Computer Standards & Interfaces 34 (2012) 171–188

Contents lists available at SciVerse ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

User-configurable semantic home automation

Yung-Wei Kao [a,*], Shyan-Ming Yuan [a,b]

[a] Department of Computer Science, National Chiao Tung University, 1001 Ta Hsueh Rd., Hsinchu 300, Taiwan
[b] College of Computing & Informatics, Providence University, Taiwan

ARTICLE INFO

Article history:
Received 24 March 2011
Received in revised form 5 August 2011
Accepted 11 August 2011
Available online 8 September 2011

Keywords:
Smart home
Semantic home
Home automation
Web service
OWL

ABSTRACT

The ideas of smart home and home automation have been proposed for many years. However, when discussing homes of the future, related studies have usually focused on deploying various smart appliances (or devices) within a home environment and employing those appliances automatically by pre-defined procedures. The difficulties of supporting user-configurable automation are due to the complexity of various dynamic home environments. Moreover, within their home domains, users usually think semantically; for example, "I want to turn off all the lights on the second floor". This paper proposes a semantic home automation system, USHAS (User-configurable Semantic Home Automation System), which adopts Web Service and WSBPEL for executing automated process; OWL and OWL-S for defining home environments and service ontology; and a self-defined markup language, SHPL (Semantic Home Process Language), for describing semantic processes.

1. Introduction

Previous studies have focused on providing solutions for smart homes [1–4], home automation [5–7], and ubiquitous homes [8–10]. Numerous of these studies [11–13] concentrate not only on controlling devices, but also on combining additional factors before doing so. For example, facial recognition can be performed for opening a door, and hand gesture recognition can be utilized to change the channel of a TV. However, regardless of whether these home appliances are controlled manually by users or indirectly and automatically by computers, most systems provide only pre-defined control of appliances. For example, hand gestures indicating numbers for changing channels on a TV cannot be used as an input for changing the temperature of air conditioner, unless users modify the programs by themselves. Such modification requirement limits the flexibility and scalability of numerous home control and automation systems.

Controlling appliances without pre-defined procedures is difficult, as several requirements must be fulfilled. First, all devices, including home appliances, sensors, and remote controls, should be designed separately with standard interfaces provided. This criterion is reasonable, because users usually purchase devices manufactured by different companies. Second, how automation processes receive information from sensors and control devices should be standardized. Finally, these processes should be designed by users in an easy and understandable manner.

To provide interconnectivity between different parties, various Service-Oriented Architecture (SOA) [14] technologies have been designed. In general, numerous home systems adopted the Open Services Gateway Initiative (OSGi) architecture [15] for smart home implementation. However, the OSGi platform does not provide a complete process management and execution solution; therefore, pre-defined procedures for executing processes are usually designed for the OSGi. By contrast, Web Service [16] focuses on providing interconnectivity between parties. Moreover, the OASIS Web Services Business Process Execution Language (WSBPEL or BPEL4WS) [17] has been defined for process management and execution of Web Services. Therefore, this paper adopted the Web Service standard for defining operations of devices and the WSBPEL standard for defining automation processes.

Although WSBPEL provides an effective process definition standard, a gap remains between what users actually want and what they must define in processes. This is because users usually think semantically; for example, "I want to turn off all the lights on the second floor". In this instance, the WSBPEL process cannot be defined, because the home system does not know about the lights located on the second floor. To solve this problem, this paper adopted the Web Ontology Language (OWL) [18] to create a knowledge base in a semantic home, and the OWL-S [19] technology to describe the Web Services of the devices.

Although the OWL-S standard provides the functionality of process definition, the difference between the OWL-S process and the WSBPEL process is quite small. Only the input and output variables are mapped to the OWL ontology, and users must specify many binding details. To achieve the goal of semantic home automation, this paper adopts OWL-S only for semantic service description and

* Corresponding author.
E-mail addresses: ywkao@cs.nctu.edu.tw (Y.-W. Kao), smyuan@cis.nctu.edu.tw (S.-M. Yuan).

0920-5489/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.csi.2011.08.002

discovery, but not for semantic process definition and execution. Because no appropriate candidate is available for defining semantic home processes, this paper designed a markup language, SHPL (Semantic Home Process Language), to describe semantic processes. In addition, the SHPL execution runtime is developed, which dynamically generates a WSBPEL process for each semantic process based on the current home environment defined in the knowledge base.

This paper proposes a semantic home automation system, USHAS (User-configurable Semantic Home Automation System), which adopts Web Service and WSDL to execute automation processes; OWL and OWL-S to describe home environments and service ontology; and SHPL to define semantic processes. To prove that the proposed system can satisfy user needs adequately, this paper designed nine demonstration scenarios: the living room, long vacation, home gym, morning rush, dinner time, good student, sweet dreams, bath time, and party night. Furthermore, this paper designed and analyzed a questionnaire to determine which scenarios were more appealing to users. Finally, the usability of USHAS was evaluated.

The research contains seven sections. In Section 2, the backgrounds and related works of the proposed system are introduced. Section 3 discusses the design issues of USHAS. The USHAS ontology and SHPL are described in Sections 4 and 5 respectively. We present the system architecture and detailed system design of USHAS in Section 6. System demonstrations and evaluations of scenarios are presented in Section 7. Finally, we end up with a conclusion and discuss the future works of proposed system in Section 8.

2. Backgrounds and related works

2.1. Smart home and home automation

Smart homes and home automation are popular topics, referring to devices and appliances in the home environment that can be controlled automatically in an intelligent manner. Thus far, numerous studies have proposed system designs and even smart home products [1–4]. In smart home systems, devices and appliances are usually controlled to facilitate the duties of daily life. Home automation systems generally contain an internal network, as well as intelligent rules and devices in the home network for convenient or special purposes. Devices and appliances can be controlled automatically by, or provide environmental information to, these home automation systems. In addition, changing the state of a device may also change the state of another device, or trigger actions in other devices within the smart home environment [6].

2.2. SOA and OSGi-based smart home

Before defining a high level of logic for automation processes, the most crucial challenge is to facilitate the interconnectivity between different devices. Due to highly varied standards for home devices, such as X10 [20], INSTEON [21], UPnP [22], and Jini [23], communication between different devices is difficult to establish using dissimilar interfaces under various standards. For such heterogeneous network integration, the Service-Oriented Architecture (SOA) [14] design principle provides interoperability between various loosely coupled services. Open Services Gateway initiative (OSGi) [15] is one of the technologies that implement the SOA paradigm, and numerous researchers have implemented smart home systems based on OSGi. Li et al. [24] designed a home network system, providing a Web interface for users to control appliances directly on the OSGi platform. Ishikawa et al. [25] proposed SENCHA, a smart appliance integration middleware framework based on OSGi. They indicated several limitations of OSGi in implementing smart home automation, such as the lack of multiple views of abstraction levels. In other words, the capability of multilevel abstraction of OSGi is not enough. Wu et al. [1] combined the OSGi platform with mobile agents [26] in their design, which involves multiple mobile agents responsible for different tasks, distributed among multiple OSGi platforms. Liao et al. [8] adopted the Message-Oriented Middleware (MOM) [27] paradigm for event handling in their context-aware smart home system. Rui et al. [9] presented a physical structure model and a multi-agent [28] based software architecture based on OSGi; this architecture encapsulated the device sensing as well as control operations into the AmI-Adaptor, and encapsulated the computation logic into the AmI-Box.

The primary problem of OSGi is that only bundles installed on the same OSGi container can inter-communicate. Therefore, technologies such as mobile agents in [1] must be designed to establish communication between different containers. Another problem of OSGi is that no standard process definition is provided. Hence, in the OSGi based systems, high level device control decisions are usually made by multi-agents and pre-defined by programmers. As a result, automation processes are fixed and not allowed to be configured by users. Although the system designed in [9] provides the capability of user configuration to AmI systems, the configuration level is at the AmI-Adaptor and AmI-Box levels, not at the process level; users must select and configure which AmI-Adaptor or AmI-Box to communicate.

2.3. Web Service, WSBPEL, and Web Service Based Home Automation

Web Service [16] is another technology that implements the concept of SOA, consisting of three main standards: Web Services Description Language (WSDL) [29], Simple Object Access Protocol (SOAP) [30], and Universal Description, Discovery and Integration (UDDI) [31]. Similar to OSGi, Web Service provides interoperability between different services; therefore, it is also a candidate for smart home platforms. Uribarren et al. [32] proposed a middleware system based on Web Service for controlling devices with different protocols. Unlike OSGi, a process execution standard, Web Services Business Process Execution Language (WSBPEL) [17], is designed to support process definition for executing Web Services. Because devices are usually provided by different manufacturers, the concept of device functionalities can be mapped to services provided by enterprises; and the concept of process execution using different devices can be mapped to cross-business process execution, including different enterprises. Anke et al. [33] revealed the drawback of OSGi: OSGi bundles are not directly accessible from clients outside of the OSGi container. To solve this problem, Anke et al. designed a system that exposes OSGi bundles using Web Service interfaces, and then executes these bundles using processes defined in WSBPEL. However, using both OSGi and Web Service involves a duplicate design, because both of them follow the SOA paradigm. Systems supporting WSBPEL process definition and execution can adopt device drivers directly implemented by Web Service, instead of by OSGi bundles, to reduce system overhead.

2.4. Semantic Web, Context-Aware Home, OWL-S, and Semantic Home Automation

Although WSBPEL is already a higher layer of both Web Service and OSGi, it is still difficult for users to write a WSBPEL document directly by themselves; many details of static binding, such as portTypes or URLs of services, must still be provided. To enable communication based on semantic ontology between different programs across the boundaries of different organizations, Semantic Web [34] technology is designed on the basis of Resource Description Framework (RDF) [35] and Web Ontology Language (OWL) [18]. In general, Semantic Web technology is usually used for context-aware home automation systems. Wang et al. [36] designed the CONON ontology for context reasoning in pervasive computing environments, including home environments. Moreover, several reasoning engines, such as OWL-QL [37], have been developed for understanding semantic meanings of OWL, and can be highly useful in semantic home

systems. For example, if Light L is located in the living room, and the living room is located on the first floor, then the query "all the lights on the first floor" can identify Light L. Furthermore, OWL-S [19] extends the capability of OWL to describe the operations of Web Service and semantic process execution of these operations. Mokhtar et al. [38] developed a QoS-aware dynamic service composition mechanism in ambient intelligence environments, using OWL-S for service description and process definition.

Although OWL-S is based on Semantic Web, it is still not abstract enough for users to express high-level concepts of processes, such as "I want to turn off all the lights on the second floor". Necessary semantic processes should still be defined explicitly by OWL-S. Given a dynamic home environment, for example, after installing a new light on the second floor, the OWL-S process should be modified to meet the requirements of using that light. Ha et al. [10] proposed an infrastructure for a ubiquitous home network service, adopting numerous technologies such as Web Service, WSBPEL, and OWL-S. Moreover, they designed a high level semantic process definition. Processes based on this definition are interpreted to WSBPEL processes dynamically according to the current home environment. However, the proposed semantic process definition language is too simplified, and not user-configurable.

Based on Semantic Web, situation-driven approaches [39–41] provide a higher level of semantic process automation abstraction. The concepts of situation, user goals, and broker goals are proposed so that Web Services can be controlled to fulfill brokers' goals, which further fulfill user goals based on particular situations. Users or experts can control process actions by providing various situational parameters. In the smart home domain, the concept of situations can be mapped to the home environment, such as the lighting, current time, and status of appliances. Although users can control the processes by adjusting the situations, the process definitions, or the user goals, should also be pre-defined. In other words, users cannot create their own customized automation processes.

