmObS can be utilized in the proposed system to accomplish given users’ goals represented by ‘Have a Meeting’ task.

In future research, we will apply QoS-based service composition approach to maintain reasonable optimality and successfully provide composed services. Since there can be lots of SmObS that provide same or similar functionalities, it is required to resolve the problem of how the service provision framework find the most appropriate one within a given QoS constraint. In addition, we will include into the scenario the cognitive resource aware approach described in [7] to reconfigure abstract service coordination obtained by the unit-task composition mechanism.

ACKNOWLEDGMENT

This research was supported by the KCC (Korea Communications Commission), Korea, under the R&D program supervised by the KCA (Korea Communications Agency) (KCA-2012-11911-05005). We also thank to Gyu-Cheol Park and Jung-Hyun Kwon for surveying the related works.

REFERENCES