2.5. User-configurable smart home

Rodden et al. [42] used the idea of Jigsaw to compose abstract processes for smart home automation. For example, a doorbell piece fitting a webcam piece, which then fits a mobile phone piece, implies that when the doorbell rings, the webcam takes a picture and sends this image to the user's mobile phone. This is a compelling idea because it seems highly simple and user friendly. However, this simple representation causes many ambiguity problems. First, the item definitions are ambiguous; if ten lamps are in a house, there must be a manner to distinguish them. Identifying each piece merely using the Jigsaw interface is difficult. Determining how users can define and reconfigure their home automation systems in an understandable manner is crucial. In addition, several abstract concepts, such as "no one is home", are not easily represented; all possible abstract concepts must be defined as pieces before using them. Second, the process definition is ambiguous; the manner in which operations should be executed is not defined. If every item is treated as the same type of piece, three pieces of webcams concatenated one-by-one is possible; however, the meaning of this process is not understandable. Finally, arguments of operations cannot be defined. For example, "turn on the air conditioner if the temperature is higher than 30 °C" cannot be defined because the number 30 cannot be assigned. Moreover, only one precondition piece is allowed to be defined in a process in this system. A trade-off exists between usability and unambiguity; a user interface design that is too simplified is not always understandable to users.

Drey et al. [43] proposed a taxonomy-driven approach to visually prototyping pervasive computing applications, including those of smart homes. This system allows users to create their own semantic automation processes, or rules, based on the sensor-controller-actuator development paradigm. Furthermore, they support the concept of "all" when defining rules using the symbol "*" after entity class names. However, the meaning of "*" differs between the sensor and actuator sides; using "*" on the sensor side implies "any one instance of this category", but implies "all of the instances of this category" on the actuator side. The ambiguous design of "*" may confuse users when designing processes. Moreover, this system supports the concept of clock time only, and concepts of repeated time, such as every day or every weekend, are not supported.

2.6. Conclusion of related works

As previously discussed, although OSGi is usually adopted as the basis of smart home systems, two major problems persist: (1) OSGi bundles cannot be directly accessed from clients outside of the OSGi container; and (2) no standard for automation process definition and execution is designed for OSGi. Therefore, the Web Service and WSBPEL standards are used to solve this interconnectivity problem. Moreover, numerous studies adopted the OWL and OWL-S standards to build context-aware smart home systems. Although OWL-S provides the capability of semantic processes, it is not adequately semantic because many details must still be specified by users. Moreover, neither the OWL-S nor situational approaches are user configurable. Therefore, a new language, SHPL, and its execution runtime are defined and developed for describing and executing semantic customizable automation processes. Finally, the user interfaces of current semantic home automation systems contain several ambiguous concepts and symbols; the proposed system must provide an understandable and user-friendly interface.

3. Design issues

From our observation, numerous home appliances are easy to be operated, even by old people, but others are not. The difference between these operations is that several operations are quite common even over different appliances. For example, almost all appliances have power buttons to switch them on or off. It is quite easy to turn an appliance on by pressing the power button even it is a new kind of appliance that you never see it before. Another example is that it is easy to change the volume of TV by pressing the volume up and volume down buttons on the remote control. If you know how to change volume of TV, it is quite easy to learn how to change volume of an amplifier or any voice-related appliance providing volume up and volume down buttons. Many people, especially old people, feel frustrated if the operations of new devices are totally different with the old ones which they already familiar with. Computer is an example of device providing uncommon operations. Except the power on operation, nothing is similar to traditional controls of appliances; therefore many old people are afraid of using computers.

On the basis of our observation, we attempt to identify certain common operations between different appliances, or within the same category of appliance. We believe that common operations not only provide an understandable interface to users, but also provide a high level of abstraction of control. For example, a command "turn off all devices at the second floor" can be created for power saving if all devices support the operation of "turn off". However, not all devices are the same; several devices, such as multi-purpose washing machines, are valuable because they provide multiple, usually uncommon operations. Another issue is that operations which are not common may become common in the future; therefore, the definition of common operations must be extendable.

Furthermore, this paper used the idea of common operations to design the definition of automation process. Since no common process definition exists for home automation until now, we attempt to identify the pattern of common requirements in smart home environment. A trade-off also exists between simplicity and capability of

process definition; if too many factors are included, the system may frustrate users, but if too little factors are included, some scenarios may not able to be expressed. Finally, the issues of semantic, unambiguous, and user-controllable process definition which are already discussed in Section 2 are also crucial to be addressed.

3.1. System assumptions

On the basis of the scope of this paper and the analysis of design issues, several assumptions for USHAS are defined as follows:

a. All the devices to be controlled have been deployed to users' home environments. For example, lights have been connected to X.10 modules for USHAS to control.
b. Manufactures or third-party driver providers can provide device drivers which are implemented as Web Services and follow the driver standard of USHAS.
c. Operations are more understandable if they are common operations.
d. Automation processes are more understandable if semantic concepts are included.
e. Automation processes are more understandable if they are defined under a common process pattern.

3.2. System overview

As discussed in Sections 1 and 2, USHAS adopts the Web Service technology for exposing services provided by drivers, WSBPEL technology for execution level process definition, Semantic Web technology for knowledge base construction, and a semantic process definition language, SHPL, for semantic process definition. The conceptual stack of USHAS is shown in Fig. 1.

Traditionally, devices can be controlled by two kinds of execution paradigms of smart home: user-controlled execution and event-driven execution. In the proposed system, user-controlled device execution is easily achieved simply by invoking the services provided by device drivers; therefore, this paper focused on event-driven execution. Layers of conceptual stack are introduced as follows:

a. Device layer
Two categories of device are designed: sensor and actuator. Sensors collect environment information, and actuators enforce executions based on commands coming from users or automation systems. In USHAS, both sensor and actuator style devices are encapsulated by device drivers following the Web Service standard.
b. Notification layer
Traditionally, some kinds of agent are designed for receiving sensing data generated by sensors. For example, images captured by cameras are analyzed, and then a face-detected event is generated if someone appears in front of the camera. USHAS adopts the Publish/Subscribe [44] paradigm for publishing events and notifying subscribers based on pre-defined topics, since that Pub/Sub systems are usually employed for event management [45–46].
c. Service query layer and process layer
Since USHAS follows the Web Service standard for service description, it adopts UDDI for service query. Also, USHAS uses the WSBPEL as the execution level process definition language, and a WSBPEL runtime for process execution.
d. Semantic notification layer
Some semantic events such as NoOneAtHomeEvent are defined in this layer. Except explicitly defined events, we found that the concept of semantic event handling is quite similar to the concept of semantic process execution; semantic events can be mapped to pre-conditions of semantic process, and semantic event handling can be mapped to what must be executed in the body of semantic process. Therefore, the semantic notification layer not only handles explicitly defined events, but also includes pre-condition checking before semantic process execution.
e. Semantic service query and semantic process layer
Although OWL-S is not abstract enough for semantic process definition, it is mature for describing and querying services of Web Services. Since USHAS uses OWL as the ontology for knowledge base construction, the OWL-S is adopted for semantic service query. Moreover, semantic process is defined in SHPL, which is described in the next section.
f. Management layer
One of the main goals of USHAS is user-configurable semantic process design; therefore, a management layer for users to create semantic processes must be provided. Also, since home environment differs from user to user, there must be an interface provided for users to define the ontology of their home environments.

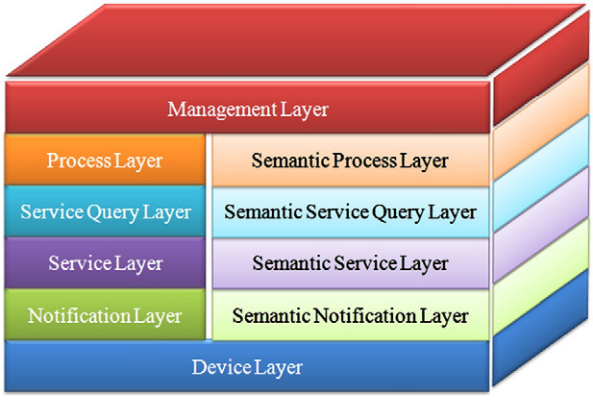

Fig. 1. Conceptual stack of USHAS.

4. USHAS ontology

To describe the status of home, ontology is needed to be defined for home domain; thus, high level information can be maintained, queried, and reasoned. Since a lot of controls and automation processes are usually executed based on the descriptions of user's home, home ontology is usually defined as the skeleton of knowledge base in smart environment systems. The OWL technology is usually used for illustrating home ontology, since it provides a well-defined concept representation model for describing semantic concepts. Wang et al. [36] proposed the CONON ontology, which includes four main classes: CompEntity, Location, Person, and Activity. However, the concept of time is not included. Also, Chen et al. [47] proposed the SOUPA ontology, which includes nine classes in the SOUPA core: Person, Agent, Policy, BDI, Event, Action, Space, Time, and Geo-M. However, since they only focus on meeting rooms, some environment concepts such as Humidity are not included. On the basis of the requirements of USHAS, the USHAS ontology is defined, which is shown in Fig. 2.

Six first-level classes are defined in the USHAS ontology: Person, Device, Time, Environment, Event, and Location. In the Person class, different in-home roles are defined, such as adult member, child member, or even a thief. In the Device class, all the sensor or actuator devices are classified based on the well-known appliance names, such as TV, or their purposes. The Time class includes the definitions of specific time, a range of time, or periodically repeated time (e.g. "Everyday"). Some invisible environmental information, such as brightness or humidity, is classified into the Environment class. Also, some semantic event classes are defined under the Event class. Finally, the Location class maintains the names of spaces, such as living room or first floor in the home domain. Details of some first-level classes are described further as follows.

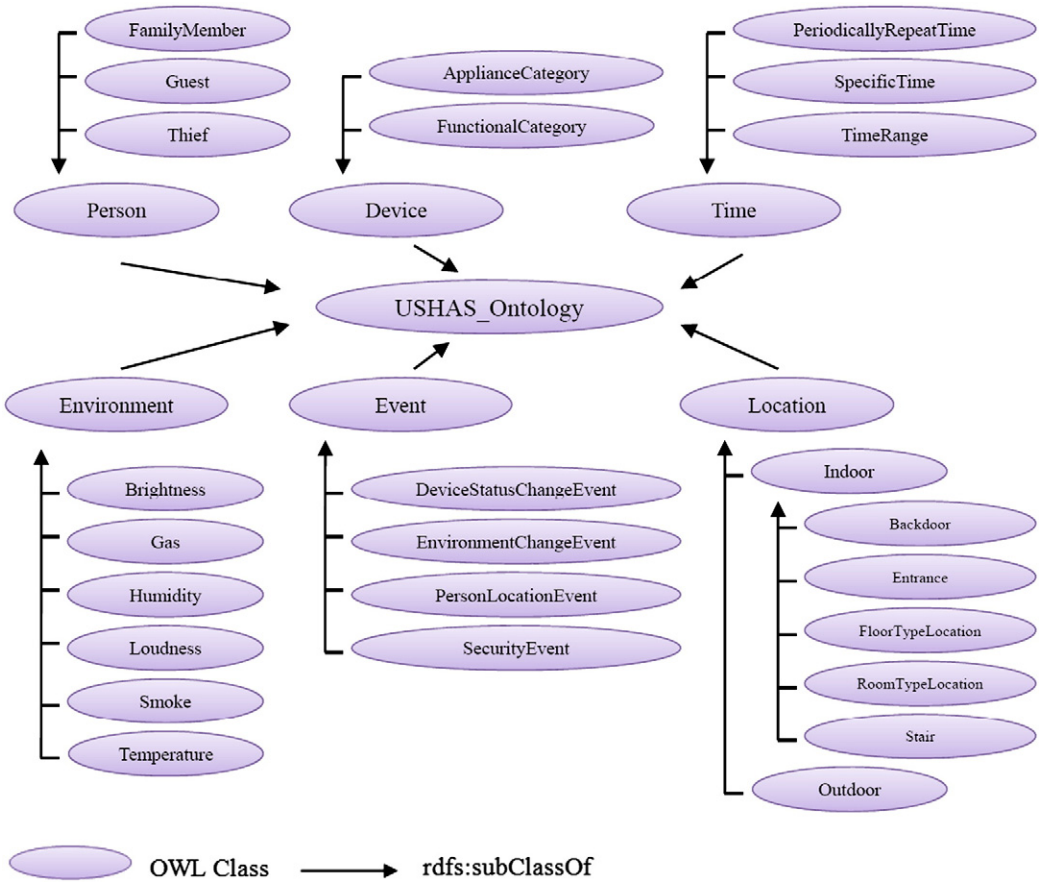

Fig. 2. The high level structure of USHAS ontology.

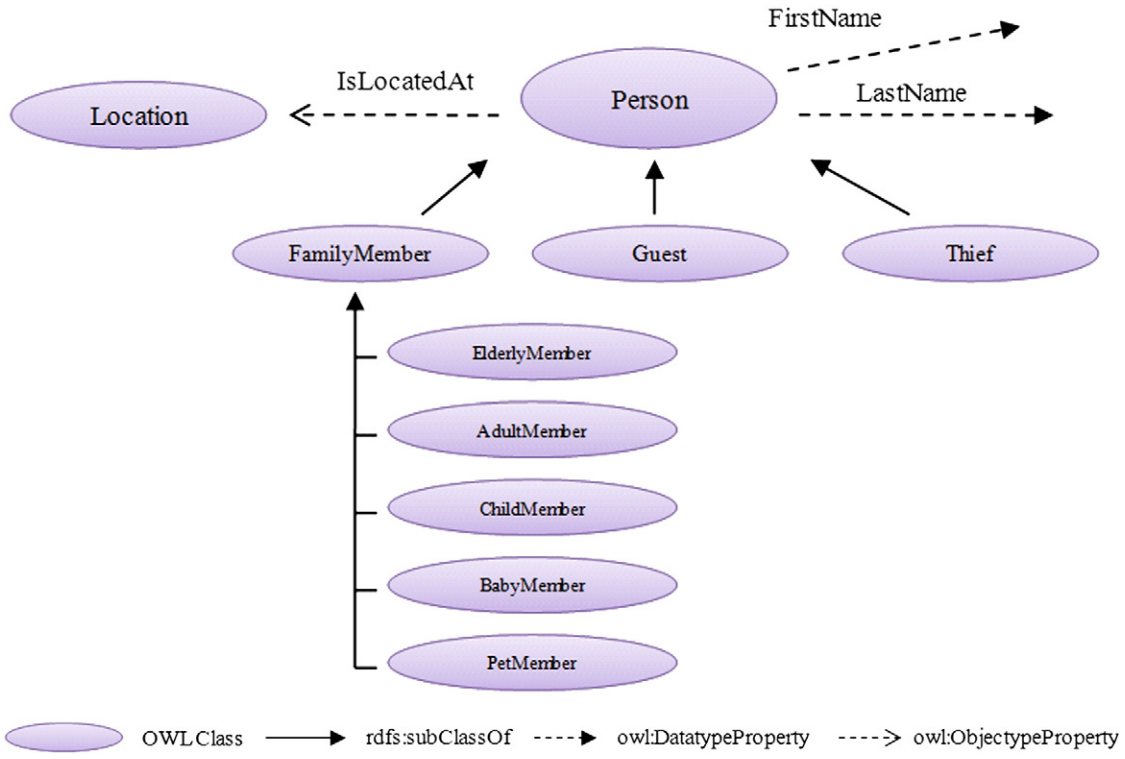

Fig. 3. The low level structure of the Person class.

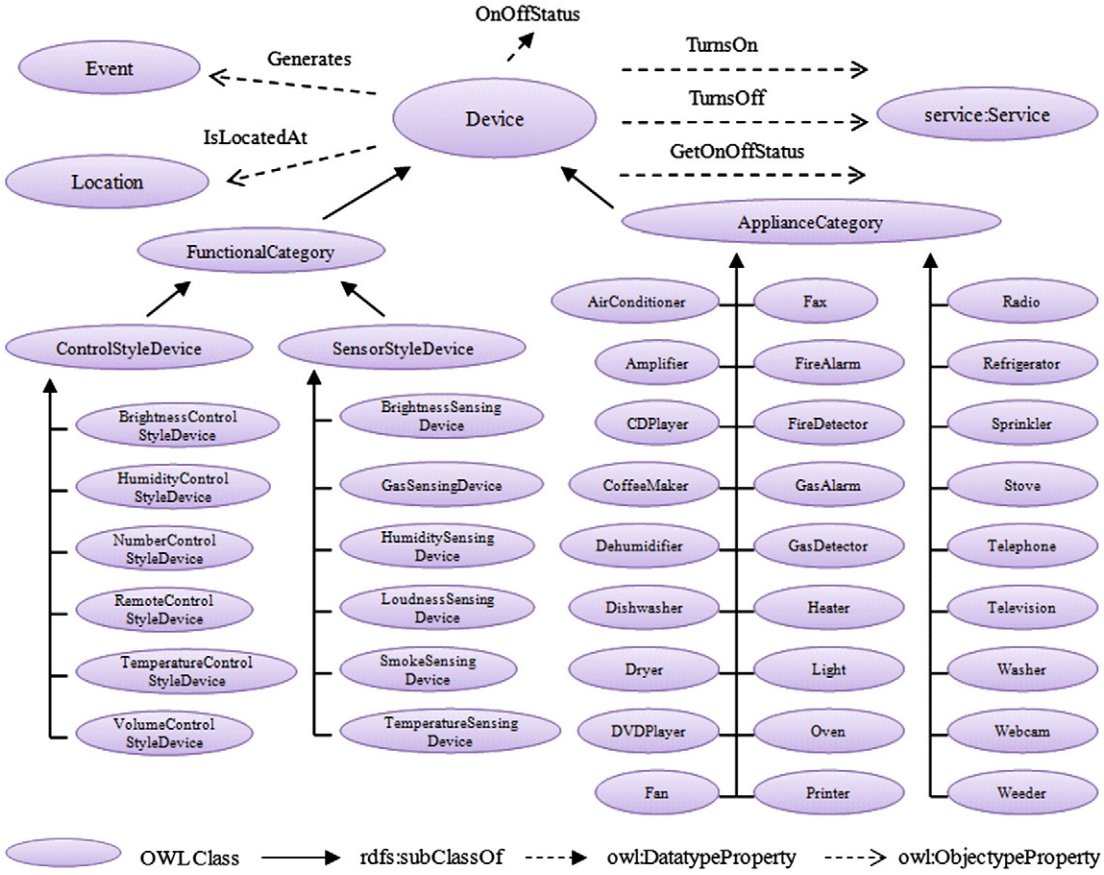

Fig. 4. The low level structure of the Device class.

4.1. Person class

Fig. 3 shows the low level structure of the Person class. The Person class has two data type properties, FirstName and LastName, to record everyone's name. In addition, this class has an object type property for maintaining the location of this person. The FamilyMember class has five sub-classes: ElderlyMember, AdultMember, ChildMember, BabyMember, and PetMember. Many applications are developed based on the age of residents. For example, fall detection usually involves elderly members, and adult members are usually used to ensure that particularly dangerous activities, such as cooking, are safely executed.

4.2. Device class

Fig. 4 shows the low level structure of the Device class. Similar to the Person class, the Device class has an object type property "IsLocatedAt" for indicating the location of this device. The Device class also has a "Generates" property, which maintains an event log for all events generated by this device. Although events are usually generated by sensor devices, generating events is not limited to sensor-style devices; for example, the TV is not a sensor device, but it is allowed to publish a "DeviceStatusChangeEvent" if someone changes the channel. More details of event are described in the Event class section.

Three basic service pointers, "TurnsOn", "TurnsOff", and "GetOnOffStatus", are maintained by this class. In OWL-S, each operation of a Web Service is encapsulated as an OWL-S service (service: Service) for invocation. In other words, each Web Service of device must support at least three operations for turning it on, turning it off, and querying its current on–off status. The reason for this requirement is that the on–off status is the most basic and fundamental information of a device. For some simple devices, such as a lamp, provide only the on–off functionality for users to control; for other complex appliances, such as a TV or air conditioner, users also have to turn them on before using them. Controlling the on–off status is the most common and vital functionality shared by all devices at home. With these three definitions, USHAS can turn on devices, turn off devices, or query the on–off status of devices to maintain the "OnOffStatus" property in a batch process.

Two sub-classes, FunctionalCategory and ApplianceCategory, are defined under the Device class. The FunctionalCategory is further sub-divided into the ControlStyleDevice and SensorStyleDevice. The SensorStyleDevice is designed for sensor devices. Other appliances and devices can be classified by the common product name or the category of their functionalities. By defining the category or product name, similar products can be controlled in a batch process; for example, we can turn on all air conditioners in the home, even the brand names are different. Conversely, by defining the category of functionality, various types of products with shared functionalities can be controlled in a batch process; for example, we can set the audio volume of all appliances at zero after midnight. Although two different manners of classification are available, one device can belong to both of them simultaneously. For example, a TV belongs to the Television class under ApplianceCategory, and also the NumberControlStyleDevice under FunctionalCategory.

4.3. Event

In general, events are generated by agents after analyzing the measurements of sensors. For example, BrightnessSensingDevice

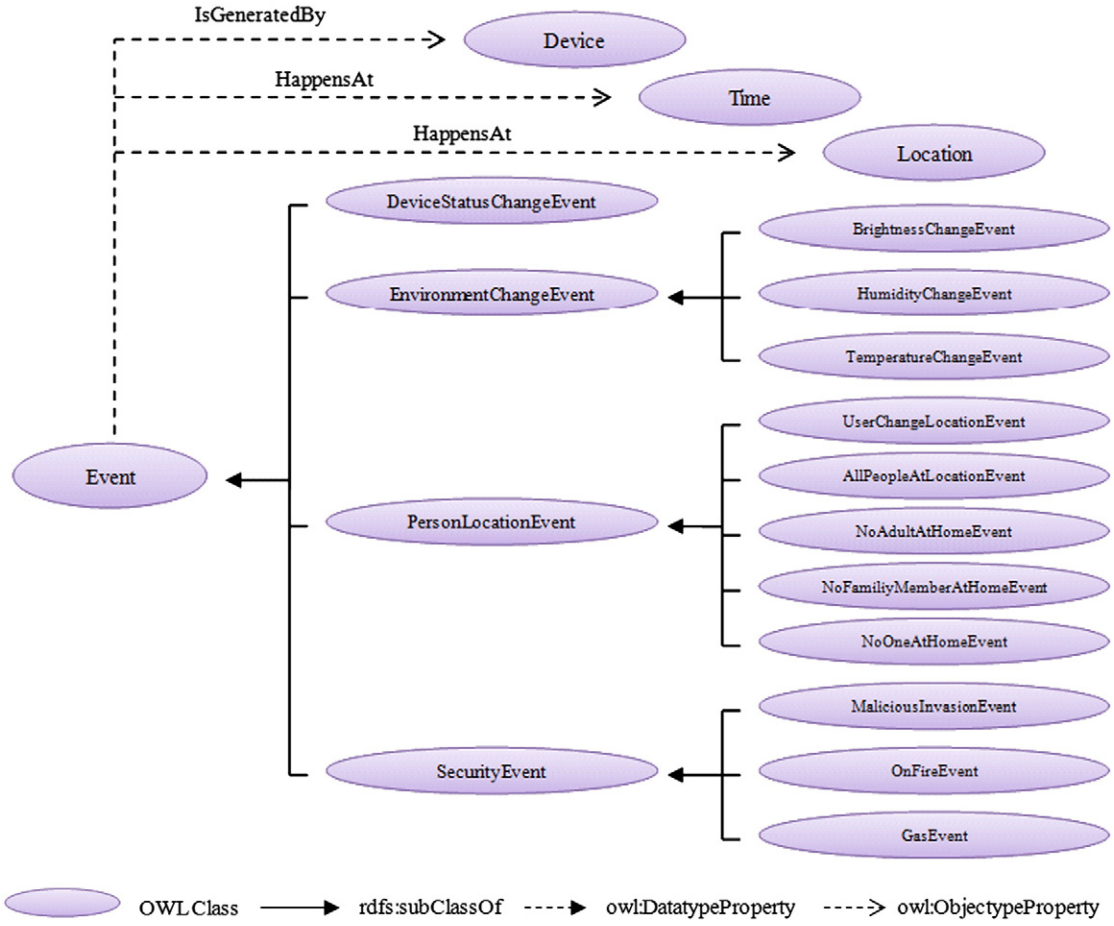

Fig. 5. The low level structure of the Event class.

generates BrightnessChangeEvent. Moreover, semantic events are also defined under the event class. For instance, the NoAdultAtHome event is generated, based on the presence of each person and the definition of the Person class at home (Fig. 5).

5. Semantic Home Process Language (SHPL)

Since the semantic process definition of OWL-S is not abstract enough, and no higher level process description based on it exists until now, we decide to define a new process description language, Semantic Home Process Language (SHPL), to support semantic process definition without static service binding.

From our observation, four vital factors are usually included in home automation processes: pre-conditions, variables, execution time, and flow of invocations. For example, given a command "if the temperature is higher than 30 °C at 6:00 PM, turn on the air conditioner", the value 30 is a variable; "if the temperature is higher than 30 °C" is a pre-condition, 6:00 PM is an execution time, and "turn on the air conditioner" is an invocation. Although post-condition is usually defined by the generic process description, in home automation situations, users usually know what the post condition is after executing each invocation; therefore, we can eliminate this definition to decrease the complexity of SHPL. The high level XSD of SHPL is shown in Fig. 6.

5.1. Variables

The design of variables provides users a manner to express information that can be input by users when defining pre-conditions and invocations. For example, we can define a variable with type integer and value 10, and then let it be the input of the "*set_channel*" invocation of the TV. The formal expression of the "*variables*" element is shown in Fig. 7.

Only one "*variables*" element is in each process, which can contain several "*variable*" elements. Each variable element has three attributes: name, type, and value. In SHPL, only four basic variable types are supported: Boolean, integer, double, and character string. The reason why only these types are supported is because they are the most understandable types for people to input. In general, home appliances do not require complex input, because inputting information is difficult if it is complex. Many operations, such as switching TV channels,

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema elementFormDefault="qualified"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="SProcess">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="variables"/>
        <xsd:element name="time_set"/>
        <xsd:element name="preconditions"/>
        <xsd:element name="flow"/>
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:string"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Fig. 6. The high level XSD of SHPL.

Element_variables:=<variables> *Element_variable** </variables>
Element_variable:= <variable *Attribute_name* *Attributes_type_value*/>
Attribute_name:= name= "*XSD_string*"
Attributes_type _value:= (type = "xsd:boolean" value = "true | false") |
(type = "xsd:int" value = "*XSD_int*") |
(type = "xsd:double" value = "*XSD_double*") |
(type = "xsd:string" value = "*XSD_string*")

Fig. 7. Formal expression of the variables element.

Element_time_set:=<time_set> *Element_time* * </time_set>
Element_time:= <time> *time_name* </time>

Fig. 8. Formal expression of the time_set element.

receive numbers as inputs; most operations, such as turning on lamps, require no input from users. Although string-type input is rarely supported by home appliances, it is understandable for users to input string-type with particular devices in the future home environment. For example, users may be able to key-in the name of a movie on a mobile phone, and the DVD player would then start displaying the movie in the living room. Therefore, the type string is also supported for a higher flexibility under the concern of usability.

5.2. time_set

The "*time_set*" element maintains all time instances, which are defined under the Time class of the USHAS ontology, and are execution time of processes. The relationship between time instances is conjunction; in other words, only when all time elements are satisfied will the process be triggered. Although the conjunction relationship limits the capability of expression, it simplifies the logic of *time_set*; when both conjunction and disjunction relationships are permitted, it is too complex for users to define and comprehend. If users wish to define a process using the disjunction relationship of time instances, they can define multiple processes using different execution time instances instead. The formal expression of the *time_set* element is shown in Fig. 8.

5.3. Preconditions

Each "*preconditions*" element is allowed to contain several "condition" elements. Similar to "*time_set*", the relationship between different "*condition*" elements is also conjunction; thus, only when all conditions are satisfied will the process be triggered. Similar to the reason for using only conjunction in time_set, users may be confused if both the conjunction and disjunction are supported in preconditions. For example, given the process "if the room temperature is lower than 20 degrees and Mary is in the living room or the baby is at home, then, switch on the heating", the conditions can be explained in two different ways: "if the room temperature is lower than 20 degrees and Mary is in the living room" or "the baby is at home" and "if the room temperature is lower than 20 degrees" and "Mary is in the living room or the baby is at home". As a result, users may create unfavorable processes due to the complex logic combination. When only conjunction is allowed, users can still create multiple processes such as "if the room temperature is lower than 20 degrees and Mary is in the living room, then, switch on the heating" and "if the baby is at home, then, switch on the heating" for achieving the same goal. The formal expression of the "*preconditions*" element is shown in Fig. 9.

The description of each condition contains three main parts: domain, property, and range. In other words, if particular subjects have properties with values, the condition is satisfied. Moreover, the domain can be further described by the "*domain_quantifier*" and "*domain_type*". The "*domain_type*" can be a category or an individual. For example, we can specify a person or the Person class as a domain. The "*domain_quantifier*" is the quantifier of a domain; it is specified only when the "*domain_type*" is a category. For example, we can specify "all people in the FamilyMember class" or "at least one instance of the OnFireEvent exists" as the subject. The property attribute can be any supported property of a specified domain; if the "*domain_type*" is a category, only the shared property of this category is allowed to be defined. Moreover, four kinds of range_type: category, individual, variable, and range, are defined. The "*individual*" and "*variable*" are the two most basic types; the "*individual*" type is used for values of object properties, and the "*variable*" type is used for values of data properties. The range type is used when specifying a range of value. For example, we can specify a range of temperature value in the range_type. Finally, if the category type is specified as range_type, all instances belonging to this category are examined. For example, the pre-condition "all family members are in the bedrooms" is satisfied even when family members are located in different bedrooms.

5.4. Flow

Each flow element is allowed to contain several invoke elements, which enable users to specify which operations of which devices are controlled. In most home automation scenarios, only actuators are controlled; therefore, we mainly focus on invoking devices. If the home automation system providers intend to provide some special agents to be invoked, such as MMS sender, they can implement these agents as special devices following the general device interface (Fig. 10).

Similar to pre-conditions, users can also specify whether all devices are, or only one device of the device category is controlled. We

Element_ preconditions := <preconditions> *Element_ condition* * </preconditions>
Element_condition := <condition *domain_modifier*
domain = "*domain_options*"
property = "*property_options*"
range_type = "*range_type_options*"
range_value = "*range_value_options*"></condition

domain_modifier := (domain_quantifier = "null" domain_type = "individual") |
(domain_quantifier = "*domain_quantifier_options*"
domain_type = "category")
domain_quantifier_options := none | all | exist
range_type _options := category | individual | variable | range

Fig. 9. Formal expression of the preconditions element.

Element_flow := <flow> *Element_invoke* * </flow>
Element_invoke := <invoke domain_quantifier= "*domain_quantifier_options*"
category = "*category_options*"
device_name = "*device_name_options*"
location_type = "*location_type_options*"
location = "*location _options*"
operation = "*operation _options*"
variable = "*variable _options*"></invoke >
domain_quantifier_options := all | one
location_type _options := category | individual

Fig. 10. Formal expression of the flow element.

observe that devices are usually specified associated with their locations with their categories, such as "the lights in the living room". Therefore, the location and category of controlled device are also included. Finally, the operation supported by the specified device is also included into the invoke element with input variable defined in the variables element.

5.5. SHPL example

Fig. 11 shows an SHPL example of the scenario "if anyone is in the dining room, and there is no one on the second floor at dinner time, then turns off all lights on the second floor, and turns on the TV in the living room". The variable "input1" is a simple variable with a true value for confirming the operation execution. The time "DinnerTime" must be defined as a part of the home environment before creating this process. Two preconditions are specified in this process: the first has the Person category domain and the SecondFloor category range, and the second has the Person domain and the MyDiningRoom individual range. Two operations are executed when these two preconditions are satisfied; the first is the TurnsOff for all Light categories on the SecondFloor, and the second is the TurnsOn for MyTV in the LivingRoom.

6. System architecture and detailed system design

6.1. System architecture

The system architecture of USHAS is shown in Fig. 12. For each device to be controlled, a Web Service driver must be implemented based on the protocol and functionalities of this device by device manufacture or third-party developers, such as TV-WS, Light-WS, and so on. Also, detail binding configurations such as house code and key code, must be able to be configured by users via these drivers. For WSBPEL automation, a BPEL runtime is included, which is able to control device drivers based on the BPEL process descriptions. Similarly, for semantic process automation, a semantic process runtime is included, which is able to generate BPEL process dynamically based on current home environment.

To maintain all the home knowledge and status of home environment, a knowledge base following the OWL-S standard is included together with the manager of it. Within the knowledge base, semantic Web Service information, location definition, people definition, time definition, and device definition are maintained. Although semantic Web Service information and device definition are both related to devices, the device definition is a higher level of representation, such as device category, which is user understandable, while the semantic Web Service information is a lower level of representation, such as binding information, which is not user understandable, and is not managed by user directly.

The execution of semantic automation process depends on the decision of the Semantic Process Manager. The Semantic Process Manager constantly checks whether the current time is satisfied with any time instance defined in the semantic processes. Moreover, the Semantic Process Manager checks whether all the pre-conditions are satisfied in each semantic process once the Home Ontology has been updated. Also, for each semantic process, an execution flag is set to indicate whether this process has been executed. Only when all the pre-conditions and time instances are satisfied when the process has not been executed, the process will be

```
<?xml version="1.0" encoding="utf-8"?>
<SProcess name="turn off lights on 2nd floor and turn on tv"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="SHPL_Schema.xsd">
    <variables>
      <variable name="input1" type="xsd:boolean" value="true"/>
    </variables>

    <time_set><time>DinnerTime</time></time_set>

    <preconditions>
      <condition domain_quantifier="none" domain_type="category"
                domain="Person" property="IsLocatedAt"
                range_type="category" range_value="SecondFloor"></condition>
      <condition domain_quantifier="exist" domain_type="category"
                domain="Person" property="IsLocatedAt"
                range_type="individual" range_value="MyDiningRoom"></condition>
    </preconditions>

    <flow>
      <invoke domain_quantifier="all" category="Light" device_name="null"
                location_type="category" location="SecondFloor"
                operation="TurnsOff" variable="input1">
      </invoke>

      <invoke domain_quantifier="one" category="Television" device_name="MyTV"
                location_type="category" location="LivingRoom"
                operation="TurnsOn" variable="input1">
      </invoke>
    </flow>
</SProcess>
```

Fig. 11. An example of using SHPL.

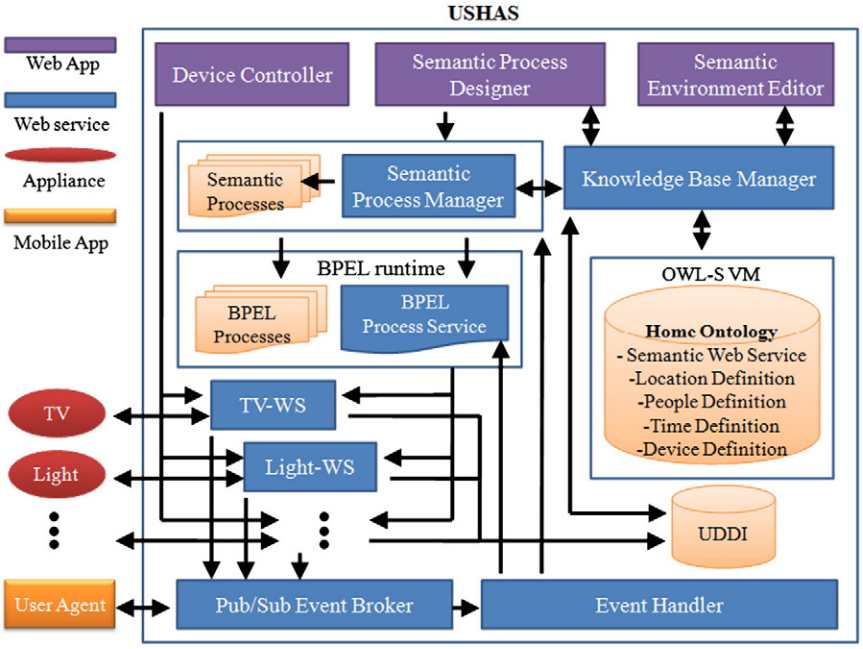

Fig. 12. System architecture.

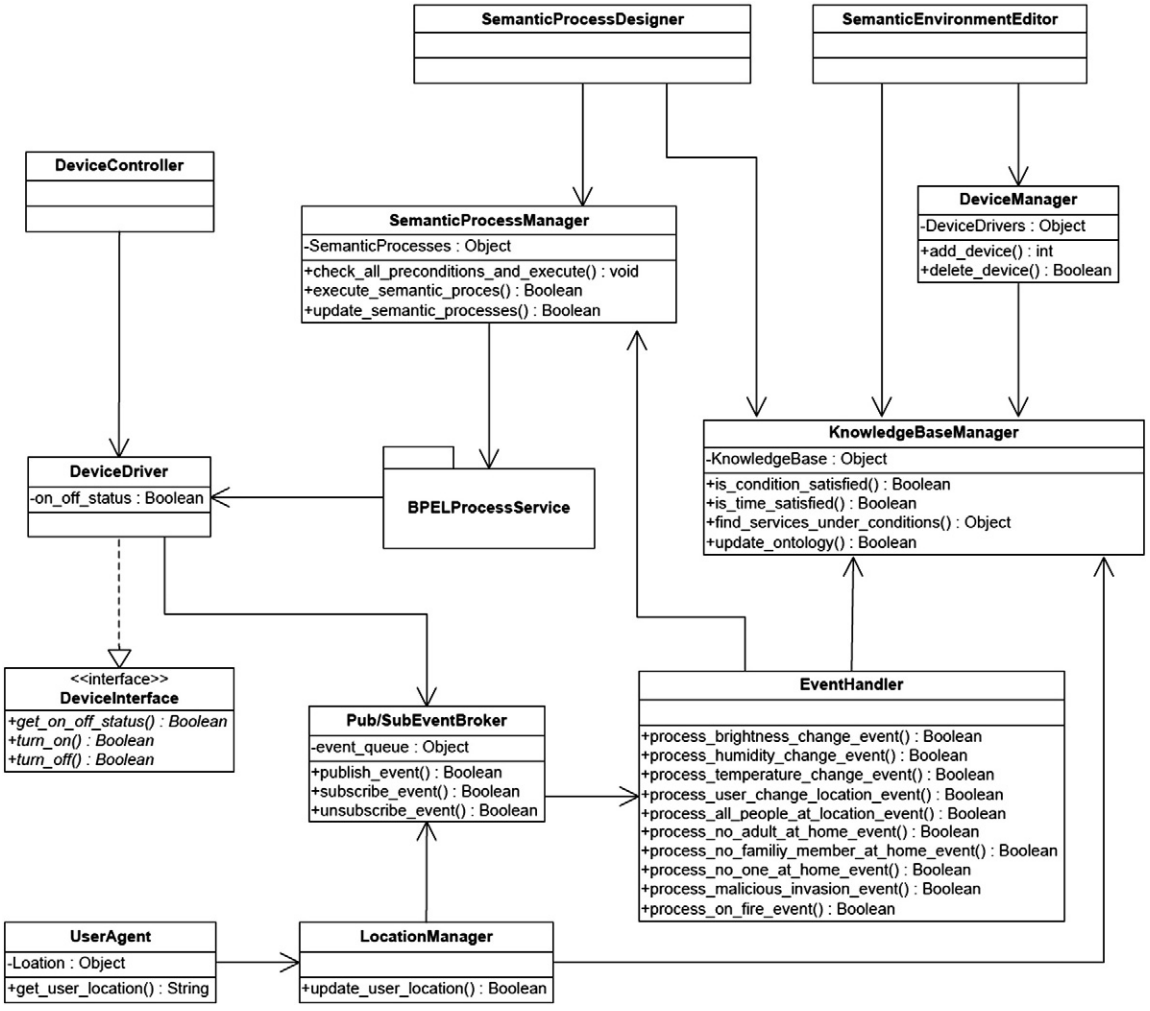

Fig. 13. System model.

executed. The design of this execution flag is important, since users may want to change the result of execution when the pre-conditions and time instances are still satisfied. For example, if no execution flag is set when executing the process "If there is anyone in the living room from 7:00 PM to 9:00 PM, turn on the TV and set the channel to 20", the TV channel will always be 20 during 7:00 PM to 9:00 PM even users intend to change it. In this manner, this is an annoying process. Therefore, only when the execution flag is set as "not has been executed", the process can be executed. After the process is executed, this flag will be set as "has been executed", and, it will be set as "not has been executed" again if any of the time instance or pre-condition is not satisfied. After the Semantic Process Manager decides to execute a semantic process, it translates the flow of this process into a WSBPEL process based on the current home environment, and then invokes the BPEL process service for WSBPEL process execution.

For event handling, firstly, both kind of explicitly defined basic events and explicitly defined semantic events must be published to the Pub/Sub Event Broker. Developers can create their own services to handle events by subscribing to the Pub/Sub Event Broker, otherwise, all the pre-defined events are sent to the Event Handler which invokes the BPEL runtime directly for basic events, and invokes the semantic process runtime for explicitly defined semantic events. Also, a user agent is designed to send location information and update users' real-time locations.

Finally, three user interfaces are provided via HTTP. The Semantic Environment Editor facilitates users to define information of home environment, such as room definition, family member definition, and time definition. Also, users can deploy bundles of devices drivers via this editor. For defining semantic processes, the Semantic Process Designer is included, which contains variable definition, pre-condition definition, invocation definition, and so on. Options provided in the Semantic Process Designer are basically provided depend on the definition of current home environment, which is based on the input coming from the Semantic Environment Editor and real-time events.

As to user interface, several home automation systems also include the maps of the homes. However, the problem of providing a map is that the structures of users' houses are usually different; users must construct their own maps before using them; it is not an easy task for users. Actually, from our observation, users do not need maps of their house for specifying and understanding the locations within their houses; people usually use semantic properties of locations for doing so, such as "the living room" or "the bath room on the second floor". Therefore, we decide not to include the map interface into the proposed system.

6.2. System model

The system model is shown in Fig. 13. All the device drivers must implement the three methods (get_on_off_status, turn_on, and turn_off) defined in the DeviceInterface interface. For each category of appliance, an interface is also defined for different functionalities of that category, such as the "set_channel" method defined in the TVInterface interface. To simplify this figure, numerous detailed definitions of these categories and functionalities are not included.

A LocationManager class is designed for translating physical location information into semantic information. This class can be implemented differently based on different location detection systems. For example, if an RFID tag is deployed in each room to transmit the ID of the room, and a table is designed to store the mapping between this ID and the semantic name of this room, then the LocationManager can be implemented as the manager of this table. For another example, if someone is recognized by a face recognition system from a camera, and this camera is located at the living room, then this person can be realized to be in the living room too.

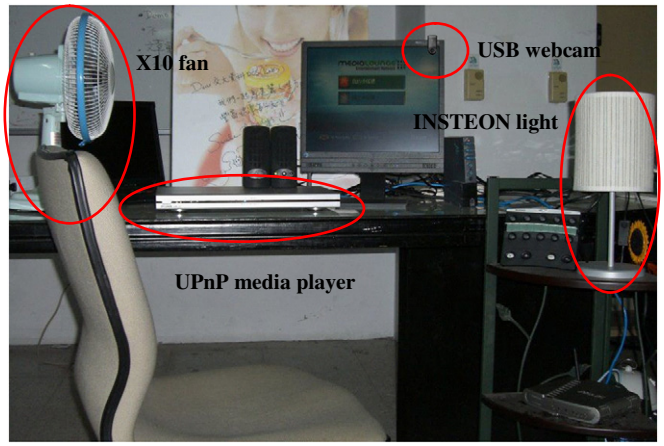

Fig. 14. The living room of UAHAS prototype.

Finally, the DeviceManager is the installer and uninstaller of device drivers; this class manages all the physical files and metadata of all the installed drivers. Also, an OWL-S service is automatically generated by DeviceManager during each operation of driver installation.

6.3. Limitation analysis

Because almost all system modules are designed as Web Services, external systems or third-party designed modules must be able to interact with these modules easily. However, currently, we only design the functionalities for USHAS modules, not for other systems. Therefore, it is possible that several functionalities required in other systems are not provided in USHAS. For example, a facial recognition system integrating facial training and recognition operations into the USHAS for family members is difficult to design because no facial database exists in the USHAS.

In addition to the aforementioned difficulties, process conflicts may exist in the current design. For example, if a process "if Light A is on, turn off Light B" is defined with another process "if Light A is on, turn on Light B", then the status of Light B depends on which process is defined later. In this scenario, because a semantic conflict exists between these two processes, the execution result may not be the same as the needs of the user. Several models, such as Petri Nets [48] and Colored Petri Nets (CPN) [49], can be used as verification

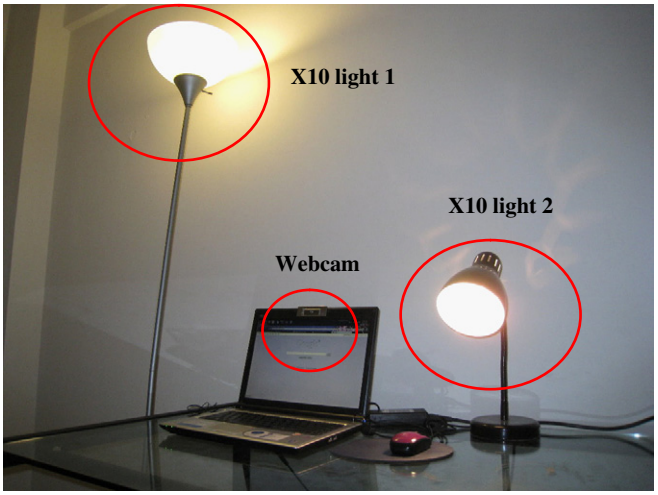

Fig. 15. The bed room of UAHAS prototype.

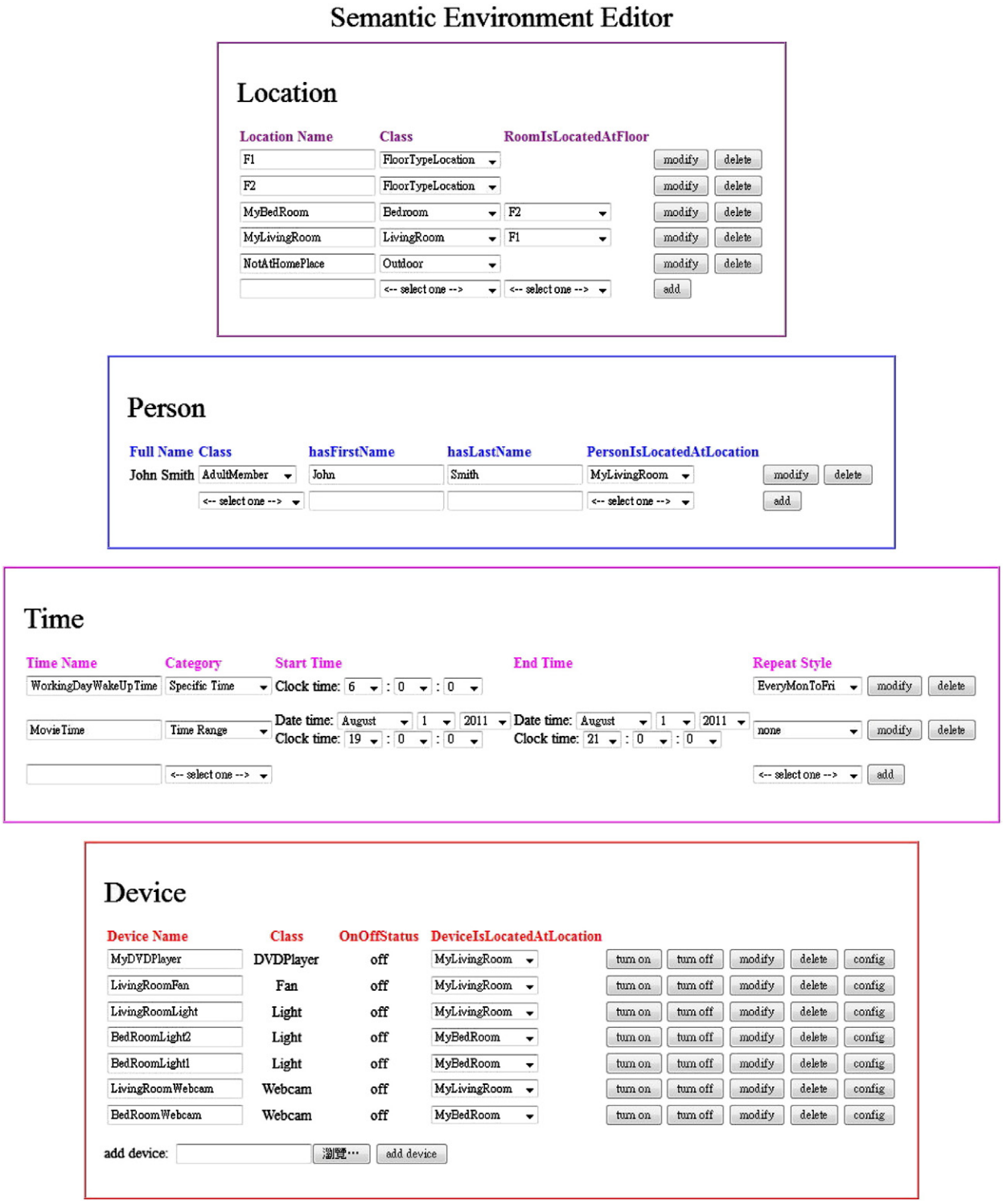

Fig. 16. Semantic Environment Editor.

tools to solve this problem. However, in the current design of USHAS, we simply assume that users are careful enough to avoid conflict in semantic processes.

Another limitation of USHAS is the security concern. Security is absolutely essential in the home environment. Various levels of security concerns are related to different types of security threats. The lowest security level is device level security: devices being controlled by no one other than the trusted home server or people must be guaranteed. Guaranteeing the security of the entire system is extremely difficult, unless devices are uniformly designed with security functionalities, such as providing a trust engine based on the framework designed by Seigneur et al. [50]. The second level is home network security. Network components such as firewalls and proxies must be configured properly so that devices and servers can communicate with

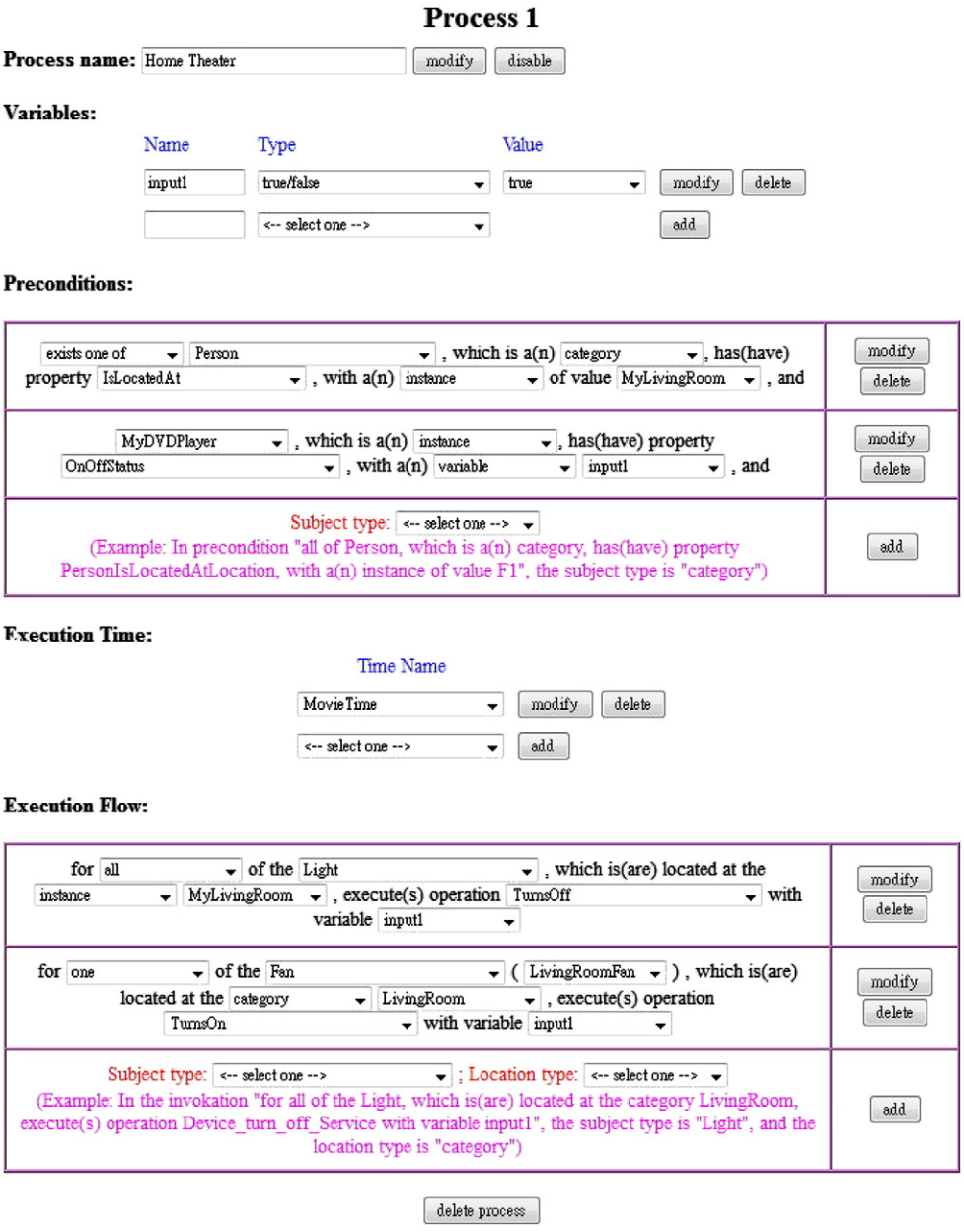

Fig. 17. A semantic process of the USHAS prototype in Semantic Process Designer.

each other within a proper and safe environment. For example, Fitzgerald et al. [51] proposed their system architecture to perform network-based access control between semantic Web services by configuring firewall rules. The third security level is home server security: home servers should be well protected to be controlled only by trusted users. Some host-based mechanisms, such as TrustVisor [52], can be adopted to ensure that only trusted software can be executed under the control of hypervisor. The forth security level is personal privacy protection. Because webcams may be deployed throughout the home environment for facial recognition and person identification, privacy must be guaranteed for activities such as changing clothes in the bedroom. A possible solution involves installing an application onto a PC or IP camera that negotiates a secret key with a home server and encrypts each frame before transmission. The final security level is user authentication and access control. People do not usually want their home appliances to be controlled by those whom they do not trust, but they may be willing to share some control with relatives or close friends. For example, a user's friend allowed to change the channel on the TV can do so using a mobile phone. To integrate semantic Web service and security, several recent studies [53, 54] have defined certain semantic security ontologies and performed rule-based access control based on these ontologies. These matters are quite complex and out of the scope of the proposed system; therefore, solutions for solving these security concerns have not been included into USHAS currently.

7. System prototype and evaluation

7.1. System prototype

In the system prototype of USHAS, almost all the core modules, such as the Knowledge Base Manager, Semantic Process Manager, and Pub/Sub Event Broker, are implemented as Web Services running

```
<?xml version="1.0" encoding="utf-8"?>
<SProcess name="Home Theater"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="SHPL_Schema.xsd">
    <variables>
      <variable name="input1" type="xsd:boolean" value="true"/>
    </variables>

    <time_set><time>MovieTime</time></time_set>

    <preconditions>
      <condition domain_quantifier="exist" domain_type="category"
                domain="Person" property="IsLocatedAt"
                range_type="individual" range_value="MyLivingRoom"></condition>
      <condition domain_quantifier="null" domain_type="individual"
                domain="MyDVDPlayer" property="OnOffStatus"
                range_type="variable" range_value="input1"></condition>
    </preconditions>

    <flow>
      <invoke domain_quantifier="all" category="Light" device_name="null"
                location_type="category" location="MyLivingRoom"
                operation="TurnsOff" variable="input1">
      </invoke>

      <invoke domain_quantifier="one" category="Fan" device_name="LivingRoomFan"
                location_type="individual" location="LivingRoom"
                operation="TurnsOn" variable="input1">
      </invoke>
    </flow>
</SProcess>
```

Fig. 18. An example of using SHPL in the prototype.

on a Apache Tomcat Web server of version 5.5.27, so that functionalities of these modules can be shared easily to other modules or even external systems. Among the existing mature BPEL runtime systems, this paper used the open source ActiveBPEL [55] system. The Semantic Process Designer, Semantic Environment Editor, and Device Controller user interfaces are implemented in JSP; they are also deployed on the Tomcat server.

Constructing a complete smart home environment requires a lot of space and smart devices. Initially, the prototype of USHAS only contains two pseudo rooms with some controllable devices for demonstration. The first floor living room of USHAS prototype, a fan controlled by X10 module, an UPnP media player, an INSTEON light, and an USB webcam are deployed; in the second floor bedroom, two X10 lights and a webcam are included. Moreover, to simplify the user location detection, only one person is defined in this prototype, and the presence of this person is detected by the webcams based on moving detection (Figs. 14 and 15).

Two semantic processes are deployed in the USHAS prototype. First, if there is someone in the living room, and the media player in the living room is turned on, then turn off all the lights and turn on the fan in the living room. Second, if there is no one in the bedroom at noon, then turn off all the lights in the bedroom. As a result, these two processes work perfectly in the USHAS prototype. The definitions of the home environment and the first process are shown in Figs. 16 and 17.

The Web user interfaces of Semantic Environment Editor and Semantic Process Designer are shown in Figs. 16 and 17. By using the Environment Editor, users can create instances of locations, people, time, and devices according to the actual home environments. Moreover, users can install device drivers, configure the detailed binding information, and control the on/off status of devices. Next, users can create their own processes based on these instances in the Semantic Process Designer. To avoid the ambiguous meanings of pictures, we use English sentences with some replaceable words for defining semantic processes. Each sentence provides all the information that SHPL needs, so that we can easily translate theses sentences into SHPL documents. Moreover, we believe that users prefer selecting options than filling blanks; therefore, we list all the legal options for each replaceable word. Whenever each replaceable word is modified, all the other dependent words are checked again; if any illegal result exists during modification, the dependent words are set to null for users to choose again. In this manner, we can ensure that all the sentences are correct. Finally, an example of using the SHPL in the prototype is shown in Fig. 18.

Table 1
Examples of different scenarios and corresponding semantic automation processes.

Scenario	Semantic automation process
Long vacation	During the period of vacation time (from Aug. 1st to Aug. 10th), turn on all the lights in the house at 7:00 PM, and turn them off at 11:00 PM every day.
Home gym	If there is anyone in the gym room, then turn on the light and TV in the gym room.
Morning rush	From Monday to Friday every week, at 7:00 AM, turn on all the lights of bedrooms and bath rooms, and the coffee maker starts to make coffee.
The living room	If the temperature in the living room is higher than 30.0 °C, then turn on air conditioner in the living room and set the temperature of air conditioner as 26.0 °C.
Dinner time	If any family member is in the dining room at dinner time, then turn on the TV in the living room, and set the channel of TV as 20.
Good student	If there is someone in the reading room during 9:00 PM to 11:00 PM every day, set the volume of all the volume style devices at home as 0.
Sweet dreams	If all the family members are located in bed rooms during 0:00 AM to 7:00 AM every day, turn off all the lights in living room, gym room, kitchen, reading room, and bath room.
Bath time	If there is a gas event published by the gas sensor in the bath room, then turn on the gas alarm.
Party night	At 10:00 PM on Sep. 12th, 2010, turn on the party light and the DVD player in the living room.

7.2. System scenarios

To analyze the user satisfaction and user usability of USHAS, we create some examples of automation processes for different scenarios; these examples also can be defined through the USHAS Web interface. Since the capability of device control has been shown in the USHAS prototype, we only focus on the expression of SHPL and usability of USHAS with some simulated devices when designing these scenarios and processes. The variety of scenarios guarantees that USHAS and SHPL are able to support a large enough amount of cases which users may need. These scenarios and corresponding processes are listed in Table 1.

7.3. User satisfaction evaluation

The user satisfaction analysis is shown in Table 2. We conducted a survey over 136 people in Taiwan who are randomly chosen on the Internet. Each one of them can select more than one scenario which they are interested in. The result shows that almost all the people are satisfied with at least one scenario and the corresponding example; the ratio is 98.53%. Even for the scenario which gets the lowest ratio, Home GYM, 48 people are interested in it. The reason why this scenario gets the lowest count maybe is that people in Taiwan usually do not have the habit of home exercise.

7.4. Usability evaluation

To analyze the usability of USHAS, we conduct a preliminary survey based on the system scenarios defined in 7.2. Twenty of 136 people in 7.3 are willing to do the questionnaire of using USHAS. Totally, 7 locations, three people, 8 time definitions, 11 devices, and 10 processes (the "Long Vacation" scenario requires two processes) must be defined by each one of them. The average spent time, average difficulty (1 for very easy and 10 for very hard), and average accuracy rate of using the Semantic Environment Editor have been listed in Table 3. The accuracy rate is defined as the number of correctly created instances over the number of all the processes which are required to be defined. The correctness of each instance is judged based on whether it makes the scenario work; for example, using different name of device is not counted as incorrect definition. In general, people can define instances within a little time and feel easy to do so with high accuracy rate.

Similar to the Semantic Environment Editor, the Semantic Process Designer is also analyzed according to the average spent time, average difficulty, and average accuracy rate of using it. The accuracy rate of process creation is defined as the number of correctly specified inputs over the number of all the inputs required for each process. Moreover, the type of each process is also listed in Table 4; here the type indicates the most difficult part of each process. For the nine scenarios, most of them are considered easy and created within a short time with high accuracy rate, while some of them are difficult with longer time and lower accuracy rate. In general, users feel easy to specify processes which control the on–off status of devices; this kind of control is also the most common and intuitive one matching the daily home control experiences. The second easy type of process is the controls with variables. We find that some people do not have the sense of variable, especially for those who do not have science backgrounds. Supporting multiple type of variable is complex to some people, although it provides the flexibility for different kind of operation. Finally, the most difficult kind of process is the event handling type process. Compared with variable, fewer people have the sense of event, and more people failed to create a correct precondition with event included. Actually, we provide two ways to make a correct precondition for the "Bath Time" scenario; they are shown in Fig. 19. The precondition of "Bath Time" can be specified from either the device's or event's point of view; however, many people still cannot make it correct. Even though some of the processes are not easy to everyone to specify, the result of overall average spent time, difficulty and accuracy rate are satisfied.

Table 3
Analysis of usability of Semantic Environment Editor.

	Average spent time (per instance) (min)	Average difficulty (1–10)	Average accuracy rate
Location definition	3.25 (0.46)	1.7	100%
Person definition	2.15 (0.717)	1.7	100%
Time definition	6.9 (0.8625)	1.7	98%
Device definition	6.95 (0.63)	1.5	100%
Average	4.81 (0.66)	1.23	100%

Table 2
Analysis of user satisfaction.

Q: Which scenario(s) and the corresponding example(s) you are interested in?		
Answer	Count	Ratio
Long vacation	116	85.29%
Home gym	48	35.29%
Morning rush	96	70.59%
The living room	90	66.18%
Dinner time	54	39.71%
Good student	74	54.41%
Sweet dreams	120	88.24%
Bath time	87	63.97%
Party night	51	37.50%
None	2	1.47%

7.5. System comparison

The comparison of USHAS and many related systems mentioned in Section 2 are shown in Table 5. We compare them in six dimensions: automated process execution, event handling, semantic process automation, automation of semantic process which allow the "all" quantifier, user configuration, and repeated time, and check that these

Table 4
Analysis of usability of Semantic Process Designer.

	Process type	Average spent time (min)	Average difficulty (1–10)	Average accuracy rate
Long vacation	On–off control	3.15	2.1	92.5%
Home GYM	On–off control	3.8	2.35	99%
Morning rush	On–off control	3.2	2.1	97%
The living room	Control with variable	7.95	8.35	76%
Dinner time	Control with variable	7.45	6.35	88.25
Good student	On–off control	4.55	4.4	94%
Sweet dreams	On–off control	3.1	2.1	98.5
Bath time	Event handling	8.65	8.9	56%
Party night	On–off control	3.25	2.35	98.5
Average		5.01	4.32	88.86

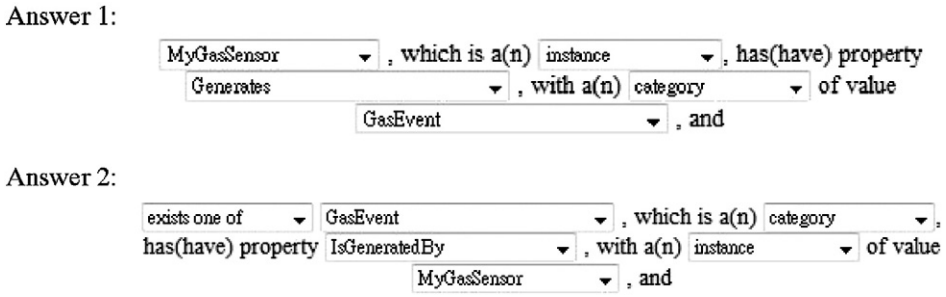

Fig. 19. Two ways to specify the precondition of the "Bath Time" scenario.

factors are supported or not. Many OSGi based systems, such as [24], only allow user to control appliances directly; automation is not supported. Some researches such as [42] only focus on whether and how automated processes can be created by users; semantic process description is not supported. For semantic automation systems such as [1] and [10], processes are usually fixed; users can not create their own processes. In the system designed by [43], user configurable semantic automation process is supported with event handling. Although this system also support the "all" quantifier, this is defined by using an ambiguous "*" symbol, which implies "exist one of" on the sensor side but "all of" on the actuator side. As to the concepts of repeated time such as every day, none of the above systems supports them. Finally, only USHAS supports all these features.

8. Conclusion and future works

8.1. Conclusion

In conclusion, we propose the USHAS system for providing user-configurable semantic home automation. Because different automation processes should be defined to meet the requirements of diverse users and home environments, the USHAS is designed to be user-configurable via Web pages. To solve the problem of interconnectivity between different devices that support distinct communication protocols, this study adopts Web Service technology, which is one of the implementations of the SOA paradigm. Web Service is selected, rather than OSGi, because WS-BPEL has been well defined for automated Web Service execution.

To improve the usability of the proposed system, this study provides the semantic automation process definition and execution, based on the USHAS home ontology. The USHAS ontology contains six first-level classes: Person, Device, Time, Environment, Event, and Location. For semantic process definition, we define the SHPL, which encompasses the definitions of variables, execution time, preconditions, and execution flow. In addition, SHPL supports the capability of expressing the concept of "all" or "none" of something with particular semantic limitations, such as "all the lights in the bed rooms". For semantic process automation, the semantic process runtime module is designed to create WSBPEL processes automatically, based on the current home environment.

Having implemented the USHAS prototype, several scenarios are designed with corresponding examples. Using these scenarios as examples, user satisfaction and system usability are evaluated and the results are satisfactory. Finally, USHAS is compared with other related home automation systems. Among these systems, only USHAS supports all features of automation process execution, event handling, semantic automation, the "all" quantifier, user configurable, and repeated time.

8.2. Future works

As described in Section 6, numerous concerns remain. For future research, we will include an external facial recognition system and attempt to integrate it into USHAS to ensure that the interfaces of USHAS are adequately extensible. To solve the process conflict problem, we will model all semantic processes using Colored Petri Nets, and query the existence of conflict before including each process. If conflict exists after a process is included, then will forgo incorporating this process. Furthermore, only a few smart devices are included in the current USHAS prototype; additional smart appliances, devices, sensors, and scenarios will be included in the future. Moreover, the current user location detection is only simulated in the USHAS prototype; an RFID-based location detection system, such as the prototype designed by Ni et al. [56] will be included. In addition, the user interfaces of both the Semantic Environment Editor and Semantic Process Designer will be improved. In the Semantic Environment Editor, a graphical designer containing a map of the house with drag and drop items will be included. In the Semantic Process Designer, the concept of NLP (Neuro-Linguistic Programming) [57] will be included so that processes can be generated automatically when introducing processes via sentences and their semantic meanings in the context

Table 5
System comparison.

	Automation process execution	Event handling	Semantic automation	The "all" quantifier supported	User configurable	Repeated time supported
System designed by Li et al. [24]	No	No	No	No	No	No
System designed by Wu et al. [1]	Yes	Yes	Yes	No	No	No
System designed by Ha et al. [10]	Yes	No	Yes	No	No	No
System designed by Rodden et al. [42]	Yes	Yes	No	No	Yes	No
System designed by Drey et al. [43]	Yes	Yes	Yes	Yes (by an ambiguous symbol "*")	Yes	No
USHAS	Yes	Yes	Yes	Yes	Yes	Yes

of the current system. Finally, we will address the security concerns and improve the security level of USHAS in the future.

References

[1] Chao-Lin Wu, Chun-Feng Liao, Li-Chen Fu, Service-oriented smart home architecture based on OSGi and mobile agent technology, IEEE Transactions on Systems, Man, and Cybernetics - Part C, Special Issue on Networking, Sensing, and Control 37 (2) (2007).

[2] V. Ricquebourg, D. Menga, D. Durand, et al., The smart home concept: our immediate future, E-Learning in Industrial Electronics, 2006 1ST IEEE International Conference, Dec. 2006, pp. 23–28.

[3] C. Cetina, P. Giner, J. Fons, et al., Using feature models for developing self-configuring smart homes, Fifth International Conference on Autonomic and Autonomous Systems, Apr. 2009, pp. 179–188.

[4] Li Jiang, Da-You Liu, Bo Yang, Smart Home Research, Machine Learning and Cybernetics, 2004, Proceedings of 2004 International Conference, vol.2, Aug. 2004, pp. 659–663.

[5] J. Ryan, Home automation, Electronics & Communication Engineering Journal 1 (4) (July, 1989) 185–192.

[6] Chun-Yuan Chen, Chi-Huang Chiu, Shyan-Ming Yuan, A MOM-based home automation platform, Lecture Notes in Computer Science, Vol. 4413, Springer Berlin, Heidelberg, 2007, pp. 373–384.

[7] Thomas Coopman, Wouter Theetaer, Davy Preuveneers, Yolande Berbers, A user-oriented and context-aware service orchestration framework for dynamic home automation systems, International Symposium on Ambient Intelligence (ISAmI 2010), Portugal, 16–18 June 2010, Ambient Intelligence and Future Trends, Advances in Intelligent and Soft Computing, volume 72, Springer, 2010, pp. 63–70.

[8] Chun-Feng Liao, Ya-Wen Jong, Li-Chen Fu, Toward a message-oriented application model and its middleware support in ubiquitous environments, International Journal of Hybrid Information Technology 1 (3) (July 2008).

[9] Chen Rui, Hou Yi-bin, Huang Zhang-qin, He Jian, Modeling the ambient intelligence application system: concept, software, data, and network, IEEE Transactions on Systems, Man, and Cybernetics Part C: Applications and Reviews 39 (3) (May 2009) 299–314.

[10] Young-Guk Ha, Joo-Chan Sohn, Young-Jo Cho, Hyunsoo Yoon, A robotic service framework supporting automated integration of ubiquitous sensors and devices, Information Sciences: an International Journal 177 (3) (February, 2007) 657–679.

[11] Yung-Wei Kao, Hui-Zhen Gu, Shyan-Ming Yuan, Personal based authentication by face recognition, Proc. of 4th International Conference on Networked Computing and Advanced Information Management (NCM 2008), vol. 2, Sep. 2–4, 2008, pp. 581–585, Gyeongju, South Korea.

[12] Yung-Wei Kao, Hui-Zhen Gu, Shyan-Ming Yuan, Integration of face and hand gesture recognition, Proc. of 3rd 2008 International Conference on Convergence and hybrid Information Technology (ICCIT08), Nov. 11–13, 2008, pp. 330–335, Busan, Korea.

[13] William T. Freeman, Craig D. Weissman, Television control by hand gestures, Proc. of Int'l Workshop on Automatic Face and Gesture Recognition, June 1995, pp. 179–183.

[14] Mike P. Papazoglou, Willem-Jan van den Heuvel, Service-oriented architectures: approaches, technologies and research issues, VLDB Journal 16 (3) (2007) 389–415.

[15] OSGi Alliance, OSGi Service Platform Release 3, Amsterdam, IOS Press, The Netherlands, Dec. 2003.

[16] W3C recommendation: Web services architectureAvailable from:, http://www.w3c.org/TR/ws-arch/.

[17] Diane Jordan, John Evdemon, et al., Web Services Business Process Execution Language Version 2.0," OASIS StandardAvailable at:, http://docs.oasis-open.org/wsbpel/2.0/April 11 2007.

[18] Guus Schreiber, Mike Dean, OWL Web Ontology Language ReferenceAvailable at:, http://www.w3.org/TR/2004/REC-owl-ref-20040210/2004.

[19] David Martin, Mark Burstein, Jerry Hobbs, Ora Lassila, Drew Mc-Dermott, Sheila McIlraith, Srini Narayanan, Massimo Paolucci, Bijan Parsia, Terry Payne, Evren Sirin, Naveen Srinivasan, Katia Sycara, OWL-S: semantic markup for web services, Available at:, W3C Member Submission, November 2004http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/.

[20] X10 Industry StandardAvailable at:, http://www.x10.com/.

[21] INSTEON - Wireless Home Control Solutions for Lighting, Security, HVAC, and A/V SystemsAvailable at:, http://www.insteon.net/.

[22] UPnP Forum, Available at:, http://www.upnp.org/.

[23] Jini Community, Available at:, http://www.jini.org/.

[24] Xie Li, Wenjun Zhang, The design and implementation of home network system using OSGi compliant middleware, Consumer Electronics, IEEE Transactions 50 (2) (May 2004) 528–534.

[25] H. Ishikawa, Y. Ogata, K. Adachi, T. Nakajima, Building smart appliance integration middleware on the OSGi framework, Proc. Seventh IEEE Int. Symp. Object-Oriented Real-Time Distrib. Comput., Vienna, Austria, May 2004, pp. 139–146.

[26] Danny B. Lange, Mitsuru Oshima, Seven good reasons for mobile agents, Communications of the ACM 42 (3) (March 1999) 88–89.

[27] Phong Tran, Paul Greenfield, Ian Gorton, Behavior and performance of message-oriented middleware systems, Proceedings of the 2nd International Conference on Distributed Computing Systems, July 02–05, 2002, pp. 645–654.

[28] Jacques Ferber, Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1999.

[29] Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana, Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/2001/NOTE-wsdl-20010315 2001.

[30] W3C, Soap version 1.2, w3c working draftAvailable at:, http://www.w3.org/TR/2001/WD-soap12-part0-20011217/,2001december 17 2001.

[31] UDDI, The UDDI Technical White PaperAvailable at:, http://www.uddi.org/2000.

[32] Aitor Uribarren, Jorge Parra, J.P. Uribe, Kepa Makibar, Ione Olalde, Nati Herrasti, Service oriented pervasive applications based on interoperable middleware, Proceedings of the 1st International Workshop on Requirements and Solutions for Pervasive Software Infrastructures, Dublin, Ireland, May 2006.

[33] Jürgen Anke, Christian Sell, Jürgen Anke, Seamless integration of distributed OSGi bundles into enterprise processes using BPEL, Proc. of Kommunikation in Verteilten Systemen - 15. ITG/GI-Fachtagung vom 26, in Bern, Schweiz - Universität Bern, February 2, March 2007.

[34] S. Decker, S. Melnik, F. van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, I. Horrocks, The semantic Web: the roles of XML and RDF, IEEE Internet Computing 4 (September/October, 2000) 63–74.

[35] D. Beckett, "RDF/XML Syntax Specification (Revised)", W3C rdf-syntax-grammarAvailable at:, http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/February 2004.

[36] Xiao Hang Wang, Da Qing Zhang, Tao Gu, Hung Keng Pung, Ontology based context modeling and reasoning using OWL, Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, March 14–17, 2004, p. 18.

[37] Richard Fikes, Pat Hayes, Ian Horrocks, OWL-QL — a language for deductive query answering on the semantic Web, Journal of Web Semantics, vol. 2, Elsevier, 2004.

[38] Sonia Ben Mokhtar, Jinshan Liu, Nikolaos Georgantas, Valérie Issarny, QoS-aware dynamic service composition in ambient intelligence environments, Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering (ASE'05), Long Beach, California, USA, November 2005.

[39] Stefan Dietze, Alessio Gugliotta, John Domingue, Situation-driven processes for semantic web services, 3second Annual IEEE International Computer Software and Applications, COMPSAC'08 (2008), 2008, pp. 987–992.

[40] Minsoo Kim, Minkoo Kim, Handling exceptions in situation-driven agent systems, International Conference on Networked Computing and Advanced Information Management, Fifth International Joint Conference on INC, IMS and IDC, , 2009, pp. 870–875.

[41] A. Garcia-Crespo, B. Ruiz-Mezcua, J.L. Lopez-Cuadrado, I. Gonzalez-Carrasco, Semantic model for knowledge representation in E-business, Knowledge Based Systems 24 (2) (2011) 282–296.

[42] Tom Rodden, Andy Crabtree, Terry Hemmings, Boriana Koleva, Jan Humble, Karl-Petter Akesson, Pär Hansson, Configuring the ubiquitous home, Proceedings of the 6th International Conference on Designing Cooperative Systems (Hyères, France, May 11–14, 2004, IOS Press, Amsterdam, 2004, pp. 227–241, COOP'04.

[43] Zoé Drey, Julien Mercadal, Charles Consel, A taxonomy-driven approach to visually prototyping pervasive computing applications, Proceedings of the IFIP TC 2 Working Conference on Domain-Specific Languages, Oxford, UK, July 15–17, 2009.

[44] Patrick Th. Eugster, Pascal, A. Felber, Rachid Guerraoui, Anne-Marie Kermarrec, The many faces of publish/subscribe, ACM Computing Surveys (CSUR) 35 (2) (2003) 114–131.

[45] Lukasz Opyrchal, Atul Prakash, Secure distribution of events in content-based publish subscribe systems, Proc. of the 10th conference on USENIX Security Symposium, August 13–17, 2001, pp. 281–295, Washington, D.C.

[46] Mariano Cilia, Christof Bornhövd, Alejandro P. Buchmann, Event handling for the universal enterprise, Information Technology and Management 6 (1) (January 2005) 123–148.

[47] Harry Chen, Filip Perich, Tim Finin, Anupam Joshi, SOUPA: standard ontology for ubiquitous and pervasive applications, First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04), August 2004, pp. 258–267.

[48] Tadao Murata, Petri nets: properties, analysis and applications, Proceedings of the IEEE 77 (4) (April 1989) 541–580.

[49] Jensen Kurt, Coloured Petri Nets. Basic concepts, analysis methods and practical use, EATCS Monographs on Theoretical Computer Science, Spring-Verlag, Basic Concepts, Vol. 1, 1992.

[50] Jean-Marc Seigneur, Christian Damsgaard Jensen, Stephen Farrell, Elizabeth Gray, Chen Yong, Towards security auto-configuration for smart appliances, Proceedings of the Smart Objects Conference 2003, 2003.

[51] W.M. Fitzgerald, S.N. Foley, Aligning semantic web applications with network access controls, Computer Standards & Interfaces 33 (1) (2011) 24–34 [52] Jonathan M. McCune, Ning Qu, Yanlin Li.

[52] Anupam Datta, Virgil D. Gligor, Adrian Perrig, TrustVisor: efficient TCB reduction and attestation, Proceedings of IEEE Symposium on Security and Privacy (Oakland 2010), 2011.

[53] Ángel García-Crespo, Juan Miguel Gómez-Berbís, Ricardo Colomo-Palacios, Giner Alor-Hernández, SecurOntology: a semantic web access control framework, Computer Standards & Interfaces 33 (1) (2011) 42–49.

[54] Carlos Blanco, Joaquín Lasheras, Eduardo Fernández-Medina, Rafael Valencia-García, José Ambrosio Toval Álvarez, Basis for an integrated security ontology according to a systematic review of existing proposals, Computer Standards & Interfaces 33 (4) (2011) 372–388.

[55] Yi Qian, Yuming Xu, Zheng Wang, Geguang Pu, Huibiao Zhu, Chao Cai, Tool support for BPEL verification in ActiveBPEL engine, Proceedings of The 18th Australian Software Engineering Conference (ASWEC' 07), 2007, pp. 90–100.

[56] Lionel M. Ni, Yunhao Liu, Yiu Cho Lau, Abhishek P. Patil, LANDMARC: indoor location sensing using active RFID, Wireless Networks 10 (2004) 701–710.
[57] Richard Bandler, John Grinder, Reframing: Neuro-Linguistic Programming and the Transformation of Meaning, Real People Press0-911226-25-7, 1981.

Yung-Wei Kao was born on March 12, 1982 in Taipei, Taiwan, Republic of China. He received his MBA degree in the Department of Information Management of National Central University in 2006. His interests include Web Technologies, Ubiquitous Computing, Distributed Objects, and Network Security.

Shyan-Ming Yuan was born on July 11, 1959 in Mauli, Taiwan, Republic of China. He received his BSEE degree from National Taiwan University in 1981, his MS degree in Computer Science from the University of Maryland, Baltimore County in 1985, and his PhD degree in Computer Science from the University of Maryland College Park in 1989. Dr. Yuan joined the Electronics Research and Service Organization, Industrial Technology Research Institute as a Research Member in October 1989. Since September 1990, he has been an Associate Professor at the Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan. He became a professor in June 1995. His current research interests include Distributed Objects, Internet Technologies, and Software System Integration. Dr. Yuan is a member of ACM and IEEE.